Adrián Jiménez Villarreal

71904816P | uo227602

Videojuegos-Infinite

Ampliaciones

Contenido

[Ampliaciones 2](#_Toc468658677)

[Varias vidas 2](#_Toc468658678)

[Modificación del Jugador 2](#_Toc468658679)

[Modificación de los controles 2](#_Toc468658680)

[Modificación de la escena de juego 2](#_Toc468658681)

[Scroll en el eje Y 2](#_Toc468658682)

[Modificación de la escena de juego 3](#_Toc468658683)

[Capacidad de disparo 3](#_Toc468658684)

[Nuevo tipo de enemigo 3](#_Toc468658685)

[Turbo 3](#_Toc468658686)

[Pinchos 4](#_Toc468658687)

[Niveles 4](#_Toc468658688)

[Control el modo plataformas 4](#_Toc468658689)

# Ampliaciones

## Varias vidas

Para realizar esta ampliación primero deberemos modificar el objeto Jugador, posteriormente la capa de controles y finalmente la escena del juego.

### Modificación del Jugador

Añadir una variable vidas inicializada a 3 y la siguiente función.

|  |
| --- |
| }, restaVida: **function**(){  **this**.vidas--; } |

### Modificación de los controles

Habrá que crear una etiqueta para las vidas, e inicializarla de la siguiente manera.

|  |
| --- |
| *// Contador Vidas* **this**.etiquetaVidas = **new** cc.LabelTTF(**"Vidas: 3"**, **"Helvetica"**, 20); **this**.etiquetaVidas.setPosition(cc.p(size.width - size.width+90, size.height - 20)); **this**.etiquetaVidas.fillStyle = **new** cc.Color(0, 0, 0, 0); **this**.addChild(**this**.etiquetaVidas); |

También habrá que crear un método para actualizar esta etiqueta cada vez que pierda una vida el jugador.

|  |
| --- |
| },actualizarVidas:**function**(vidas){  **this**.etiquetaVidas.setString(**"Vidas: "** + vidas); } |

### Modificación de la escena de juego

Habrá que modificar la colisión del jugador con enemigos, para que pierda vidas cuando chocas con ellos.

|  |
| --- |
| ,collisionJugadorConEnemigo:**function** (arbiter, space) {  **var** capaControles = **this**.getParent().getChildByTag(idCapaControles);  **this**.jugador.restaVida();  **var** shapes = arbiter.getShapes();  **this**.formasEliminar.push(shapes[1]);  **if**(**this**.jugador.vidas<=0){  cc.director.pause();  cc.director.runScene(**new** GameOverLayer());  }  capaControles.actualizarVidas(**this**.jugador.vidas); } |

## Scroll en el eje Y

Para realizar esta ampliación simplemente deberemos modificar como se mueve la cámara, no solo en el eje x sino también en él y.

### Modificación de la escena de juego

En el método update se controlará la cámara de la siguiente manera, así seguirá al jugador tanto en el eje horizontal como el vertical.

|  |
| --- |
| *// Ampliacion Scroll eje Y* **var** posicionXCamara = **this**.jugador.body.p.x - **this**.getContentSize().width/2; **var** posicionYCamara = **this**.jugador.body.p.y - **this**.getContentSize().height/2;  **if** ( posicionXCamara < 0 ){  posicionXCamara = 0; } **if** ( posicionXCamara > **this**.mapaAncho - **this**.getContentSize().width ){  posicionXCamara = **this**.mapaAncho - **this**.getContentSize().width; }  **if** ( posicionYCamara < 0 ){  posicionYCamara = 0; } **if** ( posicionYCamara > **this**.mapaAlto - **this**.getContentSize().height ){  posicionYCamara = **this**.mapaAlto - **this**.getContentSize().height ; }  **this**.setPosition(cc.p( - posicionXCamara , - posicionYCamara)); |

## Capacidad de disparo

Para poder realizar esta ampliación necesitaremos crear una clase Disparo, modificar la capa de controles y añadir colisiones extra para los disparos con los enemigos y el suelo.

### Creación Disparo

|  |
| --- |
| **var** Disparo = cc.Class.extend({  space:**null**,  sprite:**null**,  body:**null**,  shape:**null**,  layer:**null**, ctor:**function** (space, posicion, layer) {  **this**.space = space;  **this**.layer = layer;   *// Crear animacion* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 4; i++) {  **var** str = **"disparo\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **var** actionAnimacionBucle =  **new** cc.RepeatForever(**new** cc.Animate(animacion));   *// Crear Sprite - Cuerpo y forma* **this**.sprite = **new** cc.PhysicsSprite(**"#disparo\_01.png"**);  *// Cuerpo estÃ¡tica , no le afectan las fuerzas* **this**.body = **new** cp.Body(5, Infinity);   **this**.body.setPos(posicion);  **this**.body.setAngle(0);  **this**.sprite.setBody(**this**.body);  *// Se inserta el cuerpo al espacio* **this**.space.addBody(**this**.body);   *// forma* **this**.shape = **new** cp.BoxShape(**this**.body,  **this**.sprite.getContentSize().width,  **this**.sprite.getContentSize().height);  *// agregar forma dinamica* **this**.shape.setSensor(**true**);  **this**.space.addShape(**this**.shape);  **this**.shape.setCollisionType(tipoDisparo);  *// aÃ±adir sprite a la capa   // ejecutar la animaciÃ³n* **this**.sprite.runAction(actionAnimacionBucle);   layer.addChild(**this**.sprite,10);    }, eliminar: **function** (){  *// quita la forma* **this**.space.removeShape(**this**.shape);   *// quita el cuerpo* **this**.space.removeBody(**this**.shape.getBody());   *// quita el sprite* **this**.layer.removeChild(**this**.sprite);  } }); |

