Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik

Tanszék

Interakció fraktálokkal

Témavezető:
Bán Róbert
Doktorandusz, MSc.

Szerző:
Borbély Dávid
programtervező informatikus BSc.

Budapest, 2020

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

Tartalomjegyzék

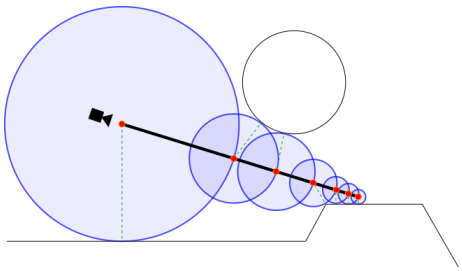
1. fejezet

Bevezetés

Ha valaki játékfejlesztésbe szeretne kezdeni, akkor nagyon sok kihívással kell szembenéznie. Szerencsére manapság már levehet egy terhet a válláról ha egy előre megírt játékmotort (**game engine**) használ, amiből akár ingyenesen elérhetőt is lehet találni.

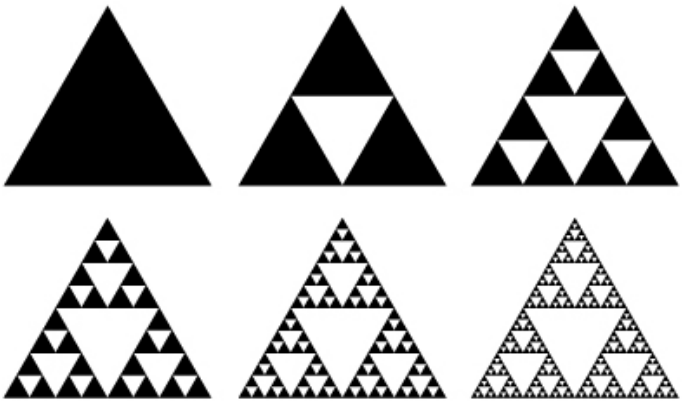
Egy játékmotor feladata hogy leegyszerűsítse a kirajzolást és az objektumok **valósághű viselkedését**. Ilyen viselkedés például, ha két szilárd tárgy ütközésekor azt várnánk el hogy azok ne menjenek bele egymásba, hanem inkább ténylegesen ütközzenek és "pattanjanak le" a másikról. Ezen viselkedés kiszámolása meglehetősen költséges tud lenni, ráadásul különböző alakzatoknál különböző algoritmusokat kell használni.

Szakdolgozatomban azzal foglalkozom hogy hogyan lehet a kirajzolást és az előbb említett valósághű viselkedést fraktálokkal elvégezni. Mivel a fraktálok nehezen leírható felülettel rendelkeznek így a lehető legáltalánosabban kell megközelíteni a velük való ütközést.



1.1. ábra. Sphere tracing: A kamerából kiinduló sugár mindig annyit lép előre amekkora a hozzá legközelebb lévő tárgy távolsága

Hogyan valósítom ezt meg? A kirajzoláshoz **Sphere tracing** módszert (1.1 ábra) alkalmazok, így minden kirajzolt objektumomhoz van távolságfüggvényem. Ezek segítségével meg tudom állapítani a virtuális terem bármely pontjáról hogy az milyen messze van a kirajzolt felületektől. Ezen tudással nagyon egyszerűen és hatékonyan meg lehet állapítani hogy egy gömb ütközött-e bármivel, hiszen csak annyit kell ellenőriznünk hogy a gömb középpontja gömbsugárnyi távolságra van-e valamilyen felülettől. Ezután a gömb sebességvektorát a felület normálvektora körül megforgatjuk 180 fokban és ellentétes előjelűvé tesszük, mintha csak egy fénysugárra hatna a teljes fényvisszaverődés a felület normálisának megfelelő beesési merőlegesben.



