Adrián Jiménez Villarreal

71904816P | uo227602@uniovi.es

Juegos Arkanoid-Tira la torre

Ampliaciones

Contenido

[Arkanoid 2](#_Toc468635700)

[Bloques aleatorios 2](#_Toc468635701)

[Niveles 3](#_Toc468635702)

[Bloques explosivos 4](#_Toc468635703)

[Modificación inicializarBloques 4](#_Toc468635704)

[Modificación update 4](#_Toc468635705)

[Tamaño de la plataforma variable 5](#_Toc468635706)

[Modificación inicializarBloques 5](#_Toc468635707)

[Modificación update 6](#_Toc468635708)

[Tira la torre 7](#_Toc468635709)

[Intentos 7](#_Toc468635710)

[Bloques objetivo 7](#_Toc468635711)

[Modificación inicializarBloques 7](#_Toc468635712)

[Modificación update 8](#_Toc468635713)

[Creación collisioners 8](#_Toc468635714)

[Llamadas a los collisioners 8](#_Toc468635715)

[Colisiones entre bloques 9](#_Toc468635716)

[Creacion collisioner 9](#_Toc468635717)

[Llamada al collisioner 9](#_Toc468635718)

# Arkanoid

## Bloques aleatorios

Para realizar está ampliación se ha modificado la función iniciazalizarBloques usando un Math.random y un Math.floor para elegir que sprite usar.

|  |
| --- |
| inicializarBloques:**function**(nivel) {  **var** insertados = 0;  **var** fila = 0;  **var** columna = 0;  **var** framesBloqueCocodrilo = [];  **var** framesBloquePanda = [];  **var** framesBloqueTigre = [];  **for** (**var** i = 1; i <= 8; i++) {  **var** strCocodrilo = **"cocodrilo"** + i + **".png"**;  **var** strPanda = **"panda"** + i + **".png"**;  **var** strTigre = **"tigre"** + i + **".png"**;  **var** framesCocodrilo = cc.spriteFrameCache.getSpriteFrame(strCocodrilo);  **var** framesPanda = cc.spriteFrameCache.getSpriteFrame(strPanda);  **var** framesTigre = cc.spriteFrameCache.getSpriteFrame(strTigre);  framesBloqueCocodrilo.push(framesCocodrilo);  framesBloquePanda.push(framesPanda);  framesBloqueTigre.push(framesTigre);  }   **while** (insertados < 50){  **var** modelo = Math.floor((Math.random() \* 3) + 1);   *//Ampliacion bloques diferentes* **switch** (modelo) {  **case** 1:  **var** animacionBloque = **new** cc.Animation(framesBloqueCocodrilo, 0.1);  **var** accionAnimacionBloque =  **new** cc.RepeatForever(**new** cc.Animate(animacionBloque));  **var** spriteBloqueActual = **new** cc.Sprite(**"#cocodrilo1.png"**);  **break**;  **case** 2:  **var** animacionBloque = **new** cc.Animation(framesBloquePanda, 0.1);  **var** accionAnimacionBloque =  **new** cc.RepeatForever(**new** cc.Animate(animacionBloque));  **var** spriteBloqueActual = **new** cc.Sprite(**"#panda1.png"**);  **break**;  **case** 3:  **var** animacionBloque = **new** cc.Animation(framesBloqueTigre, 0.1);  **var** accionAnimacionBloque =  **new** cc.RepeatForever(**new** cc.Animate(animacionBloque));  **var** spriteBloqueActual = **new** cc.Sprite(**"#tigre1.png"**);  **break**;  }  ……. |

Además, se deben cachear los sprites de los bloques en el constructor.

|  |
| --- |
| *//cachear sprite Bloques* cc.spriteFrameCache.addSpriteFrames(res.animacioncocodrilo\_plist); cc.spriteFrameCache.addSpriteFrames(res.animacionpanda\_plist); cc.spriteFrameCache.addSpriteFrames(res.animaciontigre\_plist); |

## Niveles

Para el aumento de niveles cuando se gane, se ha creado un nuevo recurso js llamado GameWinLayer, además de añadir una variable global denominada nivelActual. Posteriormente a esto se ha añadido que se gane cuando no haya más bloques en pantalla.

