A számítógépes grafika alapjai
PPKE ITK

Tantárgykövetelmények 2017

A tantárgy követelményei:

Aláírás kétféleképpen szerezhető:

1. Órai munkával
2. Házi feladatok elkészítésével

Az év végi jegy komponensei:

1. sikeres félévközi munka (30%)
2. szóbeli vizsga az elméleti és gyakorlati anyagokból (70%)

Az órai munkával történő aláírás-teljesítés feltételei

- Valamennyi órán kiadott feladatok meg kell oldani és jegyzőkönyvezve elküldeni a gyakorlatvezetőnek legkésőbb a következő gyakorlat kezdéséig. A feladatok közül szúrópróba szerint kiválasztottakat személyesen is meg kell védeni a következő gyakorlaton (a forráskód így legyen elérhető).
- Kb 30 perces zárthelyi dolgozat az elméleti anyagból április 28-ai héten (előadáson vagy gyakorlaton), 1db pótzh alkalom a május 15-i (utolsó tanítási) héten.

Házi feladatokkal történő aláírás-teljesítés feltételei

Az aláírás megszerzéséért 3 db feladatok kell megoldani, melyek közül maximum 1 lehet 2D-s. A kiválasztott 3D-s feladatok között korlátozott számban lehetnek átfedések a hallgatók között, de a bemutatott megoldásoknak lényegesen el kell térniük. Lehetőség van személyes konzultációt követően egyéni, a kiadottaknál nem könnyebb feladatok megoldására is. A megoldások tetszőleges fejlesztői környezetben készülhetnek, az egyedüli általános feltétel:

- OpenGL használata (c vagy c++ implementáció)

Valamennyi feladatra e-mail-ben kell jelentkezni február 28-ig, és az elkészült megoldásokat személyesen kell bemutatni az alábbi gyakorlat időpontokban

- 1. feladat kész verziójának bemutatása: március 28
- 2. feladat kész verziójának és a 3. feladat előrehaladásának bemutatása : április 25
- 3. feladat kész verziójának bemutatása: május 9.

Oktató elérhetősége:
Benedek Csaba - benedek.csaba@itk.ppke.hu

Konzultációs lehetőségek:
Nagy Balázs - balazs.nagy.it@gmail.com

Megjegyzés: az értékelés során csak azokat az emaileket vesszük figyelembe, amit Benedek Csabának címeztek, vagy ccz-tek neki!

2D feladatok

1. Snake

Készítsük programot, amellyel a klasszikus kígyó játékot játszhatjuk.
A játékos egy, kezdetben a képernyő közepén induló, 5 hosszú kígyóval halad a képernyőn, minden másodpercben a legutoljára beállított irányba, ha a játékos nem változtatja meg azt egy megfelelő billentyű lenyomásával. A lehetséges haladási irányok balra, jobbra, felfelé, illetve lefelé. A játékmezőn véletlenszerű pozícióban mindig megjelenik egy tojás, amelyet a kígyóval meg kell etetni. Minden etetéssel eggyel nagyobb lesz a kígyó.
A játék célja, hogy a kígyó minél tovább elkerülje az ütközést a képernyő szélével, illetve saját magával.
Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (leghosszabb kígyót elérő) játékosról, ahol megjeleníti a játékosok nevét, idejét és hosszát. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

2. Fény-motor párbaj

Készítsünk programot, amellyel a Tronból ismert fény-motor párbajt játszhatjuk felülnézetből. Két játékos játszik egymás ellen egy-egy olyan motorral, amely fénycsíkot húz maga mögött a képernyőn. A motor minden másodpercben a legutoljára beállított irányba halad feltéve, hogy a játékos nem változtatja meg azt egy megfelelő billentyű lenyomásával.
A lehetséges haladási irányok balra, jobbra, felfelé, illetve lefelé. Az a játékos veszít, aki előbb neki ütközik a másik játékos fénycsíkjának vagy a képernyő szélének.