1.2. ábra. Példa egy IFS-re: a Sierpiński háromszög 5 iterációja

Ezekhez azonban pontos és lehetőleg előjeles távolságfüggvények kellenek, így nem érdemes foglalkozni az olyan fraktálokkal amikhez a távolságfüggvény csak felső becslést ad. Ezért olyan a fraktálok egy olyan csoportjával foglalkozom, mint a Sierpiński háromszög (1.2 ábra), amik **IFS (Iterated Function System)** által jönnek létre, azaz egy egyszerűbb alakzaton - aminek jól ismerjük a pontos távolságfüggvényét - sokszor végrehajtunk egymás után transzformációkat. Az ilyen fraktálokat könnyű generálni, mert ha eldöntöttük milyen transzformációink lesznek, azok újraparaméterezésével könnyen meghatározhatunk egy újabb fraktált.

2. fejezet

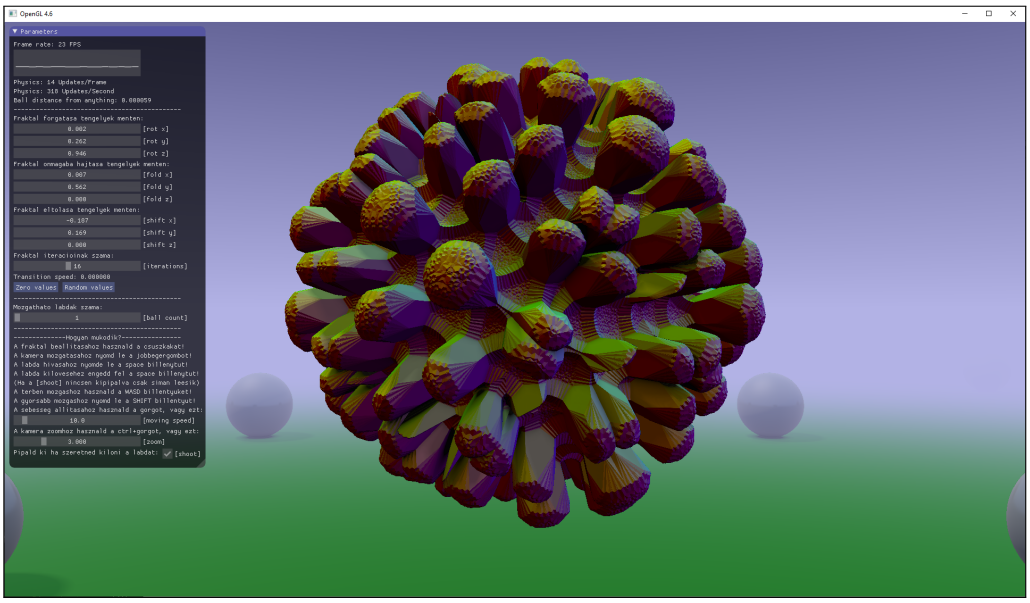
Felhasználói dokumentáció

Ezen fejezet fogja taglalni a program futtatásához és használatához szükséges információkat. A felhasználói felület is tartalmaz rövid leírást, de a program részletes használati útmutatója a soron következő alfejezetben lesz megtalálható. A programmal egy virtuális teret lehet bejárni, melynek talaján minden irányban végtelen sok mozdíthatatlan gömb található. Van a térben továbbá egy nem aktívan mozgó, de testre szabható fraktálunk, valamint vannak mindenfelé kilőhető és mindenről visszapattanó labdák, melyekkel demonstrálni lehet a programban megírt fizikát.

