**Manual de usuario**

Contenido

[**descripcion paso a paso** 2](#_Toc38564702)

[**Solución de problemas**  2](#_Toc38564703)

[**manual de usuario** 3](#_Toc38564704)

[**Nombre del autor:** 4](#_Toc38564705)

[**Algoritmo:** 4](#_Toc38564706)

[**Código Fuente** 6](#_Toc38564708)

**Descripción paso a paso**

1. Es un juego de habilidad y precisión.
2. Debe tener el debido control de manejo.
3. Es un documento echo en html5.
4. Con estilos css.

**Descripción del problema:**

El juego es algo veloz mientras el jugador no analice rápido podrá perder ya que la serpiente cada vez que pasa por un punto color rojo se hace mas grande y puede que llegue a topar con ella misma o topar en las partes del cuadro.

**Manual de Usuario:** usar las siguientes teclas

**Para ir a la derecha**

**Para ir a la derecha**

**Para ir a bajo**

**Para ir arriba**

**Nombre del autor: Andi Adonai Valenzuela Castañeda**

**Algoritmo**

A un lado hay unos textos donde podrá ver las instrucciones o estar en el inicio.

Usar las teclas para poder controlar la serpiente

Al ingresar al juego iniciara de una vez

Ingresar a la pagina o al juego

**Código**

**Index.html**

**<!doctype html>**

**<html lang="en">**

**<link rel="stylesheet" href="estilo.css">**

**<head>**

**<meta charset="UTF-8">**

**<title>Simple Snake Game</title>**

**<!-- Basic styling, centering of the canvas. -->**

**<style>**

**canvas {**

**display: block;**

**position: absolute;**

**border: 1px solid #000;**

**margin: auto;**

**top: 0;**

**bottom: 0;**

**right: 0;**

**left: 0;**

**}**

**</style>**

**</head>**

**<body>**

**<script>**

**var**

**/\*\***

**\* Constats**

**\*/**

**COLS = 26,**

**ROWS = 26,**

**EMPTY = 0,**

**SNAKE = 1,**

**FRUIT = 2,**

**LEFT = 0,**

**UP = 1,**

**RIGHT = 2,**

**DOWN = 3,**

**KEY\_LEFT = 37,**

**KEY\_UP = 38,**

**KEY\_RIGHT = 39,**

**KEY\_DOWN = 40,**

**/\*\***

**\* Game objects**

**\*/**

**canvas, /\* HTMLCanvas \*/**

**ctx, /\* CanvasRenderingContext2d \*/**

**keystate, /\* Object, used for keyboard inputs \*/**

**frames, /\* number, used for animation \*/**

**score; /\* number, keep track of the player score \*/**

**/\*\***

**\* Grid datastructor, usefull in games where the game world is**

**\* confined in absolute sized chunks of data or information.**

**\***

**\* @type {Object}**

**\*/**

**grid = {**

**width: null, /\* number, the number of columns \*/**

**height: null, /\* number, the number of rows \*/**

**\_grid: null, /\* Array<any>, data representation \*/**

**/\*\***

**\* Initiate and fill a c x r grid with the value of d**

**\* @param {any} d default value to fill with**

**\* @param {number} c number of columns**

**\* @param {number} r number of rows**

**\*/**

**init: function(d, c, r) {**

**this.width = c;**

**this.height = r;**

**this.\_grid = [];**

**for (var x=0; x < c; x++) {**

**this.\_grid.push([]);**

**for (var y=0; y < r; y++) {**

**this.\_grid[x].push(d);**

**}**

**}**

**},**

**/\*\***

**\* Set the value of the grid cell at (x, y)**

**\***

**\* @param {any} val what to set**

**\* @param {number} x the x-coordinate**

**\* @param {number} y the y-coordinate**

**\*/**

**set: function(val, x, y) {**

**this.\_grid[x][y] = val;**

**},**

**/\*\***

**\* Get the value of the cell at (x, y)**

**\***

**\* @param {number} x the x-coordinate**

**\* @param {number} y the y-coordinate**

**\* @return {any} the value at the cell**

**\*/**

**get: function(x, y) {**

**return this.