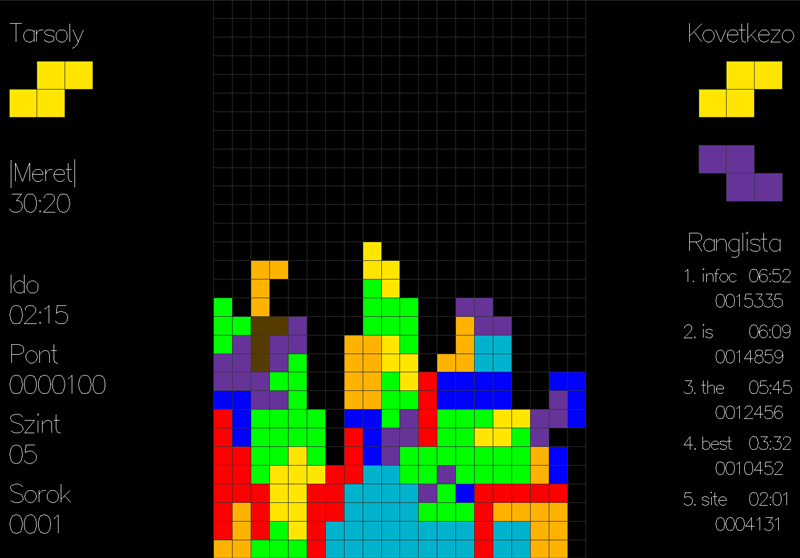
**Tetris játék**

Programozói dokumentáció

<https://github.com/Rlacko58/Tetris/tree/master/Tetris>



**Tartalomjegyzék:**

[Szükséges környezet, külső könyvtárak: 4](#_Toc531892904)

[Adatszerkezetek: 4](#_Toc531892905)

[**Pálya** ..........................................................................................................................................................................4](#_Toc531892906)

[**Hand**............ 4](#_Toc531892907)

[Játék működése, tervezési megfontolások 5](#_Toc531892908)

[Projekt felépítése 5](#_Toc531892909)

[**main**...........................................................................................................................................................................5](#_Toc531892910)

[Belső globális változók 5](#_Toc531892911)

[Függvények 5](#_Toc531892912)

[Részletesebben a **GLUT** vezérlő része a játékban: 6](#_Toc531892913)

[**map.**...........................................................................................................................................................................7](#_Toc531892914)

[Az adatszerkezetek 7](#_Toc531892915)

[Függvények 7](#_Toc531892916)

[**hand**...........................................................................................................................................................................8](#_Toc531892917)

[Az adatszerkezet 8](#_Toc531892918)

[Függvények 8](#_Toc531892919)

[**ranglista** 8](#_Toc531892920)

[Függvények 8](#_Toc531892921)

[**megjelenites** 9](#_Toc531892922)

[RGBA színkód adatszerkezete 9](#_Toc531892923)

[Függvények 9](#_Toc531892924)

[Függvények 10](#_Toc531892925)

[**MAIN** 10](file:///C:\Users\rlack\Desktop\Tetris%20végleges\Tetris3.docx#_Toc531892926)

[void Jatek(); 10](#_Toc531892928)

[void display(); 10](#_Toc531892929)

[static void Felszabadit(); 10](#_Toc531892930)

[void keyboard(unsigned char key, int x, int y); 10](#_Toc531892931)

[void specialKeys(unsigned char key, int x, int y); 10](#_Toc531892932)

[void UjJatek(int argc, char\*\* argv); 11](#_Toc531892933)

[**MAP** 11](file:///C:\Users\rlack\Desktop\Tetris%20végleges\Tetris3.docx#_Toc531892934)

[PalyaMatrix\* MatrixFoglal(Palya const \*vp); 11](#_Toc531892935)

[void MatrixInit(Palya \*vp, int const sor, int const oszlop); 11](#_Toc531892936)

[void MatrixbaMasol(Palya \*vp, Hand \*hp); 11](#_Toc531892937)

[bool Utkozes(Palya const \*vp, Hand const \*hp, bool const \*bp, int x, int y); 11](#_Toc531892938)

[int AltetrisKord(Palya const \*vp, Hand const \*hp); 11](#_Toc531892939)

[void KovTetris(Palya \*vp, Hand \*hp, bool \*vege); 12](#_Toc531892940)

[void Eltuntet\_sor(Palya \*vp, int sor); 12](#_Toc531892941)

[**HAND** 12](file:///C:\Users\rlack\Desktop\Tetris%20végleges\Tetris3.docx#_Toc531892942)

[bool\* TetroVal(int melyik); 12](#_Toc531892943)

[bool\* HandFoglal(Hand\* hp); 12](#_Toc531892944)

[void HandInit(Hand\* hp,int const \*oszlop, int const melyik); 12](#_Toc531892945)

[bool\* Forgat\_jobbra(Hand const \*hp); 13](#_Toc531892946)

[**RANGLISTA** 13](file:///C:\Users\rlack\Desktop\Tetris%20végleges\Tetris3.docx#_Toc531892947)

[static void RangCsere(Ranglista \*a, Ranglista \*b); 13](#_Toc531892948)

[void Nevhezir(Palya \*vp, int c); 13](#_Toc531892949)

