Mintakérdések a Számítógépes grafika és képfeldolgozás tárgy vizsgájára való felkészüléshez

Tartalomjegyzék

Korábbi évek vizsgái

2015 őszi félév

2015. december 21.

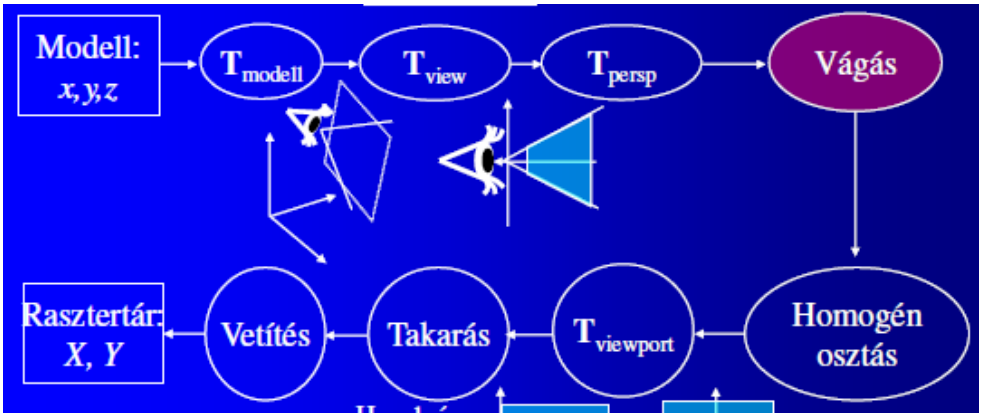
1. Egy lövedék a t=0 időpillanatban a **p1** pozícióból egyenes vonalú egyenletes mozgást végez, **v1** sebességgel. Szintén ebben az időpillanatban egy *R* sugarú **p2** pozícióban található gömb **v2** sebességgel egyenes vonalú egyenletes mozgást követve útnak indul. Írjon függvényt, mely **folytonos ütközésdetektálást** alkalmazva ellenőrzi, hogy a lövedék eltalálja-e a gömböt, és ha igen, akkor megadja a találat pontos időpontját is! Csak analitikus pontossággal megadott megoldásokat értékelünk, azokat közelítő levezetéseket nem fogadunk el! (15 pont)

Ún. folytonos ütközésvizsgálat, mely során nem csak diszkrét időpillanatokban figyelnünk, hanem a fizika folytonos egyenleteit felhasználva végig nyomon követjük az objektum pályáját. Ezzel akár az ütközés pontos pillanata is meghatározható.

2. Az alábbi OpenGL függvényeket paraméterezzük és lefuttatjuk, majd a z-buffer értékét visszaolvassuk és azt tapasztaljuk, hogy a visszaolvasott érték 0.5.

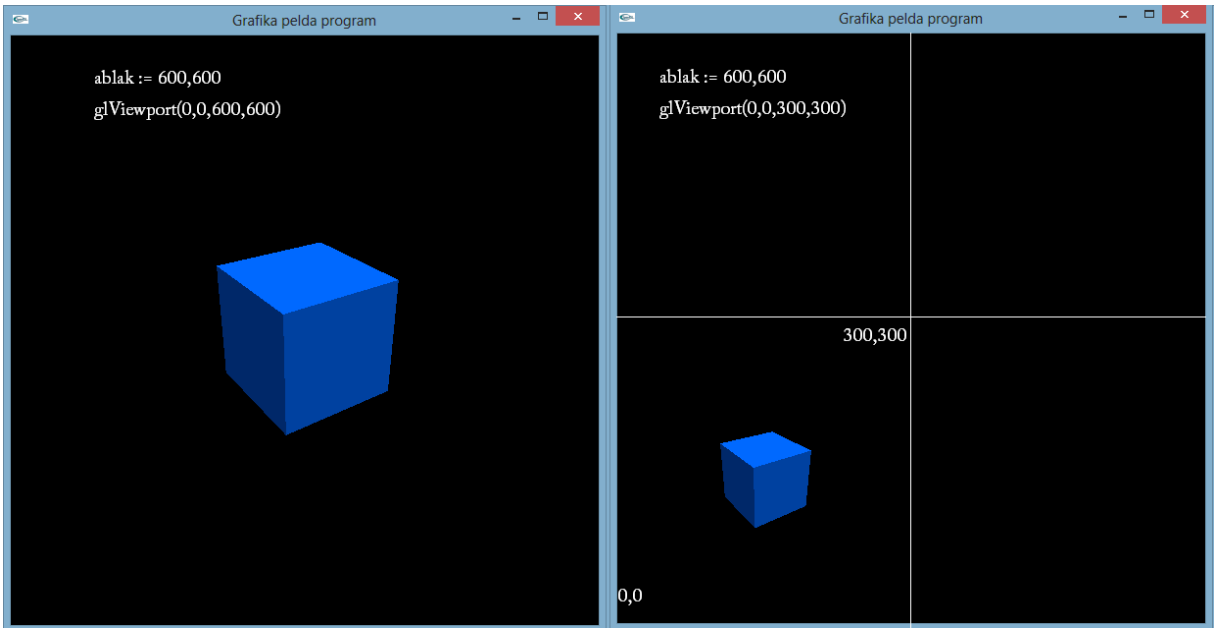
   Adja meg a (100, 100) pixelben található pont távolságát a kamera pozíciójától, ha a kamera a világkoordináták szerint az origóban helyezkedik el. (15 pont)

```
glViewport(200,200,0,0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90,1,1,2);
gluLookAt(0,0,0,0,0,-2,0,3,0);
glTranslatef(-10,-10,-10);
kirajzolás
visszaolvasás
```



1. **<u>Viewport</u>**:
   (100,100) * $T_{viewport}^{-1}$
   xw = (x+1)*(width/2) + off_x
   100 = (x+1)*(200/2) + 0
   1 = x + 1
   0 = x

   y = 0 ugyanígy
   ekk ixele (x,y, z-depth) alakú, tehát (0, 0, 0.5).



^glViewport megértéséhez egy példa - szóval a viewport határozza meg a képernyőd felbontását. Jelen esetben 200x200-as képernyőről van szó. 100,100 pixel középen van, menjünk tovább a leképezésben.

Most a kamera a 3D tér origójában található, és a -z irányba néz. A kamera lelki szemei előtt egy olyan négyzet (aspect ratio=w/h=1) tárul, amely a 3D világot egy 2D-oldalát látja. A 3D világ minden pixelére definiálva van egy kamerától való távolság, ami a 2D képernyő x,y koordinátája mellett a 3. koordináta (z, vagy z-depth) jelöl.

A z-buffer annyit jelent, hogy a legkisebb z távolságú objektum színével festjük be a képernyő x,y koordinátáját (mivel az van a legközelebb).
-> tehát (0,0, 0.5) a kamerához igazított világ 0,0 koordinátáján a kamerához legközelebbi pont 0.5-re található.

2. **<u>Homogén osztás</u>**:
   homogén szorzás kell: Koordinátánk (0,0,0.5,1) (4 = homogén koordináta)

3. **<u>Perspektív leképezés</u>**:

Az eredeti leképezés levágja a túl távol, vagy túl közel levő pontokat, valamint azokat, amik a kamera látószögén kívül esnek. Ennek az inverzét kell végrehajtanunk ugye, mert visszafele haladunk a pipe-ban.

fp, bp - font,back vágás

fov = field of view - látószög, asp = aspect ratio, width/height aránya

A feladat adataival a mátrix

1 0 0 0
0 1 0 0
0 0 -3 -1
0 0 -4 0 = $T_{perspektive}$ https://gist.github.com/gyng/8921328

A Transzponáltja (mert a transzformáció inverze kell): //ez nem az inverze a*a

Tperspektive*Tinverz=I

1 0 0 0
0 1 0 0
0 0 0 -0.25
0 0 -1 0.75 = $T_{perspektive}^{-1}$

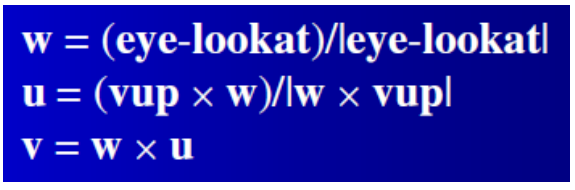
//ez hogy jön ki? https://gist.github.com/gyng/8921328, bár nem tudom általánosan hogy kell kiszámolni

szóval v * $T_{perspektive}$ = w, nekünk w= van meg, tehát w * $T_{perspektive}^{-1}$ = **v**

(0,0,0.5,1) * $T_{perspektive}^{-1}$ = (0, 0, -1, -0.125 + 0.75)

4. **Kamera transzformáció**:
Ez a transzformáció a 3D világunkat úgy mozgatja, hogy a kamera az origóban legyen és -z irányba nézzen. Tehát ennek az inverze az eredeti helyzetébe mozgatja a 3D világot.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -eye_x & -eye_y & -eye_z & 1 \end{pmatrix}^{-1} \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$\mathbf{w} = (\mathbf{eye}-\mathbf{lookat})/|\mathbf{eye}-\mathbf{lookat}|$$
$$\mathbf{u} = (\mathbf{vup} \times \mathbf{w})/|\mathbf{w} \times \mathbf{vup}|$$
$$\mathbf{v} = \mathbf{w} \times \mathbf{u}$$

```
gluLookAt(eye,lookat,vup)
gluLookAt(0,0,0, 0,0,-2, 0,3,0);
```

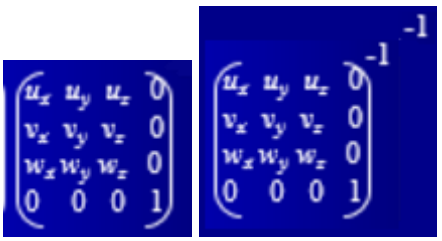
eye = 0 0 0
vup=0 3 0
lookat=0 0 -2

w = (0 0 2) ,norm = (0,0,1) **[forward]**
u = 0 3 0 x 0 0 1 = (3, 0, 0) norm = (1,0,0) **[up]**
3*1-0*0, 0*0-0*1, 0*0-3*0

v = (0,0,1) x (1,0,0) = (0,1,0) **[right]**

Azon látjuk, hogy hogy jó eredmények jöttek ki, hogy a három vektor egymással derékszögben áll (kamera nézeti iránya és a kamera lencséjére illesztett sík)

Eye = (0,0,0) ezért a kamera eltolás mátrixa egységmátrix lesz, nem módosít semmit
A képletben alapból egy mátrix inverze található, és mivel most a transzformáció inverzét hajtjuk végre:

= -vel kapjuk meg a modell világot 

$$\begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(0,0,-1, 0.625) * 0 0 0 1 = (0, 0, -1, 0.625)