2.1. Felhasználói felület és funkciók

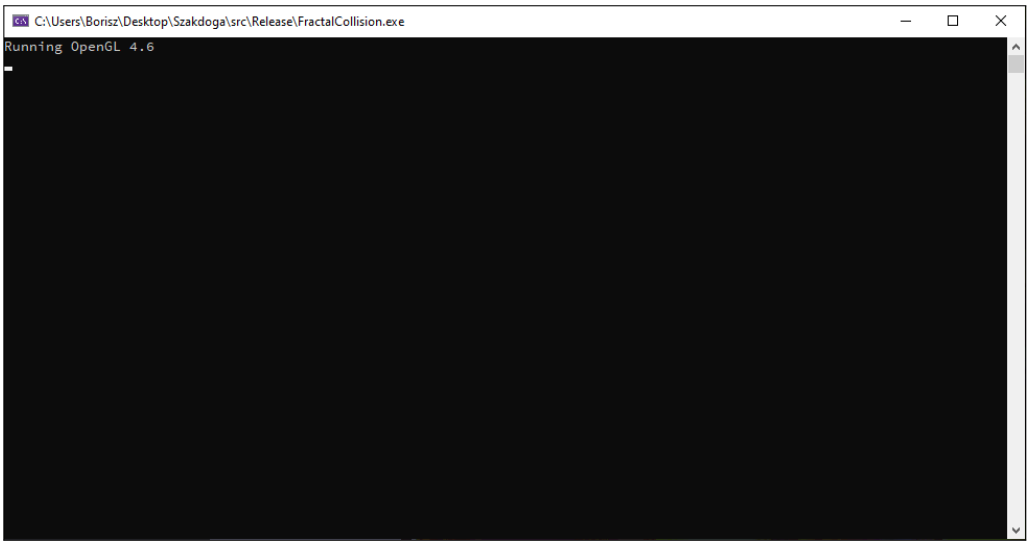
A program sikeres indítása után - melyről a 2.2. fejezetben tudhatunk meg többet - kettő darab ablakkal találjuk szembe magunkat. Ezekről a 2.1 és 2.2 ábrák mutatnak egy-egy képernyőképet.

A 2.1 ábrán látható ablak tartalmazza a program lényeges részét, itt jelenik meg a kirajzolt képünk és ebben az ablakban található a "Parameters" feliratú panel, melynek segítségével különböző paramétereket tudunk nyomon követni és módosítani. Az ablak alapértelmezetten 1920x1080 nagyságú, de szabadon átméretezhető, viszont az ablak mérete befolyással van a teljesítményre! Mindig az ablak pontos felbontásában fog renderelni, így nem optimális teljesítmény esetén érdemes megfontolni az ablak kisebbre vételét.



2.1. ábra. Fő programablak

A 2.2 ábrán vehetjük szemügyre azt a terminálablakot mely az esetleges hibaüzeneteket fogja kiírni. Ezen kívül ez az ablak más funkcióval nem bír.

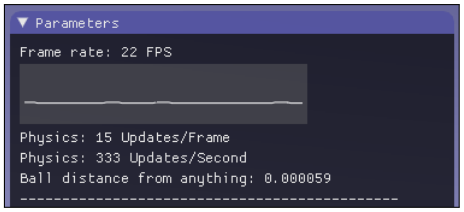


2.2. ábra. Terminál ablak

2.1.1. "Parameters" panel

A "Parameters" felirítú panel nagy jelentőséggel bír, így a jobb olvashatóság végett nem csak a 2.1 ábra részeként láthatjuk hanem külön is szerepel a 2.3, 2.4 és 2.5 ábrákon.

Ezen a panelen számos információt tudhatunk meg és állíthatunk át a program futásával kapcsolatosan. Alapértelmezetten a fő programablak bal oldalán található, de szabadon mozgatható és átméretezhető az ablakon belül, indításkor pedig az legutóbbi futtatás végén beállított pozíciót és méretet veszi fel.

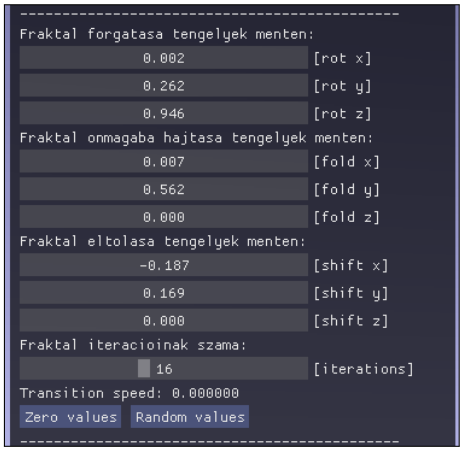


2.3. ábra. A "Parameters" feliratú panel felső harmada

A panel legtetején (2.3 ábra) találhatjuk a **"Frame rate:"** felirat után az aktuális képfrissítési rátát képkocka/másodperc mértékegységben, illetve közvetlenül ezalatt az utolsó másfél másodperc adatait követhetjük nyomon egy folyamatosan frissülő ábrán.

