SZABADKAI MŰSZAKI SZAKFŐISKOLA
SZABADKA

OPENGL 3D KACSA VADÁSZÓS JÁTÉK

projektum
Software Engineering tárgyból

Tanár: Dr. Simon János

hallgató: Kovács Róbert
leckekönyv: 16218119
szakirány: Műszaki informatika

Szabadka, 2020/21

Tartalom

Projekt leírás

Bevezető

A szoftver C++ programozási nyelvvel és az OpenGL könyvtárakkal készült. Ez a szoftver egy 3D -s kacsa lövöldözős játék, ahol belsőnézetből kell célozni kacsákra, és minél többet lelőni 1 perc alatt.

Felhasználói követelmények

1. A játéknak kell egy menü.
2. A játéknak kell legyen célja.
3. A játéknak kell lennie valamennyi kihívásnak.
4. A lövésről hang visszajelzést adni.
5. Az egy játék alatt elért pontszámot megkel jeleníteni játék közben és a hátralévő időt is.
6. Kell egy befejező képernyő, ahol a végső elért pontszámot megkel jeleníteni és lehetőséget adni az újraindításra.

Egyéb követelmények

1. A szoftvert OpenGL könyvtárral kell fejleszteni.

A szoftver felépítése

Objektum rendszer

A szoftver egyedi szöveges fájlból tölti be először az objektumokat. Ebben a szöveges fájlban megadhatjuk, hogy töltsön-e be texturat rá, az átlátszó-e vagy nem. Megadhatjuk még a színét és, az objektumnak az alakját.

```
texture img/duck.png 1
on duck

c | 1 1 1

t | 0 0
v | 0 0 0
t | 0 1
v | 0 1 0
t | 1 1
v | 0 1 1.2
t | 1 0
v | 0 0 1.2
```

Ábra 2: egyedi obj fájl

Ugyan ebben objektumban tárolódik az objektum pozíciója, forgása stb, ami fontos a különböző objektum meghatározásához.

```
glPushMatrix();
glBegin(GL_QUADS);
if (red == -1)
    glNormal3f(-1,0,0);
while(!(feof(fp)))
{
    read=fscanf(fp,"%s %s %f %f %f",str0,str1,&x,&y,&z);
    if(read==2 && string(str0)=="on")
    {
        obj_name = str1;
    }
    if(read==5 && string(str0)=="c")
    {
        if (red == -1)
        {
            glColor3f(x,y,z);
        } else
        {
            glColor3ub(red,0,0);
        }
    }
    if(read==4 && string(str0)=="t" && red == -1)
    {
        glTexCoord2f(x, y);
    }
    if(read==5 && string(str0)=="v")
    {
        glVertex3f(x,y,z);
    }

}
glEnd();
glPopMatrix();
```

Ábra 1: objektum betöltése kódból részlet

Scene rendszer

A jelent objektum tárolja a pillanatnyilag megjelenítendő objektumokat egy Vector listában. Ehhez a listához objektumokat lehet hozzáadni. Ez a Scene objektum is rendelkezik szöveges fájlból való betöltéssel. Ami a betöltendő objektum nevét, pozícióját és forgását írja le.

```
1  sn game
2
3  obj obj/grass.robj 23 -2 -17.5 0 90
4  obj obj/crosshair.robj 0 0 -10 -90 0
```

Ábra 3: egyedi scene szöveges fájl

Ehhez a scene objektumhoz menet közben is hozzáadhatunk bármikor objektumot. Ezen a scenen ciklussal végigbírunk menni és bármikor bármihez hozzátudunk férni a fő kódban.

