University of London

BSc Computer Science

CM2030 Graphics Programming

Midterm Assignment

Code PDF

By Hristo Stantchev

Guide:

1. Project Title:
    a. Code Extract
    Label i.e. “Sketch.js”

1. Solar System

```
var speed;


function setup() {
    createCanvas(900, 700);
}

function draw() {
    background(0);
    speed = frameCount;

    push();
    translate(width / 2, height / 2);
    //start of my code
    //rotate hand axis
    rotate(radians(speed / 3));
    //end of my code
    celestialObj(color(255, 150, 0), 200); // SUN
    //start of my code
    //rotate child movement axis
    rotate(radians(speed));
    push();
    translate(0, 300);
    //rotate hand axis
    rotate(radians(speed));
    celestialObj(color(0, 0, 255), 80); // EARTH
```

```
    //rotate child movement axis
    var moonAngle = -4;
    rotate(radians(speed * moonAngle));
    push();
    translate(0, 100);
    celestialObj(color(255), 30);
    rotate(radians(speed * 4));
    push();
    translate(0, 45);
    rotate(radians(speed * 4));
    celestialObj(color(255), 20);
    pop();
    pop();
    pop();
    //end of my code
    pop();
}

function celestialObj(c, size) {
    strokeWeight(5);
    fill(c);
    stroke(0);
    ellipse(0, 0, size, size);
    line(0, 0, size / 2, 0);
}
```

a. Sketch.js

2. Asteroid Game Clone
    a. asteroidSystem.js

```
class BulletSystem {


  constructor(){
    this.bullets = [];
    this.velocity = new createVector(0, -5);
    this.diam = 10;
  }

  run(){
      this.move();
      this.draw();
      this.edges();
  }

  fire(x, y){
```

```
    this.bullets.push(createVector(x,y));
  }

  //draws all bullets
  draw(){
    fill(255);
    for (var i=0; i<this.bullets.length; i++){
      ellipse(this.bullets[i].x, this.bullets[i].y, this.diam, this.diam);
    }
  }

  //updates the location of all bullets
  move(){
    for (var i=0; i<this.bullets.length; i++){
      this.bullets[i].y += this.velocity.y;
    }
  }

  //check if bullets leave the screen and remove them from the array
  edges(){
      // YOUR CODE HERE (3 lines approx)
      //start of my code
      for(var i = 0; i < this.bullets.length; i++){
        var bullet = this.bullets[i];
        if (bullet.x > width || bullet.x < 0 || bullet.y > height || bullet.y <
0) {
          this.bullets.splice(i,1);
          break;
        }
      }
      //end of my code
  }
}
```

