Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Departamentul Ingineria Software și Automatica

Raport

Lucrarea de laborator nr. 5

Grafică pe Calculator

Varianta 3

A efectuat:	Student grupa TI-231 FR	Apareci Aurica
A verificat:	asistent universitar	Ursu Adriana

Chișinău

2024

Cuprins

1. Cadrul teoretic

Tema: Transformari 3D

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

Sarcina (conform variantei): 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.

Laborator nr. 4 Varianta 3

Automobil blindat

2. Rezumat succint la temă

Formatul de fișier OBJ este unul dintre cele mai importante formate de fișiere în aplicațiile de imprimare 3D și grafică 3D. Este formatul preferat pentru imprimarea 3D multicolor și este utilizat pe scară largă ca format de schimb neutru pentru modelele 3D non-animate în aplicațiile grafice.

Un fișier OBJ este un format standard de imagine 3D care poate fi exportat și deschis de diverse programe de editare 3D. Acesta conține un obiect tridimensional, care include coordonatele 3D, hărțile de texturi, fețele poligonale și alte informații despre obiect.

Pe scurt, formatul de fișier OBJ stochează informații despre modelele 3D. Poate codifica geometria suprafeței unui model 3D și poate stoca, de asemenea, informații despre culoare și textură. Acest format nu stochează nicio informație despre scenă cum ar fi poziția luminii sau animații. Un fișier OBJ este de obicei generat de un software CAD (Computer Aided Design) ca produs final al procesului de modelare 3D. Extensia de fișier corespunzătoare formatului de fișier OBJ este pur și simplu „.OBJ".

În forma sa cea mai simplă, formatul de fișier OBJ permite utilizatorului să țeseleze suprafața modelului 3D cu forme geometrice simple precum triunghiuri, patrulatere sau poligoane mai complexe. Vârfurile poligoanelor și normala fiecărui poligon sunt apoi stocate într-un fișier care conține geometria suprafeței modelului.

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().

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, Btexturilor și culorilor. Acestea vor fi redate fără proprietăți de materiale, texturi și culori.

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.

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).

normalMaterial()

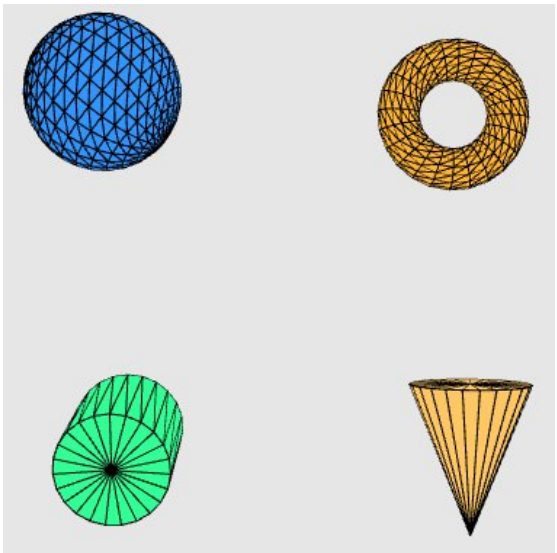
Materialul normal pentru geometrie este un material care nu este afectat de lumină. Nu este reflectant și este un material substituent folosit adesea pentru depanare. Suprafețele orientate spre axa X devin roșii, cele orientate spre axa Y devin verzi și cele orientate spre axa Z devin albastre.

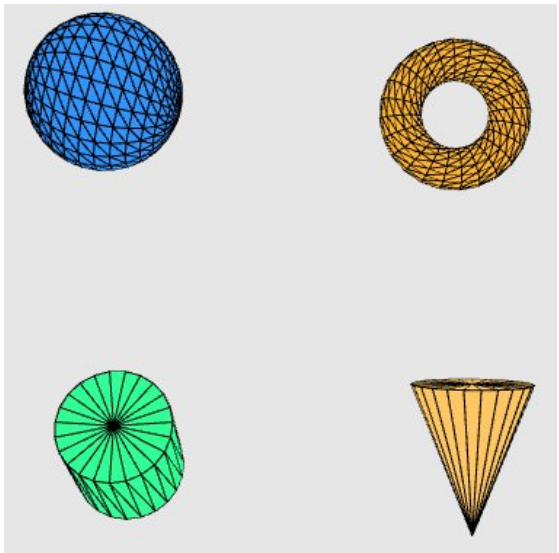
angleMode()

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.

