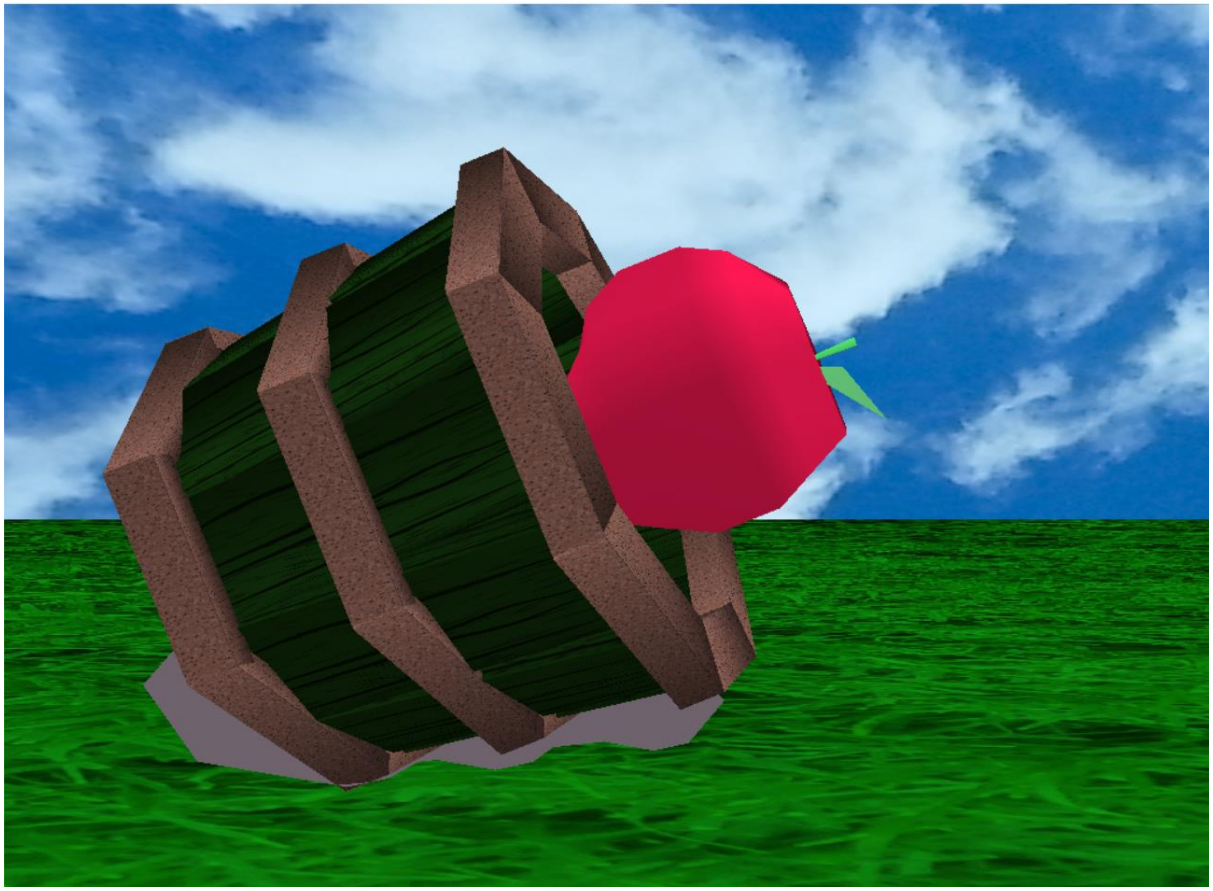
Coș cu măr



Nume: Țoghe Răzvan Constantin

Specializarea: Calculatoare

Semigrupa: 223/2

6 iunie 2020

Cuprins

Inițializare 3
Modul de vizualizare și Display 5
Main 6
Șcena principală 7
Modulul Lemn 7
Modulul Cos 7
Modulul Inel 8
Modulul Capac_jos 8
Modulul Capac_inel 8
Modulul Mar 9
Modulul Mediu 9

