Acta et Commentationes, Sciences of Education, nr. 2(40), 2025 ISSN 1857-0623
p. 7-17 E-ISSN 2587-3636

CZU: 37.018.43:004.925:004.43 DOI: 10.36120/2587-3636.v40i2.7-17

DELPHI FMX: ACTIVITĂȚI DE MODELARE-SIMULARE 3D

Nicolae BALMUȘ, dr., conf. univ.
https://orcid.org/0000-0002-0491-2918
Tatiana CHIRIAC, dr., conf. univ.
https://orcid.org/0000-0002-6122-1937
Universitatea Pedagogică de Stat „Ion Creangă” din Chișinău

Rezumat. În lucrare sunt descrise transformările principale de coordonate în baza cărora se realizează proiectarea scenelor 3D pe un plan 2D (ecranul calculatorului). Prin cod de programare Pascal sunt realizate exemple de proiectare pe ecranul calculatorului a unor figuri geometrice 3D. Pentru proiectarea scenelor 3D de orice natură, cu texturi și efecte de iluminare, se recomandă utilizarea mediului de programare Delphi FMX care are incorporate componente de grafică 3D, implementate în baza motorului de grafică OpenGL. În baza exemplelor, descrise detaliat în lucrare, utilizatori pot realiza activități interactive de modelare/simulare 3D cu interfață grafică de înaltă calitate și cod succint de programare.
Cuvinte cheie: programare vizuală, grafică 3D, modelare matematică, simulare asistată de calculator, software educațional.

DELPHI FMX: 3D MODELING-SIMULATION ACTIVITIES

Abstract. This paper describes the main coordinate transformations upon which the design of 3D scenes on a 2D plane (a computer screen) is based. We provide examples of projecting 3D geometric figures onto the computer screen using Pascal programming code. For the design of 3D scenes of any nature, with textures and lighting effects, the use of the Delphi FMX programming environment is recommended, which incorporates 3D graphics components implemented on the OpenGL graphics engine. Based on the examples detailed in this paper, users can perform interactive 3D modelling/simulation activities with a high-quality graphical interface and concise programming code.
Keywords: visual programming, 3D graphics, mathematical modelling, computer-aided simulation, educational software.

1. Introducere

Calitatea reprezentărilor grafice este foarte importantă pentru toate tipurile de aplicații implementate pe calculator, inclusiv pentru cele destinate procesului de instruire. Problema principală care trebuie să fie rezolvată în cazul desenării unei scene trei dimensionale (3D) pe ecranul calculatorului constă în identificarea formulelor cu ajutorul cărora coordonatele (x, y, z) ale unui punct P din spațiul 3D se transformă în coordonatele 2D ale punctului respectiv $P'(x', y')$ pe ecranul calculatorului. Schema de principiu pentru identificarea acestor transformări este reprezentată în figura 1.

În figura 1, direcția de proiectare este definită de unghiurile θ, φ (coordonate sferice) și α, β, γ (formate de direcția de proiectare și axele sistemului de coordonate x, y, z). Relațiile dintre coordonatele x', y' și x, y, z , în caz general au forma:

$$x' = f_1(x, y, z, \theta, \varphi, d)) = F_1(x, y, z, \alpha, \beta, \gamma, d) \quad (1)$$

$$y' = f_1(x, y, z, \theta, \varphi, d)) = F_2(x, y, z, \alpha, \beta, \gamma, d) \quad (2)$$

unde $d = SO'$ este distanța de la punctul de proiectare până la ecran. Dacă punctul de proiectare se află la distanță mare, proiectarea se numește **ortogonală** (paralelă) în caz contrar proiectarea se numește ***perspectivă***.

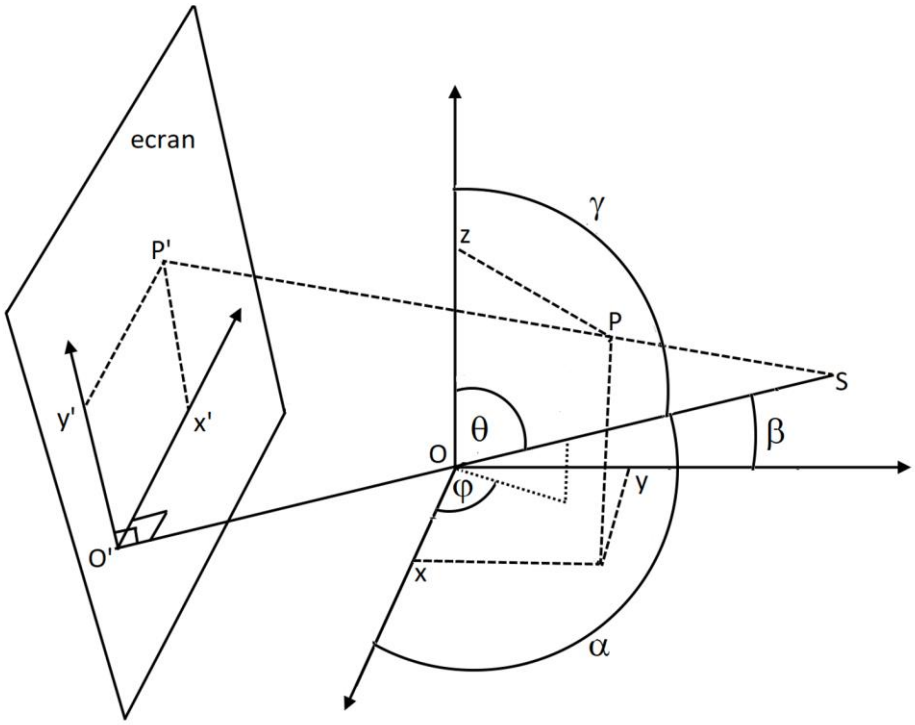

Figura1. Schema de principiu a procesului de proiectare 3D=>2D

Forma explicită a formulelor (1)-(2), pentru proiectarea paralelă este:

$$x' = ycos(\varphi) - xsin(\varphi) \quad (3)$$

$$y' = -zsin(\theta) + xcos(\theta)\cos(\varphi) + ycos(\theta)\sin(\varphi) \quad (4)$$

În cazul când limbajul de programare nu are incorporate instrucțiuni de grafică 3D, pentru proiectarea suprafețelor 3D pe ecranul calculatorului se utilizează următorul algoritm:

1. Suprafața 3D se divizează în părți elementare (*teracote*) de dimensiuni mici, de regulă de formă triunghiulară sau patrulatere.
2. Coordonatele 3D ale vârfurilor *teracotei* se transformă în coordonate 2D cu ajutorul expresiilor (1)-(2).
3. Cu ajutorul instrucțiunilor de grafică 2D, în planul ecranului se desenează toate teracotele începând cu cele mai îndepărtate, consultând de fiecare dată unghiul dintre normala la teracotă și direcția de proiectare. În baza acestui unghi se realizează efectul de iluminare. Dacă acest unghi are valoare cuprinsă între 90°-180° teracota este vizibilă și se desenează; în caz contrar teracota se consideră invizibilă și nu se desenează.

Cu cât dimensiunile *teracotelor* sunt mai mici, cu atât proiectarea 3D este mai calitativă.

2. Exemplu de grafică 3D în aplicații de tip consolă

Limbajele de programare C, Turbo Pascal, Free Pascal, Pascal ABC prin bibliotecile care se atașează suplimentar la aplicațiile consolă permit realizarea reprezentărilor grafice 2D. Reprezentările grafice 3D, în aceste limbaje pot fi realizate utilizând formulele (1)-(4) pentru transformarea coordonatelor 3D în coordonate 2D și procedurile limbajului cu ajutorul cărora se desenează poligoane 2D. Prezentăm în continuare codul integral al aplicației Pascal ABC care proiectează pe ecranul calculatorului un cub cu fețele colorate.

```
uses graphABC;
type Tct=array[0..3,0..2]of real;
const c:array[0..5] of Tct=
((( 1,-1, 1),( 1, 1, 1),(-1, 1, 1),(-1,-1, 1)),
 (( 1,-1, 1),( 1,-1,-1),( 1, 1,-1),( 1, 1, 1)),
 (( 1,-1,-1),(-1,-1,-1),(-1, 1,-1),( 1, 1,-1)),
 ((-1,-1, 1),(-1,-1,-1),( 1,-1,-1),( 1,-1, 1)),
 ((-1, 1, 1),(-1, 1,-1),(-1,-1,-1),(-1,-1, 1)),
 (( 1, 1, 1),( 1, 1,-1),(-1, 1,-1),(-1, 1, 1)));
cl:array[0..5] of Color=(clred,clgreen,clblue,clyellow,clmaroon,cllime);
var   p:array of Point;
var teta,fi,fs, sint,cost,sinfi,cosfi:real;
var x0,y0:integer;
procedure tr3_2(x,y,z:real;var P:Point);
begin
  P.y:=round((z*sint-x*cost*cosfi-y*cost*sinfi)*fs)+y0;
  P.x:=-round((x*sinfi-y*cosfi)*fs)+x0;
end; { tr3_2}
function cosNV(Tc:Tct):real;// Calculeaza cosinusul unghiului dintre normala
la teracota si directia de vizualizare
var x1,x2,x3,y1,y2,y3,z1,z2,z3,nx,ny,nz:real;
begin
  x1:=Tc[0,0]; x2:=Tc[1,0];x3:=Tc[2,0];
  y1:=Tc[0,1]; y2:=Tc[1,1];y3:=Tc[2,1];
  z1:=Tc[0,2]; z2:=Tc[1,2];z3:=Tc[2,2];
  nx:=((y2-y1)*(z3-z2)-(z2-z1)*(y3-y2));
  ny:=-((x2-x1)*(z3-z2)-(z2-z1)*(x3-x2));
  nz:=((x2-x1)*(y3-y2)-(y2-y1)*(x3-x2));
  cosNV:=(sint*cosfi*nx+sint*sinfi*ny+cost*nz);
end;
procedure Drawcrp3D;
var i:integer;
begin {Drawcrp3D}
```

```
  setlength(p,4);
  for i:=0 to 5 do
  if cosNV(c[i])>0 then
  begin
    tr3_2(c[i,0,0],c[i,0,1],c[i,0,2],p[0]);
    tr3_2(c[i,1,0],c[i,1,1],c[i,1,2],p[1]);
    tr3_2(c[i,2,0],c[i,2,1],c[i,2,2],p[2]);
    tr3_2(c[i,3,0],c[i,3,1],c[i,3,2],p[3]);
    Brush.Color:=cl[i];
    FillPolygon(P);
  end;
end;{Drawcrp3D}
begin
  x0:=300;y0:=200;
  fs:=100;
  teta:=45; sint:=sin(teta*pi/180);   cost:=cos(teta*pi/180);
  fi:=45;   sinfi:=sin(fi*pi/180);   cosfi:=cos(fi*pi/180);
  Drawcrp3D;
end.
```

