UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Arqueología informática: diseño e implementación de videojuegos clásicos con Scratch

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Patricia Ruá Lozano
Tutor: Xavier Molero

Curso 2016-2017

Resum

Aquesta memòria pertany al treball fi de grau en què es van a implementar videojocs clàssics (Space Invaders, Breakout i Puzzle Bobble) que en la seua època van ser pioners dins del seu gènere. Per al desenrotllament s'utilitzarà la plataforma de Scratch, un llenguatje de programació senzill i interactiu. Una vegada acabat el treball, passarà a estar a disposició del Museu d'Informàtica de l'Escola Tècnica Superior d'Enginyeria Informàtica (UPV) , el projecte es pujarà a la pàgina web del museu amb l'objectiu de demostrar la potència que suposa Scratch per a la programació de videojocs. Quant a l'estructura de la memòria, primer es realitzarà una introducció dels videojocs situant-los en el seu context històric seguit d'una explicació detallada del llenguatge Scratch empleat per a l'elaboració del projecte i a continuació es descriurà pas a pas el desenrotllament dels videojocs de manera que el lector puga observar i aprendre com utilitzar Scratch per a programar videojocs.

Paraules clau: disseny de videojocs, videojocs clàssics, màquines recreatives, pensament computacional, Scratch, programació.

Resumen

Esta memoria pertenece al trabajo fin de grado en el que se van a desarrollar tres videojuegos clásicos (Space Invaders, Breakout y Puzzle Bobble) que en su época fueron pioneros dentro de su género. Para el desarrollo se utilizará la plataforma Scratch, un lenguaje de programación sencillo e interactivo. Una vez terminado el trabajo, pasará a estar a disposición del Museo de Informática de la Escuela Técnica Superior de Ingeniería Informática (UPV), el proyecto se subirá a la página web del museo con el objetivo de demostrar la potencia que supone Scratch para la programación de videojuegos. En cuanto a la estructura de la memoria, primero se realizará una introducción de los videojuegos situándolos en su contexto histórico seguido de una explicación detallada del lenguaje de programación Scratch empleado para la elaboración del proyecto y a continuación se describirá paso a paso el desarrollo de los videojuegos de forma que el lector pueda observar y aprender cómo utilizar Scratch para programar videojuegos.

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

Abstract

This memory belongs to the Trabajo Fin de Grado where I will develop classic video games (Space Invaders, Breakout and Puzzle Bobble) which in their time were pioneers in their genre. In this work I have used Scratch, Scratch is a video game development language simple and interactive. Once work is completed, it will be available to the Museo de Informática de la Escuela Superior de Ingeniería Informática (UPV), the project will be uploaded to the website of the museum with the aim of demonstrating the power of Scratch for video game development. As for the structure of memory, first an introduction to the video games by placing them in their historical context, followed by an explanation of Scratch that I used for the development of the project and then I will describe step by step the development for each video game so the reader can observe and learn how to use Scratch to develop video games.

Key words: video game design, retro video game, recreational machines, computational thinking, Scratch, programming.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este primer capítulo se expondrá la motivación por la que se ha llevado a cabo el siguiente trabajo además de sus objetivos, la estructura de la memoria y su bibliografía.

1.1 Motivación

La principal motivación para realizar este proyecto ha sido mi pasión por la programación de videojuegos. Mi objetivo es dedicarme a la industria del videojuego. En cuanto vi que existía la posibilidad de programar videojuegos para el Trabajo Final de Grado supe que ese sería mi proyecto.

Otra de las razones por la que se ha realizado este proyecto es demostrar que Scratch es un lenguaje capaz de emular videojuegos clásicos a pesar de su sencillez. Scratch es un lenguaje de programación interactivo que su objetivo es enseñar al público a programar. Entre las actividades que realiza el Museo de Informática se encuentran los talleres de aprender a programar con Scratch por ese motivo este proyecto será subido a su página web para que sus usuarios disfruten a la vez que aprenden.

1.2 Objetivos

Como objetivo general es aprender a programar en el entorno de programación Scratch realizando tres influyentes videojuegos en la historia del videojuego. De esta forma se podrá observar cómo videojuegos que antiguamente suponían un gran esfuerzo tanto tecnológico como físico para su desarrollo, actualmente se pueden imitar en un entorno de programación orientado a todo el público como es Scratch. Aparte de la programación de videojuegos, también es importante conocer la historia de estos videojuegos y sus características.

Los videojuegos escogidos han sido Space Invaders, Breakout y Puzzle Bobble. Estos videojuegos han sido escogidos ya que fueron los videojuegos que marcaron la diferencia en sus épocas. Además, cada videojuego tiene ciertas particularidades a la hora de programar. Space Invaders ha sido escogido ya que fue uno de los primeros videojuegos de disparos. Breakout fue un videojuego de mi infancia y me parecía interesante que fuese inspirador en la creación de un virus llamado "*Boucing ball*". Por último Puzzle Bobble ha sido escogido por su mecánica de explotar las burbujas de su mismo color.

Además, otro de los objetivos de este trabajo es poder compartir el trabajo realizado en la página web del Museo de la Escuela de Ingeniería Informática para que sus usuarios disfruten de estos clásicos videojuegos.

1.3 Estructura de la memoria

El trabajo realizado consta de 8 capítulos, los cuales se explican brevemente a continuación:

- **Capítulo 1:** En él se expone la motivación por la que se ha escogido este proyecto, los objetivos planteados, su estructuración y comentarios sobre la bibliografía utilizada.
- **Capítulo 2:** En este capítulo se ha realizado un estudio histórico de cada videojuego explicando los sucesos más importantes a parte del objetivo y apariencia del videojuego en su época.
- **Capítulo 3:** En él se detalla desde la historia hasta la funcionabilidad del lenguaje Scratch con su plataforma y sus herramientas de programación.
- **Capítulo 4, 5 y 6:** En estos capítulos se explica detalladamente la realización de cada videojuego. Se explican las modificaciones que se han realizado respecto al videojuego original y su implementación en Scratch.
- **Capítulo 7:** En él se explica cómo se ha realizado la página incluyendo los tres videojuegos realizados además de su aspecto final.
- **Capítulo 8:** En este capítulo se realiza una opinión personal sobre el lenguaje utilizado sus ventajas y desventajas y también un análisis de los objetivos alcanzados.

1.4 Bibliografía

En esta sección se va a comentar la bibliografía empleada durante la realización de la memoria.

Para poder explicar el contexto histórico de cada videojuego se ha investigado la época en la que fueron creados y qué tecnologías fueron las empleadas en ellos. Para ello, se ha consultado los puntos [2, 3, 7, 11], en estos artículos se habla de los grandes videojuegos que lograron llamar la atención en la edad de oro de los videojuegos además de la situación histórica de los videojuegos seleccionados.

En cuanto a la estructura de la memoria, se han consultado los trabajos de fin de grado de dos antiguos alumnos de la escuela, Samuel Villaescusa y Miguel Marqués. Estos dos trabajos se pueden consultar en [6, 12]. Este trabajo resulta ser la continuación de los dos trabajos consultados por lo que se ha optado por una estructura similar a ellos: el primer capítulo constará del contexto histórico y objetivos de cada videojuego continuando con la explicación del lenguaje de programación Scratch. Después se explicará detalladamente el diseño e implementación de cada videojuego.

Para conocer cómo surgió el lenguaje de programación de Scratch se han consultados los puntos [1, 9], en ellos se conocieron las ventajas y desventajas de este lenguaje para niños y quiénes fueron sus creadores. Todo lo relacionado con las partes de este lenguaje y su funcionamiento basado en bloques se ha encontrado en los artículos [5, 8, 10], además del pensamiento computacional escrito por Jannette M Wing (consultar [13]).

Por último, otro de los documentos a tener en cuenta es [4]. Todos los videojuegos se basan en máquinas de estado, esto quiere decir que los elementos del videojuego cambiarán su estado cuando suceda una acción. Por ello es importante esta fuente, en el documento se nos explican conceptos relacionados con esta máquina de una forma intuitiva. Las máquinas de estado ayudan a diseñar e implementar los videojuegos.

CAPÍTULO 2
Historia de los videojuegos arcade

Los orígenes del videojuego se remontan a la década de 1950. Poco después de la Segunda Guerra Mundial se llevaron a cabo los primeros intentos por implementar programas de carácter lúdico.

Sin embargo, las primeras máquinas recreativas aparecieron en la década de los 70 gracias al descenso del coste de fabricación. En la figura 2.1 se pueden observar ejemplos de máquinas recreativas de esa década. Computer Space (1971) o Pong (1972) fueron los videojuegos que inauguraron las primeras máquinas recreativas[1].

Figura 2.1: Primeras máquinas recreativas de Computer Space y Pong.

2.1 Space Invaders

Space Invaders es uno de los primeros videojuegos arcade[2] diseñado por Toshihiro Nishikado y lanzado al mercado por primera vez en 1978. Fue vendido por la empresa Taito Co. en Japón, y posteriormente distribuido en Estados Unidos por Midway Games.

[1] Aparatos electrónicos donde el juego transcurre en una pantalla y se maneja al personaje mediante un *joystick* y botones.

[2] Término genérico para las máquinas recreativas disponibles en lugares públicos.

El objetivo del juego es eliminar a los alienígenas con disparos láser y obtener la mayor puntuación posible.

Space Invaders es uno de los primeros juegos *shoot 'em up*[3]. Fue uno de los precursores de los videojuegos y de gran ayuda a la expansión de este sector. Un juego exitoso y popular desde su lanzamiento.

2.1.1. Historia

Taito era una pequeña compañía japonesa fundada en 1953 dedicada al desarrollo, distribución y comercialización de videojuegos. Toshihiro Nishikado era uno de sus programadores y para uno de sus diseños se le ocurrió la idea de destruir filas de enemigos mientras se acercaban hacia la nave espacial controlada por el jugador.

De ese modo fue como nació Space Invaders en 1978, dando a conocer el mundo de los videojuegos. Su enorme éxito lo derivó a un fenómeno comercial.

Debido a que Space Invaders no estaba sujeto a *copyright*, no tardaron en aparecer las primeras copias sacadas a la venta como: Space Invaders Deluxe, Super Invaders o Fast Invaders. En la figura 2.2 se pueden observar las máquinas recreativas de Space Invaders original y su copia Space Invaders Deluxe.

Figura 2.2: Comparativa entre dos máquinas recreativas de Space Invaders.

Además, el juego tuvo innumerables adaptaciones a lo largo de los años. Una de los primeras fue "Space Invaders 95: The Attack Of Lunar Loonies", y una de las más recientes para Nintendo DS "Space Invaders Extreme", así como diversas parodias del juego.

[3]Término anglosajón para definir un género de videojuegos en el cual el jugador debe disparar a sus enemigos.

2.1.2. El juego

Space Invaders consiste en destruir marcianos en dos dimensiones. El jugador controla una palanca para moverse de derecha a izquierda, además de un botón para disparar. El jugador debe vencer a 5 filas de alienígenas de los cuales existen tres tipos: con forma de calamar, de cangrejo y de pulpo. Estos marcianos irán aumentando su velocidad de movimiento a medida que el jugador va destruyéndolos. Si los invasores llegan al nivel de la nave controlada por el jugador, el juego termina. En la figura 2.3 se puede observar visualmente cómo era el videojuego.

Figura 2.3: Pantalla de la máquina recreativa Space Invaders.

Cada cierto tiempo, aparece en la parte superior de la pantalla un platillo volante que se mueve de derecha a izquierda, y al dispararle proporciona al jugador cierta cantidad de puntos extra indefinida. Además, se tienen cuatros escudos de protección para ayudar al jugador a no recibir las balas de los alienígenas, pero gradualmente se irán destruyendo tanto por las balas de los enemigos como los disparos del jugador.

2.2 Breakout

Breakout es un videojuego lanzado en 1975 y desarrollado por Atari, Inc. Fue diseñado por Nolan Bushnell y Steve Bristow influenciados por el videojuego Pong desarrollado por la misma empresa. Breakout se distribuyó en 1978 para dos tipos de videoconsolas: Atari 2600 y MSX. El objetivo del juego es que el usuario consiga romper todos los ladrillos rebotando una pelota.

2.2.1. Historia

Atari, Inc. fundada en 1972 en Estados Unidos, fue una de las productoras más grandes dentro de la industria de los videojuegos. Tras el primer gran éxito de la empresa,

Pong[4] (1972), Nolan Bushnell y Steve Bristow idearon en 1975 una versión simplificada de este. En la figura 2.4 se puede observar la versión original del videojuego Pong.

