UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de videojuegos clásicos con Scratch

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Samuel Villaescusa Vinader

Tutor: Xavier Molero Prieto

22 de agosto de 2015

Resumen

El presente trabajo se centra en la elaboración de cinco de los grandes videojuegos de la historia (*Pong, Space Invaders, Asteroids, Pac Man y Donkey Kong*) mediante el uso del lenguaje de programación Scratch. Los cuales, una vez finalizados, serán incluidos en la página web del Museo de Informática de la Escola Tècnica Superior d´Enginyeria Informàtica con el objetivo de demostrar la potencia del lenguaje Scratch, y así animar a más usuarios a usarlo para aprender programación de forma didáctica y divertida. Primero se analizará la historia detrás de estos grandes videojuegos para dar a conocer los orígenes del mundillo, dando paso posteriormente a explicar las bases del lenguaje de programación Scratch, el cual se adapta a la creación de videojuegos de manera sencilla mediante el uso de diversas herramientas que ofrece para ello. Por último, se explicará detalladamente el procedimiento a seguir para desarrollar los cinco videojuegos mencionados anteriormente y así mostrar a los usuarios cómo adaptar videojuegos clásicos mediante un lenguaje de programación actual.

Palabras clave: videojuegos clásicos, Scratch, diseño de videojuegos, programación, pensamiento computacional.

Abstract

The current work is focused in the development of five of the greatest video games in history (*Pong, Space Invaders, Asteroids, Pac Man and Donkey Kong*) using the Scratch programming language. After their development, those video games will be included in the web site of the Computing Museum of the Escola Tècnica Superior d'Enginyeria Informàtica with the objective of demonstrate the power of the Scratch language and encourage more users to use it to learn programming in a didactic and funny way. First of all will be analyzed the history behind those great videogames about their origins, and then the basis of Scratch programming language will be explained and adapted to the creation of video games in a easy way using some tools provided for it. Finally, the development procedure of the five videogames mentioned above will be explained in detail to show Scratch users how to adapt classic video games through a current programming language.

Keywords: classic videogames, Scratch, videogames design, programming, computational thinking.

Resum

El present treball es centra en l'elaboració de cinc dels grans videojocs de la història (*Pong, Space Invaders, Asteroids, Pac Man y Donkey Kong*) mitjançant l'ús del llenguatge de programació Scratch, els quals, una vegada finalitzats, seràn inclosos a la pàgina web del Museu d'Informàtica de l'Escola Tècnica Superior d' Enginyeria Informàtica amb l'objectiu de demostrar la potència del llenguatge

Scratch, i així animar a més usuaris a emprar-lo per aprendre programació de manera didàctica i divertida. Primer s'analitzarà la història darrere d'aquests grans videojocs per a donar a conéixer els oritges d'aquestes aplicacions, donant pas posteriorment a explicar les bases del llenguatge de programació Scratch, el qual s'adapta a la creació de videojocs de manera senzilla mitjançant l'ús de diverses eines que ofereix. Finalment, s'explicarà detalladament el procediment seguit per desenvolupar els cinc videojocs nomenats prèviament i així mostrar als usuaris com adaptar videojocs clàssics mitjançant un llenguatge de programació actual.

Paraules clau: videojocs clàssics, Scratch, diseny de videojocs, programació, pensament computacional.

Índice general

Índice de figuras

CAPÍTULO 1

Motivación y objetivos

En este primer capítulo expondremos las razones por las que se a llevado a cabo la elaboración del presente trabajo, los objetivos a cumplir y la estructura de la memoria.

1.1 Motivación

El Museo de Informática[1] de la Escola Tècnica Superior d'Enginyeria Informàtica de la Universitat Politècnica de València fue inaugurado el 11 de diciembre del año 2001 y ofrece un recorrido por la historia de la informática a sus visitantes.

El museo dispone de varias actividades, en concreto los talleres de Scratch[2] (lenguaje de programación basado en bloques), los cuales intentan acercar a los visitantes al mundo de la programación a través de un lenguaje sencillo de aprender. Dentro de esta actividad, se ha pensado en añadir un ejemplo de lo que se puede llegar a realizar con conocimientos básicos de programación mediante la ayuda de Scratch , con la inclusión de cinco de los grandes videojuegos de la historia en un apartado de la web del museo, para que los visitantes puedan jugar con ellos y así ayudar a despertar su interés por la programación y el desarrollo de este tipo de programas.

Para finalizar, cabe destacar que el trabajo realizado por Antonio Ortíz sobre los videojuegos retro y los jugones [13] ha servido como motivación principal a la hora de llevar a cabo el presente trabajo de programación de videojuegos clásicos mediante el uso del lenguaje de programación Scratch.

[1] Página web disponible en `http://museo.inf.upv.es`

[2] Enlace a su web `https://scratch.mit.edu/`

1.2 Objetivos

El presente trabajo tiene como objetivo principal la elaboración de cinco de los grandes videojuegos clásicos (*Pong, Space Invaders, Asteroids, Donkey Kong y Pac Man*), para su inclusión en la web del Museo de Informática de la Escola Tècnica Superior d'Enginyeria Informàtica de la Universitat Politècnica de València.

El trabajo trata diversos temas, aparte del desarrollo de dichos videojuegos:

1. Mostrar la historia de los juegos y como éstos influyeron en la industria.
2. Explicar el funcionamiento del lenguaje Scratch y porqué se adapta tan bien a la creación de juegos sencillos.
3. Resumir el funcionamiento de los cinco videojuegos realizados.
4. Dar a conocer el porqué del desarrollo de dichos videojuegos y su inclusión en la página web del museo para su libre utilización.

Como objetivo final, el trabajo pretende dar a conocer la programación a usuarios noveles a través de la elaboración de videojuegos, y con ello motivar a los usuarios de la web del museo a desarrollar dichos artefactos con lenguajes actuales como Scratch.

1.3 Estructura de la memoria

El presente documento se estructura en once capítulos, los cuales se exponen brevemente a continuación.

- **Capítulo 1**. Se expone la motivación que ha llevado a realizar el presente trabajo, así como objetivos y estructura de la memoria.
- **Capítulo 2**. Se analiza la historia de los videojuegos clásicos así como la influencia que éstos han tenido en la evolución de la industria en la actualidad.
- **Capítulo 3**. Se detalla en profundidad la funcionalidad del lenguaje Scratch (y su plataforma), y sus herramientas de programación.
- **Capítulo 4**. En este capítulo se detalla la gran utilidad del lenguaje Scratch a la hora de crear pequeños videojuegos de forma fácil y sencilla.

- **Capítulos 5, 6, 7, 8 y 9**. Estos capítulos se centran en exponer el funcionamiento de los videojuegos *Pong, Space Invaders, Asteroids, Donkey Kong y Pac Man*, así como la descripción en detalle de cada uno de los objetos que los componen y posibles ampliaciones en los mismos.

- **Capítulo 10**. En este capítulo se va a explicar como se han incluido los cinco videojuegos desarrollados en la página web del museo de la escuela.

- **Capítulo 11**. El último capítulo recoge las conclusiones obtenidas una vez realizado el trabajo y la memoria. Analiza los objetivos alcanzados y la utilidad futura del mismo.

1.4 Uso de la bibliografía

En este apartado se va a hablar sobre la bibliografía utilizada para la redacción del trabajo, centrándonos en cómo se ha aprovechado y cuáles de los recursos que nos aporta pueden ser útiles a la hora de comenzar con el lenguaje de programación Scratch y la programación de videojuegos.

Como se comenta al principio del capítulo, la motivación inicial del trabajo fue el artículo de Antonio Ortiz disponible en [13], gracias al cual se consiguió la información inicial acerca de los videojuegos retro así como una pequeña introducción a su historia.

Para seguir con el desarrollo, se recurrieron a varios libros sobre la historia del videojuego así como detalles acerca de la mercadotecnia y los videojuegos más influyentes que nos han llevado a la actualidad de la industria (disponibles en [8], [5], [12], [1], [19], [11]).

Para la redacción los capítulos de introducción al lenguaje Scratch y el entorno que éste nos propone para el desarrollo de videojuegos, se consultaron diversos libros sobre iniciación al lenguaje, centrándonos el trabajo de una alumna de la escuela, Cynthia España Sanjuan, el cual se puede consultar en [2].

Todo lo relacionado con la creación de videojuegos y programación con Scratch que constituye el grueso del trabajo, se recoge en diversos libros (consultar los siguientes: [4], [9], [14], [15], [18], [20]) que nos introducen en el mundo de manera didáctica e intentan realizarlo de una manera divertida , aunque el más importante de todos se encuentra en [10], ya que cuenta como desarrollar videojuegos de diversos géneros desde cero en el entorno Scratch siguiendo unos pasos muy concretos y explicando el porqué de las decisiones a la hora del desarrollo de los mismos.

Por último, una referencia a tener en cuenta a la hora de realizar un videojuego en el cual uno de los personajes o enemigos requiera una lógica algo más compleja que un simple movimiento de izquierdas a derechas, se debería consultar [6],

ya que las máquinas de estados finitos nos permiten reducir la complejidad de diversas acciones o eventos en un objeto a un simple diagrama en el cual se pasa de un estado a otro, lo que nos permite visualizar el comportamiento de nuestros personajes, enemigos etc.

1.5 Derechos de autor

Cabe mencionar que las imágenes usadas para simular los escenarios y los objetos de cada uno de los videojuegos han sido obtenidas libres de derechos de autor desde diversas páginas de wikipedia y desde la propia página de Scratch (gracias a la opción de compartir el código interno de los videojuegos, así como cada uno de sus recursos); se puede aplicar lo mismo a los sonidos, ya que la gran mayoría de ellos se han obtenido de la página web `http://www.sonidosmp3gratis.com/` y de los propios recursos que ofrece Scratch.

Además, las imágenes han sido diseñadas o son modificaciones de una imagen inicial, para conseguir el efecto deseado en cada uno de los videojuegos o para modificar ligeramente la apariencia del objeto que pretenden simular (así como los sonidos).

CAPÍTULO 2

Los videojuegos clásicos: una aproximación histórica

Este capítulo se centra en describir por orden cronológico cinco grandes videojuegos clásicos de la historia (*Pong, Space Invaders, Asteroids, Pac Man, Donkey Kong*), así como sus características, tipología, plataformas, etc.

Se pretende abarcar todo lo que serían las dos primeras etapas del mundo de los videojuegos (la Época Dorada y el inicio de la Madurez), y así, en los capítulos 5, 6, 7, 8 y 9, mostrar un desarrollo actual de dichos videojuegos con el lenguaje de programación Scratch.

2.1 *Pong* (1972)

El pistoletazo de salida para el mundo de los videojuegos lo dio Ralph Baer, al patentar en 1968 lo que sería el primer videojuego interactivo para uso convencional en televisión. En 1971 nacieron las primeras máquinas recreativas de la historia, las *Computer Space Machines*, con el juego *Computer Space* diseñado por los estudiantes Ted Dabney y Nolan Bushnell.

Gracias a esto, Bushnell pensó en las posibilidades que podría tener si un videojuego similar a los anteriores pudiese llegar al público de forma masiva (más información sobre marketing y videojuegos en [19]); por lo que en 1972 fundó la empresa Atari junto con Ted Dabney, lo que revolucionaría el mercado del videojuego con *Pong* (1972). Cabe mencionar que para producirlo contrataron al que sería el primer trabajador de *Atari*, Allan Alcorn, el cual se encargó del diseño del circuito eléctrico (a pesar de que no tenía experiencia en el mundillo) y les permitió establecer su primera máquina en el bar local *Andy Caps*. En la Figura 2.1 se puede ver lo que consiguieron dichos visionarios.

Hay una anécdota acerca de la primera máquina recreativa de *Pong*, en la que Bushnell cuenta que el dueño del bar los llamó a él y a Ted, debido a que parecía que ésta se había estropeado. Al abrirla para inspeccionar cual era el fallo, descubrieron que el recipiente donde se almacenaban las monedas estaba completamente lleno y no permitía que iniciase el juego. A raíz de aquello, los ingresos de Atari se dispararon. Para más información sobre la historia de los primeros videojuegos, se puede consultar el capítulo «Entre ruedas dentadas y chorros de vapor. Primeros diálogos con la máquina» en [8].

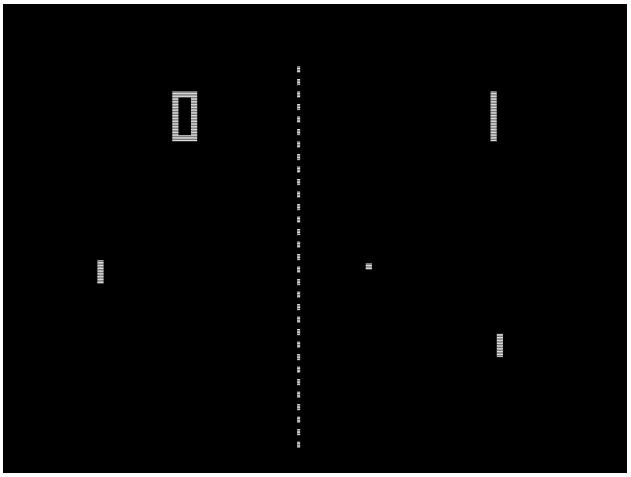

Figura 2.1: Escena de juego de *Pong*, en la que se aprecian la pelota y las dos paletas que lo forman

2.1.1. Dispositivos de juego

Aunque en el año 1970 se creó el primer microprocesador por la empresa Intel, la máquina de *Pong* no lo utilizó, ya que apenas existía documentación y era demasiado caro. La placa del videojuego *Pong* se asemejaba eléctricamente a la PDP-1[1], ya que se valía de transistores, condensadores etc, y utilizaba lógica discreta[2]. Dicha placa pertenece a lo que se denomina «Segunda generación de computadores», y ocupaba un espacio de unos pocos metros cuadrados, lo que permitió a *Atari* fabricar una máquina accesible para colocar en salas de videojuegos, bares etc. En la Figura 2.2 se puede observar el resultado de la primera máquina recreativa de Atari, *Pong*.

La primera máquina recreativa de *Pong* disponía de dos reguladores con forma cilíndrica, lo que permitía a los usuarios mover las paletas arriba y abajo, para interceptar la bola y golpearla. Posteriormente las máquinas mejoraron y se incluyeron dos *joysticks*, los cuales añadieron mayor precisión a los movimientos de las paletas.

[1]Más información sobre la PDP-1 en `http://es.wikipedia.org/wiki/PDP-1`

[2]Más información en `http://es.wikipedia.org/wiki/Sistema_combinacional`

Figura 2.2: Máquina recreativa de *Pong*, muy similar a la primera que se instaló en el bar local *Andy Caps*.

En octubre de 1977 se fabricó la primera consola denominada Atari 2600, la cual incluía entre uno de sus títulos el *Pong* original, y se manejaba con *joysticks*, *paddles*, *trackballs*, etc., incluso permitía teclados numéricos. También incluía un microprocesador a modo de CPU, el denominado MOS Technology 6507[3], además disponía de un procesador de audio y vídeo capaz de mostrar 4 colores por línea y dos vías de sonido. Se puede observar la forma de la consola en la Figura 2.3.

2.1.2. Otros detalles

El juego original de *Pong* fue catalogado como género **Arcade**, el cual más tarde daría lugar a una gran variedad de videojuegos (de tenis, *hockey*, baloncesto

[3]Más información en `http://en.wikipedia.org/wiki/MOS_Technology_6507`

Figura 2.3: Una de las primera consolas del mercado, la Atari 2600.

etc.). En la actualidad existen diversas versiones remasterizadas del *Pong*, incluso puede jugarse en otros videojuegos como *Mortal Combat 3*[4] donde se permite al usuario jugar una partida al *Pong* al finalizar el juego.

La **dificultad** del videojuego radicaba en evitar que el oponente consiguiera hacer que no le devolvieras la pelota, ya que el juego estaba formado por dos paletas enfrentadas y una pelota que rebotaba en los bordes de la pantalla. Cada una era utilizada por un usuario, y consistía en obtener más puntos que el contrincante.

Respecto al **público** al que iba dirigido, éste fue desarrollado para todo tipo de usuarios, ya que la primera máquina fue establecida en un bar local al que podían acudir tanto mujeres como hombres. Más tarde se incluirían las máquinas recreativas en los salones recreativos, más dirigidos al público adolescente. A diferencia de los videojuegos actuales que, en su gran mayoría, van dirigidos al público masculino por su alto grado de violencia y la caracterización de la protagonista femenina.

Figura 2.4: Mesa, maza y disco del Hockey de aire.

[4]Vídeo en el que se muestra el *Pong* en *Mortal Combat 3*, `https://www.youtube.com/watch?v=CBsxWLr-rUk`, [Consultado el 13/04/2015]

A modo de curiosidad, Nolan Bushnell fundó Atari con tan solo doscientos cincuenta dólares y se basó en su juego favorito (el Go[5]) para darle nombre a la que sería una de las mayores compañías de videojuegos de la historia. Se puede consultar el capítulo «Historia de las consolas del siglo XX» en [12] para más información acerca del inicio de los videojuegos.

Por último, aunque no menos importante, el videojuego *Pong* estaba pensado para **simular** un juego de tenis, aunque por la forma en que terminó mostrándose al público, se asemejaba más al *Tejo Aéreo* o *Hockey de aire*[6], en la Figura 2.4 se puede observar dicho juego.

2.2 *Space Invaders* (1978)

Space Invaders nace en el año 1978 a manos de Toshiniro Nishikado, diseñador japonés de la compañía Taito Corporation. El videojuego tuvo un éxito abrumador de cara al mercado ya que marcó el inicio de un género conocido como *Shoot 'em up* (traducido al español como «Mátalos a tiros»).

Fue uno de los precursores del videojuego moderno, ya que ayudó a expandir la industria del sector, estancado en juegos del estilo *Pong*, llegando a ser uno de los primeros fenómenos comerciales en el mundillo. Tal fue su éxito que muchas otras compañías iniciaron una lucha de clones, ya que éste no poseía *copyright*. Algunos de éstos fueron *Space Invaders Deluxe*, *Super Invaders* etc. Incluso se dice que en Japón se produjo una escasez de monedas de cien yenes, dado que eran las que se utilizaban en las máquinas recreativas de *Space Invaders*, y que el gobierno se vio obligado a aumentar el número de éstas.

Tomoshiro Nishikado tardó un año en diseñar y desarrollar el hardware necesario para producir el videojuego, del que usaría como inspiración el juego *Breakout* desarrollado previamente por Atari. A modo de anécdota, cabe decir que los primeros diseños del videojuego estaban basados en tanques y naves de combate, aunque Nishikado no estaba satisfecho con sus movimientos, ya que las limitaciones técnicas del momento dificultaban la simulación de vuelo. Se pensó en incluir humanos, aunque éste consideró inmoral crear un videojuego en el que se disparase a personas. Después de ver una revista acerca de *Star Wars*, se decidió por el uso de alienígenas para crear su videojuego (basando el diseño de los enemigos en cangrejos y calamares). En la Figura 2.5 se puede apreciar el primer diseño del videojuego.

[5]Juego de origen Chino de estrategia que consiste en acabar con mayor terreno de juego que el contrincante

[6]Más información disponible en `http://es.wikipedia.org/wiki/Hockey_de_aire`

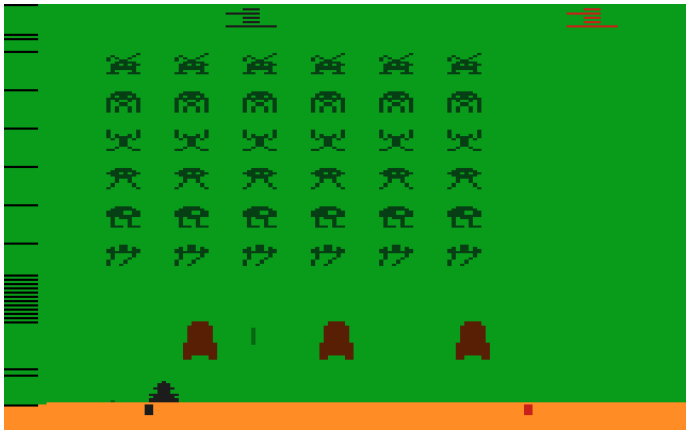

Figura 2.5: *Space Invaders* original de Toshiniro Nishikado, donde se observan las naves alienígenas y la nave defensora, la cual nos permitía acabar con éstas.

2.2.1. Dispositivos de juego

Debido a que el hardware de la época en Japón era muy pobre para simular y ejecutar las tareas que envolvían al videojuego, Nishikado tuvo que crear su propio hardware utilizando para ello microprocesadores provenientes de los Estados Unidos. Creó el denominado *Arcade board*, que es un ordenador dedicado a ejecutar videojuegos arcade. La primera versión del hardware de *Space Invaders* se diseñó basándose en el diseño de *Cocktail*, el cual consistía en una mesa con dos asientos, uno a cada lado de la misma, con los mandos y botones para jugar enfrente de los asientos y la pantalla de juego en el centro de la mesa. En la Figura 2.6 se puede apreciar una imagen de la mesa tipo *Cocktail*.

Figura 2.6: Una mesa recreativa de estilo *Cocktail*, las cuales eran muy populares en la época del *Space Invaders* en Japón.

La potencia del hardware diseñado por Nishikado no fue suficiente para poder renderizar las figuras en color, por lo que su diseño original fue en blanco y negro. Nishikado no tuvo en cuenta un pequeño detalle, ya que cuantos menos alienígenas quedaban en pantalla, más rápido se desplazaban, y es que el microprocesador procesaba más rápido los datos de posición de los mismos. En lugar de resolver el problema, decidió dejar así el videojuego, ya que le parecía que era un reto para los jugadores.

2.2.2. Otros detalles

Como se ha dicho *Space Invaders* dio inicio a un **género** denominado *Shoot 'em up*, el cual dio paso al videojuego moderno, y al género que hoy día se conoce como FPS[7] o *First Person Shooter*, que consiste en disparar al o los adversarios para obtener el mayor número de puntuación posible. En la Figura 2.7 se puede apreciar la imagen de un juego FPS.