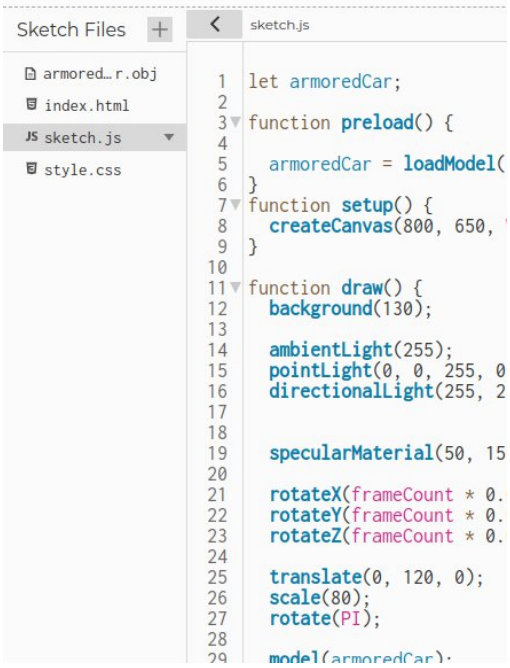
3. Listingul programului

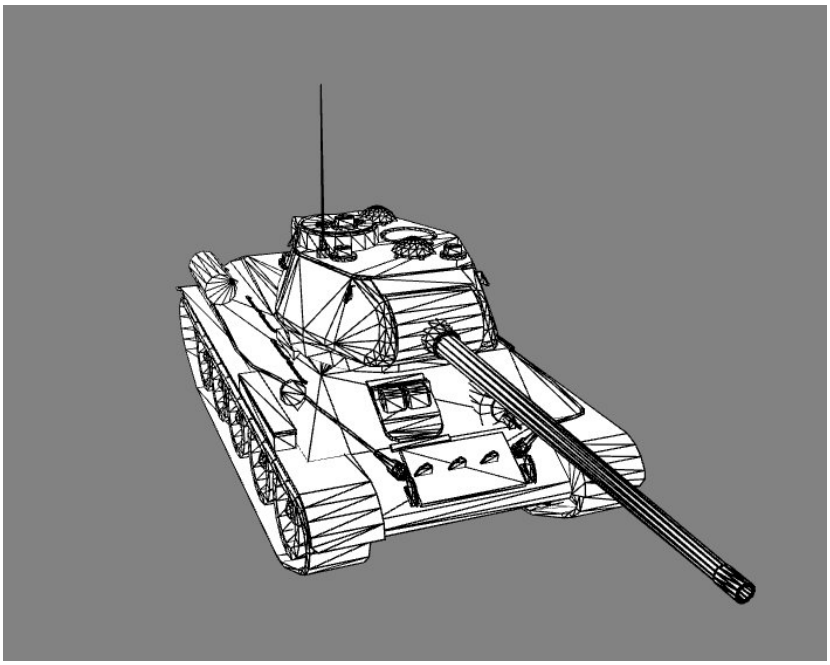
```
function setup() {
  createCanvas(400, 400, WEBGL);
}
function draw() {
  background(180);

  // Sfera
  push();
  translate(-120, -120, 50);
  fill(50, 150, 255);
  rotateZ(frameCount * 0.04);
  sphere(50);
  pop();

  // Tor
  push();
  translate(120, -120, -50);
  fill(255, 180, 50);
  rotateZ(frameCount * 0.01);
  torus(40, 15);
  pop();

  // Cilindru
  push();
  translate(-120, 120, -30);
  fill(50, 255, 150);
  rotateX(frameCount * 0.01);
  cylinder(40, 90);
  pop();

  // Con
  push();
  translate(120, 120, 30);
  fill(255, 200, 100);
  rotateY(frameCount * 0.01);
  cone(40, 100);
  pop();
}
```





```
let armoredCar;

function preload() {

  armoredCar = loadModel('try.obj');
}
function setup() {
  createCanvas(800, 650, WEBGL);
}

function draw() {
  background(130);

  ambientLight(255);
  pointLight(0, 0, 255, 0, -200, 0);
  directionalLight(255, 255, 255, 0, 0, 1);

  specularMaterial(50, 150, 150);

  rotateX(frameCount * 0.01);
  rotateY(frameCount * 0.02);
  rotateZ(frameCount * 0.015);

  translate(0, 120, 0);
  scale(80);
  rotate(PI);

  model(armoredCar);
}
```





4. Concluzii

În cadrul acestei lucrări de laborator, am dobândit cunoștințe practice în crearea și manipularea scenelor grafice 3D dinamice utilizând funcțiile de translație, rotație și scalare oferite de biblioteca p5.js. Prin aplicarea acestor funcții standard, am reușit să elaborez un program care construiește o scenă 3D conform cerințelor din sarcină, respectând specificațiile indicate în tabelul 3.1.

Am explorat modul în care obiectele grafice pot fi plasate și manipulate în spațiul 3D, aplicând transformări variate pentru a simula mișcarea și modificarea proporțiilor acestora. Utilizarea p5.js a permis o abordare eficientă și intuitivă a acestor transformări, iar obiectele grafice din repozitoriu au fost integrate pentru a crea o scenă complexă și dinamică.

Prin realizarea acestei lucrări, am consolidat înțelegerea asupra conceptelor de bază ale graficii computerizate 3D și am dobândit abilități esențiale în utilizarea bibliotecii p5.js pentru dezvoltarea de scene grafice dinamice.