



**MANUAL DEL PROGRAMADOR**

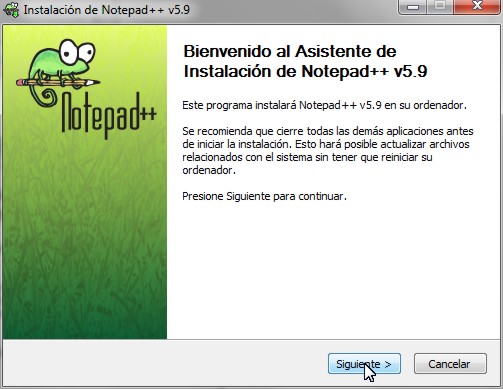
**PROGRAMAR CON NOTEPAD++**

Elige este programa porque soporta casi todas la extensiones de los lenguajes de programación haciéndome una tarea más simple posible

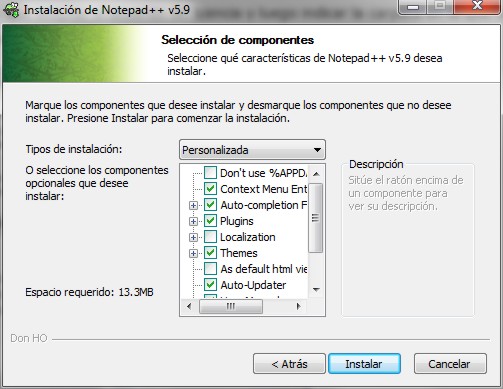
Para iniciar la instalación de Notepad++, ejecuta el archivo descargado (**npp.5.9.Installer.exe** - los números de versión 5.9 puede que sean distintos). Comienza preguntándote por el idioma:



Y se inicia el asistente para la instalación:



Es un instalador bastante estándar. Tras pulsar **Siguiente** tienes que aceptar el acuerdo de licencia y luego indicar la carpeta en la que quieres que se instale. La que propone es adecuada, así que basta con que pulses **Siguiente**.



Cuando te pregunte por la selección de componentes, deja las opciones que te ofrece por defecto. Pulsa **Instalar** y se completará el proceso de instalación. Al final, te dará la opción de ejecutar directamente Notepad++.

Puede que el asistente de instalación sea algo distinto, por ser el de una versión posterior a la que había cuando se creó este documento, pero será muy similar.

Para saber cómo usar **Notepad++**, lee el documento **Notepad++ Uso** de la página anterior.

**El paquete que la programación del juego.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*para el javascript\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// nos marca los pulsos del juego

window.requestAnimFrame = (function () {

return window.requestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.oRequestAnimationFrame ||

window.msRequestAnimationFrame ||

function ( /\* function \*/ callback, /\* DOMElement \*/ element) {

window.setTimeout(callback, 1000 / 60);

};

})();

arrayRemove = function (array, from) {

var rest = array.slice((from) + 1 || array.length);

array.length = from < 0 ? array.length + from : from;

return array.push.apply(array, rest);

};

var game = (function () {

// Variables globales a la aplicacion

var canvas,

ctx,

buffer,

bufferctx,

player,

evil,

playerShot,

bgMain,

bgBoss,

evilSpeed = 1,

totalEvils = 7,

playerLife = 5,

shotSpeed = 5,

playerSpeed = 5,

evilCounter = 0,

youLoose = false,

congratulations = false,

minHorizontalOffset = 100,

maxHorizontalOffset = 400,

evilShots = 5, // disparos que tiene el malo al principio

evilLife = 3, // vidas que tiene el malo al principio (se van incrementando)

finalBossShots = 30,

finalBossLife = 12,

totalBestScoresToShow = 5, // las mejores puntuaciones que se mostraran

playerShotsBuffer = [],

evilShotsBuffer = [],

evilShotImage,

playerShotImage,

playerKilledImage,

evilImages = {

animation : [],

killed : new Image()

