Transformari 3D

Sistem de coordonate

În geometrie, un sistem de coordonate reprezintă o modalitate prin care oricărui punct i se asociază în mod unic o mulțime ordonată de numere reale, numite coordonatele acelui punct. În spațiul euclidian sunt necesare trei coordonate (abscisa, ordonata și cota), în plan sunt necesare două (abscisa și ordonata), iar pentru localizarea punctelor pe o dreaptă este necesară doar o coordonată. În geometria analitică, utilizarea sistemelor de coordonate permite transformarea problemelor de geometrie în probleme de algebră.

Vom utiliza coordonate omogene ca și in cazul 2D

În matematică, coordonatele omogene, introduse de August Ferdinand Möbius, permit transformări afine prin reprezentarea lor sub forma unei matrici. Coordonatele omogene permit, de asemenea, efectuarea calculelor în spații proiective într-un mod similar cu cel în care coordonatele carteziene o fac în spaţiul euclidian.

Coordonatele omogene ale unui punct din spatiu proiectiv de dimensiune n sunt de obicei scrise ca (x: y: z: ...: w), un vector linie de lungime n + 1, altele decât (0: 0: 0: .. .: 0). Două seturi de coordonate, care sunt proporționale denotă același punct din spatiul proiectiv: pentru orice non-zero c scalar din domeniu care stă la baza K, (cx : cy : cz : ... : cw) reprezintă același punct. Prin urmare, acest sistem de coordonate poate fi explicat după cum urmează: în cazul în care spațiul proiectiv este construit dintr-un spațiu vectorial V de dimensiune n + 1, se introduc coordonatele în V, prin alegerea unei baze, și utilizarea acestora în P (V), clasele de echivalență proporționale non-zero vectori în V.

Transformările vor fi prezentate prin matrice 4x4

Vom utiliza sistemul de coordonate de dreapta (right-handed)

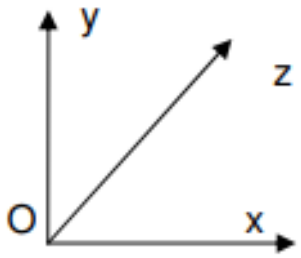

Fig.1 Sistemul de coordonate orientat stînga.

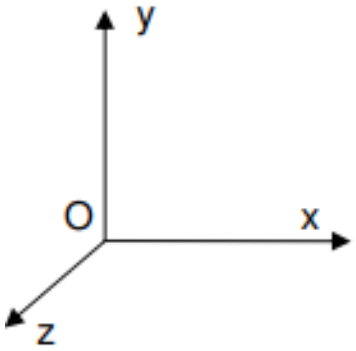

Fig.2 Sistemul de coordonate orientat dreapta.

Punct Un punct P din R^3 se poate preciza prin:

• Coordonate carteziene: P(x,y,z)

• Coordonate omogene: P(x,y,z,u)

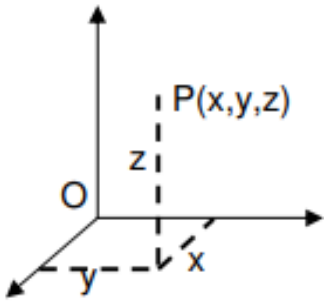

Fig.3 Sistem de coordonate carteziene.

Transformari 3D de baza

Translaţia

$$T(t_x, t_y, t_z) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotaţia in jurul unei axe

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scalarea

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformari 3D compuse

Compunerea transformărilor

Matricea generalizata

Structura pentru cazul notației prin vector-coloana:

● matricea 3x3 include transformări de scalare locală, forfecare, oglindire şi rotație

● matricea 3x1 reprezintă transformarea de translație

● matricea 1x3 reprezintă transformarea de proiectare perspectivă

● matricea 1x1 reprezintă transformarea de scalare generală

$$\begin{bmatrix} & & & \vdots & 3 \\ & 3\times 3 & & \vdots & \times \\ & & & \vdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 3 & \times & 1 & \vdots & 1\times 1 \end{bmatrix}$$

Matricea corespunzătoare transformării compuse se obține prin înmulțirea matricelor transformărilor elementare. Deoarece înmulțirea matricelor nu este comutativă, este importantă ordinea în care se aplică aceste transformări

Matricea de transformare cea mai apropiată vectorului linie (sau a vectorului colana) corespunde primei transformări care se aplică în timp ce matricea de transformare cea mai depărtată este ultima dintre cele aplicate

Matematic aceasta se exprimă prin:

[M] [VC] = [Mn] …… [M3] [M2] [M1].... [VC]

pentru vector coloana

sau

[VL] [M] =[VL] [M1]^T [M2]^T [M3]^T ... [Mn]^T

pentru vector linie unde [Mi] poate îi orice matrice de transformare

Scopul lucrării: Crearea unei scene dinamice 3D.

Sarcina lucrării de laborator constă în conceperea și construirea unei scene dinamice 3D care ar conține transformările: translație, rotație și scalare. Realizarea scenei cu ajutorul editorului online p5.js utilizînd modele de obiecte 3D stocate în fișiere .OBJ care pot fi create individual sau descărcate din internet.