5. **Model transzformáció**:
   Csak egy transzformáció van a modelben, translate:

```
glTranslatef(-10,-10,-10);
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -10 & -10 & -10 & 1 \end{pmatrix}$$

inverze kell, most nem számolom ki, eltoljuk 10-el minden koordinátában, szóval az inverze az lesz, hogy hozzáadunk 10-et

(0,0,-1, 0.625) -ból **(10, 10, 9, 10.625)** lesz

Biztos vagyok benne, hogy nem egy helyen rossz a megoldás, de szerintem Szirmay arra ad pontot, hogy látja hogy vágod milyen lépésekből áll a képszintézis.

3. **Bónusz**: A tomográfiában mit jelent a (filtered backprojection)? (max 6 pont)

2014 őszi félév

2015. január 19.

1. Egy porszívó robot háromszög alakú szobában takarít, amelynek sarokpontjai (x1, y1), (x2, y2), (x3, y3). A robot t=0 időpillantban indul el az (x0, y0) szobában lévő pontból (vx, vy) sebességgel. A fallal ütközhet, egyébként pedig egyenes vonalú egyenletes mozgást végez. Ha a fallal ütközik, akkor a falról rugalmasan visszaverődik (úgy, ahogy a

fény visszaverődik a tükörről). Írjon **folytonos ütközésdetektálást** alkalmazó programot, amely meghatározza a robot helyét tetszőleges t időpillanatban, feltételezve, hogy a robot nem repül, azaz a z=0 síkban mozog (20 pont).

2. Tekintsük azon pontok halmazát, amelyek az **f** helyvektorú ponttól mért távolsága megegyezik az **f** + **d** helyvektorú, **d** normálvektorú síktól mért távolsággal. Írja fel az alakzat egyenletét (3 pont)! Milyen alakzatról van szó? (1 pont) Írjon C++ függvényt, amely analitikusan kiszámolja az alakzat normálvektorát egy tetszőleges, az alakzathoz tartozó r pontban (6 pont).

   |r-f|=d*(r-(f+d)) //szerintetek ez jó? r:egy pont

   $|r-f|^2=d*(r-(f+d))^2$

   $(r-f)*(r-f)=(d*(r-(f+d)))^2$ //$d^2=1$

   $(x-fx)^2+(y-fy)^2+(z-fz)^2=r^2-2rf+2rd+f^2+2fd+d^2$

   parciális deriváltakból jön ki a normál vektor //nem biztos hogy jók ezeken(2dx,2dy,2dz) //vmi nem jó ebben

   ??? Ez milyen alakzat lesz? elvileg paraboloid jó ez chinchilla?

1. Feladat

Írjon OpenGL programrészletet, amely az alábbi textúrázott (glEnable(GL_TEXTURE_2D)), megvilágított (glEnable(GL_LIGHTING)) diffúz harmadrendű felületet jeleníti meg, mégpedig 2*100*100 háromszögre tesszellálva. A harmadrendű felület egyenlete:

$$xyz = 1, \quad 1 \leq x \leq 10, \quad 1 \leq y \leq 10.$$

A teljes textúrát rá kell húzni a felületre, amely a felület diffúz visszaverődési tényezőit tartalmazza. Feltételezheti, hogy a textúrát és a transzformációkat már feltöltöttük a GPU-ra, a szempozíció z koordinátája 1-nél nagyobb, a lookat pont az origo, a mélységbuffer algoritmus engedélyezve van, a hátsólap eldobás viszont tiltva, valamint a fényforrásokat már beállítottuk (glLight). Megoldandó viszont az anyagtulajdonságok beállítása (2p), a textúrázási környezet beállítása (1p) és a csővezeték táplálása (7p). A normálvektort analitikusan kell számítani.

A javasolt OpenGL függvények: glBegin, glEnd, glVertex, glNormal, glTexCoord, glMaterial, glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, ...);

Analitikus geometria

Pont: P=(x1,y1,z1)
Távolsága a síktól: ~~dist = n*P + d~~ (skaláris szorzat)
jav: dist = n*P / |n|+ d (osztás n hosszával)
dist = (a*x1+b*y1+c*z1)/sqrt(aa+bb+cc) + d
Vetített pont: ~~P'=P - dist*n = (x1 - a*dist, y1 - b*dist, z1 - c*dist)~~
jav: P'=P - dist*n / |n|= (x1 - a*dist / |n|, y1 - b*dist / |n|, z1 - c*dist / |n|)
P' = Írja fel azon művelet mátrixát, amely egy $ax+by+cz+d=0$ egyenletű síkra merőlegesen vetít.

Mátrix:

1-a^2	-ba	-ca	0
-ab	1-b^2	-cb	0
-ac	-bc	1-c^2	0
-ad	-bd	-cd	1

javítás egyszerűbben: a megoldás akkor jó, ha a normálvektor egység hosszú. Ha nem az:
n' = n / |n| a' = a / sqrt(aa+bb+cc) stb… erre már jó a megoldás

1.

2. Írja fel három pontra illeszkedő sík egyenletét az euklideszi és a projektív térben.

3. Bizonyítsa be, hogy a projektív sík egyenesének $aX+bY+ch = 0$ egyenlete összhangban van a projektív geometria azon axiómáival, hogy „két különböző egyenes egy pontba metszi egymást", és hogy „két pont meghatároz egyt egyenest".

   A sík normál vektorát megkaphatjuk a három pont közti vektorok vektoriális szorzataként,
   legyen A(x0, y0, z0), B(x1, y1, z1), C(x2, y2, z2)
   ekkor AB=(x1-x0, y1-y0, z1-z0), pl BC = … vektoriális szorzat a köv mátrix detjával számolható
   |i, j, k|
   |.AB .| = [...] * A + [...] *B + [...]*C
   |.BC. |

   A sík egyenlete pedig az egyik ismert pont (pl A) alapján $-D = Ax_0 + By_0 + Cz_0$

4. Bizonyítsa be, hogy a homogén lineáris transzformációk a konvex kombinációkat konvex kombinációkká képezik le.

5. Írja fel azon síkpontok mértani helyének egyenletét, amelyek egy egyenestől és egy ponttól ugyanolyan távolságra vannak. Milyen alakzat ez? Terjessze ki az alakzatot a projektív síkra. Mik az ideális pontjai.

   parabola $\frac{|n_1x + n_2y + c|}{\sqrt{{n_1}^2 + {n_2}^2}} = \sqrt{(x-u)^2 + (y-v)^2}$

6. Írjon C függvényt, amely egy egyenes és egy pont távolságát kiszámítja.

```
// Meghatározza egy A és B pont által meghatározott egyenes, és a P pont távolságát
float distance(Vector A, Vector B, Vector P) {
	float a, b, c;
	a = B.y - A.y;
	b = A.x - B.x;
	c = B.x*A.y - A.x*B.y;
	return (fabs(a*P.x + b*P.y + c) / sqrt(pow(a, 2) + pow(b, 2)));
}lf
```

7. Írjon C függvényt, amely két térbeli egyenes távolságát kiszámítja.

8. Írjon C függvényt, amely egy implicit egyenletével adott egyenes, és két végpontjával adott szakasz metszéspontját kiszámítja.

	Síkban??

	egyenes: $ax + by + c = 0$

	szakasz: P1 → P2