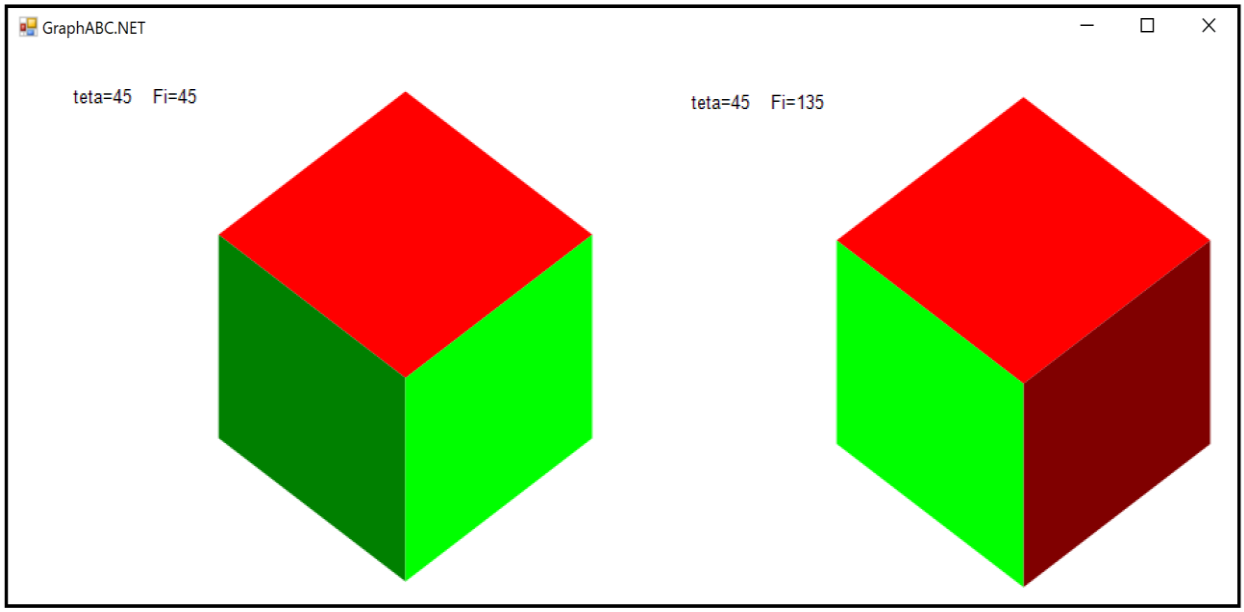

Figura 2. Exemplu de grafică 3D creată prin cod de programare Pascal ABC

3. Grafica 3D în mediul de programare Delphi FMX

Pentru obținerea unor efecte realiste de grafică 3D sunt necesare multe alte transformări de: rotire, scalare, deformare, iluminare, acoperire cu texturi, efecte de umbră etc. De regulă aceste efecte se obțin în baza unor algoritmi efectivi, bazați pe calcule matriceale implementați în diverse biblioteci de grafică. Una din aceste biblioteci este OpenGL care poate fi accesată în diverse limbaje de programare. Utilizarea bibliotecii OpenGL simplifică mult codul de programare al aplicațiilor de grafică, dar necesită

cunoștințe avansate de matematică și geometrie 3D (descrierea oficială a bibliotecii [1] conține 986 pagini). În baza bibliotecii Open GL, autorii Eric Grange și Mike Lischke au creat motorul grafic GL Scene care se instalează suplimentar în mediul de programare Delphi VCL și CBuilder. Cu acest pachet de componente, utilizatorii creează rapid aplicații vizuale cu grafică 3D calitativă și cod minimal de programare.

În mediul de programate Delphi și CBuilder, varianta FMX grafica 3D este incorporată în mod implicit. Pentru crearea aplicațiilor 3D sunt disponibile două variante:

- aplicație 2D în care se implantează ferestre de vizualizare 3D (TViewport3D) preluate din pagina de componente Vewports (figura 3A);
- aplicație 3D în care componentele 2D se accesează prin intermediul componentei Layer3D (figura 3B).