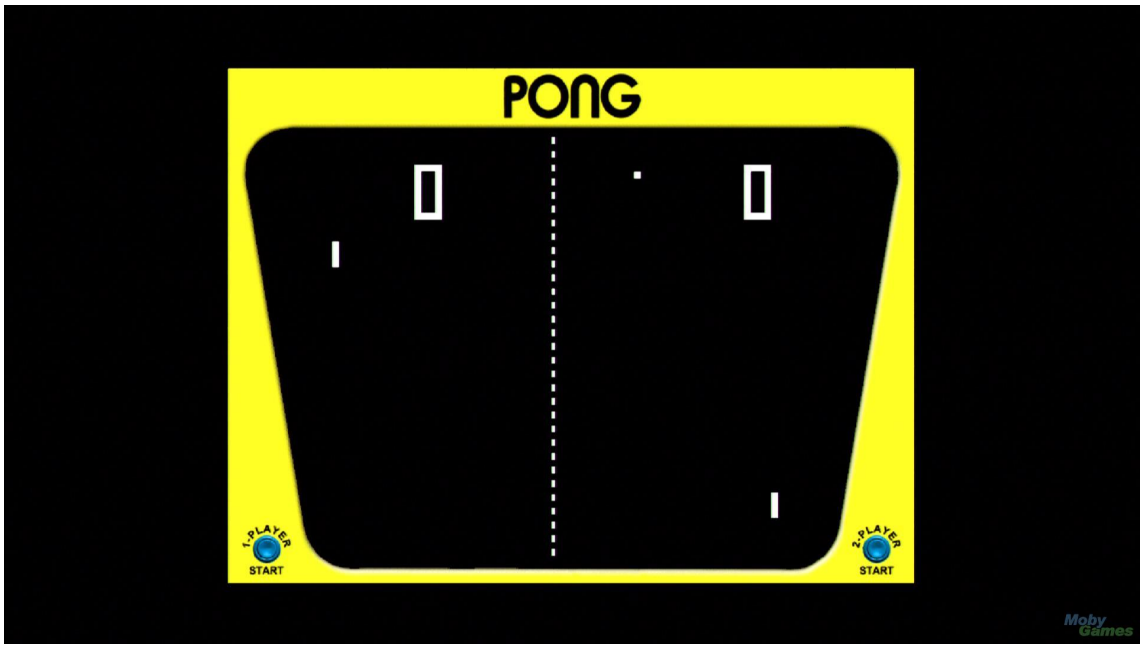

Figura 2.4: Versión original del gran éxito de Atari, Pong.

Los creadores de Breakout realizaron sencillos cambios a Pong: giraron la zona de juego 90 grados, cambiaron la paleta del segundo jugador por 8 filas de ladrillos y además, introdujeron un sistema de puntuación en el que cada ladrillo coloreado valía una cantidad de puntos distinta. Con estos cambios, Bushnell y Bristow crearon una versión para un jugador de Pong con nuevos desafíos para llamar la atención del público.

Al igual que otros juegos, Breakout fue clonado por fabricantes rivales, pero la ola de ordenadores y consolas domésticas hizo que el juego fuese más conocido y se pudiera jugar en todos los sistemas conocidos hasta la época. Atari continuó capitalizando el éxito de Breakout desarrollando nuevas versiones del juego original como: Breakout 2000 (1976) y Super Breakout (1978).

2.2.2. El juego

Breakout consiste en romper todas las filas de ladrillos situadas en la parte superior de la pantalla. El jugador mueve de izquierda a derecha una barra situada en la parte inferior. Se muestra en la figura 2.5 la versión original de Breakout.

Con esta barra, el jugador golpea una pelota que aparece de la nada y así poder rebotarla hasta llegar a los ladrillos y romperlos para conseguir más puntuación. Originalmente había 4 colores para los ladrillos, amarillo, verde, naranja y rojo. Cada uno de estos ladrillos coloreados tiene una puntuación distinta:

- Ladrillo Rojo : 7 puntos.
- Ladrillo Naranja : 5 puntos.
- Ladrillo Verde : 3 puntos.
- Ladrillo Amarillo : 1 punto.

[4]Videojuego de la primera generación de videoconsolas, basado en el deporte de tenis de mesa o Ping Pong.

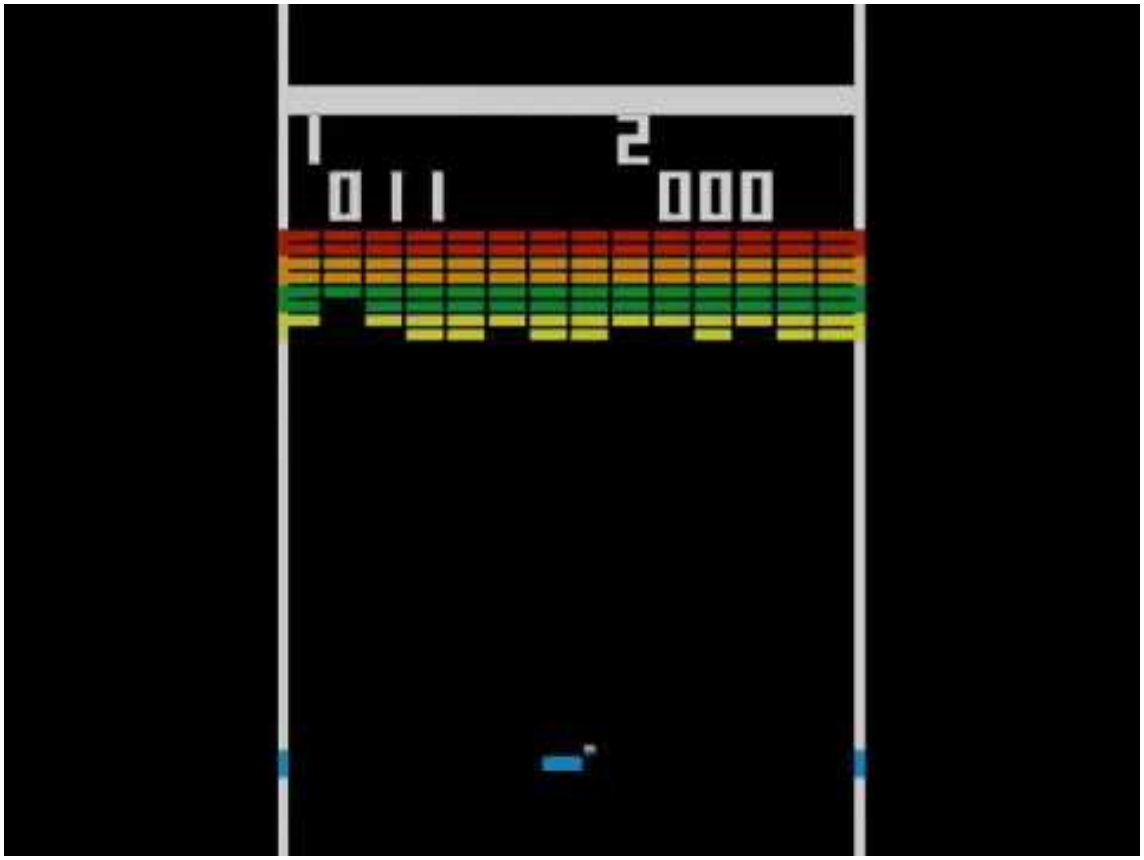

Figura 2.5: Primera versión de Breakout (1975).

2.3 Puzzle Bobble

Puzzle Bobble, también conocido como Bust-a-Move, es un videojuego de tipo puzzle[5], creado por la empresa Taito en 1994. La característica apariencia *kawaii* [6] de este videojuego, junto con su mecánica de juego y diseño de niveles hizo que Puzzle Bobble se hicieran tan popular en su época.

2.3.1. Historia

Taito, una compañía no tan conocida como Capcom, Atari o Konami, se puso en cabeza en los años de las máquinas recreativas con videojuegos como Space Invaders. La compañía fundada en 1953 por Michael Kogan, lanzó en 1994 un juego de plataformas tan sencillo como adictivo, Puzzle Bobble, inspirado en su exitoso juego Bubble Bobble del año 1986. En la figura 2.6 se puede observar la máquina recreativa del videojuego Bubble Bobble.

Se publicaron dos versiones distintas del videojuego. En junio de 1994 se publicó solo en Japón el videojuego original con el título de "Bubble Buster". Seis meses después apareció la versión Neo-Geo MVS[7] por Estados Unidos, Canadá y algunos países europeos. Era una copia idéntica al videojuego original, tan solo se diferenciaban en los textos traducidos y en algunos efectos de sonidos. Además, esta nueva versión transmitía mensajes contra la droga y a favor del reciclaje.

[5] Videojuegos caracterizados por exigir agilidad mental al jugador.
[6] Término japonés traducido como tierno o bonito.
[7] Sistema de 16 bits basado en cartuchos lanzado en 1990 para videoconsolas de hogar.

Figura 2.6: Máquina recreativa de Bubble Bobble.

2.3.2. El juego

El objetivo de este juego es eliminar todas las burbujas de la pantalla evitando que la pantalla se llene de ellas.

Al comienzo de cada ronda del juego, aparece una cierta cantidad de burbujas de colores colocadas según un patrón predefinido. En la figura 2.7 se puede observar su máquina recreativa. En la parte inferior de la pantalla, el jugador controla un puntero con el que apuntará y disparará burbujas hacia arriba. El color de las burbujas que el jugador podrá lanzar es escogido aleatoriamente.

Las burbujas disparadas por el jugador viajan en línea recta pudiendo rebotar con los bordes de la pantalla. Estas burbujas se detienen al tocar con otra burbuja o alcanzar la parte superior de la pantalla. Si se forma un grupo de tres o más burbujas del mismo color, dichas burbujas explotarán eliminando todas aquellas burbujas que cuelguen de estas. Cada pocos disparos del jugador, las burbujas descenderán un escalón generando una fila nueva en la parte superior.

Existe una versión para dos jugadores donde la pantalla se divide en dos con una distribución de burbujas idénticas para los dos jugadores. El objetivo de esta versión es eliminar más burbujas que el contrincante, ya que cada vez que un jugador elimina cuatro o más burbujas de golpe, se le transferirán al oponente. En la figura 2.8 puede observarse la interfaz del videojuego para dos jugadores.

Figura 2.7: Máquina recreativa de Puzzle Bobble.

Figura 2.8: Interfaz del videojuego Puzzle Bobble para dos jugadores.

CAPÍTULO 3
El entorno de programación Scratch

Scratch es un lenguaje de programación visual utilizado por una gran variedad de personas tanto niños como profesores académicos. En este capítulo se pretende explicar las ventajas de esta plataforma para desarrollar videojuegos, conocer un poco más este peculiar lenguaje y cómo funciona.

3.1 ¿Por qué Scratch?

Este proyecto se ha realizado en el entorno de desarrollo Scratch (figura 3.1). La principal ventaja que ha llamado la atención es su sencilla forma de escribir código, resulta una forma motivadora puesto que podemos ahorrar muchos esfuerzos y conseguir resultados rápidos.

Figura 3.1: Logo de Scratch junto con su lema "Imagina, Programa, Comparte".

Otra de las muchas ventajas que tiene Scratch es que está elaborado con código *Open Source*. Cualquiera tiene acceso al código fuente de Scratch para mejorar sus características ya que se trata de un software libre. Al tratarse de este tipo de programas, cualquier persona puede utilizarlo y es totalmente gratuito, fomentando la educación sin ningún coste.

La página web de Scratch, https://scratch.mit.edu, dispone de un editor *online*, con lo cual puede utilizarse desde cualquier lugar. Además cuenta con un editor *offline* para situaciones sin conexión a red o para mantener el trabajo realizado en nuestro equipo.

En resumen, aquí se presentan la ventajas más importantes de Scratch:

- Programa gratuito.
- Programa de software libre.
- Multiplataforma. [1]
- Está adaptado para múltiples idiomas.
- Dispone de un foro y videos tutoriales de ayuda.
- Los proyectos pueden ser compartidos en la web.
- Sencillo de utilizar.

3.2 El proyecto Scratch

Scratch es un lenguaje de programación educacional gratuito desarrollado por el laboratorio de investigación MIT (Massachusetts Institute of Technology). Actualmente, ya se ha desarrollado la versión 2.0 para sus dos plataformas: web y escritorio. Para conocer un poco más de esta versión, se sugiere consultar la wiki propia de Scratch. [8]

La primera versión de Scratch fue publicada en el año 2003 por el grupo de investigación Lifelong Kindergarten del MIT Media Lab [9], liderado por Mitchel Resnick. En la figura 3.2 se puede observar a los creadores de esta plataforma. Su propósito era desarrollar una plataforma en la cual los niños podían desarrollar sus capacidades intelectuales. Es una manera sencilla de desarrollar sus conocimientos de programación, además de su creatividad, y trabajo en equipo.

En la informática, *scratching* significa reutilizar código. La filosofía de Scratch es *Open Source*, así cualquier usuario puede realizar modificaciones y mejoras sobre proyectos públicos desarrollados por otros e incluso en el propio programa de Scratch. Además, se basa en la técnica *turntablism*, el arte de crear música mediante modificaciones de sonidos. Con esta técnica, cada usuario de Scratch puede aportar modificaciones para mejorar los proyectos subidos a la plataforma.

3.3 El pensamiento computacional

El pensamiento computacional, también conocido como *computational thinking*, se define como "Procesos de pensamiento involucrados en formular problemas y encontrar soluciones, de manera que las soluciones estén representadas tal que puedan llevarse a cabo por un agente que procesa información (humano o máquina)". Para conocer más información sobre este concepto, se puede consultar el artículo [13] escrito por Jannet M. Wing.

[1] Concepto atribuido a programas informáticos que son implementados para poder utilizarse en múltiples plataformas, en caso del Scratch: Windows, Mac y Linux.

Figura 3.2: Equipo LifeLong Kindergarten, creador de Scratch.

Scratch se basa en este pensamiento computacional para enseñar distintos conceptos informáticos tales como secuencia, iteración, datos, operadores, condiciones, ... Además, también pretende enseñar ciertas prácticas claves para la informática:

- Divide y vencerás: búsqueda de soluciones incremental.
- Probar y depurar: somos humanos y nos equivocamos, debemos probar y corregir los errores.
- Reutilizar y remezclar: a veces no nos damos cuenta que ya existen varias soluciones, solo debemos fijarnos si ese tipo de solución nos ayuda y adaptarlo a nuestro problema.
- Abstraer, modular y modelizar: crear pequeños modelos para gestionar la complejidad.

