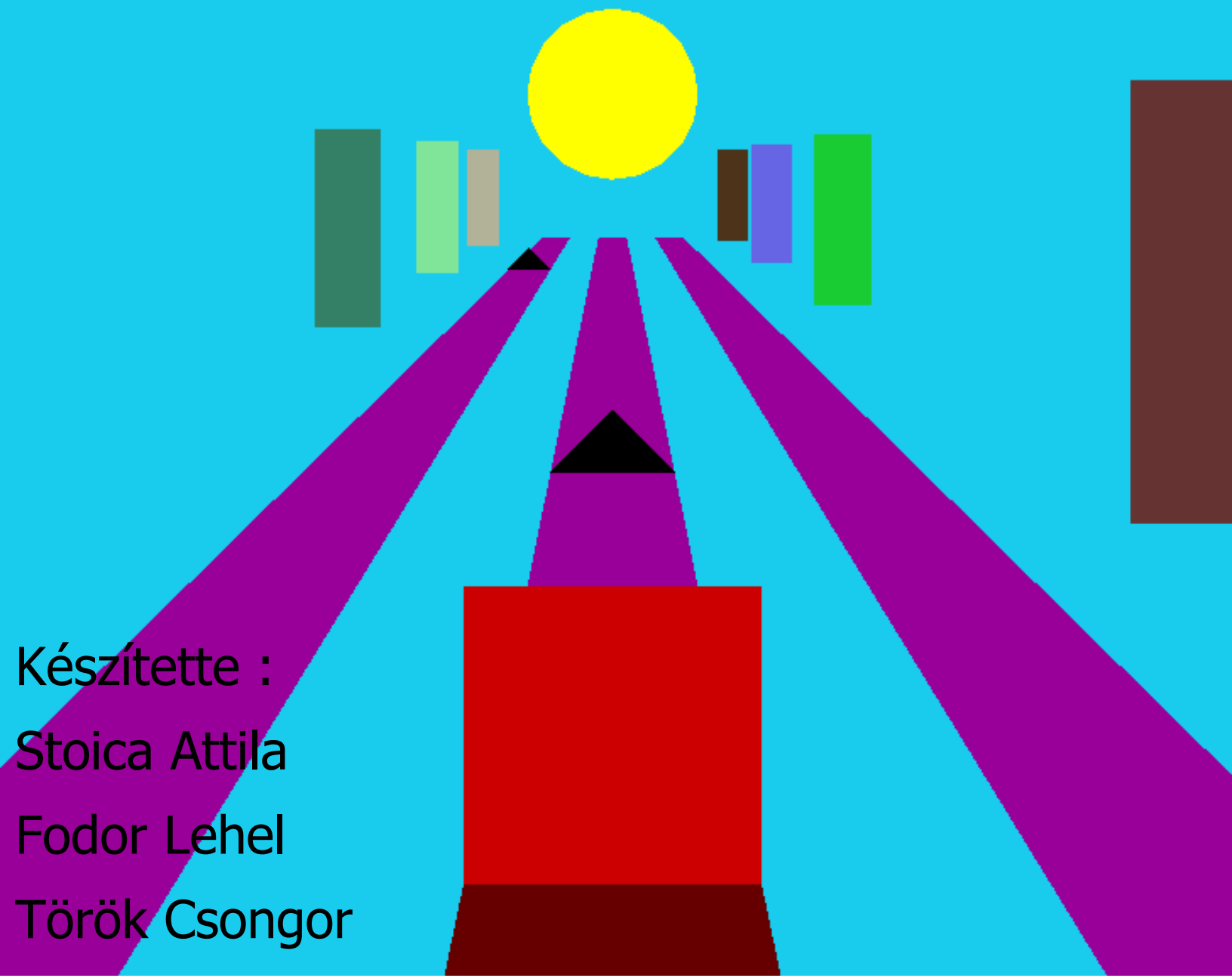
# Squares Dash



Készítette :
Stoica Attila
Fodor Lehel
Török Csongor

# Tartalomjegyzék

# Bevezető

A mai világban egyre több ember felejt el kikapcsolódni, a mindennapok sietsége,a sok munka miatt. Ezáltal az emberek sokkal stresszesebbek, fáradékonyabbak és idegesebbek lesznek ami mentális illetve testi egészségükre is meglehetősen kihathat.

Ennek fényében elgondolkodtunk, miként tudnánk orvosolni ezt a problémát. Manapság annak ellenére hogy sok képernyő vesz minket körbe, függetlenül attól, hogy munkában, szakmai életünkben is ezeket figyeljük, szabadidőnkben se idegenkedünk el tőlük, évkortól függetlenül. Egyre felkapottabbak a videójátékok a fiatalság körében főként, ezért gondoltuk hogy érdemes erre nagyobb hangsúlyt fektetnünk.

