Fun Snake

# Overview

Your goal in this assignment is to create a really fun snake game. We will start by making the snake.

## Specifications Part A

Our snake only gets bigger. We create one square at the center to be our snake, and then every DELAY milliseconds, we add one square in the direction the snake is moving. The snake starts moving east.

Hints:

* You want to keep track of a current direction the snake is facing.
* You also want to keep track of the snake’s current x and y position.
* You’ll need several functions to make this program easier. You need to respond to a keypress – the arrow keys should set the snake’s current direction. Take a look at some of your earlier code.

## Starter Code

var SNAKE\_DIM = 10;

var SNAKE\_COLOR = Color.green;

var NORTH = 0;

var EAST = 1;

var SOUTH = 2;

var WEST = 3;

var FOOD\_DELAY = 6000;

var FOOD\_RADIUS = 5;

var FOOD\_COLOR = Color.red;

var POINTS\_TEXT\_X = 5;

var POINTS\_TEXT\_Y\_BUFFER = 5;

var POINTS\_FONT = "17pt Arial";

function start(){

}

## Specifications Part B

Your goal in this part is to get the snake to know when it collides with itself. Our snake collides with itself when there is already a part of the snake in the next spot that we are going to be. You’ll want to use the function getElementAt(x,y) to see if there is already an object there.

When you lose, you should put text up on the screen displaying a message to that effect.

## Specifications Part C

Your goal in this part is to draw food for the snake every FOOD\_DELAY milliseconds. Food is a red circle drawn at a random location on screen.

Our snake should be able to eat the food if the snake goes over the same location as the food. If the snake eats the food it should be removed.

## Specifications Part D

Your goal in this part is to make this a real live and awesome snake game. This means we need to add points. And this also means the snake needs to be able to go through walls.

When you are done, you should have someone test and review your game.

Each time the snake gets bigger, you should add one point to the users score. Each time the snake eats food, you should add 100 points. If the user loses, you should stop adding points.

If the snake moves all the way past the right wall, the snake should end up on the left. If the snake goes through the left wall, the snake should end up on the right. Similarly the snake should be able to go through the top and bottom walls. Hint: You can use the modulus operator to solve this one in quite a nice way.

# Planning

You must plan out each part of the program separately. Please create flowcharts, pseudocode or storyboard for each section, before working on the code.

# Testing

# Reflection

These should be answer in complete sentences. You should attempt to answer these questions to the best of your ability. If you have a hard time writing answer out; you may record or use a speech to text tool to anwser the questions.

1. What was the most challenging part of this Project?
2. Why is the project breakdown into 4 parts? How would you do it differently?
3. How did you use functions in this program? Why did you use them that way?
4. What events did you use in the game? Why?

Rubric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Skill** | **1** | **2** | **3** | **4** |
| **10. Discuss and apply fundamental concepts of programming language. (i.e.: Data types, memory models, data structures, etc.)** | Students can recognize the fundamental concepts of programming languages | Student can identify and categorize the fundamental concepts of programming languages | Student can compare fundamental concepts and explain the differences with evidence to support them. | Student can identify the fundamental concepts of programming and justify how and why someone would use them with specific examples |
| **11. Demonstrate and use a variety of software development tools for program implementation** | Student can recognize and identify different software development tools | Student can explain the difference between two or more software development tools and use different software development tools to solve routine problems | Student analyze and select an appropriate software development tool and can use different software development tools to solve non routine problems | Students can evaluate a problem and justify which software development tools they would use to solve the problem with evidence to support their argument |
| **12. Write computer programs utilizing the structured programming paradigm.** | Student can identify the structured programming paradigm and how it functions | Student can use the structured programming paradigm to solve a routine problem | Student can use the structured programming paradigm to solve a non routine problem | Student can analyze a series of complex problems and modify them with justification of the structure in the new designs |
| **14. Write programs using modularization techniques to reduce program complexity and improve program maintainability** | Student can identify modularization techniques | Student can explain why modularization is used and how it can be useful and use it to solve routine problems | Student can formulate a modularized program for a non routine problem | Student can critique the use of modularization in a program with an explanation and evidence to support their argument |
| **19. Write programs that use events to cause program execution to react to the event by writing the appropriate event handler code** | Student can identify what events and event handlers | Student can explain the difference between an event and an event handler and solve routine problems with events | Student can explain the difference between an event and an event handler with evidence and solve non routine problems with events | Student can create a program that uses events and critique the use or justify use of those events |
| Target Grade: 12 | | TOTAL | |  |