### Modificación ControlesLayer

Habrá que añadir el botón y después controlar si se pulsa, además de añadir un tiempo entre disparo y disparo.

Insercion del botón.

|  |
| --- |
| *// BotonDisparo* **this**.spriteBotonDisparo = cc.Sprite.create(res.boton\_disparar\_png);  **this**.spriteBotonDisparo.setPosition(  cc.p(size.width\*0.8, size.height\*0.2));   **this**.addChild(**this**.spriteBotonDisparo); |

Control boton

|  |
| --- |
| **if** (cc.rectContainsPoint(areaDisparo,  cc.p(event.getLocationX(), event.getLocationY()))  && **new** Date().getTime() - instancia.tiempoDisparar > 1000 ){  instancia.tiempoDisparar = **new** Date().getTime();  **var** gameLayer = instancia.getParent().getChildByTag(idCapaJuego);  **var** disparo = **new** Disparo(gameLayer.space,  cc.p(gameLayer.jugador.body.p.x, gameLayer.jugador.body.p.y),  gameLayer);  disparo.body.vx = 600;  gameLayer.disparos.push(disparo); } |

### Modificación GameScene

Habrá que añadir las nuevas colisiones y tratar los objetos que se deben eliminar como se tratan los demás.

### Gestión nuevas colisiones

En la llamada se tratará después de la colisión no antes.

|  |
| --- |
| },colisionDisparoConEnemigo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();   **this**.formasEliminar.push(shapes[1]);  **this**.formasEliminar.push(shapes[0]); },colisionDisparoConSuelo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();   **this**.formasEliminar.push(shapes[0]); } |

## Nuevo tipo de enemigo

Para realizar esta ampliación se ha usado el mismo sprite del cuervo, simplemente que esta vez se mueve entre unas barreras y no puede pasar de ahí, además este dispara. Además, se ha modificado la clase disparo creada anteriormente para que se creen los disparos de tipo enemigo o jugador, para tratar las colisiones.

### Creación del nuevo enemigo

|  |
| --- |
| **var** NuevoEnemigo = cc.Class.extend({  direccion: **"derecha"**,  space:**null**,  sprite:**null**,  shape:**null**,  layer:**null**,  tiempoDisparo:0, ctor:**function** (space, posicion, layer) {  **this**.space = space;  **this**.layer = layer;   *// Crear animación* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 8; i++) {  **var** str = **"cuervo"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **var** actionAnimacionBucle =  **new** cc.RepeatForever(**new** cc.Animate(animacion));   *// Crear Sprite - Cuerpo y forma* **this**.sprite = **new** cc.PhysicsSprite(**"#cuervo1.png"**);  *// Cuerpo estática , no le afectan las fuerzas  // Cuerpo dinámico, SI le afectan las fuerzas* **this**.body = **new** cp.Body(5, Infinity);   **this**.body.setPos(posicion);  **this**.body.setAngle(0);  **this**.sprite.setBody(**this**.body);  *// Se añade el cuerpo al espacio* **this**.space.addBody(**this**.body);   *// forma* **this**.shape = **new** cp.BoxShape(**this**.body,  **this**.sprite.getContentSize().width,  **this**.sprite.getContentSize().height);  *// agregar forma dinamica* **this**.space.addShape(**this**.shape);  **this**.shape.setCollisionType(tipoEnemigo);  *// añadir sprite a la capa   // ejecutar la animación* **this**.sprite.runAction(actionAnimacionBucle);   layer.addChild(**this**.sprite,10);   }, moverAutomaticamente: **function**(){  *// invertir direccion   // Velocidad baja ha colisionado con algo,* **if** ( **this**.body.vx < 3 && **this**.body.vx > -3 ) {  **if** (**this**.direccion == **"derecha"**){  **this**.direccion = **"izquierda"**;  **this**.body.p.x = **this**.body.p.x -10; *// Para que salga de la colisión* **this**.sprite.scaleX = 1;  } **else** {  **this**.direccion = **"derecha"**;  **this**.body.p.x = **this**.body.p.x + 10; *// Para que salga de la zona de colisión* **this**.sprite.scaleX = -1;  }  }   *// Dar impulsos para mantener la velocidad* **if** (**this**.direccion == **"izquierda"** && **this**.body.vx > -100){  **this**.body.applyImpulse(cp.v(-100, 250), cp.v(0, 0));  }  **if** (**this**.direccion == **"derecha"** && **this**.body.vx < 100){  **this**.body.applyImpulse(cp.v(100, 250), cp.v(0, 0));  }     } , eliminar: **function** (){  *// quita la forma* **this**.space.removeShape(**this**.shape);   *// quita el cuerpo* **this**.space.removeBody(**this**.shape.getBody());   *// quita el sprite* **this**.layer.removeChild(**this**.sprite);  }, disparar: **function**(){  **if**(**new** Date().getTime() - **this**.tiempoDisparo > 1000 ){  **this**.tiempoDisparo = **new** Date().getTime();  **var** disparo = **new** Disparo(**this**.space,  cc.p(**this**.body.p.x, **this**.body.p.y),  **this**.layer,**true**);  **if**(**this**.direccion==**"derecha"**)  disparo.body.vx = 350;  **else** disparo.body.vx = -350;  **this**.layer.disparos.push(disparo);  }  } }); |

