Témalabor

Fejlesztői dokumentáció

Varga János

BJCZQ9

2021. december 16.

Pacman

Tartalomjegyzék

Feladat

Egy olyan weboldal elkészítése, amely rendelkezik bejelentkezési opcióval, eltárolja a regisztrált felhasználók adatait, illetve lehet az oldalon játszani a klasszikus Pacman játékkal. A játékot értelemszerűen a fel, le, jobbra, balra nyilakkal lehet irányítani és a játék végén a score-t elmenti egy highscore listába. Ezeket a score-okat egy adatbázisba menti ki kilépéskor, belépéskor onnan tölti be őket.

FrontEnd

A weboldal Frontend-je React és javascript technológiákat alkalmazva készült. Ez egy single page application, azaz van egy App.js fájl, ami az alap oldal és ennek a tartalmába a felhasználói kattintás hatására az adott oldal töltődik be. Ehhez van egy navigation bar az oldal tetején, amely az adott oldal linkjét tartalmazza. Minden egyes külön oldal egy Javascript osztály, amely a React Component osztályából származik le. Minden ilyen osztálynak van egy render függvénye, amely konkrétan a html kirenderelés végzi.

App.js

Ez az alap oldal, amely feladat a navigation bar megjelenítése, ez React szinten egy Router element, amely tartalmazza az adott route-okhoz tartozó linket. Az osztálynak van egy változója, amely azt tárolja, hogy a felhasználó be van-e jelentkezve. Ennek értéke üres, ha nincs és a felhasználó email címe, ha igen. Ha az értéke nem üres, akkor a navbar-t és a főoldalt jelenítjük meg tartalomként alapesetben, különben pedig a bejelentkezési felületet.

MainPage.js

Tartalma csak egy leírás az oldalról és a játék működéséről.

Navigation.js

Ez a fájl definiálja a felső navbar felépítését, ul, li tagekkel és a linkekkel együtt. Ezt minden esetben kirendereljük, amikor a felhasználó be van jelentkezve.

Profile.js

Az oldal betöltődéskor lekéri az összes felhasználó listáját a szerverről rest api-s hívással, majd megjelenít egy formot, kiszedi belőle a mi adatainkat az e-mail címünk alapján, ezeket a formba írj, ahol tudjuk őket módosítani. Ezt egy put rest api-s kéréssel teszi, elküldve a szerver adott url-jére.

Highscore.js

Betöltődéskor lekéri az összes score-t a szerverről szintén egy rest api-s hívással, majd kirenderel egy táblázatot az adatinkal.

Registration.js

Betöltődéskor betölti a meglévő user-ek listáját egy szerverre küldött kéréssel, Majd megjelenít 2db formot egymás mellet. A baloldali egy regisztrációs form. Ha a felhasználó kitölti és rámegy a Regisztráció gombra, akkor leellenőrizzük a userek listájából, hogy nem-e regisztráltak már ezzel az email címmel. Ha igen, akkor hibaüzenetet írunk, ha nem akkor pedig post utasítással elküldjük a szervernek. Jobb oldalon pedig egy bejelentkezési oldal található, amelyre az email címet és a jelszót

várja a program. Ha jó kombinációt adott meg a felhasználó, azaz a listában is így szerepelnek az adatok, akkor átengedi, különben pedig nem.

Game.js

Ez az osztály felel a játék konkrét működéséért. Betöltődéskor inicializál egy game változót, illetve elkezdi annak a működtetését.

BackEnd

Adatbázis

Ez egy egyszerű MongoDB, amihez a Nodejs kapcsolódik. A lokális gépen található az adatbázis, és adott kérésre innen szedi le az adatokat.