b. bulletSystem.js

```
//mods

//1 - randomized asteroid colours
//2 - score system
//3 - difficulty: spawn rate of asteroids is raised for every 5 points
var spaceship;
var asteroids;
var atmosphereLoc;
var atmosphereSize;
var earthLoc;
```

```
var earthSize;
var starLocs = [];
//start of my code
var score = 0;
var scoreTracker = 0;
var scoreTxt;
var lose = false;
//end of my code
////////////////////////////////////////////////
function setup() {
  createCanvas(1200, 800);
  spaceship = new Spaceship();
  asteroids = new AsteroidSystem();

  //location and size of earth and its atmosphere
  atmosphereLoc = new createVector(width / 2, height * 2.9);
  atmosphereSize = new createVector(width * 3, width * 3);
  earthLoc = new createVector(width / 2, height * 3.1);
  earthSize = new createVector(width * 3, width * 3);
}

////////////////////////////////////////////////
function draw() {
  background(0);
  sky();

  spaceship.run();
  asteroids.run();

  drawEarth();
  checkCollisions(spaceship, asteroids); // function that checks collision
between various elements
  //start of my code
  //draw score if lose condition not met;
  if (!lose) {
    fill(255);
    textSize(15);
    scoreTxt = "SCORE: " + score;
    text(scoreTxt, width / 2, 20);
  }
  //raise difficulty for every 5 score points
  if (scoreTracker >= 5) {
    asteroids.spawnRate += 0.005;
    scoreTracker = 0;
  }
```

```
  //end of my code
}

////////////////////////////////////////////////
//draws earth and atmosphere
function drawEarth() {
  noStroke();
  //draw atmosphere
  fill(0, 0, 255, 50);
  ellipse(atmosphereLoc.x, atmosphereLoc.y, atmosphereSize.x, atmosphereSize.y);
  //draw earth
  fill(100, 255);
  ellipse(earthLoc.x, earthLoc.y, earthSize.x, earthSize.y);
}

////////////////////////////////////////////////
//checks collisions between all types of bodies
function checkCollisions(spaceship, asteroids) {
  //start of my code
  //spaceship-2-asteroid collisions
  for (var i = 0; i < asteroids.locations.length; i++) {
    if (isInside(spaceship.location, spaceship.size, asteroids.locations[i],
asteroids.diams[i])) {
      gameOver();
    }
  }

  //asteroid-2-earth collisions
  for (var i = 0; i < asteroids.locations.length; i++) {
    if (isInside(earthLoc, earthSize.x, asteroids.locations[i],
asteroids.diams[i])) {
      gameOver();
    }
  }
  //spaceship-2-earth
  if (isInside(spaceship.location, spaceship.size, earthLoc, earthSize.x)) {
    gameOver();
  }
  //spaceship-2-atmosphere
  if (isInside(spaceship.location, spaceship.size, atmosphereLoc,
atmosphereSize.x)) {
    spaceship.setNearEarth();
  }
  //bullet collisions
  for (var i = 0; i < asteroids.locations.length; i++) {
```

```
    for (var j = 0; j < spaceship.bulletSys.bullets.length; j++) {
      if (isInside(asteroids.locations[i], asteroids.diams[i],
spaceship.bulletSys.bullets[j], spaceship.bulletSys.diam)) {
        asteroids.destroy(i);
        score++;
        scoreTracker++;
        break;
      }
    }
  }
  //end of my code
}

//////////////////////////////////////////////////
//helper function checking if there's collision between object A and object B
function isInside(locA, sizeA, locB, sizeB) {
  //start of my code
  if (dist(locA.x, locA.y, locB.x, locB.y) < (sizeA + sizeB) / 2) {
    return true;
  }
  return false;
  //end of my code
}

//////////////////////////////////////////////////
function keyPressed() {
  if (keyIsPressed && keyCode === 32) { // if spacebar is pressed, fire!
    spaceship.fire();
  }
}

//////////////////////////////////////////////////
// function that ends the game by stopping the loops and displaying "Game Over"
function gameOver() {
  //set lose condition to true
  lose = true;
  fill(255);
  textSize(80);
  textAlign(CENTER);
  //start of my code
  text(scoreTxt, width / 2, (height / 2) - 160);
  //end of my code
  text("GAME OVER", width / 2, height / 2);
  noLoop();
}
```

```
////////////////////////////////////////////////
// function that creates a star lit sky
function sky() {
  push();
  while (starLocs.length < 300) {
    starLocs.push(new createVector(random(width), random(height)));
  }
  fill(255);
  for (var i = 0; i < starLocs.length; i++) {
    rect(starLocs[i].x, starLocs[i].y, 2, 2);
  }

  if (random(1) < 0.3) starLocs.splice(int(random(starLocs.length)), 1);
  pop();
}
```

