***1.Documentación externa***

1.1 **Descripción del problema:** El programa tiene como función entretener al usuario por medio de un juego llamado “lluvia de asteroides”. Es juego interactivo en cual el usuario es una nave el cual debe afrontar, destruir y esquivar dichos cuerpos celestes que se mueven en aleatorio. Es un juego hecho en HTML, JS y CSS, por lo tanto, requiere una serie de funciones específicas para el comportamiento de cada variable (Nave, Partículas, Balas, Asteroides). Comportamiento que es esencial para la interactividad del mismo. Conforme se destruyan los asteroides, cada vez serán más rápidos. Para ver que tan buenos reflejos tiene el usuario para su entretenimiento ambientado a un área espacial con música emotiva que hace que se sienta la emoción de jugarlo.

1.2 **Nombre del autor:** Ingeniero Josué Emanuel Dionicio Orellana

1.3**: Programa:**



1.4 **Algoritmo:**

**INICIO**

Abrir la pagina

**NO**

Carga la pagina

**SI**

Buscar función del teclado

Busca Función de la Vec2D

Busca Función de las partículas

Busca Función de la bala

Busca Función del asteroide

Busca Función de la nave

Se observa el canvas

Pasar a la siguiente verificación

**No**

Refrescado del Canvas

Se observa la nave

**SI NO**

Refrescado del Canvas

Función del teclado

Generar partículas de avance

Se observa los asteroides

**SI NO**

Refrescado del Canvas

Generar asteroides

**SI NO**

Colisión con nave

Seguir

Generar Explosión

Generar partículas

Reinicio del juego

Función balas

**SI NO**

Generar balas

Refrescado del Canvas

**SI NO**

Colisión balas con asteroides

Refrescado del Canvas

Generar explosión

Hasta 2 veces

Generar partículas

Destruir Asteroide

Dividir Asteroide en 2

Velocidad aumentada asteroides

1.5 **Diccionario de datos:**

|  |  |
| --- | --- |
| *Etiquetas* | Descripción |
| ***HTML*** | |
| ***head*** | Provee información general del documento. |
| ***body*** | Contiene todos los aspectos de un documento. |
| ***table*** | Contiene celdas y columnas para la creación de tablas. |
| ***tr*** | Representa a las filas en cada tabla. |
| ***td*** | Representa a las celdas en cada tabla. |
| ***h1*** | Se utiliza comúnmente para las cabeceras del html. |
| ***button*** | Representa un elemento cliqueable de tipo botón. |
| ***Canvas*** | Permite la generación de gráficos en un HTML. |
| ***JavaScript*** | |
| ***if*** | Ejecuta una sentencia si una condición especificada es evaluada como verdadera en JS. |
| ***onload*** | Un evento tipo handler para el evento load event del window. |
| ***function*** | La función se ejecuta cuando "algo" lo invoca o llama. |
| ***window*** | Esta sección proporciona una pequeña referencia a todos los métodos. |
| ***setTimeout*** | El método setTimeout () llama a una función o evalúa una expresión después de un número específico de milisegundos. |
| ***array*** | Son objetos tipo-lista, cuyo prototipo tiene métodos para efectuar operaciones de recorrido y de mutación. |
| ***document*** | Se controla la página web y todos los elementos que contiene. |
| ***addEventListener*** | Registra un evento a un objeto en específico. |
| ***var*** | Declara una variable, opcionalmente inicializándola con un valor. |
| ***this*** | Atajo que refiere al objeto desde donde se invocó la función. |
| ***else*** | Si la condición es evaluada como falsa, otra sentencia puede ser ejecutada. |
| ***for*** | Crea un bucle que consiste en tres expresiones opcionales. |
| ***save*** | Guarda el estado completo del Canvas añadiendo el estado actual a una pila. |
| ***fillText*** | Dibuja texto relleno en el lienzo del Canvas. |
| ***return*** | Finaliza la ejecución de la función y especifica un valor para ser devuelto a quien llama a la función. |
| ***length*** | Representa la longitud de una cadena, en unidades de código UTF-16. |
| ***CSS*** | |
| ***display*** | Es la propiedad más importante para controlar estructuras. |
| ***margin*** | Establece el margen para los cuatro lados. |
| ***background-image*** | Establece una o más imágenes de fondo para un elemento. |
| ***background-size*** | Especifica el tamaño de las imágenes de fondo. |
| ***text-align*** | Especifica la alineación horizontal del texto en un elemento. |
| ***font-family*** | Especifica la fuente para un elemento. |
| ***text-decoration*** | Especifica la decoración agregada al texto |
| ***font-size*** | Especifica la dimensión de la letra. |
| ***background*** | Especifica el color de fondo de un elemento. |
| ***box-shadow:*** | Adjunta una o más sombras a un elemento. |
| ***right*** | Afecta la posición horizontal de un elemento posicionado. |
| ***left*** | Afecta la posición horizontal de un elemento posicionado. |
| ***border*** | Permite especificar el estilo, el ancho y el color del borde de un elemento. |
| ***transition*** | Permiten cambiar los valores de las propiedades sin problemas, durante una duración determinada. |