[void NevbolTorol(Palya \*vp); 13](#_Toc531892950)

[void RanglistaRendez(Palya \*vp); 14](#_Toc531892951)

[void Ranglistament(Palya \*vp); 14](#_Toc531892952)

[void Ranglistabeolvas(Palya \*vp); 14](#_Toc531892953)

[**MEGJELENITES** 14](file:///C:\Users\rlack\Desktop\Tetris%20végleges\Tetris3.docx#_Toc531892954)

[void initGL(); 14](#_Toc531892955)

[void Idozito(int idokoz); 14](#_Toc531892956)

[void Ujrameretez(GLsizei width, GLsizei height); 15](#_Toc531892957)

[void KirajzInit(Palya \*vp); 15](#_Toc531892958)

[static void Szovegrajzol(char\* szoveg, GLfloat x, GLfloat y, float meret); 15](#_Toc531892959)

[static void RajzolNegyzet(GLfloat \*Nsize, GLfloat x, GLfloat y, RGBA s, bool mode); 15](#_Toc531892960)

[static void RajzolHaloba(Palya \*vp, float x, float y, int sor, int oszlop, RGBA s, bool mode); 15](#_Toc531892961)

[static RGBA SzinKonverter(int szin); 15](#_Toc531892963)

[static void RajzolMatrix(Palya \*vp); 16](#_Toc531892964)

[static void RajzolTetris(Palya \*vp, Hand \*hp, int x, bool mode); 16](#_Toc531892965)

[static void RajzolVTetris(Palya static \*vp, bool\* t, int size, float x, float y, int szin) 16](#_Toc531892966)

[static void KirajzolIdo(Ido \*t, float x, float y, float size); 16](#_Toc531892968)

[static void KirajzolPont(int \*ponts, float x, float y, float size); 16](#_Toc531892969)

[static void KirajzolSzint(int \*szints, float x, float y, float size); 16](#_Toc531892970)

[static void KirajzolEltSorSzam(Palya \*vp, float x, float y, float size); 17](#_Toc531892971)

[static void RajzolTarsoly(Palya \*vp); 17](#_Toc531892972)

[static void RajzolKovi(Palya \*vp); 17](#_Toc531892973)

[static void RajzolRangLista(Palya \*vp); 17](#_Toc531892974)

[void JatekRajzol(Palya \*vp, Hand \*hp); 17](#_Toc531892975)

[void GameOverRajzol(Palya \*vp, Hand \*hp); 17](#_Toc531892976)

# Szükséges környezet, külső könyvtárak:

* Image result for visualstudio logoA játék VisualStudio-ban íródott.

Miért? Legfőképp a praktikussága és kinézete miatt.

Sötét téma, GitHub kiegészítő és fejlesztőt segítő funkciók.

* A grafikai része OpenGl, a platformal való kommunikáció pedig GLUT.

**Fordítás**hoz szükséges lépések:

* Ha az SDK verzióba beleköt, akkor a solutionre jobb klikkelés után „retarget solution”
* Project properties átállítása megfelelő könyvtárakhoz:
  + **C/C++** fülnél a **General**-ban Additional Include Directorieshez hozzá kell adni, a játék könyvtárában lévő include mappát.
  + **Linker** fülnél a **General**-ban Additional Library Directories-hez a Tetris azon mappáját kell hozzáadni, ahol a freeglut.lib és dll van.
  + Szintén linker fülnél a **freeglut.lib**-et kell hozzáírni az Additional Dependencies-hez

# Adatszerkezetek:

## Pálya

A teljes pálya és annak adatai egy struktúrában vannak eltárolva.  
Tartalmazza a **sor** és **oszlop** számot, a **sorösszeg**-eket, mely a könnyebben kigyült sorok kezelését könnyíti meg. Ezek mellett olyan statisztikai adatokat, mint a **szint**, az **idő**, mely egy perc és másodperc-el rendelkező struktúra, a **pontszám** az **Eltüntetett sorok száma** és a játékos **neve**, ami valójában csak a játék végén kerül bekérésre.

A megjelenítéshez még megtalálható benne a **pálya szélessége** és **magassága** és a **négyzetek mérete**.

A játékban még megtalálható a későbbre elrakás funkciója, melyet a  **tarsoly** tesz lehetővé. Ez mellett pedig a **következő kettő**-t is láthatjuk. Ezek igazából csak egy egész értéket tárolnak, mely az adott tetris sorszáma.

Végül pedig a nagyobb adatszerkezet a **Pálya mátrixa**, mely egy olyan dinamikus adatszerkezet, mely azt tárolja, hogy a pálya celláinak mi a szine és, hogy van-e az adott helyen.

A **ranglista** is itt kerül letárolásra, mely adatszerkezete az adott játékos nevéből, pontszámából és játékidejéből áll.

## Hand

Az egyik legegyszerűbb adatszerkezet, mivel csak annyit tárol, hogy dinamikusan a tetris hol van a pályához képest **x** és **y** kordinátában, mi a **szine**, **sorszáma**, **színe** és jelenlegi **állása** egy logikai mátrixban tárolva.