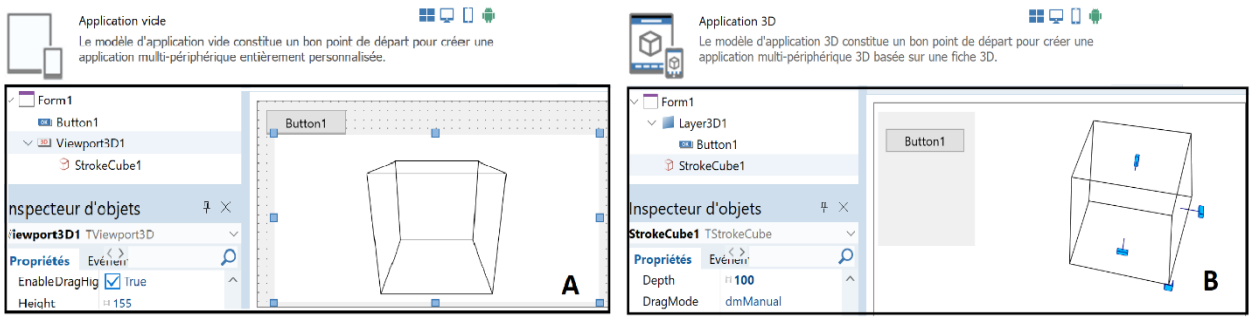

Figura 3. Tipuri de aplicații 3D în Delphi versiunea FMX

Gama completă de componente 3D disponibile în versiunea Delphi FMX este reprezentată în figura 4. Cu ajutorul componentelor din pagina 3D Shapes utilizatorul creează, fără cod de programare forme geometrice 3D: cub, cilindru, disc, sferă, elipsoid, text 3D, etc, cărora cu ajutorul componentelor din pagina Materials li se aplică texturi (TTextureMaterialSource), culori (TColorMaterialSource) lumină emisivă (TLightMaterialSource). Componentele din pagina 3D Scene servesc pentru realizarea efectelor de iluminare a corpurilor 3D (TLight), de proiectare (TCamera). TDummy este un container 3D în care pot fi plasate obiecte 3D care se gestionează (se deplasează, se rotesc) ca un tot întreg.

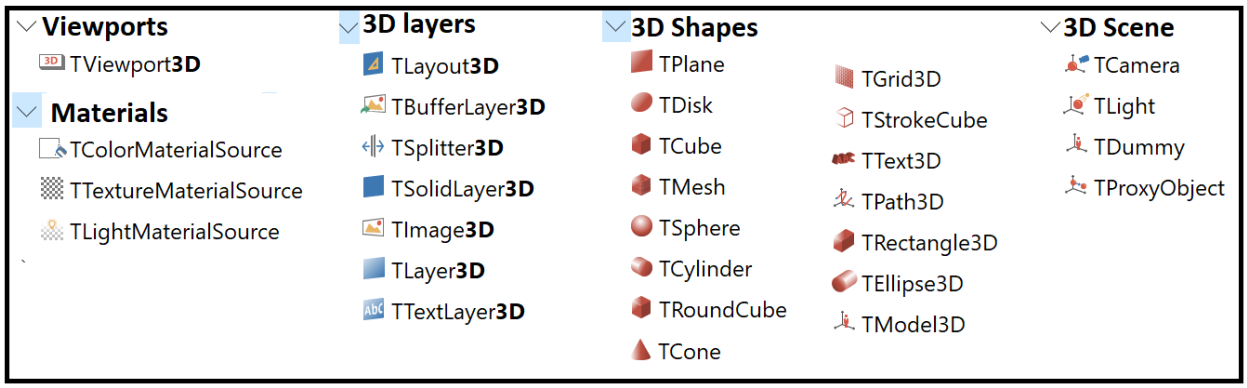

Figura 4. Paleta de componente 3D, Delphi FMX

Prezentăm în continuare câteva exemple de scene 3D și animații create în mediul de programare Delphi FMX.

4. Modelarea și simularea mișcării de rotație a Pământului

În această aplicație vom simula procesul 3D de rotire a Pământului în jurul axei cu efecte de iluminare din partea Soarelui. Pentru acest scop pe forma 2D a aplicației Delphi FMX adăugăm componenta Viewport3D1: TViewport3D din paleta de componente. În inspectorul de obiecte setăm proprietatea Align la valoarea Client. Din paleta 3D Shapes, în Viewport3D1 adăugăm componentele Sphere1(Pământul) și Sphere2 (Soarele) pentru care în inspectorul de obiecte setăm proprietatea Projection la valoarea Screen. În inspectorul de obiecte se setează dimensiunile (Deph, Hight, Width) pentru Sphere1 la valoarea 250, iar pentru Sphere2 la valoarea 25. Designul aplicației la etapa inițială este reprezentat în figura 5.