A két **"Physics:"** felirat után olvashatjuk le hogy milyen gyakran van a labdák mozgása frissítve frissítés/képkocka és frissítés/másodperc mértékegységekben. Egy frissítés során minden labda sebessége és pozíciója újraszámolódik, valamint ellenőrzésre kerül az is hogy ütközött-e valamivel.

A **"Ball distance from anything:"** felirat után olvasható a dobálható labda távolsága a tőle legközelebb lévő felülettől - több labda esetén az utoljára létrehozottra vonatkozik. Az apró ingadozásából látszik hogy igazából folyamatosan pattog a labda, csak ez a pattogás egy idő után szabad szemmel nem látható.



2.4. ábra. A "Parameters" feliratú panel középső harmada

A választóvonal alatti szekcióban (2.4 ábra) a fraktálunkat tudjuk személyre szabni. A fraktálunk úgy rajzolódik ki hogy egy 1x1x2 egység nagyságú téglatesten egymás után többször végrehajtunk különböző transzformációkat. Ezen transzformációk paramétereit tudjuk beállítani a következő 9 db határ nélküli csúszkán - mely ugyanúgy működik mint egy sima csúszka, csak nincsen minimum és és maximum értéke és az egeret tovább is lehet húzni mint a csúszka vége. Mindegyik csúszkának CTRL + kattintással begépelt értéket is meg lehet adni.

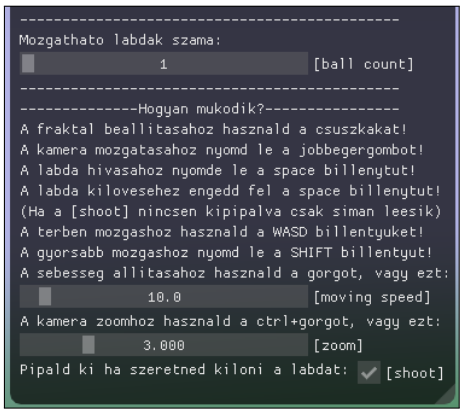
A **[rot x], [rot y], [rot z]** csúszkákkal azt tudjuk szabályozni hogy mennyire legyen elforgatva a fraktál az X, Y és Z tengelyek körül (radiánban értendők az értékek).

A **[fold x], [fold y], [fold z]** csúszkákkal azt tudjuk szabályozni hogy mennyire legyen elforgatva az adott tengely körül a tükrözősík ami a először a megadott szög (radián) kétszeresével fordul el és tükrözi az alakzatot, majd ellentétes irányban az eredeti szöggel. Itt a három érték 3 különböző tükrözősíkot forgat el a nevükben szereplő tengely mentén.

A **[shift x], [shift y], [shift z]** csúszkák szabályozzák hogy mennyire legyen eltolva az alakzat az X, Y és Z tengely mentén.

Végül pedig az **[iterations]** csúszka, amely már korlátozva van az [1,36] tartományban, beállítja hogy az előző 9 csúszka által paraméterezett 9 transzformáció hányszor legyen végrehajtva a téglatesten. Ez a paraméter van a legnagyobb hatással a futás sebességére, így gyengébb gépeken nem érdemes nagy értéket beállítani. Ha a képfrissítési ráta 10 képkocka/másodperc alá csökken akkor automatikusan elkezd csökkenni a csúszka értéke.

Itt található még két gomb: a **"Zero values"**, mely a fraktál paramétereket nullára állítja, illetve a **"Random values"**, mely véletlenszerű értékeket állít be ezeknek. Az iterációk számát egyik sem állítja át. Továbbá van még egy ezekhez szorosan kacsolódó érték ami a **"Transition speed:"** felirat után olvasható. A gombok által generált új paramétereket egy 5 másodperc hosszú fázisban folyamatosan, apránként közelíti az aktuális paraméterekkel, az előbb említett érték pedig azt fejezi ki hogy milyen súlyozással veszi az aktuális és a célérték átlagát.