Játék működése, tervezési megfontolások:

Maga a játéktér egy nagy mátrix, melyben adott mezők tartalmai vannak letárolva és a játék ezt folyamatosan adott feltételek szerint irányítja. A játékteret és a rajta mozgó tetrist ketté szettem épp azért, hogy könnyebb legyen a feltételeket ellenőrizni, vagy épp az irányítást elvégezni. Ez azért volt jó, mivel így csak olyan függvényekre volt szükségem például az ütközés vizsgáláshoz, hogy adott kordinátával eltolva a tetris ütközne-e, vagy hasonlók. Elforgatásnál és mozgatásnál is sokkal hatékonyabbá tette a dolgokat, ugyanis eltolás csak kordináta változtatás, a forgatás pedig csak egy pár mátrix művelet, mely a függvények résznél részletesebben ki van fejtve. Mivel így csak egy pályát kell módosítgatni, ezért az olyan adatok, mint a statisztikák sokkal egyszerűbben nyílvántarthatók. Például ha teljesül az a feltétel, hogy megtelik egy sor, szimplán eltüntetjük a pályáról a megfelelő módon, majd adott statisztikai elemeket változtatunk.

Miután megvoltak a megfelelő játék kezelő függvények a megjelenítést és a mechanikai részét kellett megvalósítani. A megjelenítés azon az elven működik, hogy meghív egy függvényt minden 10ms-onként ami szimplán belső óraként működve üzemelteti a játékot. Az irányítás szerencsére független ettől a belső órától, mivel event-ként vannak kezelve, tehát mikor megnyomunk egy gombot, akkor csak szimplán végrehajt adott függvényeket.

Mivel minden része a programnak apró függvényekre van bontva, elég hatékonyan lehet továbbfejleszteni.

Projekt felépítése:

### main

A játék függvényeit fogja össze, kezdeti inicializálásokat végzi el és a játékot működteti.  
Vannak itt olyan belső globális változók, amikre sajnos szükség van a glut miatt, ugyanis a működési elve azon alapul, hogy megadunk olyan függvényekre mutató pointereket, amiknek nincsenek paraméterei, egész pontosan ilyen például az, amit rendszeresen meghív az ablak újrarajzolásakor. Ezek a változók a fájlon kívül csak paraméterekként kerülnek átadásra.

Belső globális változók:  
static Palya t; //Pálya

static Hand h; //Kézben lévő tetris

static int GravInterv = 0; //Gravitáció időköz

static int MpCounter = 0; //Másodperc számláló

static bool valthat = true; //Tarsolyba lehet-e tenni?

static bool vege = false; //Vége?

Függvények:

void Jatek(); //Játékot vezérlő változók módosítása

void display(); //Rendszesen meghívódó függvény újrarajzolásra

static void Felszabadit(); //Mallocolt területek felszabadítása

void keyboard(unsigned char key, int x, int y); //Egyszerűbb gombok

void specialKeys(unsigned char key, int x, int y); //Speciális gombok

void UjJatek(int argc, char\*\* argv); //Ujjatek inicializáció

#### Részletesebben a **GLUT** vezérlő része a játékban:

* **g**lutInit(&argc, argv); // GLUT inicializáció
* **g**lutInitDisplayMode(GLUT\_DOUBLE); // Duplán bufferált mód
* **g**lutInitWindowSize(1000, 700); // Ablak Kezdő szélesség, magasság
* **g**lutInitWindowPosition( // Ablak pozíció

(glutGet(GLUT\_SCREEN\_WIDTH) - 1000) / 2,

(glutGet(GLUT\_SCREEN\_HEIGHT) - 700) / 2); //Középre elhelyezés

* **g**lutCreateWindow("Tetris játék"); // Ablak címe
* **g**lutDisplayFunc(display); // Újrarajzoláskor lefuttatandó függvény beállítása

# void glutDisplayFunc(void (\*func)(void));

* **g**lutReshapeFunc(Ujrameretez); // Ujraméretezéskor lefutó függvény

# void glutReshapeFunc(void (\*func)(int width, int height));

A width és a height automatikusan átadódik

* **g**lutTimerFunc(0, Idozito, 10); // Időzítő, 10-es értéket ad át az Idozito függvénynek

# void glutTimerFunc(unsigned int msecs, void (\*func)(int value), value);

Első érték, hogy mennyi idő múlva hívódjon meg, majd melyik függvény, és mit kapjon meg paraméterben az a függvény

* **g**lutKeyboardFunc(keyboard); // Alap gombok kezelése

# void glutKeyboardFunc(void (\*func)(unsigned char key, int x, int y));

key, hogy melyik gombot nyomta le, az x és y pedig, hogy hol volt az egér

* **g**lutSpecialFunc(specialKeys); // Speciális gombok

# void glutSpecialFunc(void (\*func)(int key, int x, int y));

* **i**nitGL(); // Extra inicializációhoz szükséges dolgok
* **g**lutMainLoop(); // Belépés a végtelen ciklusba
* Tehát a belső globális változókra szükség van.

### map

Itt vannak a pályához kapcsolódó függvények és típusok.

Az adatszerkezetek:

typedef struct { //Pálya

bool e; //Cellában van-e bármi?

int c; //Mi a szine?