### Modificación del GameScene

Para empezar, se han creado las colisiones siguientes

|  |
| --- |
| },colisionEnemigoConContencion:**function**(arbiter,space){  **var** shapes = arbiter.getShapes();  *// shapes[0] es el enemigo* **var** formaEnemigo = shapes[0];  formaEnemigo.body.vx = 0; *// Parar enemigo* },colisionDisparoConJugador:**function** (arbiter, space) {  **var** capaControles = **this**.getParent().getChildByTag(idCapaControles);  **this**.jugador.restaVida();  **var** shapes = arbiter.getShapes();  **this**.formasEliminar.push(shapes[0]);  **if**(**this**.jugador.vidas<=0){  cc.director.pause();  cc.director.runScene(**new** GameOverLayer());  }  capaControles.actualizarVidas(**this**.jugador.vidas); } |

Y ya posteriormente se ha llamado a los métodos que tienen los nuevos enemigos en el update.

|  |
| --- |
| *// Mover enemigos:* **for**(**var** i = 0; i < **this**.nuevosEnemigos.length; i++) {  **var** enemigo = **this**.nuevosEnemigos[i];  enemigo.moverAutomaticamente();  enemigo.disparar(); } |

Con todo esto tendríamos enemigos que disparasen y se moviesen de izquierda a derecha en un entorno limitado.

## Turbo

Para realizar esta ampliación se necesita insertar un botón de turbo en la capa de controles, controlar la pulsación sobre este y una función en el jugador que le dé un impulso.

### Inclusión del botón

|  |
| --- |
| *// BotonTurbo* **this**.spriteBotonTurbo = cc.Sprite.create(res.boton\_turbo\_png); **this**.spriteBotonTurbo.setPosition(  cc.p(size.width\*0.8, size.height\*0.8));  **this**.addChild(**this**.spriteBotonTurbo); |

### Control de la pulsación

|  |
| --- |
| procesarMouseDown:**function**(event) {  **var** instancia = event.getCurrentTarget();  **var** areaBoton = instancia.spriteBotonSaltar.getBoundingBox();  **var** areaTurbo = instancia.spriteBotonTurbo.getBoundingBox();   *// La pulsación cae dentro del botón* **if** (cc.rectContainsPoint(areaBoton,  cc.p(event.getLocationX(), event.getLocationY()) )){   *// Accedemos al padre (Scene), pedimos la capa con la idCapaJuego* **var** gameLayer = instancia.getParent().getChildByTag(idCapaJuego);  *// tenemos el objeto GameLayer* gameLayer.jugador.saltar();  }  **if** (cc.rectContainsPoint(areaTurbo,  cc.p(event.getLocationX(), event.getLocationY()) )){  **var** gameLayer = instancia.getParent().getChildByTag(idCapaJuego);  gameLayer.jugador.turbo();  } |

### Inclusión de la función para impulsar

Se debe crear primero una variable con los usos máximos del turbo.

|  |
| --- |
| },turbo: **function**(){  **if**(**this**.usosTurbo>0){  **this**.body.applyImpulse(cp.v(5000, 0), cp.v(0, 0));  **this**.usosTurbo--;  } } |

## Pinchos

Para la realización de este enemigo se ha seguido la estética del anterior, simplemente que se ha cambiado el sprite, a uno con forma de pinchos, además este es estático y en el momento que chocas con el mueres.