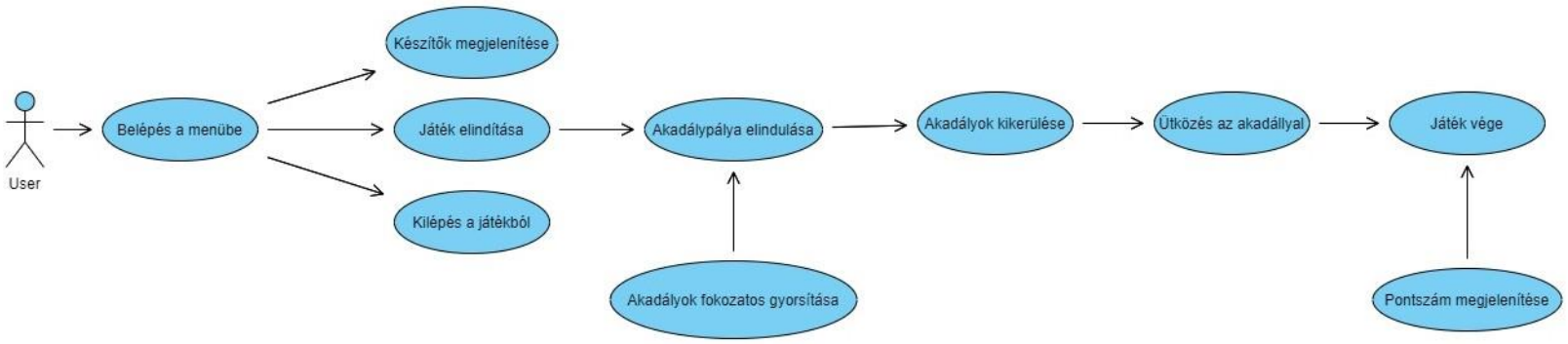
Arra jutottunk, hogy egy egyszerű, kézenfekvő, bárki számára érthető, szórakoztató játék révén ezt megvalósíthatjuk. Ezeket a szempontokat figyelembe véve, igyekeztünk hogy játékunk magas számítógépes ismeretek nélkül élvezhető legyen.

# Célkitűzések

Szerettünk volna egy ügyességi játékot létrehozni, amely bárki számára, tutorialok nélkül érthető és élvezhető lehet.

- Három dimenziós játék létrehozása
- Minimalisztikus, letisztult design a játékban
- Egyértelmű, felhasználó-barát GUI
- Akadálypálya véletlenszerű generálása, megtervezése
- Pontszám kiértékelő rendszer
- Fokozatos nehézségszint változtatás
- Adatbázishoz való összekötés játékos adatai, eredményeivel
- Textúrák hozzáadása
- Hangeffektek hozzáadása
- Platformfüggetlenítés (Console App,Webes felület, Android, IOS)

# Use Case Diagram




*Ábra 1.*

Szavakban :

- Játék elindítása után megjelenik a menü
- Három opció közül válogathatunk
  - Játék elindítása ( Start )
  - Készítők megjelenítése ( Credits )
  - Kilépés ( Exit )
- Exit gomb megnyomásakor lezárja a programot
- Credit gomb esetén megjelenik egy új ablak a készítők neveivel
- Start gomb lenyomása után egy újabb ablakban jelenik meg a játék
- A játék elindítása után megjelennek az út sávok, amelyeken a háromszög akadályok generálódnak
- Sávokkal együtt megjelennek az út két oldalán téglalapok, illetve az ablak közepén egy kör

- Lehetőve válik a négyzet karakterünk mozgatása
    - Három sávon tudunk navigálni
    - Balra, “A” gombbal
    - Jobbra, “D” gombbal léphetünk
- Ezek az akadályok fokozatosan gyorsulnak
- Újabb nehézségi szinttel párhuzamosan növekedik a pontszámok gyűlése is
- Abban a pillanatban amikor ugyanabban a sávban található a négyzetünk az akadállyal, megáll a játék
- Egy új ablakban megjelenítődik az elért pontszám

# Rendszerkövetelmények

## Funckionális követelmények

Mivel játékunk egyelőre csak konzolos felületen van jelen, ezért egy billentyűzet elengedhetetlen a játék irányitásához, illetve egy egér a menüben való navigáláshoz.