} PalyaMatrix;

typedef struct { //Idő

int p; //Perc

int mp; //Másodperc

} Ido;

typedef struct Rang { //Rang

char \*nev; //Játékos neve

int pont; //Pontszáma

Ido time; //Ideje

} Ranglista;

typedef struct { //Pálya

int sor, oszlop; //Pálya mérete

float Nsize; //Négyzetek mérete

float width, height; //Pálya szélesség, magasság

int \*sum; //Soronkénti összeg

int level; //Jelenlegi szint

Ido time; //Eltelt idő

int pont; //Pontszám

int ElSorSzam; //Eltüntetett sorok száma

PalyaMatrix \*v; //Pálya mátrixa

int Tarsoly; //Tarsolyban lévő tetris

int KoviT[2]; //Kovetkező Tetrisek

char \*nev; //Jatekos neve

Ranglista rlista[500]; //Ranglista

} Palya;

* Függvények:

//Memoria terulet lefoglalasa a pályának

PalyaMatrix\* MatrixFoglal(Palya const \*vp);

//A pálya inicializálása

void MatrixInit(Palya \*vp, int const sor, int const oszlop);

//Tetris átmásolása a pályára

void MatrixbaMasol(Palya \*vp, Hand \*hp);

//Ütközés vizsgálat x és y eltolással bp tömbre

bool Utkozes(Palya const \*vp, Hand const \*hp, bool const \*bp, int x, int y);

//Alsó tetris távolsága a jelenlegitől

int AltetrisKord(Palya const \*vp, Hand const \*hp);

//Következő tetrisre állítás

void KovTetris(Palya \*vp, Hand \*hp, bool \*vege);

//Adott sor eltüntetése, majd fölötte lévők lejebb húzása

void Eltuntet\_sor(Palya \*vp, int sor);

### hand

A kézben lévő tetrishez kapcsolódó függvények és struktúra.

Az adatszerkezet:

typedef struct { //Hand

int x, y; //Pozicioja a Maphez képest

int color; //Szine

int melyik; //Tetris tipusa

int size; //Mátrix mérete (size\*size)

bool\* v; //Tartalma

} Hand;

Belső globális változók, melyek a játékhoz kapcsolódó tetriseket tartalmazzák.



//A tetrisek lehetséges verziói

static bool I[16] = { 0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0 }; //I

static bool T[9] = { 1,1,1,0,1,0,0,0,0 }; //T

static bool L[9] = { 1,1,1,1,0,0,0,0,0 }; //L

static bool J[9] = { 1,1,1,0,0,1,0,0,0 }; //J

static bool S[9] = { 0,1,1,1,1,0,0,0,0 }; //S

static bool Z[9] = { 1,1,0,0,1,1,0,0,0 }; //Z

static bool O[4] = { 1,1,1,1 }; //O

* Függvények:

//Kiválasztott tetrist adja vissza (belső globális változók)

bool\* TetroVal(int melyik);

//Lefoglalja a tetris mátrixát

bool\* HandFoglal(Hand\* hp);

//Kézben lévő tetris inicializáció

void HandInit(Hand\* hp,int const \*oszlop, int const melyik);

//Elforgatja a kézben lévő tetrist és visszatér annak mátrixával

bool\* Forgat\_jobbra(Hand const \*hp);

### ranglista

Ranglistával kapcsolatos függvényeket tartalmazza. A szükséges adatszerkezetek a **hand** fájlban vannak, ugyanis a pálya struktúrájába kerül letárolásra a ranglista.

* Függvények:

//Ranglista két elemének cserélése

static void RangCsere(Ranglista \*a, Ranglista \*b);

//Játékos nevéhez karakter hozzáfűzése

void Nevhezir(Palya \*vp, int c);

//Játékos nevéből karakter törlése

void NevbolTorol(Palya \*vp);

//Ranglista rendezése

void RanglistaRendez(Palya \*vp);

//Ranglista mentése

void Ranglistament(Palya \*vp);

//Ranglistához hozzáadás

void Ranglistahozaad(Palya \*vp);

//Ranglista feltöltése fájlból

void Ranglistabeolvas(Palya \*vp);

Benne: typedef enum FInp {nev, score, perc, masodperc, vegeell} FInp;

### megjelenites

Az ablakra rajzoláshoz szükséges függvényeket tartalmazza.

* RGBA színkód adatszerkezete:

typedef struct { //RGBA

float r; //red

float g; //green

float b; //blue

float a; //alpha

} RGBA;

* Függvények:

//OpenGL Grafikához szükséges inicializációk

void initGL();

//Játék belső időzítője

void Idozito(int idokoz);

// Handler - windows re-size :

Mikor az ablakot átméretezik, az arányok nem romlanak el.

void Ujrameretez(GLsizei width, GLsizei height);

// Ez a függvény, az alábbi oldal példája alapján készült:

// <http://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html>

// Kis matek a megfelelő kirajzolásokhoz

void KirajzInit(Palya \*vp);

//Szöveg kirajzolása x,y kordinátára adott mérettel

static void Szovegrajzol(char\* szoveg, GLfloat x, GLfloat y, float meret);

//Négyzet rajzolása Nsize mérettel x,y kordinátára s színnel

//Egyik mód a normál tetris rajzolása, másik mikor halványan kell

static void RajzolNegyzet(GLfloat \*Nsize, GLfloat x, GLfloat y, RGBA s, bool mode);

