EÖTVÖS LORÁND
TUDOMÁNYEGYETEM

Informatika Kar

Önmagát megoldó Rubik-kocka

Témavezető:

Dr. Gregorics Tibor

Szerző:

Gál Gergely

Budapest, 2023

Tartalom

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Gregorics Tibornak, aki szakértelmével, hasznos magyarázataival és a konzultációk során biztosított nélkülözhetetlen tanácsaival hatalmas segítséget nyújtott szakdolgozatom elkészüléséhez.

Hálával tartozom továbbá szüleimnek, akik nélkül ez a szakdolgozat nem jöhetett volna létre. Köszönöm nekik, hogy tanulmányaim során türelemmel és megértéssel támogattak, és minden helyzetben mellettem álltak.

Köszönöm mindenkinek!

1. Témabejelentő

A Rubik-kocka kirakása már évtizedek óta foglalkoztatja embereket. A színek megfelelő helyre illesztése egy nagyon bonyolult probléma. A feladatra már rengeteg megoldás született, mindegyik algoritmus más és más lépéskombinációkkal oldja meg a problémát. Vannak olyan algoritmusok, amelyekkel az emberek számára gyorsan kirakható a kocka. Van, amelyik hosszú számolás után a legkevesebb lépést próbálja elérni, ez maximum 20 lépés. Tervem szerint az általam használt algoritmus azt szolgálja, hogy az emberek "vakon" tudják kirakni ezt a logikai játékot.

Az alap algoritmus, amit megtanulhatunk a kockával, az Old Pochmann metódus lesz, mely mindössze 3 permutációt használ, a J-t, az Y-t és a T-t. Az Old Pochmann metódus egy olyan algoritmus, melyben elválasztjuk az éleket a sarkoktól és ezeket két külön álló egységként kezelve oldjuk meg az elemek helyreillesztését. A kevert kockaállás megtekintése után bekötik a játékos szemét, és ez után, vakon kell kiraknia a kockát. Az én alkalmazásom ezt fogja imitálni, és megtanítja a felhasználót a vakon kirakás módszerére. A kockát a játék megkezdése előtt tetszőlegesen összekeverhetik, vagy más algoritmusok megtanulására is használhatják. Az általam kidolgozott alkalmazáshoz saját algoritmusokat (lépés-sorozatokat) lehet hozzáadni, és ezeknek a memorizálásában is tud segíteni a program.

A program Java alapú Processing 4-ben íródik. A Processing [1] egy egyszerű programozási környezet, amelyet azért hoztak létre, hogy megkönnyítse a vizuálisan orientált alkalmazások fejlesztését, különös tekintettel az animációra, és azonnali visszajelzést adjon a felhasználóknak az interakción keresztül.

2. Bevezetés

Egészen fiatal korom óta rengeteg időt szántam a Rubik-kockával, és annak különböző méretű és változatú kockáival való játékra, azok kirakására, új és új kirakási módok kitalálására. Gyakorlatilag minden szabadidőmet ezzel töltöttem. A különböző Rubik játékok, így a Rubik-kocka lényege is abban áll, hogy az összekevert színek a különböző forgatás, mozgatás kombinációkkal a helyükre kerüljenek, és színazonos oldalak jöjjenek létre. A mozgatások, forgatások lehetnek függőleges és vízszintes irányúak, illetve ezek különböző kombinációi. Legnagyobb örömömre sikerült rengeteg olyan embert megismernem, akik nálam is sokkal régebben és mélyebben foglalkoznak ezekkel a logikai játékokkal és tanítják a világot, önmagukat, és mindenkit, aki nyitott erre. Szeretném ebből a rengeteg emberből kiemelni Török Ágostont, aki a magyar Rubik-kockás palettán rengeteg internetes tartalommal segít az érdeklődőknek a fejlődésben.

A szakdolgozatom témáját a Coding Train [6] YouTube-csatorna egy „Coding Challenge #142: Rubik's Cube” című videója ihlette. Mikor ezt néztem jutott először eszembe a gondolat, hogy egy önmegoldó Rubik-kockát szeretnék csinálni, ami segíti a felhasználót megtanulni kirakni a játékot. Ez a csatorna rengeteg segítség volt, mind a Processing megtanulásában, mint az alap kocka strukturális és vizuális megalkotásában.

Ezzel a projekttel a célom, hogy akár a teljesen kezdők, akár a Rubik-kockát már kirakni képes játékosok megtanulhassák, hogyan működik a vakon kirakás a Rubik-kockázás világán belül. Ez a program azért volt számomra érdekes, mert valójában sosem sikerült megtanulnom teljesen a vakon kirakás metódusát. Tehát laikus szemmel, de ugyanakkor mégis egy tapasztalt kockázó látásmódjával és lelkesedésével állhattam neki a munkának.

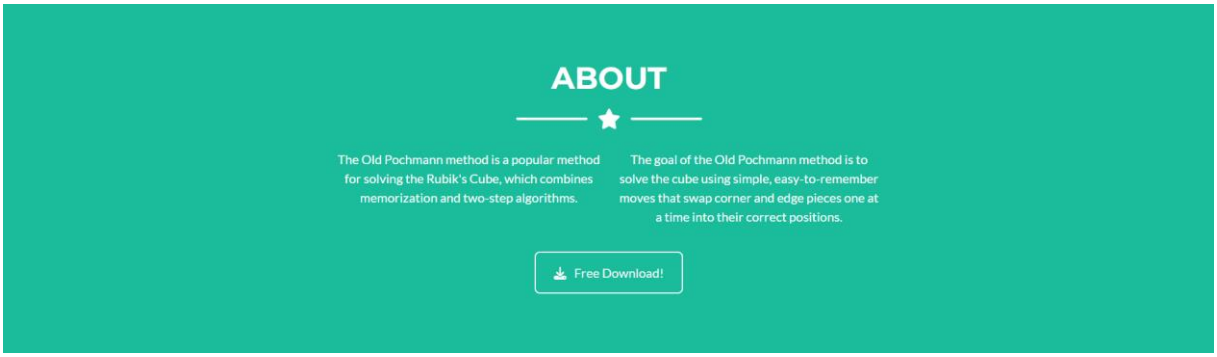
3. Felhasználói dokumentáció

3.1. Az alkalmazásról

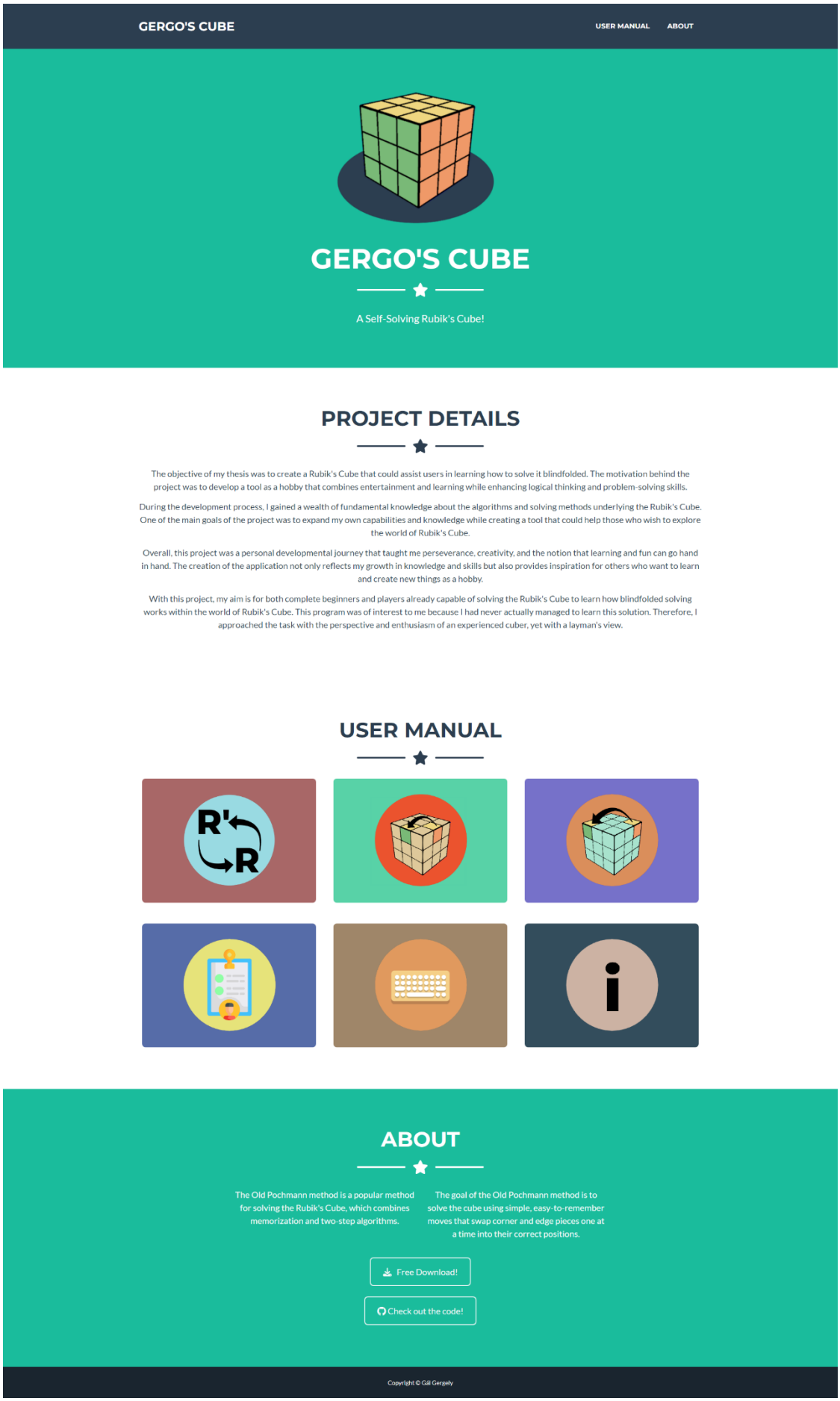
Az alkalmazás célja, hogy annak felhasználója elsajátíthassa a Rubik-kocka vakon kirakásának módszerét, saját, megtanulásra váró algoritmusait összegyűjthesse, gyakorolhassa, továbbá megtekinthesse, hogy egy gép hogyan oldja meg ezt a feladatot. Ebben a fejezetben a felhasználó megismerkedhet az alkalmazás felületével, funkcióival és azok használatával, valamint azok hasznos alkalmazásával a Rubik-kocka vakon kirakásának elsajátításához.

3.2. Az alkalmazás futtatása

Az alkalmazás egy Java formátumban megvalósított program, melynek használati útmutatója, billentyűzet beállítások, letöltési útmutatók és még sok egyéb hasznos információ megtalálható a https://people.inf.elte.hu/gt8yb1/ weboldalon. Az oldal alján található egy letöltés gomb, amely egy Java futtatható állományt és a további szükséges fájlokat tartalmazza. Amennyiben a felhasználó rendelkezik a megfelelő Java verzióval, nincs szükség további lépésekre a program használatához. A felhasználók könnyedén hozzáférhetnek a szükséges erőforrásokhoz és a program funkcionalitásához az említett weboldal segítségével.