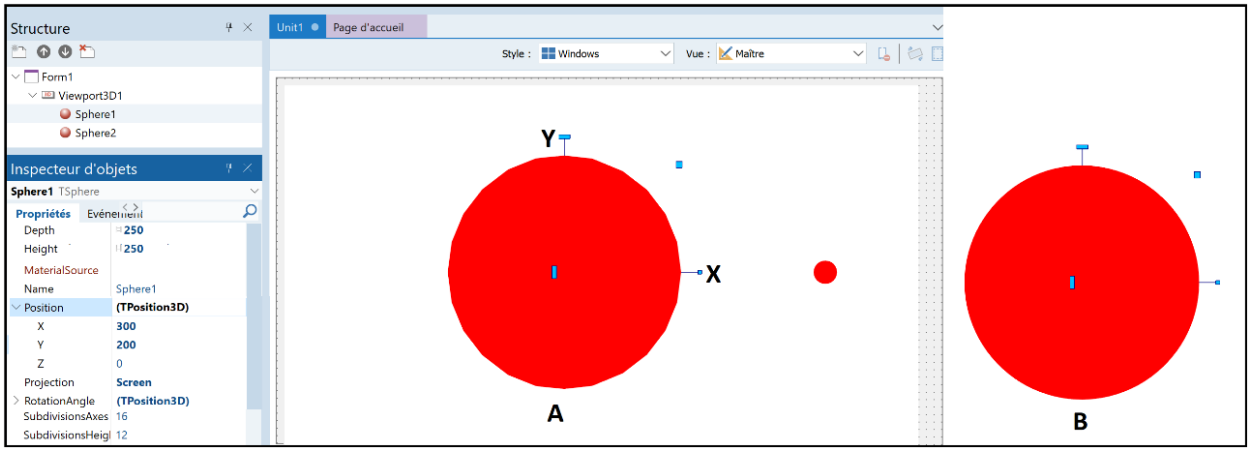

Figura 5. Designul aplicației la etapa inițială

Calitatea vizualizării 3D ale acestor sfere depinde valoarea proprietăților SubdivisionAxes și SubdivizionHeight care determină numărul de „teracote” în baza cărora se realizează proiectarea 2D a sferei 3D. Pentru valorile predefinite 16 și 12, numărul de „teracote” este 192=16*12. În figura 5 se vede că pentru aceste număr de ”teracote” Sphere1 nu se desenează calitativ (conturul nu este suficient de neted). Calitate se îmbunătățește dacă numărul de „teracote” este mai mare. În figura 5B este reprezentată Sphere1 pentru numărul maximal de „teracote” - 2500=50*50.

Etapa aplicării texturilor 3D. Din pagina de componente Materials adăugăm în aplicație obiectele LightMaterialSource1, TextureMaterialSource1. Din internet identificăm și descărcăm o textură potrivită pentru suprafața Pământului (exemplu [2]). Încărcăm imaginea descărcată în proprietatea Texture a obiectului TextureMaterialSource1 (figura 6).

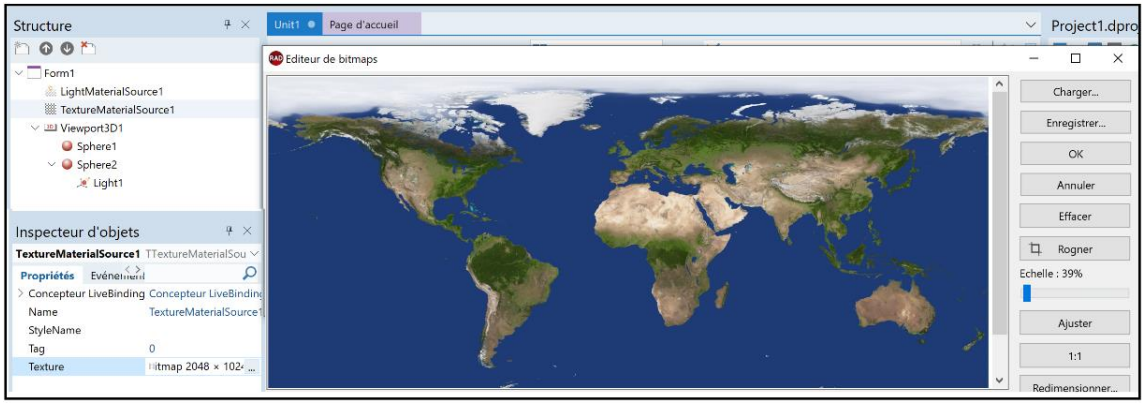

Figura 6. Exemplu de textură 2D

Aplicarea texturii pe obiectul Sphere1 se realizează prin setarea proprietății Materials în care se adaugă obiectul TextureMaterialSource1. Forma aplicației la această etapă este reprezentată în fig. 7A.