1.6 **Código fuente**

**HTML**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

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

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

    <script src="sistema.js"></script>

    <title>Juego de Asteroides</title>

</head>

<body background="fondo1.jpg" >

    <audio src="audio.mp4" autoplay loop></audio>

    <h1>Lluvia de asteroides</h1>

    <img src="nave2.png" alt="">

    <div id="info">

        <h2>Controles</h2><br><hr><br>

        <p>Utiliza los botones</p>

        <p>[A][S][W][D]</p>

        <p>[&larr;][&uarr;][&darr;][&rarr;]</p>

        <p>para moverte</p><br><hr> <br>

        <p>Usa la tecla</p>

        <p>[ESPACIO]</p>

        <p>[K]</p>

        <p>para disparar</p><br><hr><br>

        <h2>Tu Misión</h2>

        <p><strong>Sobrevive!</strong></p>

        <p>destruye todos los asteroides que puedas, mientras más asteroides destruyes más rápido vendran</p>

    </div>

    <canvas id="canvas"></canvas>

    <div id="salir">

        <p id= "p1">

          <button><a href="C:\Users\USUARIO\Desktop\Trabajo\_Colaborativo\Pagina Principal\index.html" class="btn-neon">

              <span id="span1"></span>

              <span id="span2"></span>

              <span id="span3"></span>

              <span id="span4"></span>

          Salir

          </a></button>

      </p>

      </div>

</body>

</html>

**CSS**

\* {

    margin: 0;

}

canvas {

    position: absolute;

    top: 70px;

    left: 300px;

    width: 800px;

    height: 500px;

    background: #262626;

}

#info {

    color: #FFF;

    font-family: "Lucida Sans Typewriter", "Lucida Console", Monaco, "Bitstream Vera Sans Mono", monospace;

    margin: 50px 10px 0;

    padding: 8px;

    background: #161616;

    position: absolute;

    z-index: 1;

    height: 485px;

    width: 260px;

}

#info p {

    font-size: 18px;

}

h1 {

font-size: 42px;

position: absolute;

color: white;

left: 445px;

font-family: "Lucida Sans Typewriter", "Lucida Console", Monaco, "Bitstream Vera Sans Mono", monospace;

}

img {

    position: absolute;

    height: 40px;

    width: 60px;

    left: 940px;

    top: 30px;

}

#salir p {

    position: relative;

    top: 510px;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 150px;

    width: center;

    background-size: auto;

    font-family: 'Courier New', Courier, monospace;

    margin: 20px;

    }

    .btn-neon{

    position: relative;

    display: inline-block;

    padding: 15px 30px;

    color: white;

    letter-spacing: 4px;

    text-decoration: none;

    font-size: 20px;

    overflow: hidden;

    transition: 0.5s;

    background-color: rgb(14, 241, 26);

    }

    .btn-neon:hover{

     background: gray;

     box-shadow: 0 0 10px gray, 0 0 40px gray, 0 0 80px gray;

     transition-delay: 1s;

     }

    .btn-neon span{

     position: absolute;

     display: block;

            }

     #span1{

     top: 0;

     left: -100%;

     width: 100%;

     height: 2px;

     background: linear-gradient(90deg, transparent,#a945c7);

     }

     .btn-neon:hover #span1{

     left: 100%;

     transition: 0.5s;

     }

     #span3{

     bottom: 0;

     right: -100%;

     width: 100%;

     height: 2px;

     background: linear-gradient(270deg, transparent,#a945c7);

     }

     .btn-neon:hover #span3{

      right: 100%;

      transition: 1s;

      transition-delay: 0.5s;

      }

      #span2{

      top: -100%;

      right: 0;

      width: 2px;

      height: 100%;

      background: linear-gradient(180deg, transparent,#a945c7);