c. sketch.js

```
class Spaceship {


  constructor() {
    this.velocity = new createVector(0, 0);
    this.location = new createVector(width / 2, height / 2);
    this.acceleration = new createVector(0, 0);
    this.maxVelocity = 5;
    this.bulletSys = new BulletSystem();
    this.size = 50;
  }

  run() {
    this.bulletSys.run();
    this.draw();
    this.move();
    this.edges();
    this.interaction();
  }

  draw() {
    fill(125);
    triangle(this.location.x - this.size / 2, this.location.y + this.size / 2,
      this.location.x + this.size / 2, this.location.y + this.size / 2,
      this.location.x, this.location.y - this.size / 2);
  }

  move() {
```

```
  //start of my code
  this.velocity.add(this.acceleration);
  this.velocity.limit(this.maxVelocity);
  this.location.add(this.velocity);
  this.acceleration.mult(0);
  this.velocity.normalize();
  //end of my code
}

applyForce(f) {
  this.acceleration.add(f);
}

interaction() {
  var interSpeed = 0.7;
  if (keyIsDown(LEFT_ARROW)) {
    //start of my code
    this.applyForce(createVector(-interSpeed, 0));
    //end of my code
  }
  if (keyIsDown(RIGHT_ARROW)) {
    //start of my code
    this.applyForce(createVector(interSpeed, 0));
  }
  if (keyIsDown(UP_ARROW)) {
    //start of my code
    this.applyForce(createVector(0, -interSpeed));
    //end of my code
  }
  if (keyIsDown(DOWN_ARROW)) {
    //start of my code
    this.applyForce(createVector(0, interSpeed));
    //end of my code
  }
}

fire() {
  this.bulletSys.fire(this.location.x, this.location.y);
}

edges() {
  if (this.location.x < 0) this.location.x = width;
  else if (this.location.x > width) this.location.x = 0;
  else if (this.location.y < 0) this.location.y = height;
  else if (this.location.y > height) this.location.y = 0;
```

```
  }

  setNearEarth() {
    //YOUR CODE HERE (6 lines approx)
    //start of my code
    console.log("IS NEAR EARTH");
    var gravity = createVector(0, 0.05);
    this.applyForce(gravity);
    var friction = this.velocity.copy();
    friction.mult(-1);
    friction.normalize();
    friction.div(30);
    this.applyForce(friction);
    //end of my code
  }
}
```