Legyen lehetőség új játék kezdésére. A program folyamatosan jelezze az eltelt időt (másodperceket), és a játék végén kérje be a győztes játékos nevét. Ez alapján kezeljen egy toplistát a 10 legjobb (legtöbb idővel nyerő) játékosról, ahol megjeleníti a játékosok nevét és idejét. A toplista ne vesszen el az alkalmazásból való kilépéssel, továbbá legyen bármikor megtekinthető a programban.

3. Tankcsata

Készítsünk programot, amellyel a klasszikus tankcsata játékot játszhatjuk. A játékosok egy- egy tankot irányítanak a képernyő bal, illetve jobb szélén, és tudják a tankok csövének szögét, illetve a lövés erősségét állítani megfelelő billentyűzetkombinációval. A játékosok felváltva lőhetnek, amikor is a löveg ferdehajítással az erősségnek és szögnek megfelelően célba ér. Az a játékos győz, aki előbb eltalálja a másik játékos tankját (vagyis elég közel lő hozzá).
A játékot nem csak sík terepen, hanem dombos, hegyes vidéken is lehet játszani. Ennek érdekében a programban lehessen véletlenszerű domborzatmagasságot generálni, vagy beolvasni azt alkalmas fájlból. Arra mindig ügyeljünk, hogy ugyan a két tank lehet különböző magasságban, de alattuk mindig sík legyen a terep.
Legyen lehetőség új játék kezdésére, valamint az ellenfelek számának megadásával, játék szüneteltetésére, mentésére és betöltésére. A program folyamatosan jelezze külön-külön a két játékos gondolkodási idejét (azon idők összessége, ami az előző játékos lépésétől a saját lépéséig tart, ezt nem kell elmenteni).

4. Gilisztaüldözés

Készítsünk 2D giliszta üldözéses játékot OpenGL-ben. Két játékosunk és gilisztánk van, az egyik giliszta zöld, a másik piros. A giliszták teste szinusz hullám, amelyet vonallal, háromszög vagy négyszög szalaggal lehet felrajzolni. A giliszták feje sokszög. Az egyik játékos célja, hogy a piros giliszta érje utol a zöldet, a másiké, hogy megakadályozza ezt.

A giliszták három szinten mozoghatnak, a szintek között a két játékos egy-egy lifttel viheti át őket (Bal lift fel/le = q/a, a jobb lifté pedig o/l). A giliszták gilisztamozgással haladnak előre (nyújtáskor a farkuk vége, összehúzódáskor a fejük mozdulatlan, közben a hosszuk megközelítőleg állandó), és a falnál hátra arcot csinálnak. A giliszta akár egyetlen pontján is képes kapaszkodni, ha viszont egyetlen ponton sem támaszkodhat, leesik és az alsóbb szinten folytatja a mozgását a korábbi irányban.

A játék befejeződik, ha valamely gilisztát a lifttel agyonnyomjuk, vagy a piros utoléri a zöldet.

5. Kockás

A képernyőn a játékos egérrel egy négyzetet irányít. A négyzet mindig az egér által mutatott ponton van. A képernyőre folyamatosan érkeznek pontot érő objektumok, és kikerülendő, más színű objektumok. Ha a kikerülendő objektum metszi a négyzetet, a játék véget ér. Ha a négyzet érint egy pontot érő objektumot, a pontszám megnő, és a négyzet is nagyobbá válik, így egyre nehezebb kikerülni vele a veszélyes objektumokat. A játék során folyamatosan legyen látható az

eltelt idő.

6. Tetris

Vesszük a négy négyzetből oldaluknál való illesztéssel kirakható idomokat. Ilyen idomokat kell elhelyezni egy veremben, ahol az idom addig esik lefelé, amíg a már benne levő elemekkel alulról nem érintkezik. Ha a megérkezés pillanatában egy sor (ami n mező széles) minden mezője ki van töltve, az a sor eltűnik, és a felette levő sorok lejjebb esnek.

