Programozási Technológia 3. beadandó 3. feladat: Labirintus

Készítette: Neszlényi Kálmán Balázs

Neptun kód: DPU51T

Email cím: dpu51t@inf.elte.hu

Tartalom

A feladat leírása

3. Labirintus (Labyrinth)

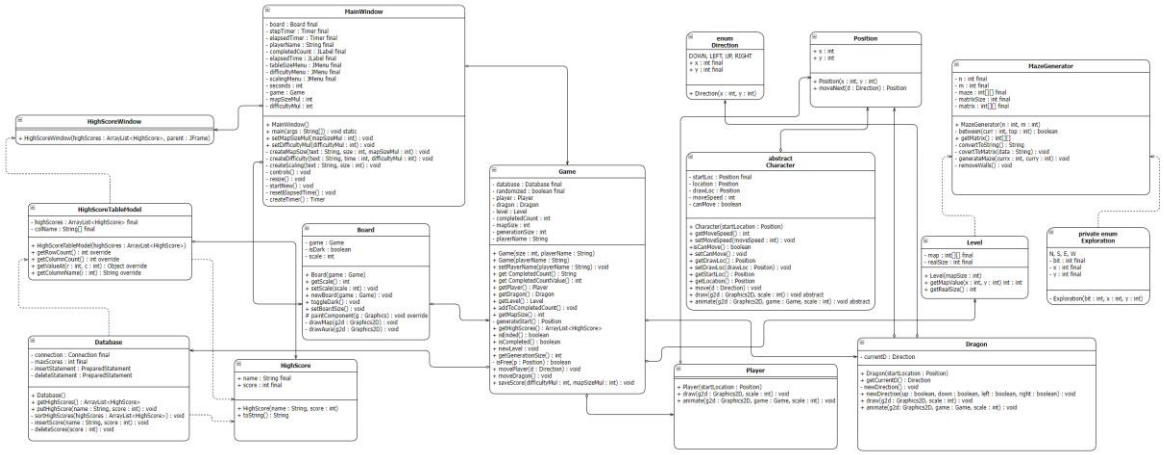
Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a labirintus sárkányát.
Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik irányba. Ha a sárkány a játékossal szomszédos területre jut, akkor a játékos meghal. Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a labirintus felépítését, távolabb nem. Tartsuk számon, hogy a játékos mennyi labirintuson keresztül jutott túl és amennyiben elveszti az életét, mentsük el az adatbázisba az eredményét. Egy menüpontban legyen lehetőségünk a 10 legjobb eredménnyel rendelkező játékost megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot indítani egy másik menüből. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon kezdjenek.

Elemzés

- A játék induláskor megkérdezi a játékos nevét, ez akár később módosítható is.
- A játékot négyféle pályamérettel játszhatjuk: kicsi (13 X 13-as pálya), közepes (17 X 17-es pálya), nagy (21 X 21-es pálya), illetve szintenként változó méretű pálya. A program indításkor közepes pályaméretet állít be és automatikusan új játékot indít.
- A pályaméret mellett 4 nehézségi szintből választhatunk (könnyű, közepes, nehéz, lehetetlen), ezek a sárkány sebességét állítják.
- A feladatot Java Swing grafikus felülettel valósítjuk meg, az ablak nagyítása a beállításokban állítható.
- A JFrame ablakban elhelyezünk egy menüt a következő menüpontokkal: Játék (Új játék, Új Játékos, Kilépés, Dicsőségtábla), Beállítások (Pályaméret, Nehézség, Nagyítás). Az ablak tetején megjelenítünk egy státuszsort, amely a teljesített pályák számát és az eltelt időt mutatja.

