Energetikai Technikum és Kollégium

Arkanoid klón

2023.

|  |  |
| --- | --- |
| Konzulens: | Készítette: |
|  | Molnár Csaba |

Nyilatkozat

Alulírott Molnár Csaba büntetőjogi felelősségem teljes tudatában nyilatkozom arról, hogy az itt szereplő záródolgozat saját, önálló munkám eredménye és sem részeiben sem egészében ne került még kereskedelmi forgalomba, ill. publikálásra, a GPL licenszelésű programrészek kivételével.

Paks, 2023. április 2.

...............................................…

Molnár Csaba

**Tartalom**

[I. Bevezető 4](#_Toc1)

[II. Témaválasztás indoklása 4](#_Toc2)

[III. A fejlesztői dokumentáció 5](#_Toc3)

[1. Az alkalmazott fejlesztői eszközök 5](#_Toc4)

[2. Adatmodell leírása 6](#_Toc5)

[3. Részletes feladatspecifikáció, algoritmusok 6](#_Toc6)

[4. Forráskód 7](#_Toc7)

[5. Tesztelési dokumentáció 7](#_Toc8)

[IV. A felhasználói dokumentáció 7](#_Toc9)

[6. A program általános specifikációja 7](#_Toc10)

[7. Rendszerkövetelmények 7](#_Toc11)

[8. A program telepítése 7](#_Toc12)

[9. A program használatának a részletes leírása 7](#_Toc13)

[V. Összegzés 7](#_Toc14)

[VI. Irodalomjegyzék, forrásmegjelölés 7](#_Toc15)

# Bevezető

A záródolgozatom témája egy videojáték, amely az Arkanoid című klasszikus játéktól vesz erős inspirációt, a saját design ötleteimmel felfrissítve.

Feladatomat interaktív weboldal képében valósítom meg, ennek érdekében pedig természetesen HTML, CSS, illetve JavaScript nyelveket fogok alkalmazni. Annak az oka, hogy nem C# illetve Python nyelvet használok, hogy bár objektum orientált megoldást szeretnék kivitelezni, a Windows Forms nem ad elegendő rugalmasságot a felhasználói interfész kidolgozására, valamint személyes véleményem szerint egyik nyelv sem megfelelő videojáték fejlesztésre.

A CSS kellően sokszínű és könnyen használható, a JavaScript pedig rendelkezik azokkal az objektum orientált eszközökkel, amelyek megkönnyítik a feladat elvégzését.

# Témaválasztás indoklása

Bár a korombéli fiatalok a kétezres évek végét, tízes évek elejét a legújabb Xbox 360 illetve Playstation 3 konzolokkal töltötték, az én családomnak egy nyolcvanas évek béli Nintendo rendszer jutott. Ennek következménye, hogy leginkább a már akkor retrónak számított játékvilággal ismerkedtem meg, és ez a periódus maradt az érdeklődési köröm központjában évekkel később is.

A záródolgozat feladatkiosztásakor azonnal tudtam, hogy valamiféle videojáték programot szeretnék csinálni. Ennek több oka is van.

Először is, a segédprogramok illetve a webshopok világán belül kevés ötletem volt, hiszen az ilyesfajta problémákra aligha tudnék egyedi és kitűnő megoldást találni. Továbbá az is fontos, hogy a hónapokon át tartó fejlesztési eljárást ne unjam meg, hiszen, ha elmegy a kedvem a szoftver kidolgozásától, akkor azzal a szakdolgozatom is bukik.

Ezokból célratörő lenne a játékfejlesztés, hiszen a debugolás és az új funkciók hozzá adása alatt állandóan játszanom is kellene, ami egyben feltárja a program hibáit, és szórakoztat is.

A téma tehát megvan. Már csak azt kell leszögezni, milyen stílusú játékot szeretnék fejleszteni.

Az első gondolatom egy bullet-hell shoot-‘em-up akciójáték volt, hiszen a feladat kiosztása idején ezzel a műfajjal foglalkoztam. Ez sajnos problémát jelentett, hiszen ilyen téren semmiféle tapasztalatom nincs, és rengeteg rendszert el kellett volna sajátítsak rövid időn belül ahhoz, hogy ez megvalósuljon. Gondolkodtam még szerepjátékon illetve platformeren, viszont végül a puzzle műfajon landoltam.