Inițializare

În inițializarea programului începem cu crearea fundalului alb prin curățarea culorilor existente în buffer, golirea buffer-ului de culoare și de adâncime, activarea texturilor 2D, a iluminării și a sursei de lumina *LIGHT0*, activarea testului de adâncime.

```
void myinit(void) {
	glClearColor(1, 1, 1, 1.0);

	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glEnable(GL_TEXTURE_2D);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_DEPTH_TEST);

	pune_texturi();
	iluminare();
}
```

Tot aici apelez o funcție pentru a încărca texturi salvate în rădăcina proiectului în format BMP și a le salva în variabile “ID_XX” pentru a fi utilizate ulterior la desenare.

```
void CALLBACK pune_texturi(void) {
	sir = ".\\granit.bmp";
	incarca_textura(sir);
	ID_G1 = IDtextura;
	…
}
```

Încărcarea texturii se face printr-o funcție `incarca_textura` care primește ca parametru locația imaginii sub forma unui pointer de char, imaginea este apoi preluată într-un obiect de tip `AUX_RGBImageRec *pImagineTextura`, obiect asupra căruia se fac operații dând proprietăți Textelilor sau eliberând zona de memorie alocată

```
void CALLBACK incarca_textura(const char* s){
	AUX_RGBImageRec *pImagineTextura = auxDIBImageLoad(s);
	if (pImagineTextura != NULL)	{
		glGenTextures(1, &IDtextura);
		glBindTexture(GL_TEXTURE_2D, IDtextura);
		glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
		glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
		glTexImage2D(GL_TEXTURE_2D, 0, 3, pImagineTextura->sizeX,
pImagineTextura->sizeY,
			0, GL_RGB, GL_UNSIGNED_BYTE, pImagineTextura->data);
	}
	if (pImagineTextura) {
		if (pImagineTextura->data) {
			free(pImagineTextura->data);
		}
		free(pImagineTextura);
	}
}
```

După ce texturile au fost salvate, se continuă cu setarea parametriilor de iluminare folosind funcția `iluminare();` Funcție care are rol de a seta coeficienți de reflexie ambientală și difuză, reflexia speculară și exponentul de reflexie speculară (strălucirea), valori ale intensității sursei de lumină alese *LIGHT0* pentru lumină ambientală, difuză și speculară.

```
void CALLBACK iluminare(void) {
	// coeficientii de reflexie pentru reflexia ambientala si cea difuza sunt cei
impliciti
	GLfloat mat_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
	GLfloat mat_diffuse[] = { 1, 1, 1, 1.0 };
	/*  reflectanta speculara si exponentul de reflexie speculara nu sunt la
valorile implicite (0.0)   */
	GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat mat_shininess[] = { 80.0 };
	// valori implicite pentru intensitatea sursei LIGHT0
	GLfloat light_ambient[] = { 0.2, 0.2, 0.2, 1.0 };
	GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };

	GLfloat lmodel_ambient[] = { 0.2, 0.2, 0.2, 1.0 };

	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);

	//permite urmarirea culorilor
	glEnable(GL_COLOR_MATERIAL);
}
```

Modul de vizualizare și Display

Tipul de vizualizare este din perspectivă, cu unghiul de vizualizare de 50grd în direcția y, distanța de la privitor la planul de tăiere apropiat de 1.0, iar la planul îndepărtat de 350.0. La final, în matricea de modelare și vizualizare, mutăm toate obiectele în spate cu 5 unități pentru a fi mai vizibile.

```
void CALLBACK myReshape(GLsizei w, GLsizei h) {
	if (!h) return;
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(50.0, (GLfloat)w / (GLfloat)h, 1.0, 350.0);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef(0.0, 0.0, -5.0);  /*  aduce obiectele in volumul de vizualizare
*/
}
```

În funcția de display, după ce am trecut în matricea de modelare și vizualizare la finalul modului de vizualizare, golim buffer-ul de culoare și adâncime după care apelez scena pe care vreau să o afișez, după care schimb între buffe cu `auxSwapBuffers()`.

```
void CALLBACK display(void) {
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	Scena_umbre();
	auxSwapBuffers();
}
```

Main

În main inițializez modul de afișare în care specific ferestrei să fie cu buffer dublu, modul culorilor RGB și buffer de adâncime de 32-bit. Apoi condițiile inițiale de afișare, în colțul din stânga sus cu dimensiunile de 900x1000px, titlul ferestrei, inițializarea, tastele folosite pentru animații, modul de vizualizare și funcția principală de repetare și anume display-ul.