1. ábra a letöltés gombhoz a weboldalon

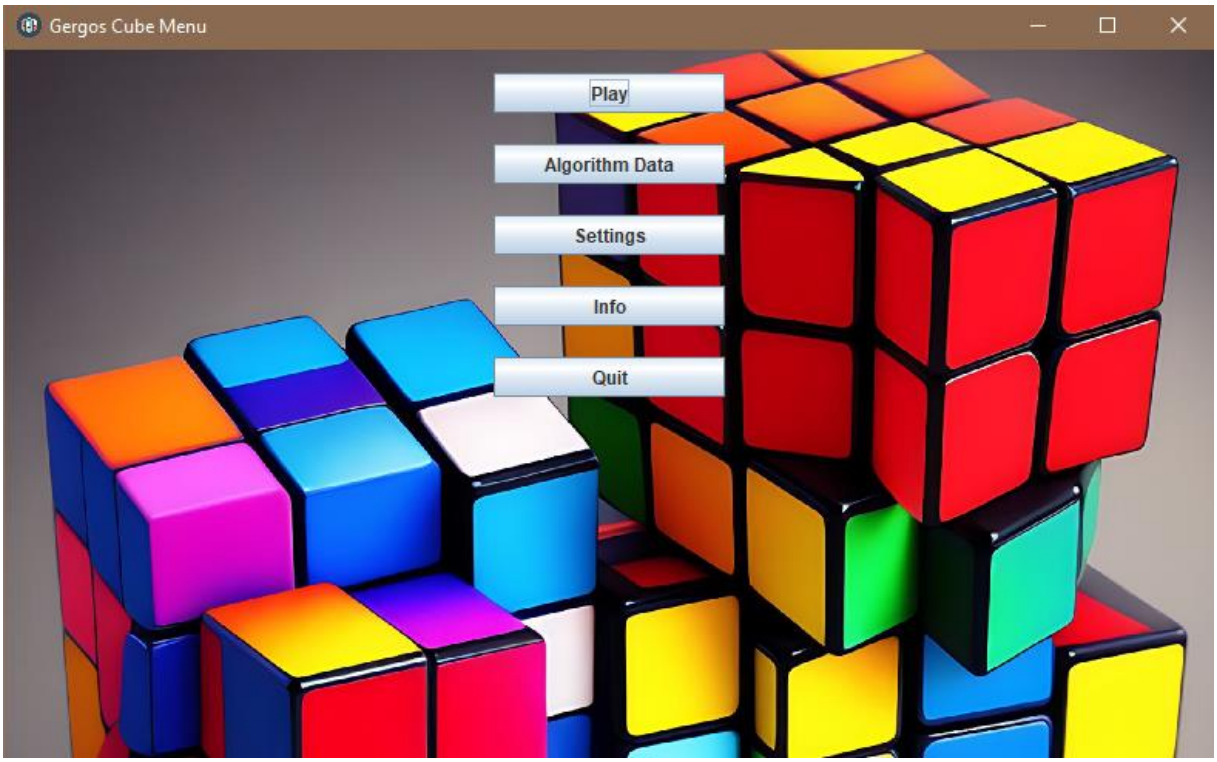


2. ábra a teljes weboldalról

3.3. Az alkalmazás használata

3.3.1. Menü

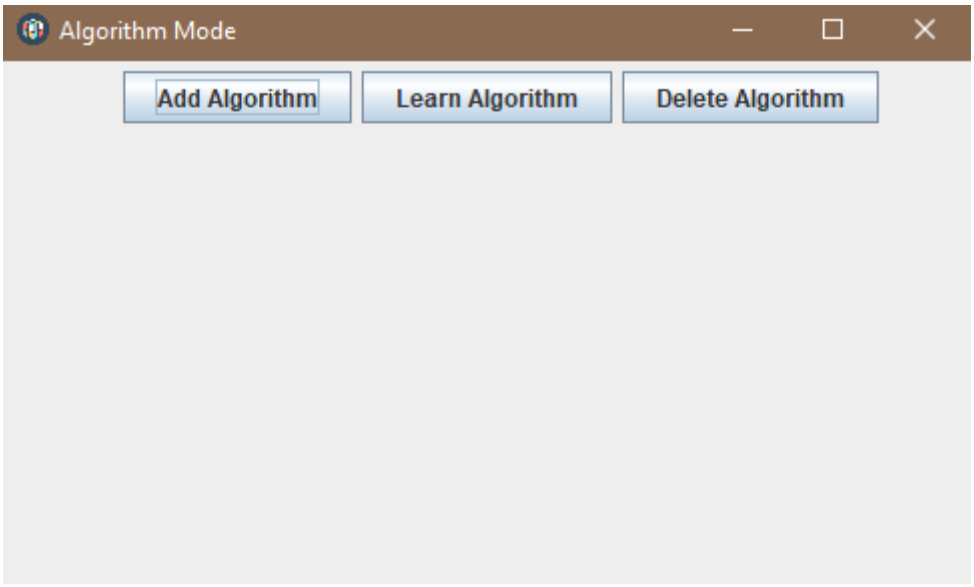
A főmenü tartalmazza a "Play" gombot, amely az automatikus megoldásért és a lépésről lépésre történő megoldásért felelős. A menüből lehet elérni az "Algorithm Data" módot, amelyben a felhasználó saját algoritmusait kezelheti. A "Beállítások" menüpont felelős az egyedi beállításokért és azok mentéséért. Ilyen beállítások például a színek, vagy hogy megjelenjenek e a Face-ID nak nevezett jelölések a matricákon. Itt található továbbá az "Info" gomb, amely az előbbiekben ismertetett https://people.inf.elte.hu/gt8yb1/ weboldalra vezet, valamint a "Kilépés" gomb, amely segítségével a felhasználó bezárhatja az alkalmazást.



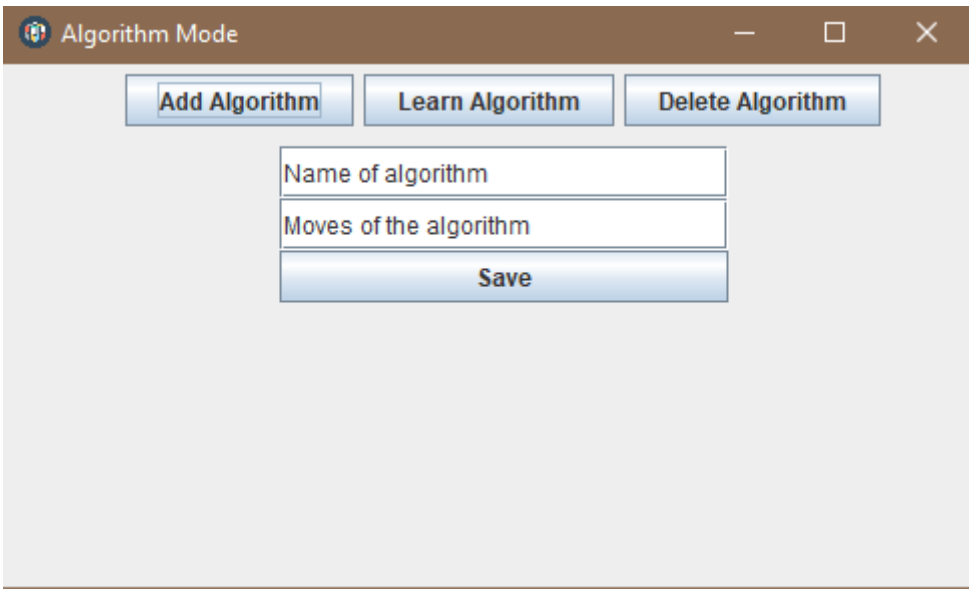
3. ábra Főmenü

3.3.2. Az „Algorithm Data” menü

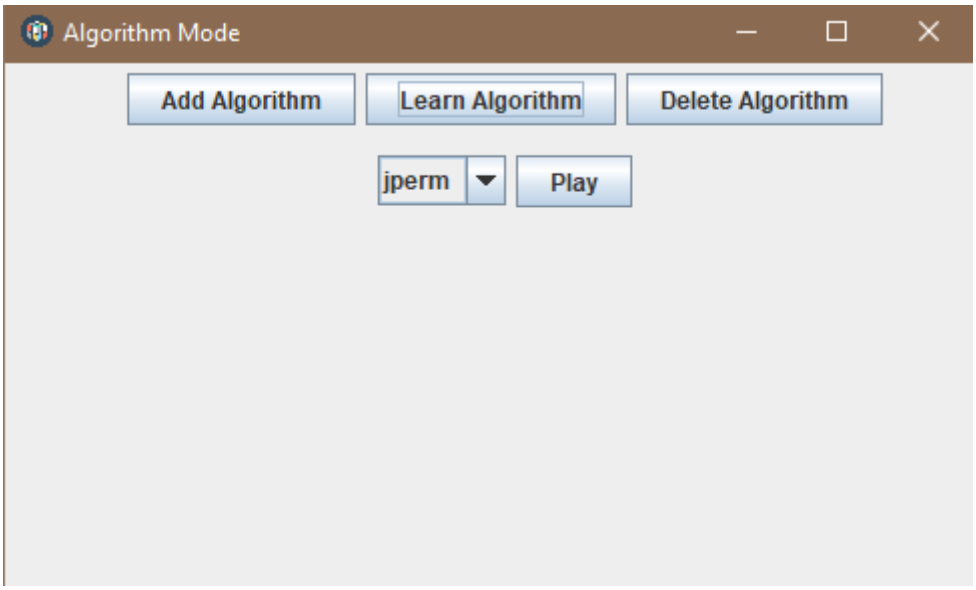
Az itt található 3 gomb az algoritmusok hozzáadásáért, törléséért, és ezek gyakorlásáért felel.