Resumiendo, Scratch ayuda en el proceso de aprendizaje basándose en cómo lo estás aprendiendo, y no en qué estás aprendiendo. Para ello, utiliza un amplio abanico de herramientas enfocadas a las fases de diseño e implementación de programas informáticos.

3.4 Programación de videojuegos con Scratch

Antes de empezar a diseñar e implementar los videojuegos propuestos para este proyecto, debemos conocer los diferentes paneles que constituyen Scratch. En este apartado se van a explicar las características más importantes de cada panel, además de su funcionalidad.

En la figura 3.3 se muestra la pantalla principal una vez abierto el programa, donde se pueden observar todos los paneles que se van a utilizar para crear videojuegos.

3.4.1. Escenario

En cada uno de los proyectos de Scratch se encuentra un escenario en el cual se van a desarrollar todas las acciones descritas en el panel de *scripts*. En el escenario es donde nuestro protagonista va a cobrar vida.

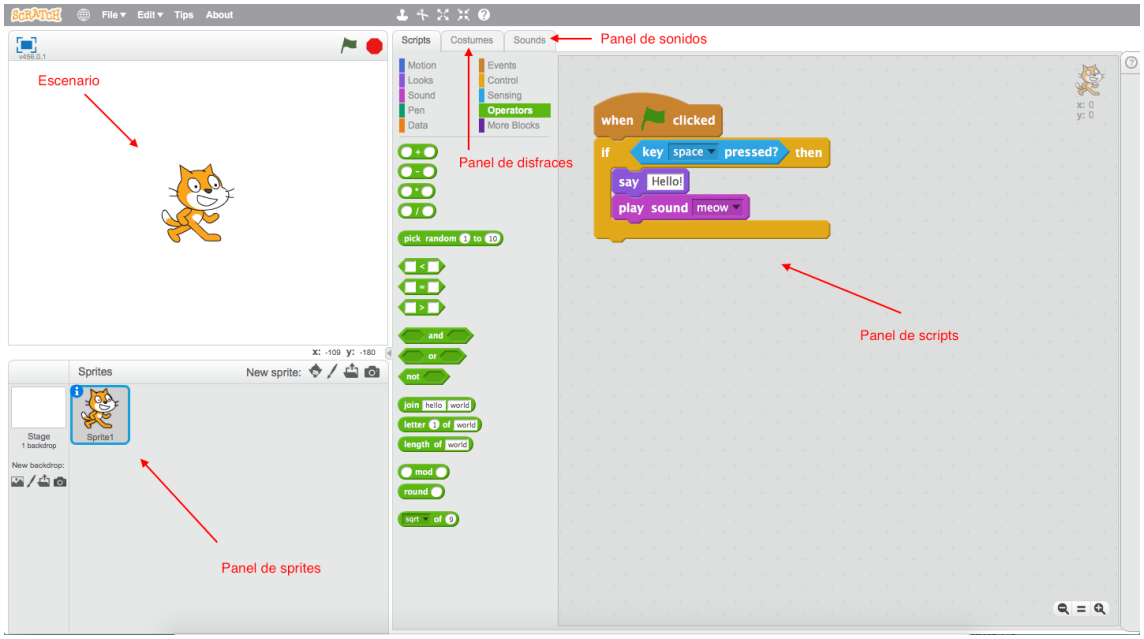

Figura 3.3: Pantalla principal de Scratch.

En la parte superior de la figura 3.4 se pueden observar 3 botones. Con el botón izquierdo podemos ampliar o disminuir la pantalla del escenario, con el botón de la bandera se podrá iniciar el modo de juego y por último, con el botón rojo lo pararemos.

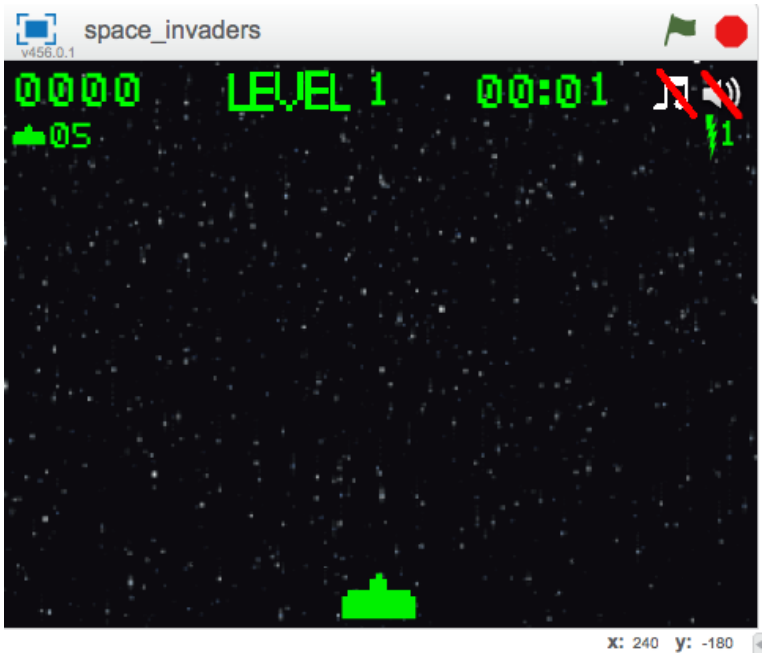

Figura 3.4: Ejemplo de la parte del escenario en el programa de Scratch.

3.4.2. Panel de *sprites*

Este panel es el más característico de Scratch, ya que lo hace distintivo de otros lenguajes de programación. La finalidad de Scratch con este panel es que el usuario tenga

todos los objetos necesarios organizados visualmente. Además, cada objeto contendrá *scripts*[2] que definen su funcionalidad.

Scratch permite crear *sprites*[3] de distintas maneras posibles. La primera de ellas permite al usuario escoger cualquier objeto, con distintas temáticas, de la librería que posee el programa. La segunda forma permite crear un objeto nuevo con las herramientas de dibujo proporcionadas. La tercera, el usuario podrá elegir cualquier imagen guardada en local e importarla como *sprite*. Finalmente, en la última opción, Scratch permite al usuario crear un objeto a partir de las capturas que realiza con la cámara del ordenador.

Todas estas posibles formas de crear objetos se encuentran en los botones de la parte superior derecha, como se muestra en la figura 3.5.

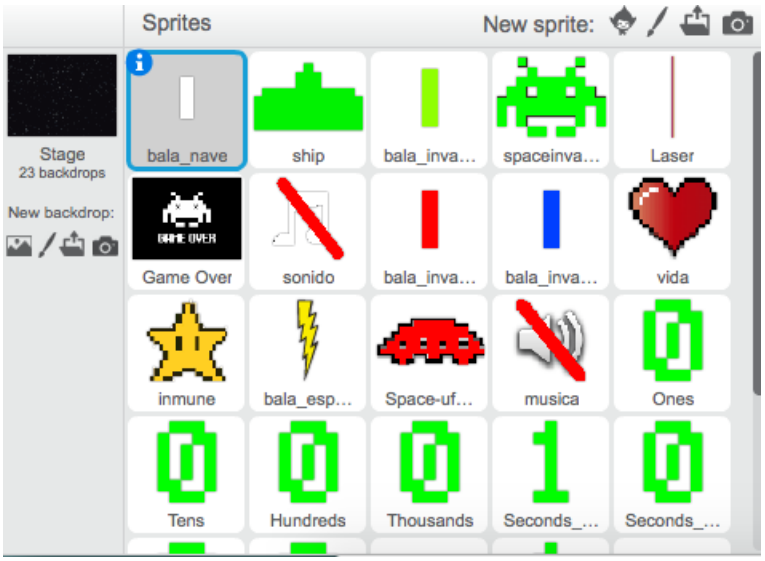

Figura 3.5: Ejemplo del panel de *sptrites* del programa Scratch.

Además de poder crear de distintas maneras los *sprites*, podemos conocer información de cada uno de ellos con tan solo pulsando en el icono de la «i» en la parte superior izquierda del objeto. En la figura 3.6 se puede observar gráficamente. Al pulsar este icono se podrá cambiar su nombre, conocer su posición en el escenario, su forma de rotación, y mucho más.

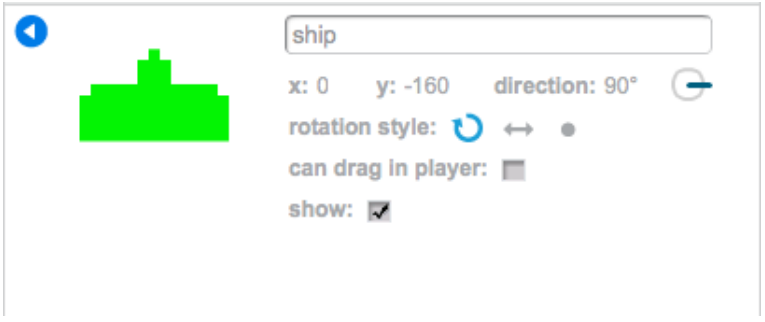

Figura 3.6: Ejemplo gráfico de la información básica de un *sprtite*.

Por último, en la parte izquierda del panel, se definen los escenarios donde los *sprites* cobrarán vida. Las distintas formas de importar un escenario son idénticas a las de un

[2]Fichero escrito en un lenguaje de programación que ejecuta todas aquellas funcionalidades que se quieren diseñar para un programa

[3]Son imágenes que poseen transparencias utilizadas para representar un objeto en videojuegos

sprite. Este objeto, será el encargado de controlar el estado del videojuego: cuándo se inicializa o finaliza el juego, como también las transiciones entre escenarios.

3.4.3. Panel de *scripts*

Como se ha mencionado anteriormente, cada *sprite* o escenario, contiene una serie de *scripts*, donde el usuario define la funcionalidad deseada de estos objetos. Scratch, a diferencia de otros lenguajes de programación, permite al usuario diseñar los *scripts* con una manera muy sencilla, arrastrando bloques y juntándolos como si fuesen piezas de LEGO.

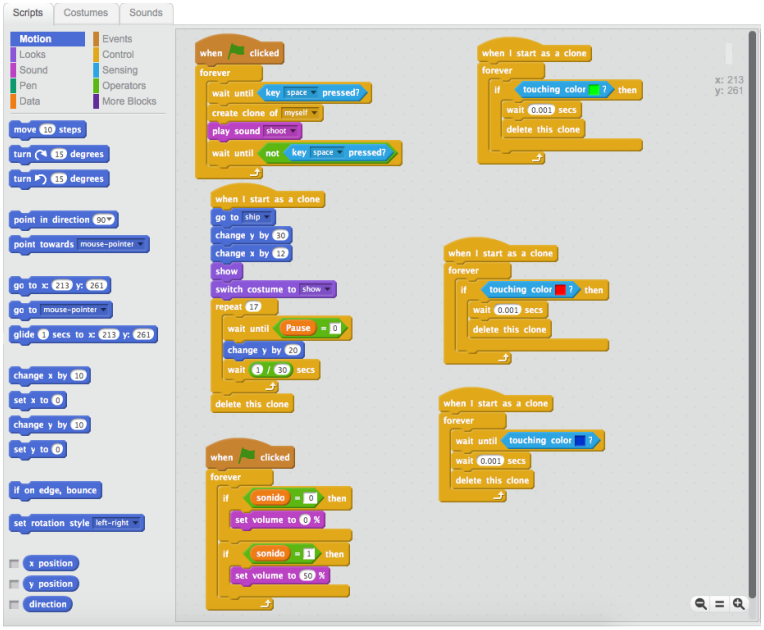

Figura 3.7: Ejemplo del panel de programas en Scratch.

Scratch proporciona distintas funcionales a los bloques (figura 3.7), caracterizándolos con colores para que sea más fácil de recordar y de visualizar, además de facilitar la programación. Se pueden encontrar todos estos tipos de bloques:

- **Bloque de movimiento o *motion*:** Este bloque se caracteriza por ser el encargado de los movimientos de nuestros objetos. Entre otras acciones, podremos mover nuestro objeto a un punto específico del escenario, mostrar su posición x e y actual e incluso, posicionarlo en diferentes ángulos. Este bloque se caracteriza por tener un color azul oscuro.

- **Bloque de apariencia o *looks*:** Como bien dice su nombre, este bloque es el encargado del aspecto de los objetos. Con él, podremos realizar cualquier animación con el objeto, como por ejemplo, mostrarlo u ocultarlo, cambiar el disfraz que posee, ... El color que define este bloque es el morado.

- **Bloque de sonido o *sound*:** Con este bloque podremos controlar los sonidos de cada uno de los objetos creados, cambiar el volumen, reproducir un sonido específico, parar la música,.... Este bloque se diferencia de los demás por tener un color rosado.

- **Bloque de lápiz o *pen*:** La funcionalidad de este bloque es un tanto curiosa, ya que podremos dibujar sobre el escenario en tiempo de ejecución del videojuego. Alguna

de sus acciones son subir o bajar el lápiz, cambiar de color, como también su grosor. El color definido para este bloque es el verde oscuro.

- **Bloque de datos o *data*:** Con este bloque podremos crear distintas variables locales para cada objeto. Una vez definida la variable, en este bloque, podremos inicializar su valor a 0, cambiarlo a cualquier valor, mostrarla en pantalla u ocultarla. Este bloque se caracteriza por su color anaranjado.