      }

      .btn-neon:hover #span2{

      top: 100%;

      transition: 1s;

      transition-delay: 0.25s;

      }

      #span4{

      bottom: -100%;

      left: 0;

      width: 2px;

      height: 100%;

      background: linear-gradient(360deg, transparent,#a945c7);

      }

      .btn-neon:hover #span4{

      bottom: 100%;

      transition: 1s;

      transition-delay: 0.75s;}

**JAVASCRIPT**

var Pool = (function()

{

    //metodos expuestos:

    var create = function(type, size)

    {

        var obj = Object.create(def);

        obj.init(type, size);

        return obj;

    };

    //Definicion:

    var def =

    {

        \_type: null,

        \_size: null,

        \_pointer: null,

        \_elements:null,

        init: function(type, size)

        {

            this.\_type = type;

            this.\_size = size;

            this.\_pointer = size;

            this.\_elements = [];

            var i = 0;

            var length = this.\_size;

            for(i; i < length; ++i)

            {

                this.\_elements[i] = this.\_type.create();

            }

        },

        getElement: function()

        {

            if(this.\_pointer > 0) return this.\_elements[--this.\_pointer];

            return null;

        },

        disposeElement: function(obj)

        {

            this.\_elements[this.\_pointer++] = obj;

        }

    };

    return {create:create};

}());

//vec2d.js ...........................................................

var Vec2D = (function()

{

    //metodos expuestos:

    var create = function(x, y)

    {

        var obj = Object.create(def);

        obj.setXY(x, y);

        return obj;

    };

    //definicion del Vec2D:

    var def =

    {

        \_x: 1,

        \_y: 0,

        getX: function()

        {

            return this.\_x;

        },

        setX: function(value)

        {

            this.\_x = value;

        },

        getY: function()

        {

            return this.\_y;

        },

        setY: function(value)

        {

            this.\_y = value;

        },

        setXY: function(x, y)

        {

            this.\_x = x;

            this.\_y = y;

        },

        getLength: function()

        {

            return Math.sqrt(this.\_x \* this.\_x + this.\_y \* this.\_y);

        },

        setLength: function(length)

        {

            var angle = this.getAngle();

            this.\_x = Math.cos(angle) \* length;

            this.\_y = Math.sin(angle) \* length;

        },

        getAngle: function()

        {

            return Math.atan2(this.\_y, this.\_x);

        },

        setAngle: function(angle)

        {

            var length = this.getLength();

            this.\_x = Math.cos(angle) \* length;

            this.\_y = Math.sin(angle) \* length;

        },

        add: function(vector)

        {

            this.\_x += vector.getX();

            this.\_y += vector.getY();

        },

        sub: function(vector)

        {

            this.\_x -= vector.getX();

            this.\_y -= vector.getY();

        },

        mul: function(value)

        {

            this.\_x \*= value;

            this.\_y \*= value;

        },

        div: function(value)

        {

            this.\_x /= value;

            this.\_y /= value;

        }

    };

    return {create:create};

}());

//particulas ...........................................................

var Particle = (function()

{

    //metodos Expuestos:

    var create = function()

    {

        var obj = Object.create(def);

        obj.radius = 2;

        obj.color = '#FFF';

        obj.lifeSpan = 0;

        obj.fric = 0.98;

        obj.pos = Vec2D.create(0, 0);

        obj.vel = Vec2D.create(0, 0);

        obj.blacklisted = false;

        return obj;

    };

    //Definicion:

    var def =

    {

        radius: null,

        color: null,

        lifeSpan: null,

        fric: null,

        pos: null,

        vel: null,

        blacklisted: null,

        update: function()

        {

            this.pos.add(this.vel);

            this.vel.mul(this.fric);

            this.radius -= 0.1;

            if(this.radius < 0.1) this.radius = 0.1;

            if(this.lifeSpan-- < 0)

            {

                this.blacklisted = true;

            }

        },

        reset: function()

        {

            this.blacklisted = false;

        }

    };

    return {create:create};

}());

//bala ...........................................................

var Bullet = (function()

{

    //metodos expuestos:

    var create = function()

    {

        var obj = Object.create(def);

        obj.radius = 4;

        obj.color = '#fff';

        obj.pos = Vec2D.create(0, 0);

        obj.vel = Vec2D.create(0, 0);

        obj.blacklisted = false;

        return obj;

    };