7. Holdraszállós

Egy holdkompot lehet irányítani billentyűzettel: jobbra, balra, és felfelé lehet gyorsítani. A kompra folyamatosan hat a gravitáció. A pályán található egy zöld színnel jelzett vízszintes szakasz. A játékot úgy lehet megnyerni, ha a komp úgy érinti meg ezt a mezőt, hogy a sebessége a földetérés pillanatában elegendően alacsony. Lehessen a gravitáció erejét változtatni. A manőverezéshez meghatározott mennyiségű üzemanyag áll rendelkezésünkre, mely a manőverezésekkor folyamatosan fogyik.

8. Falbontós

A játékos egy vízszintesen mozgó ütőt irányít, ami egy golyó visszapattintására alkalmas, amennyiben eltalálja. A pálya másik felében bontható négyzetek vannak, amik visszaütik az őket eltaláló golyót, de ettől eltűnnek. A feladat minden négyzet eltüntetése. Az ütő és a golyó helyzete a találkozás pillanatában befolyásolja a pattanás szögét.

9. Madaras csúzlis

Készítsünk madaras csúzlis programot, amelyben egy piros és egy zöld madár szerepel, valamint világos zöld talaj, a talajból kiálló tetszőleges színű kétágú csúzli, amely fekete színű széles gumival fogja körbe a kilövendő piros madarat. A csúzli tára, ahol a piros madár várja a sorsa beteljesülését, a 600x600 felbontású alkalmazói ablak bal felső sarkához képest 200 pixellel jobbra és 400 pixellel lejjebb van.

A madarak teste, szeme és szemgolyója ellipszis (nem kör!), csőrük, farktollaik és szemöldökeik háromszögek. Szárny és bóbita opcionális. A zöld madár fel-le repked akár szárnyak nélkül is, magassága az időben szinuszosan változik az ütközésig. A piros madarat csúzliból lehet kilőni a fizikai törvényeknek megfelelően, a csúzli rugóenergiáját teljes mértékben átveszi a madár induláskor. A kilövés folyamat azzal kezdődik, hogy az egér bal gombjának lenyomásával a madár ellipszis alakú testének belsejébe klikkelünk, amikor a madár a kurzorhoz ragad és követi az kurzor mozgását az egérgomb elengedéséig. A csúzli gumija végig szorosan feszül a madár fenekére. Amikor elengedjük a bal gombot a piros madarat is útjára bocsátjuk, azaz kirepül a csúzliból. Perdülettel és forgással nem kell foglalkozni. A piros madárra reptében a nehézségi erő hat, közegellenállás és felhajtóerő (nincs szárnya, amivel csapkodna) pedig nincs. Ha a piros és a zöld madár ellipszis teste ütközik (figyelem, két ellipszis ütközését pontosan kell számítani), mindketten megállnak a levegőben és a piros madár sárgává változik. Ha a piros madár kirepül a látható tartományból, automatikusan születik egy új a csúzliban fejjel lefelé a két ág "között".

A feladat megoldásához csak az előadáson eddig szerepelt OpenGL függvények használhatók, azaz: glBegin, glEnd, glColor, glVertex, glViewport.

3D feladatok

1. Molekulás

Jeleníts meg egy 3D-s molekulát. A molekulák atomjai legyenek tömör gömbök, a közöttük lévő kötések pedig egy-egy vékony henger. Az objektumok színei tetszőlegesek.
Legyen megvilágítás, árnyékolás. Lehessen egérrel forgatni, görgővel kicsinyíteni, nagyítani.
Extra: ha az egeret az egyes gömbökre húzzuk (vagy rákattintunk) legyenek világosabbak. (használni lehet: glGetFloat(GL_PROJECTION/MODELVIEW_MATRIX, ..) - az inverz transzformációhoz)

2. LEGO

Előre definiált testekből (kocka, téglatest, gömb ...) építős játék. Az elemek összeépítésével újabb testek jönnek létre, melyeket már egyben is fel lehet használni. A háromdimenziós térben lehessen az alkotást megcsodálni, mozgatni. A testek textúrát is kaphatnak. A megvilágítást és az árnyékolást is lehet használni.