- **Bloque de eventos o *events*:** Este bloque es el encargado de gestionar los eventos que suceden durante el juego. Estos eventos pueden ser cuando se inicializa el juego e incluso eventos propios, esto quiere decir que un objeto puede crear un evento y los demás estar a la espera de recibir ese evento para ejecutar un determinado *script*. Se caracteriza por tener un tono más oscuro que el del bloque de datos.

- **Bloque de control:** Es el más utilizado en la programación de videojuegos. Con la ayuda de este bloque podremos conocer ciertas condiciones de nuestro objeto y ejecutar cierta funcionalidad dentro de ellas. Podremos realizar condiciones (*if-then-else*), bucles con o sin condiciones, bucles infinitos e incluso esperar a cierta condición. Su color característico es el amarillo.

- **Bloque de sensores o *sensing*:** El bloque de sensores nos permite descubrir ciertas situaciones que puedan ocurrir en el juego. Con este bloque nos ayuda a detectar colisiones entre objetos, colores e incluso saber cuando hemos pulsado cierta tecla o no. Lo cual nos facilita enormemente en la implementación de una de las partes más importantes de un videojuego. Este bloque se caracteriza por su color azul.

- **Bloque de operadores u *operators*:** Como su nombre indica, este bloque es el encargado de realizar cualquier operación matemática: suma, resta, mayor que, menor que, multiplicación, división, junto con otras acciones. Su color característico es el verde.

- **Bloque de más bloques o *more blocks*:** Este bloque permita al usuario experto ampliar las funcionalidad que ofrece Scratch, creando él mismo su propio bloque con ciertas características. Se caracteriza por su color morado oscuro.

Para conocer más a fondo los bloques proporcionados por Scratch y cada una de las funcionalidades de cada bloque, se sugiere consultar la guía [5].

3.4.4. Panel de disfraces

Como en el anterior panel explicado, cada objeto puede poseer distintos disfraces o *costumes*. En el panel de disfraces se puede gestionar las distintas imágenes de cada *sprite*. Este panel posee las cuatro opciones explicadas anteriormente, para poder importar o crear un disfraz. En la figura 3.8 se puede observar la interfaz de dicho panel.

Una vez conseguidos los distintos disfraces que queremos para nuestro objeto, utilizaremos el bloque de apariencia para mostrar cada uno de ellos en un determinado momento o condición. En el ejemplo de la imagen 3.8 del videojuego Space Invaders, podremos crear la animación de movimiento de los marcianos. Además, en este videojuego, se ha realizado que en cada nivel aparezcan con distintos colores: verde, rojo y azul. Una vez que les alcance la bala de la nave, aparecerá el disfraz de explosión. Con esto, podremos hacer el efecto de destrucción.

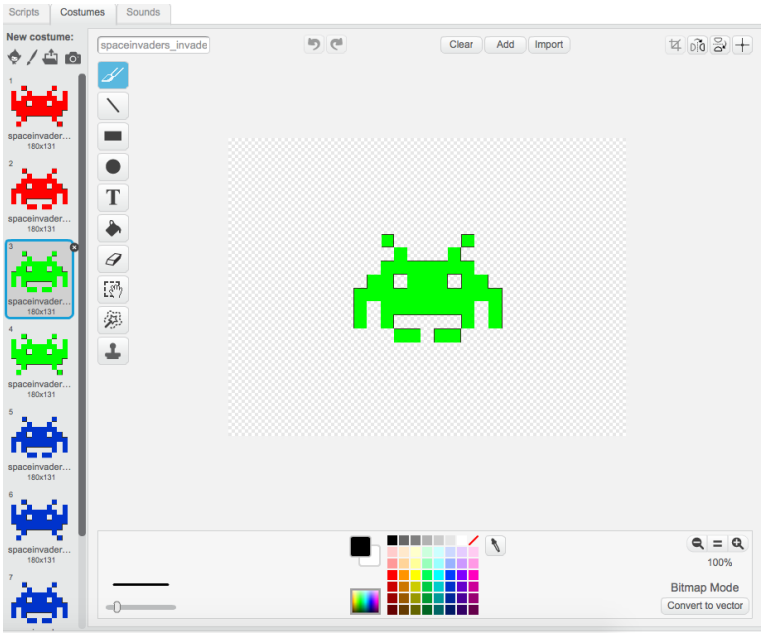

Figura 3.8: Ejemplo del panel de disfraces en la aplicación de Scratch.

3.4.5. Panel de sonido

El panel de sonido realizará la misma función que el panel de disfraces. Cada *sprite* poseerá distintos sonidos. Con el panel de sonido podremos crear o importar cualquier sonido para que la animación sea más real. En la figura 3.9 podemos ver un ejemplo de este panel.

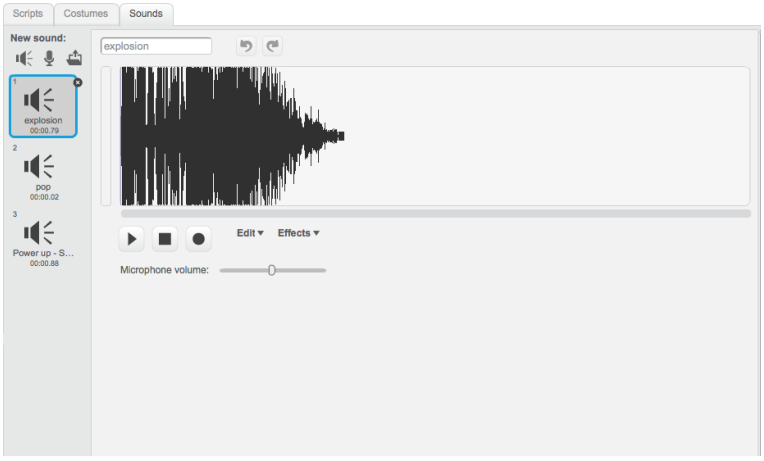

Figura 3.9: Ejemplo de la interfaz del panel de sonido en Scratch.

Al igual que con los disfraces, el usuario deberá escoger los distintos sonidos que querrá reproducir en el juego. Con la ayuda del bloque de *sounds*, en un momento determinado o condición del juego, podrá iniciar, cambiar o parar el sonido escogido. En la figura 3.9 vemos un ejemplo del videojuego Space Invaders, donde estos efectos de sonido han sido escogidos para la nave que controla el usuario. Estos tres sonidos se reproducen cuando la nave es alcanzada por una bala de los marcianos, cuando el usuario dispara o cuando la nave consigue el *power up* de inmunidad. Con estos efectos de sonidos, y con los disfraces ya seleccionados e implementados, lograremos una animación más realista.

CAPÍTULO 4
Space Invaders

En este apartado se va a explicar cómo se ha diseñado e implementado Space Invaders en Scratch. Antes de empezar, debemos saber que el objetivo de este videojuego es destruir a todos los marcianos que aparecen en la pantalla dirigiendo una nave que dispara.

4.1 Diseño del videojuego

Antes de ponerse a programar, debemos conocer ciertos datos principales del videojuego: en qué consiste, cómo se juega, las mecánicas a seguir, y si es posible, realizar un diagrama con lo esencial del videojuego, por ejemplo, un diagrama de estados.

Para poder familiarizarnos con el videojuego, se han probado varias imitaciones realizadas en Scratch para conocer el objetivo del juego y sus mecánicas más importantes. Gracias a estas imitaciones, ya podemos tener una idea de cómo se van a organizar nuestros *sprites* y los *scripts* de cada uno de ellos.

Conociendo la estructura de los demás usuarios de la comunidad de Scratch, se han encontrado las imágenes que se van a utilizar en cada *sprite* de nuestro Space Invaders. En la figura 3.5 se pueden observar la mayoría de ellos.

Además, se ha diseñado un diagrama de estados para el objeto principal, la nave. Se ha utilizado el programa SmartDraw[1]. En la figura 4.1 se puede observar el diseño final del diagrama de estados del objeto que controlará el usuario. En la realización de los videojuegos se ha llevado a cabo una nomenclatura inglesa, por lo tanto en las capturas de pantalla que se presentan veremos que la nave se le otorga el nombre de *ship*.

Se han encontrado nueve estados distintos para la nave: al iniciarse el juego, el jugador se encontrará **parado** en la parte inferior central de la pantalla. Pulsando las flechas izquierda y derecha, el jugador podrá observar como la nave se **mueve** a ambos lados, ayudándole a esquivar las balas de los enemigos, de lo contrario, el jugador perderá una vida hasta llegar a 0 (**muerto**). La nave podrá **disparar** balas contra los marcianos pulsando el espacio. El videojuego cuenta con ciertas ayuda para superar la pantalla, los *power ups* que el jugador podrá conseguir son: **vida extra**, **inmunidad** y **disparo especial**. Este último podrá utilizarse pulsando la tecla "A". Con todas estas armas y ayudas, el jugador debe ir matando a los marcianos e ir **puntuando** puntos hasta eliminarlos a todos y pasar al **siguiente nivel** hasta que finalice la partida.

A parte del *sprite* de la nave, los marcianos tendrán un movimiento horizontal en bloque e irán aumentando su velocidad según la cantidad de marcianos que quedan vivos

[1]Programa para realizar cualquier tipo de diagrama. https://cloud.smartdraw.com

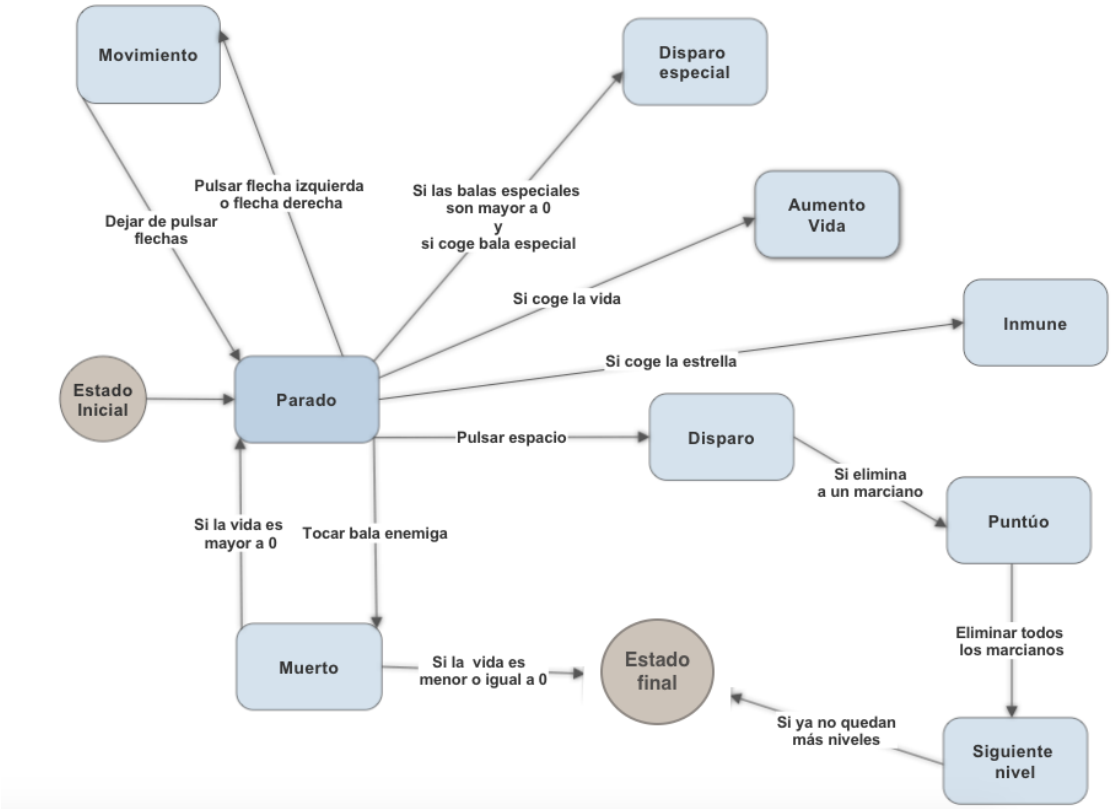

Figura 4.1: Diagrama de estados para la nave de Space Invaders.

en la pantalla. Además, se ha realizado una animación para hacer más real el movimiento horizontal. Todos los marcianos moverán sus brazos de arriba a abajo sincronizados.

4.2 Modificaciones del videojuego real

Space Invaders fue creado en los años 70. Durante esta época los videojuegos estaban disponibles en máquinas recreativas situadas en locales sociales. Estas máquinas recreativas funcionaban con monedas. Al insertar una moneda en la máquina de Space Invaders lo primero que nos aparece es un menú principal, donde nos indican la puntuación de cada marciano.

Una vez comenzado el juego, en la parte superior de la pantalla se podrá observar la puntuación obtenida hasta el momento y las vidas que le quedan al jugador. Seguidamente, en el centro de la pantalla se encontrarán las cuatro filas de marcianos que se moverán de izquierda a derecha. Cada cierto tiempo, aparecía un platillo volante por encima de los marcianos; si el jugador lograba acertarle, le sumaba una puntuación aleatoria. En la parte inferior estará la nave que controlará el jugador. Esta nave podrá disparar para destruir a todos los marcianos. Además, en el videojuego original, entre los marcianos y la nave, tenía cuatro búnkeres para protegerse de las balas de los marcianos. En la figura 2.3 podemos observar la pantalla del videojuego original.

En este proyecto, se han realizado varias modificaciones respecto al original. En la figura 4.2 se puede observar parte de los cambios realizados. El Space Invaders realizado tiene un fondo estrellado con movimiento vertical. Se ha escogido la imagen de un solo marciano para representar a los enemigos. Existen tres tipos de alienígenas cuyo color nos identifica su número de vidas: verde (1 vida), rojo (2 vidas) y azul (3 vidas). El enemigo de color verde será el único que sume puntos al jugador (5 puntos por cada marciano).