//x, y kordináta szerint egy Nsize méretű négyzetrácsos hálóba rajzolás

static void RajzolHaloba(Palya \*vp, float x, float y, int sor, int oszlop,

RGBA s, bool mode);

//Visszatér adott szám alapján a megfelelő színkóddal

static RGBA SzinKonverter(int szin);

//Pályamátrix kirajzolása

static void RajzolMatrix(Palya \*vp);

//Adott kézben lévő tetris kirajzolása

static void RajzolTetris(Palya \*vp, Hand \*hp, int x, bool mode);

//Adott bool tömbben lévő tetris kirajzolása

static void RajzolVTetris(Palya static \*vp, bool\* t, int size, float x, float y,

int szin)

//Idő kirajzolása

static void KirajzolIdo(Ido \*t, float x, float y, float size);

//Pontszám kirajzolása

static void KirajzolPont(int \*ponts, float x, float y, float size);

//Szint kirajzolása

static void KirajzolSzint(int \*szints, float x, float y, float size);

//Eltüntetett sorok számának kirajzolása

static void KirajzolEltSorSzam(Palya \*vp, float x, float y, float size);

//Tarsolyban lévő tetris kirajzolása

static void RajzolTarsoly(Palya \*vp);

//Következő tetrisek kirajzolása

static void RajzolKovi(Palya \*vp);

//Ranglista megjelenítés

static void RajzolRangLista(Palya \*vp);

//Játék közbeni Kirajzolás

void JatekRajzol(Palya \*vp, Hand \*hp);

//Játék vége kirajzolás

void GameOverRajzol(Palya \*vp, Hand \*hp);

Függvények:

##### MAIN

##### --

//Játékot vezérlő változók módosítása

###### **void Jatek();**

*Feladata*: A belső játékvezérlő változók módosítása a megfelelő feltételek alapján

* + - Gravitációs intervallum, Időkijelző átváltások, Szint növelése

*Paraméterek*: Nincsenek

*Visszatérés*: Nincs

//Rendszesen meghívódó függvény újrarajzolásra

###### **void display();**

*Feladata*: A játék megfelelő állása alapján való kirajzolási függvény kiválasztása

* + - Játékkirajzolás vagy Játék vége kirajzolás, Változók növelése (Mp,Grav)

*Paraméterek*: Nincsenek

*Visszatérés*: Nincs

//Mallocolt területek felszabadítása

###### **static void Felszabadit();**

*Feladata*: Területek felszabadítása  
- Ranglistában lévő nevek, pályamátrix, hand mátrix

*Paraméterek*: Nincsenek

*Visszatérés*: Nincs

//Egyszerűbb gombok

###### **void keyboard(unsigned char key, int x, int y);**

*Feladata*: A billentyű gombnyomás alapján a megfelelő műveletek elvégzése

* + - **(27) ESC** – kilépés  **(32) Szóköz** – Tetris lerakása **C –** Tarsolyba rakás
    - *Játék végén:* **Betűk –** névhez írás **(8) Törlés** **(13) Enter** - Mentés

*Paraméterek*:

* + - **Key**: Leütött gomb száma
    - **x, y**:Egér helyzete, mikor érkezett gomb nyomás

*Visszatérés*: Nincs

//Speciális gombok

###### **void specialKeys(unsigned char key, int x, int y);**

*Feladata*: A billentyű speciális gombnyomás alapján a megfelelő műveletek elvégzése

* + - **Fel nyíl**: Forgatás **Le nyíl**: Mozgatás lefele  **Oldal nyilak:** Mozgatás oldalra

*Paraméterek*:

* + - **Key**: Leütött gomb száma
    - **x, y**:Egér helyzete, mikor érkezett gomb nyomás

*Visszatérés*: Nincs

//Ujjatek inicializáció

###### **void UjJatek(int argc, char\*\* argv);**

*Feladata*: Kezdő inicializációk, mint pálya beállítása, vagy a megfelelő eventekhez a megfelelő függvények hozzá társítása.

*Paraméterek*: argc és \*\*argv, melyet az int main-ből kap és a GLUT-nak van szüksége rá

*Visszatérés*: Nincs

##### MAP

//Memoria terulet lefoglalasa a pályának

###### **PalyaMatrix**\* **MatrixFoglal**(Palya const \***vp**);

*Feladata*: Lefoglal a pálya mérete alapján egy sor\*oszlop Pályamátrix méretű területet

*Paraméterek*: Pálya struktúra

*Visszatérés*: Lefoglalt területre mutató pointer

//A pálya inicializálása

###### **void MatrixInit(Palya \*vp, int const sor, int const oszlop);**

*Feladata*: A pálya struktúra kezdeti értékeit adja meg sor és oszlop szám alapján.

* + - Nulázza amit kell, beállítja a soron következő tetriseket, szintet 1-re álíltja, lefoglalja a sumot és nevet, lenullázza a mátrixot.

*Paraméterek*: Pálya struktúra, melyet változót cím szerint kap meg. Sor és oszlop szám.