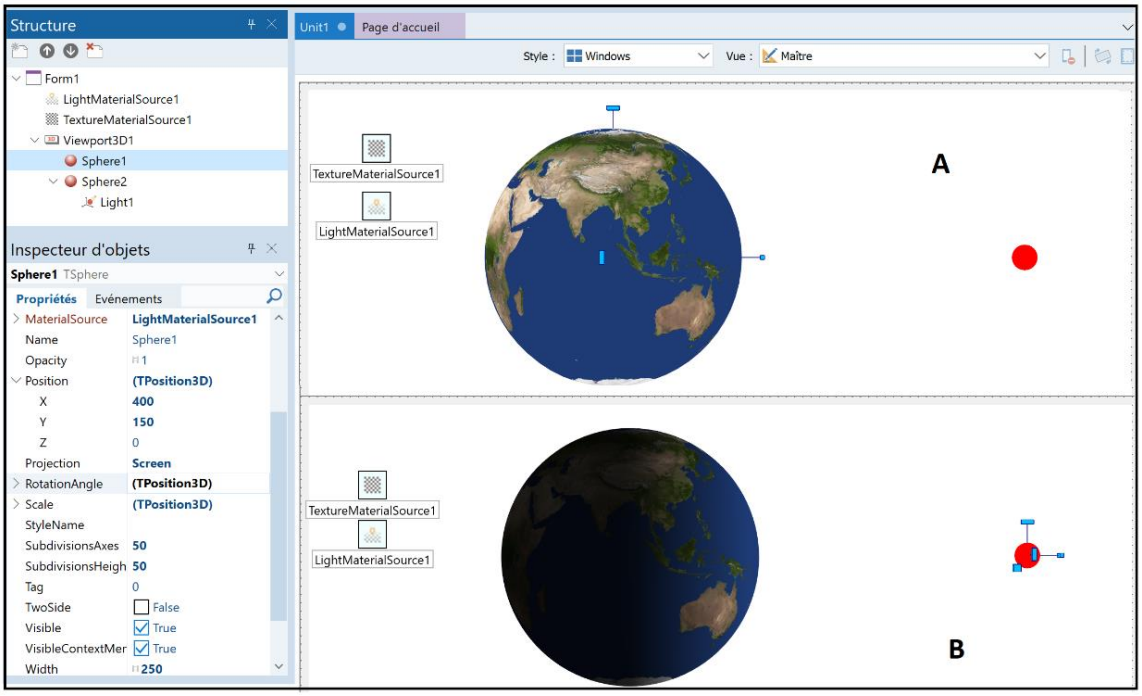

Figura 7. Exemplu de aplicare a texturii pe suprafață sferică

Pentru realizarea efectului de emitere a luminii se utilizează componenta LightMaterialSource1 cu ajutorul căreia se setează proprietățile de emitere și difuzie a luminii (proprietățile Diffuse se setează valoarea White iar Emissive le valoare Null). Pentru realizarea efectului de iluminare a sferei se utilizează componenta Light1 pe care o plasăm în centrul obiectului Sphere1. Proprietatea LightType se setează la valoare Point (sursa de lumină este punctiformă). Ca urmare se obține imaginea din fig. 7B.

Etapa simulării mișcării de rotație a Pământului. Deoarece axa de rotire a Pământului este înclinată față de planul eclipticii sub unghiul 23.5°, pentru Sphere1 setăm ughiul RotationAngle.z la această valoare. Pentru simularea efectului de rotire a

Pământului în jurul axei proprii, adăugăm în aplicație componentele Button1 și Timer1 pentru care programăm evenimentele `Button1Click și Timer1Timer:`

```
var t,TP:Extended;
procedure TForm1.Button1Click(Sender: TObject);
begin
  t:=0;  TP:=24*3600;//TP-perioada de rotire a pământului, în secunde
timer1.Interval:=1; timer1.Enabled:=true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
   t:=t+100;sphere1.RotationAngle.Y:=-360*t/TP;
label1.Text:=(t/3600).ToString +' ore';//Se Afișează timpului în ore
end;
```

În baza acestui cod simplu de programare, la apăsarea butonului Button1 utilizatorul observă procesul de rotire 3D a Pământului iluminat de Soare (figura 8).

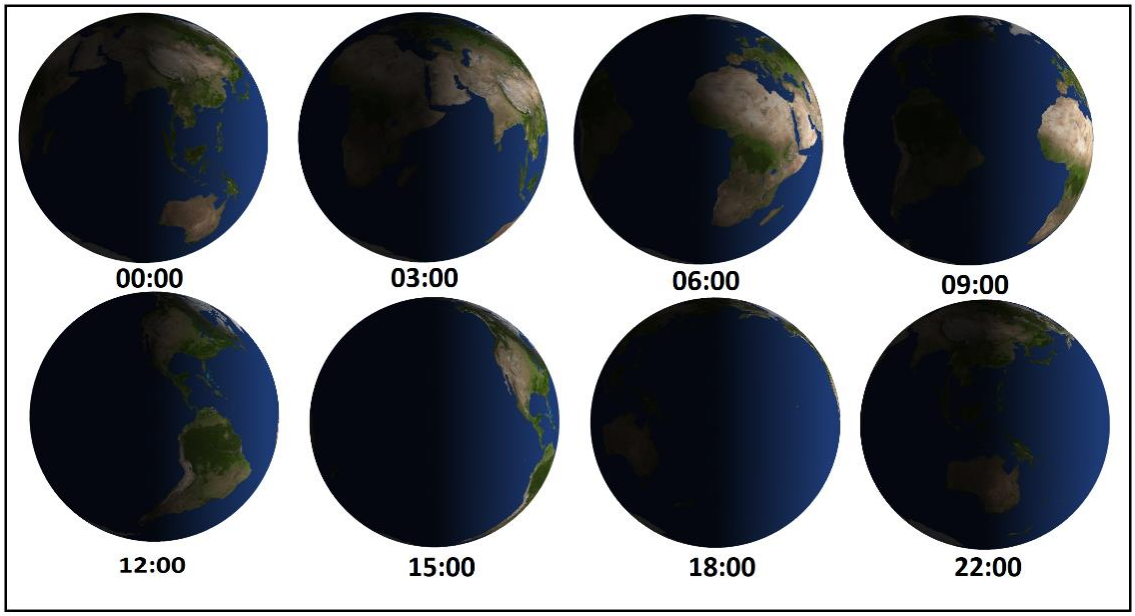

Figura 8. Exemplu de animație 3D

5. Modelarea și simularea 3D a sistemului planetar Soare – Pământ - Luna

Pentru simularea mișcării corpurilor în sistemul Soare-Pământ-Luna în fereastra Viewport3D1: TViewport3D adăugăm 4 obiecte TDummy (Dummy1, Dummy2, Dummy22, Dummy3), care joacă rolul de sisteme de coordonate 3D. În Dummy1, Dummy22 și Dummy3 plasăm câte un obiect TSphere (Sphere1, Sphere2, Sphere3) prin care se modelează Soarele, Pământul, Luna. Setăm parametrii Depth, Height, Width la valoare 150 pentru Sphere2 și respectiv 25 Sphere1 și Sphere3. Adăugăm în aplicație 3 obiecte TLightMatreialSource. În proprietate Texture a acestor obiecte încărcăm texturile corespunzătoare pentru Soare, Pământ, Lună. Pentru obiectele Shere1, Sphere2, Sphere3