Irányító gombok :

- A gomb : balra lép a négyzet
- D gomb : jobbra lép a négyzet

Menüből elérhető gombok :

- Start : játék kezdete
- Credits : kivitelezők neveinek megjelenítése
- Exit : játék lezárása

Játék végének elérésekor a “Game Over” ablakon az OK gomb lenyomásával lezárjuk a programot

# Nem-funckionális követelmények

Fő software-ek :

- Qt Creator 4.14.0 ( Community )
- SQL Developer 19.2.1

Programozási nyelvek :

- C++
  - Qt könyvtárai
    - Qlabel
    - QPushButton
    - QVBoxLayout
    - QApplication
    - QMessageBox
    - QMainWindow
    - QOpenGLWindow
    - QsurfaceFormat
    - QopenGL
    - QKeyEvent
  - OpenGL
    - glu.h
- SQL

Programozási paradigmák :

- Objektum orientált az osztályoknak köszönhetően
- Esemény orientált a connect-nek köszönhetően

Rendszerkövetelmények :

Minimális rendszerkövetelménynek a Qt Creator igényeihez :

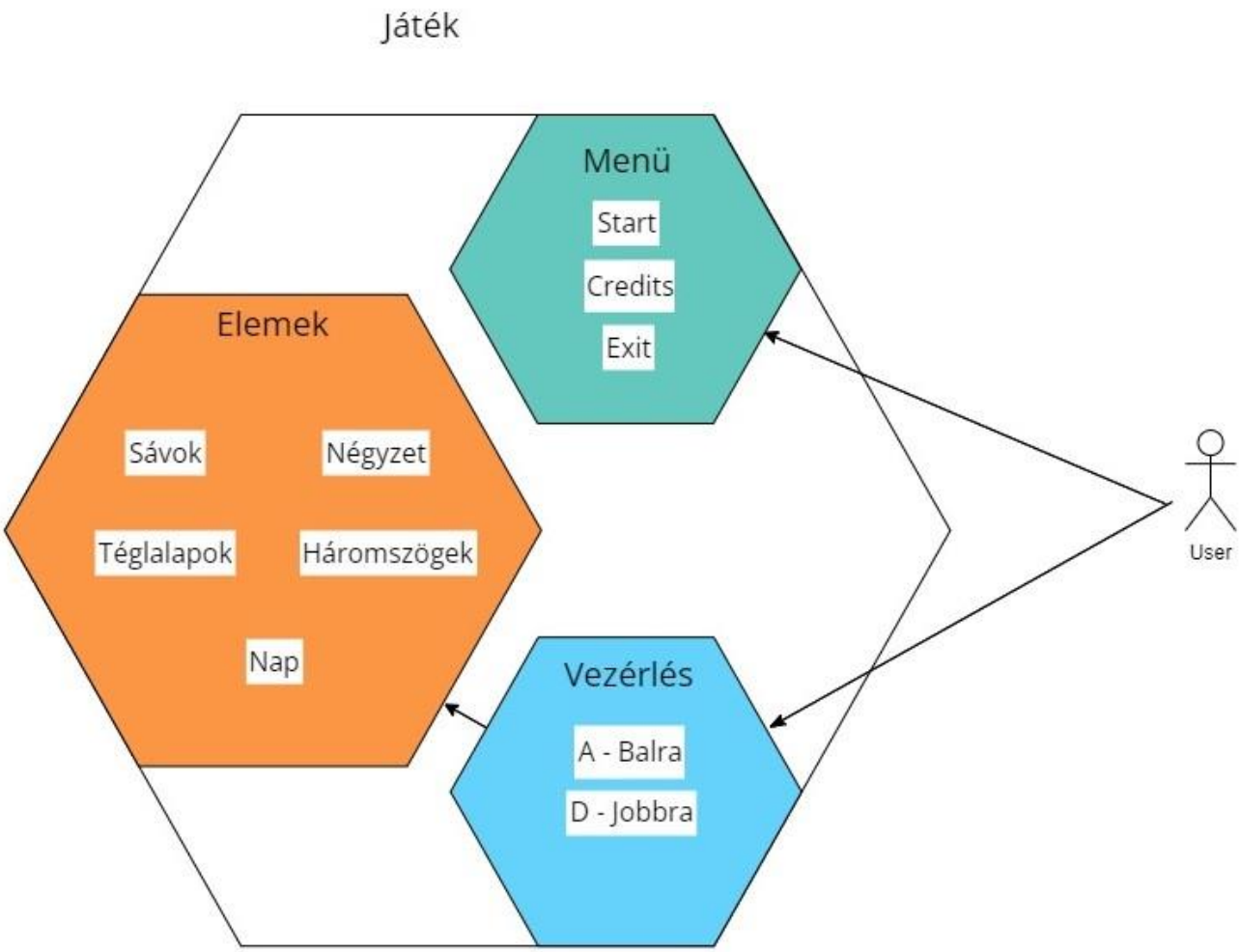
- 10.5 MB tárhely
- 1 GHz , 500 MHz CPU
- 256 MB RAM
- OpenGL ES 2.0 Support