Además, se han añadido tres *power ups*, que soltará el platillo volante, para ayudar al jugador eliminar a los marcianos. Vida extra: con un *sprite* de un corazón, rayo láser: con una imagen de un rayo y la inmunidad: con el *sprite* de una estrella.

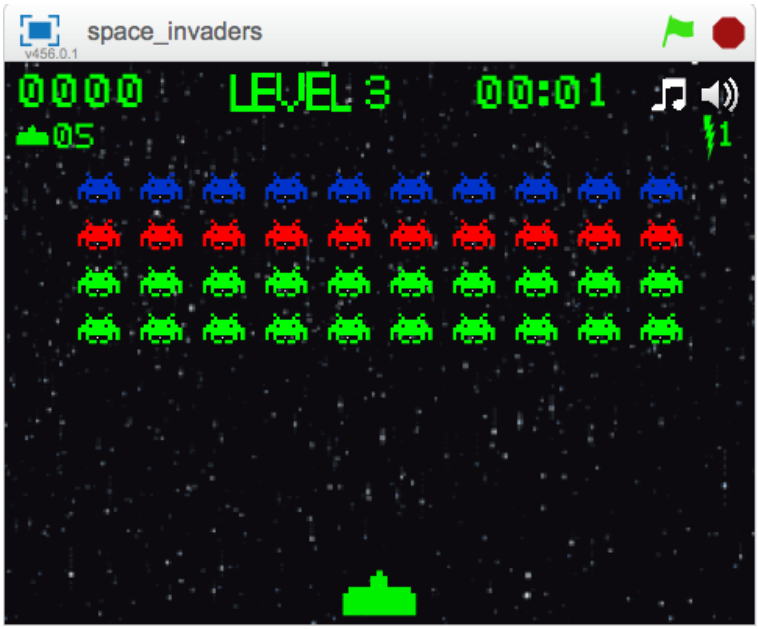

Figura 4.2: Captura de pantalla del nivel 3 del videojuego Space Invaders realizado.

Se han diseñado 6 niveles distintos, cada nivel aumentará de dificultad añadiendo una fila de marcianos más fuerte. El marciano con menor valor es el verde, al darle con una bala desaparecerá, sin embargo, los demás irán cambiando al color del marciano anterior a él, hasta llegar al color verde y así, desaparecer.

Por último, se ha realizado un menú principal para el videojuego. En la figura 4.3 se puede observar su diseño. Este menú posee tres opciones: jugar, instrucciones y créditos. Con la primera opción se iniciará el juego. Con la segunda opción encontraremos las instrucciones del videojuego y con la última opción, los créditos del videojuego donde se muestra el nombre de la creadora del videojuego y su tutor de la universidad.

Figura 4.3: Captura de pantalla del menú principal realizado para el videojuego Space Invaders.

4.3 Implementación del videojuego

La forma de trabajar a la hora de implementar cualquier videojuego escogido para este proyecto es investigar sobre el videojuego y conocer bien sus objetivos y mecánicas, como se ha mencionado en el apartado 4.1.

En el caso de Space Invaders, se han encontrado 11 objetos principales: la nave, las balas del jugador, el rayo láser, los alienígenas, las tres diferentes balas de los enemigos, el platillo volante y los tres *power ups* diseñados. A parte de los objetos más característicos, se debe tener en cuenta que el escenario, el fondo donde se colocarán nuestros *sprites*, también tendrá *scripts* asociados, ya que el escenario será el objeto que controlará continuamente el estado del videojuego. A continuación se explicarán cada uno de los objetos mencionados.

1. **Nave:** La nave va a ser controlada por el jugador y su objetivo será destruir a todos los enemigos que se muestren en la pantalla. El jugador podrá controlar la nave con las flechas izquierda y derecha. Para poder destruir a los marcianos, el jugador deberá presionar el espacio para disparar las balas. Además, el jugador podrá obtener un rayo láser por completar cada nivel o al recoger el *power up* del rayo del platillo volante. También se ha implementado que el jugador escoge entre 6 colores para la nave pulsando los número del 1 al 6. En la figura 4.4 podemos observar los *scripts* realizados para recoger los *power ups*, y el daño que recibe la nave por cada una de las balas de los marcianos: verdes, rojas y azules, según el tipo de marciano.

   Las balas de los marcianos tienen un controlador para saber si están tocando o no la nave. Según el tipo de bala mandarán el mensaje de tocado (bale verde), tocado2 (bala roja) y tocado3 (bala azul). Los 3 *scripts* que podemos observar realizarán la misma función: quitarán de 1 a 3 vidas a la nave según el color de la bala y además, comprobarán si el jugador se ha quedado sin vidas para terminar o no el juego.

   En la figura 4.4 también podemos observar las funciones realizadas al conseguir los *power ups* de inmunidad y de vida extra. Con el *power up* de vida extra la nave tan solo se encargará de decir "¡VIDA EXTRA!" durante un segundo. Sin embargo, el *power up* de inmunidad hará que la nave cambie de color durante unos segundos y que no afecten a la nave las balas de los enemigos cambiando la variable invencible a 1.

   Entre todas sus funcionalidades, dentro de este *sprite* se debe ir controlando la puntuación del jugador para poder pasar al siguiente nivel. Por como se ha realizado la puntuación de los marcianos, es tan fácil como comprobar que esta ecuación se verifique: Puntuación = Nivel $\times$ 200. El número 200 es la puntuación máxima que podemos conseguir en cada uno de los niveles: 40 marcianos verdes $\times$ 5 puntos de cada marciano.

2. **Bala de la Nave:** Las balas que disparará la nave aparecerán cuando el jugador pulse el espacio del teclado. Se ha implementado una función que controla el movimiento de la bala. Este movimiento se ha definido como vertical y hacia arriba. Además, conforme va subiendo la bala, se debe ir comprobando si se ha chocado con alguno de los enemigos. Estos controladores tendrán la condición de si toca o no, los colores verde, rojo y azul. En la figura 4.5 se puede observar una captura de las funciones realizadas para este objeto.

3. **Rayo Láser:** El rayo láser es una ayuda para poder eliminar a los enemigos más rápido. El jugador deberá pulsar la tecla "A" para poder dispararlo. En la figura 4.6 se puede observar la interfaz del láser. Al pasar al siguiente nivel, el jugador recibirá

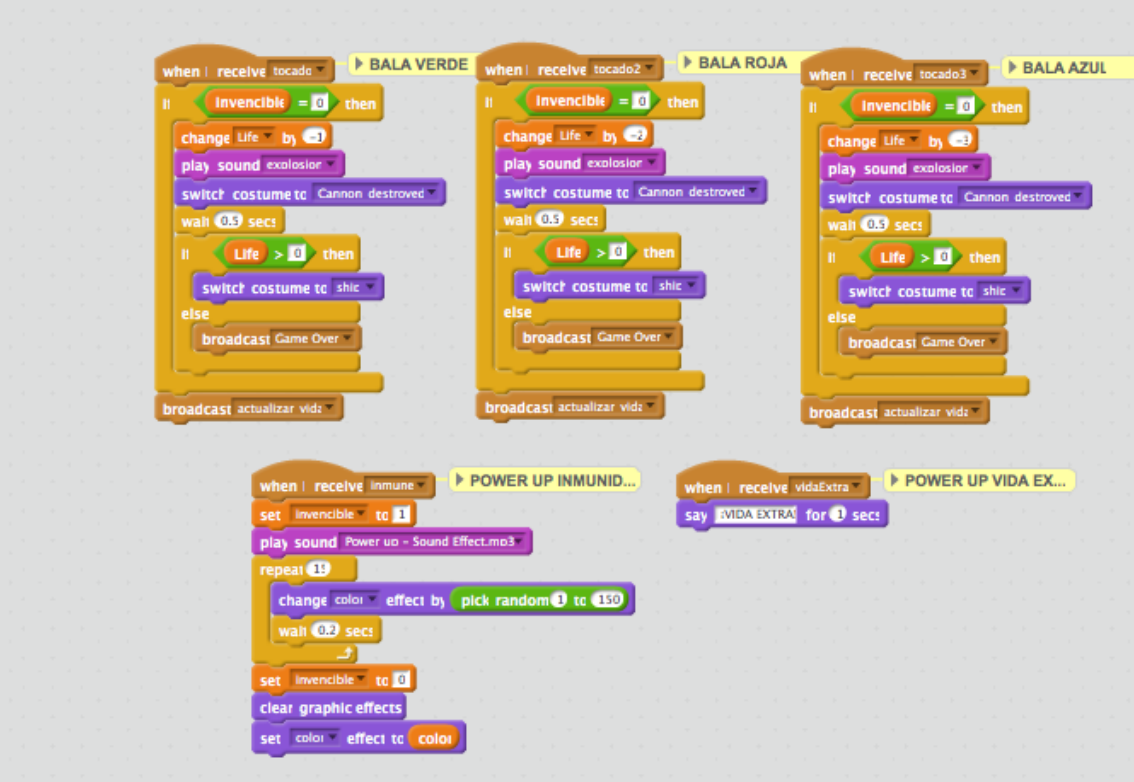

Figura 4.4: *Scripts* para los *power ups* y balas enemigas del objeto nave de Space Invaders.

un rayo láser extra. Otra manera de conseguir un rayo láser adicional es alcanzando el *power up* del rayo que lanza el platillo volante.

El rayo láser se diferencia de las balas básicas de la nave en el movimiento vertical y en la animación. Mientras que las balas se mueven hacia arriba con un movimiento vertical, el rayo láser ocupa toda la altura de la pantalla, realizando una animación de expansión hacia los lados. En la figura 4.7 se pueden observar los *scripts* implementados para la inicialización y animación de este *power up*.

4. **Enemigos:** Es el objeto más difícil de implementar del juego. Se debe tener en cuenta la formación de los enemigos, su movimiento, su velocidad, la animación de movimiento y la probabilidad de disparar. Como se han creado distintos niveles para este videojuego, se ha implementado un *script* que conforme el jugador vaya aumentando de nivel, se vayan inicializando marcianos con mayor vida.

   Este método consiste en declarar la vida de cada marciano según el nivel actual: si el nivel es 2 aparecerán marcianos de color y si es superior a 2, de color azul. En la figura 4.8 se puede observar la implementación en Scratch. Por cada fila se declarará la vida del marciano y posteriormente se crearán tantos clones como queramos de éste, en este caso 10.

   Otra funcionalidad implementada es que los marcianos se puedan mover de forma horizontal. Cada vez que se choquen con el borde, bajarán un escalón hasta chocar con la nave. En la figura 4.9 se puede observar el modo de implementación de esta funcionalidad. Junto con el movimiento de los marcianos, se ha realizado que los enemigos vayan aumentado su velocidad de movimiento. Esta fórmula se basa en el nivel y en la puntuación actual del jugador: Por cada nivel superado, los marcianos irán aumentando su velocidad un 5 % y también, dentro de cada nivel, aumentará un 60 % la velocidad según el número de marcianos que quedan vivos.

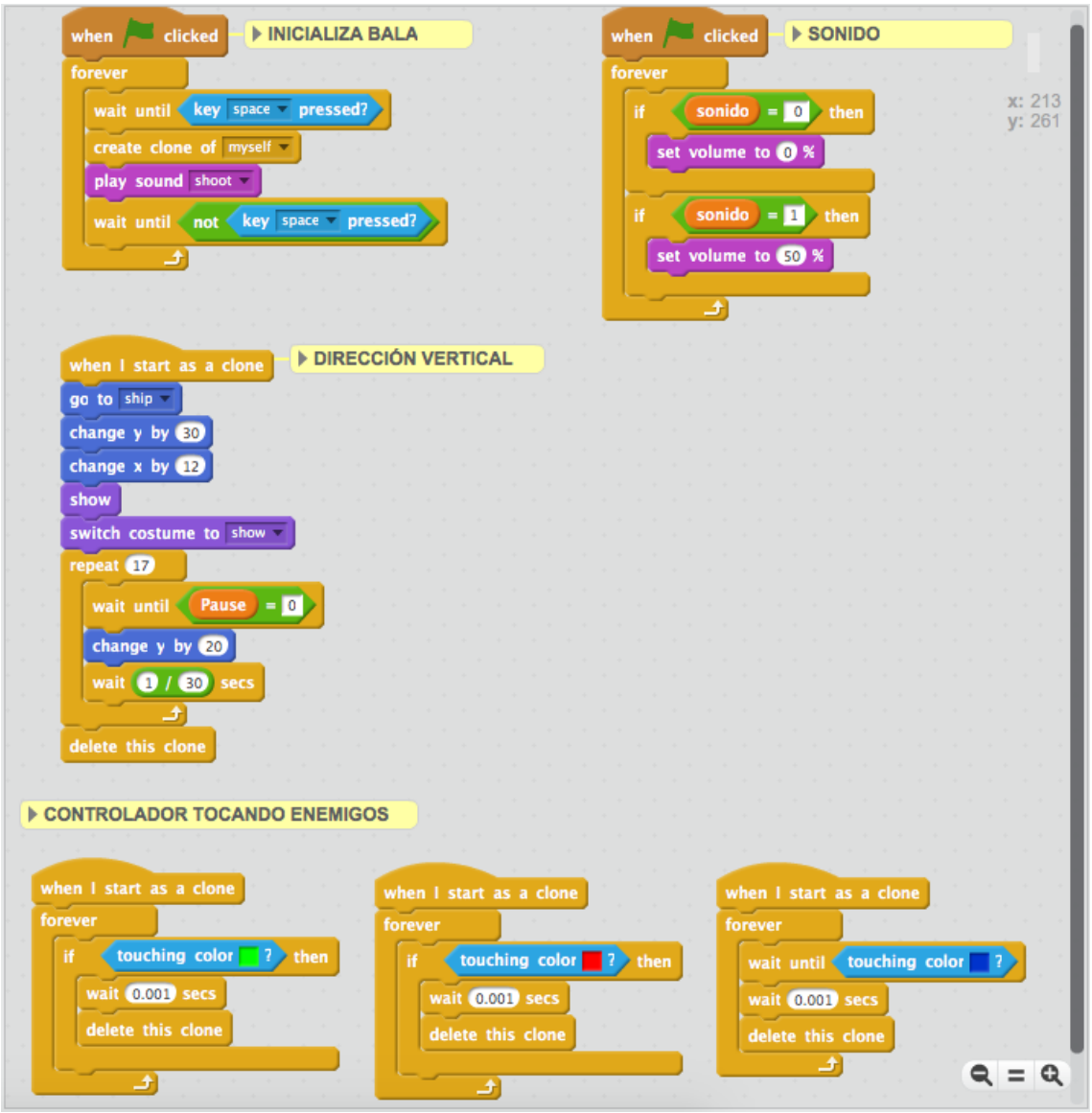

Figura 4.5: Implementación de los *scripts* para las balas de la nave de Space Invaders.

