

15-104 Introduction to Computing for Creative Practice

Fall 2021

36 Snake Game (from an example by Prashant Gupta, <https://github.com/prashantgupta24>)

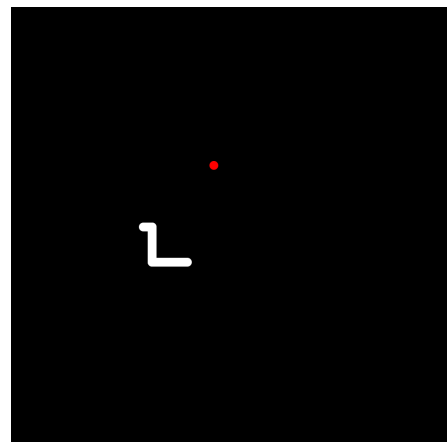
Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1




Classic Snake Game

- Control the movement of the snake with the arrow keys.
- If the snake eats the “fruit”, the snake grows longer.
- Avoid the sides of the canvas! If the snake touches a side of the canvas, the game is over.



2



Fruit (as an object)


```
function makeFruit() {
  var fruit = { x: 0, y: 0, draw: drawFruit, move: moveFruit };
  return fruit;
}

function drawFruit() {
  stroke(255, 0, 0);
  point(this.x, this.y);
}

function moveFruit() {
  this.x = floor(random(10, (width - 100) / 10)) * 10;
  this.y = floor(random(10, (height - 100) / 10)) * 10;
}
```

Set the fruit point to lie in between 100 and width-100, and be rounded off to the nearest number divisible by 10, since the snake moves in multiples of 10.
Example: $\text{floor}(124 / 10) * 10 = \text{floor}(12.4) * 10 = 12 * 10 = 120$.

3



Snake

```
var xTail = 0;
var yTail = 250;
var numSegments = 10;
var direction = 'right';
var xCor = [];
var yCor = [];


function initSnake() {
  for (var i = 0; i < numSegments; i++) {
    xCor.push(xTail + i * 10);
    yCor.push(yTail);
  }
}
```

	0	1	2	3	4	5	6	7	8	9
xCor	0	10	20	30	40	50	60	70	80	90
yCor	250	250	250	250	250	250	250	250	250	250

(tail) (head)

The xCor and yCor arrays hold the x and y coordinates of each segment of the snake. The tail is at (xCor[0], yCor[0]) and the head is at (xCor[numSegments-1], yCor[numSegments-1]).

4




Drawing the snake

```
function drawSnake() {
  stroke(0, 255, 0);
  for (var i = 0; i < numSegments - 1; i++) {
    line(xCor[i], yCor[i], xCor[i + 1], yCor[i + 1]);
  }
}
```

Connect each adjacent pair of coordinates to form the snake.
REMEMBER: if there are `numSegments` segments, then there is one less line to draw. (“the fence rail scenario”).

5




Moving the snake

```
function moveSnake() {
  var x_diff = 0;
  var y_diff = 0;
  for (var i = 0; i < numSegments - 1; i++) {
    xCor[i] = xCor[i + 1];
    yCor[i] = yCor[i + 1];
  }
  switch (direction) {
    case 'right': x_diff = 10; break;
    case 'left' : x_diff = -10; break;
    case 'down' : y_diff = 10; break;
    case 'up'   : y_diff = -10; break;
  }
  xCor[numSegments - 1] = xCor[numSegments - 2] + x_diff;
  yCor[numSegments - 1] = yCor[numSegments - 2] + y_diff;
}
```

Copy each segment to the previous segment.
 (so position 1 moves to position 0 and becomes the new tail)

Depending on the direction, add a positive or negative value to the previous head of the snake (which is now in position `numSegments-2` to create a new head of the snake in position `numSegments-1`).

6




Will the snake collide with itself?

```
function snakeCollision() {
  var snakeHeadX = xCor[numSegments - 1];
  var snakeHeadY = yCor[numSegments - 1];
  for (var i = 0; i < numSegments - 1; i++) {
    if (xCor[i] == snakeHeadX && yCor[i] == snakeHeadY) {
      return true;
    }
  }
  return false;
}
```

Check all segments of the snake (except the head) to see if any are in the same position with the head of the snake. If so, it collided with itself.


7



Putting the game together (uses DOM)

```
var scoreElem; // score element for web page
var score = 0;
var fruit;
function setup() {
  scoreElem = createDiv('Score = 0');
  scoreElem.position(20, 20);
  scoreElem.id = 'score';
  scoreElem.style('color', 'white');
  createCanvas(500, 500);
  frameRate(15);
  strokeWeight(10);
  initSnake();
  fruit = makeFruit();
  fruit.move();
}
```


8



The main draw loop

```
function draw() {
  background(0);
  fruit.draw();
  drawSnake();
  moveSnake();
  checkForCollision(); // with wall or itself
  checkForFruit();   // if it eats fruit, move fruit
                    // and extend the snake
}
```

9




Check for collision

```
function checkForCollision() {
  var xHead = xCor[numSegments - 1];
  var yHead = yCor[numSegments - 1];
  if (xHead > width || xHead < 0 || yHead > height || yHead < 0 ||
      snakeCollision() ) {
    noLoop(); // stop the game by stopping the draw function
    scoreElem.html('Game ended! Your score was : ' + score);
  }
}
```

Update the div element on the webpage with the final score.

10



Check for fruit


```
function checkForFruit() {
  var xHead = xCor[numSegments - 1];
  var yHead = yCor[numSegments - 1];
  if (xHead == fruit.x && yHead == fruit.y) {
    score++;
    scoreElem.html('Score = ' + score);
    xCor.unshift(xCor[0]);
    yCor.unshift(yCor[0]);
    numSegments += 1;
    fruit.move();
  }
}
```

Update the div element on the webpage with the updated score.

Add a new tail to the beginning of the array by inserting the current tail using `unshift`.
THINK: Why will this work?

Move the fruit to a new location to be eaten.

11



Using the arrow keys to steer the snake

```
function keyPressed() {
  switch (keyCode) {
    case LEFT_ARROW:
      if (direction !== 'right') direction = 'left'; break;
    case RIGHT_ARROW:
      if (direction !== 'left') direction = 'right'; break;
    case UP_ARROW:
      if (direction !== 'down') direction = 'up'; break;
    case DOWN_ARROW:
      if (direction !== 'up') direction = 'down'; break;
  }
}
```

If you press the left arrow, your direction will be set to left but only if you're not going right. Why?

WE NOW HAVE ALL THE COMPONENTS DONE! LET'S PLAY!

12