Mersul lucrării:

Pentru exemplificarea mersului lucrării de laborator nr. 4 se propune crearea unei scene dinamice care ar reprezenta o pompă de apă care își orientează elicea în dependență de direcția vîntului. Modelul pompei este divizat în trei componente: bază (Base), elice (Fan) și corp (Tail). Fiecare componentă este amplasată într-un fișier .skp aparte cu amplasarea modelului astfel încît originea sistemului de coordonate și orientarea axelor acestuia să coincidă cu originea și orientarea sistemului de coordonate global al scenei. Componentele 3D ale modelului pot fi create de la zero sau modelul pote fi importat din repozitoriul 3D Warehouse accesibil din meniul de bază al

editorului grafic 3d Google SketchUp Window->3D Warehouse, apoi editat corespunzător după necesitate.

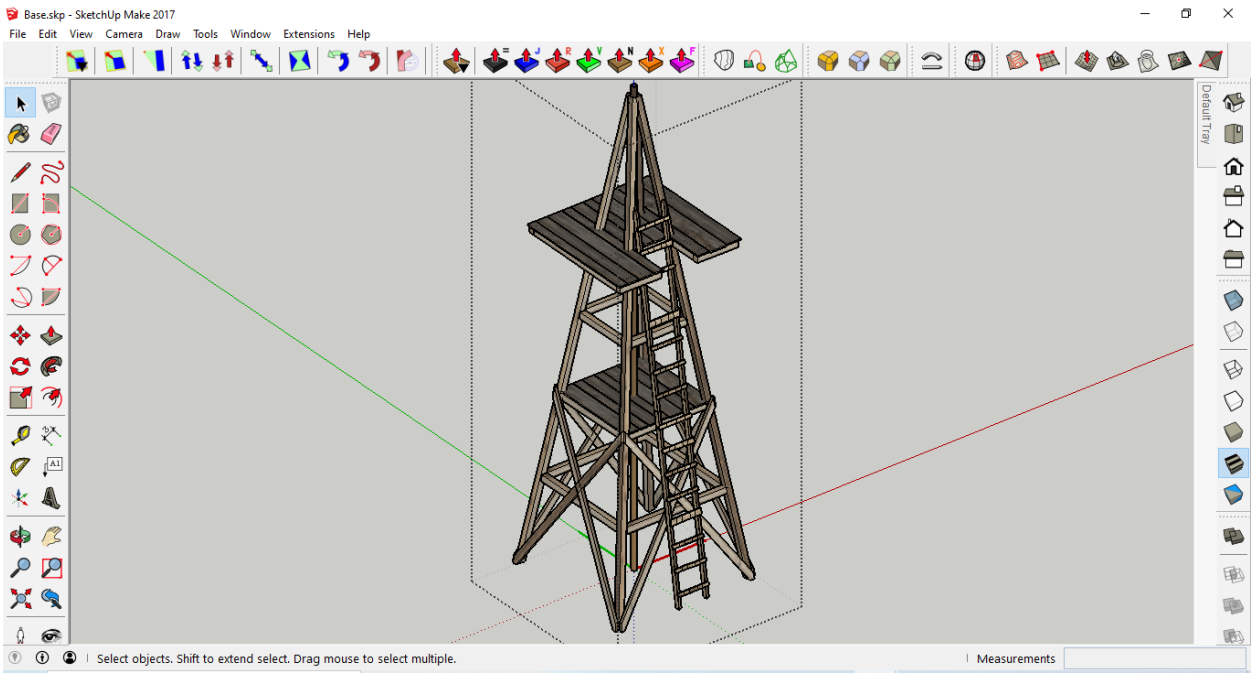

Fig. 4 Modelul bazei pompei de apă.

