Jerry Cain
CS 106AJ

Handout #24S
October 24-25, 2018

Solutions for Section #4

Portions of this handout by Eric Roberts, Patrick Young, Jeremy Keeshin, Mehran Sahami, Nick Troccoli, and Kat Gregory

Solution 1: Interactive, Animated Random Circles

```
/*
 * File: GrowCircles.js
 * --------------------
 * This program responds to each mouse click by animating a random
 * circle that grows on the location of the click.
 */

/* Constants */
const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 300;
const MIN_RADIUS = 15;
const MAX_RADIUS = 50;
const TIME_STEP = 20;
const DELTA_SIZE = 1;

/* Main program */
function GrowCircles() {
   let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);

   let createNewCircle = function(x, y) {
      let newCircle = GOval(x, y, 0, 0);
      newCircle.setFilled(true);
      newCircle.setColor(randomColor());
      return newCircle;
   };

   let clickAction = function(e) {
      let circle = createNewCircle(e.getX(), e.getY());
      gw.add(circle);

      let r = randomReal(MIN_RADIUS, MAX_RADIUS);
      let desiredSize = 2 * r;
      let currentSize = 0;

      let step = function() {
         if (currentSize < desiredSize) {
            currentSize += DELTA_SIZE;
            let x = circle.getX() - DELTA_SIZE / 2;
            let y = circle.getY() - DELTA_SIZE / 2;
            circle.setBounds(x, y, currentSize, currentSize);
         } else {
            clearInterval(timer);
         }
      };

      let timer = setInterval(step, TIME_STEP);
   };

   gw.addEventListener("click", clickAction);
}
```

Solution 2: Simulating Gravity with Bouncing Balls

```
/*
 * File: BouncingBalls.js
 * ----------------------
 * This program graphically drops a bouncing ball each time user clicks.
 */

/* Constants */
const GWINDOW_WIDTH = 500;
const GWINDOW_HEIGHT = 300;
const DIAMETER = 20;
const MIN_X_VEL = 3;
const MAX_X_VEL = 15;
const TIME_STEP = 20;
const GRAVITY = 3; // Amount Y vel is increased each cycle
const BOUNCE_REDUCE = 0.75; // Amount Y velocity is reduced during bounce

/* Main program */
function BouncingBalls() {
   let gw = GWindow(GWINDOW_WIDTH, GWINDOW_HEIGHT);

   let createNewBall = function() {
      let newBall = GOval(0, 0, DIAMETER, DIAMETER);
      newBall.setFilled(true);
      newBall.setColor(randomColor());
      return newBall;
   };

   let clickAction = function(e) {
      let ball = createNewBall();
      let xVel = randomReal(MIN_X_VEL, MAX_X_VEL);
      let yVel = 0;
      gw.add(ball);

      let checkForCollision = function() {
         // Determine if ball has dropped below the floor
         if (ball.getY() > GWINDOW_HEIGHT - DIAMETER) {
            // Change ball's Y velocity to now bounce upwards
            yVel = -yVel * BOUNCE_REDUCE;

            // Assume bounce will move ball an amount above the floor
            // equal to the amount it would have dropped below the floor.
            let diff = ball.getY() - (GWINDOW_HEIGHT - DIAMETER);
            ball.move(0, -2 * diff);
         }
      };

      let step = function() {
         if (ball.getX() < GWINDOW_WIDTH) {
            yVel += GRAVITY;
            ball.move(xVel, yVel);
            checkForCollision();
         } else { // Simulation ends when ball exits right side of screen
            clearInterval(timer);
         }
      };

      let timer = setInterval(step, TIME_STEP);
   };

   gw.addEventListener("click", clickAction);
}
```

Solution 3: Adding commas to numeric strings

```
function addCommasToNumericString(digits) {
   let result = "";
   let len = digits.length;
   let nDigits = 0;
   for (let i = len - 1; i >= 0; i--) {
      result = digits.charAt(i) + result;
      nDigits++;
      if (((nDigits % 3) === 0) && (i > 0)) {
         result = "," + result;
      }
   }
   return result;
}
```