VI. MODELAREA PROCESELOR 3D DINAMICE

Modelarea proceselor 3D dinamice, pe lângă scopuri pur științifice, poate avea și valoare aplicativă.

Biblioteca grafică p5.js poate fi utilizată pentru modelarea fizică a proceselor dinamice. Aceasta permite modelarea sistemelor mecanice (în cadrul legilor mecanicii teoretice). Cu ajutorul p5.js pot fi simulate mișcări de translație și rotație în trei planuri.

În exemplul prezentat în figura 6.1 este simulată ciocnirea a două sfere cu masele *m*1 și *m*2, care până la ciocnire au vitezele deplasării liniare a centrelor notate cu *v*1 și *v*2. După ciocnire, sferele se vor deplasa cu vitezele *v*’1 și *v*’2. Sferele se rostogolesc pe o suprafață rigidă, adică execută o mișcare de rotație specificată de unghiuri: ”omega1” și ”omega2”.

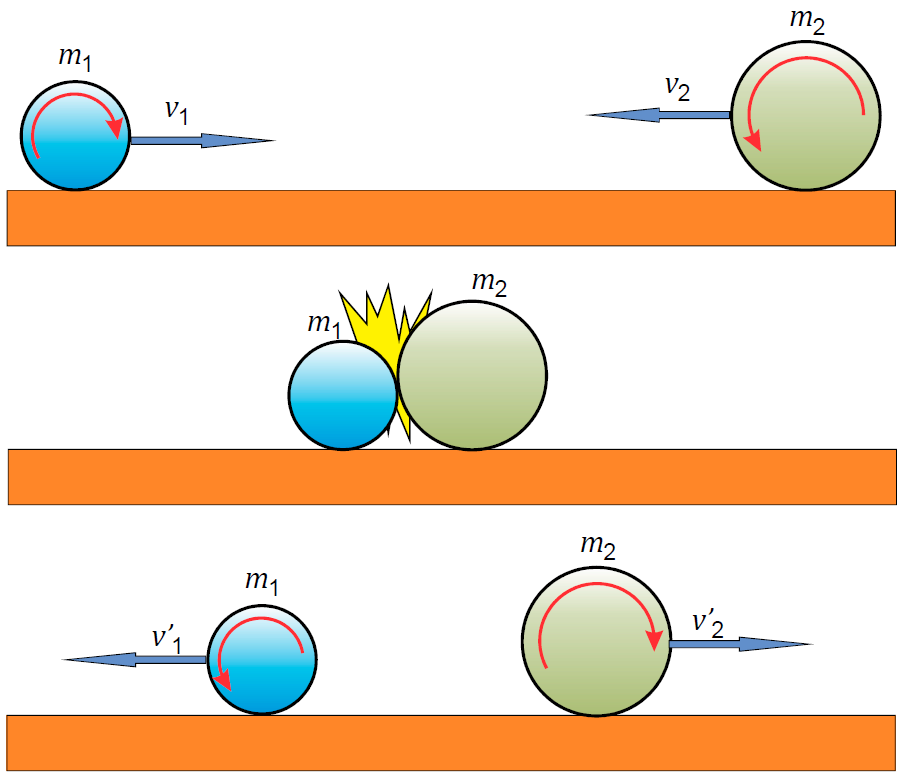


Figura 6.1. Ciocnirea a două sfere

Utilizând legea conservării energiei mecanice din formula (6.1):

, (6.1)

și legea conservării impulsului din formula (6.2):

, (6.2)

pot fi determinate relațiile dintre viteze până și după ciocnirea sferelor (formulele 6.3 și 6.4):

, (6.3)

. (6.4)

Unghiul de rotație poate fi exprimat cu ajutorul formulelor 6.5 și 6.6.