    //definicion:

    var def =

    {

        radius: null,

        color: null,

        pos: null,

        vel: null,

        blacklisted: null,

        update: function()

        {

            this.pos.add(this.vel);

        },

        reset: function()

        {

            this.blacklisted = false;

        }

    };

    return {create:create};

}());

//asteroides.........................................................

var Asteroid = (function()

{

    var create = function()

    {

        var obj = Object.create(def);

        obj.radius = 40;

        obj.color = '#07ff3c';   ////

        obj.pos = Vec2D.create(0, 0);

        obj.vel = Vec2D.create(0, 0);

        obj.blacklisted = false;

        obj.type = 'b';

        obj.sides = (Math.random() \* 2 + 50) >> 0;  //////

        obj.angle = 0;

        obj.angleVel = (1 - Math.random() \* 2) \* 0.01;

        return obj;

    };

    //definicion:

    var def =

    {

        radius: null,

        color: null,

        pos: null,

        vel: null,

        blacklisted: null,

        type: null,

        sides: null,

        angle: null,

        angleVel: null,

        update: function()

        {

            this.pos.add(this.vel);

            this.angle += this.angleVel;

        },

        reset: function()

        {

            this.blacklisted = false;

        }

    };

    return {create:create};

}());

//nave...........................................................

var Ship = (function()

{

    //metodos expuestos:

    var create = function(x, y, ref)

    {

        var obj = Object.create(def);

        obj.ref = ref;

        obj.angle = 0;

        obj.pos = Vec2D.create(x, y);

        obj.vel = Vec2D.create(0, 0);

        obj.thrust = Vec2D.create(0, 0);

        obj.idle = false;

        obj.radius = 8;

        obj.idleDelay = 0;

        return obj;

    };

    //Definicion:

    var def =

    {

        angle: null,

        pos: null,

        vel: null,

        thrust: null,

        ref: null,

        bulletDelay: null,

        idle: null,

        radius: null,

        update: function()

        {

            this.vel.add(this.thrust);

            this.pos.add(this.vel);

            if(this.vel.getLength() > 5) this.vel.setLength(5);

            ++this.bulletDelay;

            if(this.idle)

            {

                if(++this.idleDelay > 120)

                {

                    this.idleDelay = 0;

                    this.idle = false;

                    this.ref.resetGame();

                }

            }

        },

        shoot: function()

        {

            if(this.bulletDelay > 8)

            {

                this.ref.generateShot();

                this.bulletDelay = 0;

            }

        }

    };

    return {create:create};

}());

//canvas ...........................................................

//variables comunes

var canvas;

var context;

var screenWidth;

var screenHeight;

var doublePI = Math.PI \* 2;

//variables del juego

var ship;

var particlePool;

var particles;

var bulletPool;

var bullets;

var asteroidPool;

var asteroids;

var hScan;

var asteroidVelFactor = 0;

//variables del teclado

var keyLeft = false;

var keyUp = false;

var keyRight = false;

var keyDown = false;

var keySpace = false;

window.getAnimationFrame =

window.requestAnimationFrame ||

window.webkitRequestAnimationFrame ||

window.mozRequestAnimationFrame ||

window.oRequestAnimationFrame ||

window.msRequestAnimationFrame ||

function(callback)

{

    window.setTimeout(callback, 16.6);

};

window.onload = function()

{

    canvas = document.getElementById('canvas');

    context = canvas.getContext('2d');

    window.onresize();

    keyboardInit();

    particleInit();

    bulletInit();

    asteroidInit();

    shipInit();

    loop();

};

window.onresize = function()

{

    if(!canvas) return;

    screenWidth = canvas.clientWidth;

    screenHeight = canvas.clientHeight;

    canvas.width = screenWidth;

    canvas.height = screenHeight;

    hScan = (screenHeight / 4) >> 0;

};

function keyboardInit()

{

    window.onkeydown = function(e)

    {

        switch(e.keyCode)

        {

            //tecla A o Izquierda

            case 65:

            case 37:

            keyLeft = true;

            break;

            //tecla W o Arriba

            case 87:

            case 38:

            keyUp = true;

            break;

            //tecla D o derecha

            case 68:

            case 39:

            keyRight = true;

            break;

            //tecla S o abajo

            case 83:

            case 40:

            keyDown = true;

            break;

            //tecla espacio

            case 32:

       case 75:

            keySpace = true;

            break;

        }

    e.preventDefault();

    };

    window.onkeyup = function(e)

    {

        switch(e.keyCode)