3a. 3DRajz v1

3D rajzoló program. Kezdetben vagy egy 3D üres tér. Legyen lehetőség vonalvastagság és szín választására. Egér és billentyűkombinációk segítségével rajzoljunk a 3D-s térben. Legyen lehetőség színezésre is adott sík mentén.

3b 3DRajz v2 - A térben adottak véletlenszerűen elhelyezett pontok. Egér drag ‘n’ drop művelettel kössünk össze pontokat szakaszokkal (azaz megfogok egy pontot és miközben elhúzom róla az egeret már látom hogy a megfogott pontból egy egyenes indul. Az egérrel rámegyek egy másik pontra, majd felengedvén az egér gombját a szakasz vége ráugrik a legközelebbi pontra). Szükség van az inverz transzformációkra. Az aktuális direkt transzformációs mátrixokat a glGetFloat segítségével kérhetjük le.

4. 3D Panoráma

Egy kocka belső oldalfalaira textúrázzunk képeket. A kockát belülről nézve egy 360 fokos panorámát kapunk. A projekció mindenképp legyen perspektív. Egérrel forgatható.
Két féle forgatás lehetséges:
- x,y szerinti forgatás **(fapadosabb verzió)**: függőleges egérmozgatás - y irányba, vízszintes x irányba
- szélességi, hosszúsági fokok szerinti forgatás (gömbkoordináták - teta, fi)
  függőleges egérmozgatás **(expertebb verzió)**: fix irányba, vízszintes: teta irányba

Megj. Másik megközelítés: egy gömbre textúrázzunk rá egy nagy panoráma. Feladat u.a.

5. 3D fizikai animáció - itt több feladat is elképzelhető

Fizikai jelenségek szimulálása megvilágított 3D tömör testekkel.
A kiválasztott feladatnak
Lehetőségek:

5.1 labda pattogása egy dobozban

5.2 inga mozgása

5.3 rugó szimulálása: a rúgó lehet egy folyamatosan változó hosszúságú síklapocska is, amelyre rátextúrázzuk egy rúgó képét

A szintér minden esetben tartalmazzon megvilágítás demonstrációt is: diffúz és/vagy spekuláris objektumokat helyezz el a térben és ezeket világítsuk folyamatosan mozgó fényforrásokkal.

6. Tankos (2 feladatnak számít, ha szép és minden funkció működik)

Hozzunk létre egy virtuális világot és jelenítsük meg OpenGL-lel. A virtuális világ egy 1 négyzetkilométer földterület,ahol NxM piramis (N,M>4) és egy tank található. A föld zöld, diffúz, a piramisok sárgás-fehérek és spekulárisan csillogóak, a tankon terepszínű textúra látható és ugyancsak csillogó. A nap keletről süt, 20 fokot zár be a talajjal. A nap erős intenzitású fehér irányfényforrásnak tekinthető. Az égbolt (ambiens) fény sugársűrűsége konstans, türkiszkék (szép idő van).

A tank teste poligonokból áll, lövegtornya hengerszerű, az ágyú ugyancsak henger, amelynek a belseje üres, különben nem tudna lőni. A lövegtorony forgatható, az ágyú emelhető. A lánctalpak részletes kidolgozása mellőzhető (de nem tiltott). A tank lőtt, és most éppen a lövedék a cső végének közelében van. A lövedék hengerszimmetrikus, csúcsa hegyes, szürke, csillogó. A jobb láthatóság miatt a lövedék sugara lehet nagyobb mint az ágyúcső sugara.