Tastele folosite sunt:

- E , pentru a muta sursa de lumină în partea dreaptă,
- Q, pentru a muta sursa de lumină în partea stângă,
- Săgeată SUS, pentru a roti coșul și mărul în sus,
- Săgeată JOS, pentru a roti coșul și mărul în jos,
- Săgeată STÂNGA, pentru a roti coșul și mărul în stânga,
- Săgeată DREAPTA pentru a roti coșul și mărul în dreapta,
- 1, pentru a anima scena pe un timeline positiv,
- 2, pentru a anima scena pe un timeline negativ.

```
int main(int argc, char** argv) {
	auxInitDisplayMode(AUX_DOUBLE | AUX_RGB | AUX_DEPTH16);
	auxInitPosition(0, 0, 900, 1000);
	auxInitWindow("Cos cu mere");
	myinit();

	auxKeyFunc(AUX_e, movelight_x_plus);
	auxKeyFunc(AUX_q, movelight_x_minus);
	auxKeyFunc(AUX_DOWN, rot_x_p);
	auxKeyFunc(AUX_UP, rot_x_m);
	auxKeyFunc(AUX_RIGHT, rot_y_p);
	auxKeyFunc(AUX_LEFT, rot_y_m);

	auxKeyFunc(AUX_1, toggle_anim);
	auxKeyFunc(AUX_2, toggle_anim_m);

	auxReshapeFunc(myReshape);
	auxMainLoop(display);
	return(0);
}
```

Scena principală

Scena principală conține o scenă secundară care se apelează inițial fără umbră, iar apoi se apelează și se înmulțește cu matricea de umbrire pentru a crea umbra pe suprafața paralelă cu planul XOZ (orizontal). Ambele scene conțin mai multe module care ușurează extinderea ulterioară a proiectului.

Toată scena principală este gândită pentru a funcționa cu activarea eliminarea fețelor și testului de adâncime, exceptân mărul.

Modulul Lemn

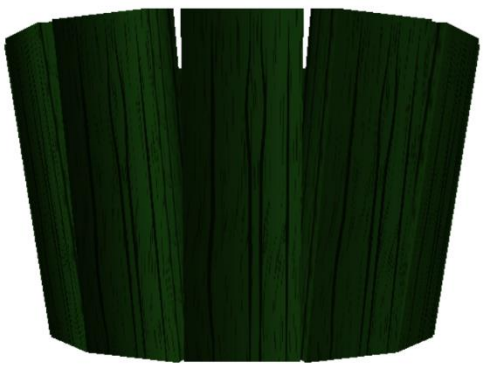
Să încep cu cel mai important modul și anume modulul pentru a crea un lemn, folosit pentru crearea celor 12 scânduri, dar și a celor 3 inele care le înconjoară.

```
void CALLBACK Lemn(float x, float y, float z, float h, float L, float l,
string s) {
	L = L / 2;
	l = l / 2;
	h = h / 2;
	glEnable(GL_CULL_FACE);				//activeaza
eliminarea fetelor
	glCullFace(GL_BACK);				//sunt eliminate
fetele spate
	glEnable(GL_DEPTH_TEST);			//activare test
adancime
	...
}
```

Acest modul primește ca parametrii coordonatele în care se v-a desena, înalțimea, lățimea si lungimea lemnului, dar și scopul pentru care se creează, de a fi o scândură sau parte din inel printr-un string “wood” sau “granit”. Modulul a fost dezvoltat de la început ca centrul coordonatelor x, y, z primite ca parametru să fie în centrul de greutate al scândurei, de aceea asupra parametrilor h, L și l se efectuează o înjumătățire a valorilor pentru a putea porni de la coordonatele x, y, z cu o jumătate din lungime într-o direcție, si cu cealaltă jumătate în direcția opusă. Modul mai dificil de dezvoltat, dar care ușurează mult mai mult continuarea dezvoltării a 12 scânduri și 16 componente ale celor 3 inele (câte 8 componente pentru fiecare inel), prin calcularea doar a coordonatelor locurilor în care vreau să fie afișate și eventuale rotații asupra lor.