        {

            //tecla A o Izquierda

            case 65:

            case 37:

            keyLeft = false;

            break;

            //tecla W o Arriba

            case 87:

            case 38:

            keyUp = false;

            break;

            //tecla D o Derecha

            case 68:

            case 39:

            keyRight = false;

            break;

            //tecla S o Abajo

            case 83:

            case 40:

            keyDown = false;

            break;

            //tecla espacio

       case 75:

            case 32:

            keySpace = false;

            break;

        }

    e.preventDefault();

    };

}

function particleInit()

{

    particlePool = Pool.create(Particle, 100);

    particles = [];

}

function bulletInit()

{

    bulletPool = Pool.create(Bullet, 40);

    bullets = [];

}

function asteroidInit()

{

    asteroidPool = Pool.create(Asteroid, 30);

    asteroids = [];

}

function shipInit()

{

    ship = Ship.create(screenWidth >> 1, screenHeight >> 1, this);

}

function loop()

{

    updateShip();

    updateParticles();

    updateBullets();

    updateAsteroids();

    checkCollisions();

    render();

    getAnimationFrame(loop);

}

function updateShip()

{

    ship.update();

    if(ship.idle) return;

    if(keySpace) ship.shoot();

    if(keyLeft) ship.angle -= 0.1;

    if(keyRight) ship.angle += 0.1;

    if(keyUp)

    {

        ship.thrust.setLength(0.1);

        ship.thrust.setAngle(ship.angle);

        generateThrustParticle();

    }

    else

    {

        ship.vel.mul(0.94);

        ship.thrust.setLength(0);

    }

    if(ship.pos.getX() > screenWidth) ship.pos.setX(0);

    else if(ship.pos.getX() < 0) ship.pos.setX(screenWidth);

    if(ship.pos.getY() > screenHeight) ship.pos.setY(0);

    else if(ship.pos.getY() < 0) ship.pos.setY(screenHeight);

}

function generateThrustParticle()

{

    var p = particlePool.getElement();

    //si el grupo de partículas no tiene más elementos, devolverá 'nulo'.

    if(!p) return;

    p.radius = Math.random() \* 3 + 2;

    p.color = '#FFF';

    p.lifeSpan = 80;

    p.pos.setXY(ship.pos.getX() + Math.cos(ship.angle) \* -14, ship.pos.getY() + Math.sin(ship.angle) \* -14);

    p.vel.setLength(8 / p.radius);

    p.vel.setAngle(ship.angle + (1 - Math.random() \* 2) \* (Math.PI / 18));

    p.vel.mul(-1);

    particles[particles.length] = p;

}

function updateParticles()

{

    var i = particles.length - 1;

    for(i; i > -1; --i)

    {

        var p = particles[i];

        if(p.blacklisted)

        {

            p.reset();

            particles.splice(particles.indexOf(p), 1);

            particlePool.disposeElement(p);

            continue;

        }

        p.update();

    }

}

function updateBullets()

{

    var i = bullets.length - 1;

    for(i; i > -1; --i)

    {

        var b = bullets[i];

        if(b.blacklisted)

        {

            b.reset();

            bullets.splice(bullets.indexOf(b), 1);

            bulletPool.disposeElement(b);

            continue;

        }

        b.update();

        if(b.pos.getX() > screenWidth) b.blacklisted = true;

        else if(b.pos.getX() < 0) b.blacklisted = true;

        if(b.pos.getY() > screenHeight) b.blacklisted = true;

        else if(b.pos.getY() < 0) b.blacklisted = true;

    }

}

function updateAsteroids()

{

    var i = asteroids.length - 1;

    for(i; i > -1; --i)

    {

        var a = asteroids[i];

        if(a.blacklisted)

        {

            a.reset();

            asteroids.splice(asteroids.indexOf(a), 1);

            asteroidPool.disposeElement(a);

            continue;

        }

        a.update();

        if(a.pos.getX() > screenWidth + a.radius) a.pos.setX(-a.radius);

        else if(a.pos.getX() < -a.radius) a.pos.setX(screenWidth + a.radius);

        if(a.pos.getY() > screenHeight + a.radius) a.pos.setY(-a.radius);

        else if(a.pos.getY() < -a.radius) a.pos.setY(screenHeight + a.radius);

    }

    if(asteroids.length < 10)

    {

        var factor = (Math.random() \* 2) >> 0;

        generateAsteroid(screenWidth \* factor, screenHeight \* factor, 60 , 'b');