Ezen belül két nyilvánvaló lehetőségem volt. Egy Tetris inspirált véget nem érő kirakós, ami egyre nehezebb lesz az idő teltével, vagy egy Arkanoid inspirált, rövidebb terjedelmű de sokkal több lehetőséggel bíró projekt, amelybe bele fűzhetem a saját ötleteimet. Mivel egy osztálytársam a Tetris mellett döntött, elhatároztam, hogy a második opcióval élek.

Ahhoz, hogy minden, amit 5 év alatt elsajátítottam megjelenjen a programomban, leszögeztem pár alap funkciót, amit implementálok:

* Alapvető Arkanoid játékmenet, mely egyetlen HTML oldalt foglal el.
* CSS használatával könnyen megformálható felhasználói interfész, pár animációval.
* Pontrendszer és ranglista, amelyet SQL adatbázissal raktározok el.
* Power-up rendszer, aminek implementálására objektum orientált programozási elveket és eszközöket használok.
* Több játékmód és életrendszer, amelyet a JavaScript kihasználásával valósítok meg.
* Saját magam által létre hozott hangeffektek, grafikai elemek.

Természetesen a fejlesztés során kis mértékben változhatnak az elképzeléseim, de az alap pontokat szorosan követni fogom.

Úgy vélem, hogy a választott téma, és annak kivitelezése elegendően demonstrálja a szoftverfejlesztő képességeimet, illetve a problémák elegáns megoldásához való szakértelmemet.

# A fejlesztői dokumentáció

## Az alkalmazott fejlesztői eszközök

Fejlesztői környezetnek Visual Studio Code-ot használtam mivel egy webapp fejlesztésére tökéletes eszközökkel rendelkezik. Támogatja az összes nyelvet, amellyel megvalósítom az ötletem, beépített Git támogatásának köszönhetően könnyű rendszerezni a kódom, valamint bővítmények használatával az SQL szerverrel is egyszerűen tudtam kommunikálni.

A debugolás Firefox és Chrome webböngészőkben történt, a beépített fejlesztői eszközöket használva. Ezekkel tudtam a CSS stíluslap hibáit kiküszöbölni, valamint a változókat nyomon követni, ami a fizika kidolgozásához segített.

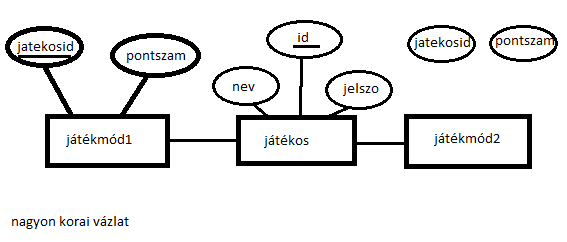
Mivel PHP és SQL is szerepel a projektemben, kellett egy webszerver ami minél több platformon elfut és felhasználóbarát telepítési procedúrával rendelkezik. Ezért választottam az XAMPP webszerver csomagot, amely önmagába foglal többek között egy Apache szervert, amin a PHP kód is elfut, és egy MySQL adatbázis kezelő programot.

Ezen kívül az adatbázis kezelését megkönnyítette a phpMyAdmin grafikus interfész, amely szintén az XAMPP csomag tartozéka. Ezt használva tudtam figyelem alatt tartani a játékékosok adatait, a játék közben szerzett pontszámokat, és a különböző lekérések érvényességét.

Külső modulokat nem használtam.

## Adatmodell leírása

A játékban szereplő adatbázis az entitás-kapcsolat modell alapján működik.



3 fő tábla létezik az adatbázison belül.

Az első tábla a játékosok adatait tartja számon. Ezek az adatok a játékosok nevei (nev), jelszavai (jelszo), valamint az azonosító számok (id). A játékos nevet és a jelszavat a program induláskor egy felugró ablakban kéri el a felhasználótól. Amennyiben a megadott adatok alapján nem létezik felhasználó, a program létrehoz egy azonosító számot és hozzá adja az adatokat a táblához. Ha a felhasználónév létezik, de a jelszó helytelen, hibaüzenet fogadja a felhasználót. Ha mindkét adat létezik, a program megkeresi az azonosító alapján a felhasználó elért pontszámait. A játék során elért pontszám felülírja az adott játékos rögzített pontszámát, feltéve, ha az elért szám magasabb, mint a rögzített szám. Ebben a táblában az elsődleges kulcs nem más, mint az azonosító szám.

