МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГИЙ УПРАВЛІННЯ

Лабораторна робота № 1

З дисципліни «Моделювання складних систем»

Виконав:

студент групи КН 36-а

Рубан Ю. Д.

Перевірила:

доц. каф. ПІІТУ

Єршова С. І.

ХАРКІВ 2019

**Тема:** Знайомство з середовищем Anylogic

**Ціль:** Вивчити процедури редагування готових моделей. Вивчити технологію обробки подій.

**Завдання:**

1 Провести експерименти з розглянутою моделлю м’яча, змінюючи:

– колір м’яча;

– кількість м’ячів;

– колір та кількість м’ячів;

– координати м’ячів.

2 Провести експерименти для вивчення впливу швидкості м’яча на втрату його енергії під час моделювання відскоку від:

– підлоги;

– стелі;

– стін.

3 Виконати індивідуальне завдання:

* У моделі 2 види куль - блакитні і пурпурні. При зіткненні куль різного кольору, зникає куля з меншою швидкістю.
* Час життя куль 15 сек. Після закінчення терміну життя кулі зникають. Термін життя куль обнуляється при зіткненні з кулею іншого кольору.
* При зіткненні куль одного кольору створюється новий шар з ймовірністю 0.5. Ця куля виникає того ж кольору, з випадковими координатами і напрямком руху. Швидкість нового кулі дорівнює сумі швидкостей зіткнулися куль.
* У презентації моделі показувати час життя кожного кулі.

**Хід виконання роботи:**

Було проведено експерименти для вивчення впливу швидкості м’яча на втрату його енергії при відскоку від стіни, стелі та полу також було враховано різні швидкості. На рисунку 1 показана зміна кінетичної та потенціальної енергії для швидкості 10.

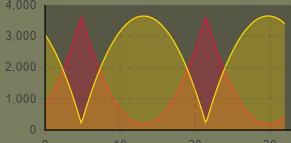


Рисунок 1 – Зміна кінетичної та потенціальної енергії

Жовтим показано потенціальну енергії, а червоним – кінетичну енергію. По симуляції моделі видно, що потенціальна енергія зменшується при зменшенні висоти, а кінетична збільшується і навпаки. На рисунку 2 – 3 показано як впливає зміна швидкості на потенціальну та кінетичну енергію.

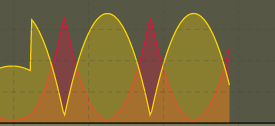


Рисунок 2 – Вплив збільшення швидкості на кінетичну та потенціальну енергію



Рисунок 3 – Вплив зменшення швидкості на кінетичну та потенціальну енергію

Отже зміна швидкості пропорційна зміні кінетичної та потенціальної енергії.

На рисунку 4 показані результати при яких куля зіштовхується зі стелею

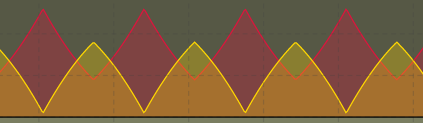


Рисунок 4 – Зміна енергій при зіткненні зі стелею

На даному рисунку видно, що при зіткненні зі стелею потенціальна енергія починає зменшуватися більш різко, та кінетична збільшується більш різко і завдяки цьому шар швидше падає на землю. Так само при зіткненні з полом. Кінетична енергія максимальна, а потенційна мінімальна.

**Індивідуальне завдання:**

Було виконано засобами мови javascript та бібліотеки p5.js. На рисунку 5 показано інтерфейс симуляції.