Kattintás észlelése

Kattintásnál lekérődik a kattintás pozíciója ezután minden objektum ami a scene-ben van a piros egyik árnyalatára változik át. 0-255-ig vehetnek fel értéket. Az-az csak ezzel 255 különböző tárgyra tudunk kattintani. Ez után lekéri az egér pozíciójánál lévő színt, amit megnéz, hogy melyik objektum tárolja. Ezek után meg végrehajtódik a specifikus kód az arra vonatkozó objekumra. A lövés is ugyan ez a történet.

```
151 void picking(int x, int y)
152 {
153     unsigned char pixel[4];
154     glReadPixels(x, window.height-y, 1, 1, GL_RGB, GL_UNSIGNED_BYTE, pixel);
155     if (pixel[1] == 0 && pixel[2] == 0)
156         for (int i = 0; i < scene.getObjectsSize(); i++)
157         {
158             obj = &scene.getObject(i);
159             if (obj->getPicking() == pixel[0])
160             {
161                 //cout << "\n" << obj->getObjName() << ":Clicked\n";
162                 Object tobj;
163                 switch (str2int(obj->getObjName().c_str()))
164                 {
165                     case str2int("duck"):
166                         game.points += 150;
167                         scene.removeObject(i);
168                         Mix_PlayChannel( -1, duckQuack, 0 );
169                     break;
170                     case str2int("exitButton"):
171                         close();
172                     break;
```

Ábra 4: kattintás észlelés kód részlet

A játék időzítésének működése

Egy timer osztály felel ezért, ami elég egyszerűen dolgozik. A fő kódrészben létrehozunk különböző timer objektumokat a különböző időzítéseknek. Amikor odaér az ellenőrzéséhez minden fő ciklusban, akkor megnézi, hogy lejárt-e az idő, ha nem akkor kivonja belőle az OpenGL rendelerelésére szükséges időtartamot ameddig tartott egy új képet kirenderelnie, szóval a delta időt.

```
if (game.duck_spawn_timer.calculaTimeRemaining(frametime.dt))
{
    game.duck_spawn_timer.setTimeRemaining(1);
    Object tobj;
    tobj.Init("obj/duck.robj");
    tobj.togglePicking(-1);
    tobj.setDuckRngVar(1 + rand() % 8, rand() % 6, rand() % 4, 1 + rand() % 20, 1.5 + rand() % 4);
    tobj.setPos(-6 + (rand() % 35), 1, -40 + rand() % 20);
    scene.addObject(tobj);
}
if (game.game_timer.calculaTimeRemaining(frametime.dt))
{
    scene.loadScene("scene/endgame.scn", 1);
    g_fps_mode = false;
    glutSetCursor(GLUT_CURSOR_INHERIT);
    Object tobj; tobj.setObjName("Points"); tobj.Init("-");
    tobj.setText("Points:" + to_string(game.points));
    tobj.setPos(-0.25,0.3,-0.1);
    scene.addObject(tobj);
    g_camera.Init();
    g_camera.SetPos(-10,0,0);
}
```

Ábra 5: időzítés rövid kód részlet (kacsa spawnolás és játék idő)

Megjelenítés a 3D-s térben

A 3D -s térben perspektív vetítést használunk.

```
249     for (int i = 0; i < scene.getObjectsSize(); i++)
250     {
251         obj = &scene.getObject(i);
252         switch (str2int(obj->getObjName().c_str())) {
253             case str2int("Points"):
254             break;
255             case str2int("Time"):
256             break;
257             case str2int("crosshair"):
258             break;
259             case str2int("duck"):
260                 glPushMatrix();
261                 obj->setcSinY(obj->getcSinY() + obj->getMultiplySinRng() * frametime.dt);
262                 obj->setPos(obj->getPosX(),
263                             sinf(obj->getcSinY()) * obj->getMultiplyOutSinRng() + obj->getAddOutSinRng(),
264                             obj->getPosZ() + obj->getDuckSpeedRng() * frametime.dt);
265                 obj->refreshPosRot();
266                 if (mouse.picking)
267                 {
268                     obj->togglePicking(i);
269                 }
270                 glCallList(obj->getMesh());
271                 glPopMatrix();
272             break;
273             case str2int("grass"):
274                 glPushMatrix();
275                 obj->refreshPosRot();
276                 glScalef(35,35,35);
277                 glCallList(obj->getMesh());
278                 glPopMatrix();
279             break;
280             default:
281                 glPushMatrix();
282                 obj->refreshPosRot();
283                 if (mouse.picking)
284                 {
285                     obj->togglePicking(i);
286                 }
287                 glCallList(obj->getMesh());
288                 glPopMatrix();
289             break;
290         }
291     }
```

Ábra 6: Különböző objektumok megjelenítése a 3D-s térben kódrészlet.