},

bossImages = {

animation : [],

killed : new Image()

},

keyPressed = {},

keyMap = {

left: 37,

right: 39,

fire: 32 // tecla espacio

},

nextPlayerShot = 0,

playerShotDelay = 250,

now = 0;

function loop() {

update();

draw();

}

function preloadImages () {

for (var i = 1; i <= 8; i++) {

var evilImage = new Image();

evilImage.src = 'images/malo' + i + '.png';

evilImages.animation[i-1] = evilImage;

var bossImage = new Image();

bossImage.src = 'images/jefe' + i + '.png';

bossImages.animation[i-1] = bossImage;

}

evilImages.killed.src = 'images/malo\_muerto.png';

bossImages.killed.src = 'images/jefe\_muerto.png';

bgMain = new Image();

bgMain.src = 'images/fondovertical.png';

bgBoss = new Image();

bgBoss.src = 'images/fondovertical\_jefe.png';

playerShotImage = new Image();

playerShotImage.src = 'images/disparo\_bueno.png';

evilShotImage = new Image();

evilShotImage.src = 'images/disparo\_malo.png';

playerKilledImage = new Image();

playerKilledImage.src = 'images/bueno\_muerto.png';

}

function init() {

preloadImages();

showBestScores();

canvas = document.getElementById('canvas');

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

buffer = document.createElement('canvas');

buffer.width = canvas.width;

buffer.height = canvas.height;

bufferctx = buffer.getContext('2d');

player = new Player(playerLife, 0);

evilCounter = 1;

createNewEvil();

showLifeAndScore();

addListener(document, 'keydown', keyDown);

addListener(document, 'keyup', keyUp);

function anim () {

loop();

requestAnimFrame(anim);

}

anim();

}

function showLifeAndScore () {

bufferctx.fillStyle="rgb(59,59,59)";

bufferctx.font="bold 16px Arial";

bufferctx.fillText("Puntos: " + player.score, canvas.width - 100, 20);

bufferctx.fillText("Vidas: " + player.life, canvas.width - 100,40);

}

function getRandomNumber(range) {

return Math.floor(Math.random() \* range);

}

function Player(life, score) {

var settings = {

marginBottom : 10,

defaultHeight : 66

};

player = new Image();

player.src = 'images/bueno.png';

player.posX = (canvas.width / 2) - (player.width / 2);

player.posY = canvas.height - (player.height == 0 ? settings.defaultHeight : player.height) - settings.marginBottom;

player.life = life;

player.score = score;

player.dead = false;

player.speed = playerSpeed;

var shoot = function () {

if (nextPlayerShot < now || now == 0) {

playerShot = new PlayerShot(player.posX + (player.width / 2) - 5 , player.posY);

playerShot.add();

now += playerShotDelay;

nextPlayerShot = now + playerShotDelay;

} else {

now = new Date().getTime();

} };

player.doAnything = function() {

if (player.dead)

return;

if (keyPressed.left && player.posX > 5)

player.posX -= player.speed;

if (keyPressed.right && player.posX < (canvas.width - player.width - 5))

player.posX += player.speed;

if (keyPressed.fire)

shoot(); };

player.killPlayer = function() {

if (this.life > 0) {

this.dead = true;

evilShotsBuffer.splice(0, evilShotsBuffer.length);

playerShotsBuffer.splice(0, playerShotsBuffer.length);

this.src = playerKilledImage.src;

createNewEvil();

setTimeout(function () {

player = new Player(player.life - 1, player.score);

}, 500); } else {

saveFinalScore();

youLoose = true;

} };

return player; }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DISPAROS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Shot( x, y, array, img) {

this.posX = x;

this.posY = y;

this.image = img;

this.speed = shotSpeed;

this.identifier = 0;

this.add = function () {

array.push(this); };

this.deleteShot = function (idendificador) {

arrayRemove(array, idendificador);