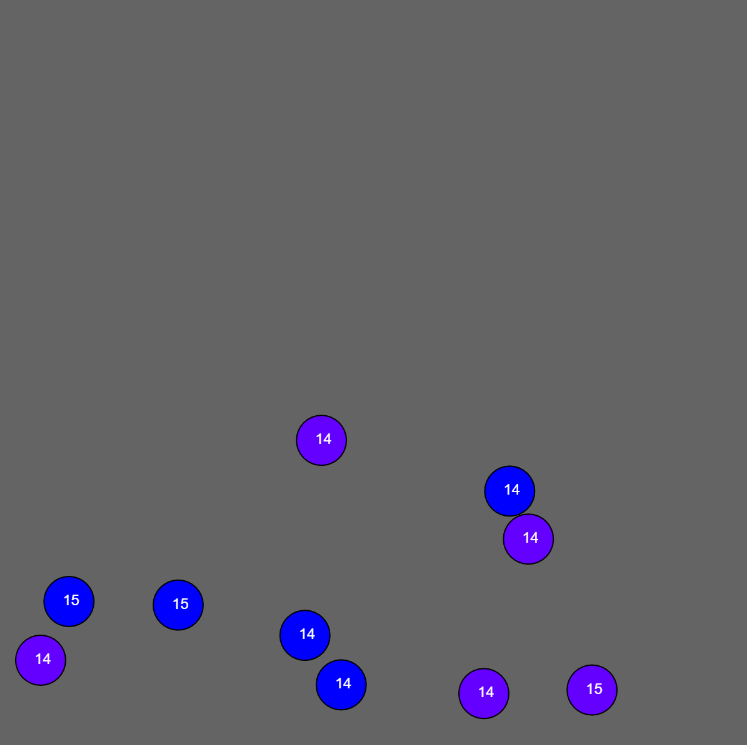


Рисунок 5 – Інтерфейс симуляції

У лістингу 1 показаний код реалізації

Лістинг 1 – Реалізація даної симуляції

const g = 0.2;

class Ball {

  constructor(pos, vel, d, color, life) {

    this.pos = pos;

    this.vel = vel;

    this.color = color;

    this.d = d;

    this.mass = 1;

    this.k = 0.01;

    this.a = createVector(0, g)

    this.life = life;

  }

  draw() {

    fill(this.color)

    ellipse(this.pos.x, this.pos.y, this.d, this.d);

    fill(255)

    text(Math.round(this.life / 1000), this.pos.x - 5, this.pos.y + 3)

  }

  update() {

    this.vel.add(this.a)

    this.vel.add(createVector(-this.vel.x \* this.k, -this.vel.y \* this.k))

    this.pos.add(this.vel)

  }

}

const w = 600

const h = 600

const d = 40

let balls = []

let overlapers = []

let BLUE;

let PURPLE;

const lifeTimeMs = 15000;

let currTime, elapsedTime, prevTime = 0

const BALL\_COUNT = 10

const MAX\_BALLS = 60

function dynamicColision(ballI, ballJ) {

  let xDiff = (ballI.pos.x - ballJ.pos.x) \* (ballI.pos.x - ballJ.pos.x);

  let yDiff = (ballI.pos.y - ballJ.pos.y) \* (ballI.pos.y - ballJ.pos.y);

  let distance = sqrt(xDiff + yDiff);

  const nx = (ballJ.pos.x - ballI.pos.x) / distance;

  const ny = (ballJ.pos.y - ballI.pos.y) / distance;

  const tgX = -ny;

  const tgY = nx;

  const dotProdTg1 = ballI.vel.x \* tgX + ballI.vel.y \* tgY;

  const dotProdTg2 = ballJ.vel.x \* tgX + ballJ.vel.y \* tgY;

  const dotProdNorm1 = ballI.vel.x \* nx + ballI.vel.y \* ny;

  const dotProdNorm2 = ballJ.vel.x \* nx + ballJ.vel.y \* ny;

  const massSum = (ballJ.mass + ballI.mass);

  const m1 = (dotProdNorm1 \* (ballI.mass - ballJ.mass) + 2\*ballJ.mass\*dotProdNorm2) / massSum

  const m2 = (dotProdNorm2 \* (ballJ.mass - ballI.mass) + 2\*ballI.mass\*dotProdNorm1) / massSum

  ballI.vel.x = tgX \* dotProdTg1 + nx \* m1;

  ballI.vel.y = tgY \* dotProdTg1 + ny \* m1;

  ballJ.vel.x = tgX \* dotProdTg2 + nx \* m2;

  ballJ.vel.y = tgY \* dotProdTg2 + ny \* m2;

  onLabCollision(ballI, ballJ);

}

function onCollision(ballI, ballJ) {

  let xDiff = (ballI.pos.x - ballJ.pos.x) \* (ballI.pos.x - ballJ.pos.x);

  let yDiff = (ballI.pos.y - ballJ.pos.y) \* (ballI.pos.y - ballJ.pos.y);

  let distance = sqrt(xDiff + yDiff);

  const overlap = 0.5\*(distance - ballI.d/2 - ballJ.d/2)

  ballI.pos.x -= overlap \* (ballI.pos.x - ballJ.pos.x) / distance;

  ballI.pos.y -= overlap \* (ballI.pos.y - ballJ.pos.y) / distance;

  ballJ.pos.x += overlap \* (ballI.pos.x - ballJ.pos.x) / distance;

  ballJ.pos.y += overlap \* (ballI.pos.y - ballJ.pos.y) / distance;