(6.5)

(6.6)

Modelul acestui experiment poate fi descris de următorul cod:

**Listingul programului:**

let x1=-150;

let v1=5;

let m1=50;

let omega1=v1/m1;

let a1=0;

let x2=150;

let v2=-15;

let m2=20;

let omega2=v2/m2;

let a2=0;

function setup()

{ createCanvas(500, 500, WEBGL);

strokeWeight(0.2);

frameRate(24); }

function draw()

{ background(200);

orbitControl();

pointLight(255, 255, 255, 0, 0, 1000);

push();

box(500,100,100);

pop();

push();

translate(x1, -100/2-m1, 0);

rotateZ(a1);

sphere(m1);

pop();

push();

translate(x2, -100/2-m2, 0);

rotateZ(a2);

sphere(m2);

pop();

if(x1+m1>x2-m2)

{ v1=2\*(m1\*v1+m2\*v2)/(m1+m1)-v1;

v2=2\*(m2\*v2+m1\*v1)/(m1+m1)-v2;

x1=x1+v1;

x2=x2+v2;

omega1=v1/m1;

omega2=v2/m2; }

x1=x1+v1;

x2=x2+v2;

a1=a1+omega1;

a2=a2+omega2; }

**Descrierea programului:**

**let** – creează și denumește o variabilă nouă. O variabilă este un container pentru a memora a valoare.

Variabilele care sunt declarate cu **let** vor avea domeniul de aplicare bloc. Aceasta înseamnă că variabila există numai în blocul în care este creată.

În programul de mai sus, în calitate de variabile considerăm:

– x1 și x2 sunt pozițiile inițiale ale sferelor;

– v1 și v2 sunt vitezele liniare inițiale ale centrelor sferelor;

– masa bilelor m1 și m2 care exprimă și razele sferelor;

– omega1=v1/m1 și omega2=v2/m2 sunt unghiurile de rotație a sferelor;

– a1 = 0 și a2 = 0 sunt unghiurile inițiale de rotație a sferelor.

Corpul pe care se rostogolesc sferele este construit cu ajutorul funcției **box(500,100,100),** fiind un obiect static.

Sferele sunt desenate cu ajutorul funcției **sphere()**, asupra lor sunt aplicate mișcarea de translație cu ajutorul funcției **translate()** șide rotație cu ajutorul funcției **rotateZ().**

Modelele corpurilor sunt apelate fiecare în blocul său, fiind delimitate cu **push()**, **pop()**.

Ultima parte a programului include formulele de calcul pentru vitezele sferelor, pozițiilor acestora și al unghiului de rotație.

Rezultatul execuției programului este dat în figura 6.2.