Szerver része

NodeJS technológiát használva működik a szerver rész. Tartalmazza az adatbázis modellek leírását (felhasználók és pontszámok), illetve későbbi bővítéshez a pálya és a szörnyek modelljeit is. Amikor elindul, akkor létrehozza a kapcsolatot az adatbázissal, illetve ha nem léteznek ezek a modellek kollekciók, akkor létrehozza őket.
A szerver fő feladat a rest api-s kérésekre való válaszolás.
Első fontos kérés, az a „/listScores” url-re küldött GET kérés, erre lekéri az összes score-t az adatbázisból és kiteszi a json objektumokat erre az endpointra.
A „/listUsers” url-re küldött GET kérésre kiteszi az endpointra az összes felhasználó adatát json formátumban.
A „/regUser” url-re küldött POST kérés hatására a kérés body-jában lévő adatokat kiszedi és létrehoz egy új felhasználói objektumot az adatbázisban.
A „/updateUser” url-re küldött PUT kérés hatására pedig frissíti az adott meglévő objektum adatait.
A szerver definiálja ezeket az alap funkciókat külön fájlokban is függvényekként későbbi felhasználás érdekében.

Game functions

Game class

Ő felelős a játék inicializálásához, illetve a pálya, a pelletek, pacman és a szörnyek létrehozásáért is.
Tárolja a map-ot egy 2D char tömbben, amiben az üres vagy pont karakter egy léphető Field-et, más karakter pedig egy nem léphető Field-et (falat) reprezentál. Ebből készít egy mátrixot, amelyben már a konkrét Field objektumok lesznek az elemek, az alapján, hogy mező vagy fal.
Ezután feltölti pellet-ekkel a mezőket, random módon, 10% az esélye, hogy egy mezőre rak pellet-et vagy nem.
A játék kezdetén elhelyezi Pacman a pálya közepére, a szörnyeket pedig a pálya 4 sarkába, ahonnan egyből elindulnak.
Ő felelős még a score és a a játék vége kiíratásáért, illetve figyeli mikor van vége a játéknak.
A kiíratások mind egy-egy div-ben lévő h1, amelynek az értékét változtatom egy külön függvényben, amely egyúttal figyeli is a pellet-ek számát.
Alapvetően kettő esetben lehet vége a játéknak:

- Ha elfogy Pacman 3 élete, akkor a játékos veszít, és kiírja a „You Lost!” feliratot, ekkor megáll a játék, a nyilakra tett actionListener nem működik tovább.

- Ha Pacman felveszi az összes pellet-et, akkor kiíródik a „You won!” felirat, ekkor megáll a játék, a nyilakra tett actionListener nem működik tovább.

Field class

Ez az osztály reprezentál egy mezőt a játékban. Mivel csak kettő fajta mező van a játékban, ezért nem származtattam le belőle azt a 2 fajtát, egyszerűen csak egy type adattag különbözteti meg a mezőt a faltól.
Ez ugyanúgy egy DIV, aminek szélessége és magassága előre meghatározott, 30x30 pixel. Két függvénye van, amelyek megkeresik és visszaadják az x,y koordinátája alapján, hogy a pálya mátrixban mi a pontos sor és oszlop indexük.

Thing class

Alapvetően egy absztrakt osztály lenne, viszont ez javascriptben annyira nem megoldható, ezért ez egy általános osztály, amely azon objektumokat reprezentálja, amik a játék fontos részei, valamilyen ütközésre képesek. Ezeket el lehet helyezni az üres mezőkön, ezért tárolják az aktuális Field objektumukat. Ezek ugyanúgy div-ek, méretük hasonlóan 30x30-as, viszont ezeknek a háttere egy kép, amelyek a /images mappában találhatóak.
Ezeket az objektumokat lehet még mozgatni is, ehhez egy olyan tagfüggvénye van, amely a pálya mátrixon belül tudja mozgatni adott indexekre.
Illetve van egy olyan mozgató függvénye, amely a szörnyeket és a pacmant mozgatja a bemenet alapján. Először megnézi, hogy amerre szeretne mozogni az objektum, az a pálya mátrixban micsoda.

- Ha üres mező, akkor odalépteti.
- Ha fal, akkor csak egyszerűen visszatér a függvény.

A bizonyos Thing-ekkel való ütközés függvények fejlécei is itt vannak deklarálva, de maga a függvények törzse csak a szükséges leszármazottakban van definiálva.