}; }

function PlayerShot (x, y) {

Object.getPrototypeOf(PlayerShot.prototype).constructor.call(this, x, y, playerShotsBuffer, playerShotImage);

this.isHittingEvil = function() {

return (!evil.dead && this.posX >= evil.posX && this.posX <= (evil.posX + evil.image.width) &&

this.posY >= evil.posY && this.posY <= (evil.posY + evil.image.height));

}; }

PlayerShot.prototype = Object.create(Shot.prototype);

PlayerShot.prototype.constructor = PlayerShot;

function EvilShot (x, y) {

Object.getPrototypeOf(EvilShot.prototype).constructor.call(this, x, y, evilShotsBuffer, evilShotImage);

this.isHittingPlayer = function() {

return (this.posX >= player.posX && this.posX <= (player.posX + player.width)

&& this.posY >= player.posY && this.posY <= (player.posY + player.height));

}; }

EvilShot.prototype = Object.create(Shot.prototype);

EvilShot.prototype.constructor = EvilShot;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN DISPAROS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ENEMIGOS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Enemy(life, shots, enemyImages) {

this.image = enemyImages.animation[0];

this.imageNumber = 1;

this.animation = 0;

this.posX = getRandomNumber(canvas.width - this.image.width);

this.posY = -50;

this.life = life ? life : evilLife;

this.speed = evilSpeed;

this.shots = shots ? shots : evilShots;

this.dead = false;

var desplazamientoHorizontal = minHorizontalOffset +

getRandomNumber(maxHorizontalOffset - minHorizontalOffset);

this.minX = getRandomNumber(canvas.width - desplazamientoHorizontal);

this.maxX = this.minX + desplazamientoHorizontal - 40;

this.direction = 'D';

this.kill = function() {

this.dead = true;

totalEvils --;

this.image = enemyImages.killed;

verifyToCreateNewEvil(); };

this.update = function () {

this.posY += this.goDownSpeed;

if (this.direction === 'D') {

if (this.posX <= this.maxX) {

this.posX += this.speed;

} else {

this.direction = 'I';

this.posX -= this.speed; }

} else {

if (this.posX >= this.minX) {

this.posX -= this.speed;

} else {

this.direction = 'D';

this.posX += this.speed;

} }

this.animation++;

if (this.animation > 5) {

this.animation = 0;

this.imageNumber ++;

if (this.imageNumber > 8) {

this.imageNumber = 1; }

this.image = enemyImages.animation[this.imageNumber - 1];

} };

this.isOutOfScreen = function() {

return this.posY > (canvas.height + 15);

};

function shoot() {

if (evil.shots > 0 && !evil.dead) {

var disparo = new EvilShot(evil.posX + (evil.image.width / 2) - 5 , evil.posY + evil.image.height);

disparo.add();

evil.shots --;

setTimeout(function() {

shoot();

}, getRandomNumber(3000));

} }

setTimeout(function() {

shoot();

}, 1000 + getRandomNumber(2500));

this.toString = function () {

return 'Enemigo con vidas:' + this.life + 'shotss: ' + this.shots + ' puntos por matar: ' + this.pointsToKill;

} }

function Evil (vidas, disparos) {

Object.getPrototypeOf(Evil.prototype).constructor.call(this, vidas, disparos, evilImages);

this.goDownSpeed = evilSpeed;

this.pointsToKill = 5 + evilCounter; }

Evil.prototype = Object.create(Enemy.prototype);

Evil.prototype.constructor = Evil;

z function FinalBoss () {

Object.getPrototypeOf(FinalBoss.prototype).constructor.call(this, finalBossLife, finalBossShots, bossImages);

this.goDownSpeed = evilSpeed/2;

this.pointsToKill = 20; }

FinalBoss.prototype = Object.create(Enemy.prototype);

FinalBoss.prototype.constructor = FinalBoss;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FIN ENEMIGOS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function verifyToCreateNewEvil() {

if (totalEvils > 0) {

setTimeout(function() {

createNewEvil();

evilCounter ++;