    }

}

function generateAsteroid(x, y, radius, type)

{

    var a = asteroidPool.getElement();

    //Si el grupo de viñetas no tiene más elementos, devolverá 'nulo'.

    if(!a) return;

    a.radius = radius;

    a.type = type;

    a.pos.setXY(x, y);

    a.vel.setLength(1 + asteroidVelFactor);

    a.vel.setAngle(Math.random() \* (Math.PI \* 2));

    asteroids[asteroids.length] = a;

    asteroidVelFactor += 0.025;

}

function checkCollisions()

{

    checkBulletAsteroidCollisions();

    checkShipAsteroidCollisions();

}

function checkBulletAsteroidCollisions()

{

    var i = bullets.length - 1;

    var j;

    for(i; i > -1; --i)

    {

        j = asteroids.length - 1;

        for(j; j > -1; --j)

        {

            var b = bullets[i];

            var a = asteroids[j];

            if(checkDistanceCollision(b, a))

            {

                b.blacklisted = true;

                destroyAsteroid(a);

            }

        }

    }

}

function checkShipAsteroidCollisions()

{

    var i = asteroids.length - 1;

    for(i; i > -1; --i)

    {

        var a = asteroids[i];

        var s = ship;

        if(checkDistanceCollision(a, s))

        {

            if(s.idle) return;

            s.idle = true;

            generateShipExplosion();

            destroyAsteroid(a);

        }

    }

}

function generateShipExplosion()

{

    var i = 18;

    for(i; i > -1; --i)

    {

        var p = particlePool.getElement();

        //si el grupo de partículas no tiene más elementos, devolverá 'nulo'.

        if(!p) return;

        p.radius = Math.random() \* 6 + 2;

        p.lifeSpan = 80;

        p.color = '#FFF';

        p.vel.setLength(20 / p.radius);

        p.vel.setAngle(ship.angle + (1 - Math.random() \* 2) \* doublePI);

        p.pos.setXY(ship.pos.getX() + Math.cos(p.vel.getAngle()) \* (ship.radius \* 0.8), ship.pos.getY() + Math.sin(p.vel.getAngle()) \* (ship.radius \* 0.8));

        particles[particles.length] = p;

    }

}

function checkDistanceCollision(obj1, obj2)

{

    var vx = obj1.pos.getX() - obj2.pos.getX();

    var vy = obj1.pos.getY() - obj2.pos.getY();

    var vec = Vec2D.create(vx, vy);

    if(vec.getLength() < obj1.radius + obj2.radius)

    {

        return true;

    }

    return false;

}

function destroyAsteroid(asteroid)

{

    asteroid.blacklisted = true;

    generateAsteroidExplosion(asteroid);

    resolveAsteroidType(asteroid);

}

function generateAsteroidExplosion(asteroid)

{

    var i = 18;

    for(i; i > -1; --i)

    {

        var p = particlePool.getElement();

        //si el grupo de partículas no tiene más elementos, devolverá 'nulo'.

        if(!p) return;

        p.radius = Math.random() \* (asteroid.radius >> 2) + 2;

        p.lifeSpan = 80;

        p.color = '#07ff3c';

        p.vel.setLength(20 / p.radius);

        p.vel.setAngle(ship.angle + (1 - Math.random() \* 5) \* doublePI);

        p.pos.setXY(asteroid.pos.getX() + Math.cos(p.vel.getAngle()) \* (asteroid.radius \* 0.10), asteroid.pos.getY() + Math.sin(p.vel.getAngle()) \* (asteroid.radius \* 0.8));

        particles[particles.length] = p;

    }

}

function resolveAsteroidType(asteroid)

{

    switch(asteroid.type)

    {

        case 'b':

        generateAsteroid(asteroid.pos.getX(), asteroid.pos.getY(), 40, 'm');

        generateAsteroid(asteroid.pos.getX(), asteroid.pos.getY(), 40, 'm');

        break;

        case 'm':

        generateAsteroid(asteroid.pos.getX(), asteroid.pos.getY(), 20, 's');

        generateAsteroid(asteroid.pos.getX(), asteroid.pos.getY(), 20, 's');

        break;

    }

}

function render()

{

    context.fillStyle = '#262626';

    context.globalAlpha = 0.4;

    context.fillRect(0, 0, screenWidth, screenHeight);

    context.globalAlpha = 1;

    renderShip();

    renderParticles();

    renderBullets();

    renderAsteroids();

    renderScanlines();