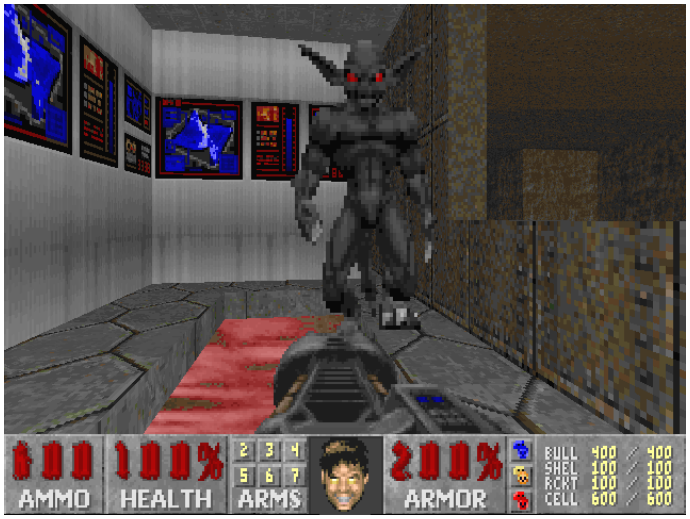

Figura 2.7: Una escena del videojuego *Doom*, precursor del género FPS.

La **dificultad** del videojuego radicaba en el progresivo aumento de la velocidad de las naves enemigas, y es que cuantas más naves destruías, más rápido disparaban y se movían, además, con el paso del tiempo, los búnkeres se iban destruyendo y cada vez el jugador tenía más dificultad para evadir los disparos enemigos.

Al igual que *Pong*, *Space Invaders* no fue diseñado para un **género** único, ya que en aquella época se pretendía llegar al máximo número de usuarios. A diferencia de *Pong*, éste se centró en un público adolescente, ya que las primeras máquinas se instalaron en los salones recreativos de Japón, aunque debido a su éxito llegó a tener un rango de jugadores muy amplio.

[7]Más información en `http://es.wikipedia.org/wiki/Videojuego_de_disparos_en_primera_persona`

2.3 *Asteroids* (1979)

Asteroids fue un popular videojuego de 1979 basado en vectores y diseñado por Lyle Rains y Ed Logg, trabajadores de Atari, que pertenece al género Arcade y consistía en destruir una serie de asteroides que aparecían en la pantalla, evitando chocar contra los fragmentos que éstos generaban. Su diseño está inspirado en el videojuego *Spacewar!*[8]. En la Figura 2.8 se puede apreciar la pantalla del videojuego.

Figura 2.8: La nave y los asteroides en una pantalla del videojuego *Asteroids*.

Asteroids ha sido uno de los videojuegos más populares de la historia, tal fue su éxito que dejó atrás al famoso *Space Invaders* en los Estados Unidos y ha sido el videojuego más vendido de la compañía Atari; de hecho, obligó a aumentar la capacidad de las cajas donde se depositaban las monedas de las máquinas recreativas, ya que éstas se llenaban demasiado rápido. La siguiente cita es partícipe del éxito del videojuego: «Los jugadores echan diez millones de monedas de cuarto de dólar al día en las máquinas recreativas de *Asteroids*. (Frank Laney Jr., Electronic Games , 1981).»

La popularidad de *Asteroids* llegó a ser tan grande que los jugadores continuamente competían para intentar obtener la mejor puntuación en el mismo. En el año 1982 el jugador Scott Safran de 15 años consiguió una marca de 41.336.440 puntos, que no se volvería a batir hasta el año 2010 por el jugador John McAllister que consiguió la marca de 41.338.740 puntos en una partida de tres días de duración. Una anécdota acerca de la hazaña es que mientras John fue al baño, estuvo a punto de perder la partida, ya que su táctica consistía en acumular suficientes vidas y en ese momento ir a comer, o realizar otras necesidades.

[8]Más información en `http://es.wikipedia.org/wiki/Spacewar!`

2.3.1. Dispositivos de juego

Asteroids fue implementado en hardware desarrollado por Delman. Al ser un videojuego de vectores, los gráficos están compuestos por líneas que se dibujaban en un monitor vectorial[9]. En la Figura 2.9 se puede observar un monitor vectorial en un osciloscopio.

Figura 2.9: Un osciloscopio que simula un monitor de vectores en el que se está jugando a *Asteroids*.

Debido al gran éxito del juego, éste se incluiría posteriormente en la consola Atari 2600, cuya funcionalidad está descrita en el apartado del videojuego *Pong*.

2.3.2. Otros detalles

El **género** del videojuego era Arcade y *Shooter*, ya que consistía en disparar a los asteroides que aparecían por la pantalla y al mismo tiempo evitar que la nave que el jugador manejaba se estrellase contra uno de éstos.

La **dificultad** de *Asteroids* radicaba en que cuanto más larga se hiciese la partida, mas cantidad de meteoritos aparecían en la pantalla, por lo que al destruir uno de éstos el área de juego quedaba completamente inundada de rocas, las cuales era muy difícil de esquivar, por lo que había que centrarse en evitar los objetos en pantalla a la vez que los destruíamos para conseguir puntuación. En la Figura 2.10 se puede observar un nivel prácticamente repleto de asteroides.

[9]Más información en `http://en.wikipedia.org/wiki/Vector_monitor`

Figura 2.10: Un nivel avanzado de *Asteroids* en el que la pantalla está prácticamente inundada por asteroides y sus fragmentos.

En *Asteroids* se estudió con más profundidad que en sus antecesores el tema del **público objetivo**, ya que antes de lanzar el videojuego al mercado, Atari se centró en reunir una serie de jugadores de edad avanzada, mediana edad y jóvenes, para probar el atractivo de su videojuego y cómo éstos reaccionaban a la plataforma de juego. Aunque en la época la calidad gráfica no permitía realizar un juego más atractivo para el género masculino o femenino, por lo que en esta distinción las empresas seguían sin hacer énfasis.

2.4 *Pac Man* (1980)

Pac Man, también llamado Comecocos en España, fue uno de los videojuegos más influyentes en el año 1980, creado por el diseñador Toru Iwatani de la empresa Namco y distribuido por *Midway Games* en América. Su principal diseñador se basó en la forma de una pizza para crear al protagonista del videojuego llegando a cosechar un éxito de gran magnitud, desbancando al famoso y exitoso *Space Invaders* y llegando a considerarse el juego arcade más famoso de los 80. A modo de dato curioso, el nombre del videojuego viene de la palabra japonesa *paku*[10], que una vez en manos de *Midway* fue traducido como *Pac Man* para el mercado occidental. En la Figura 2.11 se puede observar el diseño del protagonista.

Su diseñador quiso incluir una cantidad considerable de niveles, y es que *Pac Man* contiene la friolera de 255 niveles jugables (que es el máximo número que se puede expresar en un número de ocho cifras si nos basamos en el sistema binario), ya que éstos estaban almacenados en un byte. Hay una anécdota acerca de este detalle, ya que los jugadores más experimentados que consiguieron com-

[10]Onomatopeya japonesa que se refiere al sonido que se produce al abrir y cerrar una puerta.

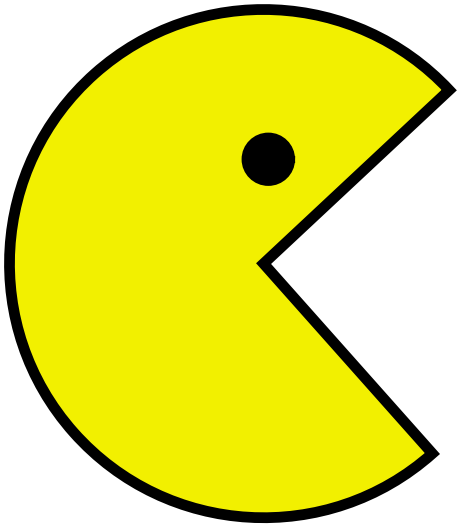

Figura 2.11: Diseño del protagonista del videojuego, en el que se aprecia la similitud con una pizza, la cual tiene un trozo menos.

pletar los 255 niveles, descubrieron que existía un último nivel (al que llamaron «Nivel final» o «Nivel de la pantalla partida»), el cual, debido a que no podía ser representado por el tema que se ha comentado anteriormente, producía un efecto de distorsión en la parte derecha de la pantalla, lo que hacía imposible completar dicho nivel, como se aprecia en la Figura 2.12.

El videojuego consiste en «comerse» todos los puntos que hay en cada nivel, y como dificultad añadida hay cuatro fantasmas (el rojo, azul, rosa y amarillo) los cuales intentan acabar con nosotros para evitar que completemos el nivel. Tanto fue el éxito cosechado por *Pac Man*, que en el año 2007 se celebró el campeonato mundial donde el ganador obtuvo un cuantioso premio en *Microsoft Points* y una consola Xbox 360 firmada por el creador del videojuego, Toru Iwatani, la cuál es única en el mundo. Incluso Google conmemoró su 30º aniversario, incluyendo en la página principal del buscador un nivel de éste que se jugaba en las letras que dan nombre a la empresa estadounidense.

2.4.1. Dispositivos de juego

Pac Man es uno de los pocos videojuegos que lleva recibiendo publicaciones durante más de 3 décadas, lo que ha conllevado el desarrollo de múltiples secuelas para diversas plataformas tanto de consola como PC.

El videojuego original fue diseñado por Toru Iwatani para máquina recreativa, las cuales iban destinadas a los recreativos japoneses, ya que en dicho territorio las consolas no triunfaban demasiado en el público. En la Figura 2.14 se puede observar un ejemplo de máquina recreativa.

Figura 2.12: Nivel final de *Pac Man*, donde se puede observar la distorsión producida en la pantalla.

En occidente, el videojuego fue adaptado a su versión en Atari 2600, aunque fue muy criticado ya que ésta tenía una memoria muy limitada en comparación con las máquinas recreativas. En el año 1985 el juego apareció en la versión para PC de IBM y más tarde en ordenadores de 8 bits, llegando a versiones en la actualidad, como el ejemplo de la versión de Google para la página principal de su buscador.

2.4.2. Otros detalles

Pac Man ha sido uno de los videojuegos más influyentes del género **Arcade**, desbancando al poderoso *Space Invaders*, ya que pasó de la acción del *Shoot 'em up* a un simple sistema que consistía en recoger elementos por el mapa, evitando para ello a una serie de enemigos que intentaban acabar con nosotros, lo que llamó la atención de un gran número de usuarios.

La **dificultad** del videojuego reside en el aumento progresivo de velocidad en los enemigos conforme avanza el tiempo que permanecemos en cada nivel, además de la dificultad progresiva de los mapas donde cada vez se dificulta más al

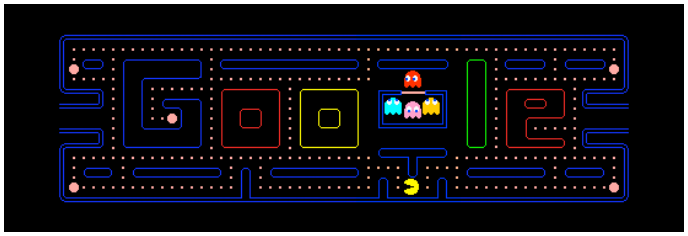

Figura 2.13: *Banner* del videojuego *Pac Man* en la página principal de Google, durante el 30º aniversario del mismo.

Figura 2.14: A la izquierda se puede observar una máquina recreativa del *Pac Man* estilo americano, y a la derecha una de estilo japonés.

usuario la recogida de los elementos que se encuentran en el mapa. Cabe mencionar que es un juego muy complejo, en el que incluso se podría decir que hay jugadores profesionales, los cuales han dedicado muchas horas para llegar a dominar en profundidad el videojuego.

El videojuego estaba dirigido a una amplia variedad de **público**. Se podría decir que fue el primero en intentar captar a varias generaciones de jugadores, y queda patente ya que consiguió ser el videojuego más vendido de la historia en sus años dorados. Además, no tenía una fuerte carga hacia una tipología de videojuego, como podía ser el caso del *Space Invaders* (el cual consistía en disparar a los enemigos y fue precursor del género FPS «First Person Shooter»), ya que consistía en recolectar una serie de elementos en cada uno de los niveles. Estas características lo hicieron atractivo para todo tipo de público, tanto jóvenes como adultos, incluso dentro del género femenino, que tanto se resiste en el mundo

de los videojuegos. En la Figura 2.15 se puede observar la carga femenina del videojuego, en una variante del mismo.

Figura 2.15: Publicidad realizada por la empresa Atari, en la que se puede observar la carga femenina que se quiso incluir en el videojuego *Pac Man*, para atraer un mayor número de público.

2.5 *Donkey Kong* (1981)

Donkey Kong nace a manos del diseñador Shigeru Miyamoto en el año 1981 por la empresa Nintendo. Es considerado el primer videojuego de plataformas y el primero en seguir una historia que va avanzando hasta su final, además fue el primer videojuego en incluir escenas en movimiento. En la Figura 2.16 se puede apreciar el primer nivel del videojuego.

Éste estaba dividido en cuatro niveles, cada uno de los cuales consistía en realizar una cierta tarea para poder pasar al siguiente, y la historia consistía en un fontanero (Mario) que tenía que rescatar una princesa, la cual estaba cautiva

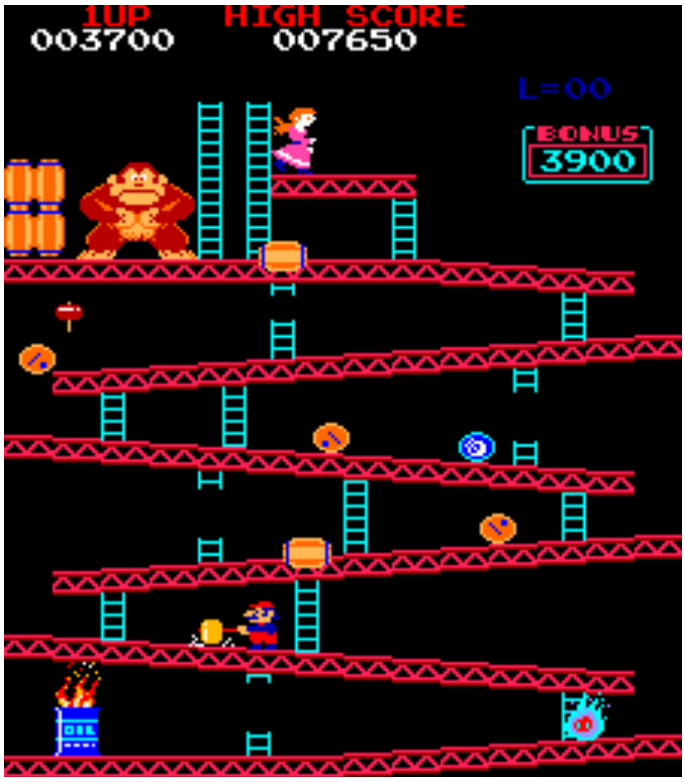

Figura 2.16: Primer nivel del videojuego *Donkey Kong*, el cual consistía en llegar a la cima del mismo esquivando los barriles.

por el malvado de la historia, *Donkey Kong*. Miyamoto se basó en Popeye y *King Kong* para crear el videojuego.

Nintendo puso todos sus esfuerzos en conseguir que el videojuego calara en el mercado americano, y aunque al principio no obtuvo demasiado éxito, para sorpresa de la compañía, más tarde llegaría a ser el juego más jugado y de mayor éxito comercial entre los 80 y los 90. Tanto es así que la mercadotecnia para el videojuego llegó incluso a las cajas de cereales americanas; en la Figura 2.17 se puede observar una de estas cajas.

Como anécdota cabe añadir que la empresa encargada de producir la película *King Kong* denunció a Nintendo por plagio, dando la justicia la razón a la empresa del videojuego nipón. Al igual que *Asteroids*, *Donkey Kong* tuvo tanto éxito que llegó al mundo competitivo, siendo uno de sus campeones el jugador Steve Wiebe, y tanto fue el éxito del videojuego que aún hoy en día se siguen sacando secuelas, en las que el protagonista es *Donkey Kong* y su amigo *Diddy Kong*.

2.5.1. Dispositivos de juego

Originalmente el videojuego fue desarrollado sobre una máquina recreativa que solo permitía la ejecución de éste (disponía de un *joystick* y botones para sal-

Figura 2.17: Una de las cajas de cereales del videojuego *Donkey Kong* como mercadotecnia del mismo.

tar, aceptar etc.), ya que la empresa no tenía claro que fuese a triunfar en el mercado americano. En la Figura 2.18 se puede apreciar una de las primeras máquinas en las que corría el videojuego *Donkey Kong*.

Debido a su gran éxito en el mercado americano, pronto se portó el videojuego a diferentes consolas que permitían al usuario jugar en sus domicilios, como la *Game Watch* o la famosa consola NES[11], fabricada para el público americano. En la Figura 2.19 se puede apreciar uno de los portes que tuvo el videojuego.

2.5.2. Otros detalles

Donkey Kong fue el pionero en su **género**, denominado de «plataformas», ya que hasta el momento ningún otro videojuego del mercado salvo *Pac Man* con-

[11] Más información en `http://es.wikipedia.org/wiki/Nintendo_Entertainment_System`

Figura 2.18: Una de las primeras máquinas recreativas del videojuego *Donkey Kong*, la cual solo permitía jugar a dicho juego.

Figura 2.19: Una consola portátil de Nintendo *Game Watch* con el videojuego *Donkey Kong*.

tenía animaciones y tampoco poseía varios niveles sobre los que ir avanzando cumpliendo para ello una serie de requisitos.

La **dificultad** del videojuego radicaba sobre todo en la puntuación que obteníamos al finalizar cada nivel, ya que conseguir el objetivo en sí para avanzar no era demasiado complejo. Para ello disponíamos de una serie de herramientas que, utilizadas de la manera correcta, otorgaban grandes cantidades de puntuación al usuario, como el martillo del primer nivel, que permitía destruir los barriles y obtener un puntaje sin necesidad de saltar por encima de éstos. En la Figura 2.20 se puede apreciar el objeto usado para destruir los barriles en *Donkey Kong*.

En cuanto al **público** al que iba dirigido el videojuego, cabe decir que fue diseñado para atraer jugadores americanos, ya que la empresa nipona estaba intentando penetrar en dicho mercado, sin demasiado éxito, por lo que éste iba destinado a atraer el mayor número posible de público, tanto masculino como femenino.

Por último cabe destacar una cita del diseñador del videojuego, en la que se resume la historia del mismo: «El gorila se escapa de la jaula, atrapa a la chica y se fuga con ella. Un hombre con un martillo en la mano le persigue esquivando barriles. Al final el gorila cae por un agujero en el suelo y muere. –Shigeru Miyamoto».

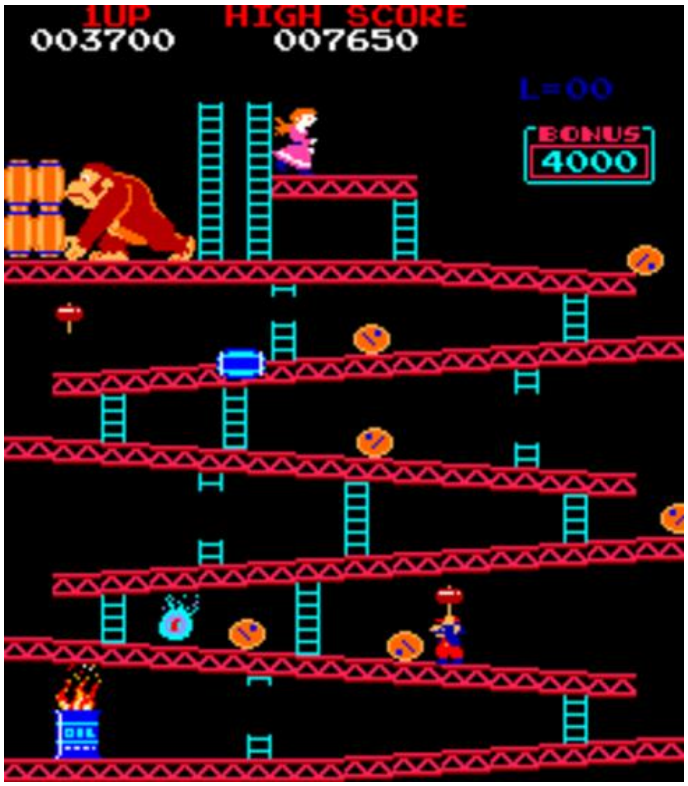

Figura 2.20: El martillo de Mario, uno de los objetos que permitía aumentar la puntuación del juego destruyendo barriles para ello.

CAPÍTULO 3

El entorno de programación Scratch

En el capítulo se describe el entorno de programación Scratch, así como el porqué de su elección para realizar los videojuegos escogidos en el trabajo.

3.1 ¿Por qué Scratch?

Se ha decidido la utilización del lenguaje de programación Scratch (Figura 3.1) por las siguientes razones:

Figura 3.1: Logo y lema de la plataforma Scratch: «Imagina, crea, comparte».

- **Facilidad de uso**. Scratch ahorra mucho tiempo a los usuarios noveles en aprender la sintaxis correcta de un lenguaje de programación tradicional, ya que se basa en el arrastre de bloques que conectan entre sí a la hora de añadir funcionalidad a nuestro programa.
- **Variedad bibliográfica**. Hay una gran cantidad de material bibliográfico para la actual versión de Scratch (la 2.0), gracias al cual se facilita la labor de iniciación a usuarios que deseen entrar en el mundo de la programación.
- **Código compartido**. Todos los programas realizados en la web de Scratch permiten la visualización de su código por parte de cualquier usuario, incluso permite crear una nueva versión del programa. Con esto enriquecemos

tanto al creador como a los usuarios que desean iniciarse en el mundo de la programación.

- **Entorno de programación**. Scratch es un lenguaje de programación que además ofrece un entorno el cual facilita la tarea a las personas sin amplios conocimientos en la materia, además permite compartir el código de manera rápida con otros usuarios. De ahí el lema del lenguaje «Imagina, programa, comparte» tal como se puede observar en la Figura 3.1.