d. spaceship.js

3. Angry Bird Clone

```
////////////////////////////////////////////////////////////////////

function setupGround() {
  ground = Bodies.rectangle(500, 600, 1000, 40, {
    isStatic: true, angle: 0
  });
  World.add(engine.world, [ground]);
}

////////////////////////////////////////////////////////////////////
function drawGround() {
  push();
  fill(128);
  drawVertices(ground.vertices);
  pop();
}
////////////////////////////////////////////////////////////////////
function setupPropeller() {
  // your code here
  //start of my code
  //init propeller position, size and options
  propeller = Bodies.rectangle(150, 480, 200, 15, { isStatic: true, angle: angle
})
  //add propeller to world
  World.add(engine.world, [propeller]);
  //end of my code
```

```
}
////////////////////////////////////////////////////////////
//updates and draws the propeller
function drawPropeller() {
  push();
  //start of my code
  //set angle and angular velocity
  Body.setAngle(propeller, angle);
  Body.setAngularVelocity(propeller, angleSpeed);

  //update angle by angleSpeed
  angle += angleSpeed;
  //draw object
  drawVertices(propeller.vertices);
  //end of my code
  pop();
}
////////////////////////////////////////////////////////////
function setupBird() {
  var bird = Bodies.circle(mouseX, mouseY, 20, {
    friction: 0,
    restitution: 0.95
  });
  Matter.Body.setMass(bird, bird.mass * 10);
  World.add(engine.world, [bird]);
  birds.push(bird);
}
////////////////////////////////////////////////////////////
function drawBirds() {
  push();
  //start of my code
  for (var i = 0; i < birds.length; i++) {
    drawVertices(birds[i].vertices);
    if (isOffScreen(birds[i])) {
      removeFromWorld(birds[i]);
      birds.splice(i, 1);
      i--;
    }
  }
  //end of my code
  pop();
}
////////////////////////////////////////////////////////////
//creates a tower of boxes
```

```
function setupTower() {
  //start of my code
  //tracker vars
  var boxPosX = 600;
  var boxPosY = 465;
  var boxSize = 80;
  //for loop for width
  for (var x = 0; x < 3; x++) {
    //for loop for height
    for (var y = 0; y < 6; y++) {
      var box = Bodies.rectangle(boxPosX + boxSize * x, boxPosY - boxSize * y,
boxSize, boxSize);
      boxes.push(box);
      colors.push(random(100, 255));
      World.add(engine.world, [box]);
    }
  }
  //end of my code
}
////////////////////////////////////////////////////////////////
//draws tower of boxes
function drawTower() {
  push();
  //start of my code
  //loop thorugh boxes
  for (var i = 0; i < boxes.length; i++) {
    //set fill to random green and draw box
    fill(0, colors[i], 0);
    drawVertices(boxes[i].vertices);
    //check for box in world
    if (isOffScreen(boxes[i])) {
      gameManager.boxes--;
      removeFromWorld(boxes[i]);
      boxes.splice(i, 1);
      colors.splice(i, 1);
      i--;
    }
  }
  //end of my code
  pop();
}
////////////////////////////////////////////////////////////////
function setupSlingshot() {
  //start of my code
  slingshotBird = Bodies.circle(200, 230, 20, {
```

```
    friction: 0,
    restitution: 0.95
  });
  Matter.Body.setMass(slingshotBird, slingshotBird.mass * 10);
  slingshotConstraint = Constraint.create({
    pointA: { x: 200, y: 200 },
    bodyB: slingshotBird,
    stiffness: 0.01,
    damping: 0.0001
  });
  World.add(engine.world, [slingshotBird, slingshotConstraint]);
  //end of my code
}
////////////////////////////////////////////////////////////////
//draws slingshot bird and its constraint
function drawSlingshot() {
  push();
  //start of my code
  fill(200, 120, 0);
  drawVertices(slingshotBird.vertices);
  drawConstraint(slingshotConstraint);
  //end of my code
  pop();
}
////////////////////////////////////////////////////////////////
function setupMouseInteraction() {
  var mouse = Mouse.create(canvas.elt);
  var mouseParams = {
    mouse: mouse,
    constraint: { stiffness: 0.05 }
  }
  mouseConstraint = MouseConstraint.create(engine, mouseParams);
  mouseConstraint.mouse.pixelRatio = pixelDensity();
  World.add(engine.world, mouseConstraint);
}
```