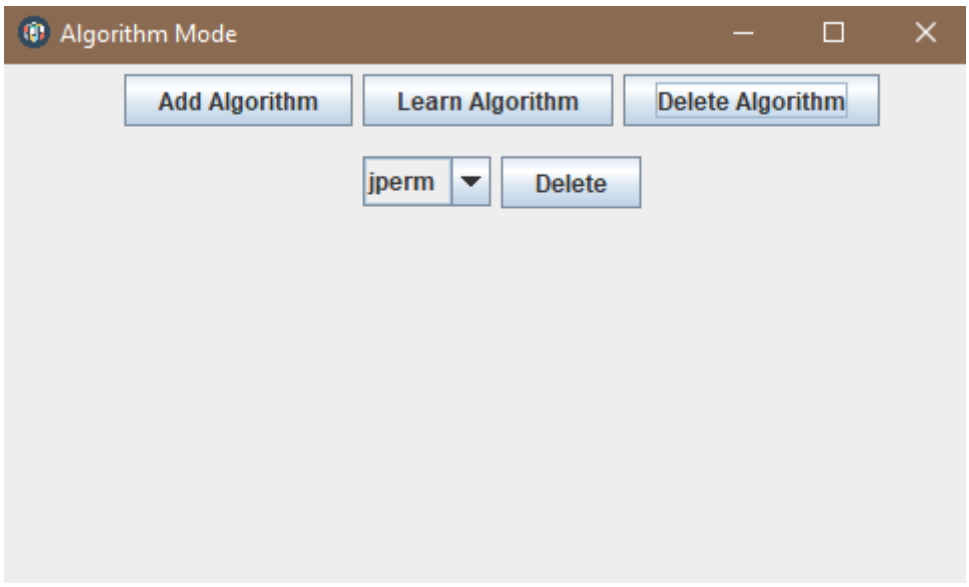
4. ábra „Algorithm Mode” menüpont



5. ábra "Algorithm Mode"- Add Algorithm



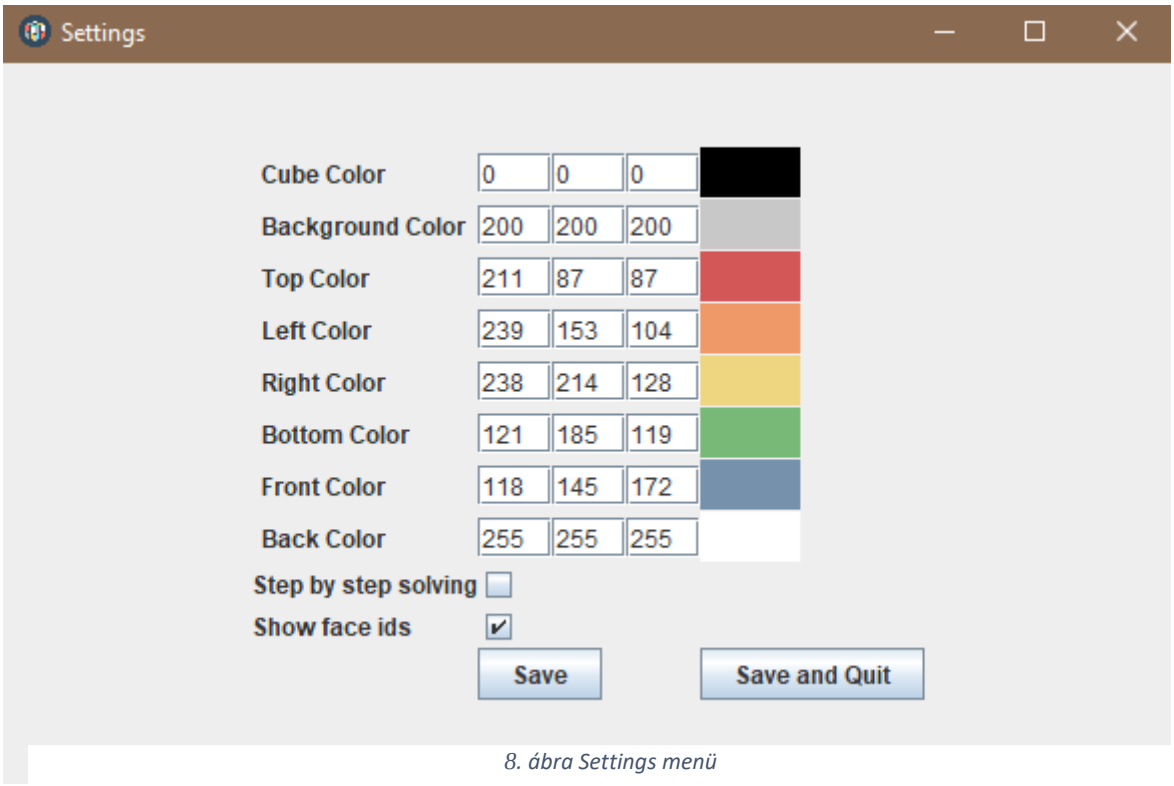
6. ábra „Algorithm Mode” – Learn Algorithm



7. ábra „Algorithm Mode” – Delete Algorithm

3.3.3. A „Settings” menü

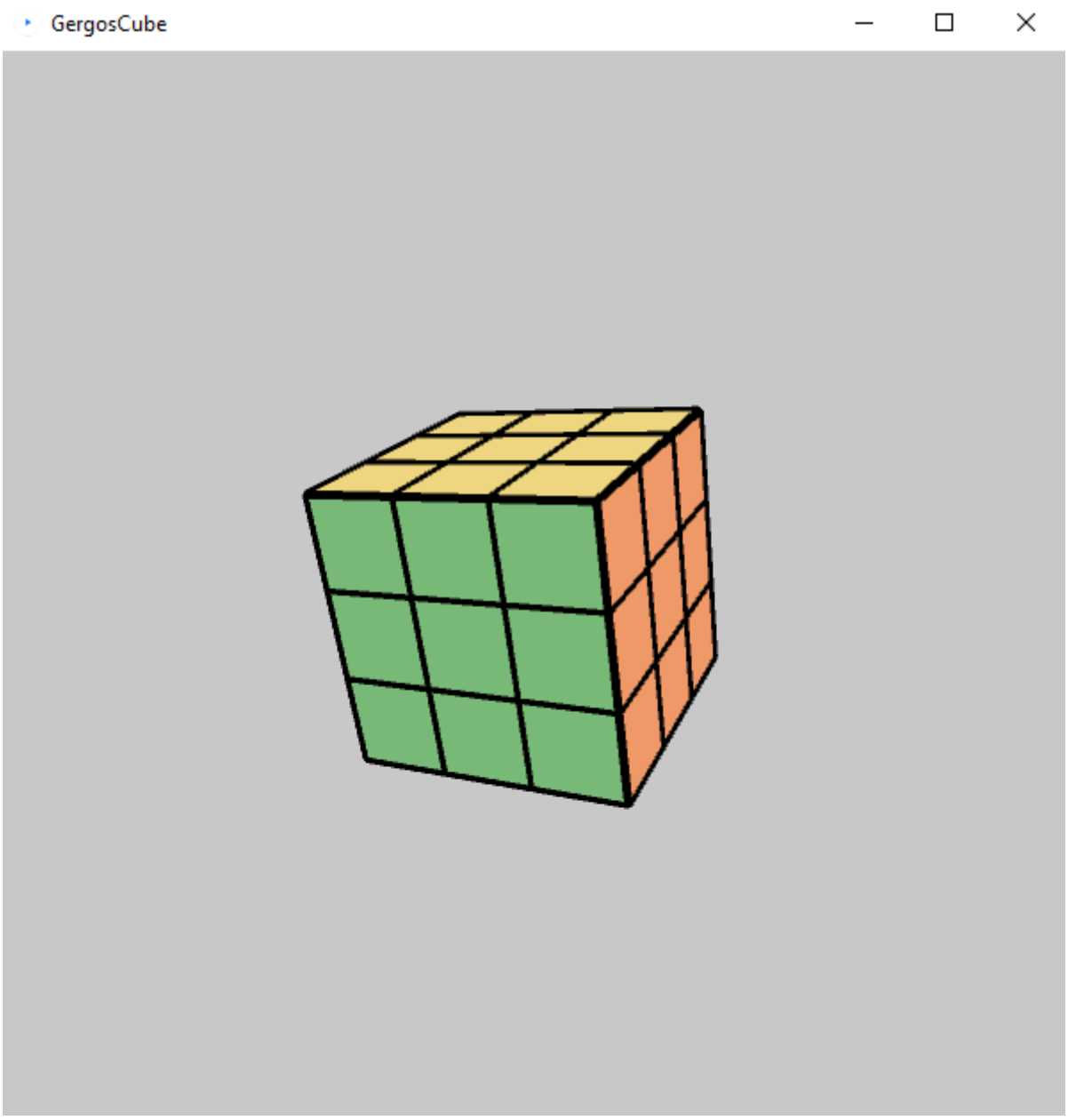
Ezen a menüponton keresztül lehet beállítani a kocka színeit, a Face ID-k megjelenését, továbbá azt, hogy a program megálljon-e a különböző lépések demonstrációja között. Ezeket az adatokat a program elmenti, és a következő indításkor már az új beállítások lesznek az alapértelmezettek.



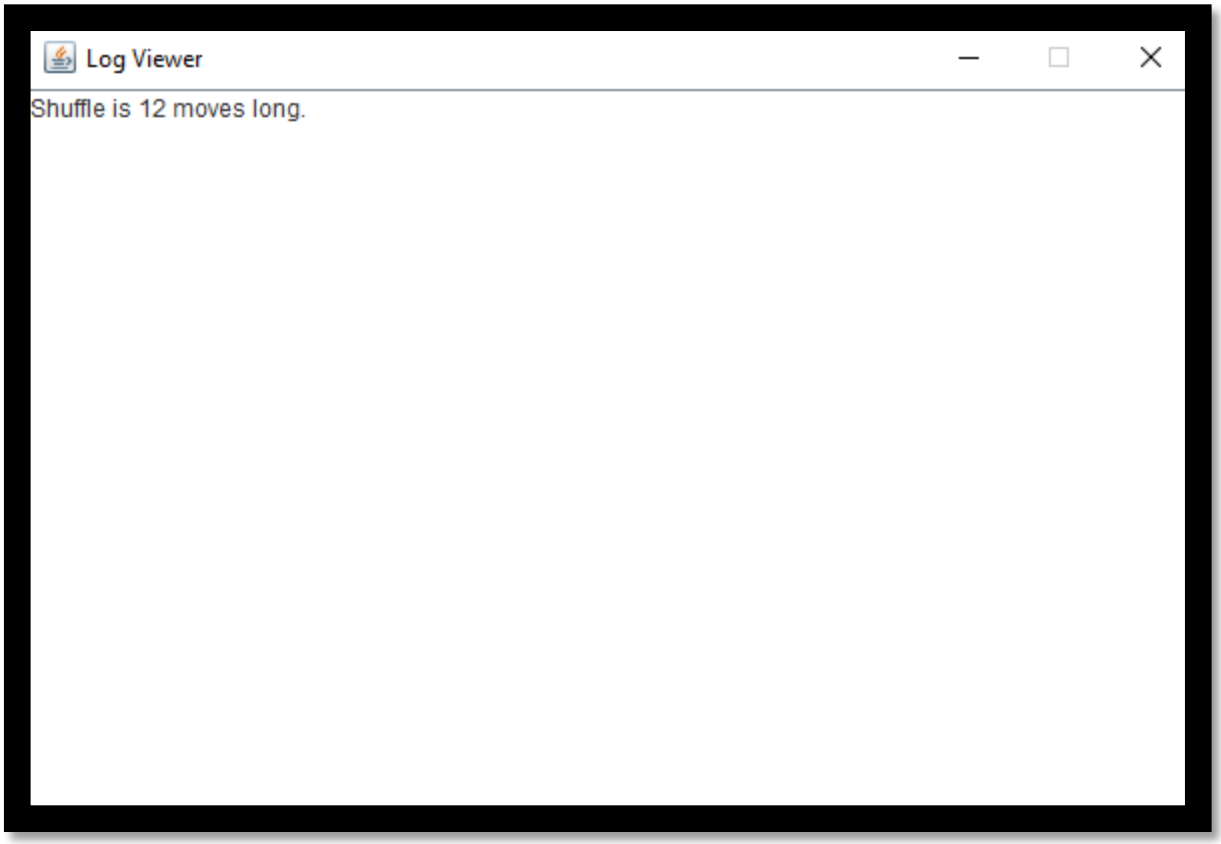
8. ábra Settings menü

3.3.4. A játék

A „Play” gomb megnyomásakor 2 ablak nyílik meg. Az első a Rubik-kockát tartalmazza, a másikon pedig az utasítások és a lépések jelennek meg.



9. ábra GergosCube ablak



10. ábra Log Viewer információs panel

4. Fejlesztői dokumentáció

4.1. Elemzés

4.1.1. Feladat leírás

A fejlesztendő alkalmazás célja, hogy a felhasználók megtanulják a Rubik-kocka vakon történő megoldását az Old Pochmann módszerrel. Az alkalmazás fő célja nem a leggyorsabb vagy legoptimálisabb megoldás megtalálása, hanem a felhasználók oktatása a módszer elsajátításában.

A projekt részletei a következők:

1. Az alkalmazásnak biztosítania kell egy intuitív és könnyen használható felületet a Rubik-kocka vakon történő megoldásának tanulására az Old Pochmann módszerrel.