3.2 Proyecto Scratch

Scratch es una plataforma multimedia de programación con versión de escritorio y *online* (consultar [7] para informarse acerca de la plataforma y como iniciarse en el lenguaje). Su público se centra sobretodo en estudiantes, escolares y profesores, los cuales aprenden a crear de manera sencilla juegos que les permiten avanzar en sus conocimientos sobre el arte de programar, pensamiento creativo, mejorar su razonamiento y trabajar colaborativamente. Se puede acceder al contenido online a través de la página `scratch.mit.edu`, cuya portada se encuentra visible en la Figura 3.2.

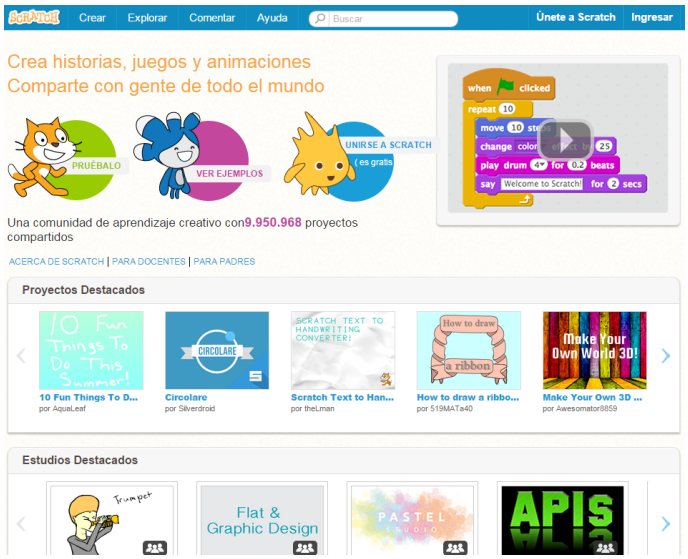

Figura 3.2: Portada de inicio de la página web de Scratch.

El proyecto Scratch nace en el año 2003 a manos del grupo de investigación *Lifelong Kindergarten* [16] del MIT, liderado por Mitch Resnick. La idea surgió ya que el grupo pensaba que era muy importante para un niño crecer aprendiendo a diseñar y expresarse, ya que hoy en día muchos de ellos no suelen tener esa oportunidad y además Resnick se basó en las ideas de Seymour Papert y el concepto de bloques de Lego, los cuales ayudaron en la creación de éste (Figura 3.3).

Figura 3.3: Mitchel Resnick con un montón de bloques de Lego (inspiración para el desarrollo de Scratch).

El término «scratch» proviene del nombre de una técnica de hip hop llamada «scratchering», mediante la cual los disc-jockeys realizan modificaciones en la melodía de una canción a base de reproducir discos de vinilo hacia delante y hacia atrás con las manos, mezclando para ello diferentes clips de música. Scratch intenta algo similar, ya que mezcla lo gráfico, animaciones, imágenes, música y sonido de manera interactiva [17].

Scratch es un proyecto de desarrollo cerrado y de código abierto. Esto significa que el equipo no persigue la contribución de la comunidad de usuarios, sino que elaboran enteramente el proyecto estándar. Por otro lado, se propone el código abierto ya que el equipo pretende liberar el código en un futuro para que la comunidad pueda experimentar con el desarrollo de extensiones y modificaciones sobre el programa.

En marzo del año 2007 la página web de Scratch, junto a su nueva versión (la 2.0), fue completamente rediseñada para introducir el componente social en la misma. A partir de entonces los usuarios empezaros a compartir sus proyectos y a poder visualizar los del resto de usuarios de la comunidad, que se definieron con el nombre de *Scratchers*. En la Figura 3.4 se puede observar el proyecto de un usuario de la comunidad desde dentro, y la posibilidad de reinventarlo, para añadir funcionalidad al mismo.

3.3 Finalidad de Scratch

Scratch fue desarrollado con la finalidad de que usuarios nuevos en el arte de la programación pudieran aprender de manera didáctica y divertida. Para ello su principal enfoque era el componente social, ya que si un usuario dispone de la

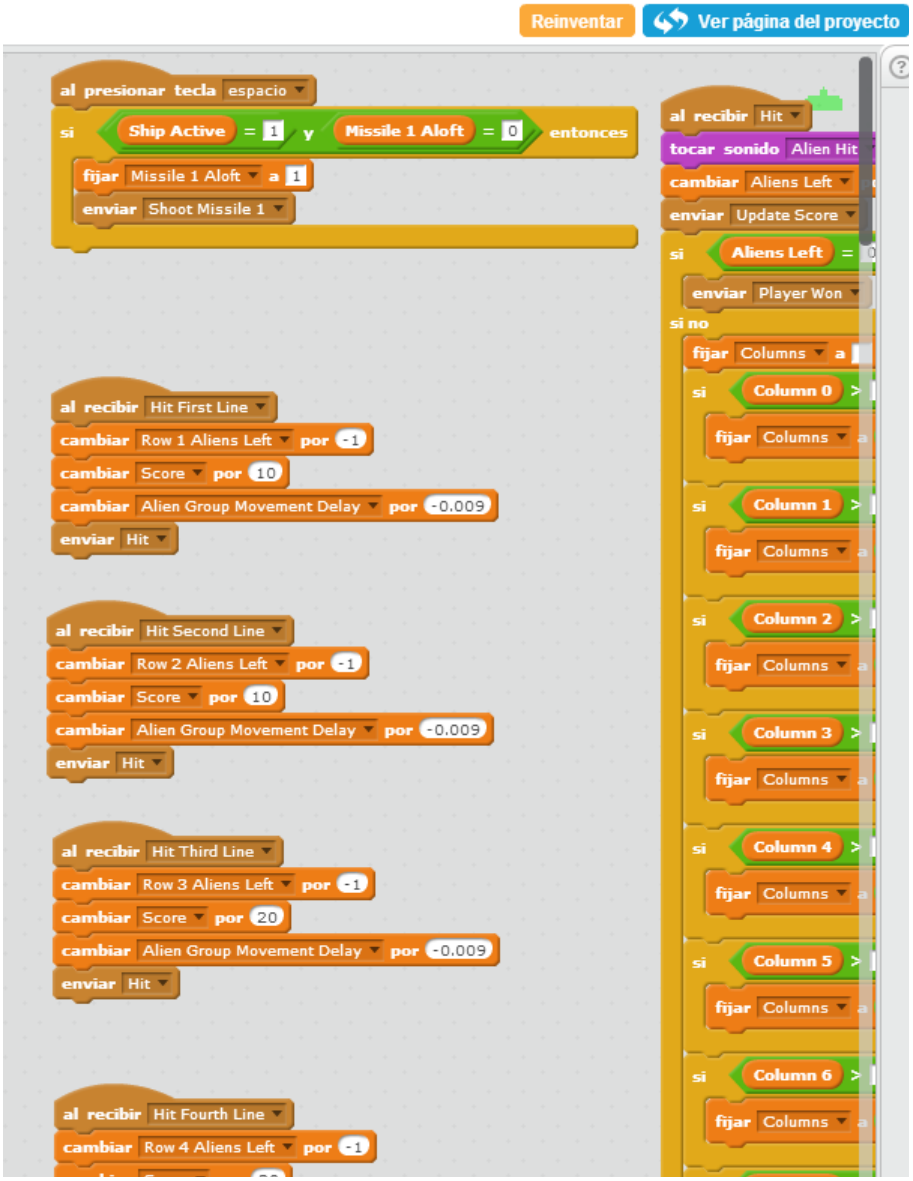

Figura 3.4: Ventana de visualización de un proyecto compartido en Scratch, con posibilidad de reinvención por parte de los usuarios.

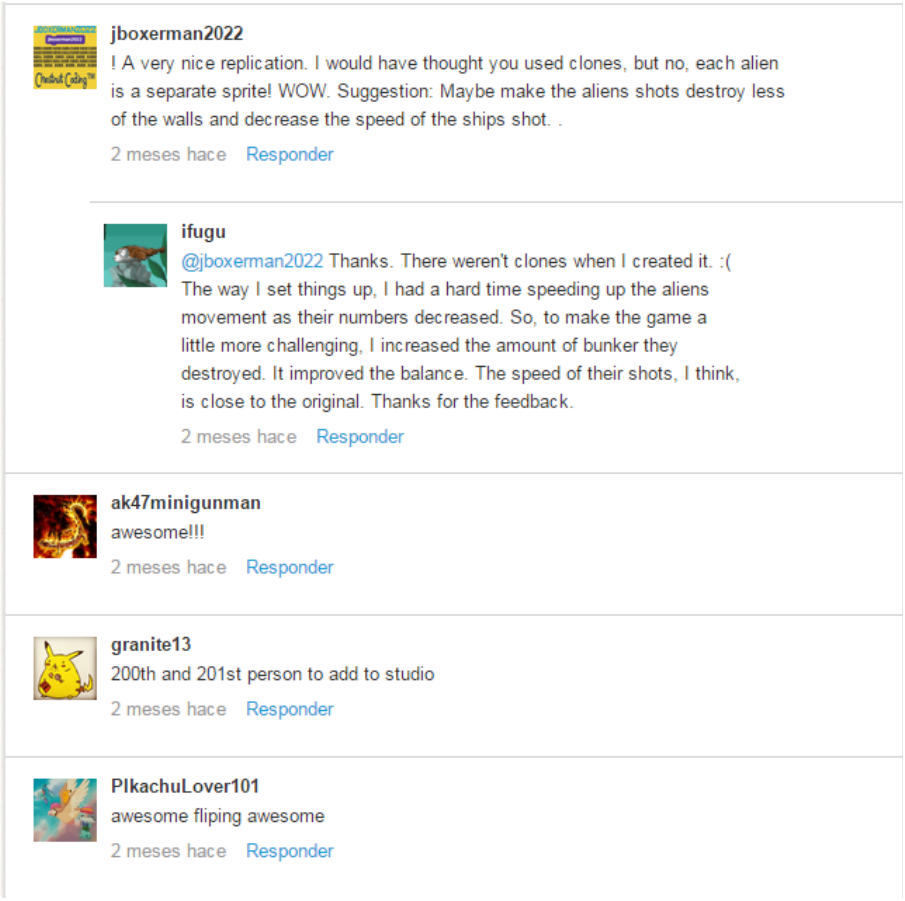

Figura 3.5: Espacio del proyecto en el que tanto usuarios como creadores pueden añadir comentarios.

opción de ver el código que otro ha desarrollado, y además puede mejorarlo o añadir funcionalidad, el aprendizaje se hace más ameno y divertido.

Las últimas actualizaciones del lenguaje Scratch y del componente *online* del mismo han ido totalmente enfocadas a la parte social, ya que se pasó a disponer de un mecanismo para compartir los proyectos y facilitar a los usuarios su acceso y modificación. Además actualmente Scratch está disponible en más de 40 idiomas, y éstos siguen aumentando, ya que es su principal fuente de socialización, además una de las mejores inclusiones es la de añadir comentarios a un proyecto en concreto, ya que esto permite tanto al creador como al usuario que lo visita el intercambio de información, lo que enriquece el aprendizaje de ambos y de toda la comunidad (Figura 3.5).

3.4 Scratch y el pensamiento computacional

El concepto de *computational thinking* o pensamiento computacional se define como «Procesos de pensamiento involucrados en formular problemas y encontrar soluciones, de manera que las éstas estén representadas tal que puedan llevare a cabo por un agente que procesa información (humano o máquina)». En [21] se puede consultar más información sobre el pensamiento computacional, creado por Jeannette M. Wing.

El pensamiento computacional de Scratch se basa en los siguientes conceptos: secuencia, iteración, paralelismo, eventos, condicionales, operadores y datos.

Además se contemplan ciertas prácticas sobre el pensamiento computacional:

- Ser incremental en la búsqueda de soluciones: divide y vencerás.
- Probar y depurar: nada sale a la primera, se cometen y corrigen errores.
- Reusar y remezclar: no hacemos todo desde cero.
- Abstraer, modelar y modularizar: creamos modelos para gestionar la complejidad.

Estás prácticas se enfocan en el proceso de pensar y aprender y se centran en cómo lo estás aprendiendo, y no en qué estás aprendiendo.

Todo lo anterior se podría resumir en que el pensamiento computacional es una habilidad para resolver problemas complicados de manera algorítmica que, además, ayuda a mejorar la eficiencia de los procesos. El concepto fue usado por primera vez por Seymour Papert en 1996, y no solo se aplica a la informática, sino a todas las disciplinas existentes.

Scratch ayuda al aprendizaje mediante pensamiento computacional ya que está enfocado a actividades de diseño, ofreciendo para ello un amplio abanico de herramientas, las cuales ayudan a resolverlas.

CAPÍTULO 4

Diseño de videojuegos con Scratch

En el presente capitulo se va a exponer porque el lenguaje de programación Scratch tiene ciertas ventajas a la hora de realizar videojuegos, y como éste y su entorno pueden ayudar a los usuarios con la tarea de creación y organización del mismo.

4.1 Panel de objetos

La primera característica que ofrece, y de cara al usuario la más importante de todas, es el panel en el que aparecen todos los objetos del programa, ya que frente a otros lenguajes de programación, Scratch busca sobre todo que el usuario tenga localizados en todo momento los fragmentos de código y su correspondiente objeto. Esta es una característica que hace que, con Scratch y su entorno de programación, un usuario que apenas tiene conocimientos sobre creación de videojuegos tenga ordenados los objetos del escenario y que no sean solo letras o variables, sino que además aparezca una imagen para reconocerlos (Figura 4.1). Además dispone de la opción de crear un objeto desde cero, añadirlo desde el ordenador o usar los ya existentes en el propio lenguaje.

En dicho panel se puede pulsar en uno de los objetos disponibles, y lo que obtenemos es otro panel a la derecha de la pantalla con tres pestañas (Programas, Disfraces y Sonidos), que se describirán en los siguientes apartados del capítulo, en el que se pueden observar los fragmentos de código que maneja dicho objeto. Además desde el panel de objetos, se nos ofrecen varias opciones, como la de duplicar dicho objeto, borrarlo o incluso acceder a la información de éste, la cual nos indica la rotación del mismo, la posición, etc. (Figura 4.2).

Por último, en dicho panel se encuentra el objeto que define por defecto Scratch denominado «Escenario», el cual maneja lo que ocurre en la escena del videojuego y se encarga de colocar los fondos del mismo (que van a modo de disfraces). Una de las funciones que puede cumplir dicho objeto es la de ser el *Game Manager*,

Figura 4.1: Panel de objetos en el cual se encuentran todos los objetos del proyecto actual en Scratch.

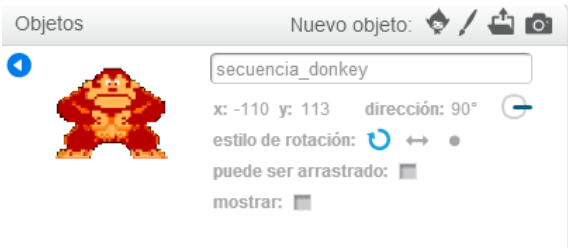

Figura 4.2: Opción del panel de objetos, que nos permite visualizar la información básica de uno de los objetos.

el cual se encarga de la inicialización de las variables globales y de controlar los estados de victoria y derrota, así como navegar a los escenarios correspondientes en cada momento.

4.2 Panel de programas

Como su propio nombre indica, es el lugar donde se desarrolla el comportamiento de un objeto determinado. A diferencia de otros lenguajes de programación, en Scratch no se utilizan líneas y líneas de código, sino que este incluye unos bloques predefinidos por el lenguaje, parecidos a los bloques de LEGO, los cuales simulan las características básicas de un lenguaje de programación, en contrato

dispone de los siguientes (en la esquina izquierda de la Figura 4.3 se pueden observar todos los tipos de bloques que proporciona Scratch).

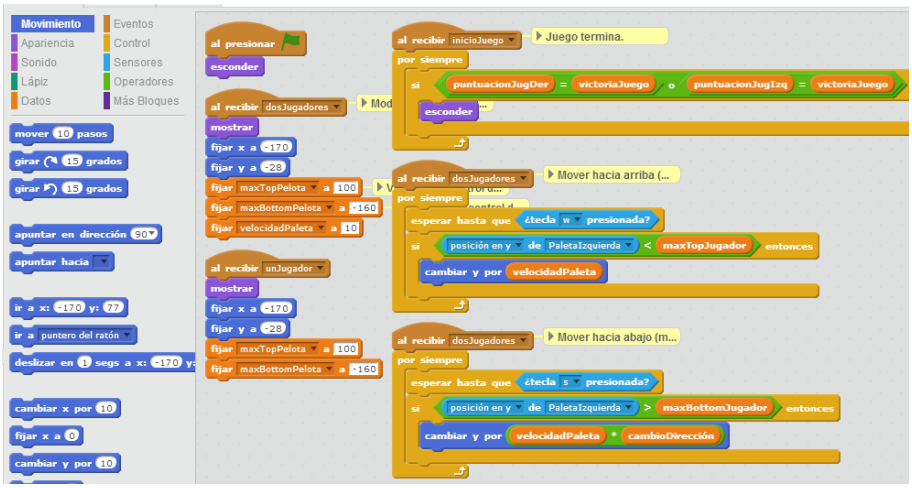

Figura 4.3: Panel de programas en el cual se encuentras todos los fragmentos de código que dan comportamiento a un objeto determinado.

- **Bloques de movimiento**: Son los encargados de desplazar y detectar desplazamientos del objeto dentro del escenario del videojuego. Con ellos podemos desde indicar la posición actual del objeto, hasta trasladarlo a una posición o incluso posicionarlo en diferentes ángulos.
- **Bloques de apariencia**: Son los encargados de manejar los disfraces que dispone un objeto. Con ellos se puede conseguir que un objeto aparente poseer movimiento propio, o incluso realizar acciones del tipo lanzar barriles, saltar etc., incluso podemos ocultar el objeto y mostrarlo cuando se desee.
- **Bloques de sonido**: Manejan los canales de sonido del proyecto dentro del escenario. Con ellos podemos aumentar el volumen, detener los sonidos, reproducir un sonido indefinidamente etc. Incluso podemos generar sonidos básicos como el sonido de un tambor, flauta, saxofón etc.
- **Lápiz**: Este objeto tiene una funcionalidad curiosa, ya que nos permite dibujar sobre el escenario desde el centro del objeto actual. Sus fragmentos de código permiten desde bajar y subir un lápiz de dibujo, hasta cambiar el color de las líneas.
- **Datos**: Son los bloques de Scratch que permiten definir las variables de los objetos. Estas pueden ser de varios tipos (cadena, enteros, booleanos etc.) y se pueden definir de manera privada para un objeto, o de manera pública para que varios objetos puedan acceder a ellas y modificarlas.
- **Eventos**: Son los encargados de enviar señales a todos los objetos disponibles en el proyecto. Las hay desde cuando empieza la ejecución del programa hasta señales propias que se pueden definir. Básicamente cuando se crea

una señal, hay un objeto que la genera y otros objetos que están a la espera de recibirla y realizar una tarea predefinida con ella, como puede ser la ejecución de un fragmento de código que mueve un objeto, la modificación de una variable, etc.

- **Control**: Proporcionan los bloques de control del lenguaje Scratch. Con ellos se pueden generar fragmentos de código del tipo si-entonces-sino, bucles o incluso fragmentos de espera, en los que la ejecución se detiene los segundos que se indiquen. Gracias a esta funcionalidad podemos obviar lo que se denominan hilos de ejecución, ya que en Scratch es totalmente transparente para el usuario. En un lenguaje de programación convencional como pueda ser Java, si se pretende desarrollar un videojuego, hay que tener muy en cuenta este tipo de situaciones, ya que a la hora de ejecutar fragmentos de código de espera, se procede a poner el hilo actual a «dormido» la ejecución del programa se detiene y no se podrían manejar varios objetos a la vez sin la inclusión de nuevos hilos que realizasen dicha tarea.

- **Sensores**: Los sensores nos permiten detectar ciertas situaciones dentro de la ejecución del programa. Scratch proporciona varios bloques de detección de colisiones, lo que facilita enormemente la tarea de crear un videojuego, ya que la interacción entre objetos es un tema bastante difícil de tratar en un lenguaje convencional y podría ser una barrera demasiado grande para que un usuario junior no pudiera desarrollar un videojuego.

- **Operadores**: Gracias a los operadores, los usuarios pueden desarrollar fragmentos de código de verificación de situaciones, suma de una variable, unión de varias palabras dentro de una variable, etc.

- **Más bloques**: Gracias a este objeto, lo que parece ser un lenguaje muy cerrado y muy limitado a los bloques de código que ofrece, amplía sus posibilidades, ya que se puede dar rienda suelta a la habilidad de cada uno para crear bloques de código propios, utilizando para ellos un conjunto de los demás bloques. En la Figura 4.4 se describe una nueva funcionalidad para incluir objetos en una lista, los cuales definen los «bonus» del videojuego *Space Invaders*.

Por último, la funcionalidad concreta que proporciona este panel es la de programar el comportamiento del objeto en cuestión. Scratch proporciona muchas facilidades al usuario *junior*[1], ya que los distintos bloques poseen un color que los identifica, para que el usuario pueda familiarizarse de manera más sencilla con la funcionalidad de los bloques, y al llegar a cierto fragmento de código reconocer la tarea que éste desempeña dentro de la ejecución del objeto. Además el hecho de evitar la escritura de más y más líneas de código hace que el usuario no pierda el tiempo en aprender un lenguaje en concreto, y se centre en arrastrar y soltar bloques que realizan una tarea determinada.

[1] Usuario sin conocimientos en la materia o que acaba de iniciarse en ésta

Figura 4.4: Nuevo bloque definido en Scratch, con la funcionalidad de añadir las variables de tipos «bonus» en el videojuego *Space Invaders*.