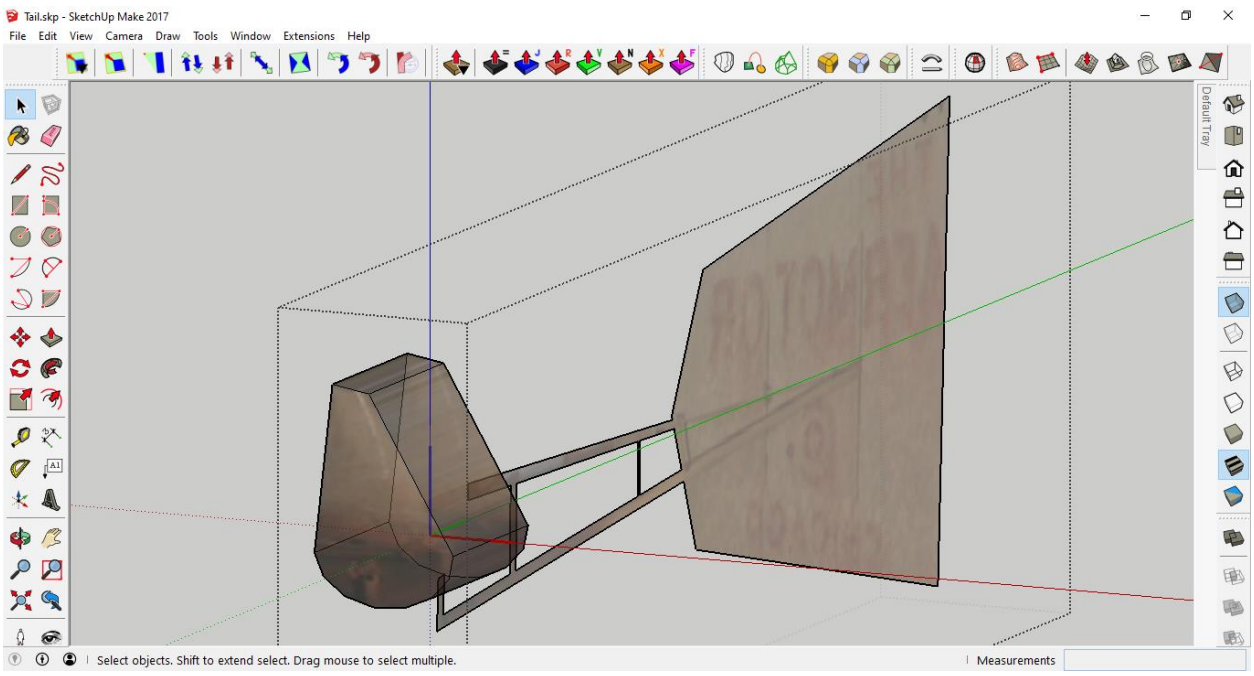

Fig. 5 Modelul corpului pompei de apă.

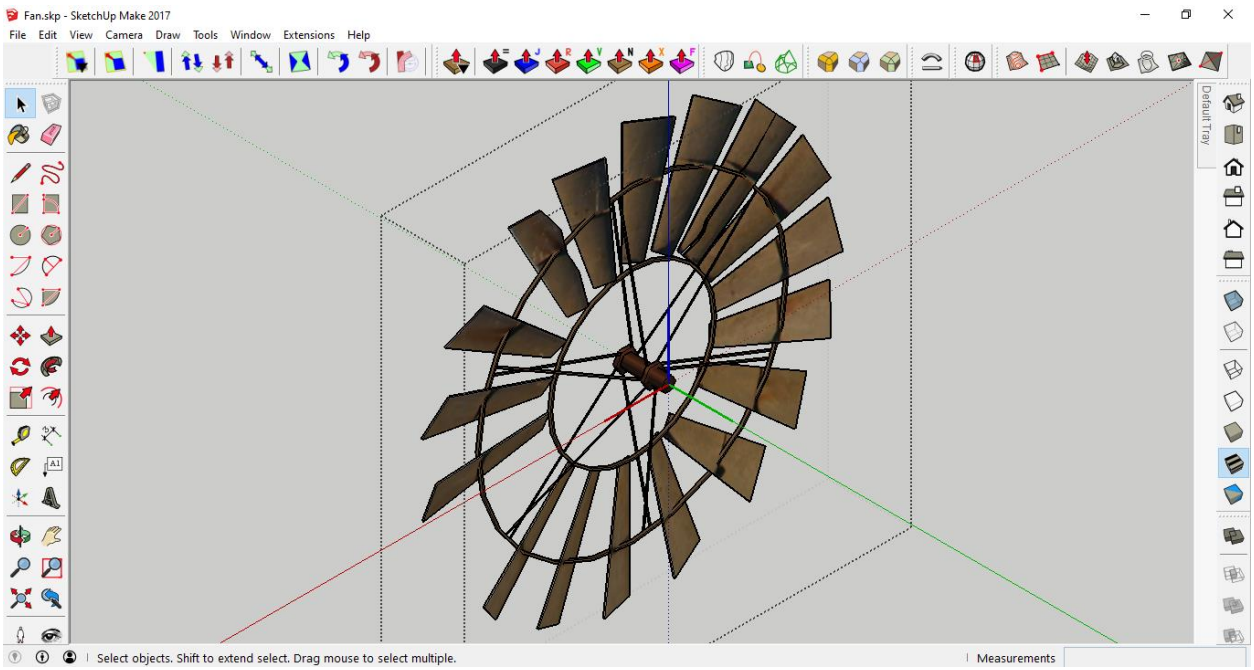

Fig. 6 Modelul elicei pompei de apă.

Pentru a avea posibilitatea de a downloada sau importa modelul selectat în sketch-ul curent utilizatorul trebuie să dispună de un cont Google Account și să fie autentificat în mediul 3D Warehouse. După ce a fost downloadat și editat corespunzător fiecare component al modelului, acestea pot fi editate după necesitate și exportate ca fișiere .OBJ prin intermediului plug-inului *LIPID OBJ Exporter* metoda de instalare a căruia a fost prezentată în cadrul lucrării de laborator Nr 3 .

Descrierea funcțiilor de bază din biblioteca P5.JS

preload()

Apelată chiar înainte de funcția ***setup()***, funcția ***preload()*** este utilizată pentru a gestiona încărcarea asincronă a fișierelor externe. Dacă este definită funcția de preîncărcare, funcția ***setup()*** va aștepta până la finalizarea oricărui apel de încărcare. Nimic în afară de apelurile de încărcare (*loadImage, loadJSON, loadFont, loadStrings* etc.) nu ar trebui să fie în interiorul funcției de preîncărcare. Dacă este preferată încărcarea asincronă, metodele de încărcare pot fi apelate în setup () sau în orice alt loc cu ajutorul unui parametru de apel invers. Nimic altceva nu ar trebui să fie realizat în interiorul funcției de ***preload()*** în afară de apelurile de încărcare, toate celelalte inițializări ar trebui să fie realizate în interiorul funcției ***setup()***.