a. physics.js

```
// Example is based on examples from: http://brm.io/matter-js/,
https://github.com/shiffman/p5-matter

// add also Benedict Gross credit

var Engine = Matter.Engine;
var Render = Matter.Render;
var World = Matter.World;
```

```
var Bodies = Matter.Bodies;
var Body = Matter.Body;
var Constraint = Matter.Constraint;
var Mouse = Matter.Mouse;
var MouseConstraint = Matter.MouseConstraint;

//start of my code
var gameManager = {
  score: 0,
  timeLeft: 60,
  gameOver: false,
  victory: false,
  boxes: 18,
  lastSec: 0
};
//end of my code

var engine;
var propeller;
var boxes = [];
var birds = [];
var colors = [];
var ground;
var slingshotBird, slingshotConstraint;
var angle = 0;
var angleSpeed = 0;
var canvas;
////////////////////////////////////////////////////////////
function setup() {
  canvas = createCanvas(1000, 600);

  engine = Engine.create();  // create an engine

  setupGround();

  setupPropeller();

  setupTower();

  setupSlingshot();

  setupMouseInteraction();
  //start of my code
  gameManager.lastSec = second();
  //end of my code
```

```
}
////////////////////////////////////////////////////////////
function draw() {
  background(0);
  //start of my code

  //set time left
  if (gameManager.lastSec != second()) {
    gameManager.timeLeft--;
    gameManager.lastSec = second();
  }
  //set victory condition
  if (gameManager.boxes == 0) {
    gameManager.victory = true;
  }
  //set loss condition
  if (gameManager.timeLeft <= 0) {
    gameManager.gameOver = true;
  }

  //draw game over
  if (gameManager.gameOver) {
    drawGameOver();
  }
  //draw victory
  else if (gameManager.victory) {
    drawVictory();
  }
  //draw game if loss condition was not met
  else {
    //end of my code
    Engine.update(engine);

    drawGround();

    drawPropeller();

    drawTower();

    drawBirds();

    drawSlingshot();
    //start of my code

    drawHUD();
```

```
  }
  //end of my code

}
////////////////////////////////////////////////////////////
//use arrow keys to control propeller
function keyPressed() {
  if (keyCode == LEFT_ARROW) {
    //start of my code
    angleSpeed += 0.01;
    //end of my code
  }
  else if (keyCode == RIGHT_ARROW) {
    //start of my code
    angleSpeed -= 0.01;
    //end of my code
  }
}
////////////////////////////////////////////////////////////
function keyTyped() {
  //if 'b' create a new bird to use with propeller
  if (key === 'b') {
    setupBird();
  }

  //if 'r' reset the slingshot
  if (key === 'r') {
    removeFromWorld(slingshotBird);
    removeFromWorld(slingshotConstraint);
    setupSlingshot();
  }
}

//************************************************************************
//  HELPER FUNCTIONS - DO NOT WRITE BELOW THIS line
//************************************************************************

//if mouse is released destroy slingshot constraint so that
//slingshot bird can fly off
function mouseReleased() {
  setTimeout(() => {
    slingshotConstraint.bodyB = null;
    slingshotConstraint.pointA = { x: 0, y: 0 };
  }, 100);
}
```

```
////////////////////////////////////////////////////////////
//tells you if a body is off-screen
function isOffScreen(body) {
  var pos = body.position;
  return (pos.y > height || pos.x < 0 || pos.x > width);
}
////////////////////////////////////////////////////////////
//removes a body from the physics world
function removeFromWorld(body) {
  World.remove(engine.world, body);
}
////////////////////////////////////////////////////////////
function drawVertices(vertices) {
  beginShape();
  for (var i = 0; i < vertices.length; i++) {
    vertex(vertices[i].x, vertices[i].y);
  }
  endShape(CLOSE);
}
////////////////////////////////////////////////////////////
function drawConstraint(constraint) {
  push();
  var offsetA = constraint.pointA;
  var posA = { x: 0, y: 0 };
  if (constraint.bodyA) {
    posA = constraint.bodyA.position;
  }
  var offsetB = constraint.pointB;
  var posB = { x: 0, y: 0 };
  if (constraint.bodyB) {
    posB = constraint.bodyB.position;
  }
  strokeWeight(5);
  stroke(255);
  line(
    posA.x + offsetA.x,
    posA.y + offsetA.y,
    posB.x + offsetB.x,
    posB.y + offsetB.y
  );
  pop();
}
//start of my code

function drawHUD() {
```

```
  push();
  fill(255);
  textAlign(CENTER);

  var txtSize = 25;
  var timeLeft = `Time left: ${gameManager.timeLeft}`;
  var boxesLeft = `Boxes left: ${gameManager.boxes}`;

  textSize(txtSize);
  text(timeLeft, width / 2, txtSize);
  text(boxesLeft, width / 2, txtSize * 2 + 3);
  pop();
}

function drawGameOver() {
  fill(255, 0, 0);
  textSize(80);
  textAlign(CENTER);
  text("GAME OVER", width / 2, height / 2);
  noLoop();
}

function drawVictory() {
  fill(0, 255, 0);
  textSize(80);
  textAlign(CENTER);
  text("VICTORY", width / 2, height / 2);
  noLoop();
}
//end of my code
```