Megjelenítés a 2D-s térben

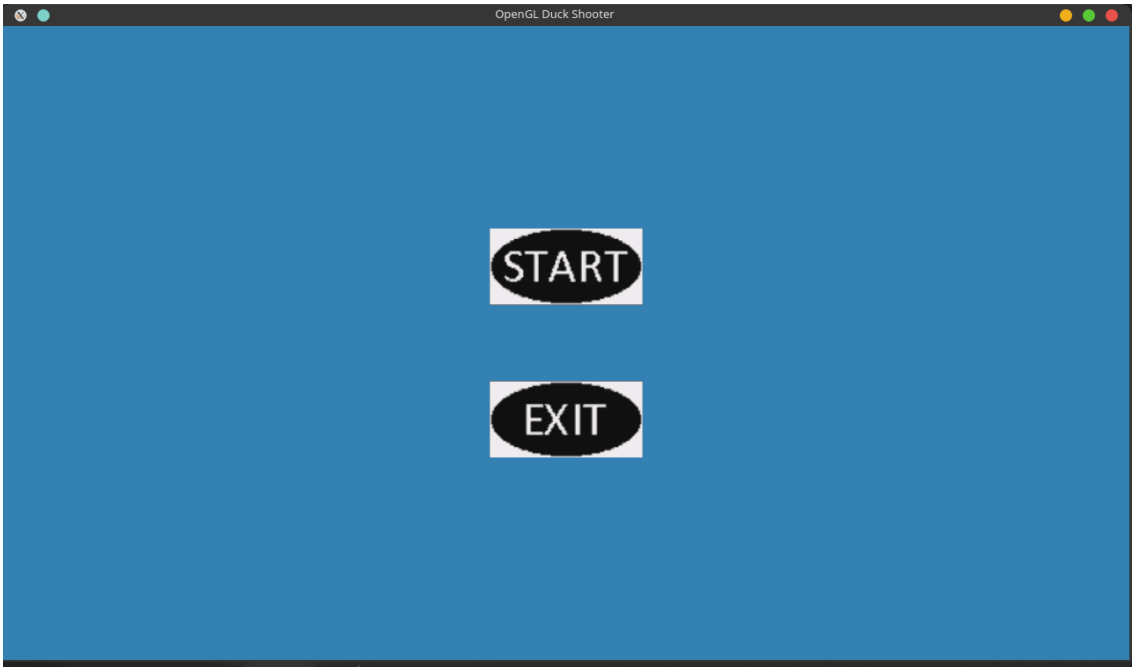
A 2D-s térben való megjelenítéshez ortogonális nézetet használunk.

```
314     glMatrixMode(GL_PROJECTION);
315     glLoadIdentity();
316     glOrtho(-1, 1, -1, 1, 0.01, 1000);
317     glMatrixMode(GL_MODELVIEW);
318     glLoadIdentity();
319     glDisable(GL_DEPTH_TEST);
320
321     for (int i = 0; i < scene.getObjectsSize(); i++)
322     {
323         obj = &scene.getObject(i);
324         switch (str2int(obj->getObjName().c_str()))
325         {
326         case str2int("Points"):
327             glPushMatrix();
328             obj->refreshPosRot();
329             glLineWidth(3);
330             glScalef(0.001,0.001,0.001);
331             obj->setText("Points:" + to_string(game.points));
332             obj->loadObj();
333             glCallList(obj->getMesh());
334             glPopMatrix();
335         break;
336         case str2int("Time"):
337             glPushMatrix();
338             obj->refreshPosRot();
339             glLineWidth(3);
340             glScalef(0.001,0.001,0.001);
341             obj->setText("Time:" + to_string((int)game.game_timer.getTimeRemaining()));
342             obj->loadObj();
343             glCallList(obj->getMesh());
344             glPopMatrix();
345             break;
346         case str2int("crosshair"):
347             glPushMatrix();
348             obj->refreshPosRot();
349             glScalef(0.1,0.1,0.1);
350             glCallList(obj->getMesh());
351             glPopMatrix();
352         break;
353         }
354     }
```