|  |
| --- |
| **var** nivelActual = 1;  …  update:**function** (dt) {  …  **if**( cc.rectIntersectsRect(areaPelota, areaBloque)){  **this**.removeChild(**this**.arrayBloques[i]);  **this**.arrayBloques.splice(i, 1);  console.log(**"Quedan : "**+**this**.arrayBloques.length);  destruido = **true**; }  …**if**(**this**.arrayBloques.length==0){  nivelActual++;  cc.director.pause();  **this**.addChild(**new** GameWinLayer()); } |

Recurso js GameWinLayer.

|  |
| --- |
| **var** GameWinLayer = 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** botonReiniciar = **new** cc.MenuItemSprite(  **new** cc.Sprite(res.boton\_siguiente\_png),  **new** cc.Sprite(res.boton\_siguiente\_png),  **this**.pulsarReiniciar, **this**);   **var** menu = **new** cc.Menu(botonReiniciar);  menu.setPosition(winSize.width / 2, winSize.height / 2);   **this**.addChild(menu);  },  pulsarReiniciar:**function** (sender) {  *// Volver a ejecutar la escena Prinicpal* cc.director.runScene(**new** GameScene());  } }); |

Por ultimo recordar añadir el recurso GameWinLayer.js al archivo project.json.

## Bloques explosivos

Para realizar esta ampliación se han tenido que modificar las funciones update y la función inicializarBloques. Además, se ha incluido una variable arrayTnts en el constructor para poder usarla posteriormente.

### Modificación inicializarBloques

Se ha añadido el nuevo tipo de bloque y se ha insertado en arrayTnts para saber cuál es exactamente.

|  |
| --- |
| **…**  **case** 4:  **var** spriteBloqueActual = cc.Sprite.create(res.tnt);  **break**; } **if**(modelo!=4)  spriteBloqueActual.runAction(accionAnimacionBloque);  …  **if**(modelo==4)  **this**.arrayTnts.push(spriteBloqueActual); |

### Modificación update

Se ha modificado la destrucción de los bloques ya que hay que buscar cuales se deben destruir cuando se rompe un bloque de tnt, en este caso todos los que estén alrededor.

|  |
| --- |
| **…**  **var** aDestruir = []; **if**(destruido){  **this**.velocidadX = **this**.velocidadX\*-1;  **this**.velocidadY = **this**.velocidadY\*-1;  **var** it=0;  **for**(**var** i=0;i<**this**.arrayTnts.length;i++){  **var** tnt = **this**.arrayTnts[i];  **if**(tnt.x == bloqueDestruido.x && tnt.y==bloqueDestruido.y){  **for**(**var** j=0; j<**this**.arrayBloques.length;j++){  **if**(**this**.arrayBloques[j].x==tnt.x-40 && **this**.arrayBloques[j].y==tnt.y-40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x && **this**.arrayBloques[j].y==tnt.y-40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x+40 && **this**.arrayBloques[j].y==tnt.y-40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x-40 && **this**.arrayBloques[j].y==tnt.y){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x+40 && **this**.arrayBloques[j].y==tnt.y){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x-40 && **this**.arrayBloques[j].y==tnt.y+40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x && **this**.arrayBloques[j].y==tnt.y+40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  **if**(**this**.arrayBloques[j].x==tnt.x+40 && **this**.arrayBloques[j].y==tnt.y+40){  aDestruir[it]=**this**.arrayBloques[j]; it++;  }  }  }  }  **for**(**var** i=0;i<aDestruir.length;i++){  **var** lugar=-1;  **var** bloqueADestruir = aDestruir[i];  **for**(**var** j = 0 ; j<**this**.arrayBloques.length;j++){  **var** bloqueBuscado = **this**.arrayBloques[j];  **if**(bloqueADestruir.x == bloqueBuscado.x && bloqueADestruir.y == bloqueBuscado.y)  lugar=j;  }  **if**(lugar!=-1){  **this**.removeChild(**this**.arrayBloques[lugar]);  **this**.arrayBloques.splice(lugar,1);  }  } }  … |

## Tamaño de la plataforma variable

Para realizar esta ampliación se han tenido que modificar las funciones update y la función inicializarBloques. Ademas se ha incluido unas variables sizePlataforma ,arrayPowerUp y arrayPowerDown en el constructor para usarlas posteriormente.