4.3 Paneles de media

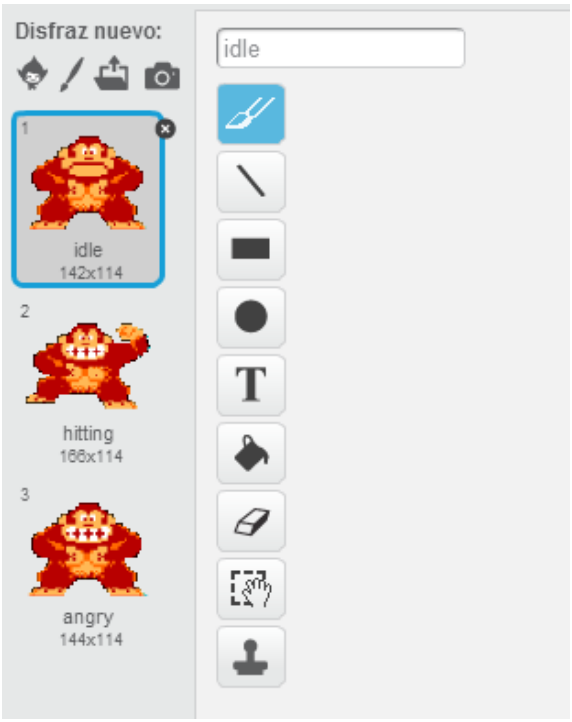

Figura 4.5: Panel de disfraces de uno de los objetos del videojuego *Donkey Kong* en Scratch.

Como última categoría de herramientas que ofrece Scratch para facilitar la vida al programador novicio, se encuentran los paneles de Disfraz y de Sonido. El primero sirve como herramienta para editar los disfraces de un objeto concreto, ya que proporciona una ventana en la que podemos editar con total libertad el color, la forma, el tamaño, etc. de dicha imagen. Aunque la característica más importante que nos ofrece este panel es la de controlar la cantidad de disfraces de un objeto, y gracias a los bloques de apariencia, podemos realizar tareas tales

Figura 4.6: Panel de sonidos con el que podemos manejarlos y ordenarlos en el proyecto.

como la de simular el movimiento del protagonista del videojuego. Lo único que debemos realizar es colocar la secuencia de disfraces que simulan dicha acción, y una vez en el panel de programas utilizar el bloque de siguiente disfraz con un tiempo de retardo entre medias para que se aprecie el cambio de imagen. Sin esta herramienta, realizar semejante acción costaría a un usuario sin mucha práctica en la programación de videojuegos demasiado tiempo. En la Figura 4.5 hay un ejemplo de objeto con los disfraces en orden, los cuales se usarán para dar un efecto de movimiento al objeto.

El otro panel que completa el apartado es el de sonido, el cual facilita la vida al programador ofreciendo una herramienta de edición de sonidos muy sencilla pero a la vez útil para aquellos sin conocimientos en este campo. Y es que la herramienta permite subir cualquier sonido y con ello acortar la duración del mismo, copiar una porción, borrar, etc. Gracias a esto podemos controlar la ejecución de los sonidos dentro del panel de programas mediante el uso de los bloques diseñados para ello, lo que hace que la engorrosa tarea de generar canales y controlar la ejecución de los mismos en un lenguaje convencional sea un juego de niños en Scratch (Figura 4.6).

Para terminar con las características de Scratch vamos a hablar del Maletín, ya que es una herramienta muy interesante y que todo usuario debería terminar utilizando, ya que añade una característica más al grado de compartir código en la plataforma, y es con el simple movimiento de coger un bloque y soltarlo dentro del maletín, obtenemos una copia de éste para usar en otros proyectos, incluso podemos obtener bloque del proyecto de otro usuario y usarlo para el nuestro.

4.4 Metodología de diseño

En esta sección se va a hablar de los pasos que se han seguido para realizar los cinco videojuegos que trata el presente trabajo, así como las decisiones que se

han tomado en ciertas características y cómo se han llevado a cabo las pruebas para detectar *bugs*[2].

Lo primero que se ha realizado ha sido la documentación de cada uno de los videojuegos, de sus estilos y sus peculiaridades, para así dar paso a jugar el videojuego original, ya que en internet se encuentran disponibles y así poder ver en que centrarnos a la hora de realizar cada uno de ellos en Scratch (también es interesante hacer uso de la herramienta que proporciona el masivo entorno para compartir código de Scratch y tomar ideas de cómo realizar cierta funcionalidad de un videojuego concreto).

Una vez documentados acerca de funcionalidad y peculiaridades del videojuego a desarrollar, hay que centrarse en el diseño del mismo (pensar en los objetos que lo van a componer, acciones de cada uno de ellos etc.), esto puede ser muy evidente o necesitar para ello un desarrollo más complejo. Por ejemplo, a la hora de desarrollar el videojuego *Pong*, la parte de desarrollo no conllevaba más que pensar en un círculo que al colisionar con uno de los dos rectángulos (las paletas de los jugadores) o tocar el borde superior/inferior de la pantalla rebotase, y si toca el borde izquierdo/derecho aumentar en uno la puntuación del jugador que ha devuelto la pelota. Por lo que el diseño apenas significó esfuerzo y se pudo pasar directamente a la programación de bloques para dar funcionalidad al mismo (tampoco se hace uso de disfraces para dar sensación de movimiento). Sin embargo, a la hora de desarrollar el videojuego *Donkey Kong*, se tiene que tener en cuenta que el objeto del protagonista (Mario), tiene que provocar sensación de movimiento al usuario al moverlo de izquierda a derecha, tenemos que simular que al subir por una escalera éste hace la acción de agarrarse a las barandillas de la misma, sensación de salto etc. por lo que necesitamos definir los distintos estados por los que va a pasar dicho objeto, incluso como va a interactuar con los demás objetos del escenario.

Por ejemplo, a la hora de desarrollar el videojuego *Space Invaders*, la nave del usuario estaba pensada para que tuviera sensación de fondo en el espacio al movernos (este efecto se explica con más detenimiento en el capítulo de descripción del videojuego), pero a la hora de desarrollar éste, no se comenzó incluyendo dicho efecto, sino que primero se dibujó a mano un triángulo (lo que vendría siendo la nave) y se le dieron las características necesarias para poder dirigirnos a la izquierda o derecha del escenario, poder disparar etc. y una vez completadas, se pasó a diseñar el efecto mencionado, ya que si intentamos hacer todos los pasos de golpe se puede terminar con una funcionalidad no deseada o poco eficiente.

Si seguimos con el desarrollo, después del diseño del videojuego ya se puede pasar a la creación de cada uno de los objetos que lo compondrán. Como peculiaridades del diseño, hay que pensar que el presente trabajo va dirigido sobre todo a la iniciación en la programación de videojuegos, por lo que algunas pautas para el desarrollo de bloques han sido las de nombrar las variables con nombres que

[2]Errores o fallos en un programa de computador o sistema de software que desencadena un resultado indeseado.

las definan, intentar hacer uso de la característica de Scratch para crear secuencias de instrucciones y nombrarlas con un nombre descriptivo, separar en varios bloques una misma funcionalidad para que sea más fácil de identificar (ya que mediante el uso de señales, se puede entender mejor la manera de activar cada uno de éstos) etc.

Por último, una vez terminado el videojuego se pasa a la fase de pruebas del mismo, ya que, aunque durante el desarrollo se intentan eliminar todos los *bugs* que se encuentran, al final de éste hay que asegurarse que el acabado es robusto y no presenta errores, o éstos apenas influyen en la jugabilidad y resultado del mismo. Para ello, al no disponer de un equipo de *testers* que realicen dicho trabajo, se ha recurrido a dos usuarios a los que se les ha indicado unas pautas a seguir y sobretodo qué partes del videojuego deben probar, si se detecta un *bug*, el *tester* simplemente hace una captura de pantalla de la situación en la que ha quedado el videojuego y se hace una breve descripción de los pasos para producirlo, así se puede corregir y mejorar la funcionalidad de los videojuegos.

CAPÍTULO 5

Diseño e implementación de *Pong*

En el siguiente capítulo se va a describir el proceso seguido para desarrollar los objetos del videojuego *Pong* con el uso de Scratch. También se incluye un apartado sobre posibles ampliaciones que se podrían implementar.

5.1 Organización del menú

El menú del videojuego *Pong* es manejado por doce objetos (se puede observar en la Figura 5.1), los cuales nos permiten dirigirnos a las diferentes pantallas que disponemos dentro del mismo. A continuación se va a describir cada uno de ellos:

1. **Un jugador:** Se encarga de mandar la señal de «unJugador» e «inicioJuego» a todos los objetos del videojuego para que la paleta de la izquierda detecte que debe aplicar inteligencia artificial en lugar de movimiento por jugador, y que todos los programas pertinentes se pongan en ejecución a partir de hacer click en el objeto.

2. **Dos jugadores:** Se encarga de mandar la señal de «dosJugadores» e «inicioJuego» a todos los objetos del videojuego para que las paletas detecten que deben permitir el movimiento de jugadores y que sus programas pertinentes comiencen a ejecutarse. Además se encarga de inicializar el cronómetro de la partida, ya que será usado para aumentar la velocidad de la pelota y así dificultar la partida a los jugadores.

3. **Opciones:** Es el objeto encargado de llevar al usuario a la pantalla de opciones, la cual dispone de los objetos «Dificultad» y «Sonido», para ello envía la señal «Opciones», la cual, al ser detectada por el objeto contexto (el encargado de los fondos en Scratch), cambia el fondo al de opciones y activa los dos objetos arriba mencionados.

Figura 5.1: Menú principal del videojuego *Pong*, así como los distintos objetos que nos transportan por las distintas pantallas del mismo.

4. **Instrucciones:** Es el objeto encargado de llevar al usuario a la pantalla de instrucciones del juego, la cual simplemente es una pantalla explicativa de la funcionalidad que dispone el videojuego, como puede ser la tecla a usar para mover al usuario de la izquierda, o cuando termina el juego. Para ello envía la señal «Instrucciones», la cual al ser detectada por el objeto contexto, permite el cambio de fondo y la ocultación de los demás objetos del menú.

5. **Volver:** Este objeto se encuentra en las pantallas de Opciones, Instrucciones y Créditos, y se encarga de permitir la vuelta al menú desde dichas pantallas, envía la señal «volverMen» la cual hace que se ejecuten los fragmentos en los demás objetos que están a la espera de la misma (vuelve a mostrar los objetos del menú).

6. **Dificultad:** Este objeto se encuentra ubicado en la pantalla de opciones y es el encargado de manejar la dificultad de la paleta de la izquierda para el modo de un jugador. Por defecto se encuentra a «Fácil», si pulsamos en el objeto, éste cambia a «Normal». Cada uno de los fragmentos que manejan la inteligencia artificial de la paleta se encuentran en el objeto «PaletaIzquierda».

7. **Sonido:** Este objeto se encuentra ubicado en la pantalla de opciones y es el encargado de manejar si el sonido se encuentra a «On» o a «Off» en el videojuego, por medio de la variable sonido, la cual se encuentra a cero por defecto (si hay sonido), y a uno si no hay sonido disponible. Envía las señales «sonidoOn» y «sonidoOff» para que otros objetos ejecuten los fragmentos que están a la espera de las mismas.

8. **Créditos**: Es el objeto encargado de mostrar al usuario los participantes en el proyecto, así como su creador y tutor. Envía la señal «Créditos», la cual se encarga de iniciar la animación de los textos de créditos, ocultar los demás objetos del menú y de llevar al usuario al Escenario correspondiente.

9. **Textos créditos**: Son tres objetos, que una vez reciben la señal «Créditos», van apareciendo por la parte superior del escenario en orden, y terminan colocados uno debajo del otro, mostrando una descripción de los participantes en el proyecto.

5.2 Objetos del videojuego

El videojuego diseñado dispone de un total de quince objetos, de los cuales ocho se encuentran funcionando en la parte de menú (y han sido descritos en el primer apartado) y los otros siete objetos se encargan del funcionamiento del juego. A continuación se va a describir el funcionamiento de cada uno de ellos.

1. **GameManager**: Este objeto se encarga de inicializar correctamente las variables de inicio de cada partida, en la Figura 5.2 se puede observar su fragmento de código. En los posteriores videojuegos se ha decidido usar de *GameManager* al propio contexto del videojuego (el objeto de Scratch que maneja los fondos).

Figura 5.2: En la imagen se puede apreciar el fragmento de código que se encarga de inicializar las variables al inicio de cada partida del videojuego *Pong*.

2. **PaletaIzquierda**: Este objeto es el que hace las veces de jugador de la izquierda y es manejado por la inteligencia artificial cuando escogemos la opción de «un jugador» en el menú de juego. En la Figura 5.3 podemos observar el fragmento de código que maneja el movimiento de la paleta (si escogemos la opción de dos jugadores), cuando pulsamos la tecla «W» ésta

cambia el valor de su «eje y» positivamente (por lo que subimos), y cuando pulsamos la tecla «S», cambia negativamente el valor del «eje y», por lo que descendemos. Cabe mencionar que si llegamos a los bordes definidos por las variables «maxTopPelota» y «maxBottomPelota», la paleta se detiene y no puede seguir avanzando en dicha dirección.

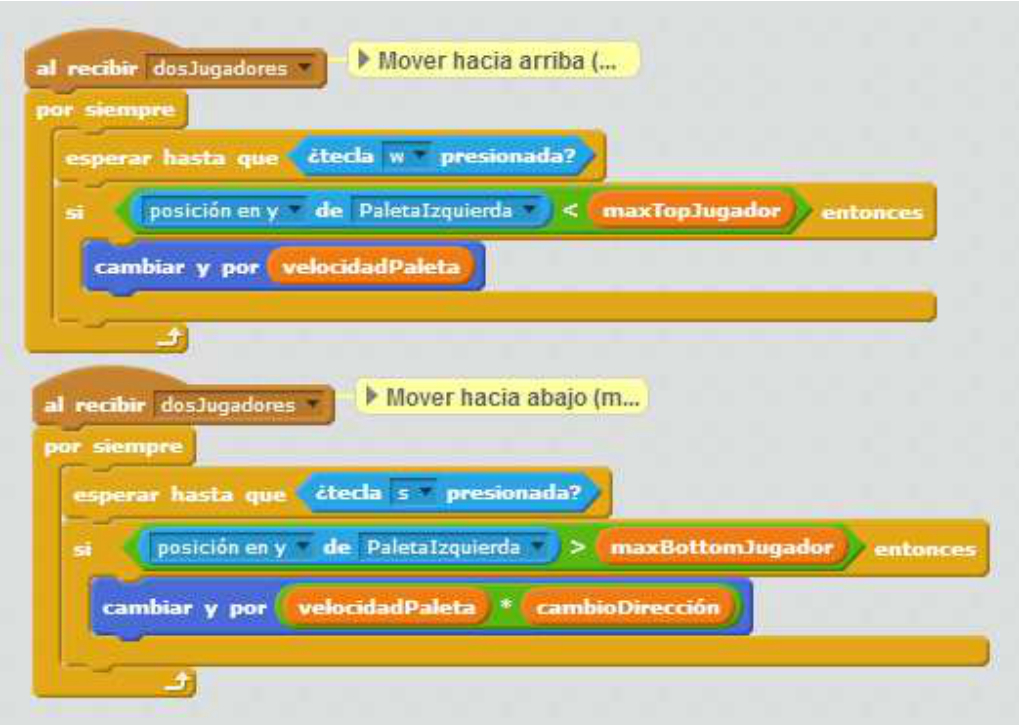

Figura 5.3: Fragmento de código que maneja el movimiento de la paleta izquierda o jugador de la izquierda.

Si escogemos la opción para un jugador, en el menú de opciones está disponible la variable que maneja la dificultad de la paleta (podemos elegir entre fácil y normal). La inteligencia artificial se basa en la posición de la pelota para moverse y devolverla al jugador, en el apartado de ampliaciones se ha descrito lo que podría ser el modo difícil, en la Figura 5.4 se puede observar el manejo de código que maneja la paleta. La inteligencia artificial también detecta si se han llegado a los límites de la pantalla, para evitar que ésta siga avanzando en dicha posición.

3. **PaletaDerecha**: El código que maneja la paleta de la derecha o jugador de la derecha, es el mismo que el que se encarga del movimiento en la paleta de la izquierda, salvo que para éste cuando pulsamos la tecla «Arriba» es cuando subimos, y cuando pulsamos la tecla «Abajo» es cuando bajamos. No tiene código de manejo de inteligencia artificial, ya que ésta solo nos interesa para el modo de un jugador.

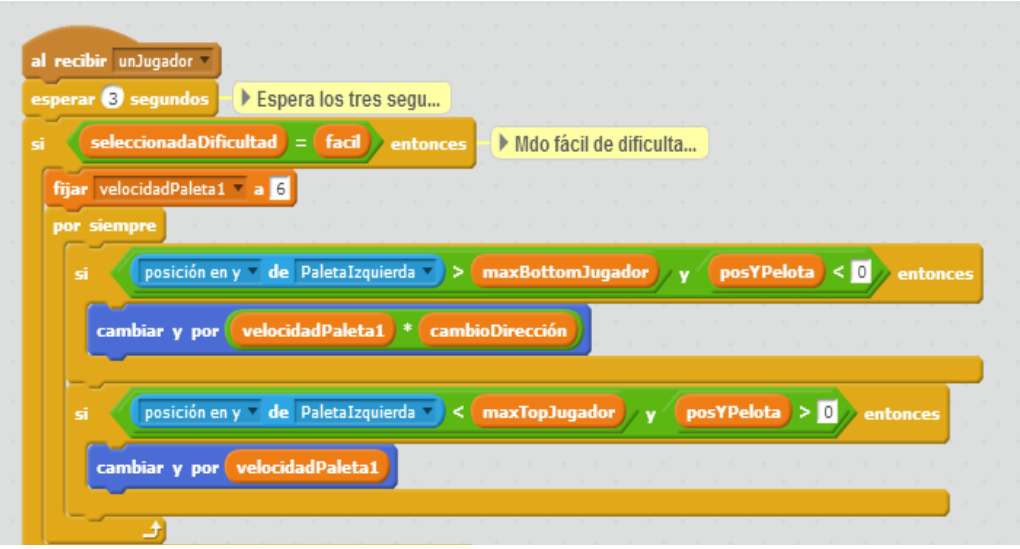

Figura 5.4: Fragmento de código que maneja todo lo relacionado con las puntuaciones del lado izquierdo y derecho.

4. **Pelota**: El objeto «Pelota» se encarga de manejar todo lo que concierne al movimiento de la pelota y el aumento de puntuaciones. Cuando la partida acaba también se encarga de enviar la señal «finJuego», la cual indica al contexto de Scratch, que debe cambiar el fondo al modo victoria del jugador que ha ganado y detiene la ejecución de los objetos que participan en el juego. Nos vamos a centrar en el fragmento de código que maneja el aumento de puntuación, se puede observar en la Figura 5.5, como cuando la pelota llega al borde izquierdo o derecho (es decir, que el jugador de ese lado no consigue devolverla), dicho fragmento se encarga de enviar una señal para aumentar la puntuación y que el marcador cambie en la pantalla (es lo más parecido al patrón observador[1] de diseño) además si se quiere consultar información adicional sobre patrones de diseño se puede consultar el libro de referencia de Erich Gamma en [3]. Por último el fragmento envía la señal de «resetPaleta», la cual ejecuta otro fragmento de código que se encarga de resetear la pelota y que vuelva a comenzar el juego.

   Otros fragmentos de código interesantes sobre el objeto «Pelota» son el que se encarga de manejar los rebotes de la misma contra la pared (cabe mencionar que para evitar que la pelota se cuele en las texturas, se ha tenido que utilizar una variable que maneja el error al detectar la colisión, la cual mueve ligeramente la pelota hacia la posición contraria en la que ha rebotado), el que se encarga de aumentar la velocidad de la misma (únicamente para el modo de dos jugadores), y que una vez alcanzada cierto aumento produce un sonido de sirena y el cambio constante de color rojo y blanco en la pelota, y el fragmento que maneja la victoria del juego, que se encarga

[1] Se puede encontrar más información en `http://www.seas.es/blog/informatica/patrones-de-diseno-en-java-patron-observer/`

Figura 5.5: Fragmento de código que detecta las puntuaciones y maneja el proceso de puntuación.

de asignar al vencedor y de detener todos los programas del objeto. En la Figura 5.6 se puede observar dicho fragmento.

5. **Marcadores**: Se dispone de un objeto que maneja el marcador de la izquierda y otro que maneja el marcador de la derecha, aunque ambos realizan la misma tarea, y es la de cambiar el número del marcador cada vez que uno de los jugadores puntúa (a través de la señal que envía el objeto «Pelota»). El fragmento de código que maneja el cambio de disfraces del mismo simplemente cambia el disfraz del objeto (cada disfraz es un número del cero al nueve).

5.3 Posibles ampliaciones

El videojuego ha sido prácticamente completado con todas las opciones del original, aunque hay un apartado del mismo que no se ha desarrollado al máximo, el apartado de la inteligencia artificial, ya que debido al tiempo disponible para el desarrollo solo se han desarrollado las dificultades «Fácil» y «Normal».