### Creación del EnemigoPinchos

|  |
| --- |
| **var** EnemigoPinchos = cc.Class.extend({  gameLayer:**null**,  sprite:**null**,  shape:**null**, ctor:**function** (gameLayer, posicion) {  **this**.gameLayer = gameLayer;   *// Crear animación* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 4; i++) {  **var** str = **"pinchos"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **var** actionAnimacionBucle =  **new** cc.RepeatForever(**new** cc.Animate(animacion));    *// Crear Sprite - Cuerpo y forma* **this**.sprite = **new** cc.PhysicsSprite(**"#pinchos1.png"**);  *// Cuerpo estática , no le afectan las fuerzas  // Cuerpo dinámico, SI le afectan las fuerzas* **this**.body = **new** cp.Body(5, cp.momentForBox(1,  **this**.sprite.getContentSize().width,  **this**.sprite.getContentSize().height));   **this**.body.setPos(posicion);  **this**.body.setAngle(0);  **this**.sprite.setBody(**this**.body);  *// Se añade el cuerpo al espacio* gameLayer.space.addBody(**this**.body);   *// forma* **this**.shape = **new** cp.BoxShape(**this**.body,  **this**.sprite.getContentSize().width - 16,  **this**.sprite.getContentSize().height - 16);  **this**.shape.setCollisionType(tipoPincho);  *// agregar forma dinamica* gameLayer.space.addShape(**this**.shape);  *// añadir sprite a la capa  // ejecutar la animación* **this**.sprite.runAction(actionAnimacionBucle);   gameLayer.addChild(**this**.sprite,10);  }, eliminar: **function** (){  *// quita la forma* **this**.gameLayer.space.removeShape(**this**.shape);   **this**.gameLayer.removeChild(**this**.sprite);  } }); |

### Modificación GameScene

Creación de las colisiones con este nuevo tipo de enemigo.

|  |
| --- |
| },collisionJugadorConEnemigoPinchos:**function**(arbiter,space){  cc.director.pause();  cc.director.runScene(**new** GameOverLayer()); } |

Con esto ya tendríamos lo enemigos pinchos realizados, y cuando tocáramos con el moriríamos.

## Niveles

Para realizar esta ampliación se ha necesitado crear una nueva layer de selección de nivel y modificar como se llama al GameScene añadiendo un constructor con un parámetro.

### Creación de GameLevelLayer

|  |
| --- |
| **var** GameLevelLayer = cc.LayerColor.extend({  ctor:**function** () {  **this**.\_super();  **this**.init();  },  init:**function** () {  **this**.\_super(cc.color(0, 0, 0, 180));   **var** winSize = cc.director.getWinSize();   **var** boton1 = **new** cc.MenuItemSprite(  **new** cc.Sprite(res.boton\_nivel1\_png),  **new** cc.Sprite(res.boton\_nivel1\_png),  **this**.pulsarBoton1, **this**);   **var** menu = **new** cc.Menu(boton1);  menu.setPosition(winSize.width / 2, winSize.height / 2);   **this**.addChild(menu);   **var** boton2 = **new** cc.MenuItemSprite(  **new** cc.Sprite(res.boton\_nivel2\_png),  **new** cc.Sprite(res.boton\_nivel2\_png),  **this**.pulsarBoton2, **this**);   **var** menu = **new** cc.Menu(boton2);  menu.setPosition(winSize.width / 2, winSize.height / 2.5);   **this**.addChild(menu);  },  pulsarBoton1:**function** (sender) {  *// Volver a ejecutar la escena Prinicpal* cc.director.runScene(**new** GameScene(1));  },pulsarBoton2:**function** (sender) {  *// Volver a ejecutar la escena Prinicpal* cc.director.runScene(**new** GameScene(2));  } }); |

### Modificación constructor GameScene

Habría que añadir un constructor.

|  |
| --- |
| **var** GameScene = cc.Scene.extend({  ctor:**function**(nivel){  **this**.\_super();  nivelActual = nivel;  },  onEnter:**function** () {  **this**.\_super();  cc.director.resume();  **var** layer = **new** GameLayer();  **this**.addChild(layer, 0, idCapaJuego);   **var** controlesLayer = **new** ControlesLayer();  **this**.addChild(controlesLayer, 0, idCapaControles);   } }); |

### Modificación GameScene

Variable global para los niveles

|  |
| --- |
| **var** nivelActual = 1; |

Cuando choque contra la meta

|  |
| --- |
| collisionJugadorConMeta:**function** (arbiter, space){  nivelActual++;  cc.director.pause();  cc.director.runScene(**new** GameWinLayer(nivelActual)); },collisionEnemigoConMuro:**function** (arbiter, space){ |

## Control el modo plataformas

Para realizar esta ampliación se han creado un nuevo gameScene para el plataformas, un nuevo jugador para el plataformas y una layer donde seleccionar el tipo de control antes de seleccionar el nivel.