b. sketch.js

4. Waving dots

```
/*Notes:

This is my solution for the waving dots assignment.

==================================
Variatons from a regular submission:
==================================

1- Implemented a mouseX normaliser which allows for the dots to stay in place as
MouseX on default
can sometimes give 1 instead of 0.
2- Included mouseX & mouseY into the noise for editing the fill colour and styled
with noStroke()
```

```
==============================
Submission notes on execution:
==============================

1.
For fill colours - R,G & B use noise values:
R - takes frameCount
G - takes mouseX
B - takes mouseY

Reasoning: it looks more interesting as colours both flicker and fade into each
other.

==================================
2.
For drawing the dots:

used ellipse position x/y + size/2
Reasoning: Avoiding the dots being half-rendered out of the canvas.
ellipse size is 1/2 of the size variable.

==================================
3.

For generating noise:
3D noise was generated using dot.x, dot.y and the current frameCount multiplied
by a scaler.

Reasoning: Animation is much smoother when frameCount is scaled.

==================================
4.

For mapping out noise (with mouseX):

mouseX was not used for noise generation but instead was added to the mapping
variables.
Reasoning: It achieved virtually the same results as generating the noise with
mouseX, however
achieving a completely still grid is much easier this way, code looks cleaner and
easier to read.

==================================
4.1.
```

```
2 different scaling variables were used
1 for position, 1 for rotation
Reasoning: Better control over the final product. Rotation and position are
different in sensitivity.

===================================
Thank you for reading! c:
===================================
*/


//helper function to get rid of mouseX being rounded to 1 when at the leftmost of
the screen
//start of my code
function mouseXNormaliser() {
  if (mouseX > 0 && mouseX < 1) {
    return 0;
  }
  return mouseX;
}

//end of my code
function setup() {
  createCanvas(500, 500);
  background(255);
}

function draw() {
  background(255);

  var noOfDots = 20;
  var size = width / noOfDots;
  for (var x = 0; x < noOfDots; x++) {
    for (var y = 0; y < noOfDots; y++) {
      //start of my code
      //generate different noise values for colours
      var rNoise = noise(frameCount);       //changes with time
      var gNoise = noise(mouseX);      //changes with mouse movement
      var bNoise = noise(mouseY);      //changes with mouse movement

      //map noise to byte values
      var r = map(rNoise, 0, 1, 0, 255);
      var g = map(gNoise, 0, 1, 0, 255);
      var b = map(bNoise, 0, 1, 0, 255);
```

```
      //generate colour var
      var clr = color(r, g, b);

      //establish an accurate X and y position
      var posX = size * x;
      var posY = size * y;

      //feed all of that data into params
      params = {
        color: clr,
        x: posX,
        y: posY,
        size: size
      }
      //end of my code
      wave(params);
    }
  }
}


function wave(params) {
  //start of my code
  //add scalers to adjust noise values
  var frameScaler = 0.06;
  var posLimit = mouseXNormaliser() * 0.003;
  var angleLimit = mouseXNormaliser() * 0.000001;

  //generate noise using X & Y pos and scaled frame count as
  //the 3D noise
  var noiseTD = noise(params.x, params.y, frameCount * frameScaler);

  //map to translation using position scaler
  //value to map: 3D noise
  //range1: 0-1
  //range2: +/- (mouseX * position scaler)
  var trans = map(noiseTD, 0, 1, -posLimit, posLimit);

  //map to rotation using rotation scaler
  //value to map: 3D noise
  //range1: 0-1
  //range2: +/- (mouseX * rotation scaler)
  var rot = map(noiseTD, 0, 1, -angleLimit, angleLimit);
```

```
  //translate and rotate canvas
  rotate(rot);
  translate(trans, trans);

  //use noise colour as fill and remove black outline
  noStroke();
  fill(params.color);

  //draw ellipse
  ellipse(params.x + params.size / 2, params.y + params.size / 2, params.size / 2);
  //end of my code

}
```

a. sketch.js