La idea de mejora para el videojuego sería la de desarrollar una inteligencia artificial «Difícil», la cual fuese prácticamente imbatible, para jugadores muy avanzados. Una idea de posible desarrollo sería que la paleta manejada por la

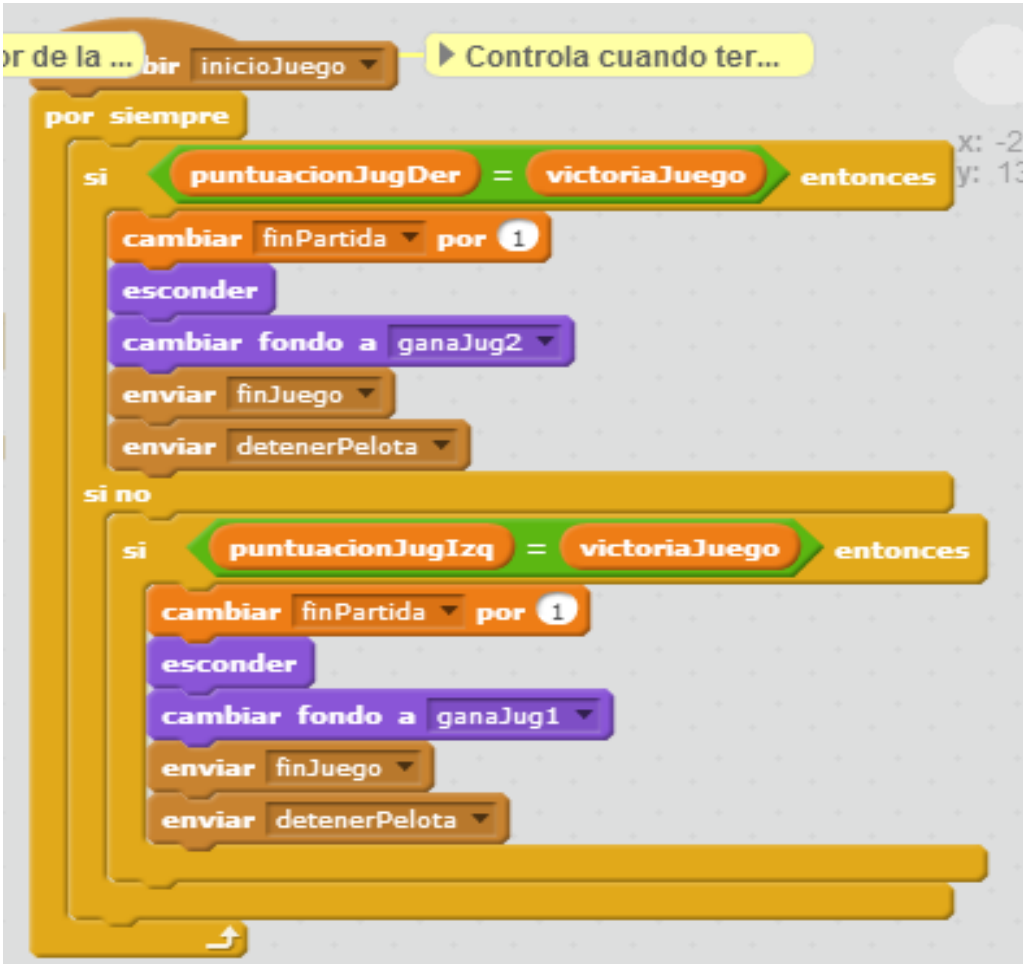

Figura 5.6: Fragmento de código que maneja la secuencia de victoria de *Pong*.

inteligencia artificial se basara en la posición exacta de la pelota para moverse hacia ésta, y a la hora de devolver la pelota al jugador, que calculase la posición exacta de éste para devolverle la pelota en la dirección contraria y así obligarlo a moverse en esa dirección, o dirigir los tiros hacia las puntas del campo de juego para dificultar que el jugador devuelva los tiros.

Otra posible mejora sería la de añadir objetos con «bonus» cada cierta franja de tiempo, que ayudasen aleatoriamente a uno de los dos jugadores, o incluso a ambos jugadores. Se podrían diseñar tres tipos de «bonus»:

1. **Aumento de tamaño**: Al tocar el objeto, la paleta del usuario aumenta de tamaño, lo que permite mayor facilidad para golpear la pelota.

2. **Superficie pegajosa**: Cuando la paleta toca la pelota, en vez de devolverla queda pegada a la superficie, lo que permite al usuario reposicionar la misma y pulsando el espacio se devuelve al contrincante (desde la posición deseada).

3. **Lentitud**: Cuando un jugador toca con la paleta este objeto, el jugador contrario se mueve mucho más lento que antes (hasta que alguno de los dos jugadores consiga un punto). Este «bonus» podría dirigirse en dirección al jugador con menos puntos de la partida, para permitirle igualarse al contrario.

Por último cabe mencionar que el juego ha sido testeado y probado por dos jugadores, los cuales son mencionados en el apartado de créditos del videojuego. En el proceso se detectaron varios *bugs*, aunque el peor de todos fue el que impedía a la pelota rebotar en una paleta, debido a que ésta, una vez había colisionado, se introducía en las texturas de la paleta, quedando atrapada dentro indefinidamente. Dicho *bug* se resolvió incluyendo un número de error en la colisión, el cual se encarga de mover una pequeña distancia la pelota en la dirección contraria a la colisión, para evitar que acabe dentro de las texturas.

CAPÍTULO 6

Diseño e implementación de *Space Invaders*

En el siguiente capítulo se van a describir cada uno de los objetos que componen el videojuego *Space Invaders* dentro del entorno Scratch. También se incluye un apartado sobre posibles mejoras que se podrían implementar (mediante la inclusión de un jefe en uno de los niveles).

6.1 Organización del menú

El menú del videojuego *Space Invaders* es manejado por diez objetos (se puede observar en la Figura 6.1), los cuales nos permiten dirigirnos a las diferentes pantallas que disponemos dentro del mismo así como observar una pequeña animación con la letra «S» que inicia el nombre del videojuego. A continuación se van a describir cada uno de ellos:

Figura 6.1: Menú del videojuego *Space Invaders*, en el cuál se pueden observar todos los objetos que lo manejan.

1. **Sonido:** Este objeto se encuentra ubicado en la pantalla de opciones y es el encargado de manejar si el sonido se encuentra a «On» o a «Off» en el videojuego, por medio de la variable sonido, la cual se encuentra a cero por defecto (si hay sonido), y a uno si no hay sonido disponible. Envía las señales «sonidoOn» y «sonidoOff» para que otros objetos ejecuten los fragmentos que están a la espera de las mismas las cuales regulan el sonido dentro del juego.

2. **Puntuación:** Objeto que lleva al usuario a la pantalla de puntuaciones (mediante la señal «Puntuación») y oculta los demás objetos del menú. En dicha pantalla se puede observar la puntuación de la última partida, así como la máxima puntuación del jugador (a lo largo de todas las partidas jugadas).

3. **Jugar:** Es el objeto encargado de manejar toda la lógica del inicio de la partida. Al pulsar en él, se envía la señal de «inicioJuego», la cual inicia todos los objetos de la partida y oculta los objetos del menú a la vez que cambia al escenario del nivel de juego y la señal «inicioPartida» la cual inicia el contador de los bonus que aparecen a lo largo de la partida.

4. **Instrucciones:** Al pulsar en él, se lanza la señal «Instrucciones», la cual oculta los demás objetos del menú y cambia al Escenario de las instrucciones, en la cual se explica el funcionamiento del juego (Figura 7.2).

Figura 6.2: Instrucciones que explican el funcionamiento del videojuego *Space Invaders*.

5. **Volver:** Este objeto se encuentra en las pantallas de Instrucciones, Créditos y Puntuaciones, y se encarga de permitir la vuelta al menú desde dichas

pantallas, envía la señal «volverMenu» la cual hace que se ejecuten los fragmentos en los demás objetos que están a la espera de la misma (vuelve a mostrar los objetos del menú).

6. **Créditos**: Es el objeto encargado de mostrar al usuario los participantes en el proyecto, así como su creador y tutor. Envía la señal «Créditos», la cual se encarga de iniciar la animación de los textos de créditos, ocultar los demás objetos del menú y de llevar al usuario al Escenario correspondiente.

7. **Textos créditos**: Son tres objetos, que una vez reciben la señal «Créditos», van apareciendo por la parte superior del escenario en orden, y terminan colocados uno debajo del otro, mostrando una descripción de los participantes en el proyecto.

8. **Fantasma:** Objeto que simula un pequeño fantasma arrastrando la letra S para colocarla en su lugar correspondiente en el título del videojuego. El objeto simplemente sigue una secuencia programada que simula una animación al iniciar el videojuego.

9. **Animación letra S:** Objeto que se encarga de simular la letra «S» del título del videojuego. Simula la animación del inicio del menú junto al objeto «Fantasma», y al igual que éste simplemente sigue una secuencia programada.

6.2 Objetos del videojuego

El videojuego diseñado dispone de un total de treinta y dos objetos, de los cuales diez se encuentran funcionando en la parte de menú (y han sido descritos en el primer apartado) y los otros veintiún objetos se encargan del funcionamiento del juego, a continuación se va a describir cada uno de ellos.

1. **Jugador:** Principal objeto del videojuego y encargado de manejar la nave que utiliza el usuario para acabar con las naves enemigas. Sus principales características son las de moverse de izquierda a derecha pulsando las teclas correspondientes y la de disparar para acabar con las naves alienígenas, mediante el pulsado de la barra espaciadora, aunque la función más interesante del objeto es la que se encarga de producir el efecto de profundidad al moverse ésta, ya que gracias al intercambio de cuatro disfraces (Figura 6.3) se consigue dar sensación de que la nave se está moviendo en el espacio infinito y con ello otorgamos más realismo al videojuego. Para ello se utiliza el fragmento de la Figura 6.4 el cual se encarga de manejar la secuencia de disfraces que producen el efecto comentado, y como se puede observar, si pulsamos hacia la izquierda se irán sucediendo dichas imágenes hasta que soltemos la tecla y éste se encargará de volver a dejar la nave de manera centrada para evitar efectos no deseados (aunque si pulsamos la tecla

del sentido contrario se encargará de intercalar las imágenes para producir dicho efecto en la dirección contraria).

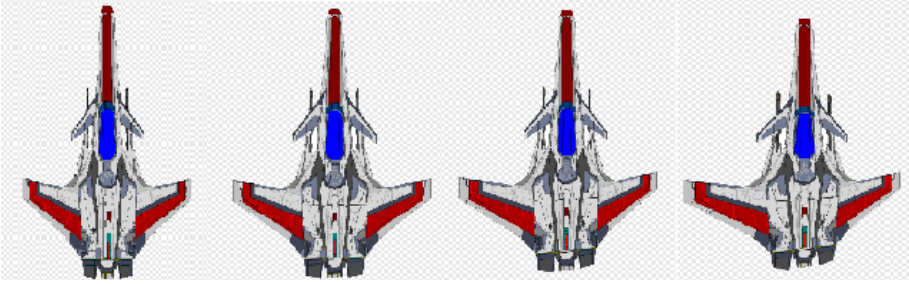

Figura 6.3: Efecto que produce la sensación de profundidad en los disfraces de la nave del videojuego *Space Invaders*.

Figura 6.4: Fragmento que maneja el cambio de disfraz en la nave de *Space Invaders*.

2. **Vidas Jugador:** Es el objeto encargado de manejar las vidas del jugador, como su propio nombre indica. Simplemente está formado por tres disfraces, cada uno indicando las vidas actuales que posee el usuario y manejando un fragmento de código que detecta cuando la salud de éste llega a cero, para cambiar al disfraz que simula una vida menos e indicarlo al jugador en el escenario (dentro de la partida, el objeto se encuentra en la esquina izquierda superior).

3. **Bonus:** Es el objeto encargado de manejar la aparición de los tres tipos de bonus que hay en el videojuego (armamento, vida y escudo). Funciona a modo de «Bonus Manager» y no aparece como tal en el escenario de juego, sino que su principal fragmento de código se encarga de decidir cuál de los tres bonus aparecerá y por dónde aparecerá (mediante el uso de una variable que escoge de manera aleatoria uno de los tres), una vez que el usuario decide colisionar con uno de éstos, automáticamente dejan de aparecer hasta que comienza una nueva partida (para evitar dar demasiadas ventajas al usuario).

4. **Armamento:** Objeto que maneja el bonus de tipo «armamento», el cual se activa al colisionar la nave con el objeto bonus que aparece en la partida

de juego. Se trata de una ampliación en la potencia de fuego del jugador, añadiendo un cañón a cada lado de la nave, los cuáles se activan al pulsar la barra espaciadora y añaden dos disparos más mediante la inicialización a uno de la variable «armamentoActivo», que maneja los misiles correspondientes (tipo de disparo).

5. **Escudo:** Objeto del tipo bonus que maneja el escudo que se añade a la salud del jugador una vez que ambos colisionan dentro de la partida de juego. Aparece encima de la salud como barras de color azul y protege al usuario de tres disparos de nave alienígena (evitando que éste pierda salud). Su funcionamiento es muy similar al del objeto que maneja la salud del jugador, con la pequeña diferencia que al llegar a cero éste no muere, sino que a partir de entonces se vuelve a perder salud.

6. **Salud Nave:** Objeto encargado de llevar la cuenta de la salud que dispone el usuario antes de que éste pierda una vida en la partida. Es un sencillo fragmento de código que simplemente monitoriza la variable «saludNave», y cada vez que ésta se reduce cambia su disfraz al que simula un punto menos de salud, para que el usuario tenga disponible en la pantalla de juego dicha información (ya que los objetos encargados de que ésta se reduzca son los que se encargan de los disparos de las naves alienígenas).

7. **Disparos:** Es un conjunto de siete objetos, los cuales realizan la misma función, aunque reaccionan a distintas señales, ya que cada uno de éstos objetos simula el disparo de uno de los tipos de nave existentes en el videojuego. Su funcionamiento se basa en reaccionar a una señal, que en el caso del disparo del jugador se produce al pulsar la barra espaciadora, y en el caso de los alienígenas cuando una variable alcanza un valor determinado de forma aleatoria y generan un clon en la posición del objeto que ha generado la señal que los activa. Su única función es la de colisionar con las nave del jugador (si se trata de disparos alien) o colisionar con las naves alienígenas (si se trata del disparo del jugador) y destruir ambos en la resolución[1] de ésta.

8. **Naves alien:** Es el conjunto de los cinco tipos de naves alienígenas que hay disponibles actualmente en el videojuego. Cada uno de los tipos de nave es un *«ShipManager»* que se encarga de crear una serie de clones que al final terminarán formando el bloque de las naves que el usuario debe destruir para completar el nivel. Como fragmento más interesante se podría comentar el el que hay disponible en la Figura 6.5, que se encarga de monitorizar la señal «bajarNivel» enviada por el *«GameManager»* y que provoca el descenso del conjunto de naves alienígenas.

9. **Sonido naves:** Es un objeto que maneja el sonido que producen las naves alienígenas al moverse, ya que así evitamos tener que incluir un bloque de

[1] Ver objeto «Disparos» en la sección del videojuego *Asteroids*.

Figura 6.5: Fragmento que maneja el envío de la señal «bajarNivel» que provoca que el bloque de naves alienígenas disminuya su posición hacia el final del escenario de *Space Invaders*.

sonido en cada una de las naves (ya que hay más de treinta clones en la partida) y así conseguimos centralizarlo. Simplemente se trata de un fragmento de código que espera a que las naves alienígenas comiencen a moverse y así disparar las señales que iniciar el sonido de movimiento.

10. **Nave destruida:** Es el objeto que se encarga de simular la destrucción de la nave del jugador y la nave bonus. Al recibir la señal de una colisión, éste inicia un bucle que va pasando por cada uno de los disfraces disponibles en el objeto, lo que para el usuario simula el efecto de una explosión, además se oculta la nave destruida para dar más credibilidad.

11. **Nave bonus:** Es el objeto que maneja la nave alienígena que aparece en la parte superior del escenario de juego, con la peculiaridad que en lugar de reducir un nivel cada vez que se manda la señal «bajarNivel» ésta se mueve de izquierda a derecha con el efecto «agujero de gusano» que se comenta en el apartado de descripción del objeto «Nave» del videojuego «Asteroids».

12. ***GameOver*:** Este objeto aparece cuando el jugador pierde las tres vidas (perdiendo antes los nueve puntos de salud). Simplemente se encarga de enviar la señal «menuGameOver» la cual recarga y muestra todos los objetos del

menú así como el fondo (al recibir esta señal, automáticamente se inicializan las animaciones de los objetos del menú).

13. **Línea de fin:** Es un objeto muy simple, que únicamente simula una línea invisible en la parte inferior del escenario de juego. Cuando las naves alienígenas colisionan con éste, se genera la señal «gameOver» y el juego termina.

14. ***GameManager*:** Es el objeto predefinido por defecto por Scratch denominado «Escenario» y que se ha decidido utilizar de *GameManager* del videojuego. Su principal función es la de manejar el flujo de escenario y sus fondos dentro del videojuego, aunque en ésta ocasión hemos aprovechado para utilizarlo de inicializador de la lista de bonus y manejador de la señal «bajarNivel», que indica a las naves alienígenas que deben descender una posición en la pantalla de juego. Como fragmento de código más interesante se encuentra el que inicializa la lista de bonus y que posteriormente permite seleccionar uno de estos por el objeto «Bonus» para que aparezca en el escenario de juego (Figura 6.6).

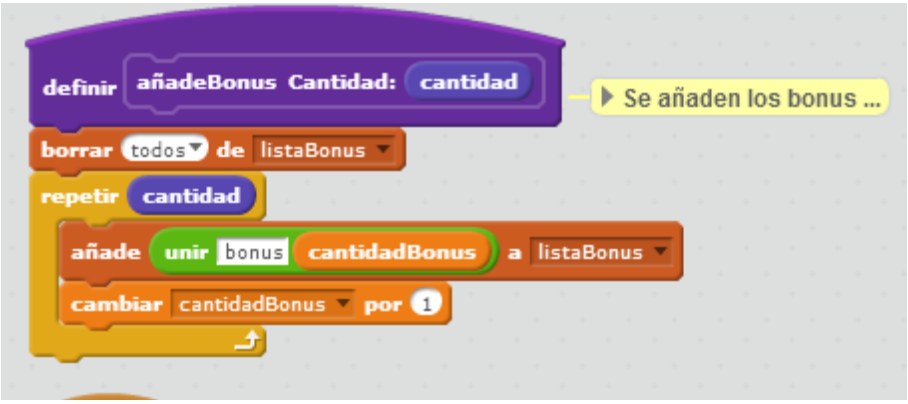

Figura 6.6: Fragmento que maneja la inicialización de la lista de bonus en *Space Invaders* (para ser usada dentro de la partida).

6.3 Posibles ampliaciones

Una característica que se pretendía incluir en el videojuego era la de un segundo nivel en la que se encontrase un jefe de partida, en lugar de una cantidad de naves, aunque por tiempo de desarrollo se decidió dejarla como posible ampliación. La idea es la de incluir un segundo nivel en la que el enemigo a batir sea un jefe, es decir, una nave con un tamaño mayor que el de las naves del nivel uno y con unos patrones de movimiento y ataque aleatorios, que provocasen al usuario una gran dificultad para derrotarlo. Uno de los patrones de movimiento que podría tener el jefe sería moverse de izquierda a derecha, y mediante una variable aleatoria, determinar que el jefe debe moverse hacia el centro de la pantalla,

girar y volver a la parte superior de la pantalla. Con esto podríamos conseguir despistar al usuario y que errase en los disparos; además, una vez comenzase a girar podríamos incluir un patrón de ataque en el que empezase a disparar en todas las direcciones para alcanzar al usuario.

Como patrones de ataque la nave podría tener el patrón normal, en el cual disparase un misil por cada cañón que posea y mediante una variable determinar en ciertos instantes de tiempo se generen disparos en forma de espiral, o que los disparos se dirijan hacia el jugador para intentar acabar con éste.

Cabe decir que el juego ha sido testeado por dos usuarios, y que en el proceso se detectaros varios *bugs*. En concreto, uno de ellos provocaba que al recibir el bonus que añade dos cañones a la nave, si ésta era destruida, el objeto que maneja los cañones adicionales no se destruía con ella, y al reaparecer quedaba estático en la posición que había sido destruida y podíamos disparar desde esa posición. La solución pasó por eliminar junto a la nave dicho objeto, y reiniciar la variable de posición, ya que aunque eliminásemos dicho objeto, si volvíamos a obtener dicho bonus, la posición anterior quedaba guardada y el funcionamiento no era el deseado.

CAPÍTULO 7

Diseño e implementación de *Asteroids*

En el siguiente capítulo se va a describir el proceso seguido para llevar a cabo el desarrollo de los diferentes objetos que componen el videojuego *Asteroids* en Scratch. También se describirá el proceso seguido para resolver una colisión entre objetos así como posibles mejoras o ampliaciones.

7.1 Organización del menú

El menú del videojuego *Asteroids* es manejado por siete objetos (se puede observar en la Figura 7.1), los cuales nos permiten dirigirnos a las diferentes pantallas que disponemos dentro del mismo y controlar el sonido de los demás objetos. A continuación se va a describir cada uno de ellos.

Figura 7.1: Menú del videojuego *Asteroids* creado en Scratch.

1. **Jugar**: Es el objeto encargado de manejar toda la lógica del inicio de la partida. Al pulsar en él, se envía la señal de «inicioJuego», la cual inicia todos los objetos de la partida y oculta los objetos del menú a la vez que cambia al escenario del nivel de juego.

2. **Puntuaciones**: Objeto que lleva al usuario a la pantalla de puntuaciones (mediante la señal «puntuaciones») y oculta los demás objetos del menú. En dicha pantalla se pueden observar los marcadores de los últimos cinco jugadores (ordenados por último jugador).

3. **Instrucciones**: Al pulsar en él, se lanza la señal «Instrucciones», la cual oculta los demás objetos del menú y cambia al Escenario de las instrucciones, en el cual se explica el funcionamiento del juego (cómo usar la nave y disparar) (Figura 7.2).

4. **Créditos**: Es el objeto encargado de mostrar al usuario los participantes en el proyecto, así como su creador y tutor. Envía la señal «Créditos», la cual se encarga de iniciar la animación de los textos de créditos, ocultar los demás objetos del menú y de llevar al usuario al Escenario correspondiente.

5. **Textos créditos**: Son tres objetos, que una vez reciben la señal «Créditos», van apareciendo por la parte superior del escenario en orden, y terminan colocados uno debajo del otro, mostrando una descripción de los participantes en el proyecto.

6. **Sonido**: Este objeto se encuentra ubicado en la pantalla de inicio y es el encargado de manejar si el sonido se encuentra a «On» o a «Off» en el videojuego, por medio de la variable sonido, la cual se encuentra a cero por defecto (si hay sonido), y a uno si no hay sonido disponible. Envía las señales «sonidoOn» y «sonidoOff» para que otros objetos ejecuten los fragmentos que están a la espera de las mismas (y silencien o reanuden el sonido).

7. **Volver**: Este objeto se encuentra en las pantallas de Puntuación, Instrucciones y Créditos. Se encarga de permitir la vuelva al menú desde dichas pantallas, envía la señal «volverMen» la cual hace que se ejecuten los fragmentos en los demás objetos que están a la espera de la misma (vuelve a mostrar los objetos del menú).

7.2 Objetos del videojuego

El videojuego diseñado dispone de un total de veintiocho objetos, de los cuales siete se encuentran funcionando en la parte de menú (y han sido descritos en el primer apartado) y los otros veintiún objetos se encargan del funcionamiento del juego, a continuación se va a describir cada uno de ellos.