}, getRandomNumber(3000));

} else {

setTimeout(function() {

saveFinalScore();

congratulations = true;

}, 2000);

} }

function createNewEvil() {

if (totalEvils != 1) {

evil = new Evil(evilLife + evilCounter - 1, evilShots + evilCounter - 1);

} else {

evil = new FinalBoss();

} }

function isEvilHittingPlayer() {

return ( ( (evil.posY + evil.image.height) > player.posY && (player.posY + player.height) >= evil.posY ) &&

((player.posX >= evil.posX && player.posX <= (evil.posX + evil.image.width)) ||

(player.posX + player.width >= evil.posX && (player.posX + player.width) <= (evil.posX + evil.image.width))));

}

function checkCollisions(shot) {

if (shot.isHittingEvil()) {

if (evil.life > 1) {

evil.life--;

} else {

evil.kill();

player.score += evil.pointsToKill;

}

shot.deleteShot(parseInt(shot.identifier));

return false; }

return true; }

function playerAction() {

player.doAnything(); }

function addListener(element, type, expression, bubbling) {

bubbling = bubbling || false;

if (window.addEventListener) { // Standard

element.addEventListener(type, expression, bubbling);

} else if (window.attachEvent) { // IE

element.attachEvent('on' + type, expression);

} }

function keyDown(e) {

var key = (window.event ? e.keyCode : e.which);

for (var inkey in keyMap) {

if (key === keyMap[inkey]) {

e.preventDefault();

keyPressed[inkey] = true;

} } }

function keyUp(e) {

var key = (window.event ? e.keyCode : e.which);

for (var inkey in keyMap) {

if (key === keyMap[inkey]) {

e.preventDefault();

keyPressed[inkey] = false;

} } }

function draw() {

ctx.drawImage(buffer, 0, 0); }

function showGameOver() {

bufferctx.fillStyle="rgb(255,0,0)";

bufferctx.font="bold 25px Arial";

bufferctx.fillText("ESTAS MUERTO!!!", canvas.width / 2 - 100, canvas.height / 2);

}

function showCongratulations () {

bufferctx.fillStyle="rgb(204,50,153)";

bufferctx.font="bold 22px Arial";

bufferctx.fillText("Enhorabuena, has ganado el juego!", canvas.width / 2 - 200, canvas.height / 2 - 30);

bufferctx.fillText("PUNTOS: " + player.score, canvas.width / 2 - 200, canvas.height / 2);

bufferctx.fillText("VIDAS: " + player.life + " x 5", canvas.width / 2 - 200, canvas.height / 2 + 30);

bufferctx.fillText("PUNTUACION TOTAL: " + getTotalScore(), canvas.width / 2 - 200, canvas.height / 2 + 60);

}

function getTotalScore() {

return player.score + player.life \* 5;

}

function update() {

drawBackground();

if (congratulations) {

showCongratulations();

return; }

if (youLoose) {

showGameOver();

return; }

bufferctx.drawImage(player, player.posX, player.posY);

bufferctx.drawImage(evil.image, evil.posX, evil.posY);

updateEvil();

for (var j = 0; j < playerShotsBuffer.length; j++) {

var disparoBueno = playerShotsBuffer[j];

updatePlayerShot(disparoBueno, j); }

if (isEvilHittingPlayer()) {

player.killPlayer();

} else {

for (var i = 0; i < evilShotsBuffer.length; i++) {

var evilShot = evilShotsBuffer[i];

updateEvilShot(evilShot, i); } }

showLifeAndScore();

playerAction(); }

function updatePlayerShot(playerShot, id) {

if (playerShot) {

playerShot.identifier = id;

if (checkCollisions(playerShot)) {

if (playerShot.posY > 0) {

playerShot.posY -= playerShot.speed;

bufferctx.drawImage(playerShot.image, playerShot.posX, playerShot.posY);

} else {

playerShot.deleteShot(parseInt(playerShot.identifier));