Pellet

Ez az osztály a Thing leszármazottja és alapvetően a klasszikus játékból ismert pellet-eket reprezentálja. Ezeket Pacman fel tudja venni, ilyenkor neki növekedik a score counter-je egyel, a pellet pedig eltűnik a pályáról.
A háttérképe a /*images*/pellet.png fájl.

PowerPellet

Ez az osztály a Thing leszármazottja és alapvetően a klasszikus játékból ismert powerpellet-eket reprezentálja. Ezeket Pacman fel tudja venni, ilyenkor nem növekedik a score counter-je, viszont ekkor a szörnyek átváltoznak kék színűre, majd 10mp-ig Pacman meg tudja őket enni. Ha ez sikerül akkor a szörnyek eltűnnek és Pacman kap ezért szörnyenként 20 pontot. Ezek mindig a bal felső sarokban jelennek meg az egyszerűség kedvéért a pelletek számával arányosan, azaz ha a pelletek darabszáma modulo 20 az egyenlő 0, akkor megjelenik egy.
A háttérképe a /*images*/powerpellet.png fájl.

Pacman

Ez az osztály a Thing leszármazottja és alapvetően a klasszikus játékból ismert Pacmant-t reprezentálja. Kezdetben 3 élete van, ez egy adattagja az osztálynak.
Ütközhet több dologgal is(, az ütközés azt jelenti, hogy a Field-jük megegyezik):

1. Pellet-el ütközik: Felveszi a pelletet, megnöveli a score értékét 1-el, és a Field-ről eltűnik a pellet.

2. PowerPellet-el ütközik: Felveszi a powerpellet-et, a szörnyek kékké változnak 10 mp-re, azaz az állapotuk „frightened” lesz. Ezt egy setTimeout-al kezelem, amely 10000ms után visszaállítja az állapotukat normálra.
3. Monster-rel: Ha még van élete, akkor abból levonok egyet és pacman középre kerül a kezdeti pozíciójába. Ha már csak egy élete van, akkor vége a játéknak és kiírja, hogy „You Lost!”.

A háttérképe függ az aktuális iránytól, amely felé megy: a *images*/pac_[irány].gif fájl, ahol az [irány] rész az a right, left, up vagy down.

Monster

Ez az osztály a Thing leszármazottja és alapvetően a klasszikus játékból ismert Monster-t reprezentálja.
A mozgásukat kezdetben az eredeti Pacman játékhoz hasonlóan úgy oldottam meg, hogy elsősorban megnézem, hogy a 4 léphető irány közül melyik léphető, azaz melyik üres. Utána a meglévő irányok közül pedig egy egyszerű Pitagorasz segítségével azt választom, amelyik a legkisebb távolságra van.
4 fajta szörny létezik, Inky, Pinky, Blinky és Clyde. Őket a programban csak a színük különbözteti meg, ugyanúgy mozognak.
Alapból 2 állapotuk lehet:

- A normál állapot az a sima, ahol megpróbálják elkapni pacmant.
- A frightened állapotban pont ellenkezőleg, megpróbálnak elfutni Pacman elől, mert ilyenkor ő meg tudja őket enni.