### Modificación inicializarBloques

|  |
| --- |
| **…**  **case** 5:  **var** spriteBloqueActual = cc.Sprite.create(res.power\_up);  **break**; **case** 6:  **var** spriteBloqueActual = cc.Sprite.create(res.power\_down);  **break**;  }  **if**(modelo!=4 && modelo!=5 && modelo!=6)  spriteBloqueActual.runAction(accionAnimacionBloque);  …  **if**(modelo==5)  **this**.arrayPowerUp.push(spriteBloqueActual); **if**(modelo==6)  **this**.arrayPowerDown.push(spriteBloqueActual);  … |

### Modificación update

Dentro de cuando se destruye un bloque se comprueba si es un power\_up o power\_down.

|  |
| --- |
| **…**  **for**(**var** i=0; i<**this**.arrayPowerUp.length;i++){  **var** powerUp = **this**.arrayPowerUp[i];  **if**(bloqueDestruido.x==powerUp.x && bloqueDestruido.y==powerUp.y){  **if**(**this**.sizePlataforma==1){  **var** xBarra = **this**.spriteBarra.x + **this**.velocidadX;  **var** yBarra = **this**.spriteBarra.y + **this**.velocidadY;  **this**.removeChild(**this**.spriteBarra);  **this**.spriteBarra = cc.Sprite.create(res.barra\_3\_png);  **this**.spriteBarra.setPosition(cc.p(xBarra , yBarra ));  **this**.addChild(**this**.spriteBarra);  **this**.sizePlataforma=2;  }  **if**(**this**.sizePlataforma==0){  **var** xBarra = **this**.spriteBarra.x + **this**.velocidadX;  **var** yBarra = **this**.spriteBarra.y + **this**.velocidadY;  **this**.removeChild(**this**.spriteBarra);  **this**.spriteBarra = cc.Sprite.create(res.barra\_2\_png);  **this**.spriteBarra.setPosition(cc.p(xBarra , yBarra ));  **this**.addChild(**this**.spriteBarra);  **this**.sizePlataforma=1;  }  } } **for**(**var** i=0; i<**this**.arrayPowerDown.length;i++){  **var** powerDown = **this**.arrayPowerDown[i];  **if**(bloqueDestruido.x==powerDown.x && bloqueDestruido.y==powerDown.y){  **if**(**this**.sizePlataforma==1){  **var** xBarra = **this**.spriteBarra.x + **this**.velocidadX;  **var** yBarra = **this**.spriteBarra.y + **this**.velocidadY;  **this**.removeChild(**this**.spriteBarra);  **this**.spriteBarra = cc.Sprite.create(res.barra\_1\_png);  **this**.spriteBarra.setPosition(cc.p(xBarra , yBarra ));  **this**.addChild(**this**.spriteBarra);  **this**.sizePlataforma=0;  }  **if**(**this**.sizePlataforma==2){  **var** xBarra = **this**.spriteBarra.x + **this**.velocidadX;  **var** yBarra = **this**.spriteBarra.y + **this**.velocidadY;  **this**.removeChild(**this**.spriteBarra);  **this**.spriteBarra = cc.Sprite.create(res.barra\_2\_png);  **this**.spriteBarra.setPosition(cc.p(xBarra , yBarra ));  **this**.addChild(**this**.spriteBarra);  **this**.sizePlataforma=1;  }  } }  … |

# Tira la torre

## Intentos

Para realizar esta ampliación se ha necesitado modificar la función procesarMouseDown. Además se ha añadido una variable intentos en el constructor.

|  |
| --- |
| },procesarMouseDown:**function**(event) {  *// Ambito procesarMouseDown* **var** instancia = event.getCurrentTarget();  **if**(instancia.tiempo == 0 ){  instancia.mundoActivo = **true**;  instancia.tiempo = **new** Date().getTime();   **var** body = instancia.spritePelota.body;  body.applyImpulse(cp.v( event.getLocationX() - body.p.x, event.getLocationY() - body.p.y), cp.v(0,0));  instancia.intentos--;  }  **else if**(instancia.tiempo > 0 && instancia.intentos>0){  instancia.mundoActivo = **false**;  instancia.spritePelota.setPosition(cc.p(cc.winSize.width\*0.1 , cc.winSize.height\*0.5));  instancia.tiempo = 0;  } |

## Bloques objetivo

Para esta ampliación deberemos crear un array para los bloques objetivo y separarlos de los bloques normales, ya que los bloques objetivo se destruirán siempre y cuando la velocidad en el eje x sea mayor que 100, o cuando estos caigan al vacío. Ahora no solo habrá tipoBloque y tipoMuro, también existirán tipoBola y tipoObjetivo para las diferentes colisiones.