*Visszatérés*: Nincs

//Tetris átmásolása a pályára

###### **void MatrixbaMasol(Palya \*vp, Hand \*hp);**

*Feladata*: A pálya struktúra mátrixára bemásolja a Hand struktúra mátrixát, emellett ellenőrzi ha megtelik egy sor és arra meghívja az eltüntető függvényt.

*Paraméterek*: Pálya struktúra és Hand struktúra

*Visszatérés*: Nincs

//Ütközés vizsgálat x és y eltolással bp tömbre

###### **bool Utkozes(Palya const \*vp, Hand const \*hp, bool const \*bp, int x, int y);**

*Feladata*: Ha a pályára adott méretű bool tömb adott x és y kordinátára másolunk ütközés történne-e

*Paraméterek*: Pálya, Hand (a méret miatt), a bool mátrixunk és az x és y kordináta ahol van.

*Visszatérés*: Igaz, ha van ütközés, Hamis ha nincs.

Figyelni kell, hogy a függvény meghívása előtti bool mátrixot ne felejtsük el felszabadítani ha kell.

//Alsó tetris távolsága a jelenlegitől

###### **int AltetrisKord(Palya const \*vp, Hand const \*hp);**

*Feladata*: Mi a maximum, amíg lejebb lehet tolni a tetrist?

*Paraméterek*: A pálya és a Hand struktúra.

*Visszatérés*: Eltolás egész értékben.

//Következő tetrisre állítás

###### **void KovTetris(Palya \*vp, Hand \*hp, bool \*vege);**

*Feladata*: A kézben lévő tetrist kicseréli a sorban következőre és egy újat generál a sorba.

*Paraméterek*: A pálya mátrix, hogy tudja mik vannak a sorban, a Hand, hogy tudja hova másolni és a vége bool változó, ami azért felelős, hogy vizsgálja ha bemásoláskor már ütközés lenne, tehát nem tudja bemásolni, azaz vége a játéknak.

*Visszatérés*: Nincs.

//Adott sor eltüntetése, majd fölötte lévők lejebb húzása

###### **void Eltuntet\_sor(Palya \*vp, int sor);**

*Feladata*: A kiválasztott sor eltüntetése a pályáról

*Paraméterek*: A pálya és az adott sor egész értékben.

*Visszatérés*: Nincs.

##### HAND

//Kiválasztott tetrist adja vissza (belső globális változók)

###### **bool\* TetroVal(int melyik);**

*Feladata*: Megfelelő sorszámú tetris-szel való visszatérés.

*Paraméterek*: A sorszám, amely egész.

*Visszatérés*: A tömbre mutató pointer.

//Lefoglalja a tetris mátrixát

###### **bool\* HandFoglal(Hand\* hp);**

*Feladata*: A kézben lévő tetris méretéhez megfelelő mátrix lefoglalása.

*Paraméterek*: A Hand struktúra.

*Visszatérés*: A lefoglalt területre mutató bool pointer.

//Kézben lévő tetris inicializáció

###### **void HandInit(Hand\* hp,int const \*oszlop, int const melyik);**

*Feladata*: A Hand struktúra kezdeti értékeinek beállítása. A pályához képesti pozíciója az oszlop szám alapján és a tetris kiválasztása a melyik egész alapján.

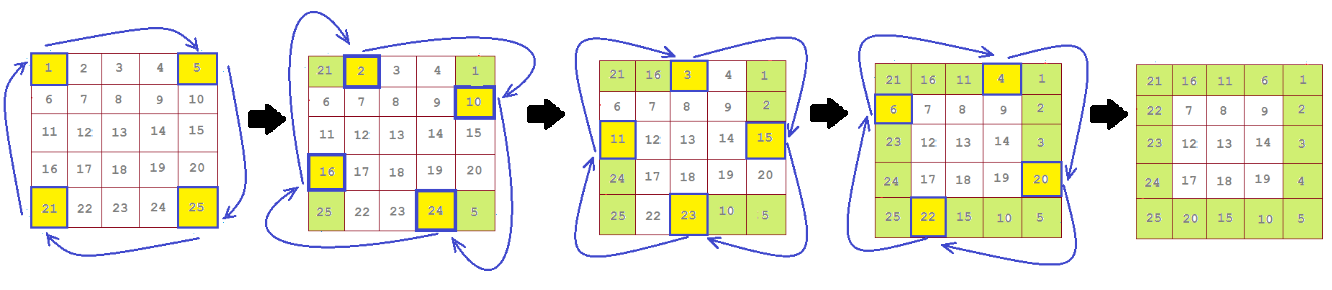
*Paraméterek*: Hand struktúra, az oszlopszámra mutató pointer és a melyik egész.

*Visszatérés*: Nincs.

//Elforgatja a kézben lévő tetrist és visszatér annak mátrixával

###### **bool\* Forgat\_jobbra(Hand const \*hp);**

*Feladata*:



Mikor 90°-al forgatunk egy n\*n-es mátrixot, valójában a fenti ábra szerint pakolunk át elemeket, melyek indexét ha felírjuk, akkor megkapjuk a fenti megoldást, vagyis:

Segéd\_Tömb [j] [Méret - 1 - i] = Tömb [i] [j];

Balra forgatáskor, pont ugyanez történne, csak a két a két kordináta helyet cserélne.

Segéd\_Tömb [j] [i] = Tömb [i] [Méret - 1 - j];

*Paraméterek*: A Hand struktúra.