Operációs rendszerek :

- Windows 7, 32/64 bit
- Windows 10, 32/64 bit
- Ubuntu Linux 64 bit 16.04 LTS

# Tervezés

## Architektúra diagram




*Ábra 2*

- Menü :
    - Tartalmazza az alapvető funkciókat, amelyeket már a funkcionális követelményeknél említettünk

- Vezérélés :
    - Az egeren kivüli való vezérlés mellett van a két input gombunk
        - A , amivel bal irányba mozdíthatjuk karakterünket
        - D, amivel jobb irányba léphet karakterünk
- Elemek :
    - A fő karakterünk a piros négyzet, amelyhez hozzá tartozik egy sötétebb árnyék
    - Fő ellenfelek, vagyis akadályok, a fekete háromszögek, melyek véletlenszerűen érkezhetnek a három sávon
    - Három lila sáv, amely magát a játékteret képezi
    - Bal illetve jobb odalon változó szinű téglalapok közelednek az egyébként statikus karakterünkhöz, a mozgás illuzióját keltve
    - A középpontban a kör a napot jelképezi 😊

## Verziókövetés

Munkafolyamatunk követése érdekében a GitHub-ot használtuk, management kivitelezéséhez pedig Trello-t alkalmaztunk.

Trello-ban három fő listát használtunk : elsőben a hosszabb távu terveink, elképzeléseink amiket meg szeretnénk valósítani, második a megoldandó, kivitelezendő, nagyobb fontossággal rendelkező task-okat, legvégül pedig a már elvégzett feladatok listája.

Github-on stabilabb változatait a kódunknak commit-oltuk hogy elérhető legyen számunkra.

# Kivitelezés

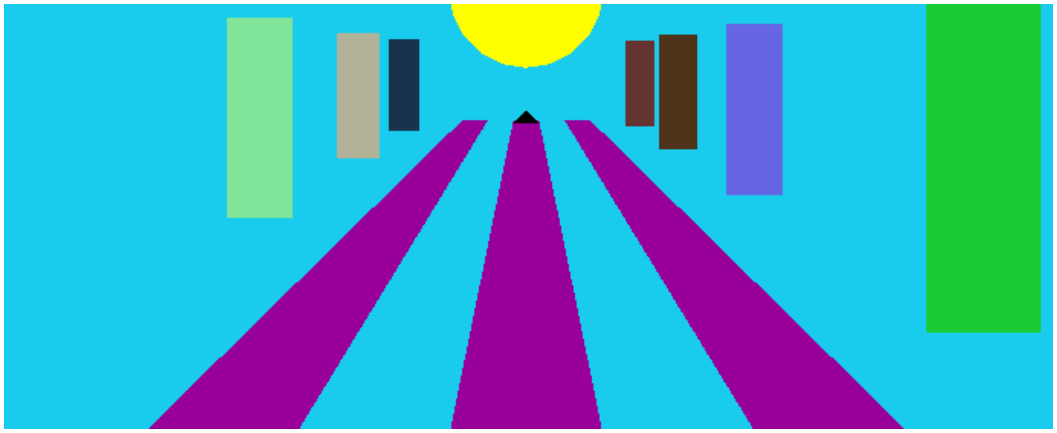
Egyszerű, mondhatni akár primitív megoldást egy három dimenziós tér ábrázolására az OpenGL-es parancsokkal jutottunk megoldásra, amellyel minden alapvető játékbeli elemet ki tudtunk rajzolni.

Legelső lépésben megterveztük a pályát, amelyhez három útra volt szükségünk. Próbálkozások után találtunk egy megfelelő dőlésszöget a perpektívának, illetve hosszat az utaknak ahonnan megfelelően láthatjuk a közeledő akadályokat nagyobb sebességeknél is.