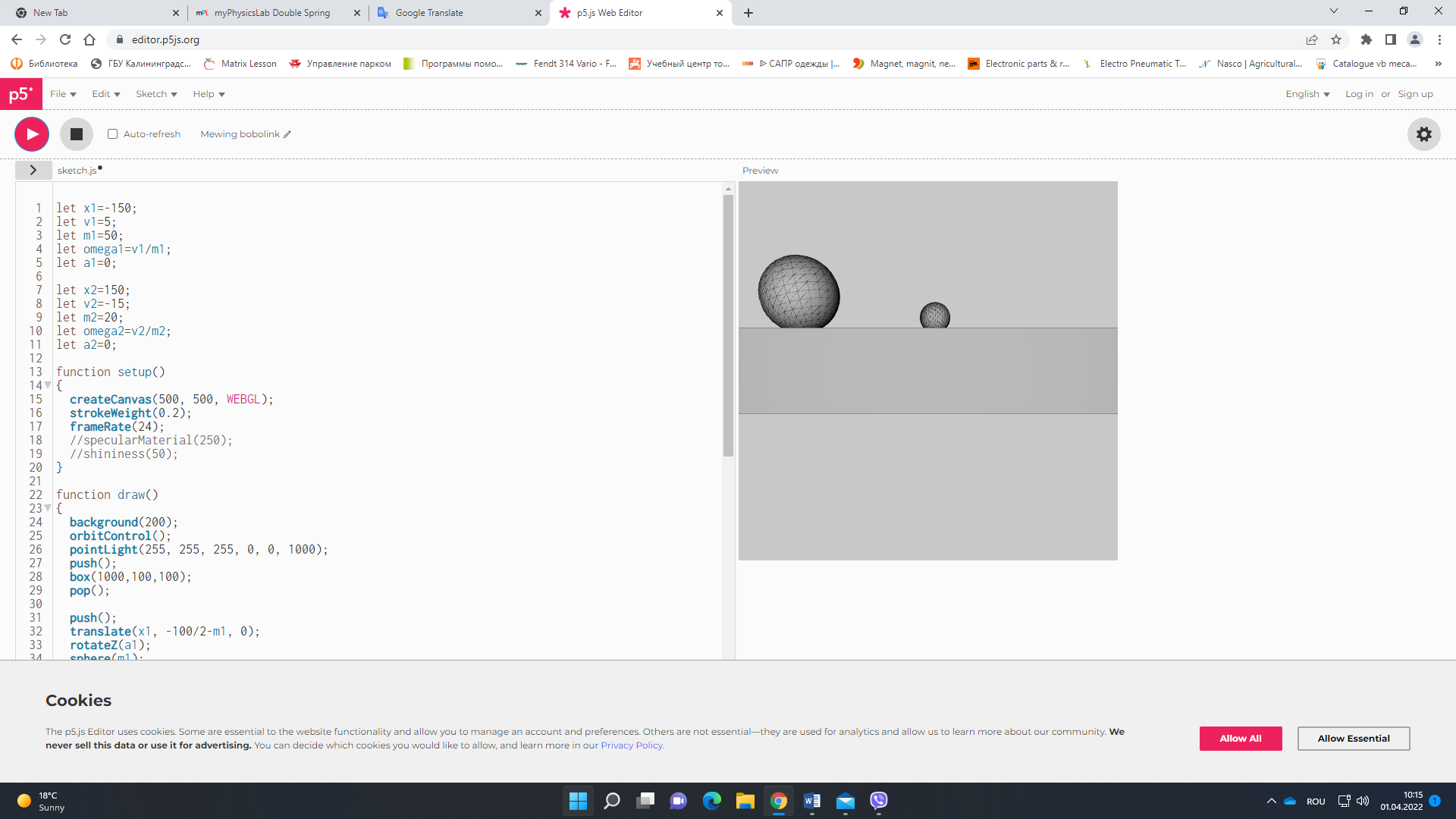


Figura 6.2. Modelul simulării ciocnirii a 2 sfere în p5.js

Pentru realizarea modelului se utilizează primitivele 3D din biblioteca p5.js.

### 6.1. Crearea primitivelor grafice 3D simple

Pentru crearea corpurilor în p5.js sunt utilizate primitivele grafice 3D simple cum ar fi:

* [**plane()**](https://p5js.org/reference/#/p5/plane);
* [**box()**](https://p5js.org/reference/#/p5/box);
* [**sphere()**](https://p5js.org/reference/#/p5/sphere);
* [**cylinder()**](https://p5js.org/reference/#/p5/cylinder);
* [**cone()**](https://p5js.org/reference/#/p5/cone);
* [**ellipsoid()**](https://p5js.org/reference/#/p5/ellipsoid);
* [**torus()**](https://p5js.org/reference/#/p5/torus).

**Funcția plane():** desenează un plan cu o lățime și o înălțime dată

**Sintaxa:**

**plane([width],[height], [detailX], [detailY]);**

unde: [width]: lățimea (opțional);

[height]: înălțimea (opțional);

detailX: numărul de triunghiuri pe axa x (opțional);

detailY: numărul de triunghiuri pe axa y (opțional).

**Funcția box():** desenează un paralelipiped cu lățimea, înălțimea și adâncimea specificată.

**Sintaxa:**

**box([width],[Height],[depth],[detailX], [detailY]);**

unde: [width]: lățimea figurii (opțional);

[height]: înălțimea figurii (opțional);

[depth]: adâncimea figurii (opțional);

detailX: numărul de triunghiuri pe axa x (opțional);

detailY: numărul de triunghiuri pe axa y (opțional).