Para dar más sensación de movimiento, se diseñaron distintos disfraces para el *sprite*. Esta animación se basa en mover los brazos hacia arriba y hacia abajo. A parte de implementar una funcionalidad para el cambio de disfraces, se ha tenido que implementar la velocidad de la animación y así, estar acorde con su movimiento horizontal.

5. **Balas Enemigos:** El encargado de inicializar cada tipo de bala es el objeto del marciano ya que, según el tipo del enemigo, creará un tipo de bala, que hará más o menos daño a la nave: bala verde (1 vida), bala roja (2 vidas) y bala azul (3 vidas). En la figura 4.10 se puede observar el método empleado. La probabilidad de que aparezcan estas balas es de un 20 %.

   Su movimiento es vertical como las balas de la nave, pero se mueven en sentido contrario, es decir, hacia abajo. En la figura 4.11 se puede observar la implementación del movimiento. Además, tienen cierta particularidad, por como se ha implementado la creación de los distintos marcianos y la de sus respectivas balas, puede que ocurra un solapamiento entre ellas. Para ello, se ha implementado una función en

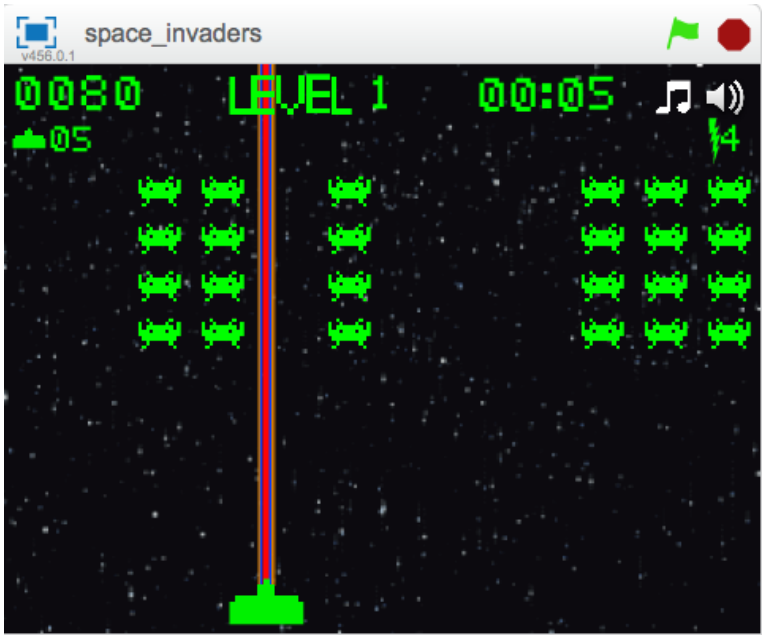

Figura 4.6: Animación gráfica diseñada para el rayo láser de Space Invaders.

la que si se están tocando dos balas, la bala de mayor coste, es decir, la roja o la azul, según el caso, será la que se visualizará en el juego.

6. **Platillo volante:** Este objeto aparece de vez en cuando en el juego, la mecánica elegida es similar a las balas de los enemigos. Tendrá un 20 % de probabilidad de crearse una instancia del platillo. Una vez creada la instancia, el platillo tendrá un movimiento horizontal hacia la izquierda. Además, el platillo volante será el encargado de crear las instancias de los distintos *power ups*. El método de crear los *power ups* tiene un 45 % de probabilidad de caer cada uno de los *power ups*. En la figura 4.12 se muestra el *script* realizado.

7. ***Power ups*:** Los tres *power ups*, tendrán una funcionilidad muy similar entre ellos. En la figura 4.13 se muestra la implementación realizada para el *power up* del rayo láser: su movimiento vertical hacia abajo.

   La única diferencia entre ellos es la característica que beneficiará a la nave del jugador:

   - La inmunidad hará al jugador ser invencible durante 3 segundos. Este *power up* es característico por cambiar de color constantemente a la nave.
   - La vida extra, como bien dice su nombre, añadirá una vida más al jugador.
   - El rayo láser es igual que el de la vida extra, salvo que añadirá un rayo láser extra al jugador.

8. **Escenario:** Como se ha mencionado en puntos anteriores, el escenario será el que gestione el estado del videojuego. En el caso de Space Invaders, inicializará todas las variables como la vida, la puntuación, el nivel actual,... Pero no solo se le ha asignado esta función; como se puede observar en la figura 4.14, es el encargado de la animación del fondo del juego, el controlador de cambiar el tiempo jugado, saber si el jugador ha pausado el juego con la tecla "P" o no, y además, conocer si se ha ganado o perdido la partida.

Como se ha visto en otras imágenes (figura 4.2), se han implementado ciertas animaciones para mostrar en la pantalla de juego: la puntuación obtenida, el nivel actual del

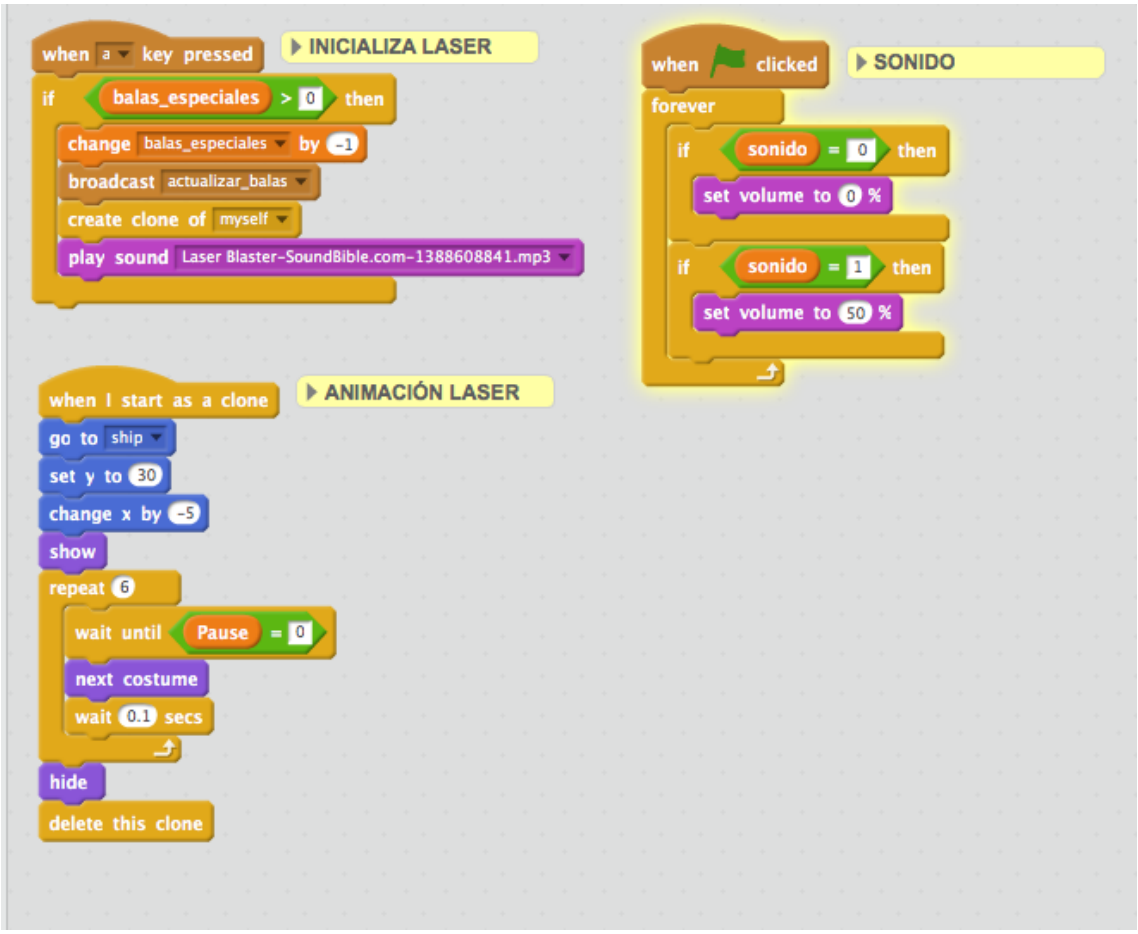

Figura 4.7: *Scripts* realizados para el rayo láser de Space Invaders.

jugador, los minutos que lleva jugando, así como también, las vidas y los rayos láser que le quedan por el momento al jugador.

Además de todo lo descrito anteriormente, se puede observar en muchos *sprites* la misma funcionalidad para el sonido. Se han añadido dos iconos para encender o apagar la música o los efectos especiales de cada objeto. Por último, se han añadido dos imágenes para mostrar la victoria o la derrota del juego.

Figura 4.8: *Script* de inicialización de los enemigos de Space Invaders.

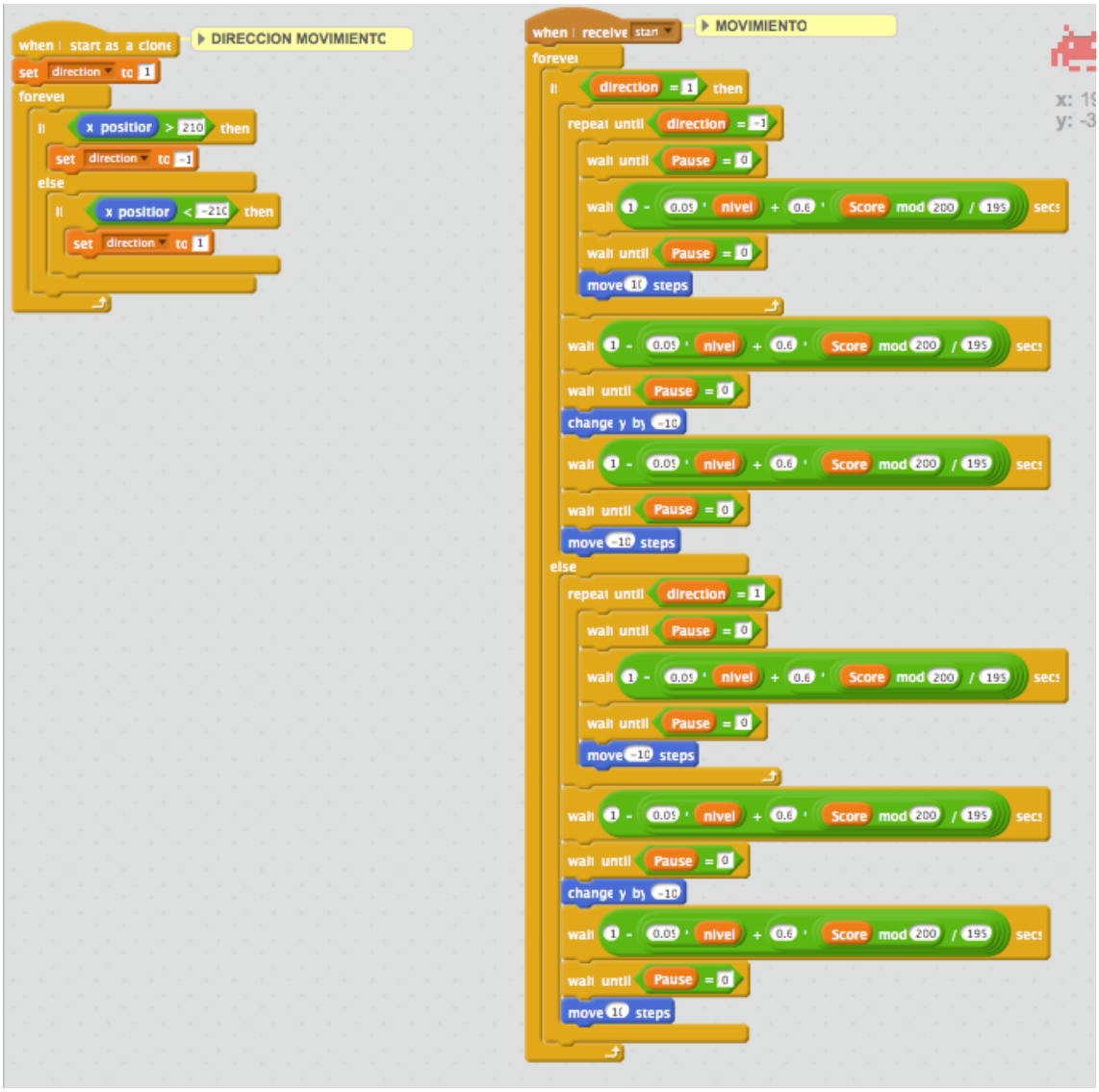

Figura 4.9: *Scripts* del movimiento horizontal de los enemigos de Space Invaders.