A képernyő méretének átállításakor

```
,,
//road left
glBegin(GL_POLYGON);
glColor3f(0.6f, 0.0f, 0.6f);
glVertex3f(road_left, -0.1f, -70.0f);
glVertex3f(road_left-2, -0.1f, -70.0f);
glVertex3f(road_left-2, -0.1f, 10.0f);
glVertex3f(road_left, -0.1f, 10.0f);
glEnd();
//
//road middle
glBegin(GL_POLYGON);
glColor3f(0.6f, 0.0f, 0.6f);
glVertex3f(road_middle, -0.1f, -70.0f);
glVertex3f(road_middle-2, -0.1f, -70.0f);
glVertex3f(road_middle-2, -0.1f, 10.0f);
glVertex3f(road_middle, -0.1f, 10.0f);
glEnd();
//
//road right
glBegin(GL_POLYGON);
glColor3f(0.6f, 0.0f, 0.6f);
glVertex3f(road_right, -0.1f, -70.0f);
glVertex3f(road_right-2, -0.1f, -70.0f);
glVertex3f(road_right-2, -0.1f, 10.0f);
glVertex3f(road_right, -0.1f, 10.0f);
glEnd();
```

*Ábra 3*



*Ábra 4*

A 4. ábrán látható három lila sávot ábrázolja 3. ábrán levő kódrészlet, melyben ezek 10-es Z koordinátából -70-es pontig haladnak. Az X koordinátájukat külön változókba mentettük, hiszen ezek segítségével fogjuk később ellenőrizni az akadályokkal való ütközések pályáját.

Ezt követően a játékos által irányítható négyzetet hoztuk létre, ehhez tartozó kódot, illetve a mozgatását valósítottuk meg.

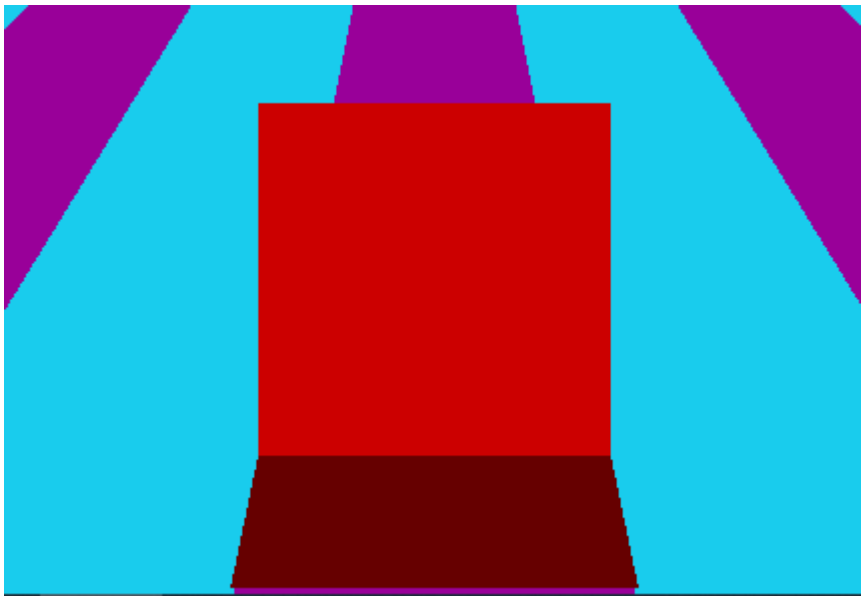
```
//square
glBegin(GL_POLYGON);
glColor3f(0.8f, 0.0f, 0.0f);
glVertex3f(square_corner1, 0.0f, 0.0f);
glVertex3f(square_corner2, 0.0f, 0.0f);
glVertex3f(square_corner2, 2.0f, 0.0f);
glVertex3f(square_corner1, 2.0f, 0.0f);
glEnd();
glBegin(GL_POLYGON);
glColor3f(0.4f, 0.0f, 0.0f);
glVertex3f(square_corner1, 0.0f, 0.0f);
glVertex3f(square_corner2, 0.0f, 0.0f);
glVertex3f(square_corner2, 0.0f, 1.0f);
glVertex3f(square_corner1, 0.0f, 1.0f);
glEnd();
//
```

*Ábra 5*

```
void MainWindow::keyPressEvent(QKeyEvent *event)
{
    if( event->key() == Qt::Key_A )
    {
        if(square_corner1==2.0)
        {
            square_corner2=-4.0;
            square_corner1=-2.0;
        }
        else if(square_corner1>-2.0)
        {
            square_corner1-=4;
            square_corner2-=4;
        }

    }
    if( event->key() == Qt::Key_D )
    {
        if(square_corner1==2.0)
        {
            square_corner2=4.0;
            square_corner1=6.0;
        }
        else if(square_corner1<6.0)
        {
            square_corner1+=4;
            square_corner2+=4;
        }
    }
}
```

*Ábra 6*

Az ötödik ábrán található a kirajzolása, hatodikon pedig az irányitása, a négyzet X koordinátájának változtatása billentyűzetgombok lenyomásakor, megakadályozva azokat az eseteket amikor kilépne a pályáról jobbra illetve balra.