Modulul Cos

Acest modul se folosește de modulul Lemn și de 3 vectori care conțin pozițiile pe axa X și Z, dar și cu cât să se rotească față de axa Y. Pentru un finisaj al aspectului, scândurile se mai rotesc cu 9grd pe axa X pentru a fi vizibile câteva spații în partea superioară, lăsând puțin vizibil conținutul coșului.

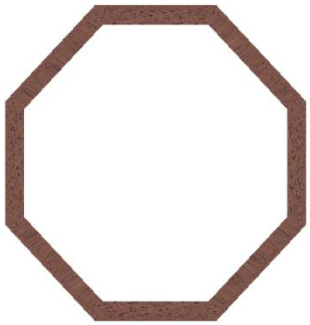


Modulul lemn este modelat într-un push-pop matrix astfel încât să obțin acest aspect.

```
for (int i = 0; i < 12; i++) {
	glPushMatrix();
		glTranslatef(poz_x[i], 0, poz_z[i]);
		glRotatef(alpha[i], 0, 1, 0);
		glRotatef(beta, 1, 0, 0);
		glTranslatef(-poz_x[i], 0, -poz_z[i]);
		Lemn(poz_x[i], 0, poz_z[i], h, L, l, "wood");
	glPopMatrix();
}
```

Respectând ordinea efectuării operațiilor, de la sfârșit la început, se creează o scândură în coordonatele x, y, z, se translatează în centrul axelor de coordonate 0, 0, 0 și ținând cont de faptul că parametrii x, y, z ai scândurei sunt fix în centrul ei, adică în centrul de greutate, după translatare, scândura se află acum fix în centru, așadar pot acum să aplic orice rotație vreau fără să stric restul scenei, după care translatez scândura în poziția inițială.

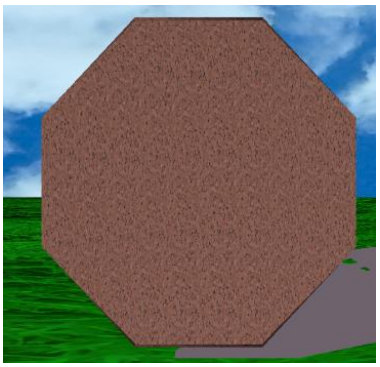
Prima scândură este cea din spre fereastra de vizualizare



Modulul Inel

Acest modul se folosește tot de modulul Lemn, dar alterând parametrii h, L și l se poate obține un cub alungit, pentru fiecare inel se crează 8 astfel de module, apoi principiul de creare este același ca la coș, prin vectori de poziție, rotație și variabile pentru axa Y de data aceasta, deoarece sunt create 3 inele pe 3 poziții ale axei Y.

Modulul Capac_jos



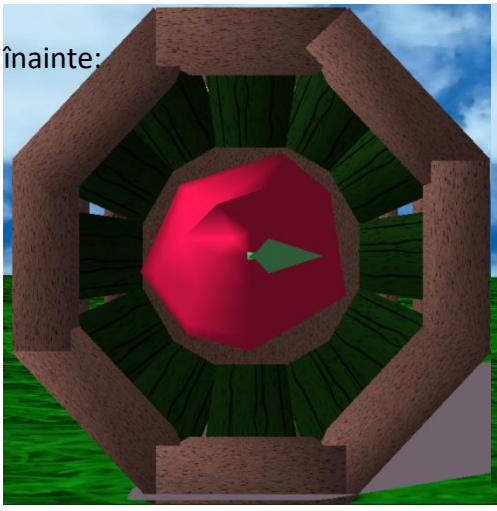
Acest modul primește ca parametrii coordonatele x, y și z, reprezentând centrul capacului inferior, cu un decalaj de - 0.751 pe axa Y hardcodat în vertex-uri

De ce cu 0.001 mai mult? Pentru că dacă se păstra aceeași distanță de 0.750, în locurile în care capacul inferior s-ar fi întâlnit cu partea inferioară a inelului mic, s-ar fi suprapus figurile.