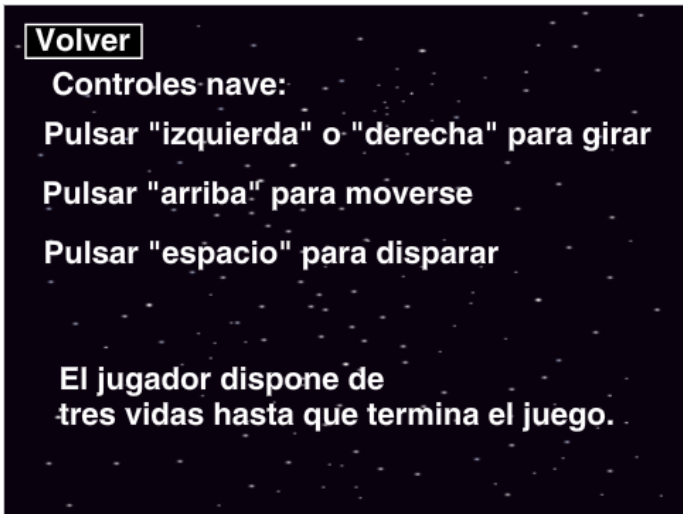

Figura 7.2: Instrucciones del videojuego *Asteroids*, las cuales explican cómo jugarlo.

1. **Nave:** Este objeto es el principal del juego, ya que es el que maneja el usuario y es en él donde recae la lógica más compleja. Par empezar, el objeto dispone de una tecla de movimiento «fecha arriba», la cual hace que el objeto pueda moverse por el escenario, y mediante las flechas «izquierda» y «derecha», podemos hacer que la nave gire. El fragmento de código que se encarga de manejar el movimiento se encuentra en la Figura 7.3; éste está dotado de cierta lógica, por la cual la nave se mueve de golpe cierta distancia, y poco a poco se va deteniendo debido a la simulación de las fuerzas de rozamiento. Además entre *frame*[1] y *frame*, el fragmento de código se encarga de intercalar los dos disfraces que dispone el objeto, ya que con esto se consigue simular el efecto del fuego de los motores de la nave (Figura 7.4).

   Otra funcionalidad peculiar de la nave es que cuando ésta toca algún meteorito, o la nave alienígena la impacta, queda destruida, lo que produce que aparezca en el medio del escenario y perdamos una vida. Para evitar que un meteorito nos vuelva a pasar por encima mientras reaparecemos, la nave se pone en estado de invulnerabilidad, lo que evita que sea destruida al tocar meteoritos, hasta que el usuario dispare o se mueva.

   Por último, tanto la nave del jugador, como la nave alienígena y los meteoritos, disponen de un sistema de detección de los bordes de la pantalla, que simula el efecto de «agujero de gusano», por el cual éstos, al tocar uno de los bordes, aparecen por el borde contrario de la pantalla, así se consigue el efecto del espacio infinito (Figura 7.5).

2. **Disparos:** Aquí se reúnen los dos objetos que manejan los disparos de la nave del jugador y del alienígena. Por un lado, el objeto disparo del jugador, reacciona a la pulsación de la barra espaciadora, la cual genera un clon de

[1]Cada una de las imágenes instantáneas en las que se divide una película de cine que dan sensación de movimiento al ser proyectadas secuencialmente.

Figura 7.3: Fragmento de código que maneja el efecto de movimiento espacial de la nave del usuario en el videojuego Asteroids.

éste en la posición actual de la nave del jugador, el cual se dirige en línea recta hasta que toca los bordes, o hasta que impacta contra un meteorito o nave alienígena.

Las colisiones del videojuego *Asteroids* tienen tres estados principales:

- El primer estado es cuando se detecta que dos objetos han colisionado (aunque la lógica de colisiones la proporciona el propio lenguaje Scratch), al que se denomina «Detección»
- El segundo estado es cuando se decide que ocurre en la colisión (esta parte es la que ha sido implementada en el videojuego), y se denomina «Lógica de colisión».
- El tercer y último estado es el que resuelve lo que se ha decidido en la lógica de colisiones, y se denomina «Resolución».

Una vez explicados los estados de una colisión, se puede decir que al colisionar un disparo y un meteorito, en la lógica de colisión se decide destruir el disparo, y el meteorito se divide en tres clones más pequeños, los cuales salen en direcciones aleatorias por el escenario, y en la parte de resolución se ejecutan los fragmentos de código correspondientes. En el caso de la colisión con la nave alienígena se ejecuta el objeto «Sprite Destrucción», el cual

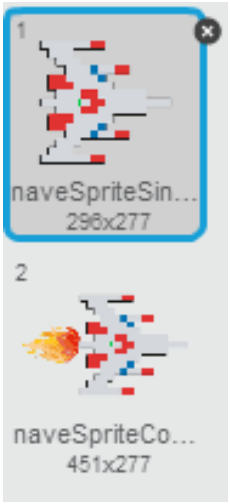

Figura 7.4: Disfraces de la nave del videojuego *Asteroids*, los cuales realizan el efecto del fuego de los motores al moverse.

hace pasar una serie de disfraces que simulan el efecto de una explosión, y hacen desaparecen la nave alienígena.

Por otro lado el disparo alienígena no tiene colisión con los meteoritos, ya que estos tipos de disparo solo dañan la nave del jugador. Al producirse la colisión con ésta, la lógica manda señales al objeto de «Sprite Destrucción» y elimina el disparo alien, además reduce en uno el número de las vidas por medio del objeto «Vidas».

3. **Meteoritos:** Se dispone de cuatro objetos que manejan la lógica de los meteoritos, aunque se han juntado, ya que los cuatro funcionan de la misma manera pero con particularidades en el movimiento, dirección etc. Los meteoritos se generan en la escena al comienzo de la partida, y mantienen el movimiento en una misma dirección, hasta que colisionan con el jugador o con un disparo, por lo cual si son los meteoritos grandes, se dividen en tres más pequeños, y si la colisión se produce en uno de los pequeños, éste queda completamente destruido junto al disparo.

   Gracias a Scratch 2.0, los meteoritos reales solo son cuatro, siendo el resto clones que se generan al colisionar, y que contienen el mismo comportamiento que sus padres. Al destruir un meteorito la puntuación de los marcadores aumenta, quedando registrada al finalizar la partida.

   Estos también disponen de un fragmento que maneja el cambio de disfraz, lo que simula el movimiento rotatorio de un meteorito en el espacio, dicho fragmento de código se detiene al colisionar con un disparo o con la nave del jugador.

4. **Marcadores:** Los marcadores se encargan de mostrar al usuario la puntuación actual que dispone (cada meteorito otorga un punto, y la nave alie-

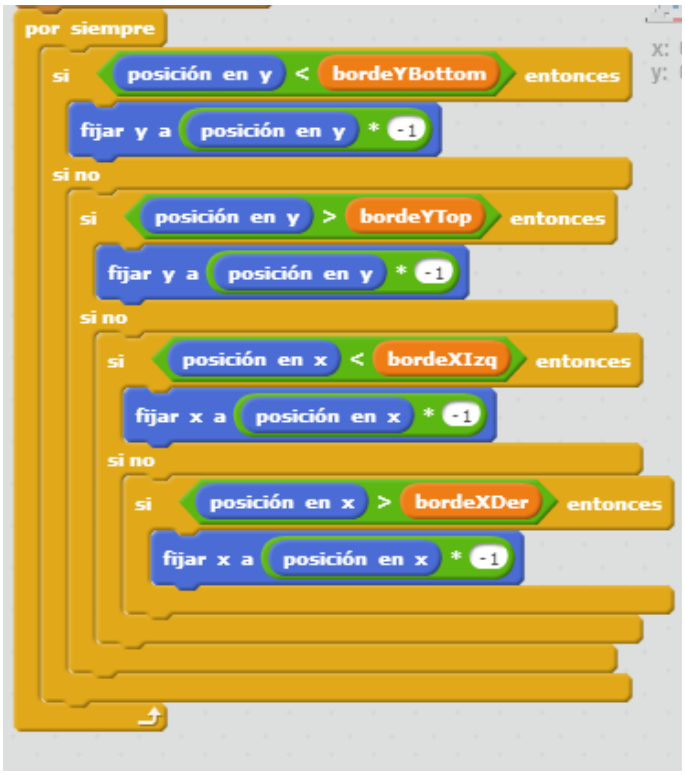

Figura 7.5: Fragmento de código que maneja la detección de los bordes en el videojuego Asteroids (con ello conseguimos el efecto «agujero de gusano»).

nígena otorga diez). Son tres objetos (el de unidades, decenas y centenas), los cuales se encargan de reaccionar a la señal que envía un meteorito al ser destruido (patrón observador), con la consiguiente lógica de cambiar el *sprite* de las unidades al siguiente.

5. **Vidas:** Los tres objetos «Vida» se encargan de manejar la muerte del usuario. Cuando este colisiona con un disparo de la nave alienígena o con uno de los meteoritos, su atributo «vida» se reduce en uno y los objetos detectan que ésta se ha reducido en uno, eliminando una de ellas y mostrando al usuario que dispone de una vida menos antes de perder la partida.

   Cuando el número de vidas llega a cero, la partida se acaba y se envía la señal «GameOver», la cual ejecuta el objeto «GameOver» descrito al final del apartado.

6. **NaveAlien:** Este objeto se encarga de controlar la nave alienígena que aparece en la parte superior del escenario. Tiene un fragmento de código igual al de los meteoritos y la nave del jugador para detectar los bordes de la pantalla y crear el efecto «agujero de gusano» por el cual aparece en el borde contrario. Como siempre va de derecha a izquierda, desaparece por el borde izquierdo y aparece por el borde derecho.

Además dispone de un fragmento que maneja los disparos de la nave de manera aleatoria cada cierto tiempo, enviando una señal al objeto «disparoAlien», el cual genera un clon del mismo y envía el disparo en línea recta hasta que toca el borde inferior. Si este colisiona contra la nave del jugador, ésta queda destruirla con lo que ello conlleva.

7. **Sprite Destrucción:** Es el objeto que se encarga de simular la destrucción de las dos naves del videojuego. Al recibir la señal de una colisión, éste inicia un bucle que va pasando por cada uno de los disfraces disponibles en el objeto, lo que para el usuario simula el efecto de una explosión, además se oculta la nave destruida para dar más credibilidad.

8. **GameOver**: Este objeto aparece cuando el jugador pierde las tres vidas. Se encarga de mostrar una barra para escribir el nombre del usuario, y que éste quede registrado en el apartado de «Puntuaciones», que hay disponible en el menú del juego.

7.3 Posibles ampliaciones

El apartado de «movimiento espacial» del videojuego sería un perfecto candidato para ampliar el videojuego, ya que aunque se ha logrado simular con bastante acierto el efecto del espacio al mover la nave, se podrían incluir fuerzas de aceleración para que el movimiento de ésta fuera más realista. Podría funcionar de la siguiente manera, una vez el usuario pulsase la tecla de avanzar, la nave comenzaría a moverse lentamente, y la fuerza de aceleración iría aumentando la velocidad de ésta hasta cierto valor. Si el usuario soltase la tecla, la nave tardaría un poco en detenerse debido a la fuerza espacial que la empujaría contra los bordes del escenario, de esta manera el juego podría llegar a ser más desafiante si cabe.

En cuanto a la nave alienígena que aparece en la pantalla, actualmente solo dispone de movimiento lineal, y una vez que llega al borde izquierdo de la pantalla, ésta desaparece y vuelve a entrar en escena en el borde derecho. Se podría incluir un movimiento sinusoidal para la nave, lo que daría más realismo al efecto de movimiento en el espacio, aplicando para ello la ecuación correspondiente al fragmento que maneja el movimiento de ésta. Otra ampliación que podríamos añadir sería la de otorgar la habilidad de apuntar a la nave alienígena, provocando que los disparos de ésta fueran dirigidos hacia el jugador en lugar de en línea recta hacia el fondo del escenario.

Por último, cabe mencionar que el juego ha sido testeado y probado por dos jugadores, los cuales son mencionados en el apartado de créditos del videojuego. En el proceso se detectaron varios *bugs*, aunque el más curioso era uno que provocaba que si el jugador llegaba a uno de los bordes del escenario con muy poca velocidad, no se producía el efecto «agujero de gusano», el cual hace que la

nave aparezca por el borde contrario al que acaba de entrar (en la escena), y la nave quedaba eternamente rebotando contra éste. Este *bug* surgió debido a que las primeras pruebas con el videojuego se realizaban con un modelo de nave a mayor escala, por lo que la variable de detección de bordes estaba fijada acorde al tamaño de ésta. Al cambiar de modelo, la nave se hizo más pequeña, y si no llevaba suficiente velocidad, la variable no detectaba que había llegado al borde, y quedaba atrapada en un bucle hasta que se reiniciaba el juego.

CAPÍTULO 8

Diseño e implementación de *Pac Man*

En el siguiente capítulo se explica el desarrollo seguido para la creación de los distintos objetos que forman el videojuego *Pac Man* mediante el uso de Scratch. También se explicará la forma en la que se utiliza la máquina de estados que permite el funcionamiento del objeto que maneja a *Pac Man*, así como a los fantasmas, además de posibles mejoras para el videojuego.

8.1 Organización del menú

El menú del videojuego *Pac Man* es manejado por once objetos (se puede observar en la Figura 8.1), los cuales nos permiten dirigirnos a las diferentes pantallas que disponemos dentro del mismo y generar una animaciones que animan el escenario del menú. A continuación se van a describir cada uno de ellos:

Figura 8.1: Menú del videojuego *Pac Man* en el que se pueden observar todos los objetos que lo componen, así como las animaciones.

1. **Jugar:** Es el objeto encargado de manejar toda la lógica del inicio de la partida. Al pulsar la barra espaciadora se envía la señal de «inicioJuego», la cual inicia todos los objetos de la partida y oculta los objetos del menú a la vez que cambia al escenario del nivel de juego (y detiene las animaciones).

2. **Volver:** Este objeto se encuentra en las pantallas de Puntuación, Instrucciones y Créditos. Se encarga de permitir la vuelva al menú desde dichas pantallas, envía la señal «volverMenu» la cual hace que se ejecuten los fragmentos en los demás objetos que están a la espera de la misma (vuelve a mostrar los objetos del menú).

3. **Instrucciones:** Al pulsar en él, se lanza la señal «Instrucciones», la cual oculta los demás objetos del menú y cambia al Escenario de las instrucciones, en la cual se explica el funcionamiento del juego (Figura 8.2).

Figura 8.2: Instrucciones en la que se explica como terminar una partida de *Pac Man* y las puntuaciones de cada objeto.

4. **Puntuaciones:** Objeto que lleva al usuario a la pantalla de puntuaciones (mediante la señal «Puntuación») y oculta los demás objetos del menú. En dicha pantalla se puede observar la puntuación de la última partida, así como la máxima puntuación del jugador (a lo largo de todas las partidas jugadas).

5. **Créditos**: Es el objeto encargado de mostrar al usuario los participantes en el proyecto, así como su creador y tutor. Envía la señal «Créditos», la cual se encarga de iniciar la animación de los textos de créditos, ocultar los demás objetos del menú y de llevar al usuario al Escenario correspondiente.

6. **Textos créditos**: Son tres objetos, que una vez reciben la señal «Créditos», van apareciendo por la parte superior del escenario en orden, y terminan colocados uno debajo del otro, mostrando una descripción de los participantes en el proyecto.

7. **Fantasma:** Objeto que simula un fantasma del videojuego que está siendo perseguido por *Pac Man* y va apareciendo por los bordes de la pantalla junto a éste. Simplemente es un fragmento de código que sigue una secuencia programada en bucle hasta que el jugador pulsa la barra espaciadora y comienza la partida.

8. **PacmanMenu:** Objeto que simula a *Pac Man* persiguiendo un fantasma durante la pantalla del menú del videojuego. Va apareciendo por los bordes de ésta junto al fantasma y simplemente está formado por un fragmento de código que sigue una secuencia programada en bucle hasta que el jugador pulsa la barra espaciadora y la partida comienza.

9. **Efecto Velocidad:** Son dos objetos que se han creado para simular un efecto futurista en el menú, dando una animación que simula la velocidad. Es un fragmento de código el cual está constituido por un bucle que desplaza los disfraces de estos dos objetos por el escenario del menú (hasta que se comienza una nueva partida).

8.2 Objetos del videojuego

El videojuego diseñado dispone de un total de veintidos objetos, de los diez se encuentran funcionando en la parte de menú y las animaciones (y han sido descritos en el primer apartado) y los otros doce objetos se encargan del funcionamiento del juego y los distintos niveles del mismo, a continuación se va a describir cada uno de ellos:

1. **Pac Man:** Es el objeto principal del videojuego, ya que es dirigido por el usuario y es el encargado de recolectar los puntos amarillos que hay por el mapa para completar la partida. Es manejado por una máquina de estados que le permite variar entre las diferentes direcciones, así como detener la animación de los disfraces una vez se ha colisionado con uno de los muros, o pasar al estado «Muerto» al colisionar con uno de los fantasmas. La máquina de estados que lo controla se encarga de detectar la pulsación de una de las cuatro teclas de dirección para cambiar los estados, o si se colisiona con uno de los fantasmas, pasar al estado de «Muerto» y finalizar la partida (además de ejecutar la animación de muerte de *Pac Man*).

   Una peculiaridad que se ha incluido es la de incluir los puntos amarillos del mapa como parte del escenario (así nos evitamos toda la lógica de creación de clones, que podría llegar a ser muy pesada debido a la enorme cantidad de puntos) y para simular que *Pac Man* se los come al pasar por encima de ellos, se ha utilizado el bloque del pincel, así cada vez que se detecta el color amarillo, se baja el pincel y la zona se pinta de negro, además de restar uno al contador de puntos (al volver a iniciar el nivel se borra el pincel, así vuelven a aparecer los puntos).

El funcionamiento de los fragmentos de bloque de *Pac Man* es muy sencillo, ya que simplemente puede realizar acciones como ir hacia arriba, abajo, derecha o izquierda, por lo que la verdadera dificultad de diseño del objeto reside en su máquina de estados y en cómo los colisionadores interactúan con el entorno y los objetos.

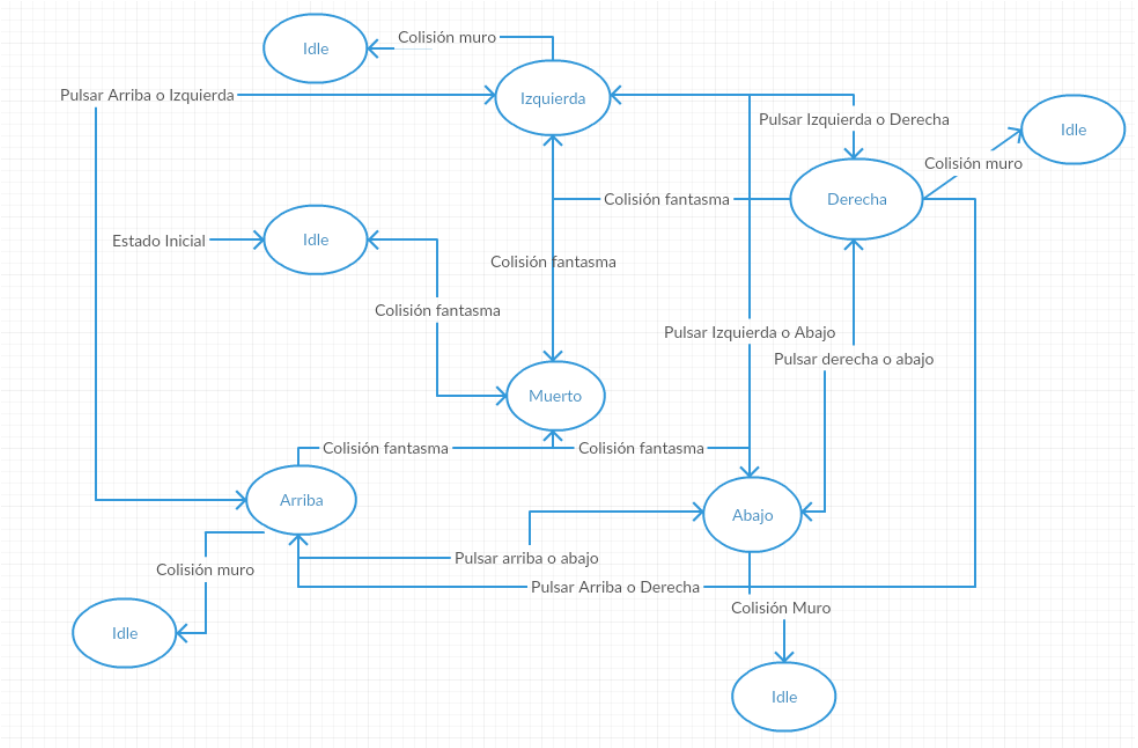

Figura 8.3: Diagrama en el que se puede observar la máquina de estados que compone el objeto *Pac Man*, por el cual éste realiza las diferentes acciones dentro del videojuego.

Para consultar más información acerca de las máquinas de estados, se puede ver [6], artículo consultado para la realización del presente trabajo y que explica muy bien cómo definir y utilizar una máquina de estados finita.

2. **Fantasmas:** Este objeto está formado por los cuatro fantasmas del videojuego, cada uno de los cuales tiene un comportamiento ligeramente diferente al resto, aunque todos son manejados por la misma máquina de estados, la cual les proporciona autonomía para desplazarse por el escenario de juego.

    - El **fantasma azul** tiene el comportamiento más sencillo y tonto de los cuatro, ya que simplemente se dedica a «rebotar» en los muros, y es que cada vez que su colisionador detecta un muro, éste cambiará de dirección aleatoriamente.
    - El **fantasma rojo** tiene un comportamiento más agresivo, ya que siempre se dirigirá en la dirección de *Pac Man*, así obligaremos al jugador a ir cambiando constantemente de posición si no quiere acabar presa de éste enemigo.