**Funcția sphere():** Desenează o sferă cu raza specificată. DetailX și detailY determină numărul de segmente în dimensiunea x și dimensiunea y a sferei. Mai multe segmente fac sfera mai netedă. Valoarea maximă recomandată este 24. Utilizarea unei valori mai mari de 24 poate face lucrul browserului mai lent.

**Sintaxa:**

**sphere([radius], [detailX], [detailY]);**

unde: radius: raza cercului (opțional);

detailX: numărul de segmente pe axa x (opțional);

detailY: numărul de segmente pe axa y (opțional).

**Funcția cylinder():** desenează un cilindru cu raza și înălțimea specificată. DetailX și detailY determină numărul de segmente pe axa x și axa y a cilindrului. Mai multe segmente fac ca cilindrul să pară mai neted. Valoarea maximă recomandată pentru detailX este 24. Utilizarea unei valori mai mari de 24 poate încetini browserul.

**Sintaxa:**

**cylinder([radius], [height], [detailX], [detailY], [bottomCap], [topCap]);**

unde: radius: raza bazei cilindrului (opțional);

height: înălțimea cilindrului (opțional);

detailX: numărul de segmente pe axa x, implicit 24 (opțional);

detailY: numărul de segmente pe axa y, implicit 1 (opțional);

bottomCap Boolean: să deseneze sau nu partea de jos a cilindrului (opțional);

topCap Boolean: să deseneze sau nu partea de sus a cilindrului (opțional).

**Funcția cone():** desenează un con cu rază și înălțimea dată. DetailX și detailY determină numărul de segmente pe axa x și axa y a conului. Mai multe segmente fac conul să pară mai neted. Valoarea maximă recomandată pentru detailX este 24. Utilizarea unei valori mai mari de 24 poate încetini browserul.

**Sintaxa:**

**cone([radius], [height], [detailX], [detailY], [cap]);**

unde: radius: raza bazei conului (opțional);

height: înnălțimea conului (opțional);

detailX: numărul de segmente pe axa x, implicit 24 (opțional);

detailY: numărul de segmente pe axa y, implicit 1 (opțional);

Cap Boolean: să deseneze sau nu partea baza cunului (opțional).

**Funcția ellipsoid():** desenează un elipsoid. DetailX și detailY determină numărul segmente pe axa x și axa y a figurii. Mai multe segmente fac elipsoidul mai neted. Evitați numărul de detaliere mai mare de 150, deoarece acesta poate bloca browserul.

**Sintaxa:**

**ellipsoid([radiusx], [radiusy], [radiusz], [detailX], [detailY]);**

unde: radiusx: x - raza elipsoidului (opțional);

radiusy: y - raza elipsoidului (opțional);

radiusz: z - raza elipsoidului (opțional);

detailX: numărul de segmente pe axa x, implicit este 24. Evitați numărul segmentelor mai mare de 150 poate bloca browserul. (Opțional);

detailY: numărul de segmente pe axa y, implicit este 16. Mai mare de 150 poate bloca browserul. (Opțional).

**Funcția torus():** desenează un inel. DetalX și detalY determină numărul de segmente pe axa x și axa y a torului. Valorile maxime implicite pentru detailX și detailY sunt 24 și, respectiv, 16. Setarea lor la valori relativ mici, cum ar fi 4 și 6, vă permite să creați noi forme, altele decât un tor.

**Sintaxa:**

**torus([radius], [tubeRadius], [detailX], [detailY]);**

**Parametrii:**

radius: raza externă a figurii (Opțional);

tubeRadius: raza internă a figurii (Opțional);

detailX: numărul de segmente pe axa x, implicit este 24 (opțional);

detailY: numărul de segmente pe axa y, implicit este 16 (opțional).

### Lucrarea de laborator nr. 6

### Tema: MODELAREA PROCESELOR ÎN 3D

**Obiectivele lucrării:**

**1. Familiarizarea cu** **procesele fizice** utilizate pentru crearea scenelor grafice 3D.

**2. Utilizarea primitivelor grafice** – obținerea cunoștințelor practice privind utilizarea primitivelor grafice 3D pentru crearea scenei.