Sintaxă:

preload()

loadModel()

Încărcarea unui model 3D dintr-un fișier OBJ sau STL. Funcția ***loadModel()*** trebuie plasată în interiorul funcției ***preload()***. Acest lucru permite modelului să se încarce complet înainte de a rula restul programului.

Una dintre limitările formatului OBJ și STL este că nu are un sens de scară încorporat. Aceasta înseamnă că modelele exportate din diferite editoare grafice 3D pot avea dimensiuni diferite. Dacă modelul nu se afișează, metoda ***loadModel()*** poate fi apelată cu valoarea true a parametrului de normalizare. Aceasta va redimensiona modelul la o scară adecvată pentru p5. De asemenea, pot fi făcute modificări suplimentare ale dimensiunii finale a modelului utilizînd funcția ***scale()***. Un alt neajuns îl reprezintă faptul că biblioteca p5.js nu asigură suportul pentru fișiere STL colorate și fișiere OBJ în care conțin definiția materialelor, texturilor și culorilor. Acestea vor fi redate fără proprietăți de materiale, texturi și culori.

Sintaxă:

loadModel(path, normalize, [successCallback], [failureCallback], [fileType])

loadModel(path, [successCallback], [failureCallback], [fileType])

Parametrii:

path	String: Calea modelului de încărcat
normalize	boolean: dacă este adevărat, se scalează modelul la o dimensiune standardizată la încărcare
funcția successCallback	(p5.Geometry): funcție care trebuie apelată după încărcarea modelului. Va fi transmis obiectul model 3D. (Opțional)
failureCallback	(Event): funcție apelată cu eveniment eroare de dacă modelul nu reușește să se încarce. (Opțional)
fileType	String: Extensia de fișier a modelului (.stl, .obj). (Opțional)

Returnează:

obiect de tipul p5.Geometry

setup()

Funcția ***setup()*** este apelată o singuură dată când pornește programul. Este folosită pentru a defini proprietățile mediului inițial, cum ar fi dimensiunea ecranului, culoarea de fundal și pentru a încărca suporturi media, cum ar fi imagini și fonturi, pe măsură ce rulează programul. Nu poate exista decât o singură funcție ***setup()*** în fiecare program și nu trebuie apelată repetat după executarea sa inițială.

Notă: Variabilele declarate în interiorul funcției ***setup()*** nu sunt accesibile în alte funcții, inclusiv în funcția ***draw()***.

Sintaxă:

setup()

creatCanvas()

Creează un element de canvas în documentul HTML și setează dimensiunile acestuia în pixeli. Această metodă trebuie apelată o singură dată la începutul setării inițiale. Apelarea createCanvas() de mai multe ori într-un proiect va avea ca rezultat un comportament imprevizibil. Pentru crearea mai multor pânze de desen, poate fi utilizată funcția createGraphics (ascuns în mod implicit, dar poate fi afișat).

În modul 2D (adică când p5. Renderer nu este setat) originea (0,0) este poziționată în partea stângă sus a ecranului. În modul 3D (adică când p5.Renderer este setat la WEBGL), originea este poziționată în centrul pânzei.

Variabilele de sistem lățime și înălțime sunt setate de parametrii acestei funcții. Dacă createCanvas() nu este utilizat, ferestrei va primi o dimensiune implicită de 100x100 pixeli.

Sintaxă:

createCanvas (w, h, [renderer])

Parametrii:

w Number: lățimea pânzei

h Number: înălțimea pânzei

render constant: P2D, sau WEBGL (opțional)

Returnează:

p5.Renderer:

normalMaterial()

Materialul normal pentru geometrie este un material care nu este afectat de lumină. Nu este reflectant și este un material substituent folosit adesea pentru depanare. Suprafețele orientate spre axa X devin roșii, cele orientate spre axa Y devin verzi și cele orientate spre axa Z devin albastre.

Sintaxă:

normalMaterial()

frameRate()

Specifică numărul de cadre care trebuie afișate în fiecare secundă. De exemplu, apelul funcției ***frameRate(30)*** va încerca să redeseneze scena de 30 de ori pe secundă. Dacă procesorul nu este suficient optimizat pentru a menține rata specificată, rata cadrelor nu va putea fi asigurată. Se recomandă setarea

ratei cadrelor în interiorul funcției ***setup()***. Rata implicită a cadrelor se bazează pe rata cadrelor a afișajului (numită aici și „rată de reîmprospătare”), care este setată la 60 de cadre pe secundă pe majoritatea computerelor. O rată de cadre de 24 de cadre pe secundă (de obicei pentru filme) sau mai mare va fi suficientă pentru animații fluide. Această funcție este analogică funcției ***setFrameRate (val)***.

Apelul funcției frameRate() fără argumente returnează frecvența curentă a cadrelor. Funcția ***draw()*** trebuie să ruleze cel puțin o dată înainte ca funcția ***frameRate()*** să returneze o valoare. Această funcție este analogică funcției ***getFrameRate()***. Apelul funcției ***frameRate()*** cu argumente care nu sunt de tipul **number** sau care nu sunt pozitive returnează, de asemenea, frecvența curentă a cadrelor.

Sintaxă:

frameRate (fps)

frameRate ()

Parametri:

Number fps: numărul de cadre care trebuie afișate în fiecare secundă

angleMode()

Setează modul curent de reprezentare a unității de măsură a unghiurilor în p5 în radiani sau grade. Modul implicit reprezentare a unității de măsură a unghiurilor este în radiani.