### Creación de las clases

GameScene

|  |
| --- |
| **var** tipoJugadorPlataformas = 1; **var** tipoMonedaPlataformas = 2; **var** tipoEnemigoPlataformas = 3; **var** tipoContenedorPlataformas = 4; **var** tipoDisparoPlataformas = 5; **var** tipoSueloPlataformas = 6; **var** nivelActualPlataformas = 0;  **var** GameLayerPlataformas = cc.Layer.extend({  monedas:0,  tiempoDisparar:0,  disparos:[],  disparosEliminar:[],  enemigos:[],  enemigosEliminar:[],  formasEliminar:[],  teclaIzquierda:**false**,  teclaDerecha:**false**,  teclaArriba:**false**,  teclaBarra:**false**,  monedas:[],  jugador: **null**,  space:**null**,  ctor:**function** () {  **this**.\_super();  **var** size = cc.winSize;   cc.spriteFrameCache.addSpriteFrames(res.playershootright\_plist);  cc.spriteFrameCache.addSpriteFrames(res.disparo\_plist);  cc.spriteFrameCache.addSpriteFrames(res.animacion\_cuervo\_plist);  cc.spriteFrameCache.addSpriteFrames(res.moneda\_plist);  cc.spriteFrameCache.addSpriteFrames(res.playerrunright\_plist);  cc.spriteFrameCache.addSpriteFrames(res.playerjumpright\_plist);  cc.spriteFrameCache.addSpriteFrames(res.playeridleright\_plist);  cc.spriteFrameCache.addSpriteFrames(res.playerdieright\_plist);  cc.spriteFrameCache.addSpriteFrames(res.disparo\_plist);  cc.spriteFrameCache.addSpriteFrames(res.playershootright\_plist);     *// Inicializar Space* **this**.space = **new** cp.Space();  **this**.space.gravity = cp.v(0, -350);   *// DepuraciÃ³n  //this.depuracion = new cc.PhysicsDebugNode(this.space);  //this.addChild(this.depuracion, 10);   // jugador y moneda  // IMPORTANTE: Invocamos el mÃ©****todo antes de resolver la colisiÃ³n (realmente no habrÃ¡ colisiÃ³n).* this**.space.addCollisionHandler(tipoJugadorPlataformas, tipoMonedaPlataformas,  **null**, **this**.colisionJugadorConMoneda.bind(**this**), **null**, **null**);    **this**.jugador = **new** JugadorPlataformas(**this**.space,  cc.p(50,150), **this**);   cc.eventManager.addListener({  event: cc.EventListener.KEYBOARD,  onKeyPressed: **this**.teclaPulsada,  onKeyReleased: **this**.teclaLevantada  }, **this**);   **this**.cargarMapa();  **this**.scheduleUpdate();   *// jugador y moneda  // IMPORTANTE: Invocamos el mÃ©****todo antes de resolver la colisiÃ³n (realmente no habrÃ¡ colisiÃ³n).* this**.space.addCollisionHandler(tipoJugadorPlataformas, tipoMonedaPlataformas,  **null**, **this**.colisionJugadorConMoneda.bind(**this**), **null**, **null**);   *// enemigo y contenedor  // IMPORTANTE: Invocamos el mÃ©****todo antes de resolver la colisiÃ³n (realmente no habrÃ¡ colisiÃ³n).* this**.space.addCollisionHandler(tipoEnemigoPlataformas, tipoContenedorPlataformas,  **null**, **this**.colisionEnemigoConContenedor.bind(**this**), **null**, **null**);   *// disparo y enemigo  // IMPORTANTE: Invocamos el mÃ©****todo antes de resolver la colisiÃ³n (realmente no habrÃ¡ colisiÃ³n).* this**.space.addCollisionHandler(tipoDisparoPlataformas, tipoEnemigoPlataformas,  **null**, **this**.colisionDisparoConEnemigo.bind(**this**), **null**, **null**);   *// disparo y muro  // IMPORTANTE: Invocamos el mÃ©****todo antes de resolver la colisiÃ³n (realmente no habrÃ¡ colisiÃ³n).* this**.space.addCollisionHandler(tipoDisparoPlataformas, tipoSueloPlataformas,  **null**, **this**.colisionDisparoConSuelo.bind(**this**), **null**, **null**);      **return true**; },update:**function** (dt) {  **this**.space.step(dt);   **var** capaControles =  **this**.getParent().getChildByTag(idCapaControles);   **if** ( capaControles.monedas >= 40){  nivelActualPlataformas = nivelActualPlataformas + 1;  cc.director.runScene(**new** GameScenePlataformas(nivelActualPlataformas));  }     *// Mover enemigos:* **for**(**var** i = 0; i < **this**.enemigos.length; i++) {  **var** enemigo = **this**.enemigos[i];  enemigo.moverAutomaticamente();  }   console.log(**"Formas eliminar: "**+**this**.formasEliminar.length);  *// Eliminar formas:* **for**(**var** i = 0; i < **this**.formasEliminar.length; i++) {  **var** shape = **this**.formasEliminar[i];   **for** (**var** r = 0; r < **this**.monedas.length; r++) {  **if** (**this**.monedas[r].shape == shape) {  **this**.monedas[r].eliminar();  **this**.monedas.splice(r, 1);  }  }   **for** (**var** r = 0; r < **this**.enemigos.length; r++) {  **if** (**this**.enemigos[r].shape == shape) {  console.log(**"Enemigo eliminado"**);  **this**.enemigos[r].eliminar();  **this**.enemigos.splice(r, 1);  }  }   **for** (**var** r = 0; r < **this**.disparos.length; r++) {  **if** (**this**.disparos[r].shape == shape) {  **this**.disparos[r].eliminar();  **this**.disparos.splice(r, 1);  }  }  }   **this**.formasEliminar = [];   *// CaÃ­da, sÃ­ cae vuelve a la posiciÃ³n inicial* **if**( **this**.jugador.body.p.y < -100){  **this**.jugador.body.p = cc.p(50,150);  }   **if** ( **this**.teclaBarra && **new** Date().getTime() - **this**.tiempoDisparar > 1000 ){  **this**.tiempoDisparar = **new** Date().getTime();  **var** disparo = **new** Disparo(**this**.space,  cc.p(**this**.jugador.body.p.x, **this**.jugador.body.p.y),  **this**);   **if** ( **this**.jugador.sprite.scaleX > 0){  disparo.body.vx = 400;  } **else** {  disparo.body.vx = -400;  }   **this**.disparos.push(disparo);  **this**.jugador.disparar();  }   **if** ( **this**.teclaArriba ){  **this**.jugador.moverArriba();  }  **if** (**this**.teclaIzquierda){  **this**.jugador.moverIzquierda();  }  **if**( **this**.teclaDerecha ){  **this**.jugador.moverDerecha();  }  **if** ( !**this**.teclaIzquierda && !**this**.teclaIzquierda  && !