2. Az alkalmazásnak lehetővé kell tennie a felhasználók számára, hogy saját lépéssorozatokat, úgynevezett "algoritmusokat" adjanak hozzá, melyek segítségével saját technikákat is kifejleszthetnek a kocka oldalainak azonos színnel történő kirakására.

3. A felhasználóknak lehetőségük kell legyen az algoritmusok mentésére, törlésére és módosítására.

4. Az alkalmazásnak tartalmaznia kell egy olyan felületet, ahol a felhasználók gyakorolhatják a megtanult algoritmusokat, és fokozatosan fejleszthetik a Rubik-kocka megoldására vonatkozó képességeiket.

5. A kocka körül forgatható, és gombnyomással „hajtható” kell legyen.

6. A fejlesztendő alkalmazásnak kompatibilisnek kell Windows 10 rendszerrel.

4.2. Funkcionális leírás

1. Az alkalmazásfejlesztés először egy Processing [1] “sketch” elkészítésével kezdődjön.

2. A sketch-et a kocka elkészültekor fordítsuk Java alkalmazássá.

3. Ezután ezt az egységesen generált kód kerüljön feldarabolásra, és Java platformon működjön.

4. Az alkalmazás grafikus felhasználói felületet (GUI) használjon a könnyebb kezelhetőség érdekében, például JavaFX vagy Swing technológiával.

5. Az alkalmazásban jelenjen meg egy 3D-s Rubik-kocka modell, amellyel a felhasználók interaktívan gyakorolhatják a megoldást.

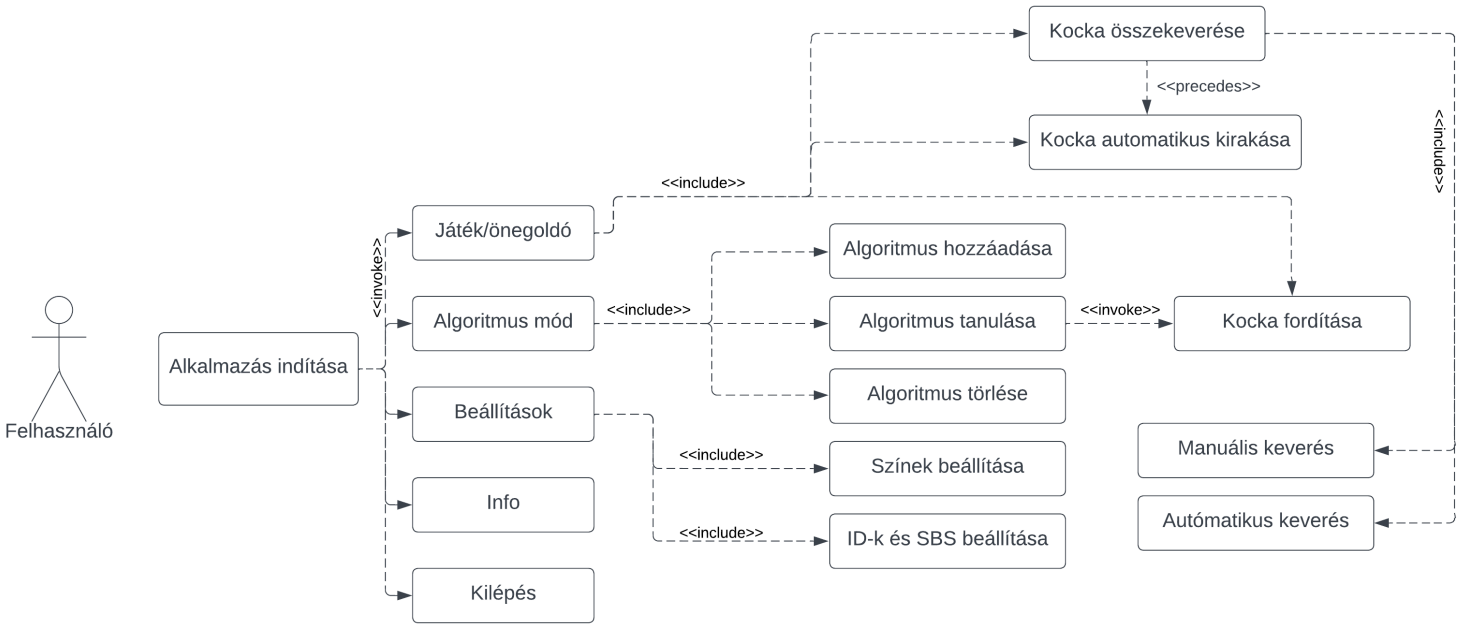
6. Az alkalmazásnak támogatnia kell az Old Pochmann módszer alapján történő vakon megoldást, és ehhez lépésről lépésre útmutatást kell nyújtania.

7. Az alkalmazásban legyen lehetőség saját algoritmusok hozzáadására, mentésére, törlésére és módosítására.

8. Az alkalmazásban legyen lehetőség a felhasználók számára, hogy a megtanult algoritmusokat gyakorolhassák, és visszajelzéseket kapjanak a sikeres vagy sikertelen próbálkozásról.

9. Az alkalmazásnak tartalmaznia kell egy súgó vagy útmutató részt, ahol a felhasználók segítséget kaphatnak az alkalmazás használatával kapcsolatban.

4.2.1.1. Usecase diagram



11. *ábra Felhasználói esetdiagram*

4.2.1.2. User Story táblázat

ID	Felhasználói eset	Leírás	
1	Alkalmazás indítása	GIVEN:	Az alkalmazás telepítve van
		WHEN:	Alkalmazás indítása
		THEN:	A menü megjelenik
2	Kilépés	GIVEN:	Menü felület látható
		WHEN:	Kilépési szándék
		THEN:	Alkalmazás bezárása
3	Algoritmus tanulás menü	GIVEN:	Menü felület látható
		WHEN:	„Algorithm Data” menüpont választása
		THEN:	Az algoritmus tanulás menü megjelenik
4	Meglevő algoritmus tanulása	GIVEN:	Algoritmus tanulási menü látható
		WHEN:	A „Learn Algorithm” menüpot választása
		THEN:	A játék elindul a választott algoritmussal
5	Új algoritmus tanulása	GIVEN:	Algoritmus tanulási menü látható
		WHEN:	„Add Algorithm” menüpont választása
		THEN:	A felhasználó megadhatja az általa választott algoritmus adatait
6	Algoritmus törlése	GIVEN:	Algoritmus tanulási menü látható
		WHEN:	„Delete Algorithm” menüpont választása
		THEN:	Az algoritmus törlődik
7	Önmegoldó mód	GIVEN:	Menü felület látható
		WHEN:	Play menüpont választása
		THEN:	A játék elindul a megadott beállításokkal
8	Beállítások menü	GIVEN:	Menü felület látható
		WHEN:	„Settings” menüpont választása
		THEN:	A Beállítások menü megjelenik
9	Infó menüpont	GIVEN:	Menü felület látható
		WHEN:	„Info” menüpont választása
		THEN:	Az információs weboldal megjelenik
10	Kocka fordítása	GIVEN:	A játék fut
		WHEN:	Valamelyik fordítási input (billentyű kombináció) lenyomásra kerül
		THEN:	A kocka elfordul
11	Kocka összekeverése	GIVEN:	A játék fut, és a kirakás nem kezdődött el
		WHEN:	A keverés input (billentyű kombináció) lenyomásra kerül
		THEN:	A kocka 10-20 lépés alatt összekeveri magát
12	Sikeres algoritmus kirakás	GIVEN:	A játék fut, algoritmus tanulás módban
		WHEN:	A felhasználó sikeresen kirakja az algoritmust
		THEN:	Tetszőleges mozgatással a kocka visszaáll alaphelyzetbe
13	Sikertelen algoritmus kirakás	GIVEN:	A játék fut, algoritmus tanulás módban
		WHEN:	A felhasználó elrontja az algoritmust
		THEN:	A kocka visszaáll alaphelyzetbe

14	Kocka automatikus kirakása	GIVEN:	A játék fut
		WHEN:	A kirakás input (billentyű kombináció) lenyomásra kerül
		THEN:	A kocka nekiáll kirakni magát
15	Szabad játék	GIVEN:	A program nem kezdte el a kirakást
		WHEN:	Valamelyik fordítási input (billentyű kombináció) lenyomásra kerül
		THEN:	A kocka szabadon kirakható, forgatható

4.2.2. Nem-funkcionális leírás

1. Teljesítmény: Az alkalmazásnak gyorsan és zökkenőmentesen kell működnie, a felhasználói interakciókra gyors válaszidővel reagálva.

2. Felhasználóbarát: Az alkalmazásnak könnyen használhatónak és érthetőnek kell lennie a felhasználók számára, függetlenül attól, hogy mennyi tapasztalattal rendelkeznek a Rubik-kocka megoldásában. A program feltételezi, hogy a felhasználó alapvető ismeretekkel rendelkezik a Rubik-kockáról és áttanulmányozta az alkalmazás leírását.

3. Kompatibilitás: Az alkalmazásnak kompatibilisnek kell lennie Windows 10 operációs rendszerrel.

4. Megbízhatóság: Az alkalmazásnak hibamentesen kell működnie, és kezelnie kell a váratlan helyzeteket vagy felhasználói hibákat anélkül, hogy összeomlana vagy adatvesztést okozna.

5. Karbantarthatóság: Az alkalmazásnak jól strukturált és dokumentált kóddal kell rendelkeznie, hogy az könnyen karbantartható és bővíthető legyen.

6. Tesztelhetőség: Az alkalmazásnak lehetővé kell tennie a könnyű tesztelést és hibakeresést, valamint támogatnia kell az automatizált tesztelési megoldásokat.

7. Rugalmasság: Az alkalmazásnak képesnek kell lennie arra, hogy új funkciók és módosítások könnyen integrálhatók legyenek anélkül, hogy a meglévő funkcionalitást befolyásolnák.

8. Átadhatóság: Az alkalmazást úgy kell megtervezni és implementálni, hogy könnyen átadhassuk azt más fejlesztőknek vagy csapatoknak a további fejlesztés és karbantartás érdekében.

4.3. Tervezés

4.3.1. Modell és nézet

A tervezési folyamatot jelentősen befolyásolta a Processing [1] technológia, amely sajnálatos módon nem nyújt elegendő támogatást a modell és nézet éles elválasztásához. Ahol csak lehetőség nyílt rá, a menük és a tisztán logikai komponensek esetén végrehajtottam az elválasztást. Azonban a Processing osztályok esetén, melyek a kocka kezeléséért felelősek, ez a szétválasztás nem volt megvalósítható. További részleteket és magyarázatot a megvalósítás során, illetve a modell nézet menüpontban található leírásban olvashatunk.