Sintaxă:

angleMode (mode)

Parametri:

mode constant: RADIANS, sau DEGREES

draw()

Apelată direct după ***setup()***, funcția ***draw()*** execută continuu liniile de cod conținute în blocul său până când programul este oprit sau se apelează ***noLoop()***. Dacă funcția ***noLoop()*** este apelată în interiorul funcției ***setup()***, funcția ***draw()*** va fi executată încă o dată înainte de oprire. Funcția ***draw()*** este apelată automat și nu trebuie apelată în mod explicit. Apelul funcției ***draw()*** ar trebui să fie întotdeauna controlat cu ***noLoop()***, ***redraw()*** și ***loop()***. După ce ***noLoop()*** oprește executarea codului din ***draw()***, ***redraw()*** face ca codul din ***draw()*** să se execute o singură dată, iar apelul funcției ***loop()*** va relua executarea ciclică ca codul din funcția ***draw()***. Numărul de execuții al funcției ***draw()*** în fiecare secundă poate fi controlat cu ajutorul funcției ***frameRate()***. Poate exista o singură funcție ***draw()*** pentru fiecare scenă în fiecare proiect. Funcția ***draw()*** trebuie să existe dacă pentru ca codul să ruleze continuu sau să proceseze evenimente de tipul ***mousePressed()***.

Trebuie specificat faptul că toate transformările realizate asupra sistemului de coordonate al proiectului vor fi resetate la începutul fiecărui apel al funcției ***draw()***. Dacă se efectuează transformări în cadrul funcției ***draw()*** (ex: scalare, rotire, translare), efectele acestora vor fi anulate la începutul funcției

draw(), astfel transformările nu se vor acumula în timp. Pe de altă parte, stilul aplicat (ex: umplere, linie, etc.) va rămâne în vigoare.

Sintaxă:

draw()

orbitControl()

Permite mișcarea în jurul unei scene 3D folosind un mouse sau un trackpad. Făcând clic stânga și mișcînd mouse-ul se va roti poziția camerei în jurul centrului scenei, făcând clic dreapta și mișcînd mouse-ul se va deplasa poziția camerei fără rotație, iar folosind roata mouse-ului (scrolling) se va mișca camera mai aproape sau mai departe de centrul scenei. Această funcție poate fi apelată cu parametri care specifică sensibilitatea la mișcarea mouse-ului de-a lungul axelor X și Y. Apelarea acestei funcții fără parametri este echivalentă cu apelul funcției ***orbitControl(1, 1)***. Pentru a inversa direcția de mișcare în ambele axe, trebuie introdusă o valoare negativă pentru sensibilitate.

Sintaxă:

orbitControl([sensitivityX], [sensitivityY], [sensitivityZ])

Parametrii:

sensitivityX	Number: sensibilitate la mișcarea mouse-ului de-a lungul axei X (Opțional)
sensitivityY	Number: sensibilitate la mișcarea mouse-ului de-a lungul axei Y (opțional)
sensitivityZ	Number: sensibilitate la mișcarea de derulare de-a lungul axei Z (opțional)

background()

Funcția ***background()*** setează culoarea utilizată pentru fundalul scenei în p5.js. Fundalul implicit este transparent. Această funcție este de obicei utilizată în interiorul funcției ***draw()*** pentru a șterge fereastra de afișare a fiecărui cadru, dar poate fi utilizată și în interiorul funcției ***setup()*** pentru a seta fundalul primului cadru al animației sau dacă fundalul trebuie setat o singură dată.

Culoarea este fie specificată în unul din formatele RGB, HSB sau HSL, în funcție de modul color actual setat. Formatul de culoare implicit este RGB, cu fiecare valoare în intervalul de la 0 la 255. În mod implicit, intervalul *alfa* variază de la 0 la 255.

Dacă este furnizat un singur argument, sunt acceptate formatele de culori RGB, RGBA și Hex CSS și toate formatele de culoare denumite. În acest caz, nu este acceptată valoarea pentru parametrul *alfa* ca al doilea argument, trebuie utilizat formularul RGBA.

Un obiect p5.Color poate fi, de asemenea, utilizat pentru a seta culoarea de fundal.

Puteți utiliza și o imagine in p5.js pentru a seta imaginea de fundal.

Sintaxă:

background(color)

background(colorstring, [a])

background(gray, [a])

background(v1, v2, v3, [a])

background(values)

background(image, [a])

Parametri:

color	p5.Color: orice valoare creată de funcția color()
colorstring	String: șir de culoare, formatele posibile includ: întreg rgb() sau rgba(), procent rgb() sau rgba(), hex de 3 cifre, hex de 6 cifre
a	Number: opacitatea fundalului în raport cu gama de culori curentă (implicit este 0-255) (Opțional)
gray	Number: specifică o valoare între alb și negru
v1	Number: roșu sau nuanță (în funcție de formatul de culoare curent)
v2	Number: verde sau valoare de saturație (în funcție de formatul de culoare curent)
v3	Number: albastru sau valoarea luminozității (în funcție de formatul de culoare curent)
values	Number[]: o matrice care conține componentele roșu, verde, albastru și alfa ale culorii
image	p5.Image: imagine creată cu ajutorul funcției ***loadImage()*** sau ***createImage()***, pentru a seta ca fundal (imaginea trebuie să fie de aceeași dimensiune ca scena)

noStroke()

Dezactivează desenarea liniei de contur. Dacă sunt apelate atât funcția ***noStroke()***, cât și ***noFill()***, nimic nu va fi afișat pe ecran.

