Pintér Tamás
JY4D5L
2020. November 26.

Snake Dokumentáció

Tervezési megfontolások

A program megírását úgy kezdtem, hogy létrehoztam egy ablakot, amivel a tervem az volt, hogy a program állapotától függően rajzolok rá dolgokat.

Ahhoz, hogy a játéknak állapotai lehessenek, létrehoztam egy `GameStatus` enumerációt, ami a `MENU, GAME, NEWGAME, LEADERBOARD` és `ENDGAME` értékeket veheti fel. Mivel ezeket az értékeket más osztályokból is szerettem volna állítgatni, ezért létrehoztam egy `GameStatusListener` nevű interface-t, amiket implementáltam minden olyan osztályban, ahol a játék állapotát szeretném megváltoztatni. Ugyan így jártam el a nehézségi szint változtatása esetében a `Difficulty` enumerációval és a `DifficultyListener`-rel.

Ahhoz, hogy a játék állapotaitól függően változzanak a képernyőre rajzolt dolgok, egy állapotgépet hoztam létre, amit egy `AnimationTimer`-be tettem.

Mivel a megoldásom folyamatosan újra rajzolja a képernyőt, ezért szükség volt egy függvényre ami azért felelős, hogy a képernyőről letöröljön mindent és újra lehessen rá rajzolni. Ez a `Screen` osztály `resetScreen` nevű függvénye lett.

Azért nem fxml-t használtam a képernyők megtervezéséhez, mert nem volt sok megjelenítendő tartalom a képernyőn, és gyorsabb volt kézzel összerakni a gombokat. Mivel gyakran jelennek meg ismétlődő stílusú dolgok a képernyőn, létrehoztam egy `style.css` fájlt, amiben definiáltam a gombok és feliratok stílusát. A gombok stílusát az interneten találtam meg, ezen a honlapon:
`http://fxexperience.com/2011/12/styling-fx-buttons-with-css/`

A kígyó és az ételek megjelenítéséhez létrehoztam egy `DrawableUnit` nevű osztályt, aminek annyi a lényege, hogy van egy X és Y pozíciója, és egy fix mérete. A kígyó mozgását úgy oldottam meg, hogy van egy függvény ami azt figyeli, hogy a felhasználó melyik gombot nyomta meg és ennek megfelelően ad értéket egy `Direction` enumerációnak és egy másik, ami az aktuális irányba elmozdítja a kígyót. Amikor a kígyó megeszik egy ételt akkor hozzáadok egy új `DrawableUnit`-ot, aminek a koordinátája a (0,0). Ez azért van, mert a mozgás függvény úgyis át fogja mozdítani abban a lefutásban a kígyó "farkának" a végéhez. Ugyan ekkor az ételt átmozgatom a pályán kívülre. Ez azért van, mert az ételek újra rajzolását a következő éppen oldottam meg: Minden lefutásnál meghívódik a `generateFood` függvény, ami elhelyez a pályán annyi ételt, amennyi az adott nehézségi szinthez szükséges, feltéve hogy még nincsen elég a pályán. Ez után meghívódik minden étel `update` függvénye, ami megnézi hogy a pályán kívül van-e az étel, vagy ha teleportálós ételekről van szó, akkor lejárt-e az ideje. Ha ezek közül valamelyik teljesül, akkor meghívódik a `relocateFood` függvény, ami új helyre teszi az adott ételt és a nehézségtől függően állítja be a típusát.

A pontszámok kezeléséhez létrehoztam egy `SingleScore` nevű osztályt, ami eltárol egy pontszámot, és egy nehézséget, valamint csináltam egy `Scores` osztályt is, amiben egy `ArrayList` segítségével tárolok el `SingleScore`-okat. Ezeket az osztályokat szerializálhatóvá tettem, hogy könnyen elmenthető legyen a ranglista.

A ranglistát úgy kezelem, hogy a játék végeztével megnézi egy függvény, hogy az aktuális pontszám nagyobb-e az eddigi ranglista legrosszabb pontjánál. Ha igen, akkor felveszi ide a pontot a hozzá tartozó nehézséggel, amihez csináltam egy `ScoreComparator`-t. Ez után sorba rendezem megint a pontszámokat, és elmentem a `leaderboard.ser` fájlba a `Leaderboard` osztály `scores` adattagját. Ahhoz hogy ne mentse el a program folyamatosan és az első mentéstől feleslegesen a pontokat, bevezettem egy `savedToFile` `boolean`-t, amit az első mentés után `true`-ra állítok, majd az új játék kezdésekor `false`-ra. A fájlok betöltésénél figyeltem arra, hogy ha a `leaderboard.ser` fájl nem létezik, akkor hozzon létre egy újat.