### Modificación inicializarBloques

|  |
| --- |
| **while**(altoTorre < 4){   **var** spriteBloque = **new** cc.PhysicsSprite(**"#panda1.png"**);   *// Masa 1* **var** body = **new** cp.Body(1, cp.momentForBox(1, spriteBloque.width, spriteBloque.height));   body.p = cc.p(cc.winSize.width\*0.73 , cc.winSize.height\*0.4 + 10 + 20 + spriteBloque.height\*altoTorre);   spriteBloque.setBody(body);  *// Este si hay que añadirlo* **this**.space.addBody(body);   **var** shape = **new** cp.BoxShape(body, spriteBloque.width, spriteBloque.height);  shape.setFriction(1);  shape.setCollisionType(tipoObjetivo);  **this**.space.addShape(shape);  **this**.addChild(spriteBloque);   *// agregar el Sprite al array Bloques* **this**.arrayObjetivo.push(spriteBloque);   altoTorre++;  } altoTorre=0; **while**(altoTorre<2){  **var** spriteBloque = **new** cc.PhysicsSprite(**"#cocodrilo1.png"**);   **var** body = **new** cp.Body(1, cp.momentForBox(1, spriteBloque.width, spriteBloque.height));   body.p = cc.p(cc.winSize.width\*0.67 , cc.winSize.height\*0.4 + 10 + 20 + spriteBloque.height\*altoTorre);   spriteBloque.setBody(body);  *// Este si hay que añadirlo* **this**.space.addBody(body);   **var** shape = **new** cp.BoxShape(body, spriteBloque.width, spriteBloque.height);  shape.setFriction(1);  shape.setCollisionType(tipoBloque);  **this**.space.addShape(shape);  **this**.addChild(spriteBloque);   *// agregar el Sprite al array Bloques* **this**.arrayBloques.push(spriteBloque);   altoTorre++;  } |

### Modificación update

|  |
| --- |
| **if**( **~~this~~**~~.arrayBloques.length > 0~~ **this**.arrayObjetivo.length > 0){  …. |

### Creación collisioners

Los diferentes tipos de collisioners que necesitamos, ya que hay que tratar nuevos tipos de colisión, además cuando se colisiona bola y bloqueObjetivo o un bloque normal tiene que llevar más de cierta velocidad, para destruirlo.

|  |
| --- |
| },collisionBloqueConMuro:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();  **this**.formasEliminar.push(shapes[1]); },collisionBolaConObjetivo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();  **if**(**this**.spritePelota.body.vx>100)  **this**.formasEliminar.push(shapes[0]); }collisionObjetivoConMuro:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();  **this**.formasEliminar.push(shapes[1]); } |

### Llamadas a los collisioners

|  |
| --- |
| *// muro y bloque* **this**.space.addCollisionHandler(tipoMuro, tipoBloque,  **null**, **null**, **this**.collisionBloqueConMuro.bind(**this**), **null**); *// pelota y objetivo* **this**.space.addCollisionHandler(tipoObjetivo, tipoBola,  **null**, **null**, **this**.collisionBolaConObjetivo.bind(**this**), **null**);  *// pelota y bloque* **this**.space.addCollisionHandler(tipoBloque, tipoBola,  **null**, **null**, |

## Colisiones entre bloques

Para realizar esta ampliación solo se deberá crear un nuevo tipo de collisioner y definir para que velocidad x e y un bloque destruirá un bloque objetivo.

### Creacion collisioner

|  |
| --- |
| },collisionBloqueConObjetivo:**function** (arbiter, space) {  **var** shapes = arbiter.getShapes();  **var** body = arbiter.body\_a;  **if**(body.vx > 100 || body.vy<-50){  **this**.formasEliminar.push(shapes[0]);  } } |

### Llamada al collisioner

|  |
| --- |
| *// bloque y objetivo* **this**.space.addCollisionHandler(tipoObjetivo, tipoBloque,  **null**, **null**, **this**.collisionBloqueConObjetivo.bind(**this**), **null**); |