```
int intersect(float a, float b, float c, Vector p1, Vector p2, Vector* res)
{
	//  p1 rajta van az egyenesen
	if  (fabs(a*p1.x + b*p1.y - c) < epsilon) {
		if  (fabs(a*p2.x + b*p2.y - c) < epsilon) {
			// végtelen sok megoldás (kivéve, ha p1 = p2)
			*res = p1;
			return -1;
		} else {
			*res = p1;
			return 1;
		}
	} else {
		if  (fabs(a*p2.x + b*p2.y - c) < epsilon) {
Q			*res = p2;
			return 1;
		} else {
			// ugyan azon az oldalon vannak
			if ((a*p1.x+b*p1.y+c<0)== (a*p2.x+b*p2.y+c<0))
				return 0;
			else {
				// ??
				res->x =
				res->y =
				return 1;
			}
		}
	}
}
```

9. Írja fel a projektív sík egyeneseinek implicit (azaz nem paraméteres) egyenletét homogén koordinátákban Bizonyítsa be, hogy a sík minden invertálható homogén lineáris transzformációja ezt az egyenest egyenesre képezi le. Melyek azok a transzformációk, amelyek az egyenest önmagára képezik le. Mi történik az egyenes normálvektorával?

10. Lehet-e egy affin – azaz párhuzamos egyeneseket párhuzamos egyenesekbe leképző – transzformáció mátrixának negyedik oszlopa [0, 0, 0, 2]?

	hógyne

11. A projektív tér síkjainak homogén lineáris transzformációi. A projektív tér síkjának egyenlete. A transzformáció mibe viszi át a síkot (feltételezheti, hogy a transzformációs mátrix invertálható), bizonyítás. Mi történik a sík normálvektorával. Mi ennek a következménye az OpenGL működésére.

12. Tekintsük a projektív térben a [1, 2, 3, 4] és [-4,-3,-2,-1] homogén koordinátákkal azonosított végpontok konvex kombinációiként előálló szakaszt! Mekkora a szakasz hossza?

    A=(0.25, 0.5, 0.75)
    B=(4, 3, 2)
    AB = sqrt(3.75^2 + 2.5^2 + 1.25^2)

13. Írja fel azon homogén lineáris transzformáció mátrixát, amely egy síkbeli pontot az *xc, yc* vetítési középponttal az *ax+by+c=0* egyenletű egyenesre vetít.

14. Tekintsük a következő homogén lineáris transzformációs mátrixot (a transzformálandó pontot a mátrix bal oldalára kell írni):

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

    Mit csinál a transzformáció? Mi keletkezik a transzformáció után a [2, 1] és [-1, 1] pontokat összekötő szakaszból.

15. Adott a következő homogén lineáris transzformáció (a transzformálandó pont helyvektorát sorvektornak kell ttekinteni).

$$\begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 8 & 6 & 4 & 2 \end{bmatrix}$$

Adja meg azon ponthalmaz egyenletét, amelyre ez a transzformáció a *8x+6y+8z + 6 = 0* egyenletet kielégítő ponthalmazt leképezi! Először írja le a megoldás menetét, azután végezze el a szükséges számításokat.

16. Adott két pont Descartes koordinátákkal: (3,6), (-2,5). Adja meg erre a két pontra illeszkedő egyenes ideális pontját homogén koordinátákban!

17. Bizonyítsa be, hogy ha egy invertálható transzformációs mátrix 4. oszlopa [0, 0, 0, 1], akkor a transzformáció **affin**, azaz párhuzamos egyeneseket párhuzamos egyenesekre képez le (a transzformálandó pont homogén koordinátáit sorvektornak tekintjük és a mátrixszal jobbról szorozzuk)!

18. Írja fel a 3D projektív tér összes ideális síkjának és ideális egyenesének egyenletét z(az ideális térelem csak ideális pontokat tartalma). Hány ideális sík és egyenes van?

19. Írja fel a projektív sík egy körének egyenletét! Mi lesz abból a körből, amelynek középpontja egy ideális ponton van? Segítség: a kör egyenlete homogén koordinátá*ban annak analógiájára, ahogyan az egyenes egyenletét a síkban, és a sík egyenletét a térben bevezettük.*

20. Tekintsük a projektív tér [*a,b,c,d*] homogén négyessel megadott síkját. Írja fel ezen sík ideális egyenesének paraméteres egyenletét!

21. Írjon C++ függvényt, amely a bemeneti paraméteréül a projektív sík két egyenesét kapja, kiszámítja a két egyenes metszéspontját, a metszéspontot vetíti az origó középpontú, egység sugarú körre, és a vetület Descartes koordinátáit adja vissza.

22. Adott a következő **r** -> **r'** 2D transzformáció, ahol **r, r'** és **a, b, c** a sík vektorai:

$$\mathbf{r'} = [(\mathbf{a} \cdot \mathbf{r})/(\mathbf{c} \cdot \mathbf{r}), (\mathbf{b} \cdot \mathbf{r})/(\mathbf{c} \cdot \mathbf{r})].$$

Mibe vihet át ez a transzformáció egy szakaszt?

23. Adott a következő **r** -> **r'** 2D transzformáció, ahol **r, r'** és **b** a sík vektorai:

$$\mathbf{r'} = \mathbf{r}/|\mathbf{r}| + \mathbf{b}.$$

A képletben $|\mathbf{r}|$ az **r** vektor abszolút értékét jelenti**.** Mibe vihet át ez a transzformáció egy szakaszt?

egységkörbe, és eltolja b-vel

Geometriai modellezés

24. Hogyan definiálható a B-spline és milyen tulajdonságai vannak.

Sünis Könyv 58.
A B-spline olyan. görbeleírás, amely a lokális vezérelhetőség és a símaság (deriválhatóság) között ad kompromisszumot. Approximációs görbe, tehát a vezérlőpontokon jellemzően nem halad át. (Az első és az utolsó vezérlőponton sem) Általános esetben a szomszédos csomópontok távolságára nincs megkötés
A görbe bázisfüggvényeit úgy származtatjuk, hogy kiindulunk olyan bázisfüggvényekből, amelyek a hozzájuk tartozó csomópontintervallumon Bi=1 értéket vesznek fel, egyébként pedig nullát. Ezután a B-spline fokszámának megfelelő számszor lineáris simítást végzünk. Ha egy B-spline fokszáma n. akkor egy vezérlőpont a görbe n+1 szegmensére van hatással, tehát a fokszám növelésével a lokális vezérelhetőség csökken.

25. Mi a NURBS.

Non-Uniform Rational B-Spline, a B-Spline egy kezelhetőbb változata. A vezérlőpontokhoz még egy w súlyt is rendelünk, ennek növelésével a görbe az adott pontban egyre jobban csúcsosodik. Előnye, hogy a kúpszeletek tökéletesen leírhatók legalább harmadfokú NURBS-ökkel, hátránya hogy (hacsak homogénkoordinátákban nem számolunk), osztásokra is szükség van a görbe kirajzolásához.

egy adott pont: $$r(t) = \frac{\sum w_i B_i^{NUBS} r_i}{\sum w_j B_j^{NUBS}} = B_i^{NURBS} r_i$$

Bázis fv $$B_i^{NURBS} = \frac{w_i B_i^{NUBS}}{\sum w_j B_j^{NUBS}}$$

26. Bizonyítsa be, hogy a Bézier görbét érinti az első két kontrolpontokra fektetett egyenes.

27. Hogyan lehet előállítani egy Bézier görbét NURBS segítségével?

28. Adja meg a kvadratikus felületek általános definícióját. Milyen konkrét tagjai vannak ennek a családnak.

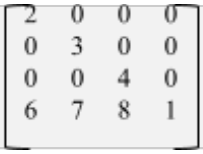
29. Rajzolja fel a szárnyas ér adatstruktúrát, és írjon programot, amely egy lapnak kiírja az összes csúcsát.

30. Mik az Euler operátorok és miért van rájuk szükség.

minden poligonra: lapok + csúcsok = élek + 2

31. Írjon C++ nyelven egy CSG fát megvalósító osztályt.

32. Adott egy 9 vezérlőpontra illeszkedő 3x3-as Bézier felület. A vezérlőpontok rendre p11,…,p33. Adja meg a felület normálvektorát a (0, 0.5) paraméterértékek mellett! Hogyan változik az előző feladatban kiszámolt normálvektor, ha a vezérlőpontokat a



homogén lineáris transzformációs mátrixszal megszorozzuk?

33. Számítsa ki a harmadrendű (azaz MÁSODFOKÚ) uniform B-spline görbe bázisfüggvényeinek értékét a 0.8 értékre! *Segítség: a Cox – deBoor módszert érdemes alkalmazni, és ellenőrzésképpen meg lehet nézni a konvex burok tulajdonságot.*

34. Adott egy $\mathbf{r}\mathbf{A}\mathbf{r}^T = 0$ ($\mathbf{r}$=(x,y,z,1) sorvektor és $\mathbf{A}$ 4x4-es mátrix) kvadratikus felület, valamint azon egy $\mathbf{p}$ pont. Írja fel a felület normálvektorát a $\mathbf{p}$ pontban.

35. Adott egy Bézier felület $\mathbf{c}00,\ldots\mathbf{c}44$ vezérlőpontokkal. Írja fel a felület normálvektorát a (0.5,0.5) paraméterértékeknél.

2D képszintézis

36. Írjon erősen emelkedő szakaszt rajzoló programot, a Bresenham algoritmus alapján.

37. Írjon DDA szakaszrajzoló függvényt,
    `DDADraw(int x1, int y1, int x2, int y2, int R, int G, int B)`
amely bármilyen **pozitív meredekségű** szakaszt képes felrajzolni. Használhat lebegőpontos aritmetikát. Egy rasztertárbeli pixel színét a
    `PutPixel(int x, int y, int R, int G, int B)`
függvénnyel változtathatja meg.

38. Írjon programot, amely egy szakaszt egy konvex sokszögre vág.

39. Írjon programot, amely egy tetszőleges sokszöget egy konvex sokszögre vág.

40. Írjon háromszög kitöltő programot, amely háromszögenként 3 osztást és fixpontos összeadásokat tartalmazhat.

41. Írjon C++ programot, amely eldönti, hogy egy p1, p2, p3 és egy v1, v2, v3 síkbeli háromszögnek van-e közös része. Feltételezheti, hogy a vektor összeadás (+), kivonás (-), skaláris szorzás (*), és 90 fokkal való elforgatás ([x,y] -> [-y,x], .Rot90) műveletek a vektorokra (Vec) rendelkezésre állnak. Bónusz: általánosítsa a programot két konvex sokszögre.

42. Írjon NURBS görbe megjelenítőt, amelyben felhasználhatja a NUBS bázisfüggvényeket.

43. Írjon háromszögvágó programot, amely a csúcsaival definiált háromszögből eltávolítja azokat a pontokat, amelyek a paraméterként kapott ymin érték alatt vannak, és visszatérési értékében jelzi, hogy maradt-e egyáltalán valami a vágás után. Az eredményt a három-szögek csúcsait tartalmazó tömbben adja vissza. A megoldás során feltételezheti, hogy rendelkezésre áll egy Intersect eljárás, ami két egyenes metszéspontját kiszámítja.

44. Írjon programot, amely egy paraméterként kapott, vezérlő pontjaival (x[], y[], npoint) definiált Bézier görbét felrajzol. A vezérlőpontok koordinátái képernyőkoordinátákban (pixel egység) értendők. A rajzoláshoz a glBegin(GL_LINE_STRIP), glVertex2d(xi, yi) és glEnd( ) utasításokat használja! A képszintézis gyorsítása érdekében ne rajzoljon, ha a Bézier görbe befoglaló téglalapjának és az XRES, YRES méretű nézetablaknak nincs közös része. Feltételezheti, hogy az adott programozási nyelven az „n alatt az i” és a hatványozás beépített műveletek.

45. Írjon 2D aláosztott görbe (Catmull-Clark subdivision curve) rajzoló rutint C++ nyelven. A függvény bemeneti változói: `int n` a csomópontok száma, `Vector2 points[]` a 2D Vector2 típusú csomópontok tömbje, `int level` az aláosztás szintje (0 az aláosztatlan görbét jelenti). Feltételezheti, hogy a Vector2-ben a szokásos vektorműveletek operátor overloaddal már megvalósítottak. A megjelenítést GL_LINE_STRIP segítségével OpenGL-lel valósítsa meg, a felhasználható függvények (glBegin, glVertex2d, glEnd).