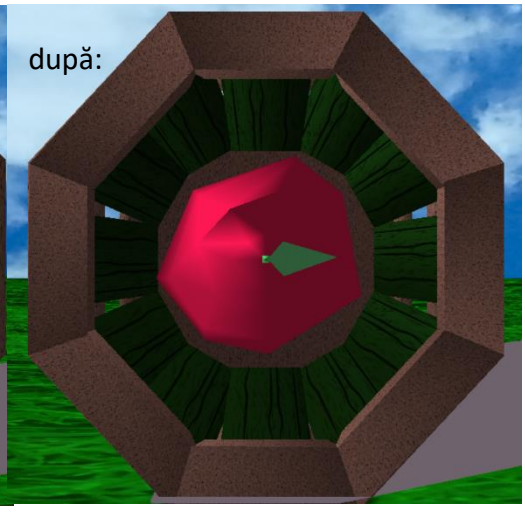
Modulul Capac_inel

Acest modul a fost conceput pentru a remedia un glitch care se poate observa în figurile de mai jos:

înainte:

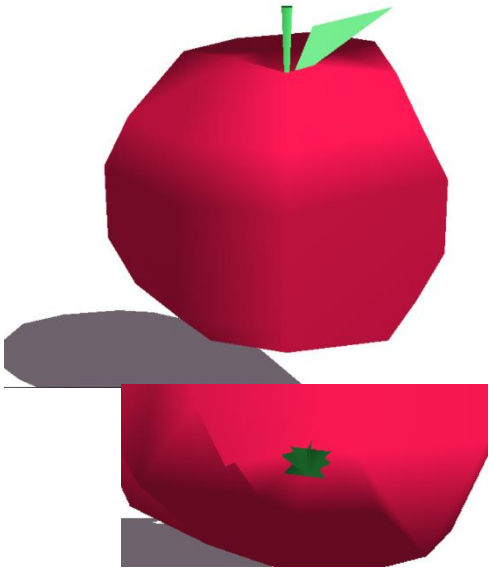


după:



Aici poate fi observat de ce este util acel decalaj de 0.001 și suprapunerea modulelor, și acest modul are rol de a finisa obiectul final având un aspect mai plăcut pentru privitor. Iluminarea însă se comportă diferit, sunt făcute să ilumineze ori partea dreaptă a fiecărui modul, ori partea stangă în funcție de poziția sursei de lumină.

Modulul Mar



Acest modul primește ca parametrii x, y, z, reprezentând centrul mărului, decalat cu 1 în jos pe axa Y.

Este conceput cu primitive de tip `GL_QUAD_STRIP` și `GL_TRIANGLE_STRIP` .

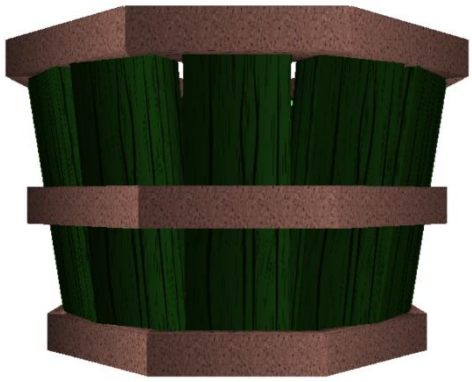
Pentru detalii s-a conturat o formă inegală, în realitate nu toate merele sunt perfect sferice, o tulpină, frunză și caliciu (partea inferioară a mărului de culoare maronie)

Modulul Mediu

Acest modul conține soarele, ale cărui coordonate se află într-un vector global de poziție și reprezentat fizic printr-o mică sferă solidă de culoare galbenă. Poziția soarelui a fost dati inițial în partea dreaptă în spre fereastra de vizualizare

Tot aici se regăsesc și cerul sub forma unei cuadrice, pe care s-a pus o textură cu nori, dar și pământul, creat din vertex-uri, pe care s-a pus o textură cu iarbă. Acest modul este singurul care nu are umbre. Reprezentând mediul în care se află coșul umbra trebuie să se redea pe acest mediu.