- A játéktáblát egy JPanel reprezentálja, melyre kirajzoljuk a labirintus mezőit, falait, a játékost és a sárkányt. A játékost egy kék, a sárkányt egy piros kör reprezentálja.
- A játékost a billentyűzet nyilai segítségével mozgathatjuk.
- A játék automatikusan feldob egy dialógusablakot, amin megjelenik a játékos pontszáma, amikor vége a játéknak (a sárkány egy a játékossal szomszédos mezőre lépett). Ezután automatikusan új játék indul.
- A játék tartalmaz egy fejlesztői menüt is, ahol felfedhetjük a pályát a labirintusok megtekintéséhez, illetve folyamatosan új pályákat generálhatunk ezzel ellenőrizve a labirintusgenerálás működését.

Az osztályokat és kapcsolataikat bemutató osztálydiagram



A program szerkezete

A játékot háromrétegű architektúrával (model-view-persistence) valósítjuk meg. A model intézi az üzleti logikát, a view a megjelenést tartalmazza a felhasználó felé, míg a perzisztencia osztály az adatok tartós tárolásáért felel. Az adattárolást a labyrinth nevű MySql adatbázis scores táblájának segítségével valósítjuk meg. Az architektúra segítségével egyszerűen cserélhető a program mindhárom rétege.

A megoldáshoz szükséges típusok rövid osztályleírása

Character

A Character absztrakt osztály a játékos és a sárkány ősosztálya, a karakterek pozícióiért és mozgatásukért, kirajzolásukért felel.

Player

A Character osztály leszármazottja, implementálja a játékos kirajzolását és megvalósítja a játékos mozgásának animálását.

Dragon

A Character osztály leszármazottja, implementálja a sárkány kirajzolását és megvalósítja a sárkány mozgásának animálását. Továbbá implementálja a sárkány irányváltásának metódusait.

Game

A Game osztály implementálja a játék szabályait, többek között felel a szabályos lépésekért, a játék végének a vizsgálásáért, a pontszám kiszámolásáért, a labirintusok léptetéséért és a sárkány kezdőpozíciójának kiválasztásáért.

Level

A Level osztály a játék egy pályáját valósítja meg, egy labirintus és egy pályaméret alkotja.

MazeGenerator

A labirintusok generálásáért felel, a rekurzív osztás algoritmusával az utakat bitekkel leírva alkot egy labirintust, melyet ezután hagyományosabb formára hoz.

Direction

Enum osztály, a mátrixban a mozgás lehetséges irányait tartalmazza a könnyebb karaktermozgatás segítéséhez.

Position

Segédosztály, a karakterek koordinátáinak tárolásához.

Database

A MySql adatbázisunkkal tartja a kapcsolatot, JDBC kapcsolat segítségével lekérdezéseket, beszúrásokat és frissítéseket futtat le a táblán.

HighScore

Segédosztály a játékosnév, pontszám rekordpárok tárolásához.

Board

A játéktábla implementációja, összefogja a játék grafikus részeinek kirajzolását.

MainWindow

A játék ablaka, eseménykezelőket köt a játék megfelelő részeihez.

Implementáció

Labirintusgenerálás

- A rekurzív osztásmódszer algoritmusával működik. Nevezzük kamrának azokat a helyeket, ahol még nincsenek falak. Az algoritmus egy kamrában kezd. Ezt a kamrát véletlenszerűen elhelyezett falakkal osztja fel és minden fal tartalmaz egy véletlenszerűen elhelyezett átjárót. Az algoritmus rekurzívan ismétlődik az összes alkamrán, amíg el nem fogynak. Az implementációkban az átjárókat bit-ekként tartalmazza. Észak 1 bit, Dél 2 bit, Kelet 4 bit, Nyugat 8 bit. Kamránkként mindig kevert sorrendben indulunk el a négy irányban. Eltároljuk az aktuális és az új koordinátákat a bejárás során. A between segédfüggvény ellenőrzi, hogy a pályán belül maradjon, illetve csak az üres kamrákon hívhatjuk rekurzívan az algoritmust. Üres kamra esetén az aktuális bit értéket az aktuális irány bit értékével módosítjuk a bitwise vagy operátor segítségével. Ezzel párhuzamosan az új koordinátákat is ezzel a módszerrel változtatjuk, de ott az irány ellentétét használjuk.
- Az algoritmus lefutása után egy tökéletes labirintust generálunk. A játékosnak több útvonalon is ki kell tudni jutnia, ezért néhány falat ki kell vennünk a kész labirintusból.
- A labirintust először játszható formára hozzuk. A biteket értéküknek megfelelően egy beolvasható 1-esekből és 0-ákból álló mátrixxá alakítjuk a bitwise és operátor használatával, majd minden hét hosszúságú egységes falból eltávolítunk egy-egy blokkot új átjárókat képezve. Ha valahol vastagabb út keletkezne, akkor azt kipótoljuk. A kész mátrixot átadjuk a játéknak.