*Visszatérés*: Az elforgatott bool mátrix.

##### RANGLISTA

//Ranglista két elemének cserélése

###### **static void RangCsere(Ranglista \*a, Ranglista \*b);**

*Feladata*: A ranglista struktúra tömb két elemét cseréli fel. Rendezéshez van rá szükség

*Paraméterek*: Ranglista egyik és másik elemére mutató pointer.

*Visszatérés*: Nincs.

//Játékos nevéhez karakter hozzáfűzése

###### **void Nevhezir(Palya \*vp, int c);**

*Feladata*: A játékos nevéhez egy karaktert hozzáfűz.

*Paraméterek*: Pálya struktúra, melyben a név van és a karakter integer értéke.

*Visszatérés*: Nincs.

//Játékos nevéből karakter törlése

###### **void NevbolTorol(Palya \*vp);**

*Feladata*: A játékos nevéből törli az utolsó karaktert.

*Paraméterek*: A pálya struktúra, melyben a név szerepel.

*Visszatérés*: Nincs.

//Ranglista rendezése

###### **void RanglistaRendez(Palya \*vp);**

*Feladata*: Ranglista rendezése egyszerű szélső érték kereséses rendezéssel. Rendezés után automatikusan mentésre kerül a fájlba a Ranglistament függvény által.

*Paraméterek*: Pálya struktúra.

*Visszatérés*: Nincs.

//Ranglista mentése

###### **void Ranglistament(Palya \*vp);**

*Feladata*: A ranglista fájlba írása.

*Paraméterek*: Pálya struktúra

*Visszatérés*: Nincs

//Ranglistához hozzáadás

**void Ranglistahozaad(Palya \*vp);**

*Feladata*: Ranglista végére egy új rekord felvétele.

*Paraméterek*: Pálya struktúra.

*Visszatérés*:Nincs.

//Ranglista feltöltése fájlból

###### **void Ranglistabeolvas(Palya \*vp);**

*Feladata*: Ranglista beolvasása fájlból. Állapotgépes módszerrel van megvalósítva, karakterenként beolvasva a fájlból. Lehetett volna sokkal egyszerűbben is, de akartam állapotgépet is a programba.

typedef enum FInp {nev, score, perc, masodperc, vegeell} FInp;

*Paraméterek*: Pálya struktúra.

*Visszatérés*: Nincs.

A fájlnak vagy üresnek kell lennie, vagy megfelelően formázott adatokkal feltöltöttnek.

##### MEGJELENITES

// <http://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html>

A GLUT és OpenGL megértéséhez a fenti oldal segített.

Az ujraméretező függvény az oldalon lévő példája alapján készült.

//OpenGL Grafikához szükséges inicializációk

###### **void initGL();**

*Feladata*: OpenGl-hez szükséges alap dolgok beállítása, ez esetben a törlő szín.

*Paraméterek*: Nincs.

*Visszatérés*: Nincs.

//Játék belső időzítője

###### **void Idozito(int idokoz);**

*Feladata*: Az a függvény, melyet a glutTimerFunc hív meg rendszeresen. A függvény meghiváskor újrarajzolandónak jelöli az ablak tartalmát.

*Paraméterek*: idokoz, melyet megkap a TimerFunc-on keresztül. Lehetne még egy paraméter is, de arra jelen esetben nincs szükség.

*Visszatérés*: Nincs

// Handler - windows re-size :

Mikor az ablakot átméretezik, az arányok nem romlanak el.

###### **void Ujrameretez(GLsizei width, GLsizei height);**

*Feladata*: A kirajzolt dolgok újraméretezése az ablak mérethez igazítva.

*Paraméterek*: Két egész tipusú változó, melyek az ablak szélessége és magassága.

*Visszatérés*: Nincs.

// Kis matek a megfelelő kirajzolásokhoz

###### **void KirajzInit(Palya \*vp);**

*Feladata*: A pálya méretekkel kapcsolatos változók kiszámolása és lementése.

*Paraméterek*: Pálya struktúra.

*Visszatérés*: Nincs.

//Szöveg kirajzolása x,y kordinátára adott mérettel

###### **static void Szovegrajzol(char\* szoveg, GLfloat x, GLfloat y, float meret);**

*Feladata*: x és y kordinátára egy szöveg kirajzolása adott mérettel

*Paraméterek*: Egy szövegre mutató pointer, x és y egész kordináta és egy float méret

*Visszatérés*: Nincs.

//Négyzet rajzolása Nsize mérettel x,y kordinátára s színnel

//Egyik mód a normál tetris rajzolása, másik mikor halványan kell

###### **static void RajzolNegyzet(GLfloat \*Nsize, GLfloat x, GLfloat y, RGBA s, bool mode);**

*Feladata*: Négyzet kirajzolása x és y kordinátára adott színnel. Két módja közt lehet váltogatni, a különbség az árnyalatban van, ugyanis a fenti és lenti tetris kirajzolása is ennek segítségével történik.

*Paraméterek*: A négyzetek mérete, x és y kordináta, a szín RGBA struktúrában és végül, hogy milyen módban működjön a függvény.

*Visszatérés*: Nincs.