4.3.1.1. Kapcsolat az osztályok között

A rendszerünkben az osztályok közötti kapcsolat a következőképpen formalizálható:

Ha a felhasználó az "Önmegoldás" módot választja, több osztály kerül használatba:

1. A Kocka Felépítéséért Felelős osztályok, melyek a kocka strukturális adatait kezelik:

   Cube, Cubie, Face

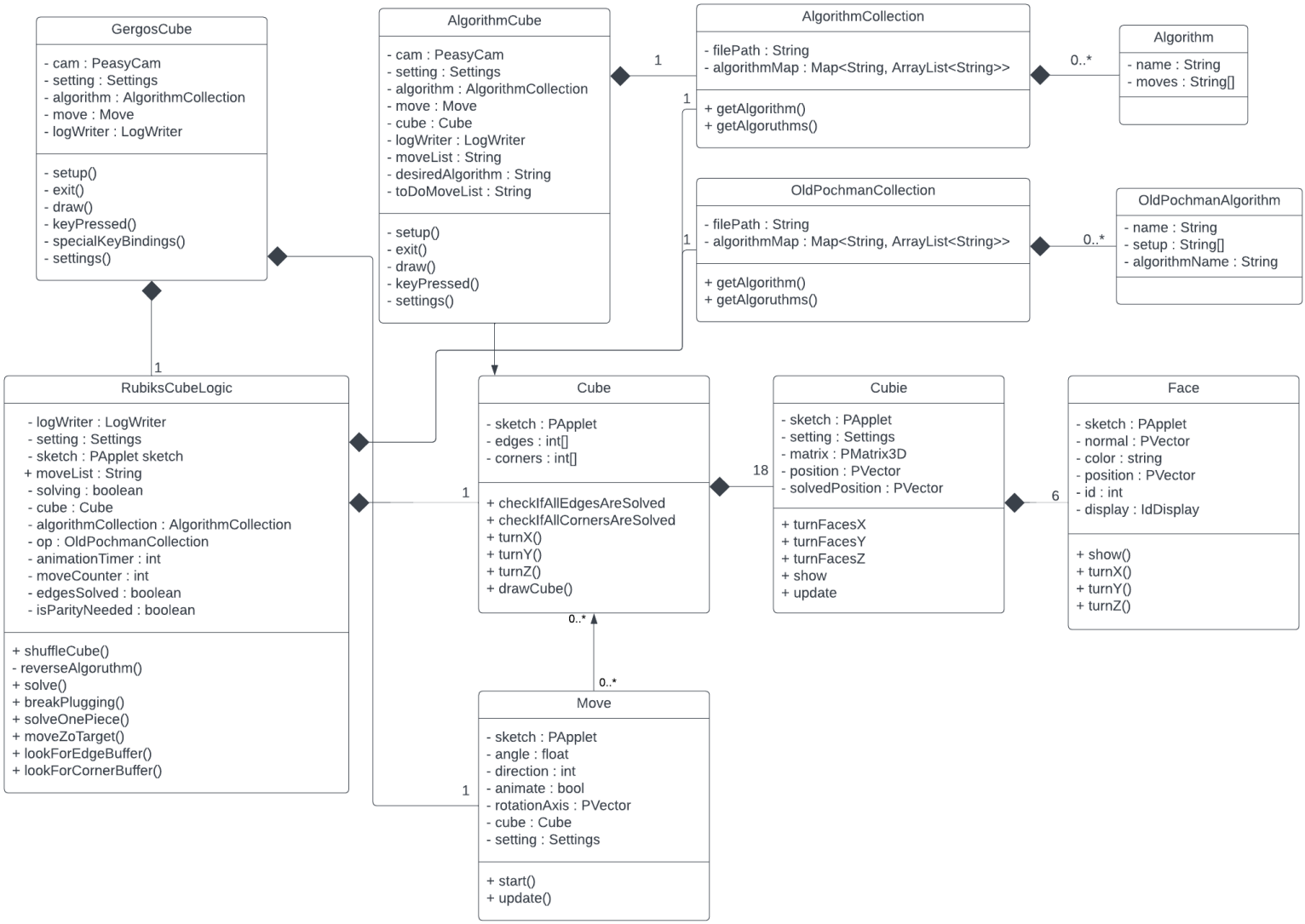
2. A "GergosCube" osztály, ami egy Processing [1] ablak. Ezen jelenik meg a kocka, melyet a kamera segítségével lehet kezelni.

3. A "RubiksCubeLogic" osztály, mely az Old Pochman metódus megvalósításáért felelős.

Ezen osztályok együttes működése eredményezi az önmegoldást. A Collection végződésű osztályok pedig a JSON fájlok olvasásáért és kezeléséért felelősek.

Az "Algoritmus" módot választva a "GergosCube" és a "RubiksCubeLogic" osztályok nem kerülnek példányosításra. Ebben az esetben az "AlgorithmCube" osztály kerül használatba, ami szintén egy Processing [1] osztály. Ez felelős a megjelenítésért, a kocka felépítéséért, valamint a lépéssorozatok ellenőrzéséért.

4.3.1.2. Osztály diagram



12. ábra Osztály diagram

4.3.1.3. Osztályok és főbb metódusaik

- **Face:** → A kockán egy matricát reprezentál. Van egy színe, id-je, és egy irány amerre néz. Metódusai a „facek" forgatásáért és megjelenítéséért felelősek.
    - **turnX():** → A kocka forgatásakor elforgatja a „face"-t a térben, az X tengely mentén, egy bizonyos fokkal.
    - **turnY():** → A kocka forgatásakor elforgatja a „face"-t a térben, az Y tengely mentén, egy bizonyos fokkal.
    - **turnZ():** → A kocka forgatásakor elforgatja a „face"-t a térben, a Z tengely mentén, egy bizonyos fokkal.
    - **rotateFacingY():** → A „face" forgatását követően beállítja hogy a térben merre néz.
    - **rotateFacingY():** → A „face" forgatását követően beállítja hogy a térben merre néz.
    - **rotateFacingZ():** → A „face" forgatását követően beállítja hogy a térben merre néz.

- **Cubie:** → A Rubik kocka egy kis elemét reprezentálja, ami lehet sarok, él, vagy akár a kocka magja. Metódusai a kocka térben forgatásáért, illetve a megjelenítéséért felelősek. Egy kockán 27 található belőle.
    - **turnFacesX():** → A kocka forgatásakor elforgatja a „cubie"-t a térben, az X tengely mentén, egy bizonyos irányba.
    - **turnFacesY():** → A kocka forgatásakor elforgatja a „cubie"-t a térben, az Y tengely mentén, egy bizonyos irányba.
    - **turnFacesZ():** → A kocka forgatásakor elforgatja a „cubie"-t a térben, az Z tengely mentén, egy bizonyos irányba.
    - **update():** → A „cubie" saját pozícióját frissíti az aktuális forgatás után egy 3 dimenziós mátrixban.
    - **show():** → A „cubie" és ezeken levő matricák megjelenítéséért felelős.

- **Cube:** → Egy Rubik kockát reprezentál. A metódusai a kocka forgatásáért, megjelenítéséért, és a kirakott státusz megállapításáért felelnek.
    - **checkIfAllEdgesAreSolved():** → Ellenőrzi, minden él kirakottnak tekinthető-e a kockán.
    - **checkIfAllCornersAreSolved():** → Ellenőrzi, minden sarok kirakottnak tekinthető-e a kockán.
    - **turnX():** → Elforgatja a kockát a választott tengelyen, irányba és oldalon.
    - **turnY():** → Elforgatja a kockát a választott tengelyen, irányba és oldalon.
    - **turnZ():** → Elforgatja a kockát a választott tengelyen, irányba és oldalon.
    - **drawCube():** → A Rubik kocka megjelenítéséért felelős.

- **Move:** → Az osztály egyetlen forgatást reprezentál egy megadott Rubik kockán. Tartalmazza a szükséges információkat a kocka adatainak frissítésére, és a lépés végrehajtására.
    - **start():** → Elindítja a mozgatási animációt.
    - **update():** → Frissíti, milyen szögben áll most a forgatás és annak animációja.

- **RubiksCubeLogic:** A Rubik-kocka megoldásának alapvető logikáját tartalmazza az Old Pochmann módszer használatával. Felelős a kocka állapotáért, mozgatásáért, keveréséért és megoldási algoritmusokáért. Kezeli továbbá az él és a sarokpuffer logikát, szükség esetén megszakítja a „dugulást".
    - **shuffleCube():** → A kocka összekeveréséért felelős.
    - **reverseAlgorithm()** → Visszafordít egy adott algoritmust.
    - **solve():** → A kocka autómatikus megoldásáért felelős algoritmus.
    - **breakPlugging():** → A kocka ezen megoldása esetén időnként előforduló úgynevetett „dugulás" elhárítására szolgáló metódus.
    - **SolveOnePiece():** → Az éppen aktuális buffer elem megoldásáért felelős.
    - **moveToTarget():** → Az éppen aktuális buffer elem helyét a csere pozícióba juttatja.
    - **lookForEdgeBuffer():** → Megkeresi az él bufferben elhelyezkedő elemet.
    - **lookForCornerBuffer():** → Megkeresi az sarok bufferben elhelyezkedő elemet.
    - **logMove():** → A lépést egy a felhasználó által értelmezhető formátumban kiírja.

- **AlgorithmCollection:** → Az adatbázisként használt JSON file-ból összegyűjti az OldPocham metódushoz szükséges algoritmusokat, továbbá a felhasználó által eltárolt algoritmusokat is. A fájl felépítése az pontban tekinthető meg.
    - **add():** → Egy algoritmust ad az adatbázishoz.
    - **delete():** → Egy algoritmust eltávolít az adatbázisból.

- **OldPochmanCollection():** → Az adatbázisként használt JSON file-ból összegyűjti az OldPocham metódushoz szükséges setupokat és a hozzájuk tartozó csere algoritmust. A felhasználó nem tud elemet adni az adatbázishoz. A fájl felépítése az Adatbázis pontban tekinthető meg.
    - **isItInAlgorithms():** → Megmondja, hogy a keresett lépés része e az adatbázisnak.

- **AlgorithmCube:** → Ez az osztály felelős azért, hogy a felhasználó egy adott algorutmust gyakorolhasson a kockán. Figyeli a forgatásokat, visszajelez a felhasználónak, továbbá felelős a megjelenítésért is. Az egyik fő oka a modell nézet ütközésének.

- **GergosCube:** → Ennek az osztálynak a felelőssége a szabad játék, és a kocka önkirakása. Az animáció, a logika kezelése, és a Processing [1] ablak megjelenítése a feladata. Az egyik fő oka a modell nézet ütközésének.

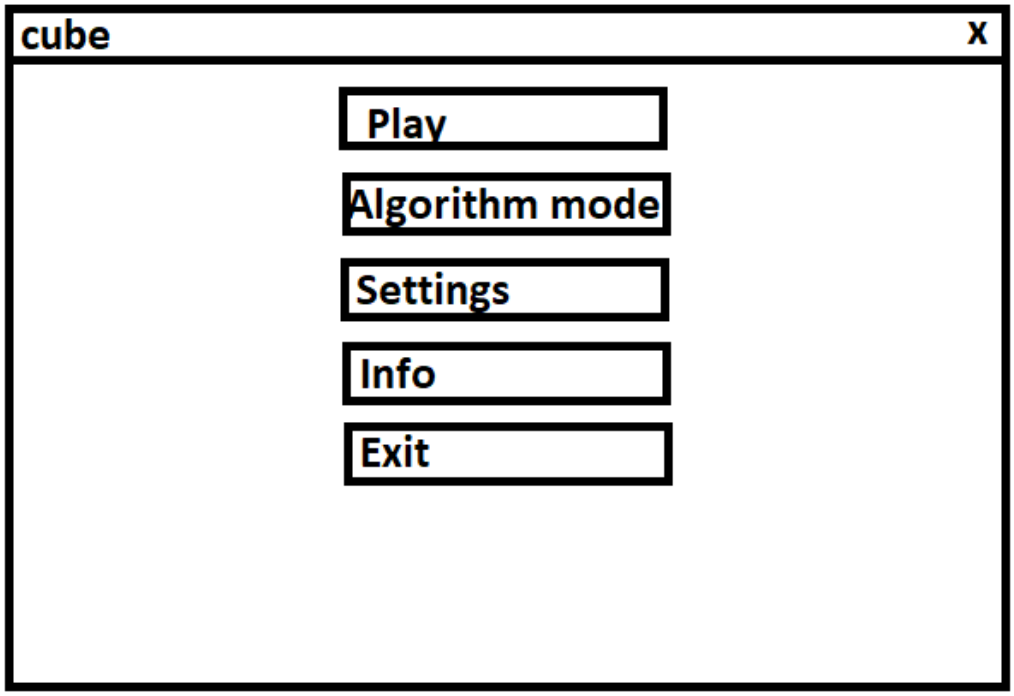
4.3.2. A megoldás heurisztikája

A Rubik-Kocka kirakásának a heurisztikáját a Kockások [5] YouTube-csatorna tulajdonosának köszönheti a dolgozat ugyanis a „Hogyan rakjuk ki a Rubik Kockát vakon? (Old Pochmann metódus)” című videoból tanultam meg a megoldás menetét, amelyet később felhasználtam a dolgozathoz.