A piramisok és a tank földre vetített árnyékát nem kötelező megjeleníteni (de lehet). Állítsa be a virtuális kamerát úgy, hogy a tank, a repülő lövedék és a piramisok egy része látható legyen.

Ezután az ellenséges tankját pályaanimációval mozgassa a [0,5] sec időintervallumon egy Bézier görbe mentén. A lövegtorony mindig az avatár felé fordul, a löveg emelkedik úgy, hogy a ferde hajítás éppen az avatárnál fejeződjön be. A t=1 sec pillanatban a tank lő, a golyó ferde hajítással elrepül. Az avatár szintén egy tank, de csak a környezetet látja (FPS). Az avatártank két botkormánnyal vezérelhető, amelyek a bal és jobb lánctalpakhoz vannak kötve. A botkormányokat q, illetve p billentyűkkel lehet előre dönteni (a botkormányok alaphelyzetbe állításához bármilyen megoldást kitalálhat, azt úgy sem fogjuk tesztelni). Az avatártank a piramisokon nem hatolhat át.

Az időlekérdezéshez a glutGet(GLUT_ELAPSED_TIME) függvény használható.

7 Madárkergető (2 feladatnak számít, ha szép és minden funkció működik)

Építsen fel egy virtuális világot libával és teherautóval, és jelenítse meg OpenGL környezetben.

A virtuális világ elemek:

1. Liba: paraméteres felületekből (pl. henger, kúp, ellipszoid stb.) álló lábak, test, nyak, fej, szemek, csőr és farok. A fej a nyakon forgatható, a lábak a csípőben, térben és a lábfejben előre/hátra forgathatók, a térd libalábként viselkedik, azaz előre hajlik. A liba az út szélén várja sorsának beteljesülését.
2. Teherautó: Diffúz/spekuláris fémes csillogású, poliéderekből (pl. téglatest, extrúdált poligon) összerakott karosszériából és textúrázott hengerekből áll. A teherautó az úton áll, indulásra készen.
3. Füves, lapos, zöld, diffúz terep, amelyen egy textúrázott aszfalt út vezet át.

A virtuális világot egy irányfényforrás (a Nap) világítja meg, a csirke és az úthenger a terepre és az útra koromfekete árnyékot vetnek. Ambiens fény van, de az sem világosítja fel az árnyékokat.

A virtuális kamerával külső szemlélőként követjük az eseményeket. Virtuális kamerával külső szemlélőként követjük az eseményeket

A következő lépésben valósítsunk meg animációt az elkészült helyszínen: két libánk van amelyek az út szélén várakoznak, és Játékos I. a 'c' billentyű lenyomására indíthatja őket egyesével. Az elindított liba átsétál az úttesten, és ha túlélte boldogan ácsorog a túloldalon. A sétálás közben a támaszkodó láb nem csúszkál (inverz kinematika), a test az ízületekben nem szakad szét (karakteranimáció). Ha a teherautó eltapossa (ütközésdetektálás), a liba kilapítottan az aszfalton marad örök mementóként (ütközésválasz).

Játékos II. a teherautót irányítja. Az 'f' billentyűvel az teherautót az orra irányába konstans gyorsulásra bírhatja, a 'b' hatására hátrafelé hasonló nagyságú gyorsulást állíthat be, az 'n' hatására a gyorsulás zérus (fizikai animáció: a sebesség és a pillanatnyi hely a dinamika alaptörvényéből számítandó).

A játékban három kamera van, az egyik a libák kiindulási helye körül köröz, a második a soron következő, fejét jobbra-balra forgató liba szemszögéből mutatja be a színteret, a harmadik pedig a teherautó vezetőüléséből (a sebességvektor irányában). A három kamera képét három nézetben (viewport) kell megjeleníteni az alkalmazói ablakban.

A fényforrás napként viselkedik, 10 másodperces periódusidővel vöröses árnyalattal felkel, fehéren delel, majd vörösesen nyugovóra tér (fényanimáció).