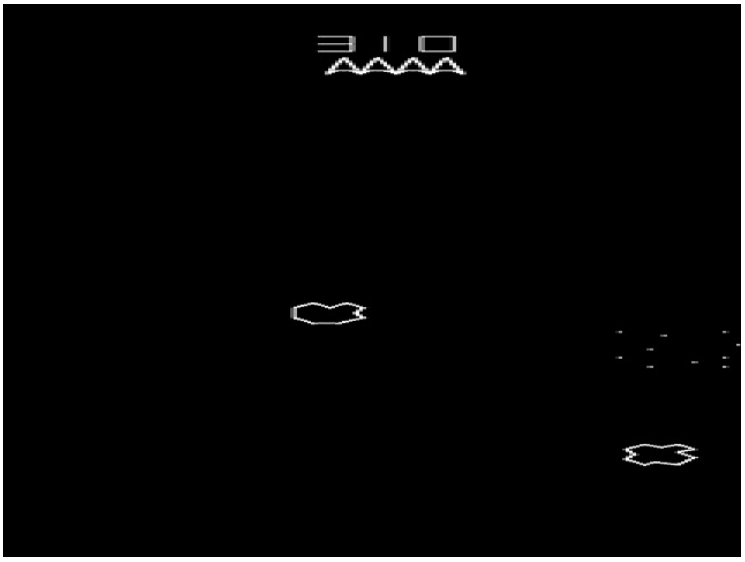
3D Játék - Aszteroidák

Forrás: http://en.wikipedia.org/wiki/Asteroids_%28video_game%29

Alapfeladat (Σ 15%)

Az alkalmazás indításakor jelenjen meg egy fekete háttér, a közepén egy űrhajóval és két aszteroidával! (**5%**) A feladatok csak akkor fogadhatóak el, ha az alakzatok 3D-sek.
Az alkalmazás működése közben legyenek láthatóak lövedékek (hengerek), aszteroidák (módosított, torzított gömbök) és egy űrhajó (tetraéder test és egyéb tetszőleges díszítés, https://hu.wikipedia.org/wiki/Tetra%C3%A9der). (**5%**) A különböző alakzatok eltérő színek segítségével legyenek kirajzolva (kitöltés ill. körvonalak)! (**5%**)
Megjegyzés: Ha körvonalakat használsz, valamennyire idézze az eredeti “retro” stílust!



(retro style)

Űrhajó mozgás + lövés (Σ 25%)

Az játéktér az **XZ síkon fekszik**, tehát mind az aszteroidák, mind az űrhajó az XZ síkon mozog.

- Az űrhajót lehessen **forgatni** a jobbra-balra billentyűk segítségével a megfelelő irányba. (**5%**)
- Az űrhajó legyen képes **előre haladni** a jobb egérgomb segítségével. (**5%**)
- A SPACE billentyű lenyomására az űrhajó orrából szálljanak **lövedékek** nagyjából abba az irányba, amerre néz! (**5%**)
- Ha többet repültek már, mint 300 egység, semmisítse meg a lövedékeket a program. (**5%**)
- Az űrhajó oldalainak háromszögei közül legalább kettőt lásson mindig a kamera és kövesse az űrhajót mindig azonos távolságból! (**5%**)

Aszterodiák paraméterei (Σ 20%)

Az aszteroidák alapja egy-egy gömbfelületről random mintavételezett (normális irányokban eltolt) **torzított gömbfelület**. Azaz, ha a gömb középpontjától 1 egységre volt egy pont, azt random el lehet tolni az alakzat generálásakor. (**5%**)

- Minden aszteroida különböző legyen és random sebességgel forogjanak a különböző irányokban! (**5%**)
- Az aszteroidáknak legyen háromféle mérete rendre 7, 6 és 5 “detailedness” felbontásban! (**5%**)
- A legnagyobb aszteroida 10 egység sugarú legyen, a kisebbek 7 illetve 4 egység. (**5%**)

Aszterodiák szétesése (Σ 20%)