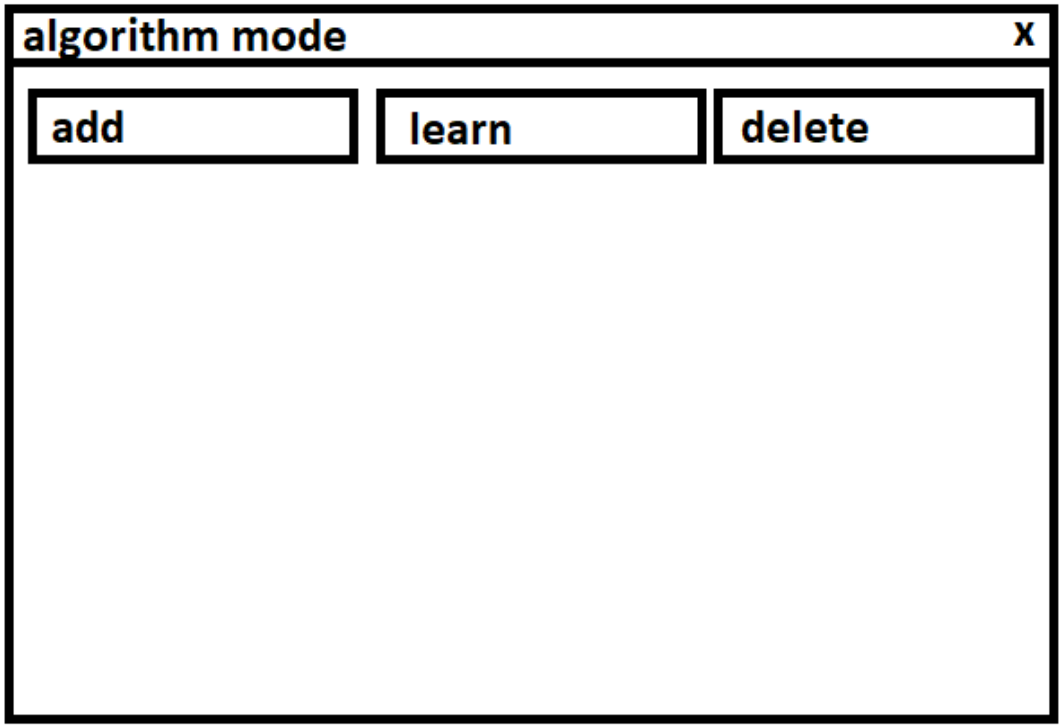
Különösen köszönöm Szántai Szabolcsnak a sarkok és élek „setup” -jához elkészített táblázatokat, melyek hatalmas segítséget nyújtottak a program elkészítéséhez.

4.3.3. Képernyőtervek

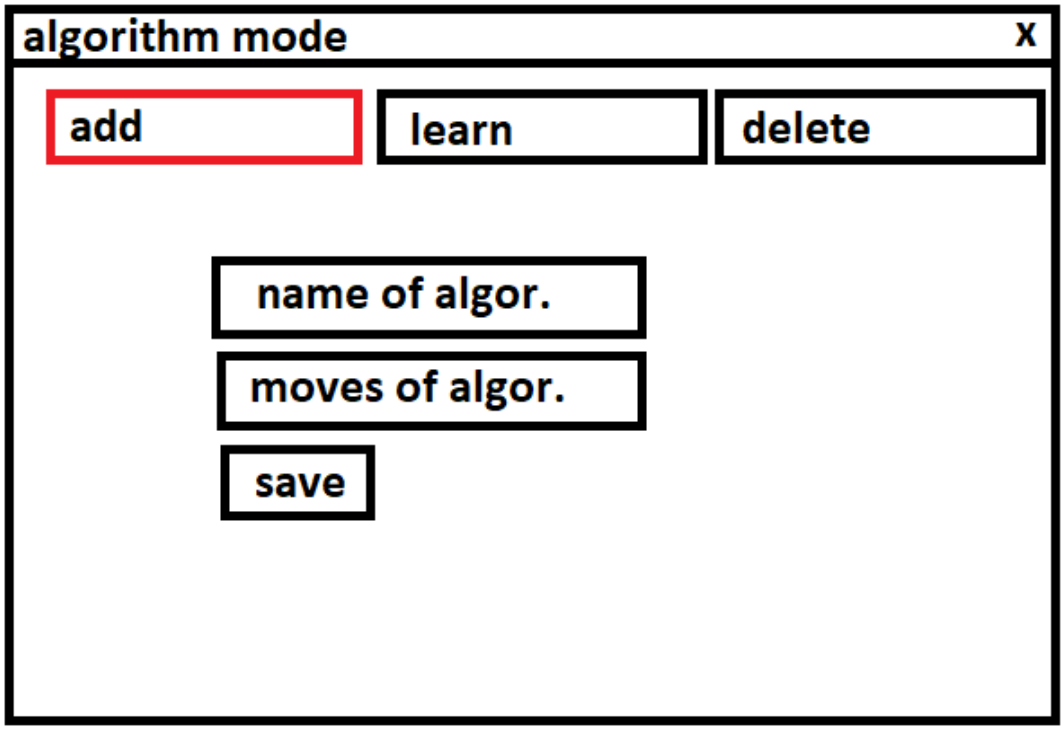
A főmenüből kiindulva érhetőek el a program különböző funkciói. A „Play” gombra kattintva használhatjuk az önkirakó módot, továbbá a szabad játékot. Az „Algorithm mode” menü tartalmazza magában a lehetőséget az algoritmusok tárolására, törlésére és a gyakorlásukra. A „Settings” pontban beállíthatjuk a kocka színeit, továbbá, hogy önkirakás esetén megálljon e minden lépés után. Az „Info” menü megnyitja a program weboldalát, ahol több hasznos leírást is találhatunk.