\_grid[x][y];**

**}**

**}**

**/\*\***

**\* The snake, works as a queue (FIFO, first in first out) of data**

**\* with all the current positions in the grid with the snake id**

**\***

**\* @type {Object}**

**\*/**

**snake = {**

**direction: null, /\* number, the direction \*/**

**last: null, /\* Object, pointer to the last element in**

**the queue \*/**

**\_queue: null, /\* Array<number>, data representation\*/**

**/\*\***

**\* Clears the queue and sets the start position and direction**

**\***

**\* @param {number} d start direction**

**\* @param {number} x start x-coordinate**

**\* @param {number} y start y-coordinate**

**\*/**

**init: function(d, x, y) {**

**this.direction = d;**

**this.\_queue = [];**

**this.insert(x, y);**

**},**

**/\*\***

**\* Adds an element to the queue**

**\***

**\* @param {number} x x-coordinate**

**\* @param {number} y y-coordinate**

**\*/**

**insert: function(x, y) {**

**// unshift prepends an element to an array**

**this.\_queue.unshift({x:x, y:y});**

**this.last = this.\_queue[0];**

**},**

**/\*\***

**\* Removes and returns the first element in the queue.**

**\***

**\* @return {Object} the first element**

**\*/**

**remove: function() {**

**// pop returns the last element of an array**

**return this.\_queue.pop();**

**}**

**};**

**/\*\***

**\* Set a food id at a random free cell in the grid**

**\*/**

**function setFood() {**

**var empty = [];**

**// iterate through the grid and find all empty cells**

**for (var x=0; x < grid.width; x++) {**

**for (var y=0; y < grid.height; y++) {**

**if (grid.get(x, y) === EMPTY) {**

**empty.push({x:x, y:y});**

**}**

**}**

**}**

**// chooses a random cell**

**var randpos = empty[Math.round(Math.random()\*(empty.length - 1))];**

**grid.set(FRUIT, randpos.x, randpos.y);**

**}**

**/\*\***

**\* Starts the game**

**\*/**

**function main() {**

**// create and initiate the canvas element**

**canvas = document.createElement("canvas");**

**canvas.width = COLS\*20;**

**canvas.height = ROWS\*20;**

**ctx = canvas.getContext("2d");**

**// add the canvas element to the body of the document**

**document.body.appendChild(canvas);**

**// sets an base font for bigger score display**

**ctx.font = "12px Helvetica";**

**frames = 0;**

**keystate = {};**

**// keeps track of the keybourd input**

**document.addEventListener("keydown", function(evt) {**

**keystate[evt.keyCode] = true;**

**});**

**document.addEventListener("keyup", function(evt) {**

**delete keystate[evt.keyCode];**

**});**

**// intatiate game objects and starts the game loop**

**init();**

**loop();**

**}**

**/\*\***

**\* Resets and inits game objects**

**\*/**

**function init() {**

**score = 0;**

**grid.init(EMPTY, COLS, ROWS);**

**var sp = {x:Math.floor(COLS/2), y:ROWS-1};**

**snake.init(UP, sp.x, sp.y);**

**grid.set(SNAKE, sp.x, sp.y);**

**setFood();**

**}**

**/\*\***

**\* The game loop function, used for game updates and rendering**

**\*/**

**function loop() {**

**update();**

**draw();**

**// When ready to redraw the canvas call the loop function**

**// first. Runs about 60 frames a second**

**window.requestAnimationFrame(loop, canvas);**

**}**

**/\*\***

**\* Updates the game logic**

**\*/**

**function update() {**

**frames++;**

**// changing direction of the snake depending on which keys**

**// that are pressed**

**if (keystate[KEY\_LEFT] && snake.direction !