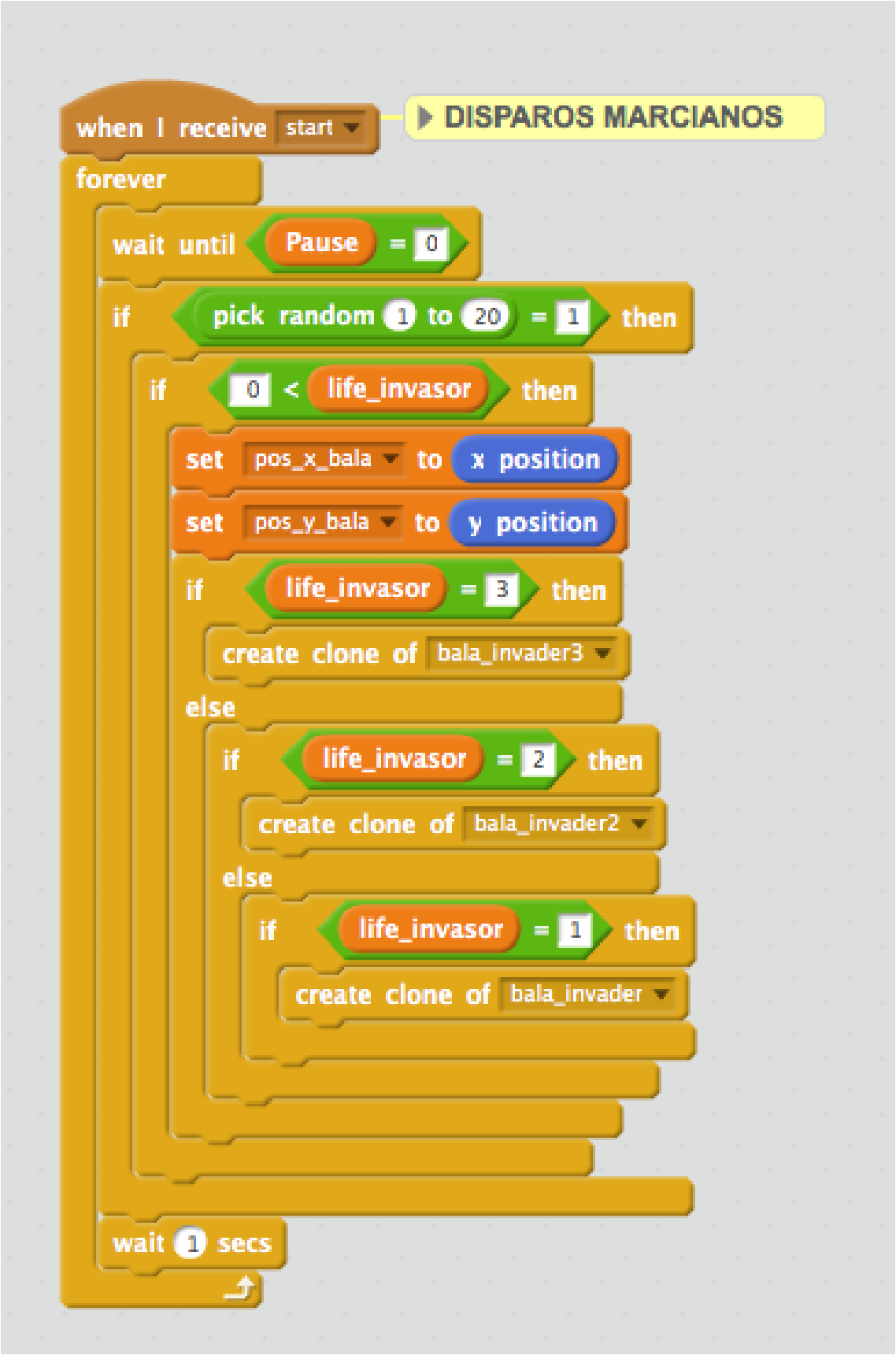

Figura 4.10: *Script* de inicialización de las balas de los enemigos de Space Invaders.

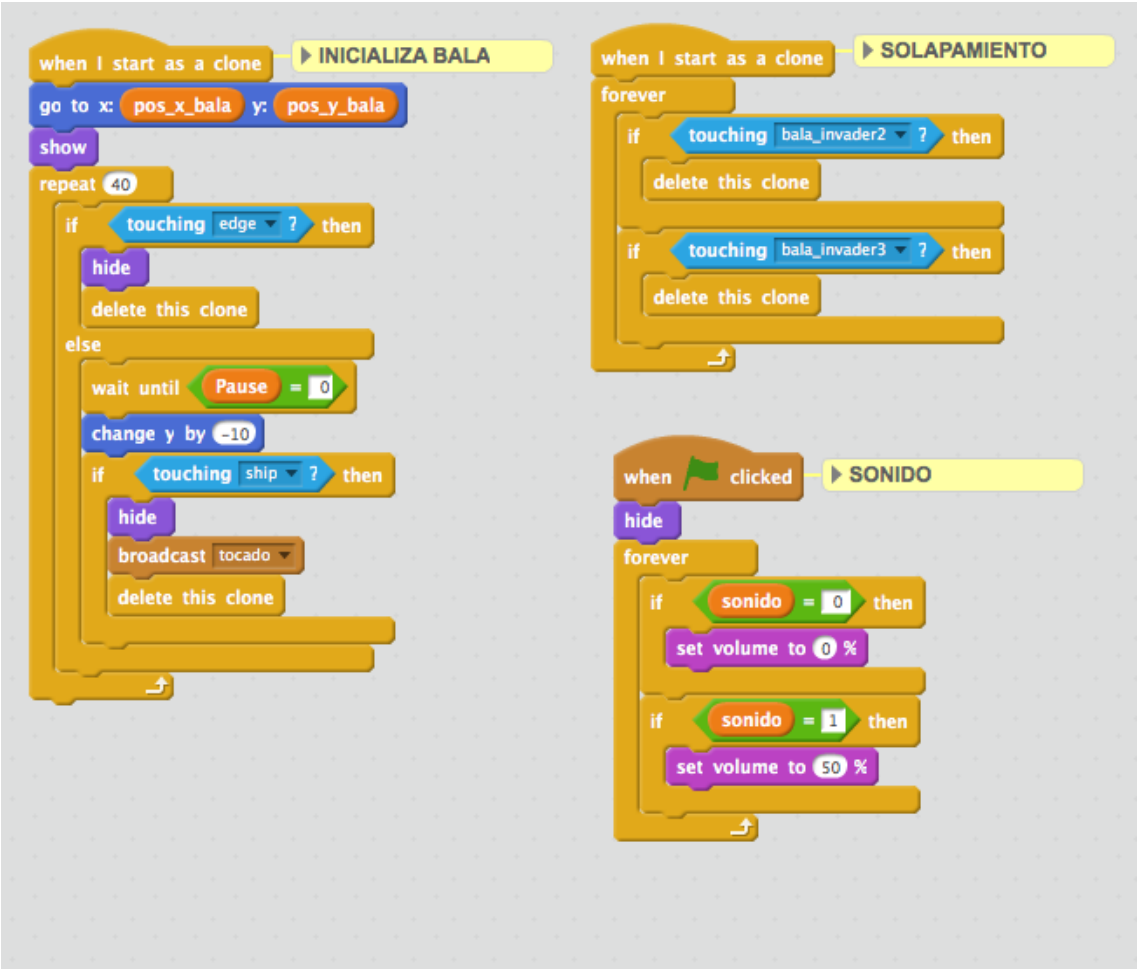

Figura 4.11: *Scripts* realizados para cada una de las balas de los marcianos de Space Invaders.

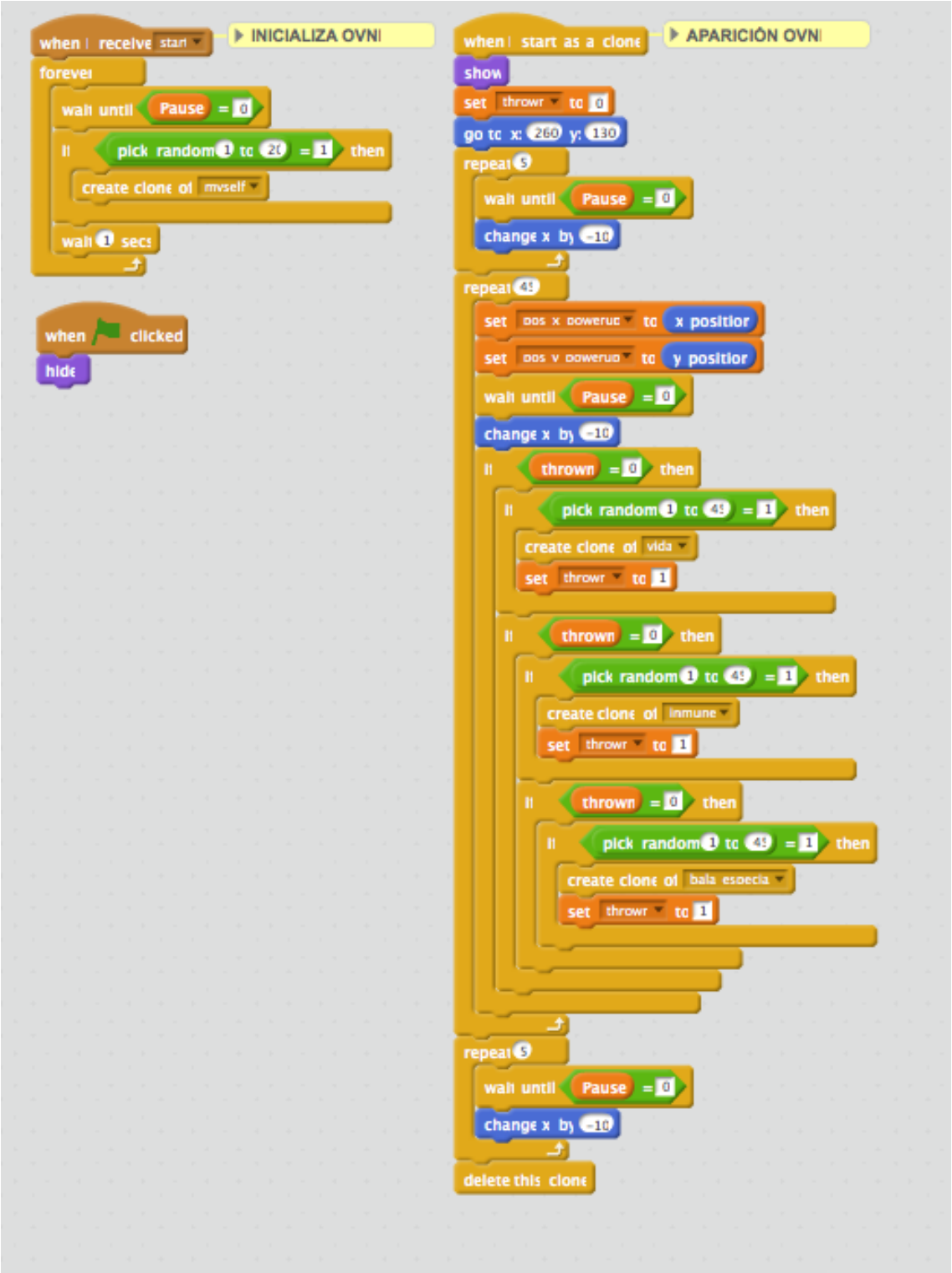

Figura 4.12: Creación de los *power ups* en el objeto platillo volante de Space Invaders.