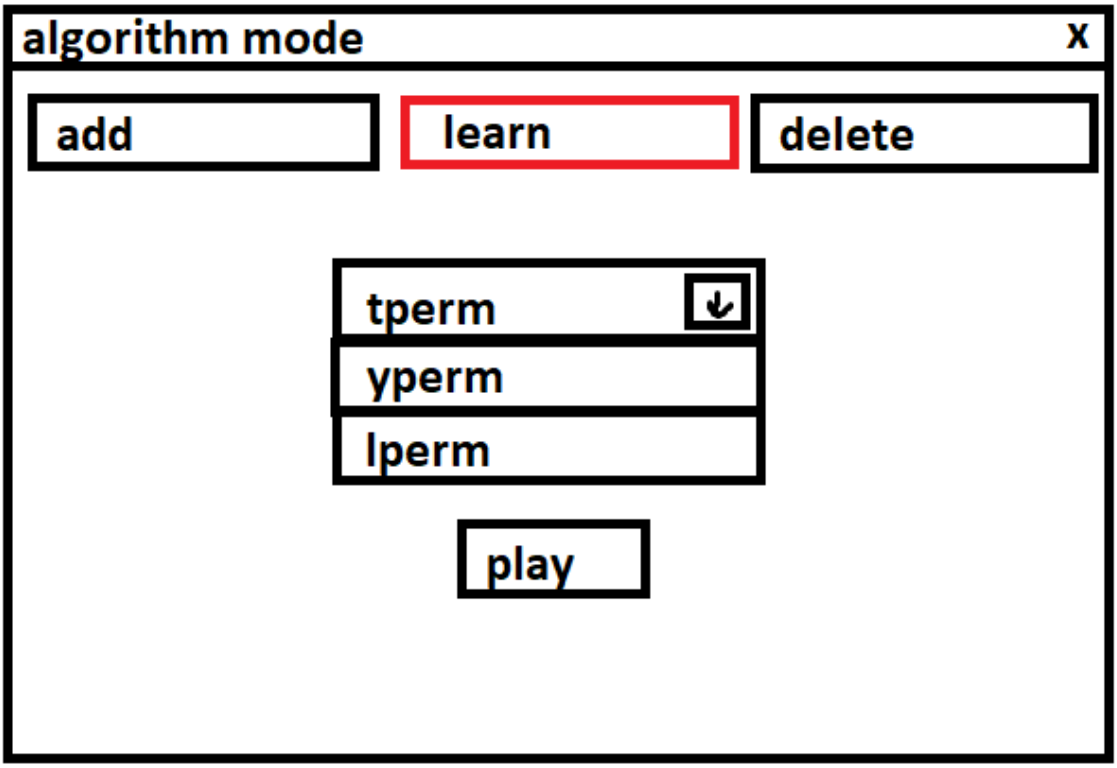
13. ábra Főmenü terve



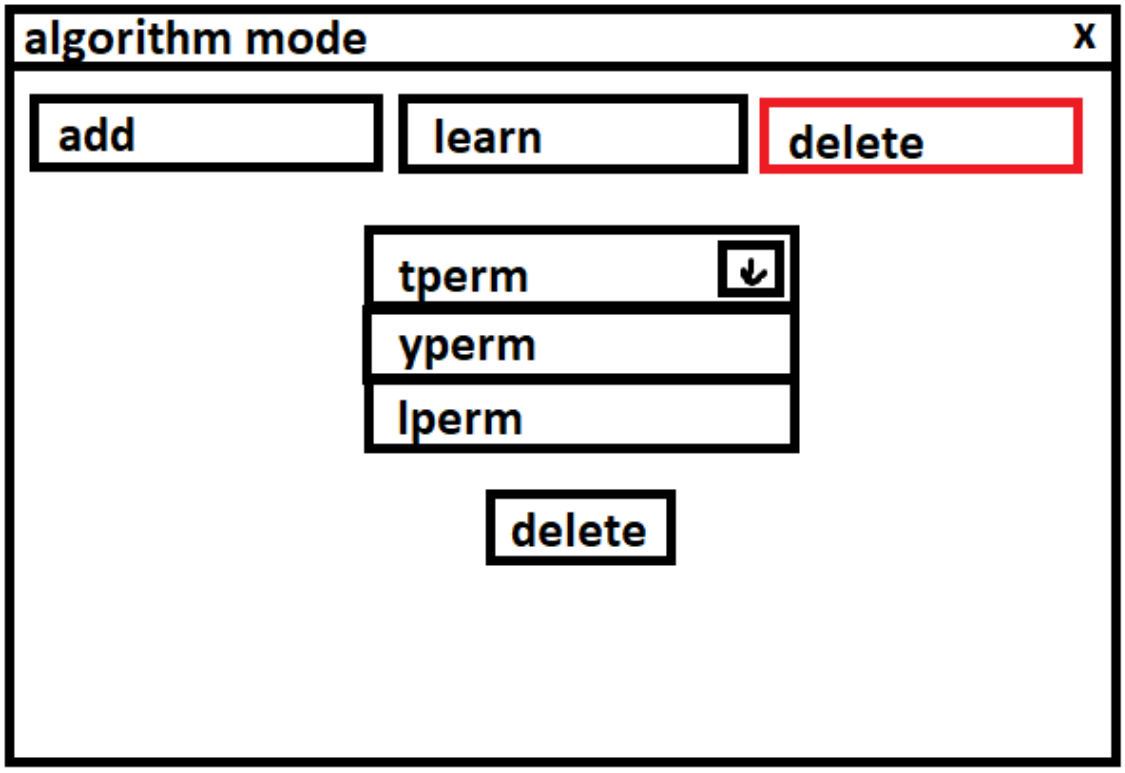
14. ábra Algorithm mode menü terve



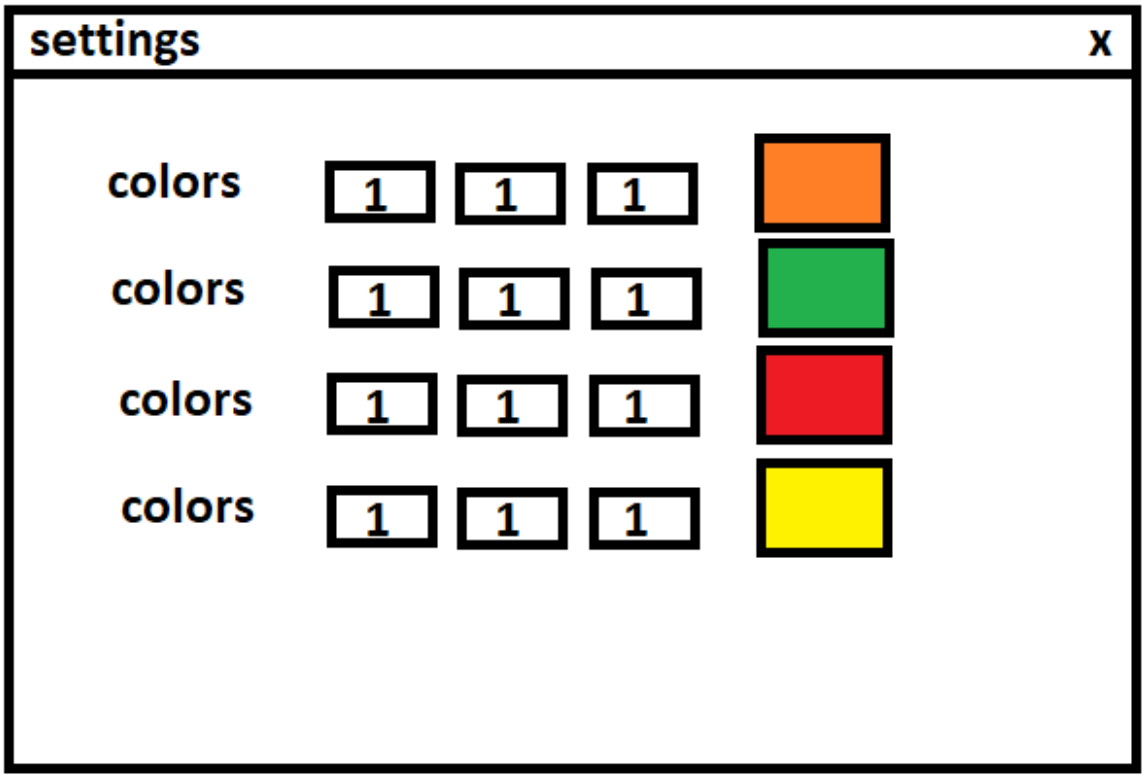
15. ábra Add Algorithm terve



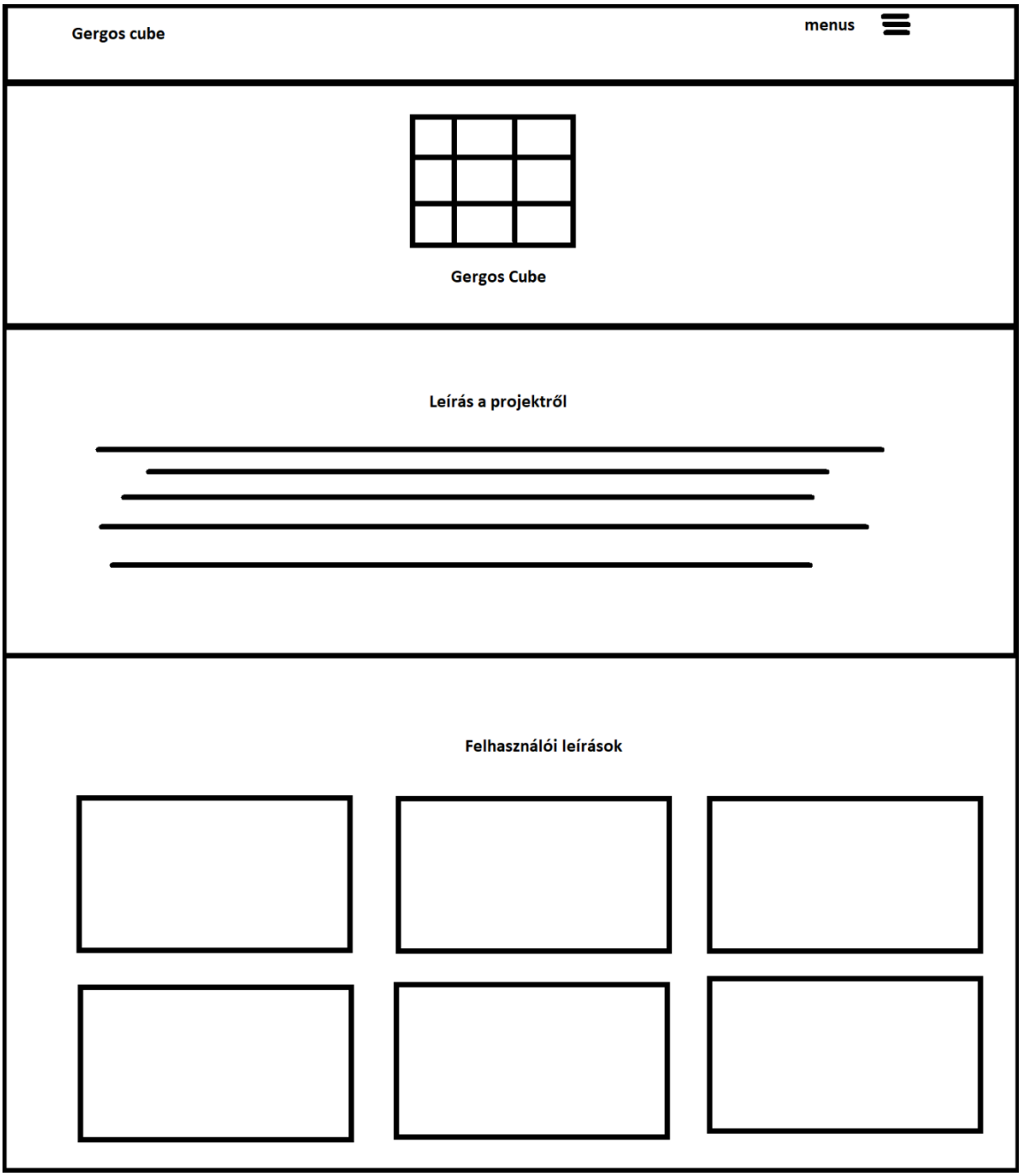
16. ábra Learn Algorithm terve



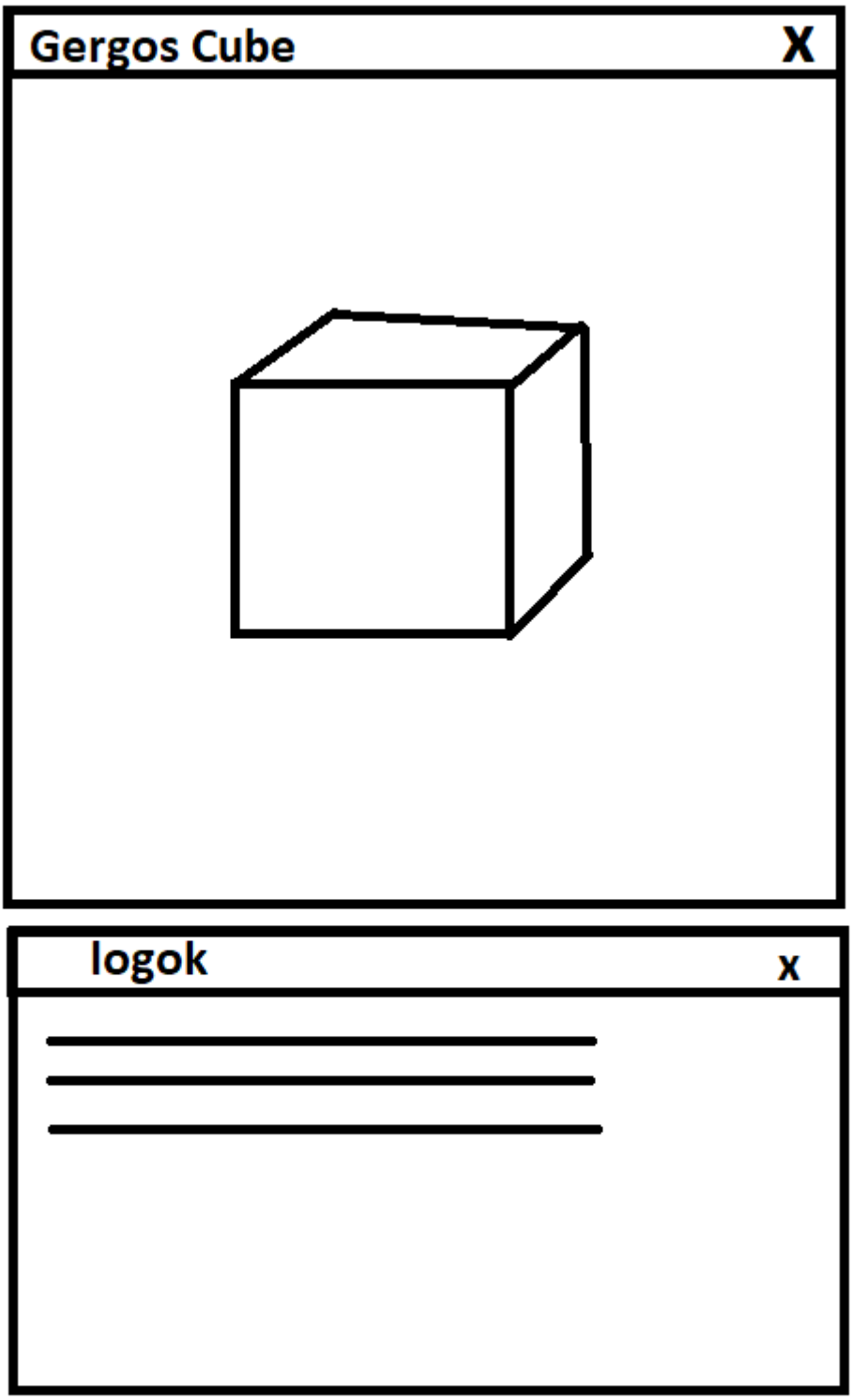
17. ábra Delete Algorithm terve



18. ábra Settings menü terve



19. ábra A Weboldal terve



20. ábra A játék megjelenésének terve

4.3.4. Adatbázis

Az alkalmazás adatbázisainak tekintett JSON fájlok az input almappában helyezkednek el. Ezeknek a felépítése egyenként eltér egymástól, viszont a kezelésük megegyezik.