- El **fantasma verde** tiene un comportamiento similar al fantasma azul, en el sentido de que cambia aleatoriamente de dirección al colisionar con un muro, aunque además puede cambiar en cualquier momento de dirección (sin necesidad de colisionar).
- Por último el **fantasma rosa** tiene un comportamiento fijo, ya que se ha decidido que éste se dirija a la parte superior del escenario y permanezca en ese cuadrante para evitar que el jugador quede sin oposición enemiga (ya que los demás fantasmas tienden a estar en zonas bajas del escenario).

El funcionamiento genérico de la máquina de estados que se muestra en la Figura 8.4 indica que los fantasmas comienzan en el estado «Idle» o parado al iniciar la partida, y aleatoriamente se dirigen hacia cualquiera de las cuatro direcciones disponibles. Al colisionar con un muro cambiarán de dirección y si *Pac Man* recoge una de las píldoras de la partida éstos entrarán en el estado «Vulnerable» donde pueden morir.

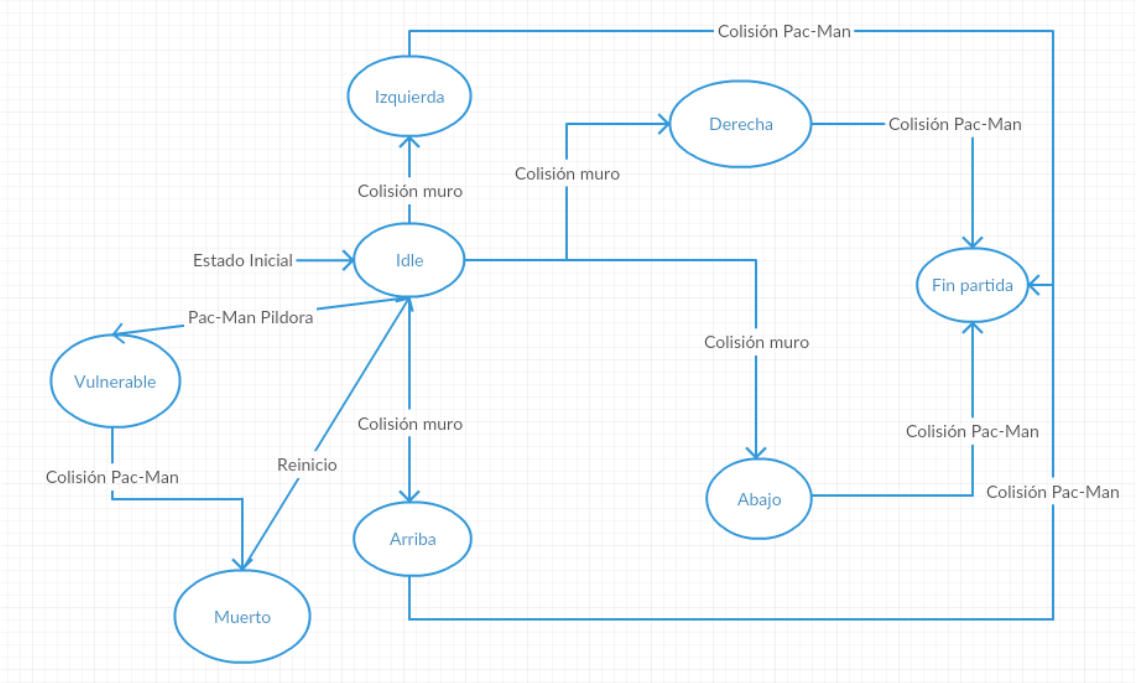

Figura 8.4: Diagrama en el que se puede observar la máquina de estados que compone los objetos de tipo fantasma, mediante la cual son capaces de actuar de manera autónoma (simulando una ligera inteligencia artificial).

3. **Píldoras:** Es el objeto que permite a *Pac Man* acabar con los fantasmas al colisionar con ellos, ya que en su estado natural, si se produce la colisión es *Pac Man* quien muere y la partida termina. Cuando *Pac Man* recolecta una de éstas píldoras (que se encuentran parpadeantes en ciertos sectores del escenario) los fantasmas alcanzan el estado «Vulnerable» en su máquina de

estados, en el cuál pueden ser derrotados. Una vez muertos se dirigen a la zona de inicio, y vuelven a revivir.

4. **Colisionadores:** Objeto formado por cinco colisionadores (uno de *Pac Man* y cuatro de los fantasmas) encargados de detectar cuándo uno de los objetos que rodean colisiona con uno de los muros del videojuego. En la Figura 8.5 se encuentra el fragmento que cada uno de éstos objetos posee, con ligeras variaciones debido al cambio de comportamiento de Pac Man y los fantasmas. Su función principal es la de cambiar los estados de la máquina de estados que maneja y así simular esta inteligencia artificial en los fantasmas y permitir transiciones en las animaciones de *Pac Man* de manera más sencilla.

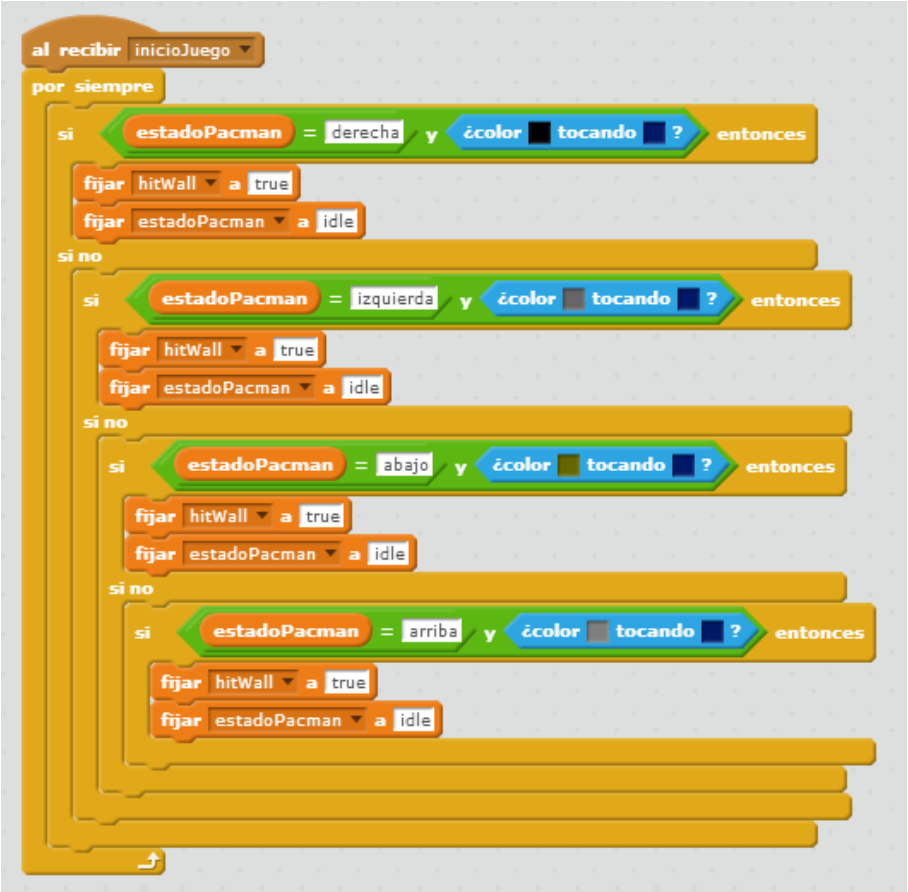

Figura 8.5: Fragmento de código que maneja el objeto colisionador de *Pac Man*, el cual detecta cuando éste colisiona con los muros.

5. **Frutas:** Es el objeto que maneja la lista de frutas que se inicializa previamente al iniciar la partida y aparecen en la posición inicial de *Pac Man* una vez éste comienza a moverse de manera aleatoria. Su función es la de otorgar puntos cada vez que *Pac Man* las recolecta (así le damos valor a que el ju-

gador aguante el nivel para que aparezcan dichos objetos y así obtenga una mayor puntuación general).

6. **Texto puntuación:** Objeto que aparece en el centro del mapa cada vez que *Pac Man* recoge una fruta en la partida o acaba con uno de los fantasmas mientras éstos se encuentren en el estado «Vulnerable». Simplemente realiza la función de aparecer y desvanecerse indicando al jugador que acaba de aumentar su puntuación en la partida.

7. **GameOver:** Este objeto aparece cuando *Pac Man* muere y simplemente se encarga de enviar la señal «menuGameOver» la cual recarga y muestra todos los objetos del menú así como el fondo (al recibir esta señal, automáticamente se inicializan las animaciones de los objetos del menú).

8.3 Posibles ampliaciones

La principal ampliación que se podría hacer es la de mejorar la inteligencia artificial del videojuego, ya que la actual se basa en cambiar de dirección al colisionar con un muro, cambiar aleatoriamente de dirección o dirigirse a la zona en la que se encuentra *Pac Man*. La idea sería crear un fantasma muy agresivo, que fuera directamente a por éste, es decir, que detectara la posición exacta en la que se encuentra y mediante algoritmos de cálculo del mejor camino posible (podría utilizarse el algoritmo de Dijkstra[1]), dirigirse hacia *Pac Man* directamente, sin dar vueltas por el mapa. Simplemente cada cierto intervalo de tiempo debería recalcular su ruta ya que nosotros como jugadores no paramos de movernos por el mapa.

También se podrían añadir diversos niveles en los que por ejemplo cambiase la velocidad con la que *Pac Man* y los fantasmas se desplazan por los pasillos y cuanto más se avanzase en dichos niveles, mayor dificultad encontrase el jugador para completarlos (así como unas mecánicas más agresivas en los fantasmas).

Por último, lo que a la parte de test se refiere, uno de los *bugs* más característicos que se han encontrado en el juego ha sido el que permitía a *Pac Man* atravesar las paredes una vez éste había colisionado con ellas. Se trataba de un error en la implementación de la máquina de estados, ya que ésta posicionaba a *Pac Man* en el estado *idle* repetidas veces, lo que producía que fuese avanzando en pequeñas cantidades dentro de la pared del escenario, hasta que la atravesaba. La solución consistía en evitar que una vez el colisionador detectase que éste estaba tocando el muro, evitar poder cambiar al estado de esa misma posición, y así no abandonar el estado *idle* hasta ir por un camino sin muros.

[1]Más información en `https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra`

CAPÍTULO 9

Diseño e implementación de *Donkey Kong*

En el siguiente capítulo se va a describir el proceso seguido para desarrollar el videojuego *Donkey Kong* con el uso de Scratch así como la inclusión del diagrama de estados finitos que se encarga de manejar la lógica de Mario, el objeto manejado por el jugador. También se incluye un apartado sobre posibles mejoras que se podrían implementar así como uno de los *bugs* que se encontraron durante el desarrollo.

9.1 Organización del menú

El menú del videojuego *Donkey Kong* es manejado por nueve objetos (se puede observar en la Figura 9.1), los cuales nos permiten dirigirnos a las diferentes pantallas que disponemos dentro del mismo así como iniciar el videojuego con una pequeña introducción, además se han creado efectos de animación para los distintos botones de éste. A continuación se van a describir cada uno de ellos:

1. **SecuenciaDonkey:** Es el objeto encargado de manejar la secuencia de introducción cuando iniciamos el videojuego. Se encarga de seguir una serie de acciones programadas las cuales simulan un *Donkey Kong* en movimiento que sube por los pisos de una estructura, para terminar saltando sobre el último piso y provocando el desmoronamiento de las losetas de cada uno de los pisos y generando lo que será el primer nivel de juego. En la Figura 9.2 se puede observar el final de la introducción.

2. **Reglas:** Al pulsar en él, se lanza la señal «Reglas», la cual oculta los demás objetos del menú y cambia al Escenario de las reglas del juego, en el cual se explica como manejar a Mario, el protagonista del videojuego (Figura 9.3).

Figura 9.1: Menú de inicio del videojuego *Donkey Kong*, gracias al cual podemos navegar hacia las distintas pantallas del mismo.

3. **Puntuación:** Objeto que lleva al usuario a la pantalla de puntuaciones (mediante la señal «Puntuación») y oculta los demás objetos del menú. En dicha pantalla se puede observar la puntuación de la última partida, así como la máxima puntuación del jugador (a lo largo de todas las partidas jugadas).

4. **Créditos:** Es el objeto encargado de mostrar al usuario los participantes en el proyecto, así como su creador y tutor. Envía la señal «Créditos», la cual se encarga de iniciar la animación de los textos de créditos, ocultar los demás objetos del menú y de llevar al usuario al Escenario correspondiente.

5. **Texto créditos:** Son tres objetos, que una vez reciben la señal «Créditos», van apareciendo por la parte superior del escenario en orden, y terminan colocados uno debajo del otro, mostrando una descripción de los participantes en el proyecto.

6. **Volver:** Este objeto se encuentra en las pantallas de Puntuación, Reglas y Créditos, y se encarga de permitir la vuelta al menú desde dichas pantallas. Envía la señal «volverMenu» la cual hace que se ejecuten los fragmentos en los demás objetos que están a la espera de la misma (vuelve a mostrar los objetos del menú).

7. **Jugar:** Este objeto se diferencia de los demás en que no hay que hacer click para enviar la señal que inicia la partida, en este caso se ha creado de manera que una vez accedemos al menú de juego, éste permanece parpadeando para llamar la atención del jugador, y al pulsar la tecla «1», la partida se inicia, enviando éste la señal «InicioJuego», ocultando todos los objetos del menú y cambiando al escenario correspondiente.

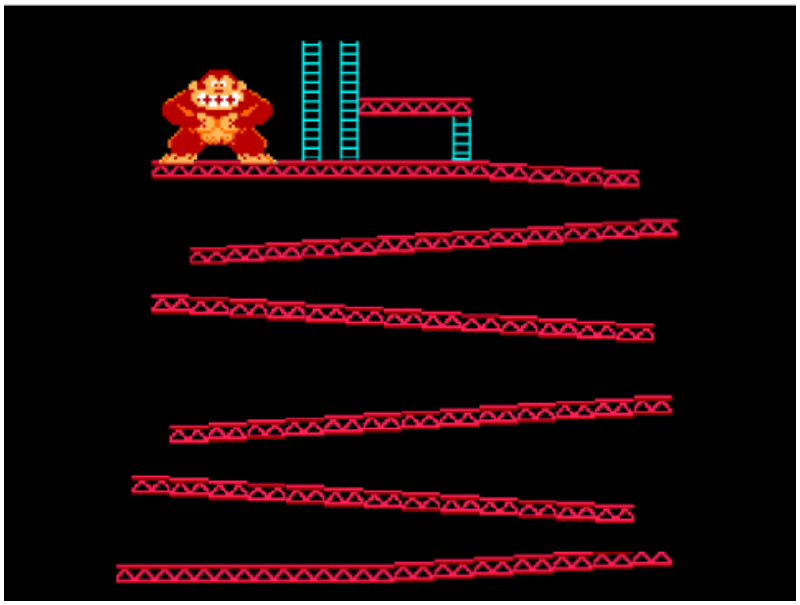

Figura 9.2: Introducción del videojuego *Donkey Kong* en la que se puede observar el final de la animación inicial del videojuego.

9.2 Objetos del videojuego

El videojuego diseñado dispone de un total de veinticuatro objetos, de los cuales nueve se encuentran funcionando en la parte de menú (y han sido descritos en el primer apartado) y los otros catorce objetos se encargan del funcionamiento del juego, a continuación se va a describir cada uno de ellos.

1. **Mario:** Es el objeto que maneja al protagonista del videojuego, el cual es utilizado por el usuario. Antes de explicar algunos de los fragmentos que forman dicho objeto, hay que explicar cómo funciona la máquina de estados que maneja las diferentes acciones que podemos realizar con dicho artefacto (Figura 9.4). El estado de comienzo se denomina «Parado», y en dicho estado Mario se encuentra quieto en la pantalla de juego, y no se ejecuta ninguna de sus acciones, desde dicho estado podemos acceder al estado «Caminando» pulsando la tecla derecha o izquierda del teclado, por lo cual Mario comenzará a desplazarse por el escenario y la animación de los disfraces del objeto comenzará a ejecutarse, para que de sensación de realismo al caminar. También podemos pulsar la barra espaciadora, por la cual saltaremos cierta distancia, a la vez que se ejecutarán los disfraces correspondientes; además, una vez sobre el suelo, si pulsamos derecha o izquierda avanzaremos en dicha dirección a la vez que realizamos un salto. Por último, si nos dirigimos hacia unas escaleras y pulsamos la tecla arriba del teclado, Mario comenzará a escalar por dicho objeto y pasará al estado «Escalando», hasta que volvamos a tocar el suelo, por lo cual volveremos al estado de

Figura 9.3: Pantalla del videojuego *Donkey Kong* en la que se explica cómo manejar al protagonista, Mario.

«Caminando». Si un barril o un elemental colisiona con Mario, el personaje pasará al estado de «Muerto», quedando todas las acciones inutilizadas y la partida se reiniciará, volviendo éste al estado «Parado».

Para terminar con el objeto, cabe mencionar que uno de los fragmentos de código más interesantes del mismo es el que se encarga de manejar la máquina de estados, que se puede consultar en la Figura 9.5, simplemente se encarga de detectar si desde cierto estado se puede pasar a otro estado de la máquina. Por ejemplo, una vez nos encontremos en el estado «Escalando», el fragmento no permite que nos dirijamos a ningún otro estado de la máquina, ya que la experiencia de juego no sería correcta si esto ocurriese.

2. **Donkey Kong:** Objeto que se encuentra en la esquina izquierda superior de la pantalla y se encarga de manejar al antagonista del videojuego. Simplemente genera de manera aleatoria un clon del tonel marrón o del tonel azul e inicia la secuencia de estos. Cada vez que se genera una señal (de manera aleatoria) se ejecuta un fragmento que cambia los disfraces para que simulen el agarre de los toneles que tiene a la izquierda y los suelte en las losetas.

3. **BarrilFuego:** Este objeto permanece inmóvil en la parte izquierda inferior del Escenario del primer nivel de juego, y su función es la de destruir los toneles que colisionan con él y generar elementales de fuego cada vez el tonel colisionado es de color azul. El fragmento de código más interesante del objeto es el que se encarga de manejar el cambio de disfraces para simular

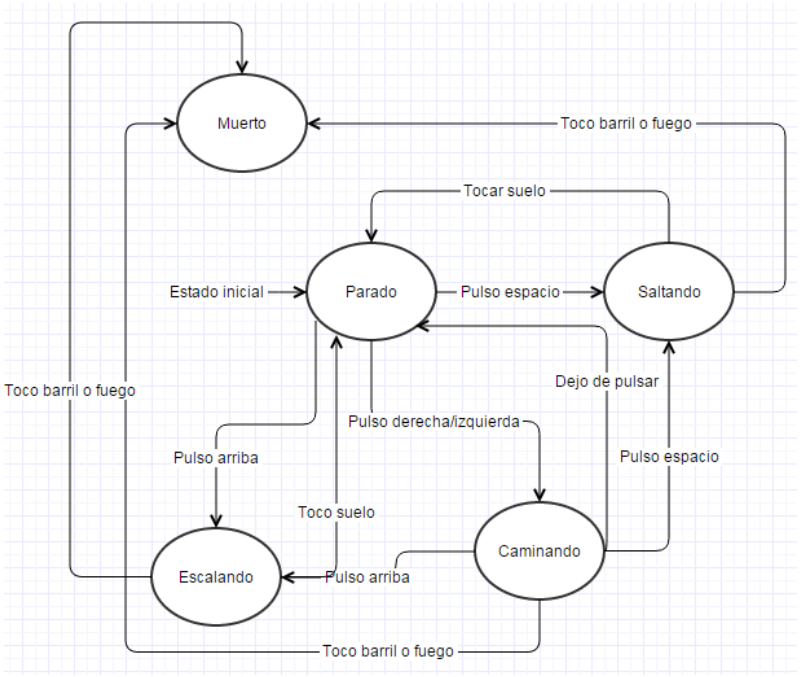

Figura 9.4: Máquina de estados que permite el manejo de las diferentes acciones que puede realizar el objeto Mario, el cual es manejado por el usuario.

que el fuego que sale de éste se mueve, simplemente cambiando cada cierto intervalo entre los dos disfraces del objeto.

4. **BarrilAzul:** Es el objeto que inicia el primer nivel del videojuego, ya que un vez hemos comenzado la partida, *Donkey Kong* lanza un primer barril el cual se dirige directamente hacia el Barril de fuego, y una vez colisiona con éste, se genera el primer elemental y empiezan a caer los toneles por las losetas. El fragmento de código más interesante del objeto es el que maneja la colisión con el «BarrilFuego» y manda la señal para generar el elemetal.

5. **Toneles:** Son los dos objetos que manejan los toneles marrones y azules. El objeto tonel marrón se encuentra oculto durante toda la duración de la partida, y simplemente se encarga de generar un clon al lado del objeto *Donkey Kong* cada vez que éste manda la señal para crearlo. Una vez iniciado simplemente sigue la secuencia de descenso hasta que colisiona con el objeto «Mario» o con el objeto «BarrilFuego» y es destruido. El objeto tonel azul se diferencia del marrón en que una vez ha colisionado con el Barril de fuego, éste genera una señal la cual crea un clon del objeto «Elemental».

6. **PrincesaRosa:** Objeto que se encarga de simular la princesa que ha sido capturada por el malvado del videojuego *Donkey Kong*. Simplemente se encarga de animar dos disfraces que simulan una princesa pidiendo auxilio al protagonista cada intervalo entre dos y cinco segundos.

7. **Elemental:** Objeto programado que se genera una vez que recibe la señal «inicioFuego», la cual es mandada por el objeto «BarrilFuego» una vez ha colisionado con un tonel de color azul. El objeto simula un elemental de fuego el cual va ascendiendo por los niveles hasta que desaparece saliendo del escenario. Su misión es la de acabar con el jugador colisionando con éste y provocando que se acabe la partida.

8. **Sensores:** Consta de cinco objetos, cada uno de los cuales se encarga de detectar ciertas situaciones o acciones durante la partida. El primero de ellos es el sensor de las escaleras, y se encarga de detectar cuando Mario se encuentra sobre una de ella y permitir que éste ascienda al siguiente nivel del Escenario (todos los sensores se encuentran ocultos al usuario, aunque permanecen en el escenario de juego).