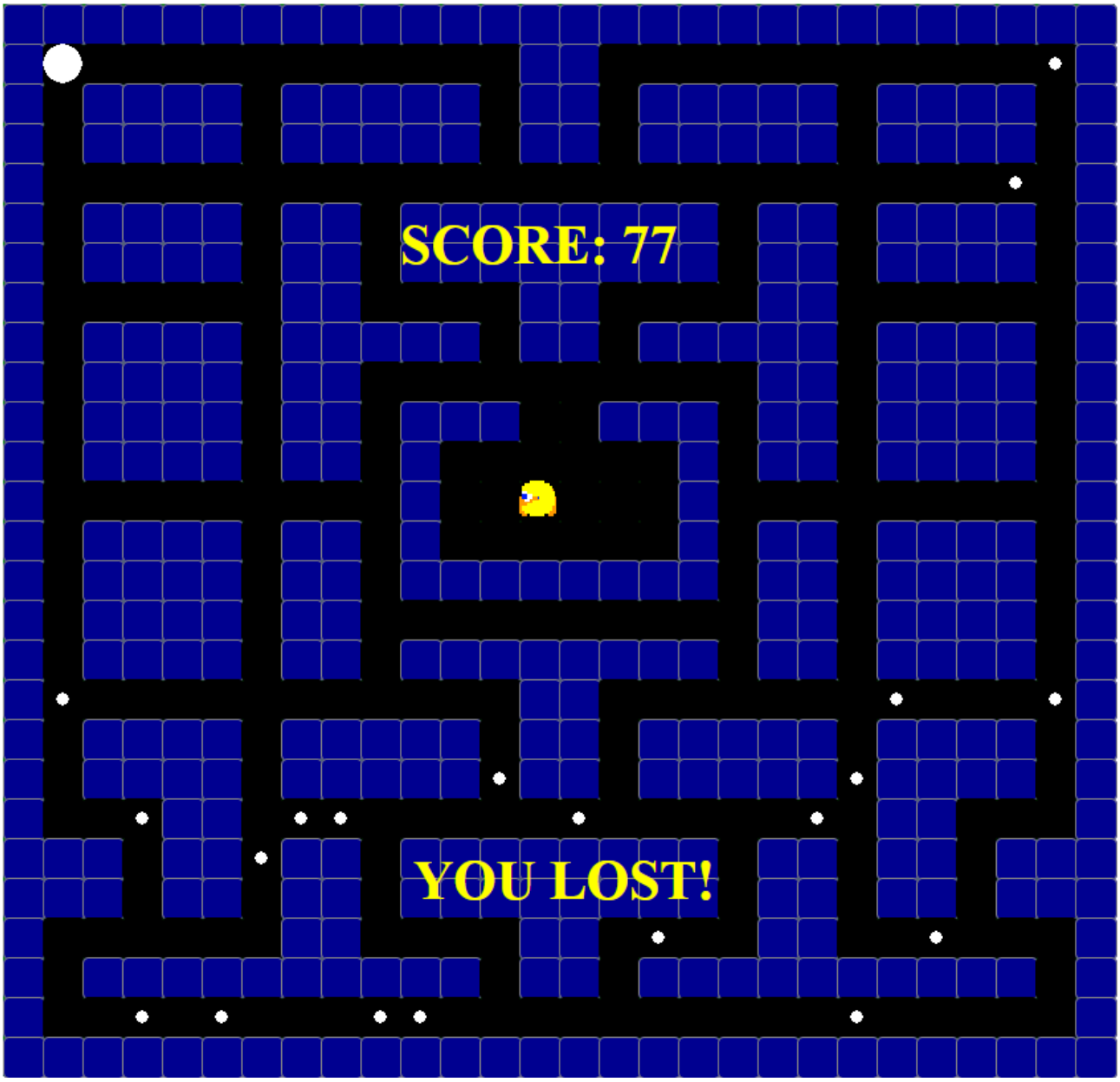
Viszont jelenleg úgy működik, hogy a mátrixban elfoglalt helye szerint megkeresi a legrövidebb utat egy útkereső algoritmussal és az alapján dönt a legjobb irányról. Ez egy BFS-el kiszámolja minden lépés utána legrövidebb utat a mátrixban, lép egyet majd újra az egészet és azért minden lépés után mert Pacman, folyamatosan mozoghat.

A háttérképe függ az aktuális iránytól, amely felé megy: a *images*/[type]_[irány].gif fájl, ahol az [irány] rész az a right, left, up vagy down. A [type] értéke lehet c,b,p avyg i az alapján, hogy milyen szörnyről van szó.

A billentyűlenyomásokat egy egyszerű eventListener figyeli és a lenyomott nyilak állapota alapján hívja meg a pacman move függvényét, illetve az ütközés függvényeket.

A szörnyeket pedig egy setInterval segítségével mozgatom, amely 400ms-onként hívja meg minden Monster move függvényét egymás után.

Amikor vége egy játéknak akkor pedig az eredmény hozzáadódik a highscore listához és beíródik az adatbázisba is.


SCORE: 77
YOU LOST!