setăm proprietatea MaterialSource la valoarea LightMatreialSource1, respectiv LightMatreialSource2, LightMatreialSource3. Pentru orientarea corectă a axei de rotire a Pământului față de planul traiectoriei, obiectul Sphere2 a fost rotit în jurul axe Ox cu 90° și cu 23.5° [3], în jurul axei Oz. Direcția de rotire este indicată de obiectul Cylinder1:TCylinder, inclus suplimentar în aplicație. La această etapă designul aplicației are forma reprodusă în fig. 9. Pentru iluminarea corpurilor din sistem se adaugă componenta Light1:TLight pe care o plasăm în sistemul Dummy2 pe axa X la distanță mare de Pământ (Sphere2) (Light1.Position.X=-10000. Proprietatea LightType pentru Light1 se setează la valoarea Point (sursă punctiformă de lumină) iar Position.X . La această etapă structura și designul aplicație sunt reprezentate în figura 9.

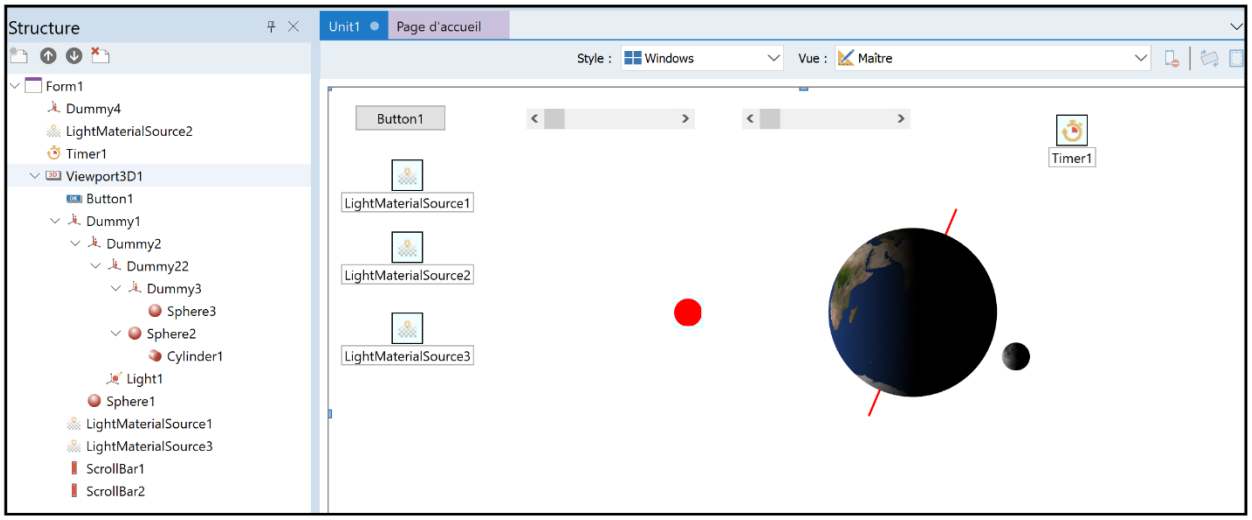

Figura 9. Designul aplicației la etapa de proiectare

Pentru gestionarea animației, în aplicație au fost adăugate componentele Button1, Timer1, Scrollbar1, Scrollbar2. Prezentăm în continuare codul integral al aplicației.

```
var t,TPS,TRP,TLP,dps,DLP:extended;
procedure TForm1.Button1Click(Sender: TObject);
begin t:=0; TPS:=365*24*3600;TRP:=24*3600;TLP:=28*24*3600;
   DPS:=Viewport3D1.Height/2.5; DLP:=100;
   Dummy22.Position.X:=dps;timer1.Enabled:=true;
end;

procedure TForm1.FormResize(Sender: TObject);
begin
dummy1.Position.X:=Viewport3D1.Width/2;
dummy1.Position.Y:=Viewport3D1.Height/2;
DPS:=Viewport3D1.Height/2.5; Dummy22.Position.X:=dps;
end;

procedure TForm1.ScrollBar2Change(Sender: TObject);
begin dummy1.RotationAngle.X:=scrollbar2.Value; end;
```