Ábra 7: 2D-s térben megjelenítés kód részlet

A játék bemutatása

Amikor elindítjuk a játékot akkor a menü tárul elénk. Itt kitudunk lépni és eltudjuk indítani a játékot. A játék és a kilépést kattintással tehetjük meg, illetve a kilépést még az „esc” gomb lenyomásával is elérhetjük.

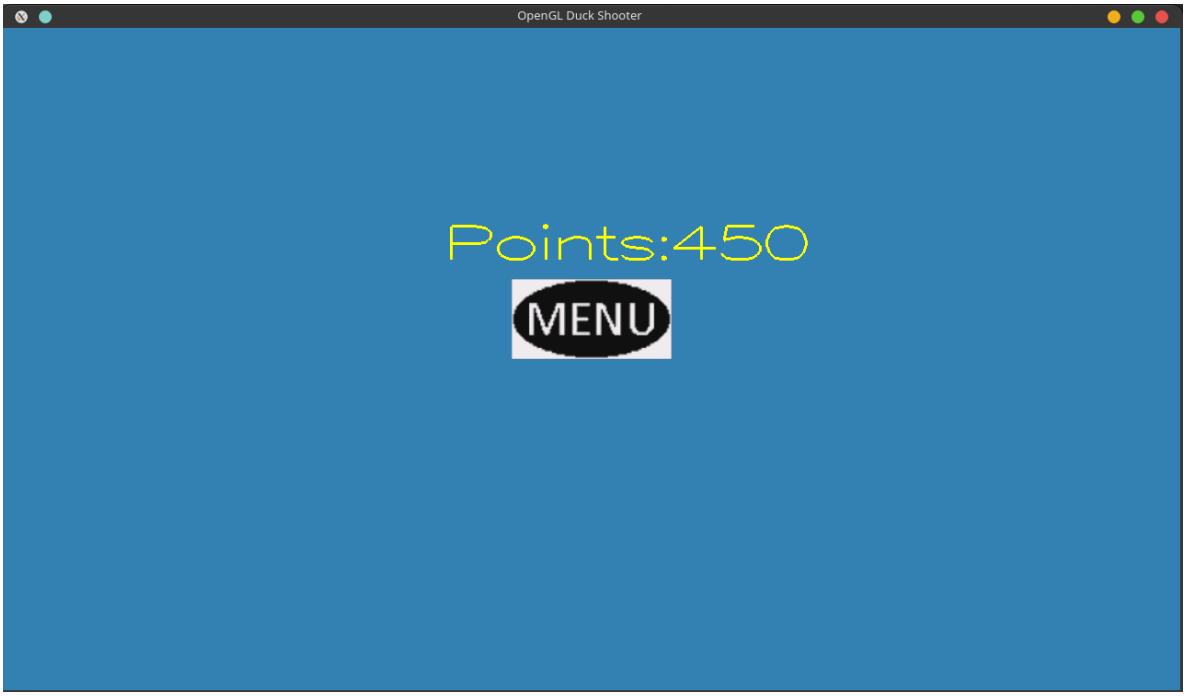


Ábra 9: a játék menű jelenete

Start gomb lenyomása után a játék fő jelenetére kerülünk. Ahol elkel találni az egér mozgatásának a segítségével a kacsákat, ha eltaláljuk akkor plusz 150 pont, ha pedig mellé lövünk, akkor mínusz 50 pont. „esc” gomb lenyomásával visszakerülhetünk a menűre.

Ábra 8: a játék fő jelenete

A játék eredménye jelenetnél láthatjuk, hogy mennyi pontot sikerült összeszedni. A menü gombra kattintva visszajuthatunk a fő menűre, illetve ezt „esc” gomb lenyomásával is megtehetjük.



Ábra 10: a játék eredmény jelenete

Források

https://docs.gl/