   El segundo es el sensor de la bajada de escaleras, que permite detectar cuando Mario toca el suelo de un nivel para mandar una señal al fragmento que maneja su máquina de estados y los disfraces, ya que una vez llegamos al suelo del nivel deseamos que su estado y disfraz cambien, como se ha explicado en el apartado del objeto.

   El tercer sensor es el que se encuentra en la parte superior de los toneles que lanza *Donkey Kong*, y su misión es la de detectar la colisión con Mario y enviar la señal «textoOn», la cual hace aparecer el texto de puntuación en la posición de Mario para indicar que se ha obtenido el salto con éxito, el fragmento que genera dicho comportamiento se puede consultar en la Figura 9.6. El cuarto de los sensores se encarga de detectar el cambio de piso dentro del nivel uno de juego, cuando Mario alcanza el final de una escalera y colisiona con dicho objeto, se aumenta en uno el contador de piso (no existe sensor de descenso de piso, ya que una vez llegamos a un piso, Mario no puede descender de nuevo). Por último, el sensor de victoria se encarga de detectar cuando Mario llega al piso en el que se encuentra la princesa secuestrada, y simplemente al colisionar con éste envía la señal de «victoria» la cual activa el objeto «GameOver» y finaliza la partida.

9. **GameOver:** Es el objeto que aparece una vez Mario alcanza la cima del nivel y rescata a la princesa rosa, y simplemente se encarga de enviar la señal «volverMenuOver» la cual recarga y muestra todos los objetos del menú así como el fondo, y la señal «inicioMenu», la cual se encarga de inicializar las animaciones de éste.

10. **PuntuaciónSalto:** Se encarga de manejar la muestra de puntuaciones cuando saltamos por encima de un tonel, mostrando el total de puntuación obtenido en la posición de Mario, y dando un efecto de desvanecimiento sobre el número mostrado. La señal que activa este objeto la manda el sensor colocado en la parte superior de los toneles cuando colisiona con Mario (el objeto se encuentra oculto para el usuario).

9.3 Posibles ampliaciones

Puesto que *Donkey Kong* es una videojuego dividido en varios niveles, en el trabajo se ha querido mostrar cómo sería la creación del primero de los posibles niveles, por lo que una de los posibles ampliaciones sería la de añadir más niveles al videojuego, por los cuales se va a avanzando hasta llegar la nivel final en el cual el protagonista rescata a la princesa. Podríamos basarnos en los niveles del videojuego original, y crear un segundo nivel en el que la parte central del mapa fueran cintas las cuales arrastran al jugador hasta que caía de la cinta, mientras caen barriles para intentar acabar con nosotros. Otra posible variante de nivel sería incluir a los lados del mapa plataformas que aparecieran por la parte de abajo y fuesen subiendo niveles y en las que pudiéramos subir para avanzar más rápido.

Una ampliación que se podría hacer en el nivel uno es añadir el objeto del martillo (Figura 2.16) al primer nivel del juego, con el cual podríamos destruir los barriles y los elementales que aparecen en el mapa para obtener una puntuación extra. El objeto estaría en el piso dos del nivel uno de juego, y una vez lo tocásemos con el personaje, se añadiría el disfraz del martillo y lo iría moviendo de arriba a abajo, si tocamos con el martillo un barril o un elemental, éstos quedan eliminados y otorgarían puntos extra al usuario, en lugar de los puntos normales por saltarlos aunque si seguimos tocando los objetos con el cuerpo deberíamos ser eliminados nosotros. Para conseguir esto tendríamos que tener por un lado el disfraz del protagonista, y por otro lado e del martillo, así si la colisión que se produce es entre este último y uno de los dos objetos del mapa, ganaríamos puntos, y si la colisión se produce entre el disfraz del protagonista y uno de los objetos, seríamos eliminados.

El juego fue testeado por varios usuarios, los cuales se nombran en el apartado de créditos del mismo. Uno de los usuarios encontró un *bug* muy curioso, y es que si pulsabas la tecla del «uno» dentro del juego varias veces, o la mantenías pulsada a la vez que saltabas con el protagonista, se conseguía generar el efecto de vuelo en el mismo. Esto ocurría porque dicha tecla es la que se utiliza para iniciar la partida, pero no se bloqueaba una vez iniciado el juego, por lo que producía un efecto raro de reinicio del nivel pero quedando atrapado en un bucle infinito. La solución fue simplemente bloquear dicha tecla al inicio de la partida, y volverla a desbloquear al terminar el nivel.

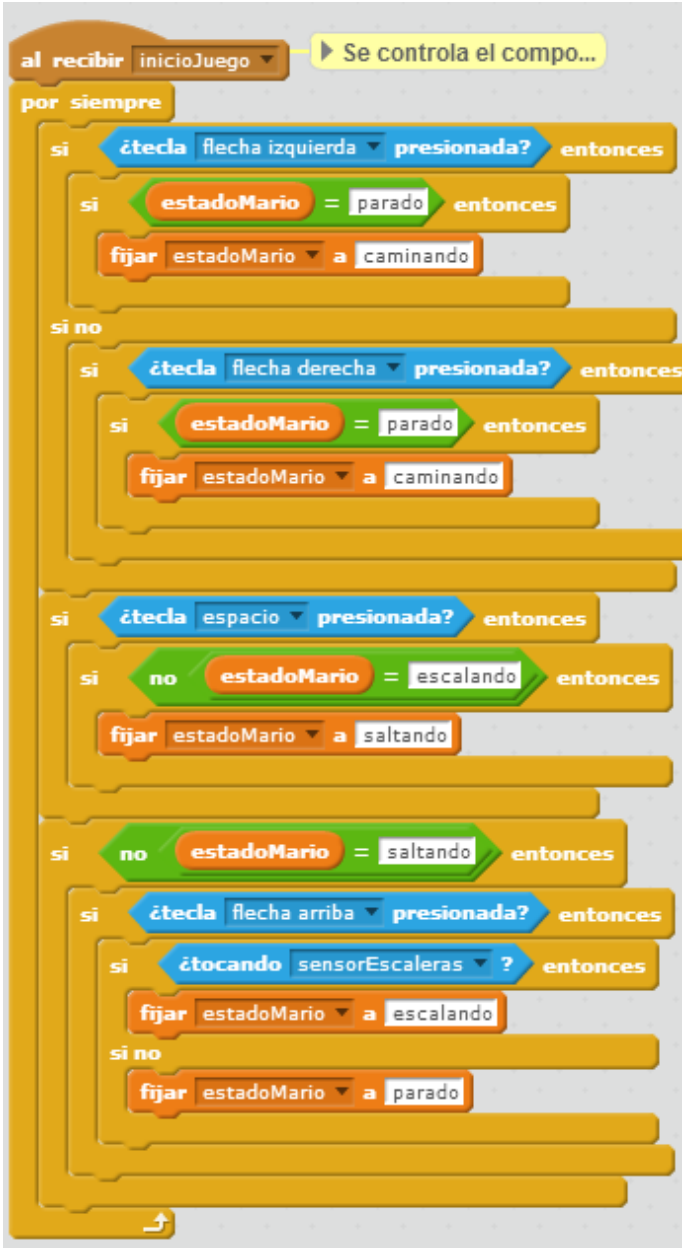

Figura 9.5: Fragmento de código que se encarga de manejar la máquina de estados del objeto Mario.

Figura 9.6: Fragmento de código que maneja el envío de señales y aumento de puntuación una vez que el sensor de los toneles colisiona con Mario.

CAPÍTULO 10

Página web diseñada

El siguiente capítulo se centra en la página diseñada para ser incluida en la web del museo de la escuela con los cinco videojuegos disponibles, así como un enlace en cada uno de ellos que nos permitirá acceder a la página de Scratch en la que podremos ver los comentarios de otros usuarios y podremos reinventarlos (así como acceder al código interno para visualizarlo).

10.1 Implementación

El principal objetivo del trabajo era la creación de cinco grandes videojuegos de la historia mediante el lenguaje de programación Scratch para así poder incluirlos en la web del Museu d'Informàtica de l'Escola Tècnica Superior d' Enginyeria Informàtica, y que los visitantes de éste pudieran disfrutar jugando con ellos y motivarlos de esta manera a comenzar con el aprendizaje de un lenguaje de programación.

Si nos fijamos en la Figura 10.1, podemos ver como se estructuran los videojuegos dentro de la web del museo. Se ha optado por incluir una breve descripción de cada uno de ellos, en la que se cuenta un poco de su historia y de sus creadores, para dar paso a una descripción de las reglas del videojuego y de su funcionamiento. Si nos fijamos debajo de cada uno, se encuentra disponible el enlace que nos lleva a la sección del videojuego en la página web de Scratch, en la cual los usuarios pueden comentar lo que quieran acerca del mismo, así como la opción de consultar el código interno para ver como se ha realizarlo o probar a reinventarlo para cambiar alguna funcionalidad o para mejorarlo y con esto animar a los visitantes del museo y de la página web a aprender programación de forma sencilla y divertida a través de la creación de un videojuego.

Los videojuegos se encuentran alojados en la página web de Scratch, ya que estos se ejecutan en su entorno de ejecución, por lo que para poder mostrarlos en la página web del museo se ha hecho uso de una funcionalidad disponible en la

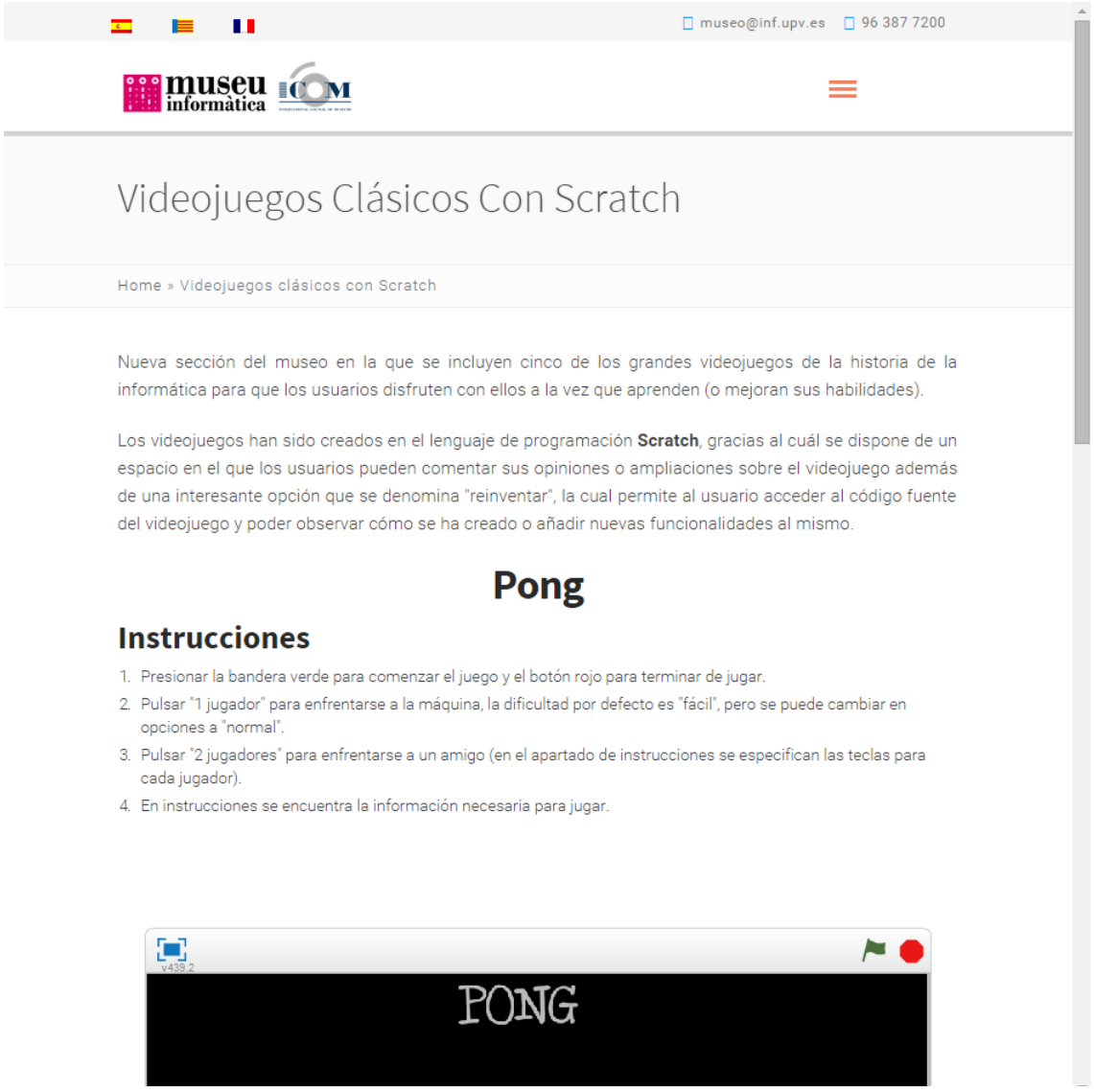

Figura 10.1: Ejemplo de uno de los videojuegos incluidos en la página web del museo en el que se pueden observar las instrucciones para jugarlo y una breve introducción al apartado de la web.

ficha de cada videojuego, que permite extraer el código embebido de éste, para así poder mostrarlo y jugarlo dentro de esta.

Por último, se encuentran disponibles los nombres del creador del videojuego y del tutor del trabajo, así como sus emails, por si alguien quiere consultar algo acerca de la funcionalidad de alguno de los videojuegos, algún *bug* encontrado etc. y así poder proporcionar ayuda o incluso seguir mejorando la calidad de cada uno de los videojuegos disponibles.

CAPÍTULO 11

Conclusiones

En este último capítulo se plantean unas consideraciones finales obtenidas a partir del trabajo desarrollado, realizando un resumen final y se revisan los objetivos propuestos al principio del mismo.

11.1 Consideraciones finales

Mediante el presente trabajo se ha demostrado que con un lenguaje sencillo como Scratch se pueden recrear fácilmente una serie de videojuegos retro. Para ello se han consultado una serie de libros, los cuales vienen descritos en la bibliografía, que han aportado las ideas básicas del lenguaje Scratch para poder comenzar con la tarea descrita en el documento.

Se ha puesto especial énfasis en el lenguaje Scratch como innovador y de actualidad, ya que como se explica en los capítulos tres y cuatro, éste ofrece una serie de herramientas a los usuarios junior que un lenguaje convencional como pueda ser Java no ofrece, ya que Scratch elimina por completo la fase de aprendizaje de la escritura de código, para dar lugar a un simple rompecabezas en el que el usuario simplemente tiene que escoger una serie de bloques y montarlos de manera que se genera un fragmento de código totalmente funcional y que maneja un objeto determinado (pasaríamos de líneas de código y clases a un objeto y sus fragmentos de bloque de código). Con esto conseguimos eliminar la fastidiosa tarea de iniciar a un usuario en las bases del lenguaje para, directamente, ponerlo a crear la funcionalidad del objeto y además ofrecerle la posibilidad de crear algo con lo que divertirse a la vez que aprender (los videojuegos descritos).

Por ello, lo primero que se ha realizado en el trabajo ha sido la introducción a la historia de los cinco videojuegos escogidos (*Pong*, *Asteroids*, *Space Invaders*, *Pac Man y Donkey Kong*) para que el usuario entienda por qué marcaron una época y pueda empaparse de las bases que se van a seguir para crear dichos artefactos. A continuación se ha pasado a explicar por qué el lenguaje escogido para el trabajo

ha sido Scratch, y cuáles son las herramientas que ofrece al usuario junior para facilitar la labor de introducirse en el mundo de la programación, a la vez que se divierte creando videojuegos de manera sencilla.

Lo segundo que narra el documento son las partes más importantes de cada uno de los videojuegos que se han creado con el lenguaje de programación Scratch, así como el funcionamiento de los mismos. Cada uno está dividido en el funcionamiento de su menú interno, por el cual podemos navegar a través de las diferentes escenas que lo componen, las funcionalidades de cada uno de los objetos que componen el escenario de juego y cómo estos interactúan a través de las señales que facilita Scratch para producir eventos. También se ha incluido un apartado en cada uno de los capítulos con las posibles ampliaciones que podrían llevarse a cabo si alguien decidiera escoger este trabajo para ampliarlo.

Por último, aunque no menos importante, se describe un capítulo en el que se narran las intenciones del presente trabajo en el que se explica que se ha diseñado un apartado en la página web del Museo de Informática de la Escola Tècnica Superior d'Enginyeria Informàtica en la que se podrá observar cada uno de los cinco videojuegos diseñados, con una pequeña descripción y una bases para poder jugar con ellos, además de la posibilidad de poder reinventar cada uno de los mismos, ya que Scratch dispone de una gran plataforma para la difusión de material informático, con la opción de observar el código interno de cada uno de los proyectos y la opción de reimplementarlo para añadir funcionalidad a los mismos. Se espera que el trabajo despierte el interés de los usuarios en el mundo de la programación y se animen a probar con algo divertido como pueda ser la creación de un pequeño videojuego.

11.2 Líneas de futuro

Para terminar con el desarrollo del trabajo, cabe mencionar que en cada uno de los capítulos dedicados a explicar el desarrollo de los videojuegos se ha incluido un apartado de posibles ampliaciones, en el que se describen algunas de las mejoras que se podrían realizar para ampliar la funcionalidad del videojuego a la hora de partir de este trabajo. Como añadido, si alguien decide escogerlo, se podría completar el catálogo de videojuegos con otros dos grandes de la historia, que podrían ser *Super Mario Bros* y *Double Dragon*.

El primero de ellos es el referente mundial de los videojuegos de plataformas, y debería estar presente en esta colección de videojuegos retro. Si nos centramos en el tema de desarrollo del mismo, podríamos realizar el primer nivel del videojuego, ya que llevar a cabo el desarrollo completo del mismo podría llevar demasiado tiempo. Como temas a tener en cuenta la hora del desarrollo, habría que hacer especial énfasis en el tema de las colisiones de Mario con los enemigos del mapa, ya que si éste colisiona con el cuerpo, debe morir, y si colisionamos con la parte inferior de lo que serían las botas, debemos acabar con el enemigo, por

lo que un tema muy importante a la hora del diseño del videojuego es dejar claro como deben desarrollarse las fases de colisión entre objetos.

El segundo y último es uno de los referentes y más célebres juegos de lucha. A la hora del desarrollo, se podrían incluir dos luchadores (cosa a ampliar en futuras versiones) con una serie de combos[1] a realizar. Podríamos tener el típico golpe de puño, la patada y un combo especial como pudiera ser el lanzamiento de una bola de energía o algo similar (en la primera versión ambos personajes realizarían los mismos ataques). A la hora del diseño del mismo, tendríamos que llevar especial cuidado en definir bien la máquina de estados que nos permitiera detectar cuando debemos realizar la secuencia de disfraces del puñetazo, el patazo etc. así como las colisiones del videojuego, ya que se debería detectar que objeto es el que golpea al otro (tema muy delicado en la resolución de las colisiones).

[1] Abreviatura de combinación usada en los videojuegos de lucha que consiste en un conjunto de acciones que se realizan en secuencia y producen un beneficio.

Bibliografía

[1] DILLON, R. (2011). *The Golden Age of Video Games*. New York: Taylor and Francis Group.

[2] ESPAÑA SANJUAN, C. (2015). *Diseño de actividades educativas en Scratch para la dinamización del Museo de Informática*. Trabajo Fin de Grado.

[3] ERICH, G., HELM, R., JOHNSON, R., VLISSIDES, J.M. (2002). *Design Patterns*. New York: Addison-Wesley.

[4] GARRIDO, S. (2015). *Scratch para niños... y no tan niños*. Francia: Amazon Media EI S.à.r.l.

[5] GOLDBERG, M. VENDEL, C. (2012). *ATARI Inc. Business Is Fun.* Unites States of America: Company Press.

[6] GUTIÉRREZ OROZCO, J.A. (2008). *Máquinas de estados finitos*. Ciudad de México: Escuela Superior de Cómputo.

[7] LÓPEZ GARCÍA, J.C. (2013). *Guía de referencia de Scratch 2.0*. Colombia, EDUTEKA, <http://www.eduteka.org/pdfdir/ScratchGuiaReferencia.pdf>, [Consulta: junio de 2015.]

[8] LÓPEZ BARINAGA, B. (2010). *JUEGO Historia, Teoría y Práctica del Diseño Conceptual de Videojuegos*. Madrid: Alesia (Games and Studies).

[9] MARJI, M. (2014). *Learn to Program with Scratch: a visual introduction to programming with games, art, science and math*. San Francisco: No Starch Press.

[10] MCMANUS, S. (2013). *Scratch programming in Easy Steps*. United Kingdom: East Steps Limited.

[11] MOTT, T. (2011). *1001 Videojuegos a los que hay que jugar antes de morir*. Grijalbo Ilustrados.

[12] MUVIM. (2014). *Del Tilt al Byte*. Diputació de València. Museu Valencià de la Il·lustració i la Modernitat.

[13] ORTIZ, A. Xataka. *Treintañeros jugones y sus nostalgias*. <http://www.xataka.com/especial/retrogaming-nostalgia-videojuegos>, [Consulta: 15 de febrero de 2015.]

[14] POLLOCK, W. (2010). *Super Scratch programming adventure*. Canadá: The LEAD Project.

[15] PRUDENCIO, M. (2013). *Una herramienta lúdica de iniciación a la programación, SCRATCH.* Linux Magazine, 28:78-82.

[16] RESNICK, M. (2013). *Lifelong Kindergarten.* Cultures of Creativity, LEGO Foundation. <http://web.media.mit.edu/~mres/papers/CulturesCreativityEssay.pdf> [Consulta: Junio 2015]

[17] RESNICK, M. et al. (2009). *Scratch: Programming for all*. Communications of the ACM, 52(11):60-67.

[18] SERRANO, G. (2015). *Programación Scratch para niños.* Francia: Amazon Media EI S.à.r.l.

[19] STEINBERG, S. (2007). *Videogame marketing and PR*. Power Play Publishing.

[20] VAN PUL, S. CHIANG, J. (2014). *Scratch 2.0 Game Development*. United Kingdom: Packt Publishing Ltd.

[21] WING, J.M. (2006). *Computational Thinking. Viewpoint*. Communications of the ACM, 49(3):33-35.