A másik két tábla felépítésben azonos, de két külön játékmód eredményeit követik. Tartalmazzák ezáltal a játékosok azonosítóit (playerid), az objektíva elérése alapján kiszámított pontszámot (score), valamint a pontszám elérésének dátumát (date). Elsődleges kulcs szintén az azonosító szám.

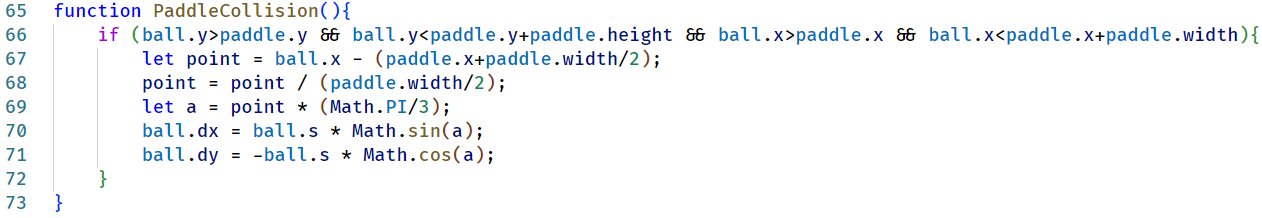
## Részletes feladatspecifikáció, algoritmusok

A program megvalósításához ezek voltak a fő elvárásaim:

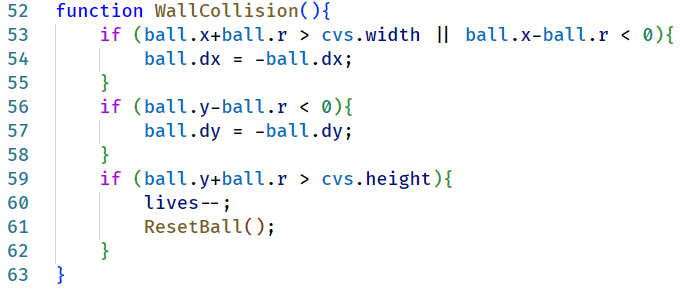
* Fizikai motor, amely megfelelően kezeli a labda mozgását és ütközését
* Téglák, amelyek ütközésnél törnek
* Időmérő rendszer
* Power-up-ok, melyek a játékos hasznára szolgálnak
* Választható játékmódok
* Adatbázis a pontok követésére

### Fizika

A fizika két részből áll, ütközés ellenőrzésből, és vektor kiszámításból.

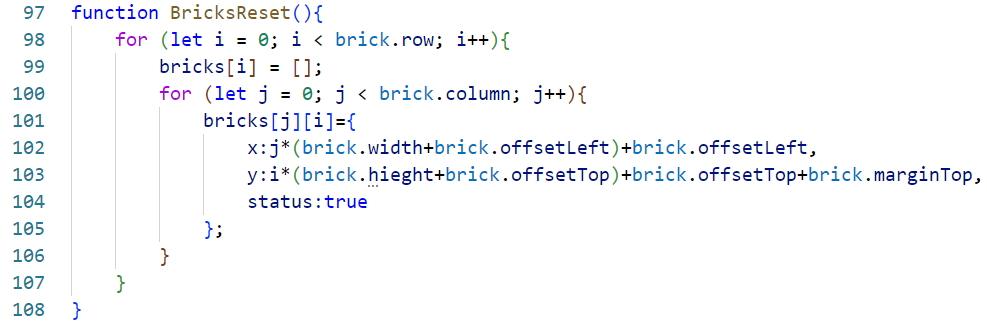


A program először ellenőrzi, hogy a labda koordinátái érintik-e azt a területet, ahol a játékos karakter helyezkedik el. Ha igen, kiszámítja pontosan hol találta el a játékost, majd ezt a találati pontot normalizálja -1 és 1 között. Ezt az értéket egy szinusz és koszinusz metódusnak átadva, a labda sebességét is bele számítva, megkapjuk pontosan milyen irányba milyen gyorsasággal kell pattanjon a labda.