} } } }

function updateEvilShot(evilShot, id) {

if (evilShot) {

evilShot.identifier = id;

if (!evilShot.isHittingPlayer()) {

if (evilShot.posY <= canvas.height) {

evilShot.posY += evilShot.speed;

bufferctx.drawImage(evilShot.image, evilShot.posX, evilShot.posY);

} else {

evilShot.deleteShot(parseInt(evilShot.identifier));

}

} else {

player.killPlayer();

} } }

function drawBackground() {

var background;

if (evil instanceof FinalBoss) {

background = bgBoss;

} else {

background = bgMain;

}

bufferctx.drawImage(background, 0, 0);

}

function updateEvil() {

if (!evil.dead) {

evil.update();

if (evil.isOutOfScreen()) {

evil.kill();

} } }

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MEJORES PUNTUACIONES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function saveFinalScore() {

localStorage.setItem(getFinalScoreDate(), getTotalScore());

showBestScores();

removeNoBestScores(); }

function getFinalScoreDate() {

var date = new Date();

return fillZero(date.getDay()+1)+'/'+

fillZero(date.getMonth()+1)+'/'+

date.getFullYear()+' '+

fillZero(date.getHours())+':'+

fillZero(date.getMinutes())+':'+

fillZero(date.getSeconds());

}

function fillZero(number) {

if (number < 10) {

return '0' + number;

}

return number; }

function getBestScoreKeys() {

var bestScores = getAllScores();

bestScores.sort(function (a, b) {return b - a;});

bestScores = bestScores.slice(0, totalBestScoresToShow);

var bestScoreKeys = [];

for (var j = 0; j < bestScores.length; j++) {

var score = bestScores[j];

for (var i = 0; i < localStorage.length; i++) {

var key = localStorage.key(i);

if (parseInt(localStorage.getItem(key)) == score) {

bestScoreKeys.push(key);

} } }

return bestScoreKeys.slice(0, totalBestScoresToShow);

}

function getAllScores() {

var all = [];

for (var i=0; i < localStorage.length; i++) {

all[i] = (localStorage.getItem(localStorage.key(i)));

}

return all;

}

function showBestScores() {

var bestScores = getBestScoreKeys();

var bestScoresList = document.getElementById('puntuaciones');

if (bestScoresList) {

clearList(bestScoresList);

for (var i=0; i < bestScores.length; i++) {

addListElement(bestScoresList, bestScores[i], i==0?'negrita':null);

addListElement(bestScoresList, localStorage.getItem(bestScores[i]), i==0?'negrita':null);

} } }

function clearList(list) {

list.innerHTML = '';

addListElement(list, "Fecha");

addListElement(list, "Puntos");

}

function addListElement(list, content, className) {

var element = document.createElement('li');

if (className) {

element.setAttribute("class", className); }

element.innerHTML = content;

list.appendChild(element); }

// extendemos el objeto array con un metodo "containsElement"

Array.prototype.containsElement = function(element) {

for (var i = 0; i < this.length; i++) {

if (this[i] == element) {

return true;

} }

return false; };

function removeNoBestScores() {

var scoresToRemove = [];

var bestScoreKeys = getBestScoreKeys();

for (var i=0; i < localStorage.length; i++) {

var key = localStorage.key(i);

if (!bestScoreKeys.containsElement(key)) {

scoresToRemove.push(key);

} }

for (var j = 0; j < scoresToRemove.length; j++) {

var scoreToRemoveKey = scoresToRemove[j];

localStorage.removeItem(scoreToRemoveKey);

} }

return {

init: init }

})();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN CSS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

body {

background: #ccc;

color: #333;

font: 14px 'Arial',sans-serif;

text-align: center;

}

a {

color: #506e2f;

text-decoration: none;

}

a:hover {

color: #ff4500;

}

div.contenedor {

width: 100%;

margin: 0 auto;

}

div.izquierda, div.derecha {

width: 300px;