*Ábra 7*

A hetes ábra az irányítható négyzetet ábrázolja a játékban a középső sávon.

Nélkülözhetetlen eleme a játéknak az akadály, amelyet véletlenszerűen megjelenő háromszögekkel oldottunk meg, egyszerre legtöbb három akadályt jelenhet meg a képernyőn.

```
//basic triangle obstacle
glBegin(GL_TRIANGLES);
glColor3f(0.0f,0.0f,0.0f);
glVertex3f(triangle_coord1-2, 0.0f, triangle_distance_a);
glVertex3f(triangle_coord1, 0.0f, triangle_distance_a);
glVertex3f(triangle_coord1-1, 1.0f, triangle_distance_a);
glEnd();
//
//double triangle obstacle
if(speed>1){
    glBegin(GL_TRIANGLES);
    glColor3f(0.0f,0.0f,0.0f);
    glVertex3f(triangle_coord2-2, 0.0f, triangle_distance_a);
    glVertex3f(triangle_coord2, 0.0f, triangle_distance_a);
    glVertex3f(triangle_coord2-1, 1.0f, triangle_distance_a);
    glEnd();
}
//
//half-distance triangle
if(triangle_distance_a>-35){
    triangle_distance_b+=speed;
    glBegin(GL_TRIANGLES);
    glColor3f(0.0f,0.0f,0.0f);
    glVertex3f(triangle_coord3-2, 0.0f, triangle_distance_b);
    glVertex3f(triangle_coord3, 0.0f, triangle_distance_b);
    glVertex3f(triangle_coord3-1, 1.0f, triangle_distance_b);
    glEnd();
}
```

*Ábra 8*

*Ábra 9*

Az akadályok generálásánál három esetet hoztunk létre, az első amikor csak egy háromszög halad végig egyik úton, egészen addig ameddig eléri az út végét. Második esetben két egymással egy síkba induló háromszögek közelednek a játékos fele, legvégül pedig a harmadik esetben két egymással ugyancsak egy síkban indulókat követi egy harmadik is amely akkor jelenik meg amikor az első két háromszög eléri az út felét, vagyis amikor a Z koordináta értékük eléri a 35-t.

A 10. ábra kódrészletében láthatjuk, hogy random számok segítségével döntjük el melyik eset fog teljestülni, vagyis azokra az esetekre hogy osztható hárommal, páros, illetve páratlan a kigenerált random számunk.

11. ábrán egy „wait” változót alkalmazunk, ennek segítségével fogjuk növelni az akadályok sebességét ; minden 3 kikerült eset után növeljük a „speed” értékét, amely a másodpercenkénti sebességét tárolja a háromszögeknek illetve a bal és jobb oldalt megjelenő téglalapoknak.

```
if(triangle_distance_a > 5.0 && triangle_distance_b > 5.0){
   triangle_distance_a = -70;
   triangle_distance_b = -70;
   int n = rand()%10;
   if(n % 3 == 0){
       triangle_coord1 = -2.0;
   }
   else if(n % 3 == 1){
       triangle_coord1 = 2.0;
   }
   else{
       triangle_coord1 = 6.0;
   }
   int q = rand()%10;
   if(q % 3 == 0){
       triangle_coord3 = -2.0;
   }
   else if(q%3 == 1){
       triangle_coord3 = 2.0;
   }
   else{
       triangle_coord3 = 6.0;
   }
```

*Ábra 11*

```
    if(wait == 2){
        if(speed <= 10){
            speed++;
            while(70 % speed != 0){
                speed++;
            }
        }
        wait = 0;
        int m = rand()%10;
        if(m % 3 == 0){
            triangle_coord2 = -2.0;
        }
        else if(m % 3 == 1){
            triangle_coord2 = 2.0;
        }
        else{
            triangle_coord2 = 6.0;
        }
    }
    else{
        wait++;
    }
}
```

*Ábra 10*

A sebességet minden két esetnyi kikerült háromszög után növeljük az update függvényben, egészen amíg eléri a maximális sebességét ( 10 ). Ugyancsak ebben a függvényben növeljük a Score értékét, amely a játék végekor megjelenített pontszámot mutatja.