46. Írja fel azt az időfüggő homogén lineáris transzformációs mátrixot, amely egy [2, 0] sebességű autó 2 egység sugarú kerekét mozgatja és forgatja. Az animáció kezdetén a kerék transzformációs mátrixa az egységmátrix.

47. Írjon szakaszvágó programot, amely a szakaszból eltávolítja azokat a pontokat, amelyek a paraméterként kapott ymin érték alatt vannak és visszatérési értékében jelzi, hogy maradt-e egyáltalán valami a vágás után.

48. Rajzolja fel annak a hardvernek a vázlatos kapcsolási rajzát, amely minden órajelciklusra egy erősen emelkedő (x2 > x1, y2 > y1, y2-y1 > x2-x1) egyenes szakasz újabb pixelének x, y címét előállítja. Milyen értékekkel kell inicializálni a tárolókat?

49. Területkitöltés geometriai reprezentáció alapján (belső pont azonosítása, naív algoritmus, gyorsítási lehetőségek, inkrementális elv, aktív él tábla, él tábla).

50. Területek vágása ablakra (algoritmus, több részre eső területek kezelése)

51. Írja fel egy 2D autókerék transzformációs mátrixát az idő függvényében, ha az autó az *x* teng
52. ely irányába halad *v* sebességgel, a talaj y koordinátája zérus, a kerék sugara *r*. A lokális modellezési-koordinátarendszerben a kerék tengelye az origóban van, és *t=0*-ban az *x* koordináta ugyancsak zérus. A kerék a talajon gördül.

53. 2D grafikus rendszerek felépítése (bemeneti és kimeneti csővezeték felépítése és működése, OpenGL és a GLUT hol jelenik meg)

54. Adott egy zárt töröttvonal a síkban. A csúcsok koordinátái a px[n], py[n] tömbökben vannak, a csúcsok száma az n változóban. Írjon C++ függvényt, amely egy (x, y) pontról eldönti, hogy a töröttvonal által definiált sokszög belsejében van-e.

55. Kedves barátnője (barátja) radiológusként dolgozik a kórházban, és naponta nagyon sok röntgen képről kell eldöntenie, hogy a képen látható csont törött-e vagy sem. Milyen szerencse, hogy tanulta a számítógépes grafika területelárasztó kitöltő algoritmusát (floodfill), így tud segíteni neki! Készítsen karácsonyi ajándékként egy C++ függvényt, amely a kép alapján eldönti, hogy azon egyetlen csont található, vagy több különálló csontdarab (azaz törött-e a csont vagy sem)! Az `NxM` felbontású szürkeárnyalatos kép az `unsigned char image[N][M]` tömbben van. Tudjuk, hogy a levegő a 0..10 tartományban, a lágyszövet 10-100 tartományban, a csont pedig a 100-200 tartományban lévő szürkeségi szinteknek felel meg. A 200-256 tartományban nincs érték. Két csontdarabról akkor mondjuk, hogy különállóak, ha közöttük legalább egy pixelnyi nem csont anyag található (azaz a két tartomány csontszínű pixelei még sarkaikban sem érintkeznek). A függvény megkapja az `image[N][M]` tömböt és az `int N,M` felbontási értékeket és egy `int` értékkel tér vissza, ami akkor 1, ha több különálló csont is látható a képen, egyébként 0.

56. Az erdészetnél vállalt munkát, ahol tölgyfaerdőt telepítettek egy szabályos rács szerint N sorban és M oszlopban. Sajnos távvezeték építés miatt az erdőből fákat kell kivágni a távvezeték egyenes vonala alatt. Tudjuk, hogy a távvezeték melyik sorban/oszlopban lép be az erdőbe és melyikben távozik. Valamilyen szakaszrajzoló algoritmus adaptálásával írjon C++ programot, amely a sztandard outputra kiírja, hogy mely sorban és oszlopban lévő fákat kell kivágni. Természetesen, csak minimális számú fát szeretnénk kivágni, ezért egy kivágott fának legfeljebb két, ugyancsak kivágott szomszédja lehet (két fa szomszéd, ha sor és oszlopszámuk külön-külön legfeljebb eggyel térnek el egymástól!

57. Tőzsdeügynöknek csapott fel, és a részvényekről egy nagyon nagy méretű adatbázissal rendelkezik. Egy részvényt a sorszámával, a hozammal (0..1000) és a törzstőkével (0..1000) jellemezhetünk. A kuncsaftok azt a kérdést tehetik fel, hogy melyek azon részvények sorszámai, amelyek hozama az általuk megadott hozamtól legfeljebb 10-zel tér el, ÉS az általuk megadott törzstőkétől ugyancsak legfeljebb 10-zel tér el. Szerencsére még emlékszik a sugárkövetés szabályos rács felosztó eljárására, így tőzsdeügynökként is megállja a helyét. Írjon olyan programot, amely egy előfeldolgozási lépésből és egy lekérdező lépésből áll C++ nyelven. Az előfeldolgozási fázis időbonyolultsága nem érdekes, úgyis csak egyszer kell végrehajtani. Viszont a nagy piaci verseny miatt csak olyan lekérdező lépés engedhető meg, amelynek válaszideje nem nő a részvények számával abban az esetben, ha a részvényadatok valószínűségi eloszlása egyenletes (olyan megoldások, ahol a lekérdezési idő a részvények számával nő, különösen, ha azzal egyenesen arányos, nem értékelhetők ezen a ZH-n sem). A bemeneti adatok a részvények tömbje és száma. A lekérdezési fázis bemenete a kívánt hozam és tőke, a kimenete pedig a kívánalomnak eleget tevő részvények sorszámainak tömbje. A megoldás során a tömbök allokálásával nem kell foglalkozni.

58. A katasztrófa elhárításnál kapott munkát, és éppen a Tisza vízállásjelentését (érték a tengerszinthez képest méterben) hallgatja, ahol gátszakadás történt. Nagyon gyorsan meg kell mondania, hogy milyen területeket fog elönteni a víz. Rendelkezésére áll a körzet digitalizált magasságtérképe (`float height[N][M]`), ahol minden pixelben a pont tengerszint feletti magassága található. Milyen szerencse, hogy tanulta a számítógépes grafika területelárasztó kitöltő algoritmusát (floodfill), amivel meg tudja oldani a feladatot, és ezért nem kell januártól új állást keresnie! Írjon C++ függvényt, amely a sztenderd kimenetre kiírja azon térkép pixelek koordinátáit, amelyeknek megfelelő pont víz alá fog kerülni. A függvény bemenete a gátszakadás helye ugyanabban a koordinátarendszerben, amelyben a magasságtérképet is kapjuk, és a Tisza

vízállása. Feltételezhetjük, hogy a térkép határain biztosan nem terjed tovább az ár, és hogy a Tisza vízszintje állandó.

59. Pistike egy labirintusban szeretné megtalálni a kijáratot. Segítsen neki! Tételezze fel, hogy a labirintus térképét Pistike már beszkennelte, és a kapott képet a számítógép memóriájába az `unsigned char * terkep` címre betöltötte. A kép bináris, egy pixelt egy `unsigned char` reprezentál, a kép felbontása 400x400, a fekete pixelek (szín = 0) a folyosókat, a fehérek (szín = 1) a falakat jelzik. A bejárat a kép [10,10], a kijárat pedig a [390,390] ponton van. Írjon C programot, amely eldönti, hogy el lehet-e a jutni a bejárattól a kijáratig (a tényleges utat nem kell meghatározni, csak azt, hogy lehetséges-e eljutni a célig) (Ha nincs ötlete, akkor gondoljon a területkitöltő algoritmusokra).

60. Írjon C++ függvényt, amely egy paraméterként kapott x, y középpontú ellipszist felrajzol. Az ellipszis főtengelyeinek hossza *a*, illetve *b*. Az *a* hosszúságú főtengely a koordinátatengelyekkel α szöget zár be. A középpont és a tengelyhossz képernyő koordinátákban (pixel egység) értendők. A rajzoláshoz a glBegin(GL_LINE_STRIP), glVertex2f(xi,yi) és glEnd( ) utasításokat használja! Az ellipszis töredezettségét elkerülendő, egy rajzolt szakasz maximális vízszintes és függőleges mérete a 10 pixelt nem haladhatja meg. A színbeállítással és az ablak megnyitásával nem kell foglalkozni.

61. Írjon görbesimító programot C++-ban OpenGL és GLUT felhasználásával. Az windows ablak háttérszíne kék, felbontását szabadon megválaszthatják. A program egy "pontok.txt" nevű ascii fájlt olvas be, amelynek első sorában egy int van (a pontok száma), amit ennyi sor követ, minden sorban két float számmal, amelyek a pontok x,y világkoordinátáit tartalmazzák. A program ezen csúcspontokra egy zárt töröttvonalat illeszt és egy (0,0),(100,100) sarokpontokkal definiált ablakú kamerával lefényképezi, az eredményt a képernyő windows ablakában megjeleníti. A zárt töröttvonalat fehér színnel kell felrajzolni. Ha a felhasználó a bal egér gombbal ráklikkel a töröttvonalra, a program Catmull-Clark algoritmussal simít egyet a töröttvonalon. Jobb egérklikk pedig csökkenti a simítás mértékét.

62. Írjon görbesimító programot C++-ban OpenGL és GLUT felhasználásával. Az windows ablak háttérszíne kék, felbontását szabadon megválaszthatják. A program egy "pontok.txt" nevű ascii fájlt olvas be, amelynek első sorában egy int van (a pontok száma), amit ennyi sor követ, minden sorban két float számmal, amelyek a pontok x,y világkoordinátáit tartalmazzák. A program ezen csúcspontokra egy zárt töröttvonalat illeszt és egy (0,0),(100,100) sarokpontokkal definiált ablakú kamerával lefényképezi, az eredményt a képernyő windows ablakában megjeleníti. A zárt töröttvonalat fehér színnel kell felrajzolni. Ha a felhasználó a bal egér gombbal ráklikkel a zárt töröttvonal által határolt területre, a program Catmull-Clark algoritmussal simít egyet a töröttvonalon. Jobb egérklikk pedig csökkenti a simítás mértékét. A maximális simítás mértéke legalább 3, azon túl korlátozható. A zárt töröttvonal által határolt terület pontjai azok, amelyekhez a végtelenből érkezve páratlanször lépjük át a töröttvonalat.

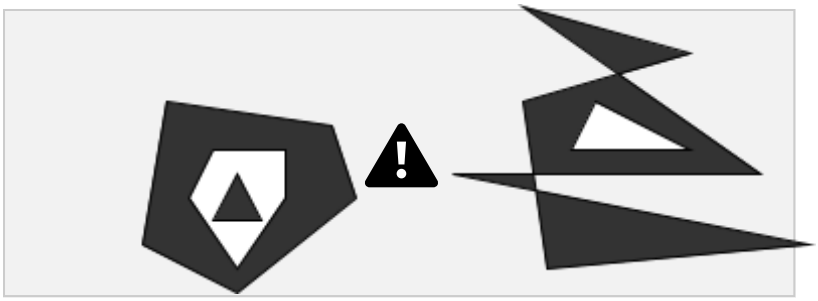
63. Írja meg a glutility csomag `gluOrtho2D(double Left, double Right, double Bottom, double Top)` függvényét. Az implementációban a `void glMultMatrixd(const double m[4][4])`függvényt használhatja fel.

64. Írjon C++ függvényt, amely egy zárt sokszögre eldönti, hogy rámutatással kijelöltük-e. Akkor tekintjük a sokszöget kijelöltnek, ha a (x,y) kijelölési pont középpontú, 10 egység oldalhosszúságú téglalap tartalmazza a sokszögből egy részt (ez a rész bármilyen kicsiny lehet, a vizsgálatot tehát nem elég egész koordinátákra elvégezni). A függvény deklarációja:

```
struct Vertex { float x, y; };
bool Talalat(float x, float y, int n, Vertex * p);
```

ahol `n` a csúcspontok száma, `p` pedig a csúcspontok tömbje.

65. Írjon C++ függvényt, amely egy többszörösen összefüggő sokszöget kitölt (akkor nevezünk egy sokszöget többszörösen összefüggőnek, ha a határát több zárt töröttvonallal lehet megadni). A függvény bemenete az egyes töröttvonalak csúcsszáma, illetve a töröttvonalak csúcspontjai (2 dimenziós tömb). Felételezheti, hogy a művelet előtt az ablakra vágtunk. Az ablak sarokpontjai (0,0), (1000,1000). Egyetlen pixelt a SetPixel(x,y) függvénnyel színezhet át a sokszög színére. Példák kitöltött, többszörösen összefüggő sokszögekre:



Illuminació

66. Írjon egy BRDF osztályt, ami diffúz és Phong szerinti illuminációs modellt valósít meg.

67. A fény egy ideális tükörre V irányból érkezik, a tükör normálvektora N. Milyen irányba halad tovább?

68. A fény egy ideális fénytörő felületre a V irányból érkezik, a tükör normálvektora N. Milyen irányba halad tovább, ha a törésmutató n?

69. Egy tökéletesen sík felületre érkezik a fény [3,4,5] irányból. Milyen irányban verődik vissza, ha a sík normálvektora [-3,8,1], és a fény a felületre merőlegesen polarizált?

70. Írjon egy BRDF fügvényt, amely a fény, nézeti és normálvektorokat kapja meg és a BRDF tényezőt adja vissza RGB hullámhosszokon. A függvény a diffúz és az eredeti Phong szerinti illuminációs modellt valósítja meg.

71. * Oldja meg az előző feladatot úgy, hogy a tárolt adatok, a kd hullámhosszfüggő diffúz albedó, a ks a hullámhosszfüggő spekuláris albedó jelenti merőleges megvilágítás esetén, az n spekularitási tényezőt (fényesség illetve shininess) tekintse globális változóknak. Hogyan kellene kiszámítani ks-t a beesési szög, a törésmutató a komplex törésmutató alapján?

72. Adott egy irányfényforrás, amely a [3,4,0] irányból világít, intenzitása [200, 10,100]. Miyen színűnek látja a megfigyelő a z=0 sík [3,5,0] pontját, ha ő a [7,5,3] pontban van, és a felület Phong-Blinn illuminációs modell szerint veri vissza a fényt. A spekuláris visszaverődési ks = [25/9,25/9,25/9], a fényesség (shininess) 2.

73. Egy (0,0,1) normálvektorú piros műanyag (nem fém!!!) felületről tudjuk, hogy Phong modell szerint veri vissza a fényt. A (1,1,1) irányból érkező (1,1,1)W/st/m^2 sugársűrűségű (R,G,B hullámhosszain) megvilágítás hatására a piros hullámhosszán a (1,1,1) irányból ránézve a felületre 0.3 W/st/m^2 sugársűrűséget, a (-1,-1,1) irányból ránézve pedig 0.5 W/st/m^2 sugársűrűséget észlelünk. Mekkora a *kd* diffúz visszaverődési tényező és a *ks* spekuláris visszaverődési tényező az R,G,B hullámhosszakon?

74. Tekintsük a *12x + 4y + 3z = 19* egyenletű sík (1, 1, 1) koordinátájú pontját, amit a (13, 6, 1) pontbeli az r, g, b hullám-hosszokon egyaránt $169^2 * 4 * \pi$ W teljesítményű fényforrás világít meg, és a (4, 13, 5) pontból nézzük. A sík anyagát diffúz + Phong-Blinn modellel írjuk le. Milyen intenzitású fényt érzékel a szem az r, g, b hullámhosszakon, ha a felület diffúz visszaverődési tényezője ugyanezeken a hullámhosszokon [0.1, 0.2, 0.3], a spekuláris visszaverődési tényező minden hullámhosszon 2.89, a fényesség (shininess) pedig 2. Numerikus eredményt várunk, nem elég csak a képleteket felírni, ki is kell az eredményt számítani (nem hiszünk el semmit, csak ami le van írva). *Segítség: A pontszerű fényforrás által keltett sugársűrűség a teljesítmény osztva távolság négyzetével és 4pi-vel. A Phong-Blinn modell, amit az OpenGL is használ, a felezővektor és normálvektor közötti szöggel dolgozik.*

75. Egy diffúz felületet változó erősségű égboltfény világít meg egy adott hullámhosszon, amelyen a felület albedója 1. A megvilágítás erőssége az alábbi függvénnyel jellemezhető: 1/cosθ [W/st/m^2], ha θ > 45 fok, illetve sqrt(2) [W/st/m^2], ha θ <= 45 fok, ahol θ a felület normálisa és az égboltpont iránya közötti szög. Mekkora sugársűrűség éri a szemünket az adott hullámhosszon, ha a nézeti irány éppen 60 fokot zár be a

felületi normálissal, és 3 méter távolságból szemléljük a diffúz felületet? Segítség:

$$L(\vec{x},\vec{\omega}) = L^e(\vec{x},\vec{\omega}) + \int_{\Omega'} L^{in}(\vec{x},\vec{\omega}') \cdot f_r(\vec{\omega}',\vec{x},\vec{\omega}) \cdot \cos\theta' \, d\omega'$$

Sugárkövetés

76. Írja le azon inkrementális algoritmus elvét (ötlet + képletek), amely egy 3D szabályos rács azon celláit kiírja, amelyet egy adott sugár metsz.

77. Írjon sugár és kvadratikus felület (a kradratikus felület implicit egyenlete egy kvadratikus alak) metszési eljárást egy sugárkövető algoritmushoz. Bemenet: sugár kezdőpontja, irányvektora, a kvadratikus alak 4x4-es mátrixa.

78. Kd-fa. A fa szerkezete, jelentése. A fa felépítésének algoritmusa. A fa bejárása, ha egy sugár által metszett cellákat szeretnénk meglátogatni a sugár kezdőpontjától mért távolság sorrendjében.

79. Írjon sugárkövető programot, amely pontszerű fényforrások által megvilágított diffúz gömböket tud megjeleníteni. A szem a 0,0,0 pontban van, a z irányba néz. Az ablak közepe a 0,0,1 pontban van, a z irányra merőleges, mérete 2x2. Az képernyő felbontása 1000x1000 pixel. Feltételezheti, hogy a 3d vektort és RGB színt megvalósító típusok rendelkezésre állnak, rájuk a szokásos operátorok működnek

80. Octree felépítése és bejárása sugárkövetés során.

81. BSP-fa (kd-fa) felépítése és bejárása sugárkövetés során.

82. Reguláris térháló felépítése és bejárása sugárkövetés során.

83. A 2D síkon élők (Laposföld) lakói is sugárkövető programra vágynak. Segítsen nekik! A laposföldieknél a kamera mindig a koordinátarendszer origójában van, és az x irányba néz. A látószög 90 fokos. Az ablak a szemtől egységnyi távolságban van, a felbontása 100. Laposföldön a nap irányfénynek tekinthető, amelyet a (-1, 1) irányba tekintve érzékelünk, csak 350 nm-en sugároz, mégpedig egységnyi intenzitással. Laposföldön *n* darab kör található, amelyeket középpontjával és sugarával adunk meg. A tárgyak diffúzak, az *i.* gömb diffúz visszaverődési tényezője 350 nm-en éppen *i.* A laposföldi monitor csak 350 nm-en sugároz, a *j*. pixel intenzitását a SetPixel(*j, intenzitás*) függvénnyel lehet beállítani. Készítse el a laposföldi sugárkövető programot C++ nyelven!

84. Írjon sugár háromszög metszéspontszámító függvényt C++-ban. A sugarat a kezdőpontjának helyvektorával, és irányvektorával adjuk meg. A háromszöget a három csúcsának helyvektorával.

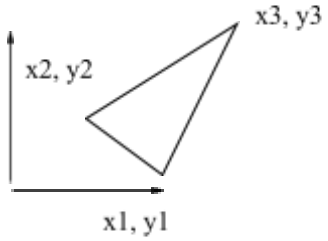
Inkrementális képszintézis

85. A szem a koordináta rendszer origójában van és z irányba néz, a látószög x-ben és y-ban is 90 fokos, az első vágósík 0.1-en, a hátsó 2-n van. Adja meg azt a transzformációs mátrixot, amely a látható tartományt a [-1,-1,0], [1,1,1] sarokpontokkal definiált téglatestbe viszi át. Miért kell vágni a transzformáció végrehajtása előtt a z=0.1 síkra?

86. Írja fel a festőalgoritmus pszeudokódját.

87. Írja fel a Warnock algoritmus pszeudokódját.

88. Írjon 3D háromszög megjelenítőt, amely a képernyőkoordináta rendszerbe transzformált csúcspontokat kapja meg, és a takaráshoz z-buffer algoritmust használ. A háromszöget konstans színnel kell kitölteni. Feltételezheti, hogy a vektor műveletek rendelkezésre állnak és azt is, hogy a háromszög vetületében a koordináták az alábbi relációban állnak.



89. Phong árnyalás. Adja meg, hogy az egyes vektorok milyen koordinátarendszerben értendők.

90. Phong árnyalással jelenít meg egy háromszöget, amelynek csúcspontjai világ-koordinátarendszerben (6, 4, 0),(10, 4, 0),(6, 6, 0). A háromszög csúcsaihoz rendre a következő normálvektorokat rendeltük (1,0,0),(0,1,0),(0,0,1). A háromszög diffúz, a visszaverési együttható (1, 0.5, 0.5). Egyetlen irányfényforrás van, amely a (11, 5, 4) irányból világít (R=3,G=1,B=2) intenzitással. A szem a (5, 6, 7) pontban van. Milyen színű lesz RGB színrendszerben a háromszög (8,5,0) pontja (az eredményt egy értékes jegy pontossággal kérjük)

91. Gouraud árnyalás. Milyen hardvertámogatás építhető hozzá?

92. Háromszög textúrázása inkrementális képszintézisben.

93. Buckaleképzés. Elv, és cél, a buckák megadásának módja, Normálvektor számítása.

94. Írjon egy C függvényt, amely egy 3D háromszöget az x=0 síkra vág. A függvény bemenete tehát három pont, kimenete pedig azon háromszögek tömbje és darabszáma, amelyek a kapott háromszögnek az x=0 sík feletti részét adják ki.

95. Implementálja a Sutherland-Hodgeman poligonvágó algoritmust 3D-ben, Descartes koordinátákban megadott háromszögekre. A megvalósítandó C++ `Vagas` függvény bemenete a háromszögek száma a háromszögeket tartalmazó tömb (`in`), a vágási tartomány két sarka (`Vector min, Vector max`), kimenete (referenciaváltozó) pedig a vágott háromszögeket tartalmazó tömb (`out`).
A tömb típus (`Tomb`) a következő műveletekkel kezelhető:
- `Add( Haromszog h )`: egy új háromszög tömbhöz vétele
- `int Size( )`: a tárolt háromszögek számának lekérdezése
- `Elem(int i)`: az i. háromszög lekérése
A háromszög típus a következő műveletekkel kezelhető:
- `Vector& P(int i)`: Az i. csúcs koordinátái,
- `Vector& T(int i)`: Az i. csúcshoz tartozó uv textúrakoordináták
A `Vector`-on az összeadás, skalárral szorzás műveletek már implementáltak. Ha a megoldás hasonló blokkokból áll, akkor elegendő egyetlen blokkot leírni, és szövegesen utalni arra, hogy a többiben mik

lennének az eltérések. *Segítség: a vágás során egy háromszögből négyszög is keletkezhet, amelyet két háromszögre kell bontani. Feltételezheti, hogy a Haromszog típus elbír 4 csúcspontot is.*

OpenGL/Cg

75. Egy Catmull-Clark subdivision görbe négy kontrolja: (0,0), (1,0), (1,1), (2,1). Írjon C++/OpenGL függvényt, amely a görbét 1 felosztás után felrajzolja! (Segítség: a Catmull-Clark súlyértékei: ¼, ½, ¼). Az ablak megnyitásával és a transzformációk beállításával nem kell foglalkozni.

76. A színtér két háromszögből áll, az egyik nem átlátszó, a másik átlátszósága 50%-os. Hogyan lehet megjeleníteni a színteret OpenGL-lel.

77. Mi a billboard ?

78. Mit csinál a fragmens árnyaló (fragment shader) modulatív textúrázás esetén, mi a bemenete és a kimenete.

79. Mit csinál (milyen programot hajt végre) az OpenGL beépített pixel (fragment) árnyalója (shader) a következő OpenGL hívások esetén:

```
glDisable(GL_LIGHTING);
glEnable(GL_TEXTURE_2D);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glEnable(GL_DEPTH_TEST);
glShadeModel(GL_SMOOTH);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glMatrixMode(GL_PROJECTION);
glLoadIdentity( ); gluPerspective(fov, (double)width/height, front, back);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity( ); gluLookAt(eyex, eyey, eyez, vrpx, vrpy, vrpz, vupx, vupy, vupz);
glBegin(GL_TRIANGLES);
glTexCoord2f(u1, v1);        glVertex4f(x1, y1, z1, w1);
glTexCoord2f(u2, v2); glVertex4f(x2, y2, z2, w2);
glTexCoord2f(u3, v3); glVertex4f(x3, y3, z3, w3);
glEnd();
```

78. Milyen algoritmust valósít meg az alábbi program? (2p). Alakítsa át úgy, hogy Phong árnyalással dolgozzon (vigyázat, nem a Phong BRDF-ről van szó).

Vertex shader:

```
struct inputs {
   float4 position    : POSITION;
   float3 normal      : NORMAL;
   float2 texcoord    : TEXCOORD0;
};

struct outputs {
   float4 hposition   : POSITION;
   float4 color       : COLOR0;
   float2 texcoord    : TEXCOORD0;
};

outputs main( inputs IN,
       uniform float4x4 modelviewIT,        // modelview matrix inverz-transzponáltja
       uniform float4x4 modelviewproj,      // modelview és projekciós mátrix szorzata
       uniform float3 lightdir,             // fény iránya
       uniform float4 I,                    // fény intenzitása
       uniform float4 kd ) {                // diffúz visszaverődési tényező
   outputs OUT;
   OUT.hposition = mul(modelviewproj, IN.position);
   float3 N = normalize( mul(modelviewIT, IN.normal).xyz );
   float costheta = dot(N, normalize(lightdir));
   if (costheta < 0) costheta = 0;
   OUT.color = Idiff * kd * costheta;
   return OUT;
}
```

Fragment shader:

```
float4 main( in float4 color : COLOR0 ) : COLOR { return color; }
```

96. Miért használ float4 adatformátumot az OpenGL egy pont leírására?

97. Írjon OpenGL programot, amely a földgömböt rajzol fel, mégpedig úgy, hogy az egyes pontok vörös komponense a hosszúsági, a kék komponense pedig a szélességi körrel legyen arányos.

98. Rajzolja fel az OpenGL megjelenítési csővezetékét. Milyen operációkat hajt végre az OpenGL az egyes pontokon? Mikor van szükség a mátrix vermekre?

99. A grafikus kártya a modell-view és projektív transzformáció, vágás, homogén osztás és képernyő-transzformáció után egy (x1, y1, z1), (x2, y2, z2), (x3, y3, z3) csúcsú háromszög xy vetületét kitölti, és az egyes pixeleket a z koordinátájukkal a z-buffernek továbbítja a takarási feladat megoldásának érdekében. Mennyivel növeli meg a hardver a z értékét, két, ugyanabban a pásztában lévő (közös y koordinátájú), szomszédos (x és x+1 koordinátájú) pixel között?

100. Írja le, hogy mi történik a csúcspont árnyaló (vertex shader) kimenete és a pixel árnyaló (fragment shader) bemenete között egy GL_TRIANGLES primitív feldolgozása során! A leírásnak olyan részletesnek kell lennie, hogy egy koordinátageometriában jártas programozó implementálni tudja az egyes lépéseket.

101. Adott egy háromszög képernyő koordinátarendszerben a következő csúcsokkal: (0,0,0), (10, 0, 10), (8, 10, 4). Mekkora azon két pont z koordinátáinak különbsége, amelyek az (5,5) illetve a (6,5) pixelen keresztül látszanak?

102. Mit csinál a csúcspont árnyaló a következő beállítások hatására? A választ egy C++ függvénnyel adja meg, amelyben feltételezheti, hogy már létezik egy float3 háromelemű és float4 4 elemű vektor, valamint float4x4 mátrix típus a szokásos műveletekkel. Egy float4 típusú p adatot p.xyz művelettel alakíthat float3-ra.

```
static float kd[4]={kdr, kdg, kdb, kda}, ks[4] = {kdr, kdg, kdb, kda};
glMaterialfv( GL_FRONT, GL_DIFFUSE, kd );
glMaterialfv( GL_FRONT, GL_SPECULAR, ks );
glMaterialf( GL_FRONT, GL_SHININESS, n);

static float I[4]={Ir, Ig, Ib, Ia}, zero[4] = {0, 0, 0, 0};
glLightfv( GL_LIGHT0, GL_DIFFUSE,  I );
glLightfv( GL_LIGHT0, GL_SPECULAR, zero );
glLightfv( GL_LIGHT0, GL_AMBIENT, zero );

static float lightpos[4] = {lpx,lpy,lpz,0}; // Vigyázat a 4. koordináta zérus!!!
glLightfv( GL_LIGHT0, GL_POSITION, lightpos );
glEnable( GL_LIGHT0 );
glEnable( GL_LIGHTING );

glDisable(GL_NORMALIZE);
glEnable(GL_DEPTH_TEST);

float P[4][4], Q[4][4];
ComputeTranforms(P, Q);
glViewport(0, 0, windowWidth, windowHeight);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glMultMatrixf(P);
glMatrixMode(GL_MODELVIEW);
glMultMatrixf(Q);

glNormal3f( 0, 0, 1 );
glBegin( GL_POINTS );
glVertex3f( x1, y1, z1 );
glEnd( );
```

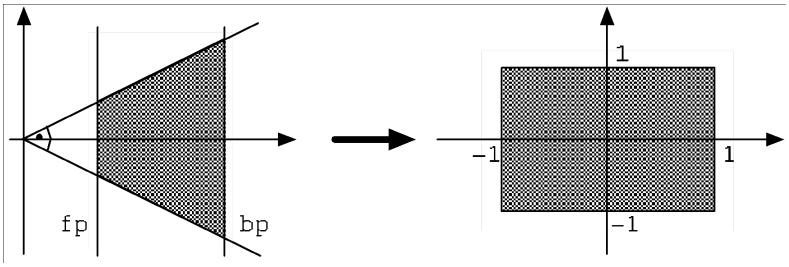
103. Síklapra vetített és árnyék térkép (depth mapped shadows) árnyék számító algoritmusok.

104. Beugrató kérdés: a glEnable(GL_SHADOW) melyik algoritmust kapcsolja be?

105. Készítsen OpenGL autót! Az autó karosszériája téglatest alakú, hosszúsága 4, szélessége 2, magassága 1 m. Az autónak négy kereke van, amelyek henger alakúak, szélességük 0.5 m, átmérőjük 1 m. A kerek tengelye az autó aljához van rögzítve, az autó elejétől illetve a végétől 1-1 méterre. Az autó az x-y síkon az x tengely irányába 1m/s sebességgel halad, t=0-ban az origóból indul. A kerekek a talajon gördülnek. Írjon egy C nyelvű Render(float t) kirajzoló függvényt, amely t időpontban felrajzolja az autót. Feltételezheti, hogy készen áll egy Kocka függvény, amely egységoldalú, origó középpontú, koordinátatengelyekkel párhuzamos élű kockát rajzol ki, valamint egy Henger függvény, amely egység sugarú és magasságú, origóban álló hengert rajzol ki (a körlap az x-y síkon van). Az ajánlott OpenGL függvények (glScalef, glRotatef, glTranslatef, glPushMatrix, glPopMatrix).

106. Készítsen OpenGL tengert! A tenger egy 100x100 m-es téglalap, amelynek közepén egy kis (pontszerű) hajó 1 méter magasságú, 1/sec frekvenciájú hullámokat kelt. A hullámok 2 m/s sebességgel haladnak a tenger széle felé. A hullámzás amplitúdója a forrástól vett távolság négyzetével csökken. A tenger képe felülnézetből a hullámok ellenére állandó. Ezt a képet az id azonosítójú OpenGL textúra tartalmazza. Írjon egy C nyelvű Render(float t) kirajzoló függvényt, amely t időpontban felrajzolja a tengert! Feltételezheti, hogy a kamera már megfelelően be van állítva. A tenger felületét 2 millió háromszögre tesszellálja fel! Az ajánlott OpenGL függvények (glBindTexture, glBegin, glEnd, glutSwapBuffers, glVertex3f, glTexCoord2f).

107. Segítsen a GLU rendszer elkészítésében, és írja meg a gluPerspective függvény egy egyszerűsített változatát. A függvénynek egy xy síkkal párhuzamos alapú csonka gúlát kell leképeznie az origó középpontú 2 egység oldalú kockába (figyelem, a z irányban is 2 egységnyi a céltartomány, lásd a lenti ábra). A csonka gúla levágott csúcsa az origóban van, az alapja az origótól *bp*, a teteje *fp* távolságra található. A csonka gúla nyílásszöge mind x, mind pedig y irányban 90 fok.



108. Milyen transzformációt (melyik OpenGL transzformácót és milyen értékekkel) módosít a **gluPerspective(fovy, aspect, front, back)** GLU függvény. *Segítség: fovy= field of view, nézeti szög az y irányban; aspect =aspect ratio, a magasság és szélesség aránya, width/height; front=első vágósík távolsága a szemtől; back=hátsó vágósík távolsága a szemtől.*

109. Írja le részletesen, hogy mit csinál a grafikus kártya csúcspont árnyaló (vertex shader) és pixel árnyaló (pixel shader) egysége glEnable/glDisable(GL_TEXTURE_2D), glEnable/glDisable(GL_LIGHTING), glEnable/glDisable(GL_NORMALIZE), beállítások mellett, a következő utasítások hatására:

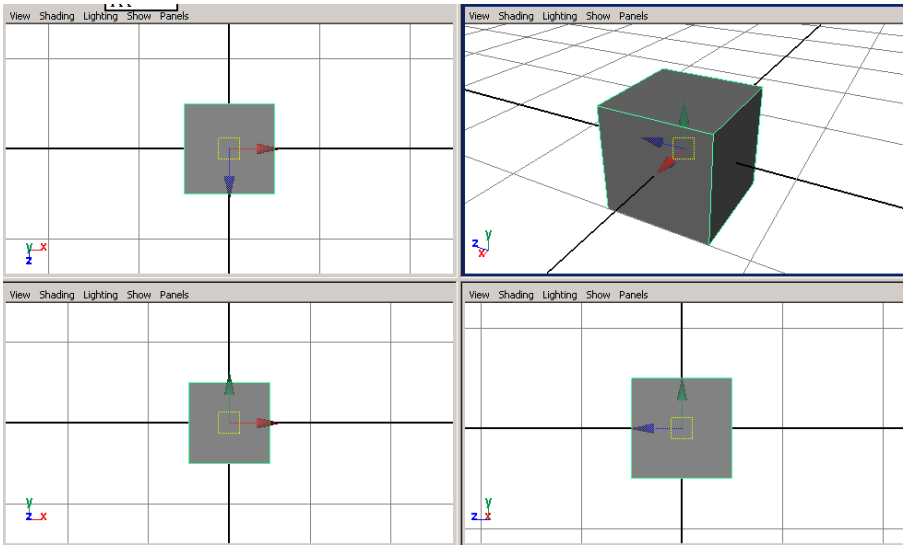
```
glBegin(GL_TRIANGLES);
for(int i = 0; i < 3; i++) {
    glColor3f(r, g, b); glNormal3f(nx, ny,  nz); glTexCoord2f(u, v); glVertex(x, y, z);
}
glEnd( );
```

110. Melyik pixelt színezi át a következő program (4p) és milyen színűre (1p)?