**this**.teclaDerecha ){  **this**.jugador.body.vx = 0;  }   **this**.jugador.actualizarAnimacion();   *// actualizar camara (posiciÃ³n de la capa).* **var** posicionX = **this**.jugador.body.p.x -200;  **var** posicionY = **this**.jugador.body.p.y -200;   **if**(posicionX < 0){  posicionX = 0;  }  **if**(posicionY < 0){  posicionY = 0;  }   **this**.setPosition(cc.p( -posicionX, -posicionY ));    **if** (**this**.jugador.body.vx < -200){  **this**.jugador.body.vx = -200;  }   **if** (**this**.jugador.body.vx > 200){  **this**.jugador.body.vx = 200;  }  }, cargarMapa:**function** () {  **var** nombreMapa = **"res/mapa"**+nivelActual+**".tmx"**;  **this**.mapa = **new** cc.TMXTiledMap(nombreMapa);  *// AÃ±adirlo a la Layer* **this**.addChild(**this**.mapa);  *// Ancho del mapa* **this**.mapaAncho = **this**.mapa.getContentSize().width;   *// Solicitar los objeto dentro de la capa Suelos* **var** grupoSuelos = **this**.mapa.getObjectGroup(**"Suelos"**);  **var** suelosArray = grupoSuelos.getObjects();   *// Los objetos de la capa suelos se transforman a  // formas estÃ¡ticas de Chipmunk ( SegmentShape ).* **for** (**var** i = 0; i < suelosArray.length; i++) {  **var** suelo = suelosArray[i];  **var** puntos = suelo.polylinePoints;  **for**(**var** j = 0; j < puntos.length - 1; j++){  **var** bodySuelo = **new** cp.StaticBody();   **var** shapeSuelo = **new** cp.SegmentShape(bodySuelo,  cp.v(parseInt(suelo.x) + parseInt(puntos[j].x),  parseInt(suelo.y) - parseInt(puntos[j].y)),  cp.v(parseInt(suelo.x) + parseInt(puntos[j + 1].x),  parseInt(suelo.y) - parseInt(puntos[j + 1].y)),  10);  shapeSuelo.setFriction(0);  shapeSuelo.setCollisionType(tipoSueloPlataformas);  *//shapeSuelo.setElasticity(0);* **this**.space.addStaticShape(shapeSuelo);  }  }   **var** grupoMonedas = **this**.mapa.getObjectGroup(**"Monedas"**);  **var** monedasArray = grupoMonedas.getObjects();  **for** (**var** i = 0; i < monedasArray.length; i++) {  **var** moneda = **new** Moneda(**this**,  cc.p(monedasArray[i][**"x"**],monedasArray[i][**"y"**])  ,**"Plataformas"**);   **this**.monedas.push(moneda);  }   **var** grupoEnemigos = **this**.mapa.getObjectGroup(**"Enemigos"**);  **var** enemigosArray = grupoEnemigos.getObjects();  **for** (**var** i = 0; i < enemigosArray.length; i++) {  **var** enemigo = **new** NuevoEnemigo(**this**.space,  cc.p(enemigosArray[i][**"x"**],enemigosArray[i][**"y"**]),  **this**,**"Plataformas"**);   **this**.enemigos.push(enemigo);  console.log(**"Enemigo agregado"**);  }      **var** grupoContenedores = **this**.mapa.getObjectGroup(**"Muros"**);  **var** contenedoresArray = grupoContenedores.getObjects();  **for** (**var** i = 0; i < contenedoresArray.length; i++) {  **var** contenedor = contenedoresArray[i];  **var** puntos = contenedor.polylinePoints;   **for**(**var** j = 0; j < puntos.length - 1; j++){  **var** bodyContenedor = **new** cp.StaticBody();   **var** shapeContenedor = **new** cp.SegmentShape(bodyContenedor,  cp.v(parseInt(contenedor.x) + parseInt(puntos[j].x),  parseInt(contenedor.y) - parseInt(puntos[j].y)),  cp.v(parseInt(contenedor.x) + parseInt(puntos[j + 1].x),  parseInt(contenedor.y) - parseInt(puntos[j + 1].y)),  5);   shapeContenedor.setSensor(**true**);  shapeContenedor.setCollisionType(tipoContenedorPlataformas);  shapeContenedor.setFriction(1);   **this**.space.addStaticShape(shapeContenedor);  }  }  },teclaPulsada: **function**(keyCode, event){  **var** instancia = event.getCurrentTarget();   *// Flecha izquierda* **if**( keyCode == 37){  instancia.teclaIzquierda = **true**;  }  *// Flecha derecha* **if**( keyCode == 39){  instancia.teclaDerecha = **true**;  }  *// Flecha arriba* **if**( keyCode == 38){  instancia.teclaArriba = **true**;  }  *// Barra espaciadora* **if**( keyCode == 32){  instancia.teclaBarra = **true**;  }  },teclaLevantada: **function**(keyCode, event){  **var** instancia = event.getCurrentTarget();  console.log(**"Tecla Levantada "**+keyCode);  *// Flecha izquierda* **if**( keyCode == 37){  instancia.teclaIzquierda = **false**;  }  *// Flecha derecha* **if**( keyCode == 39){  instancia.teclaDerecha = **false**;  }  *// Flecha arriba* **if**( keyCode == 38){  instancia.teclaArriba = **false**;  }  *// Barra espaciadora* **if**( keyCode == 32){  instancia.teclaBarra = **false**;  }  },colisionJugadorConMoneda:**function** (arbiter, space) {   *// Marcar la moneda para eliminarla* **var** shapes = arbiter.getShapes();  *// shapes[0] es el jugador* **this**.formasEliminar.push(shapes[1]);   **this**.tiempoEfecto = 100;   **var** capaControles =  **this**.getParent().getChildByTag(idCapaControles);  capaControles.agregarMoneda();  },colisionEnemigoConContenedor:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();  *// shapes[0] es el enemigo* **var** formaEnemigo = shapes[0];  formaEnemigo.body.vx = 0; *// Parar enemigo* }, colisionDisparoConEnemigo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();   **this**.formasEliminar.push(shapes[0]);  **this**.formasEliminar.push(shapes[1]);  }, colisionDisparoConSuelo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();   **this**.formasEliminar.push(shapes[0]);  } });  **var** idCapaJuego = 1; **var** idCapaControles = 2;  **var** GameScenePlataformas = cc.Scene.extend({  ctor:**function**(nivel){  **this**.\_super();  nivelActual = nivel;  },  onEnter:**function** () {  **this**.\_super();  **var** layer = **new** GameLayerPlataformas();  **this**.addChild(layer, 0, idCapaJuego);   **var** controlesLayer = **new** ControlesLayer();  **this**.addChild(controlesLayer, 0, idCapaControles);   } }); |