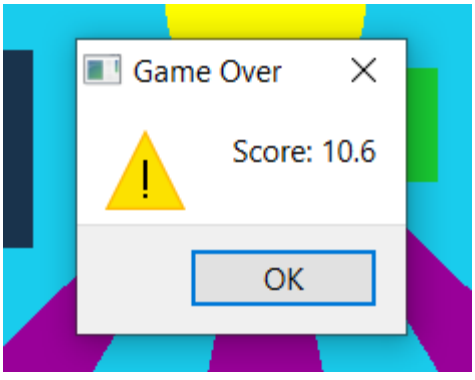
```
this->update();
score+=0.1;
//triangle speed
triangle_distance_a += speed;
```

*Ábra 12*

Abban az esetben amikor a négyzetünk X koordinátája megegyezik valamelyik háromszögével  és a Z koordináta is 0, akkor egy QMessageBox-ot használva értesítjük a játékost az elért pontszámával.

```
//Moment of impact
QLabel* label=new QLabel("Squares dash");
if(triangle_distance_a==0 && ((square_corner1 == triangle_coord1)||(square_corner1 == triangle_coord2)))
{
    QMessageBox::warning(label,"Game Over", QString("Score: %1").arg(score));
    exit(1);
}
if(triangle_distance_b==0 && (square_corner1 == triangle_coord3))
{
    QMessageBox::warning(label,"Game Over", QString("Score: %1").arg(score));
    exit(1);
}
//
```

*Ábra 13*

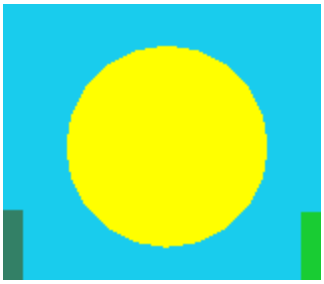



*Ábra 14*

Végső, de nem legutolsó sorban, a játék felületén megjelenik 8 különböző színü téglalap és egy kör amelyek fákat illetve egy napot illusztrálnak.

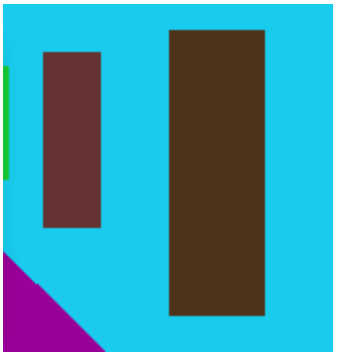
```
//sun
glBegin(GL_TRIANGLE_FAN);
glColor3f(1.0f, 1.0f, 0.0f);
glVertex3f(sun_x, sun_y, sun_z); //-> center of circle
for(int i = 0; i <= 20;i++) {
    glVertex3f(
        sun_x + (6 * cos(i * twicePi / 20)),
        sun_y + (6 * sin(i * twicePi / 20)),
        sun_z = -70
    );
}
glEnd();
//
//tree1
glBegin(GL_POLYGON);
glColor3f(0.2f, 0.5f, 0.4f);
glVertex3f(-6, 0.0f, tree1);
glVertex3f(-8, 0.0f, tree1);
glVertex3f(-8, 6.0f, tree1);
glVertex3f(-6, 6.0f, tree1);
glEnd();
//
```

*Ábra 15*



*Ábra 16*

A 17. ábrán levő téglalapok sebessége megegyezik a háromszögekével, mozgás érzetét keltve a játékban, a 18. ábra kódja szerint.



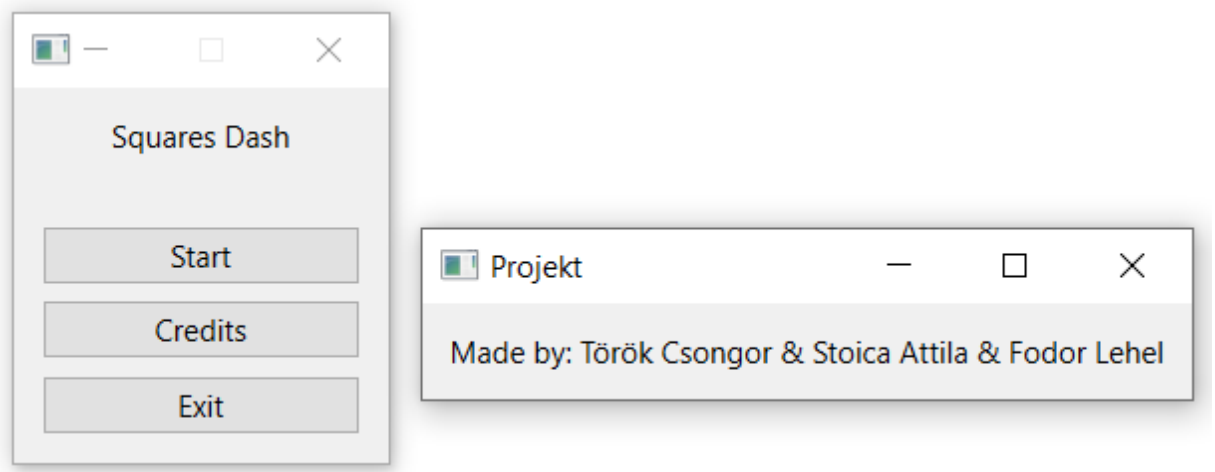
Ábra 18

```
tree1 += speed;
tree2 += speed;
tree3 += speed;
tree4 += speed;
tree5 += speed;
tree6 += speed;
tree7 += speed;
tree8 += speed;
```