```
glViewport(0,0,200,200);
glDisable(GL_DEPTH_TEST);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity( );
gluLookAt(0,0,0, 3,4,0, 0,0,1);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glColor3f(1,2,3);
glBegin(GL_POINTS); glVertex3f(0.5,0.5,0.5); glEnd( );
```

Segítség: `gluLookAt(eyex,eyey,eyez, lookatx,lookaty,lookatz, upx,upy,upz);`

111. Az alábbi ábrán a Maya kezelői felülete látható, amely a virtuális világot három ortografikus (merőlegesen vetített) és egy perspektív nézetben mutatja meg.



Az ábrán egy egységoldalú kocka látható. A Maya OpenGL-t használ a megjelenítéshez. Milyen MODELVIEW és PROJECTION transzformációt állított be a Maya a jobb alsó ablakban, ha a kocka képe 100x100 pixelt fed le, 400x200 felbontású nézet közepén? A koordinátarendszer közepe az ablak közepére kerül, a y tengely felfelé mutat, a z tengely balra, az x pedig befelé.

Globális illumináció

112. Mi az inverz fényútkövetés (path tracing). Miért választjuk meg benne véletlenül az irányokat. Milyen valószínűségsűrűséget célszerű használni. Mire nem jó ez az eljárás?

113. Mi fontosság szerinti mintavételezés?

Térfogat vizualizáció

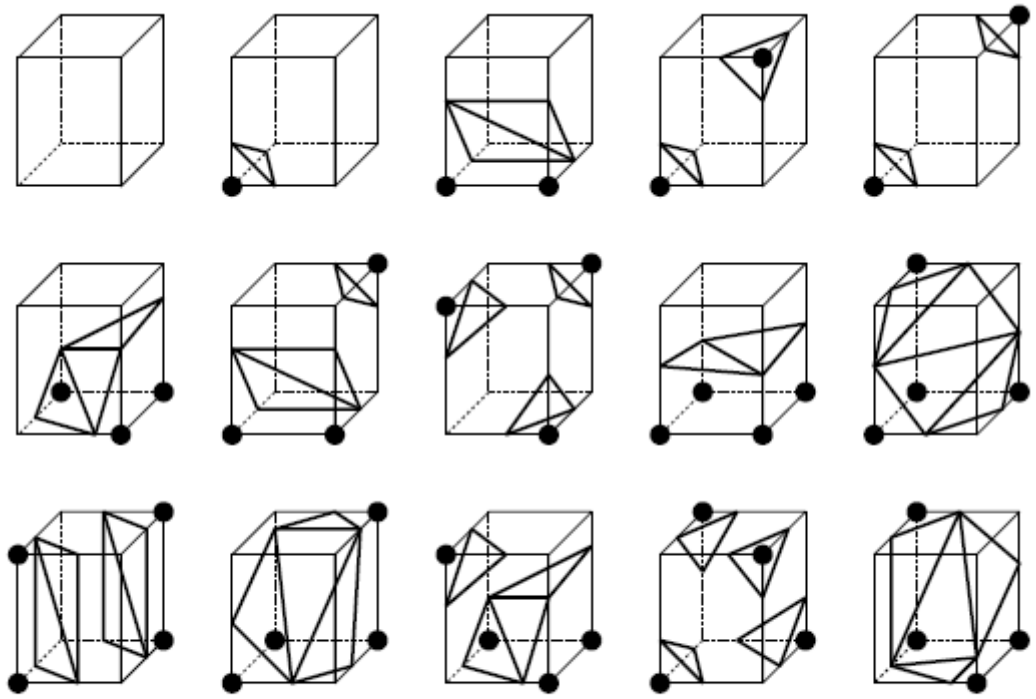
114. Írjon programot, amely egy L(x,y,z) saját színnel és C(x, y, z) opacitásfüggvénnyel jellemzett, a [-1,-1,-1], [1,1,1] kockába zárt térfogatot megjelenít. A szem a [0,0,-2] pontban van, a z irányba néz, az ablak a [0,0,-1] pontban, a z-tengelyre merőlegesen áll és mérete 1x1-s.

115. Masírozó kockák (marching cubes) algoritmus.

Egy térfogati modellből elvileg úgy nyerhetünk felületeket, hogy azonosítjuk a 3D térfogat szintfelületeit, azaz azon 2D ponthalmazokat, ahol a v(x,y,z) megegyezik a megadott szintértékkel. A funkcionális reprezentációnál a felületet definíciószerűen a zérus szintérték képviseli, tehát a zérushoz tartozó szintfelületet kell előállítani. A térfogati modellek általában mintavételezett formában egy 3D tömbben, úgynevezett voxeltömbben állnak rendelkezésre, a funkcionális modellből pedig mintavételezéssel hozhatunk létre voxeltömböt. A továbbiakban a voxeltömbben két rácspontot szomszédosnak nevezünk, ha két koordinátájuk páronként megegyezik, a harmadik koordináták különbsége pedig éppen az adott tengely menti rácsállandó. A rács pontjaiban ismerjük a függvény pontos értékét, a szomszédos rácspontok közötti változást pedig általában lineárisnak tekintjük. A voxeltömb alkalmazása azt jelenti, hogy az eredeti függvény helyett a továbbiakban annak egy voxelenként tri-lineáris közelítésével dolgozunk (a tri-lineáris jelző arra utal, hogy a közelítő függvényben bármely két koordinátaváltozó rögzítésével a harmadik koordinátában a függvény lineáris). A lineáris közelítés miatt két szomszédos rácspont közötti él legfeljebb egyszer metszheti a közelítő felületet, hiszen a lineáris függvénynek legfeljebb egyetlen gyöke lehet.

A felületet háromszöghálóval közelítő módszer neve masírozó kockák algoritmus (marching cubes algorithm). Az algoritmus a mintavételezett érték alapján minden mintavételezési pontra eldönti, hogy az a szintértéknél kisebb vagy nagyobb-e. Ha két szomszédos mintavételezési pont eltérő típusú, akkor közöttük felületnek kell lennie. A határ helyét és az itt érvényes normálvektort a szomszédos mintavételezési pontok közötti élen az értékek alapján végzett lineáris interpolációval határozhatjuk meg.

Végül az éleken kijelölt pontokra háromszögeket illesztünk, amelyekből összeáll a közelítő felület. A háromszögillesztéshez figyelembe kell venni, hogy a tri-lineáris felület a szomszédos mintavételezési pontokra illeszkedő kocka éleinek mindegyikét legfeljebb egyszer metszheti. A kocka 8 csúcsának típusa alapján 256 eset lehetséges, amiből végül 15 topológiailag különböző — azaz egymásból elforgatással nem létrehozható — konfiguráció különíthető el (3.57. ábra).



3.57. ábra. *Egy voxelenkénti tri-lineáris implicit függvényű felület és egy voxel lehetséges metszetei. Az ábrán az azonos típusú mintavételezett értékeket körrel jelöltük.*

Az algoritmus sorra veszi az egyes voxeleket és megvizsgálja a csúcspontok típusát. Rendeljünk a szintérték alatti csúcspontokhoz 0 kódbitet, a szintérték felettiekhez pedig 1-et. A 8 kódbit kombinációja egy 0–255 tartományba eső kódszónak tekinthető, amely éppen az aktuális metszési esetet azonosítja. A 0 kódszavú esetben az összes sarokpont a testen kívül van, így a felület a voxelt nem metszheti. Hasonlóan, a 255 kódszavú esetben minden sarokpont a test belsejében található, ezért a felület ekkor sem mehet át a voxelen. A többi kódhoz pedig egy táblázatot építhetünk fel, amely leírja, hogy az adott konfiguráció esetén mely kockaélek végpontjai eltérő előjelűek, ezért metszéspont lesz rajtuk, valamint azt is, hogy a metszéspontokra miként kell háromszöghálót illeszteni.

116. Írjon térfogati modellek megjelenítéséhez *fénysugárkövető (segítség: térfogati sugárkövetés, amikor a fénnyel megegyező irányban haladunk)* programot. Feltételezheti, hogy a térfogati adatok egy 256x256x256-os voxel tömbben vannak, amely a világ koordinátarendszerben a [0,0,0,], [1,1,1] pontok közötti, a tengelyekkel párhuzamos élű kockának felel meg.. A szem a [0.5, 0.5, -∞] pontban van (*segítség: párhuzamos vetítés*!), és a z irányba néz. Az ablak közepe a [0.5, 0.5, 0] pontban van, a z irányra merőleges, mérete 1x1. Az képernyő felbontása 256x256 pixel. A voxel színek és opacitások számításánál csak a 0.5 és 0.6 közötti értékeket tekintse nem átlátszónak és a diffúz visszaverődési törvényt adaptálja a szín számításához *(segítség: a normálvektort a gradiensből számítsa, az opacitást pedig a gradiens abszolút értékével szorozza*). A diffúz BRDF az RGB hullámhosszokon (0.1, 0.4, 0.9). A pontszerű fényforrás a 0,0,0 pontban van, az intenzitása tetszőleges távolságban, helyen és hullámhosszon 1. Feltételezheti, hogy a szokásos vektor és színműveletek rendelkezésre állnak.

Animáció

117. Írjon Animate függvényt C++-ban, amely egy galaxist égitestjeinek a helyzetét (newtoni) fizikai animációval számítja ki a t időpillanatra. Összesen 100 égitest van, amelyek tömegei az m[100] tömbben, sugaraik az r[100] tömbben vannak. Az égitestek kezdeti helyzetei (x[100], y[100], z[100]) tömbökben találhatók. A t=0 időpillanatban az égitestek állnak. Az égitestek ütközése teljesen rugalmatlan (figyelem, az impulzus megmarad!), azaz ütközéskor összetapadnak, innentől úgy tekintjük őket, mint egy gömböt, amelynek sugarai a találkozó gömbök sugarainak összege. A Newton féle gravitációs állandó f.

118. Inverz kinematika. Cél. Új állapot meghatározása. Pszeudo-inverz. Algoritmus.

119. Adja meg a forward és inverz kinematikát alkalmazó mozgástervezés lépéseit és hasonlítsa össze lehetőségeit!

120. Adott egy kör alakú biliárdasztal. Az asztal középpontjában vesszük fel a koordinátarendszerünket. Az asztal sugara R. Írjon Animate függvényt C++-ban, amely kiszámítja egy biliárdgolyó helyzetét a t időpillanatra, feltételezve, hogy t=0-ban a golyót az (x0, y0) pontból és (vx, vy) kezdősebességgel indítottuk el, csak ez az egyetlen golyó az asztalon, a golyó az asztal oldalával tökéletesen rugalmasan ütközik, és a súrlódást elhanyagoljuk.

121. Vesse össze a Lagrange interpolációs görbét, a B-spline-t és a NURBS-t az animációs mozgásgörbék szempontjából. Táblázatosan mutassa be, hogy milyen előnyös és hátrányos tulajdonságaik vannak. A tulajdonságok között feltétlenül szerepeljenek az alábbiak: konvex burok tulajdonság, lokális kontroll, paraméter-idő hozzárendelés, interpolációs tulajdonság.

122. Egy labda pattogását a magasságfüggvény kulcspozícióival (keyframe) adjuk meg:
    t=0, y=0; t=1, y=1; t=3, y=0; t=6, y=1;
    Mi a labda y koordinátája t=2-kor, ha Catmull-Rom spline-t használunk interpolációra (a Catmull-Rom spline a Kochanek Bartels spline speciális esete, amikor a tau zérus, beta pedig 0.5).

123. Adott egy, a koordinátarendszer origójában rögzített csukló, amely a z tengely körüli elfordulását és a hosszát képes változtatni. Írja fel a rendszer Jacobi mátrixát. Írjon programot, amely csukló másik végpontját a [1,1], [0, 2] pontok között egyenletes sebességgel végigvezeti. Feltételezheti, hogy az adott programozási nyelven a mátrixműveletek rendelkezésre állnak.

124. Adott egy 2D világban mozgó robotkar, amelynek egyik vége az origóhoz rögzített. A robotkar a rögzített vége körül el tud fordulni, és a hosszát és meg tudja változtatni. Írja fel a robotkar Jacobi mátrixát, állítsa elő a mátrix pszeudó inverzét. Írjon C++ függvényt, amely dt időszeletenként lépdelve kiszámítja a robotkar elfordulási szögét és hosszát, ha annak nem rögzített végét az x(t), y(t) pályán kell végigvezetni.

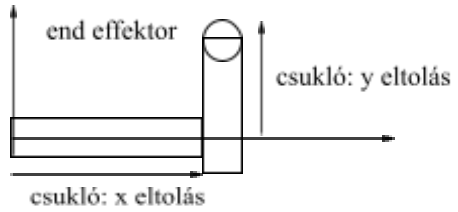
125. Egydimenziós mozgást kulcskeret animációval definiál. A tárgy a t=0 időpillanatban az x=0 pontban, a t=1 időpillanatban az x=1 pontban, a t=2 időpillanatban megint az x=0 pontban van. A mozgás nyugalmi helyzetből indul és így is fejeződik be (sebesség zérus). Hol van a tárgy a t=0.4 időpontban, ha Catmull-Rom spline-nal interpolálunk.

126. Saját magát szeretné egy animációs filmben szerepeltetni, amint éppen egy lépcsőfokra föllép. Írja le pontokként, hogy ehhez mit kell tenni, ha kulcskeret (keyframe) animációs eljárást használ.

127. Saját magát szeretné egy animációs filmben szerepeltetni, amint éppen egy lépcsőfokra föllép. Írja le pontokként, hogy ehhez mit kell tenni, ha mozgáskövető (motion tracking) animációs eljárást használ.

128. Egy létező tárgy 3D modelljét szeretné létrehozni. Hogyan valósítható meg a sztereolátás alkalmazásával. Mi változik a 2-nél több kamerát használ?

129. Adott egy két transzlációs csuklót tartalmazó robot. A transzlációs csukló orientációja állandó, csak a hosszát képes változtatni. Az ábra szerint, az első csukló x irányú eltolást, a második y irányú eltolást jelenthet. Írja fel a robot Jacobi matrixát! Adja meg a mátrix pszeudoinverzét! Hogyan kell vezérelni a csuklókat, hogy az endeffektor egy körön menjen végig 2PI szögsebességgel.



130. Egy egység tömegű test tömegközéppontjának pályáját egy síkbeli Bézier görbével adhatjuk meg, amelynek vezérlőpontjai: (0,0), (1,1), (2,0). Mekkora eredő erő hat a testre a t=0.33 időpillanatban?

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2 \ , \ t \in [0,1].$$

ennek deriváljta t szerint megadja a sebességet, annak deriváltja meg a gyorsulást, mivel m=1
igy F=ma=B”(t) * 1

131. Egy labdát szeretne animálni **kulcskeret animációval**, **Catmull-Rom spline** alkalmazásával. A kulcskeretek a következők (t-vel az időt jelöljük és másodpercben értelmezzük):
t= 0: a labdát a [0, 1, 0] pontból [1, 1, 0] sebességgel elhajítjuk.
t= 1: a labda az [1, 2, 0] pontban repül
t= 2: a labda a [2, 0, 0] pontba csapódik [1,-1,0] sebességgel és tökéletesen rugalmatlanul ütközik a talajjal.

Tételezzük fel, hogy az előző feladat megoldásául kapott paraméteres egyenletet egy **`Vector r(float t)`** függvényben már megvalósítottuk. Egészítse ki az alábbi programot a Labda( ) függvény implementációjával, amely a labda mozgását OpenGL segítségével animálja. A labda fehér, 40x40 háromszögre tesszellált gömb, amelynek sugara 2.