}

function renderShip()

{

    if(ship.idle) return;

    context.save();

    context.translate(ship.pos.getX() >> 0, ship.pos.getY() >> 0);

    context.rotate(ship.angle);

    context.strokeStyle = '#FFF';

    context.lineWidth = (Math.random() > 0.9) ? 2 : 1;

    context.beginPath();

    context.moveTo(10, 0);

    context.lineTo(-10, -10);

    context.lineTo(-10, 10);

    context.lineTo(10, 0);

    context.stroke();

    context.closePath();

    context.restore();

}

function renderParticles()

{

    //inverso para loop = más rendimiento.

    var i = particles.length - 1;

    for(i; i > -1; --i)

    {

        var p = particles[i];

        context.beginPath();

        context.strokeStyle = p.color;

        context.arc(p.pos.getX() >> 0, p.pos.getY() >> 0, p.radius, 0, doublePI);

        if(Math.random() > 0.4) context.stroke();

        context.closePath();

    }

}

function renderBullets()

{

    //inverso para loop = más rendimiento.

    var i = bullets.length - 1;

    for(i; i > -1; --i)

    {

        var b = bullets[i];

        context.beginPath();

        context.strokeStyle = b.color;

        context.arc(b.pos.getX() >> 0, b.pos.getY() >> 0, b.radius, 0, doublePI);

        if(Math.random() > 0.2) context.stroke();

        context.closePath();

    }

}

function renderAsteroids()

{

    //inverso para loop = más rendimiento.

    var i = asteroids.length - 1;

    for(i; i > -1; --i)

    {

        var a = asteroids[i];

        context.beginPath();

        context.lineWidth = (Math.random() > 0.2) ? 4 : 3;

        context.strokeStyle = a.color;

        var j = a.sides;

        context.moveTo((a.pos.getX() + Math.cos(doublePI \* (j / a.sides) + a.angle) \* a.radius) >> 0, (a.pos.getY() + Math.sin(doublePI \* (j / a.sides) + a.angle) \* a.radius) >> 0);

        for(j; j > -1; --j)

        {

            context.lineTo((a.pos.getX() + Math.cos(doublePI \* (j / a.sides) + a.angle) \* a.radius) >> 0, (a.pos.getY() + Math.sin(doublePI \* (j / a.sides) + a.angle) \* a.radius) >> 0);

        }

        if(Math.random() > 0.2) context.stroke();

        context.closePath();

    }

}

function renderScanlines()

{

    //inverso para loop = más rendimiento.

    var i = hScan;

    context.globalAlpha = 0.05;

    context.lineWidth = 1;

    for(i; i > -1; --i)

    {

        context.beginPath();

        context.moveTo(0, i \* 4);

        context.lineTo(screenWidth, i \* 4);

        context.strokeStyle = (Math.random() > 0.0001) ? '#FFF' : '#222';

        context.stroke();

    }

    context.globalAlpha = 1;

}

function generateShot()

{

    var b = bulletPool.getElement();

// si el conjunto de viñetas no tiene más elementos, devolverá 'nulo'.

    if(!b) return;

    b.radius = 1;

    b.pos.setXY(ship.pos.getX() + Math.cos(ship.angle) \* 14, ship.pos.getY() + Math.sin(ship.angle) \* 14);

    b.vel.setLength(10);

    b.vel.setAngle(ship.angle);

    bullets[bullets.length] = b;

}

function resetGame()

{

    asteroidVelFactor = 0;

    ship.pos.setXY(screenWidth >> 1, screenHeight >> 1);

    ship.vel.setXY(0, 0);

    resetAsteroids();

}

function resetAsteroids()

{

    var i = asteroids.length - 1;

    for(i; i > -1; --i)

    {

        var a = asteroids[i];

        a.blacklisted = true;

    }

}

***2.Manual del usuario***

Este manual tiene como finalidad dar a conocer al usuario la forma en que se utiliza el juego de la página llamada “Juego de Asteroides”.