4.3.4.1. algorithms.JSON

Ez a fájl egy String kulcs, és String tömb érték párokat tartalmaz. Ebben tároljuk a felhasználó által hozzáadott egyedi algoritmusokat, és a program számára szükséges lépéssorozatokat is.


```
{
  "jperm":["r","u","R","F","r","u","R","U","R","f","r","r","U","R","U"],
  "tperm":["r","u","R","U","R","f","r","r","U","R","U","r","u","R","F"],
  "yperm":["r","U","R","U","r","u","R","F","r","u","R","U","R","f","r"],
  "lperm":["l","U","R","u","L","u","u","r","U","R","u","u","r"],
  "test":["l","l","r","r"]
}
```


21. ábra Példa a algorithms.JSON fájlra

4.3.4.2. pochmanTargetPositions.JSON

Ez a fájl String kulcs és String tömb érték. Az értékként megadott tömb első eleme egy string tömb, mely a megoldáshoz nélkülözhetetlen setup algoritmusokat tartalmazzak, a második eleme pedig egy String, amely a folytatáshoz szükséges csere algoritmus nevét tárolja. Ezt a felhasználó nem tudja szerkeszteni. A setupok Szántai Szabolcs azaz a Kockások YouTube-csatorna készítőjétől származnak.


```
{
    "57": [[],"lperm"],
    "69": [[],"jperm"],
    "9": [[],"tperm"],
    "117": [[],""],

    "67": [["m"],"lperm"],
    "139": [["w", "d", "w", "d", "l"],"tperm"],
    "103": [["m"],"jperm"],
    "31": [["L"],"tperm"],
```


22. ábra Példa a pochmanTargetPositions.JSON fájlra

4.3.4.3. settings.JSON

Ez a fájl a felhasználó saját beállításának tárolásáért felelős. A kulcs mindig egy String, az érték változik a beállítástól függően.

```
"orange": [
  239,
  153,
  104
],
"red": [
  211,
  87,
  87
],
"backgroundColor": [
  200,
  200,
  200
],
"green": [
  121,
  185,
  119
],
"blue": [
  118,
  145,
  172
],
"white": [
  255,
  255,
  255
],
"stepByStepSolving": false,
"strokeWeight": 0.1,
"yellow": [
  238,
  214,
  128
],
"cameraZoomIn": 500,
"strokeColor": [
  0,
  0,
  0
],
"faceIdsCheckBox": false
```


23. ábra Példa a settings.JSON fájlra

4.4. Megvalósítás

4.4.1. Felhasznált technológiák

1. Processing könyvtár:

A könnyű és precíz kamerakezelés érdekében a Processing [1], egy PeasyCam könyvtárát használtam, melyet Jonathan Feinberg készített.

A készítőt idézve: „A mouse driven camera-control library for 3D sketches.”

2. Java könyvtár:

A program megvalósításához használt további könyvtár a Gson. A leírást idézve:

„Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.”

Fordítása:

„A Gson egy Java könyvtár, amelyet Java objektumok JSON reprezentációjává alakítására lehet használni. Ezenkívül használható JSON karakterlánc egyenértékű Java objektummá alakítására is.”

4.4.2. Eltérések a tervtől

A Processing programnyelv használata az alkalmazás fejlesztésében bizonyos kihívásokkal járt, amelyek megnehezítették munkát. Íme néhány példa a felmerült problémákra:

- **Alkalmazás bezárása:** A Processing [1] alkalmazásokban a grafikus ablak bezárásakor az egész alkalmazás automatikusan leállt. Ez problémát okozott, ha csak egy adott részt akartam bezárni, vagy ha a bezárás után még folyamatban lévő műveleteket szerettem volna végrehajtani.
- **Korlátozott könyvtárak és eszközök:** A Processing programnyelv néhány szakterületre szűkített könyvtára és eszköze lehet, hogy nem elegendő a fejlesztők számára, amikor bonyolultabb alkalmazásokat kell létrehozniuk.
- **Teljesítmény problémák:** A Processing programnyelv, mivel magas szintű és könnyen használható, néha nem biztosít optimális teljesítményt.
- **Kompatibilitás:** A Processing alapvetően Java alapú, de nem minden Java könyvtár vagy eszköz használható közvetlenül a Processingben.
- **Grafikus felhasználói felület (GUI) tervezés:** A Processing nem rendelkezik beépített eszközökkel vagy könyvtárakkal a bonyolult grafikus felhasználói felületek létrehozásához.

Összességében a Processing programnyelv jelentős előnyökkel jár a grafikus megjelenítés során, de számos kihívás is felmerülhet a fejlesztők számára.

4.5. Tesztelés

4.5.1. Egység tesztek

Ebben tesztelésre kerülnek az alábbi függvények:

- AlgorithmCollection:
  - add()
  - delete()

- OldPochmanCollection:
  - isItInAlgorithms()

- Cube:
  - checkIfEdgesAreSolved()
  - checkIfCornersAreSolved()
  - turnX()
  - turnY()
  - turnZ()

- Cubie
  - turnFacesX()
  - turnFacesY()
  - turnFacesZ()

- Face
  - turnX()
  - turnY()
  - turnZ()
  - rotateFacingX()
  - rotateFacingY()
  - rotateFacingZ()

4.5.2. Végfelhasználói tesztesetek

ID	Teszteset	
1	Alkalmazás indítása	Az alkalmazás elindul
2	Kilépés	Az alkalmazás bezárul
3	Algoritmus tanulás menü	Az algoritmus tanulás ablak megjelenik
4	Meglevő algoritmus tanulása	A „Learn Algorithm” menüpot választása után a játék elindul a választott algoritmussal
5	Új algoritmus tanulása	„Add Algorithm” menüpont választása után a felhasználó megadhatja az általa választott algoritmus adatait
6	Algoritmus mentése az adatbázisban	Az algoritmus a mentést követően az adatbázisba kerül.
7	Algoritmus törlése	„Delete Algorithm” menüpont választása után az algoritmus törlődik
8	Algoritmus törlése az adatbázisban	Az algoritmus a törlést követően kikerül az adatbázisból.
9	Önmegoldó mód	Play menüpont választása után a játék elindul a megadott beállításokkal
10	Önmegoldás	A kocka megoldja saját magát
11	Beállítások menü	„Settings” menüpont választása után a Beállítások menü megjelenik
12	Infó menüpont	„Info” menüpont választása után az információs weboldal megjelenik
13	Kocka fordítása	Valamelyik fordítási input (billentyű kombináció) lenyomása után a kocka elfordul
14	Kocka összekeverése	A keverés input (billentyű kombináció) lenyomására a kocka 10-20 lépés alatt összekeveri magát

15	Sikeres algoritmus kirakás	A felhasználó sikeresen kirakja az algoritmust, ezután tetszőleges mozgatással a kocka visszaáll alaphelyzetbe
16	Sikertelen algoritmus kirakás	A felhasználó elrontja az algoritmust ezután a kocka visszaáll alaphelyzetbe
17	Kocka automatikus kirakása	A kirakás input (billentyű kombináció) lenyomása után a kocka nekiáll kirakni magát
18	Szabad játék	Valamelyik fordítási input (billentyű kombináció) lenyomása után kocka szabadon kirakható, forgatható
19	Forgatási biztosság	Kirakás közben a felhasználó nem forgathat a kockán
20	Lépés szám	A program megfelelően számolja a lépéseket
21	Sok input	Sok input esetén a program sorba állítja a lépéseket, így nem veszik el egyik sem.

5. Továbbfejlesztési lehetőségek

- Processing lecserélése, mint grafikus megoldás.
- További metódusok implementálása, többféle kirakási módszer megtanítására.
- Több méretű kocka implementálása.
- Különböző formájú Rubik játékok implementálása, mint például a Piraminx vagy Megaminx.
- Útkereső algoritmus implementálása, a legrövidebb setupok megkeresésére.
- Mesterséges intelligencia beépítése, az önkirakás folyamatába.
- A weboldal továbbfejlesztése, hogy még több információt adjon át a kirakással kapcsolatban
- A kirakásban előre és hátra tekerés hozzáadása.
- A kocka forgatási sebességének állítására lehetőség a beállításokban.
- Valamiféle megoldás a Processing alkalmazásból való visszalépésre.
- Kocka állásának elmentése.
- A weboldalon, a leírások szekcióban az ikonokhoz valamiféle magyarázó szöveg megjelenítése, mikor rájuk mozdítja az egeret a felhasználó.
- A billentyűparancsok integrálása a programba valamilyen módon.

6. Ábrajegyzék

7. Összefoglalás

A szakdolgozatom célja egy olyan Rubik-kocka megalkotása volt, amely képes segíteni a felhasználóknak megtanulni, hogyan lehet vakon kirakni azt. A motiváció a projekt mögött az volt, hogy hobbiként egy olyan programot alkossak, amely összekapcsolja a szórakozást és a tanulást, egyúttal fejleszti a logikai gondolkodást és a problémamegoldó készségeket.

A fejlesztés alatt rengeteg alapvető tudást szereztem a Rubik-kocka mögött rejlő algoritmusokról és megoldási módszerekről. A projekt egyik fő célja a saját képességeim és ismereteim bővítése volt, miközben egy olyan eszközt alkottam, amely azoknak segíthet, akik szeretnének megismerkedni a Rubik-kocka világával.

Összességében ez a projekt egy személyes fejlődési utazás volt, amely megtanított a kitartásra, a kreativitásra és arra, hogy a tanulás és a szórakozás kéz a kézben járhat. Az alkalmazás létrehozása nemcsak az én tudásom és képességeim fejlődését tükrözi, hanem mások számára is inspirációt nyújt, akik hobbiként szeretnének új dolgokat megtanulni és alkotni. A választott téma középpontba állítja Rubik Ernő feltaláló munkásságát, s vele együtt a világ minden táján ismert és népszerű logikai játékot, a Rubik-kockát.

8. Irodalomjegyzék

1. Processing dokumentáció (online, linkelve: 2023.04.20.) https://processing.org/reference/
2. Java 17 dokumentáció (online, linkelve: 2023.04.20.) https://docs.oracle.com/en/java/javase/17/docs/api/index.html
3. GSON dokumentáció (online, linkelve: 2023.04.20.) https://github.com/google/gson
4. PeasyCam dokumentáció (online, linkelve: 2023.04.20.) https://mrfeinberg.com/peasycam/
5. Kockások YouTube-csatorna (online, linkelve: 2023.04.20.) https://www.youtube.com/@kockasok1903
6. The Coding Train YouTube-csatorna (online, linkelve: 2023.04.20.) https://www.youtube.com/@TheCodingTrain
7. A weboldal ingyenes tempalteje (online, linkelve: 2023.04.20.) https://startbootstrap.com/theme/freelancer