**3. Crearea scenelor grafice 3D dinamice** – dezvoltarea abilităților de creare și compunere a scenelor grafice 3D dinamice prin combinarea eficientă a transformărilor geometrice.

**4. Înțelegerea și aplicarea conceptelor de procese fizice** –utilizarea noțiunilor de viteză, putere, gravitație și altele utilizate pentru dirijarea dinamicii vizuale în grafica 3D.

**Numărul de ore necesare pentru realizare – 4 ore academice.**

**Scopul lucrării:** obținerea cunoștințelor practice privind crearea scenelor 3D în modelarea proceselor 3D dinamice prin utilizarea funcțiilor standard de transformare geometrică, cum ar fi translația, scalarea și rotația din biblioteca p5.js.

**Sarcina lucrării:** elaborați un program care creează o scenă 3D în care să modelați un proces fizic, utilizând funcțiile standard de translație și rotație din biblioteca p5.js, conform variantei indicate în tabelul 6.1. Pentru crearea scenei pot fi utilizate obiecte grafice 3D existente în repozitorul 3D.

Simularea și descrierea procesului fizic bidimensional din sarcină pot fi consultate pe pagina <https://www.myphysicslab.com/>.

**Tabelul 6.1. Variante pentru realizarea lucrării de laborator**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Nr.*** | ***Model*** | ***Reprezentare*** | ***Nr.*** | ***Model*** | ***Reprezentare*** |
| 1 | Arc de extensie singular |  | 6 | Pendul rigid cu mișcare pe cerc |  |
| 2 | Arc de extensie dublu |  | 7 | Pendul dublu rigid |  |
| 3 | Pendul haotic |  | 8 | Coliziunea blocurilor rigide |  |
| 4 | Mișcarea sferei pe traiectorie de tip cocoașă |  | 9 | Pendulul lui Newton |  |
| 5 | Mișcarea sferei pe traiectorii multiple |  | 10 | Atracția corpurilor |  |

**Criterii de evaluare**

**1. Corectitudinea codului** (20%) – verificarea corectitudinii codului, fără erori de sintaxă și funcționare.

**2. Utilizarea primitivelor grafice 3D** (20%) – evaluarea utilizării corecte și varietatea primitivelor grafice în conformitate cu cerințele lucrării.

**3. Respectarea instrucțiunilor și cerințelor** (10%) – verificarea corectitudinii cerințelor sarcinii cum ar fi anumite forme sau dimensiuni specifice ale primitivelor grafice și dinamica lor.

**4. Optimizarea codului** (10%) – evaluarea eficienței codului în utilizarea resurselor și evitarea codului redundant.

**5. Interactivitatea** (10%) – răspunsuri la mouse sau tastatură, se va evalua dacă aceste funcții sunt implementate corect.

**6. Estetica vizuală** (10%) – analiza atractivității vizuale a compoziției, cum ar fi armonia culorilor, proporțiile și echilibrul grafic.

**7. Respectarea termenului de predare** (10%) – evaluarea punctajului în funcție de punctualitate, dacă lucrarea a fost predată în termenul stabilit.

**8. Evaluarea cunoștințelor** (10%) – explicații privind procesul de realizare a lucrării, ceea ce poate include descrierea funcțiilor principale și a logicii utilizate.

**Întrebări de verificare a cunoștințelor**

1. Enumerați primitivele grafice 3D simple.

2. Enumerați funcțiile utilizate pentru transformările grafice.

3. Cum poate fi realizată modificarea atributelor de afișare a primitivelor grafice 3D?

4. Numiți formate de rotație și funcțiile corespunzătoare.

5. Ce primitive grafice pot fi utilizate pentru a crea forme de bază într-o scenă grafică 2D?

6. Cum poate fi realizată combinarea transformărilor grafice?

7. Ce rol are funcția push() și pop() în gestionarea transformărilor geometrice?

8. Cum se activează modul 3D în p5.js?