Layer

|  |
| --- |
| **var** GameModeSelect = cc.LayerColor.extend({  ctor:**function** () {  **this**.\_super();  **this**.init();  },  init:**function** () {  **this**.\_super(cc.color(0, 0, 0, 180));   **var** winSize = cc.director.getWinSize();   **var** boton1 = **new** cc.MenuItemSprite(  **new** cc.Sprite(res.boton\_plataformas\_png),  **new** cc.Sprite(res.boton\_plataformas\_png),  **this**.pulsarBoton1, **this**);   **var** menu = **new** cc.Menu(boton1);  menu.setPosition(winSize.width / 2, winSize.height / 2);   **this**.addChild(menu);   **var** boton2 = **new** cc.MenuItemSprite(  **new** cc.Sprite(res.boton\_infinite\_png),  **new** cc.Sprite(res.boton\_infinite\_png),  **this**.pulsarBoton2, **this**);   **var** menu = **new** cc.Menu(boton2);  menu.setPosition(winSize.width / 2, winSize.height / 2.5);   **this**.addChild(menu);  },  pulsarBoton1:**function** (sender) {  *// Volver a ejecutar la escena Prinicpal* cc.director.runScene(**new** GameLevelLayer(**"Plataformas"**));  },pulsarBoton2:**function** (sender) {  *// Volver a ejecutar la escena Prinicpal* cc.director.runScene(**new** GameLevelLayer(**"Infinite"**));  } }); |