Sintaxă:

noStroke ()

push() și pop()

Funcția ***push()*** salvează setările și transformările curente ale stilului de desen, în timp ce ***pop()*** restabilește aceste setări. Trebuie menționat faptul că aceste funcții sunt utilizate întotdeauna împreună. Aceste funcții permit modificarea stilului și transformărilor pentru a reutiliza ulterior la setările anterioare. Când o nouă stare este pornită cu ***push()***, aceasta se bazează pe stilul curent și transformă informațiile. Funcțiile ***push()*** și ***pop()*** pot fi încorporate pentru a oferi mai mult control.

Funcția ***push()*** stochează informații legate de transformărilr curente și setările de stil controlate de următoarele funcții: ***fill(), noFill(), noStroke(), stroke(), tint(), noTint(), strokeWeight(), strokeCap(), strokeJoin(), imageMode(), rectMode(), ellipseMode(), colorMode(), textAlign(), textFont(), textSize(), textLeading(), applyMatrix(), resetMatrix(), rotate(), scale(), shearX(), shearY(), translate(), noiseSeed()***.

În modul ***WEBGL*** sunt stocate setări de stil suplimentare. Acestea sunt controlate de următoarele funcții: ***setCamera(), ambientLight(), directionalLight(), pointLight(), texture(), specularMaterial(), shininess(), normalMaterial()*** și ***shader()***.

Sintaxă:

push()

pop()

scale()

Mărește sau scade dimensiunea unui obiect prin extinderea sau contractarea vârfurilor. Obiectele se scalează întotdeauna în raport cu originea lor relativă a sistemului de coordonate. Valorile scalei sunt specificate ca procente zecimale. De exemplu, apelul funcției ***scale(2.0)*** mărește dimensiunile unui obiect cu 200%.

Transformările se aplică la tot ceea ce se întâmplă după și apelurile ulterioare ale funcție amplifică efectul acestora. De exemplu, apelul funcției ***scale(2.0)*** și apoi ***scale(1.5)*** este echivalent cu ***scale(3.0)***. Dacă funcția ***scale()*** este apelată în interiorul funcției ***draw()***, transformarea este resetată la fiecare apel al acestei funcții.

Utilizarea acestei funcții cu parametrul ***z*** este disponibilă numai în modul ***WEBGL***. Această funcție poate fi controlată suplimentar cu ajutorul funcției ***push()*** și ***pop()***.

Sintaxă:

scale(s, [y], [z])

scale(scales)

Parametri:

s	Number | p5.Vector | Number[]: procent pentru a scala obiectul sau procent pentru a scala obiectul pe axa x dacă sunt date mai multe argumente
y	Number: procentul de scalare a obiectul pe axa y (opțional)
z	Number: procentul de scalare a obiectul pe axa z (numai în regim webgl) (opțional)
scales	p5.Vector | Number[]: procentul de scalare a obiectului

translate()

Specifică valoarea pentru deplasarea obiectelor din scenă. Parametrul ***x*** specifică translarea stânga/dreapta, parametrul ***y*** specifică translarea sus/jos.

Transformările sunt cumulative și se aplică la tot ceea ce se întâmplă după și apelurile repetate către funcția dată acumulează efectul. De exemplu, apelarea ***translate(50, 0)*** și apoi a ***translate(20, 0)*** este echivalent cu apelul ***translate(70, 0)***. Dacă ***translate()*** este apelat în interiorul funcției ***draw()***, transformarea este resetată în momentul apelului repetat al acestei funcții. Această funcție poate fi controlată cu ajutorul funcțiilor ***push()*** și ***pop()***.

Sintaxă:

translate(x, y, [z])

translate(vector)

Parametri:

x	Number: translează la stânga/dreapta
y	Number: translează în sus/jos
z	Number: translează înainte/înapoi (numai webgl) (opțional)
vector	p5.Vector: vectorul cu care se translează

rotate()

Rotește un obiect cu cantitatea specificată de parametrul unghiului. Această funcție reprezintă angleMode, astfel încât unghiurile pot fi introduse fie în radiani (RADIANS), fie în grade (DEGREES).

Obiectele sunt întotdeauna rotite în jurul poziției lor relative la origine, iar numerele pozitive rotesc obiectele în sensul acelor de ceasornic. Transformările se aplică la tot ceea ce se întâmplă după și apelurile repetate ale acestei funcții cumulează efectul acestora. De exemplu, apelarea ***rotate(HALF_PI)*** și apoi ***rotate(HALF_PI)*** este echivalent cu ***rotate(PI)***. Toate transformările sunt resetate la apelul repetat al funcției ***draw()***.

Tehnic, funcția ***rotate()*** realizează înmulțirea matricei de transformare curente cu o matrice de rotație. Această funcție poate fi controlată cu ajutorul funcțiilor ***push()*** și ***pop()***.