A pálya sötétítése és a játékos látóterének beállítása

- Amennyiben nem vagyunk fejlesztői módban, a Graphics2D g2d osztály segítségével egy a pálya területét lefedő méretű területet veszünk fel.
- A játékos rajzoltpozíciója köré egy 3 mező sugarú kört rajzolunk.

- A kört kivonjuk a területből.
- Kitöltjük a területet.

A sárkány irányválasztása

- A sárkány newDirection metódusának átadjuk a sárkány környező mezőinek állapotát, a mezők igazak, a falak hamisak.
- Az available nevű ArrayListben eltároljuk azokat az irányokat, amelyek nem falak.
- Az available ArrayListből tetszőlegesen választunk egy irányt, ami a sárkány új iránya lesz.

Eseménykezelő-esemény párok és a hozzájuk tartozó tevékenységek

Eseménykezelő	**Esemény**	**Tevékenység**
newGame **ActionListener(actionPerformed)**	A Játék menüben az Új Játék elemre kattintás.	Új játék indítása.
newPlayer **ActionListener(actionPerformed)**	A Játék menüben az Új Név elemre kattintás.	A játékos átnevezése.
exit **ActionListener(actionPerformed)**	A játék menüben a kilépése elemre kattintás.	Kilépése a játékból.
menuHighScores **AbstractAction(actionPerformed)**	A Játék menüben a Dicsőség tábla elemre kattintás.	Egy JTable-ben megjeleníti a legjobb 10 pontszámot.
mapSize **ActionListener(actionPerformed)**	A Beállítások menüben a Pályaméret opció elemeire kattintás.	Új játék indítása az adott opció pályaméretével.
difficulty **ActionListener(actionPerformed)**	A Beállítások menüben a Nehézség opció elemeire kattintás.	Új játék indítása, a sárkány sebessége az opció alapján változik.
scalingVal **ActionListener(actionPerformed)**	A Beállítások menüben a Nagyítás opció elemeire kattintás.	Az ablak nagyítása az opciónak megfelelően változik.
dark **ActionListener(actionPerformed)**	A Fejlesztés menüben a Sötétség opcióra kattintás kattintás.	Felfedi vagy újra elfedi a pályát.
loadTest **ActionListener(actionPerformed)**	A Fejlesztés menüben a LoadTest opcióra kattintás kattintás.	Folyamatosan új pályát generál.
keyPressed **KeyListener(KeyEvent)**	Egy nyíl gomb lenyomása vagy nyomva tartása a billentyűzeten.	A játékos a nyílnak megfelelő irányba lép, ha nincs abban az irányban akadály.

Tesztelési terv

1. A játékfelület megjelenik.
2. A pályaméret váltás sikeres minden méretre.
3. A sárkány sebessége helyesen változik.
4. A játékos mozgatható.
5. A karakterek nem a falon kezdenek.
6. A játék véget ér, ha a sárkány a játékos mellé lép.
7. A cél elérésekor új pálya kezdődik.
8. A dicsőségtábla helyesen listázza a legjobb 10 pontszámmal rendelkező játékost.
9. A labirintusokból mindig ki lehet jutni.
10. A labirintusból nem csak egy útvonalon lehet kijutni.