Revenind la șcena principală, aici se creează matricea de umbrire, care primește că parametrii planul pe care se redă, poziția sursei de lumină și matricea de umbrire, se activează sursa de lumină *LIGHT0*, se setează variabila booleană *umbra* cu 0, care semnifică faptul că acum nu se generează umbra, se afișează în consolă parametrii de debug pentru animații și în final se apelează șcena secundară care conține modulele și animațiile



```
void CALLBACK Scena_umbre(void) {
	MatriceUmbra(puncte, position, matUmbra);
	glPushMatrix();
		glLightfv(GL_LIGHT0, GL_POSITION,
position);
		umbra = 0;
		Scena();
	glPopMatrix();
}
```

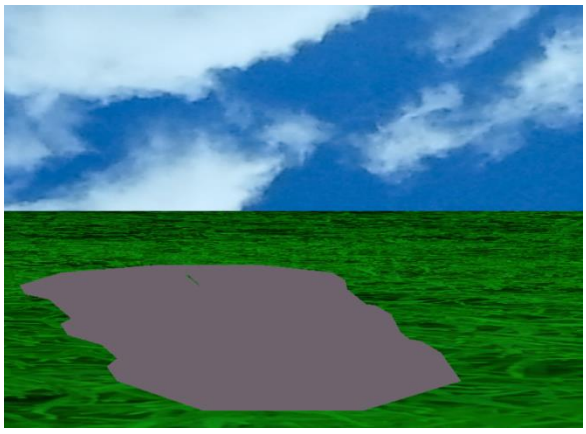
```
void CALLBACK Scena(void) {
	glPushMatrix();
		animatie();
		glRotatef(x, 1, 0, 0);
		glRotatef(y, 0, 1, 0);
		Cos();
		Inel();
		Capac_jos(0, 0, 0);
		Capac_inel(0, 1.70, 0);
		glPushMatrix();
			if (anim <= 125) {
				glTranslated(anim_deplasare, anim_mar, 0);
			}
			else {
				glTranslated(anim_deplasare_min, anim_mar, 0);
			}
			glTranslatef(0, 0.2, 0);
			glRotated(anim_rot_mar, 0, 0, -1);
			glTranslatef(0, -0.2, 0);
			Mar(0, 0.2, 0);
		glPopMatrix();
	glPopMatrix();
}
```

Tot în scena principală, după ce s-a desenat scena secundară fără umbre, trebuie să desenăm umbrele și mediul înconjurător

Într-un push-pop matrix, se face `glPushAttrib(GL_LIGHTING_BIT);` pentru a salva pe stivă culorile și intensitățiile pentru tipurile de iluminare specificate în inițializare, se dezactivează iluminarea, se apelează scena cu parametrul *umbra* pe 1 semnificând că acum vreau să desenez umbra, și se înmulțește matricea de umbrire cu matricea care conține scena, după care se restaurează datele salvate pe stivă cu `glPopAttrib();`

```
void CALLBACK Scena_umbre(void) {
	...
	glPushMatrix();
		glPushAttrib(GL_LIGHTING_BIT);
		glDisable(GL_LIGHTING);
			glPushMatrix();

		glMultMatrixf((GLfloat*)matUmbra);
				umbra = 1;
				Scena();
			glPopMatrix();
			Mediu();
		glPopAttrib();
	glPopMatrix();
}
```



Toate modulele, mai conțin și o verificare pentru umbre, dacă se desenează umbra, atunci se schimbă culoarea inițială cu o culoare pentru umbra și se dezactivează texturile