Sintaxă:

rotate(angle, [axis])

Parametri:

angle	Number: unghiul de rotație, specificat în radiani sau grade, în funcție de modul curent de reprezentarea a unghiurilor
axis	p5.Vector | Număr []: (în regim 3d) axa de rotit (Opțional)

rotateX(), rotateY(), rotateZ()

Rotește un obiect în jurul axei ***X*** cu valoarea unghiului specificată în parametrul ***angle***. Unghiurile pot fi introduse fie în radiani (RADIANS), fie în grade(DEGREES).

Obiectele sunt întotdeauna rotite în jurul poziției lor relative la origine, iar numerele pozitive rotesc obiectele în sensul acelor de ceasornic. Toate transformările sunt resetate înaintea unui nou apel al funcției draw().

Sintaxă:

rotateX(angle)

rotateY(angle)

rotateZ(angle)

Parametri:

angle	Number: unghiul de rotație, specificat în radiani sau grade, în funcție de modul curent de reprezentare a unghiurilor

model()

Redă un model 3D în scenă.

Sintaxă:

model(model)

Parametri:

model	p5.Geometrie: Modelul 3D încărcat care trebuie redat

Listingul programului:

```
let fr = 30;
let angle = 0;
function preload()
{
  base = loadModel('Base.obj');
  fan = loadModel('Fan.obj');
  tail = loadModel('Tail.obj');
}
function setup()
{
  createCanvas(400, 400, WEBGL);
```

```
  normalMaterial();
  frameRate(fr);
  angleMode(DEGREES);
}
function draw()
{
  angle = 45*sin(millis()/50);
  orbitControl();
  background(50);
  noStroke();
  push();
  scale(0.025);
  translate(0, 0, -4000);
  model(base);
  pop();
  push();
  scale(0.025);
  rotateZ(angle);
  translate(0, -200, 4650);
  rotateY(millis()/10);
  model(fan);
  pop();
  push();
  scale(0.025);
  translate(0, 0, 4650);
  rotateZ(angle);
  model(tail);
  pop();
}
```

Descrierea programului:

Variabila ***fr*** păstrează numărul de cadre redate pe secundă, numărul de apeluri a funcției draw() pentru redarea scenei.

Variabila ***angle*** este destinată pentru păstrarea unghiului de abatere (orientare) a corpului pompei de vînt și a a elicei.

În funcția ***preload()*** destinată pentru încărcarea modelelor obiectelor 3D ale bazei (***Base.obj***), elicei (***Fan.obj***) și corpului (***Tail.obj***) modelului pompei de apă cu acțiune eoliană stocate în fișiere .OBJ, Fig. ?.

Modul de adăugre în proiect a fișierelor .obj care conțin geometria obiectelor 3D este descris în cadrul lucrării de laborator nr. 3.

În funcția ***setup()*** care este apelată o singură dată la lansarea programului, sînt realizate toate configurările necesare pentru a lucra în regim 3D cum ar fi apelul funcției ***createCanvas(400, 400, WEBGL)*** care creează o scenă 3D redată cu ajutorul unui canvas cu dimensiunile 400 pe 400 de pixeli.

Este apelată funcția ***normalMaterial()*** cu ajutorul coreia este setat un material normal pentru geometrie care este un material care nu este afectat de lumină, nu este reflectant și iar uprafețele orientate spre axa ***X*** devin roșii, cele orientate spre axa ***Y*** devin verzi și cele orientate spre axa ***Z*** devin albastre.

Cu ajutorul funcției ***frameRate(fr)*** este setată rata de redare a cadrelor egală cu 30 de cadre pe secundă.

Cu ajutorul funcției ***angleMode(DEGREES)*** este setat modul de reprezentare a gradelor în grade.

Ulterior este apelată repetat cu frecvența de 30 de ori pe secundă funcția ***draw()*** care desenează repetat scena. Variabila angle stochează valoarea în grade a unghiului de orientare a pompei. Această valoare variază în timp în dependență de timpul care a trecut de la începutul lansării programului reprezentat în milisecunde. Valoarea unghiului variază în timp în intervalul de la -45 la +45 de grade. Această variație este realizată cu ajutorul instrucțiunii ***angle = 45*sin(millis()/50)*** care și asigură animația în cadrul scenei.

Este utilizată funcția ***orbitControl()*** care asigură controlul și poziționarea camerei din cadrul scenei. Cu ajutorul funcției ***background(50)*** este setată culoarea sură a fundalului. Cu ajutorul funcției noStroke() este setat modul grafic de redare a muchiilor astfel încît acestea să nu fie vizibile. Pentru aplicarea individuală a transformărilor pentru fiecare obiect în parte, acestea se încadrează fiecare în interiorul unui bloc ***push()***, ***pop()***. În interiorul acestei construcții sunt realizate toate transformările necesare pentru fiecare obiect grafic. Pentru încararea obiectelor în scenă toate obiectele sunt scalate cu ajutorul funcției ***scale(0.025)*** care micșorează dimensiunile obiectelor cu coeficientul respectiv. Modelul bazei pompei de apă este translat pe axa ***Z*** în jos cu 4000 de unități pentru a fi centrat în vizorul camerei. Modelul bazei este redat în scenă cu ajutorul apelului funcției ***model(base)***.

Pentru reprezentarea elicei, asupra acesteia sunt realizate o serie de transformări cum ar fi scalarea, rotația modelului deja scalat în jurul axei ***Z*** cu valoarea unghiului păstrată în variabila angle, apoi elicea este translată cu ajutorul funcției ***translate(0, -200, 4650)*** cu 200 de unități înainte pe axa ***Y*** și cu 4650 de unități în sus pe axa ***Z***. Ulterior cu ajutorul funcției ***rotateY(millis()/10)*** elicea este rotită în jurul axei ***Y*** al poziției relative la origine și simulează influiența vîntului asupra elicei.