== RIGHT) {**

**snake.direction = LEFT;**

**}**

**if (keystate[KEY\_UP] && snake.direction !== DOWN) {**

**snake.direction = UP;**

**}**

**if (keystate[KEY\_RIGHT] && snake.direction !== LEFT) {**

**snake.direction = RIGHT;**

**}**

**if (keystate[KEY\_DOWN] && snake.direction !== UP) {**

**snake.direction = DOWN;**

**}**

**// each five frames update the game state.**

**if (frames%5 === 0) {**

**// pop the last element from the snake queue i.e. the**

**// head**

**var nx = snake.last.x;**

**var ny = snake.last.y;**

**// updates the position depending on the snake direction**

**switch (snake.direction) {**

**case LEFT:**

**nx--;**

**break;**

**case UP:**

**ny--;**

**break;**

**case RIGHT:**

**nx++;**

**break;**

**case DOWN:**

**ny++;**

**break;**

**}**

**// checks all gameover conditions**

**if (0 > nx || nx > grid.width-1 ||**

**0 > ny || ny > grid.height-1 ||**

**grid.get(nx, ny) === SNAKE**

**) {**

**return init();**

**}**

**// check wheter the new position are on the fruit item**

**if (grid.get(nx, ny) === FRUIT) {**

**// increment the score and sets a new fruit position**

**score++;**

**setFood();**

**} else {**

**// take out the first item from the snake queue i.e**

**// the tail and remove id from grid**

**var tail = snake.remove();**

**grid.set(EMPTY, tail.x, tail.y);**

**}**

**// add a snake id at the new position and append it to**

**// the snake queue**

**grid.set(SNAKE, nx, ny);**

**snake.insert(nx, ny);**

**}**

**}**

**/\*\***

**\* Render the grid to the canvas.**

**\*/**

**function draw() {**

**// calculate tile-width and -height**

**var tw = canvas.width/grid.width;**

**var th = canvas.height/grid.height;**

**// iterate through the grid and draw all cells**

**for (var x=0; x < grid.width; x++) {**

**for (var y=0; y < grid.height; y++) {**

**// sets the fillstyle depending on the id of**

**// each cell**

**switch (grid.get(x, y)) {**

**case EMPTY:**

**ctx.fillStyle = "#fff";**

**break;**

**case SNAKE:**

**ctx.fillStyle = "#0ff";**

**break;**

**case FRUIT:**

**ctx.fillStyle = "#f00";**

**break;**

**}**

**ctx.fillRect(x\*tw, y\*th, tw, th);**

**}**

**}**

**// changes the fillstyle once more and draws the score**

**// message to the canvas**

**ctx.fillStyle = "#000";**

**ctx.fillText("SCORE: " + score, 10, canvas.height-10);**

**}**

**// start and run the game**

**main();**

**</script>**

**<br>**

**<a href="../index.html">!!PUCHALE AQUI PARA REGRESAR A LA PAGINA INICIO!!</a></br>**

**<br>**

**<a href="../and-val/instrucciones.html">!PUCHALE AQUI PARA VER LAS INSTRUCCIONES!</a></br>**

**</body>**

**</html>**

**Intrucciones.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Juego Snake</title>

</head>

<h1>

<body text="white">

<link rel="stylesheet" href="estilo.css">

<p align="center"> Snake</p>

<br>

<p align="center">Usar las teclas de direccion para controlar la serpiente</p>

<br>

<p align="center">Debe comerce el cuadro rojo y asi la serpiente se hara mas grande</p>

<br>

<p align="center">No dejar que la serpiente tope en los lados del cuadro o en si misma sino se acabar su juego y reiniciara otro</p>

<br>

<a href="../and-val/index.html">!PUCHALE PLAY!</a></br>

<br>

<a href="../index.html">!!PUCHALE AQUI PARA REGRESAR A LA PAGINA INICIO</a></br>

</h1>

</body>

</html>

**Estilo.css**

body{

background-position: center;

background-image: url(fondo-tecnológico.jpg);

background-size: cover;

}