Jugador

|  |
| --- |
| **var** quieto = **"quieto"**; **var** correr = **"correr"**; **var** saltar = **"saltar"**; **var** disparando = **"disparando"**;  **var** JugadorPlataformas = cc.Class.extend({  contadorVelYCero: 0,  estado: quieto,  tiempoDisparando: 0,  animacionDisparar:**null**,  animacionQuieto:**null**,  animacionCorrer:**null**,  animacionSaltar:**null**,  space:**null**,  sprite:**null**,  shape:**null**,  body:**null**,  layer:**null**,  ctor:**function** (space, posicion, layer) {  **this**.space = space;  **this**.layer = layer;   *// animaciones - correr* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 8; i++) {  **var** str = **"playerrunright\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **this**.animacionCorrer = **new** cc.RepeatForever(**new** cc.Animate(animacion));  *//this.animacionCorrer.retain();   // animaciones - saltar* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 3; i++) {  **var** str = **"playerjumpright\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **this**.animacionSaltar = **new** cc.RepeatForever(**new** cc.Animate(animacion));    *// animaciones - quieto* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 5; i++) {  **var** str = **"playeridleright\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **this**.animacionQuieto = **new** cc.RepeatForever(**new** cc.Animate(animacion));   *// Crear animaciÃ³n* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 5; i++) {  **var** str = **"playeridleright\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **var** actionAnimacionBucle =  **new** cc.RepeatForever(**new** cc.Animate(animacion));    *// Crear animaciÃ³n - disparar* **var** framesAnimacion = [];  **for** (**var** i = 1; i <= 3; i++) {  **var** str = **"playershootright\_0"** + i + **".png"**;  **var** frame = cc.spriteFrameCache.getSpriteFrame(str);  framesAnimacion.push(frame);  }  **var** animacion = **new** cc.Animation(framesAnimacion, 0.2);  **this**.animacionDisparar =  **new** cc.Repeat(**new** cc.Animate(animacion),1);    *// Crear Sprite - Cuerpo y forma* **this**.sprite = **new** cc.PhysicsSprite(**"#playeridleright\_01.png"**);  *// Cuerpo dinamico, SI le afectan las fuerzas* **this**.body = **new** cp.Body(1, Infinity);  **this**.body.setPos(posicion);  *//body.w\_limit = 0.02;* **this**.body.setAngle(0);  **this**.sprite.setBody(**this**.body);   *// Se aÃ±ade el cuerpo al espacio* **this**.space.addBody(**this**.body);   *// forma* **this**.shape = **new** cp.BoxShape(**this**.body,  **this**.sprite.getContentSize().width,  **this**.sprite.getContentSize().height);  **this**.shape.setFriction(0);  **this**.shape.setCollisionType(tipoJugadorPlataformas);  *//this.shape.setElasticity(0);  // forma dinamica* **this**.space.addShape(**this**.shape);  *// aÃ±adir sprite a la capa   // ejecutar la animaciÃ³n* **this**.sprite.runAction(actionAnimacionBucle);  layer.addChild(**this**.sprite,10);    }, moverIzquierda: **function**(){  **if** ( **this**.estado != correr && **this**.tiempoDisparando <= 0 ) {  **this**.estado = correr;  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionCorrer);  }  **this**.sprite.scaleX = -1;   **this**.body.applyImpulse(cp.v(-100, 0), cp.v(0, 0));   }, moverDerecha: **function**(){  **if** ( **this**.estado != correr && **this**.tiempoDisparando <= 0) {  **this**.estado = correr;  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionCorrer);  }  **this**.sprite.scaleX = 1;   **this**.body.applyImpulse(cp.v(100, 0), cp.v(0, 0));   }, moverArriba: **function**(){   **if** ( **this**.body.vy < 3 && **this**.body.vy > - 3 ){  **this**.contadorVelYCero = **this**.contadorVelYCero +1 ;  }   console.log(**"vel Y:"**+**this**.body.vy );   **if** ( **this**.contadorVelYCero > 1 ){  **if** ( **this**.estado != saltar && **this**.tiempoDisparando <= 0) {  **this**.estado = saltar;  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionSaltar);  }   **this**.body.applyImpulse(cp.v(0, 300), cp.v(0, 0));  **this**.contadorVelYCero = 0;  }   }, actualizarAnimacion: **function**(){  **if** ( **this**.tiempoDisparando > 0 ){  **this**.tiempoDisparando = **this**.tiempoDisparando - 1;  }  **if** ( **this**.body.vy <= 10 && **this**.body.vy >= -10  && **this**.body.vx <= 0.1 && **this**.body.vx >= -0.1 ){  **if**( **this**.estado != quieto && **this**.tiempoDisparando <= 0){  **this**.estado = quieto;  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionQuieto);  }  }   **if** ( **this**.body.vy >= 10 || **this**.body.vy <= -10 ){  **if**( **this**.estado != saltar && **this**.tiempoDisparando <= 0){  **this**.estado = saltar;  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionSaltar);  }  }  }, disparar: **function**(){  **this**.sprite.stopAllActions();  **this**.sprite.runAction(**this**.animacionDisparar);  **this**.tiempoDisparando = 40;  **this**.estado = disparando;  } }); |

### Como funciona

La idea de cómo funciona es que dependendiedo de que seleccionas cuando eliges el tipo de control cargara un gameScene u otro. Cuando carga uno u otro en objetos que se usa comunes se ha pasado un parámetro tipo que es el tipo de control que tiene para asignarle los tipos de colisión propios de ese gameScene.