//x, y kordináta szerint egy Nsize méretű négyzetrácsos hálóba rajzolás

###### **static void RajzolHaloba(Palya \*vp, float x, float y, int sor, int oszlop,**

###### **RGBA s, bool mode);**

*Feladata*: Kirajzolja egy négyzetet adott kordinátára sor és oszlop szám mellett a hálóra.

Lényegi része, hogy így nem kell minden kordinátát kiszámolni, ugyanis elég megadni a bal felsőt és onnan kiszámolja x és y kordinátát, majd oda rajzol.

*Paraméterek*: A pálya mátrixa, az x és y kordináta, a sor és oszlop szám és a mode.

*Visszatérés*: Nincs.

//Visszatér adott szám alapján a megfelelő színkóddal

###### **static RGBA SzinKonverter(int szin);**

*Feladata*: Adott számot konvertál a megfelelő szinre.

*Paraméterek*: Egy egész szám.

*Visszatérés*: RGBA struktúra típusú változó.

//Pályamátrix kirajzolása

###### **static void RajzolMatrix(Palya \*vp);**

*Feladata*: Végig fut a pályamátrixon és annak elemeit kirajzolja a RajzolHaloba segítségével.

*Paraméterek*: Pályamátrix.

*Visszatérés*: Nincs.

//Adott kézben lévő tetris kirajzolása

###### **static void RajzolTetris(Palya \*vp, Hand \*hp, int x, bool mode);**

*Feladata*: Adott eltolással kirajzol egy tetrist. Ez azért jó, mert így elég ez az egy függvény az al és normál tetris kirajzolásához.

*Paraméterek*: Pálya és Hand struktúra, az eltolás egész értékként és mód.

*Visszatérés*: Nincs.

//Adott bool tömbben lévő tetris kirajzolása

###### **static void RajzolVTetris(Palya static \*vp, bool\* t, int size, float x, float y,**

###### **int szin)**

*Feladata*: Hasonló a RajzolTetrishez, de ez egy adott kordinátára rajzol ki egy bool tömbben lévő tetrist. Ez felelős azért hogy a tarsolyt és a következő tetriseket megjelenítsük.

*Paraméterek*: A pályamátrix, a bool mátrix benne a tettris-szel, annak mérete, x, y kordinátája és szine.

*Visszatérés*: Nincs.

Talán kicsit sok a paramétere, későbbi fejlesztés során akár átlehetne rendezni ezeket egy saját struktúrába.

//Idő kirajzolása

###### **static void KirajzolIdo(Ido \*t, float x, float y, float size);**

*Feladata*: Ido megfelelő formátumban való kirajzolása x és y kordinátára.

Valójában ez az a függvény, ami a pálya méretét is kirajzolja.

*Paraméterek*: Az idő struktúrája, x és y kordináta és a szöveg mérete.

*Visszatérés*: Nincs.

//Pontszám kirajzolása

###### **static void KirajzolPont(int \*ponts, float x, float y, float size);**

*Feladata*: Pontszám kirajzolása adott kordinátára.

*Paraméterek*: A pontra mutató változó, x, y kordináta és szövegméret.

*Visszatérés*: Nincs.

//Szint kirajzolása

###### **static void KirajzolSzint(int \*szints, float x, float y, float size);**

*Feladata*: A szint kirajzolása, ahol jár a játékos.

*Paraméterek*: A szintre mutató változó, x, y kordináta és szövegméret.

*Visszatérés*: Nincs.

//Eltüntetett sorok számának kirajzolása

###### **static void KirajzolEltSorSzam(Palya \*vp, float x, float y, float size);**

*Feladata*: Kirajzolja a játék során eltüntetett sorok számát.

*Paraméterek*: Pálya struktúra, x, y kordináta és szövegméret.

*Visszatérés*: Nincs.

//Tarsolyban lévő tetris kirajzolása

###### **static void RajzolTarsoly(Palya \*vp);**

*Feladata*: A tarsoly kirajzolása.

*Paraméterek*: Pálya struktúra melyben benne van, hogy mi van a tarsolyban.

*Visszatérés*: Nincs.

//Következő tetrisek kirajzolása

###### **static void RajzolKovi(Palya \*vp);**

*Feladata*: Következő tetrisek kirajzolása.

*Paraméterek*: Pálya struktúra, melyben benne vannak.

*Visszatérés*: Nincs.

//Ranglista megjelenítés

###### **static void RajzolRangLista(Palya \*vp);**

*Feladata*: A ranglista első 5 elemének kirajzolása.

*Paraméterek*: A pálya struktúra.

*Visszatérés*: Nincs.

//Játék közbeni Kirajzolás

###### **void JatekRajzol(Palya \*vp, Hand \*hp);**

*Feladata*: A fenti függvényekből összeállítja a kirajzolást a megfelelő sablon szerint.

*Paraméterek*: A pálya és a Hand struktúra.

*Visszatérés*: Nincs.

//Játék vége kirajzolás

###### **void GameOverRajzol(Palya \*vp, Hand \*hp);**

*Feladata*: Ez esetben a játék végét rajzolja ki, tehát azt, mikor már csak nevet kér be a játék

*Paraméterek*: Pálya és Hand struktúra.

*Visszatérés*: Nincs.