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
t:=t+scrollbar1.Value;
dummy2.RotationAngle.Z:=t/tps*360;
dummy22.RotationAngle.Z:=-dummy2.RotationAngle.Z;
sphere2.RotationAngle.y:=t/trp*360;
dummy3.RotationAngle.y:=t/tlp*360;
end;
```

În figura 10 sunt reprezentate 4 secvențe ale animației în care utilizatorul observă succesiunea anotimpurilor și fazele Lunii. Cu ajutorul ScrollBar1 se modifică viteza animației iar cu ScrollBar2 se modifică unghiul de proiectare.

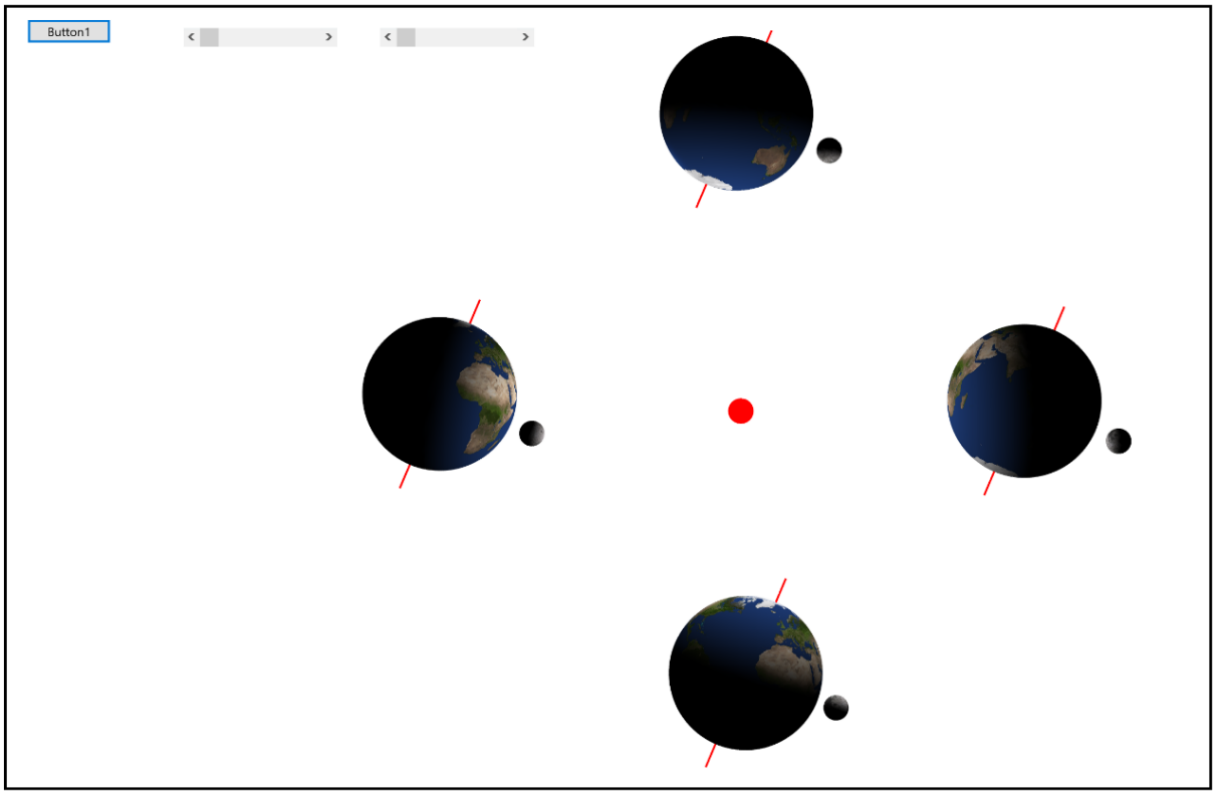

Figura 10. Exemplu de animație 3D

În mod analogic, cu cod de programare minimal pot fi realizate aplicații de modelare simulare a structurilor moleculare și cristaline, prevăzute în manualele de chimie [4]. Pentru aceasta sunt necesare investigații în baza cărora se identifică informația cu privire la coordonatele 3D ale atomilor și dimensiunilor lor relative.

Concluzii

În baza investigațiilor realizate și a codurilor de programare Pascal, prezente în lucrare, utilizatorul final (elevii, studenții, masteranzii, cadrele didactice), cu eforturi

minimale și cunoștințe medii de programare Delphi poate restabili aplicațiile descrise și dezvolta în continuare alte aplicații de modelare-simulare 3D de concepție proprie, necesare pentru activitatea didactică. În procesul de realizare a acestor aplicații se consolidează competențele specifice disciplinelor școlare studiate: fizica, chimia, astronomia, matematica și se dezvoltă noi competențe specifice disciplinei informatica. Subiectul acestei lucrări în mod explicit face referință la competența a noua din curriculumul disciplinei informatica pentru liceu [5]: *Explorarea situațiilor-problemă prin modelare, planificare și efectuare de experimente virtuale în mediile digitale, dovedind spirit analitic, claritate și concizie*.

Bibliografie

1. SHREINER, Dave et al. OpenGL Programming. The Official Guide to Learning OpenGL", Version 4.3, C 2013 Pearson Education, 986 p. https://www.cs.utexas.edu/~fussell/courses/cs354/ handouts/Addison. Wesley. OpenGL.Programming.Guide.8th.Edition.Mar.2013.ISBN.0321773039.pdf
2. Planet Texture Maps https://planet-texture-maps.fandom.com/wiki/Earth
3. MARINCIUC, Mihai, RUSU, Spiridon, NACU, Ion [et al.]. Fizică și Astronomie: Manual pentru clasa a 12-a /; Min. Educației al Rep. Moldova. Chișinău: I.E.P. *Ştiinţa*, 2017. 168 p. ISBN 978-9975-85-074-2.
4. DRAGALINA, Galina, VELIȘCO, Nadejda, BULMAGA, Petru [et al.]. Chimie: Manual pentru clasa a 12-a: Profil real. Profil umanist / Ed. a 2-a. Chișinău: Arc, 2017. 192 p. ISBN 978-9975-0-0006-2.
5. Curriculum național. Disciplina Informatică, clasele X-XII. Chișinău 2019. https://mecc.gov.md/sites/default/files/informatica_curriculum_liceu_rom.pdf

Recepționat / Received: 02.05.2025
Acceptat / Accepted: 19.06.2025
Email: balmus.nicolae@upsc.md
chiriac.tatiana@upsc.md