2.5. ábra. A "Parameters" feliratú panel alsó harmada

Az alsó harmadban (2.4 ábra) található a **[ball count]** csúszka, itt 1 és 100 között lehet értékeket beállítani. Ez is jelentősen befolyásolja a futás gyorsaságát, így 15 FPS alatt ez az érték is automatikusan csökken.

Van egy rövid ismertető szövege a panelnek, ami azt a célt szolgálja hogy ezen dokumentáció nélkül is tudja használni egy felhasználó ha leül a program elé. Ebben kerülnek ismertetésre a virtuális tér bejárásához szükséges irányítások is. A virtuális térben mozgáshoz a **WASD** billentyűket kell használni a legtöbb játékban megszokott módon. A mozgási sebességet a **[moving speed]** csúszkával lehet személyre szabni, illetve ugyanezen csúszka értékét az **egérgörgővel** is lehet szabályozni. A **shift** billentyű lenyomására ideiglenesen megnégyszereződik a sebesség, felengedésére visszaáll. A virtuális kamera mozgatásához le kell nyomni a jobbegérgombot és mozgatni az egeret. A kamera látószögét lehet csökkenteni a **[zoom]** csúszka értékének növelésével, vagy a **CTRL + görgő** segítségével is.

A labdákat a **szóköz** billentyű lenyomásával lehet magunkhoz hívni. Egy labda esetén az ablak közepére, több labda esetén a labdák az ablak közepe körül keringenek egy korvonal mentén egyenletesen elhelyezkedve. A szóköz billentyűt felengedve egyszerre "kilövődnek" a labdák, ha be van pipálva a panel alján a **[shoot]** jelölőnégyzet, ha nincsen bepipálva csak leesnek a gravitációnak megfelelően.

2.2. Rendszerkövetelmények és futtatás

Az alkalmazás üzembe helyezésének egyetlen követelménye a Windows 7 vagy afeletti operációs rendszer. Azonban az alkalmazás rettentően GPU intenzív, így az optimális futáshoz elengedhetetlen a dedikált videokártya. A teszteléshez használt számítógép specifikációi:

- Intel® Core™ i7-8700 CPU
- 16 GB RAM
- NVIDIA GeForce GTX 1660 GPU
- Windows 10 operációs rendszer

Ezen konfigurációval, 1920x1080 felbontás mellett a program sebessége elfogadható volt.

Az alkalmazás elindításához a **FractalCollision.exe** fájlt kell futtatni. Fontos hogy az exe fájl mellett ott legyen a **myFrag.frag** és a **myVert.vert** fájlok, illetve ha a rendszeren nincsen külön telepítve akkor az **SDL2.dll**, valamint a **glew32.dll** fájloknak is az exe mellett kell lenniük.

Az alkalmazásból való kilépéshez lehet az **ESC** billentyűt vagy a jobb felső sarokban az ablak bezárás gombját használni. Ha bezárjuk a terminálablakot akkor mindkét ablak bezárul, ha először a fő programablakot zárjuk be akkor utána még külön be kell zárni a terminálablakot.

Az alkalmazás **Microsoft Visual Studio** segítségével készült, így ha újra akarnánk fordítani akkor a **C:/** helyre csomagoljuk ki a mellékelt OGLPack.zip állományt (ez az OpenGL-hez szükséges fájlokat tartalmazza), majd futtassuk a **subst T: C:/** parancsot. Ezután megnyithatjuk a **.vcxproj** projektfájlt.

3. fejezet

Fejlesztői dokumentáció

folyamtban...

4. fejezet

Összegzés

folyamatban...

A. függelék

Függelék

folyamtban...

Ábrák jegyzéke

Táblázatok jegyzéke

Forráskódjegyzék