Pentru eprezentarea corpului pompei dimensiunile acestuia sunt scalate cu același coeficient și apoi translat în sus pe axa ***Z*** cu ajutorul funcției ***translate(0, 0, 4650)***, apoi rotit în jurul axei ***Z*** cu ajutorul funcției ***rotateZ(angle)***.

Toate modelele sunt redate în scenă cu ajutorul funcției ***model()*** apelat fiecare în blocul său ***push()***, ***pop()***.

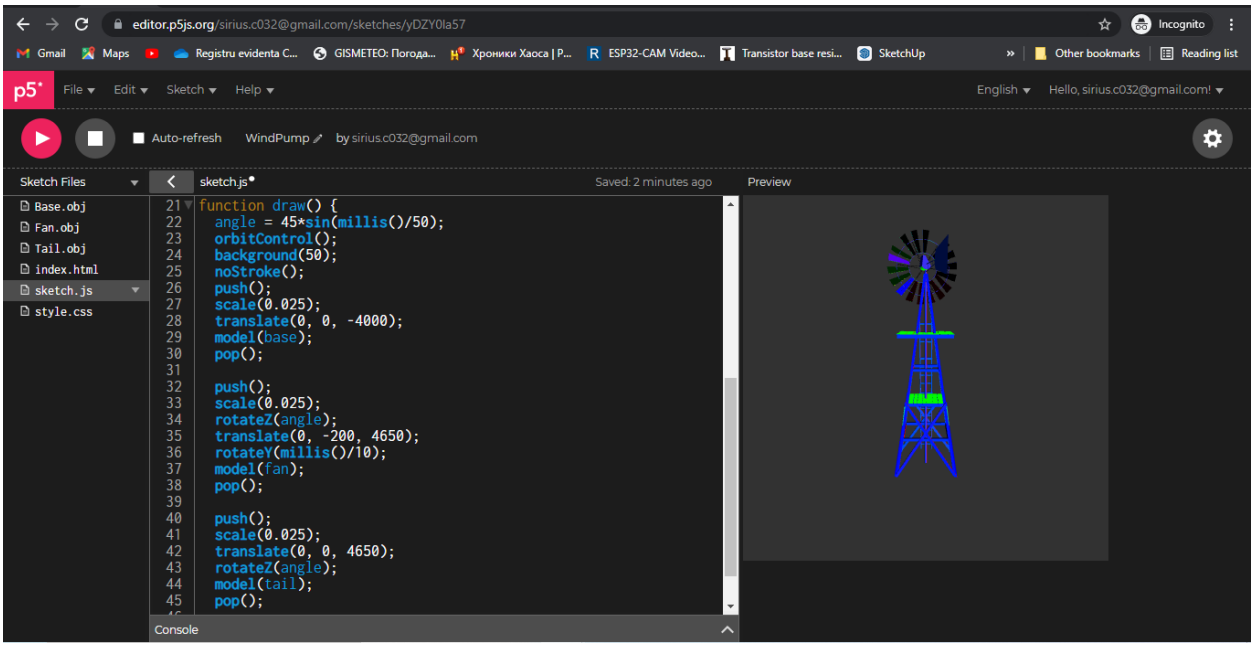

Fig. 7 Modelul pompei de apă în editorul p5.js.

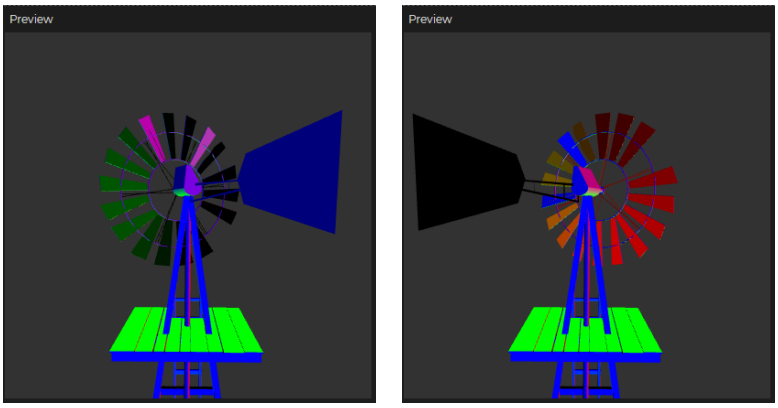

Fig. 8 Rezultatul execuției programului în editorul p5.js.

Lucrarea de laborator nr. 4

Tema: Transformări 3D

Scopul lucrării: Obţinerea cunoştinţelor practice în sinteza scenelor grafice 3D dinamice, utilizînd funcțiile standard de translație, rotație și scalare din biblioteca p5.js.

Sarcina lucrării:

1. Elaborați un program pentru sinteza unei scene 3D dinamice utilizîng funcțiile standard de translație, rotație și scalare din biblioteca p5.js.

2. Elaborați un program care crează o scenă 3D dinanamică conform variantei indicate în tabelul 3.1. Pentru crearea scenei pot fi utilizate obiecte grafice 3D existente în repozitoriul 3D.