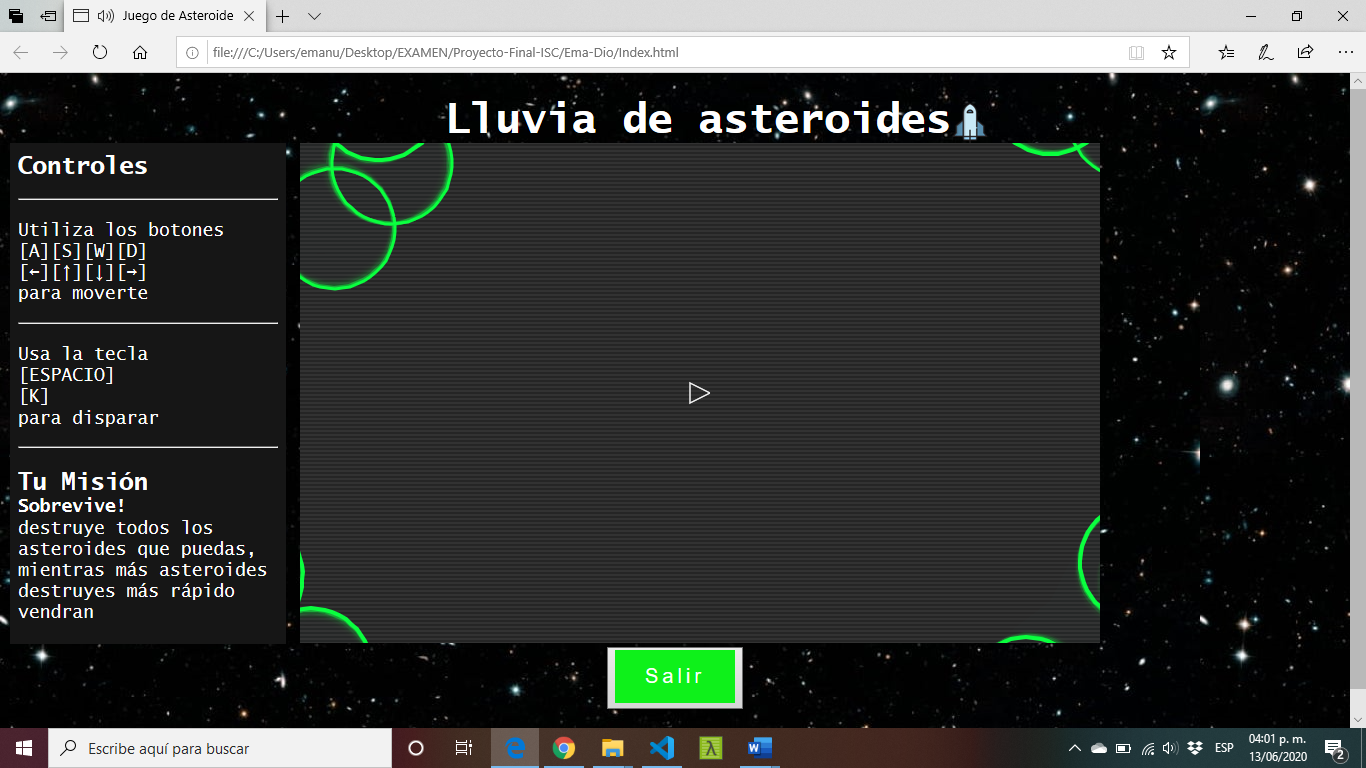
**2.1 Requerimiento técnico para la utilización de la página.**

* Navegador Internet Explorer 2019 o cualquier navegador actualizado al 2019 o superior.
* Tener cuenta de Google (para el navegador).

**2.2 Acceso a la página.** El ingreso al programa se realiza desde esta dirección electrónica desde el navegador de preferencia.

<file:///C:/Users/emanu/Desktop/EXAMEN/Proyecto-Final-ISC/Ema-Dio/Index.html>

A continuación, se muestra esta página.



Una vez dentro de la pagina puede leer las instrucciones en la parte izquierda de la pantalla (resaltado en un cuadro rojo).



Una vez leído ciertas instrucciones y aprenderse los controles puede jugar libremente en el juego. Si algún asteroide representado como un circulo color verde toca la nave representado con un triangulo color blanco el mismo explota y se reinicia el juego.



Para eso tiene un defensa incorporado, la cual son balas infinitas las cuales puede utilizar para destruir cada asteroide. El asteroide una vez que le impacte la bala se dividirá en 2 partes de menor tamaño y lo mismo a cada uno de ellos llegando a salir 4 asteroides pequeños de 2 asteroides medianos de 1 asteroide grande.



Llegando al punto en que quiera salir de la pagina haga clic en el botón color verde ubicado en la parte inferior de en medio llamado “salir” la cual se ilumina una vez puesto el mouse encima de él (resaltado en un cuadro rojo).