Ábra 17

A játék felületének az elérését egy minimalisztikus menüvel oldottuk meg, felhasználva egy label-t a játék nevével, továbbá 3 gombot, a játék elindításához, készítők megjelenítéséhez és az alkalmazásból való kilépéshez.

A Credits gomb lenyomásakor egy újabb ablak jelenik meg egy szimpla label-el, ahogy a 19. ábra is mutatja, az előbb említett menüvel együtt.




Ábra 19

Egyúttal létrehoztunk egy adatbázist is ahol elmentenénk a játékosok nevét, elért pontszámaikat pedig egy újabb táblába helyezve.

```
--Creating the tables
create table Players(
   Player_ID number
   generated always as identity primary key,
   Player_name varchar(30) not null,
   Email_address varchar(50) not null,
   Player_password varchar(20) not null
);

create table Results(
   Result_id number
   generated always as identity primary key,
   Score Double precision not null,
   Player_ID number references Players(Player_ID)
);
```

*Ábra 20*

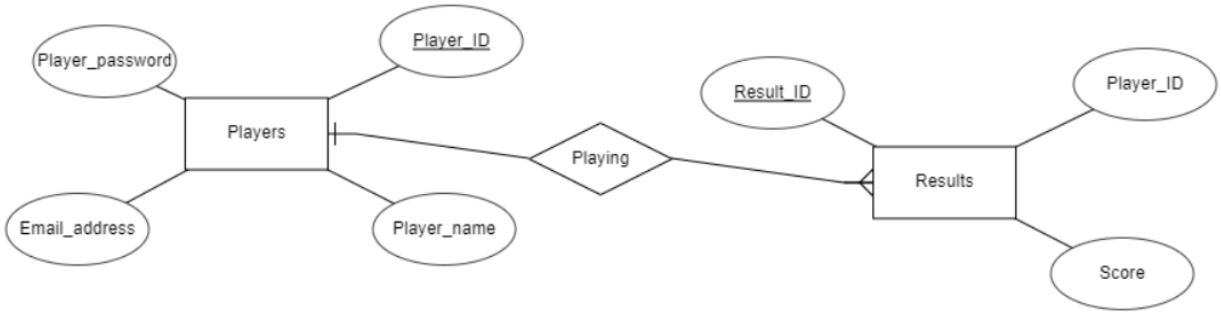
```
--Testing
select * from players;
select * from results;

--Testing insert

insert into Players(
    Player_name,
    Email_address,
    Player_password)
    values(
    'Stoica Attila',
    'stoica.attila@student.ms.sapientia.ro',
    'chpc1122'
    )
;
insert into Players(
    Player_name,
    Email_address,
    Player_password)
    values(
    'Török Csongor',
    'torok.csongor@student.ms.sapientia.ro',
    'tppad57'
    )
;
```

*Ábra 21*

```
insert into Results(
     Score,
     Player_ID)
     values(
     36.5,
     1
     )
;

--Testing update
update results
set score=49.03
where player_id=1;

--Query with join
select r.score
from
players p join results r
on p.Player_ID = r.Player_ID;

--Testing delete
delete from players
where player_id=2;
```

*Ábra 22*

Az adatbázis EK diagramja a következőképpen néz ki :




*Ábra 23*

# Továbbfejlesztési lehetőségek

- Adatbázisunk összekapcsolása a programunkal, High Score lista létrehozására illetve felhasználók adatainak egyszerűbb tárolásához
- Hangeffektusok hozzáadása mozgáskor, ütközéskor illetve háttérben zene amely a nehézségi szint szerint változik
- Texturák hozzáadása az elemekhez
- Power-up-ok, segítségek a játékos számára, pld. extra élet, felszedhető pontgyüjtő elemek
- Androidon, IOS-en, webes felületen való kivitelezés
- Komplexebb akadályok generálása

# Hivatkozások


- https://doc.qt.io/
- https://stackoverflow.com/