}

div.izquierda {

float: left;

text-align: left;

}

div.juego {

}

div.juego canvas {

box-shadow: 0 0 30px #000;

}

div.derecha {

float: right;

}

div.derecha h3.tituloPuntuaciones {

background: url("../images/medalla.png") no-repeat left;

padding: 10px;

}

div.derecha ul#puntuaciones {

width: 300px;

margin: 0;

padding: 0;

list-style: none;

}

div.derecha ul#puntuaciones li{

line-height:1.5em;

border-bottom:1px solid #aaa;

float:left;

display:inline;

width: 50%;

padding: 10px 0;

}

.negrita {

font-weight: bold;

}

ul.lista {

margin: 25px 0;

padding: 0;

list-style: none;

text-align: left;

}

ul.lista li {

padding: 10px 0;

}

ul.lista li.firefox, ul.lista li.chrome, ul.lista li.safari, ul.lista li.izquierda, ul.lista li.derecha, ul.lista li.espacio {

padding-left: 40px;

}

ul.lista li.izquierda {

background: url("../images/izquierda.png") no-repeat left;

}

ul.lista li.derecha {

background: url("../images/derecha.png") no-repeat left;

}

ul.lista li.espacio {

background: url("../images/space.gif") no-repeat left;

}

ul.lista li.firefox {

background: url("../images/firefox.png") no-repeat left;

}

ul.lista li.chrome {

background: url("../images/chrome.png") no-repeat left;

}

ul.lista li.safari {

background: url("../images/safari.png") no-repeat left;

}

ul.lista li.titulo {

font-weight: bold;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RESET CSS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

html,body,div,span,applet,object,

iframe,h1,h2,h3,h4,h5,h6,p,blockquote,

pre,a,abbr,acronym,address,big,cite,

code,del,dfn,em,font,img,ins,kbd,q,s,

samp,small,strike,strong,sub,sup,tt,var,

b,u,i,center,dl,dt,dd,ol,ul,li,fieldset,form,

label,legend,table,caption,tbody,tfoot,thead,tr,th,td

{

margin:0;

padding:0;

border:0;

outline:0;

font-size:100%;

vertical-align:baseline;

}

body

{line-height:1}

ol,ul{list-style:none}

blockquote,q{quotes:none}

blockquote:before,

blockquote:after,

q:before,

q:after{content:'';content:none}:

focus{outline:0}ins{text-decoration:none}

del{text-decoration:line-through}

table{border-spacing:0}a{text-decoration: none}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JUEGO HTML\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Videojuego con Javascript y HTML5</title>

<link type="text/css" rel="stylesheet" href="css/main.css?14042012">

<script src="js/videojuego-javascript.js?04042012" type="text/javascript"></script>

</head>

<body onload="game.init()">

<div class="contenedor">

<div class="izquierda">

<p>

<a class="negrita">Videojuego con HTML5 y Javascript</a>

</p>

<p>

Rediseñado por <a> hebert de la cruz</a>

</p>

<ul class="lista">

<li class="titulo">CONTROLES:</li>

<li class="izquierda">izquierda</li>

<li class="derecha">derecha</li>

<li class="espacio">disparar</li>

</ul>

<ul class="lista">

<li class="titulo">PROBADO CON:</li>

<li class="firefox">Mozilla Firefox 11</li>

<li class="chrome">Google Chrome 18</li>

</ul>

<ul class="lista">

<li class="titulo">OTROS:</li>

<li><a href="http://www.adictosaltrabajo.com"\_blank">Tutorial</a></li>

</ul> </div>

<div class="derecha">

<h3 class="tituloPuntuaciones">MEJORES PUNTUACIONES</h3>

<ul id="puntuaciones"></ul>

</div>

<div class="juego">

<canvas id="canvas" width="600" height="650">TU NAVEGADOR NO SOPORTA HTML5!</canvas>

</div> </div>

</body>

</html>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*