}

function onLabCollision(ballI, ballJ) {

  if(ballJ.color != ballI.color) {

    const lowVel = ballI.vel.mag() < ballJ.vel.mag() ? ballI : ballJ;

    const highVel = lowVel == ballJ ? ballI : ballJ;

    const index = balls.indexOf(lowVel);

    if(index == -1)

      return;

    balls.splice(index, 1);

    highVel.life = lifeTimeMs;

  } else {

    if (random() < 0.5) {

      createBall(ballJ.color, ballJ.vel.copy().add(ballI.vel))

    }

  }

}

function setup() {

  createCanvas(w, h);

  frameRate(60)

  BLUE = color(0, 0, 255)

  PURPLE = color(100, 0, 255)

  for(let i = 0; i < BALL\_COUNT; i++) {

    createBall();

  }

}

function createBall(color, vel) {

  let pos;

  if(balls.length > MAX\_BALLS) return;

  if(balls.length > MAX\_BALLS / 2) {

    const cols = w / d;

    const rows = h / d;

    let flag = false

    for (let j = 1; j < rows; j++) {

      for (let i = 1; i < cols; i++) {

      pos = createVector(i \* d, j \* d)

      flag = balls.every(ball => {

          const xDiff = (ball.pos.x - pos.x) \* (ball.pos.x - pos.x);

          const yDiff = (ball.pos.y - pos.y) \* (ball.pos.y - pos.y);

          const distance = sqrt(xDiff + yDiff);

          return distance > d;

        });

      if(flag)

      break;

      };

      if(flag)

        break;

    }

    if(!flag)

      return;

  } else {

    pos = createVector(random(d, w - d), random(d, h - d));

  }

  if(!vel){

    vel = p5.Vector.random2D();

    vel.mult(5)

  }

  if(!color) {

    color = random() < 0.5 ? BLUE : PURPLE;

  }

  const ball = new Ball(pos, vel, d, color, lifeTimeMs)

  balls.push(ball);

}

function draw() {

  currTime = millis();

  elapsedTime = (currTime - prevTime)

  prevTime = currTime

  background(100)

  balls.forEach(ball => {

    ball.draw()

    ball.update()

    ball.life -= elapsedTime

  });

  balls.forEach(ball => {

    if(ball.pos.x <= d/2 || ball.pos.x >= w - d/2) {

      ball.vel.x \*= -1

    }

    if(ball.pos.y <= d/2 || ball.pos.y >= h - d/2) {

      ball.vel.y \*=  -( 1 - ball.k)

    }

    if(ball.pos.x < d/2){

      ball.pos.x = d/2

    }

    if(ball.pos.y < d/2){

      ball.pos.y = d/2

    }

    if(ball.pos.x > w - d/2){

      ball.pos.x = w - d/2

    }

    if(ball.pos.y >  h - d/2){

      ball.pos.y =  h - d/2

    }

  });

  for (let i = 0; i < balls.length; i++) {

    const ballI = balls[i];

    for (let j = 0; j < balls.length; j++) {

      const ballJ = balls[j];

      if(i == j) continue;

      const xDiff = (ballI.pos.x - ballJ.pos.x) \* (ballI.pos.x - ballJ.pos.x);

      const yDiff = (ballI.pos.y - ballJ.pos.y) \* (ballI.pos.y - ballJ.pos.y);

      const rSum = (ballI.d/2 + ballJ.d/2) \* (ballI.d/2 + ballJ.d/2);

      if(xDiff + yDiff <= rSum) {

        onCollision(ballI, ballJ);

        overlapers.push({ballI, ballJ});

      }

    }

  }

  overlapers.forEach(op => {

    dynamicColision(op.ballI, op.ballJ);

  })

  overlapers = []

  balls = balls.filter(ball => ball.life > 0)

}

**Висновки:**

В даній лабораторній роботі було вивчено процедуру редагування готових моделей та технологію обробки подій, а також було виконано індивідуальне завдання для ознайомлення з моделюванням руху округлого тіла.