A lövedékek különböző aszteroidákkal érintkezve semmisüljenek meg és a következőket váltsák ki:

- Nagy aszteroida: essen 3 közepes aszteroidára, induljanak különböző irányokba! (**5%**)
- Közepes aszteroida: essen 2 kicsi aszteroidára, repüljenek különböző irányokba! (**5%**)
- Kicsi aszteroida: semmisüljön meg! (**5%**)

Az ütközést gömb (aszteroidát befoglaló gömb) és gömb (a hengert befoglaló gömb) érintkezésvizsgálatával oldd meg! (**5%**) Ha az asztoroidák távolabb repülnek mint 300 egység: semmisüljenek meg, és generálódjanak újak az űrhajó háta mögött, nem látható területen, a közelben. **(5%)**

Tömegvonzás és fizikai szimuláció (Σ 15%)

Az űrhajó és az aszteroidák rendelkezzenek tömeggel. Ha egy aszteroida kisebb darabokra esik, a kezdeti tömeg osztódjon szét a darabok között! (**5%**) Az aszteroidák vonzzák egymást a tömegükkel arányosan minden iterációban! (**5%**) Az aszteroidák az űrhajóra is gyakoroljanak gravitációs erőhatást minden iterációban! (**5%**)

Pluszpontok / Extra (Σ 15%)

- Legyenek egymást követő pályák, ahol egyre nagyobbak a tömegek, így egyre erősebb a gravitáció! Legyen pontszámláló is! +**10%**
- Legyen csillagokkal teli háttér, ami körülveszi a szemlélőt és az űrhajót! **+5%**
- A kamera űrhajót dinamikusan kövesse (nem statikus módon rögzített a kamera a térben az űrhajóhoz képest): **+5%**.

Ponthatárok: 0-39%: 1; 40-54%: 2; 55-69%: 3; 70-84%: 4; 85-100+%: 5

Fizikai szimuláció

Ahhoz, hogy a fizikai szimulációt kényelmesen lehessen megvalósítani, minden tárgynak kell rendelkeznie valamilyen sebességgel (“Velocity”), pozícióval (“Position”) és gyorsulással (“Acceleration”). Minden megjelenítés (draw) előtt az előző megjelenítés óta eltelt idővel arányosan érdemes frissíteni (vagy úgym. integrálni) a pozíciót:

Position += Velocity * delta_time;

Hogy a különböző erőhatásokat is lehessen szimulálni testek között, érdemes előbb hasonlóan a “Accelerationt”-t is frissíteni. Minden iterációs lépés legelején összegezni kell az egy testre ható erőket (“Force”) és gyorsulásba (“Acceleration”) konvertálni őket (“erő alkalmazása”):

Acceleration = Force / Mass;

… majd magát a “Velocity”-t is integrálni idő szerint, mielőtt a “Position”-t frissítenénk azzal, tehát:

Velocity += Acceleration * delta_time;

Egy testre ható gravitációs erőt a következőképpen lehet kiszámolni:

nagyság(Force) = G * tömeg1*tömeg2 / Távolsaguk²;

(*megjegyzés:* A G egy konstans. Ezt növelve, vagy csökkentve befolyásolható az erőhatások erőssége! A képletek csak segítségek, butított verzióik is elfogadhatóak!)

Sematikusan a draw() függvény belsejének egy része (ezt még lehet egyszerűsíteni is):

1. delta_time = …
2. background(0,0,0);
3. for each body **A**
    1. **A**.Acceleration.x = 0;
    2. **A**.Acceleration.y = 0;
    3. **A**.Acceleration.z = 0;
    4. for each body **B**
        1. if(**A** == **B**) continue;
        2. Force = ERŐ(**A** és **B** között, **A**-ból **B**-be mutató 2D vektor);
        3. **A**.Acceleration += Force / **A**.mass;
    5. **A**.Velocity += **A**.Acceleration * delta_time;
    6. **A**.Position += **A**.Velocity * delta_time;
    7. **A.draw();**