A fallal való ütközet sokkal egyszerűbb, hiszen a szög (és ezáltal a sebesség) nem változik. Egyszerűen a falnak függvényében megfordítjuk a labda horizontális, illetve vertikális irányát. Kivétel az alsó fal, hiszen azzal ütközve életet vesztünk és a játék újra indul.

### Téglák

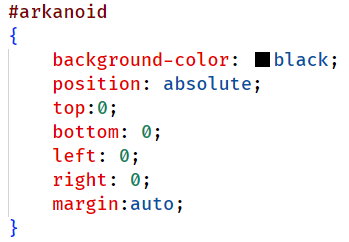


A téglák egy tömb változóban tárolódnak el. Ezen a tömbön a program sorokként és oszlopokként átmegy, majd az adott tégla elhelyezését kiszámítja annak méreteiből, bele számolva a kihagyásokat. Ezek után a tégla státusza true értékűvé változik, ezzel jelezve a tégla létezését.

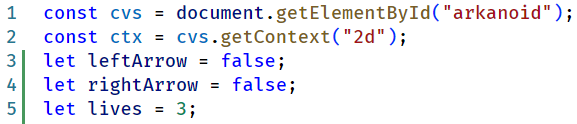
## Forráskód



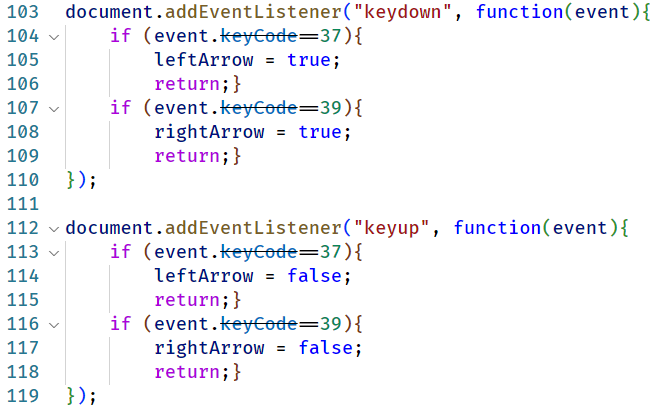
Mivel a program egy webapp, a „gerince” HTML nyelven íródott. A retro érzet elérése érdekében 4:3 képformátumba formáztam a canvas elemet, ami lehetővé teszi a 2D formák kirajzolását a monitorra. Az ezt követő script tag megidézi a játékmódhoz tartozó JavaScript fájlt.



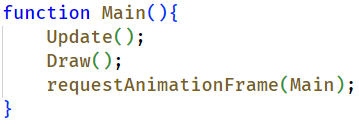
CSS stílus formázás csak minimális formában jelenik meg a projektben, törekedtem úgy tervezni a programot, hogy csak minimális korrigálás legyen szükséges.



A játéklogika kidolgozása előtt pár fontosabb változó van deklarálva, amik később fontos szerepeket töltenek be. Ilyen az élet számláló, az iránygombok státuszai, valamint a canvas context, ami a 2D elemeket rajzolja a képernyőre.



Az irányítás egyszerű EventListener metódus használatával működik. Gomblenyomáskor illetve felengedéskor ellenőrzi melyik gomb státusza változott, és az alapján egy bool változó értékét igazítja.



A fő metódus csupán a logika frissítését és a képernyőre rajzolás meghívását tartalmazza, majd egy animációs loopot hív aminek hála folyamatosan futni fog a program.

## Tesztelési dokumentáció

### Fizika teszt

Fallal való ütközéskor a labda a várt módon reagált, hibák nélkül. Megjegyeztem, hogy rendkívül magas sebességeknél a labda kitört a falak határán kívülre, de ez átlagos játékmenet alatt nem okoz problémát.

Játékossal való ütközéskor a labda eleinte váratlan hibákat mutatott, váratlan módon változott a sebessége. Ezt a hibát ki tudtam küszöbölni, és a labda gyorsasága az egész játék alatt konzisztens.

A téglákkal való ütközés szintén a várt eredményt mutatta. A tégla kollíziója megegyezik a faléval, majd ütközéskör a tégla példány eltűnik a játéktérről.

# A felhasználói dokumentáció

## A program általános specifikációja

## Rendszerkövetelmények

## A program telepítése

## A program használatának a részletes leírása

# Összegzés

# Irodalomjegyzék, forrásmegjelölés