```
struct Vector{float x, y, z; };
GLUquadricObj * labda;
Vector r( float t );

void Labda( ) { IDE MI JÖN?????? }

main( argc, argv ) {
     glutInitWindowSize(width, height);  glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_RGBA| GLUT_DOUBLE | GLUT_DEPTH);
     glutCreateWindow(”Labda…");
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(60, 1, 1, 1000);
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     labda = gluNewQuadric( );
     glEnable(GL_DEPTH_TEST);
     glutDisplayFunc( Labda );
     glutIdleFunc( Labda );
     glutMainLoop();
}
```

Segítség: a következő OpenGL függvényekből érdemes válogatni (nem szükségképpen mindet!):
`glPushMatrix( ), glPopMatrix( ), glColor3f(), glVertex3f(), glTranslatef( ), glRotatef( ), glScalef( ), gluSphere( ); glClearColor( ); glutSwapBuffers(), glTexCoord2f(), glBindTexture(),`

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glutGet(GLUT_ELAPSED_TIME);
```

132. Egy háromszög csúcsainak koordinátái a lokális modellezési koordinátarendszerben [0,0,0], [1,0,0], [0,1,0]. Animálja a háromszöget keyframe (kulcskeret) animáció segítségével úgy, hogy a háromszög orientációja állandó maradjon, a pozíciót pedig a kulcskeretekből lineáris interpolációval számítsa ki! Tételezze fel, hogy a kulcskeretek időértékei a **float t[]** globális tömbben, az első csúcs pozíciója pedig ezekben a kulcskeretekben a **Vector v[]** tömbben vannak. A kulcskeretek száma **int nf**. A rajzolást OpenGL hívásokkal végezze el. Feltételezheti, hogy IdleCallback-ként az Idle függvényt regisztráltuk, tehát csak az Idle-t kell megírni, az inicializálási és kamerabeállító részt nem. A rajzolást **GL_TRIANGLE** beállítással végezze, a háromszög színe fehér. Az alkalmazás indítása óta eltelt időt **glutGet(GLUT_ELAPSED_TIME)** hívással kérdezheti le. Segítségnek az ajánlott OpenGL parancsok: **glBegin, glEnd, glFlush, glVertex3f, glColor3f, glTranslatef, glPushMatrix, glPopMatrix, glLoadIdentity, glutSwapBuffers, glClear .** Mit és hogyan kell változtatni a programban ahhoz, hogy ne lineáris, hanem Catmull-Rom spline interpolációval dolgozzon (a Catmull-Rom spline a Kochanek Bartels spline speciális esete, amikor a tau zérus, beta pedig 0.5)?

133. Az alábbi lövedék röpül a hegyes végét elől tartva (úgy, ahogy egy tisztességes lövedéktől elvárható) a 3D virtuális térben (lásd a lenti rajzot). A lövedék hengeres részének magassága 4 m, a kúpos részének magassága 2 m, átmérője 1.5 m. A lövedék végének középpontja az [x0, y0, z0] pontról indul [vx0, vy0, vz0] kezdősebességgel. A lövedék az ágyúcső hornyolása miatt *f* [fok/sec] szögsebességgel forog a főtengelye körül. A nehézségi gyorsulás g [m/sec^2], a közegellenállás elhanyagolható, ütközés nincs. Írjon C függvényt, amely képletanimációval meghatározza a *t* pillanatban érvényes mozgásállapotot, és OpenGL függvényhívások segítségével fel is rajzolja a lövedéket. A rajzoláshoz felhasználhatja a Henger() függvényt, amely egy 1 m magasságú, origo középpontú, z tengelyű, 1 m átmérőjű hengert jelenít meg, és a Kúp() függvényt, amely egy xy síkon álló, 1 m magas, 1 m átmérős alapkörrel rendelkező kúpot rajzol fel.

134. Egy pontszerű test kétdimenziós mozgását Catmull-Rom spline-nal adja meg. A kulcspontok:

t = 0: x = 0, y = 0, vx = VX, vy = VY
t = 1: x = VX, y = H,
t = 2: x = 2 * VX, y = 0, vx = VX, vy = -VY

A VX, VY, H értékek konstansok. Adja meg az *x(t), x(t)* mozgásgörbék algebrai alakján a [0, 1] időintervallumban!

Pontozás: Catmull-Rom spline def : 1p; Feltételrendszer megfogalmazása: 3p; Paraméterek számítása: 1p.
(segítség: A Kochanek-Bartels spline speciális esete, amikor a tau = ½ és a bias=0)

135. Marcus Aurelius a barbárok ellen hadakozik. A barbárok kőhajítókkal támadnak, Marcus bátran és mozdulatlanul áll. A sziklák lényegesen kövérebbek Marcusnál, tehát nem lehetne Marcus belsejében elrejteni azokat, viszont nem feltétlenül magasabbak, mint Marcus. Készítsen C++/OpenGL függvényt, amely megjeleníti Marcust és a sziklát a képernyőn, és diszkrét idejű ütközésdetektálási eljárással eldönti, hogy a szikla eltalálja-e Marcust. A függvény bemeneti paraméterei:

- Marcust definiáló háromszögek tömbje (`Tomb hMarcus`), Marcus jelenlegi állapotában,
- a sziklát definiáló háromszögek tömbje (`Tomb hSzikla`), a szikla referencia állapotában,
- a szikla jelenlegi helyzetét és orientációját definiáló homogén lineáris transzformáció (`float tSzikla[4][4]`).

Akkor mondjuk, hogy Marcust a szikla eltalálja, ha a szikla AABB-je (axis-aligned bounding box) Marcus valódi geometriájával ütközik (azaz az AABB Marcus bármely belső pontját tartalmazza). Feltételezheti, hogy Marcust definiáló háromszögháló zárt felület. Ütközés esetén a függvény TRUE értékkel tér vissza, egyébként FALSE-zal. Az ütközés felismeréshez használja fel az előző feladat `Vagas` függvényét. Marcus textúra azonosítója `tMarcus`, a szikláé `tSzikla`. A textúra kép betöltésével és a kamera, illetve az ablak

beállításával nem kell foglalkozni. A következő OpenGL függvényeket alkalmazza: `glBegin`, `glEnd`, `glVertex3d`, `glTexCoord2d`, `glBindTexture`. (6 pont)
Segítség: A sziklák és Marcus kövérségére vonatkozó feltétel arra utal, hogy nem kell foglalkozni azzal az esettel, amikor Marcus teljes egészében tartalmazza a sziklát. A Marcus magasságára vonatkozó megjegyzés szerint viszont előfordulhat, hogy egy szikla Marcus háromszöglistájának egyetlen csúcspontját sem tartalmazza, mégis ütközik vele, mert Marcus háromszögei metszik a szikla AABB-jét.

korábbi vizsgák