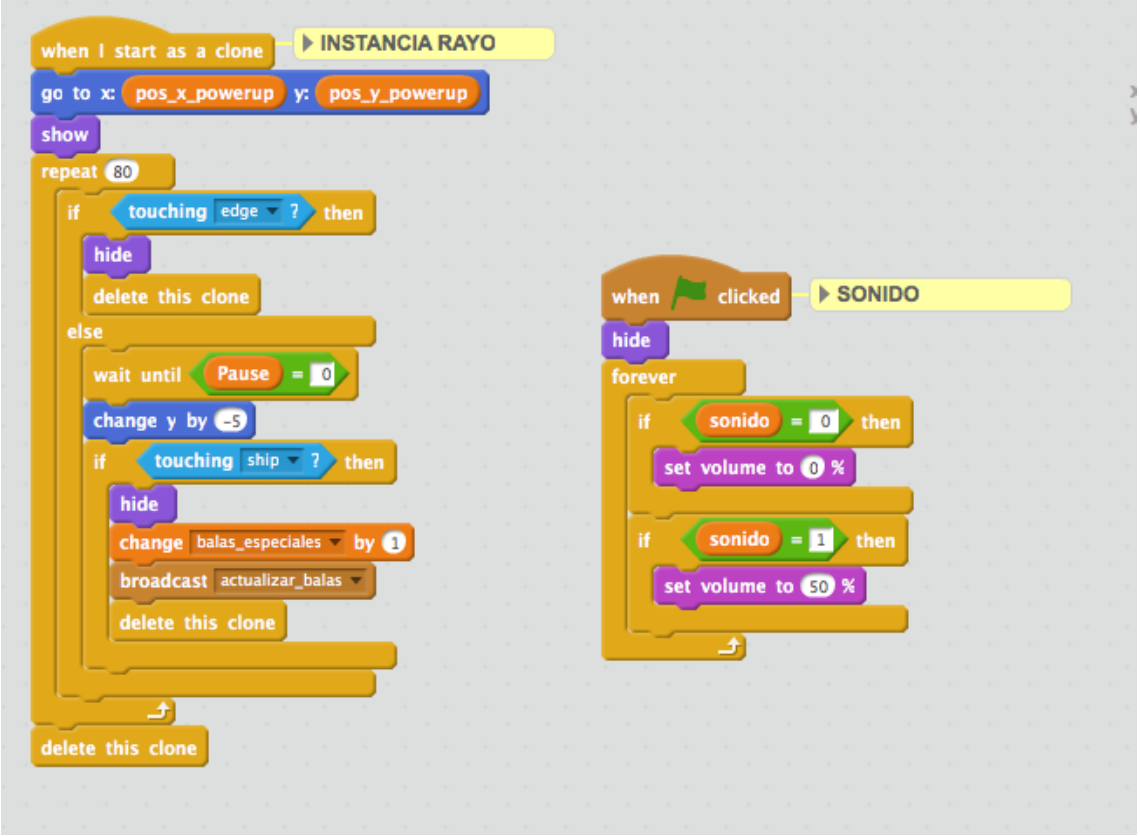

Figura 4.13: Ejemplo de la implementación del *power up* del rayo láser de Space Invaders.

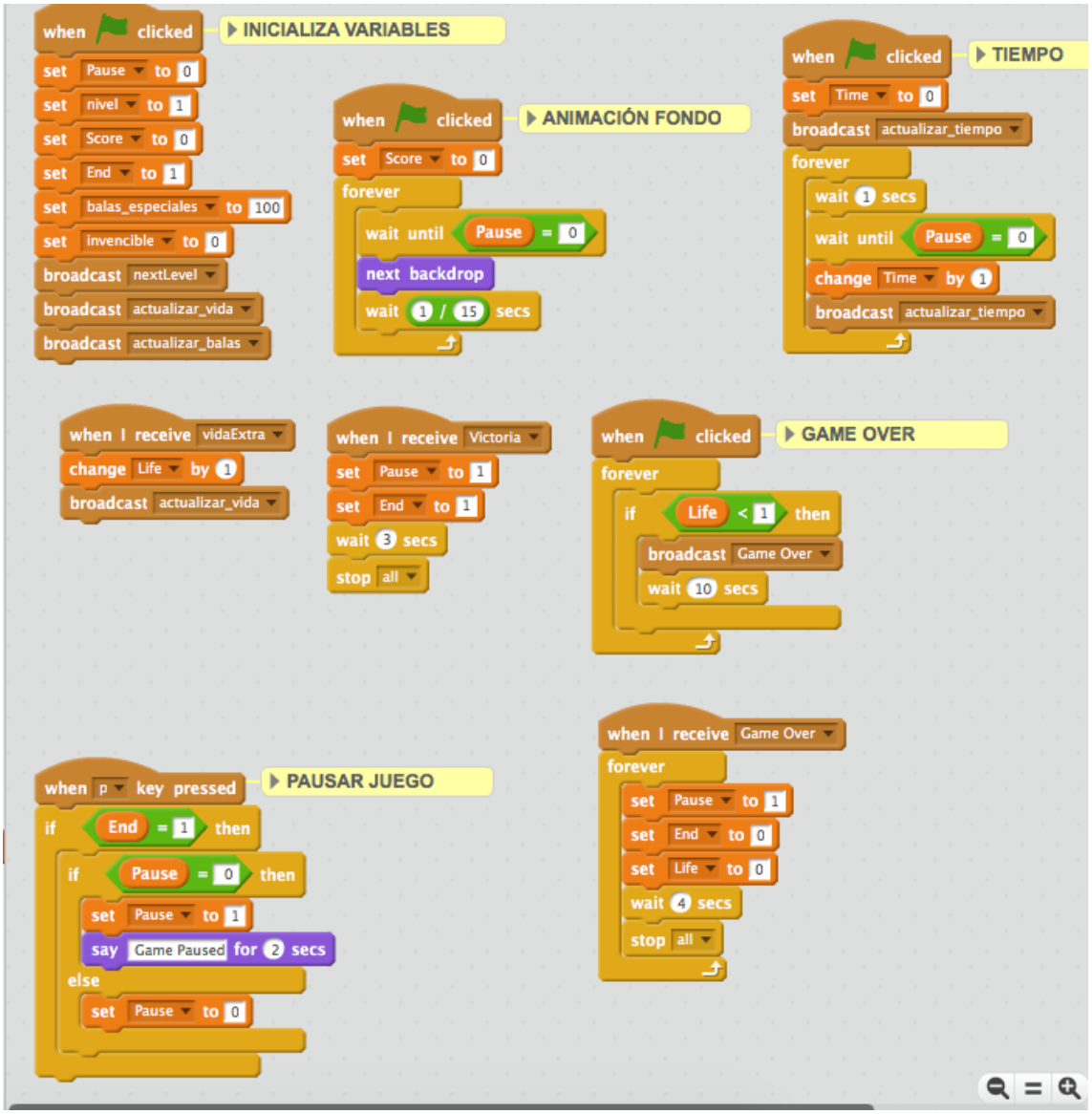

Figura 4.14: *Scripts* del escenario de Space Invaders.

CAPÍTULO 5
Breakout

El videojuego consiste en destruir todos los ladrillos con una pelota que rebota por la pantalla. En este apartado se explica como se ha diseñado este videojuego tan popular en su época con herramientas actuales como Scratch.

5.1 Diseño del videojuego

En este videojuego se han seguido las mismas pautas explicadas en el apartado 4.1. Al igual que en el videojuego Space Invaders se han buscado imitaciones del Breakout original. Entre todas las copias encontradas, se ha escogido la imitación de Google. Esta versión se ha encontrado buscando "atari breakout" en Google Imágenes.

Una vez probado el videojuego, se ha construido la estructura a seguir en Scratch. En la figura 5.1 podemos observar la mayoría de los objetos utilizados. Los objetos más característicos del videojuego son la paleta, la pelota y los ladrillos.

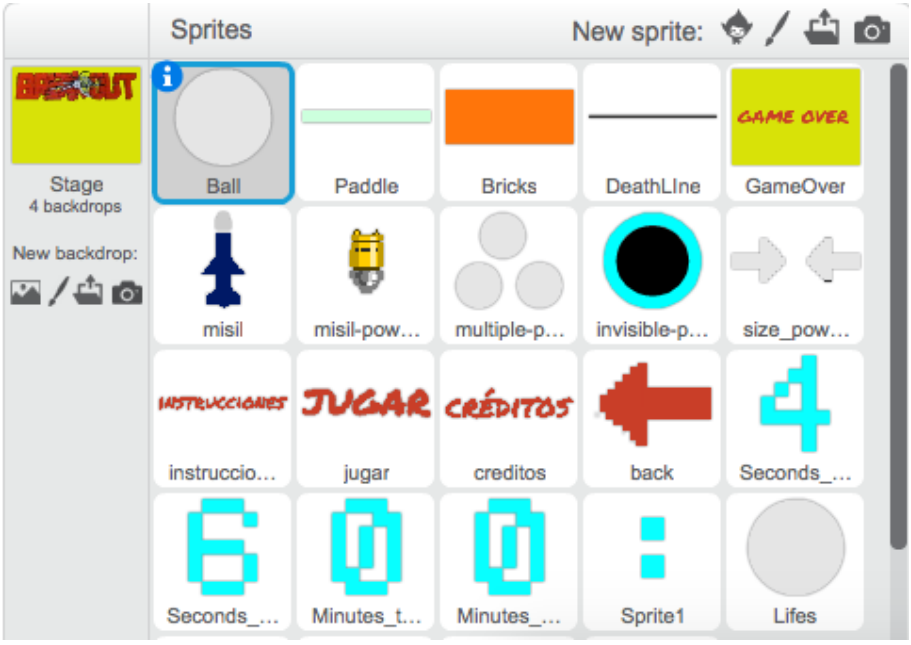

Figura 5.1: Objetos creados para el videojuego Breakout.

Para poder entender la mecánica utilizada para este videojuego se ha realizado un diagrama de estados (figura 5.2) para el *sprite* de la paleta, el cual manejará el jugador.

Para la realización del diagrama de estados se ha utilizado una herramienta de diseño gratuita: www.draw.io.

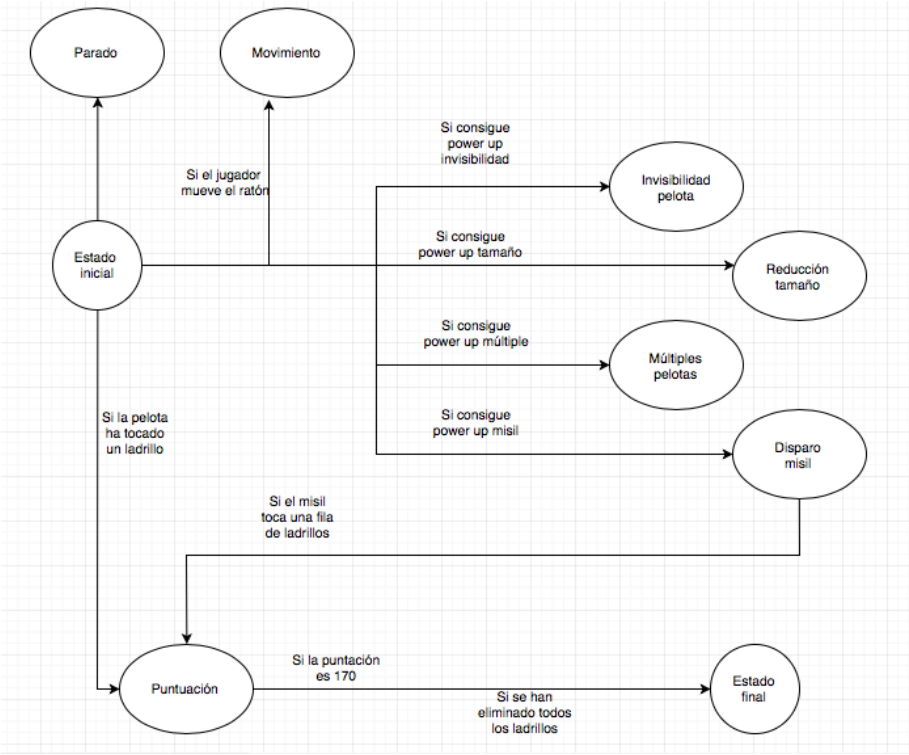

Figura 5.2: Diagrama de estados para el objeto Paleta de Breakout.

Como se puede observar en la figura 5.2, existen 7 estados de los cuales tres afectan al objeto de la pelota. Al iniciarse el juego, el jugador encontrará la paleta **parada**. En cuanto el jugador mueva el ratón, la pelota se **moverá** de izquierda a derecha. Para poder **puntuar**, el jugador hará rebotar la pelota contra los ladrillos que aparezcan en la parte superior de la pantalla. Al ser eliminado, la puntuación aumentará, y además, existirá la posibilidad de que caigan uno de los 4 *power ups* que se han implementado: **misil, múltiples pelotas, reducción del tamaño de la paleta e invisibilidad de la pelota**.

Desde el objeto de la pelota se irá controlando si toca o no a los ladrillos y así conseguir la puntación especificada. Además, dos de los *power ups* que se han implementado para el videojuego afectarán a la pelota: la aparición de múltiples pelotas y la invisibilidad.

5.2 Modificaciones al videojuego original

El videojuego original, creado en 1975, consta de una sola pantalla principal. En la parte superior de la pantalla podemos ver la puntuación conseguida actual del jugador. Más abajo se sitúan los ladrillos que el jugador intentará destruir. Como se puede apreciar en la figura 2.5, se forman 8 filas de 14 ladrillos cada una. Como se ha explicado en el apartado 2.2, cada ladrillo coloreado tiene una puntuación distinta. Los ladrillos más cercanos a la paleta tendrán una puntuación menor respecto a los más alejados. Por último, en la pantalla se muestra una pelota de forma cuadrada. Esta figura destruirá los ladrillos rebotando por los bordes de la pantalla y la paleta que maneja el jugador.

Para este trabajo se ha intentado realizar una copia bastante similar al videojuego original. Se ha seguido la misma puntación y coloreado para los ladrillos. Sin embargo, por falta de espacio en la pantalla, se han realizado 5 filas de ladrillos en vez de 8. En la

figura 5.3 se puede observar la pantalla de juego desarrollada. En cuanto a la pelota y a la paleta, se han producido cambios en sus apariencias: la pelota tendrá forma circular y la paleta será más alargada.

Figura 5.3: Pantalla de juego del videojuego Breakout realizado.

Para hacer el videojuego más atractivo, se han creado 4 *power ups* de los cuales dos ayudarán al jugador y los otros dos aumentarán la dificultad de la partida. Los *power ups* creados para el videojuego son: un misil para destruir toda una columna de ladrillos, la aparición de múltiples pelotas en la pantalla, la reducción de tamaño de la paleta y la invisibilidad de la pelota. Todos estos *power ups* tendrán una durabilidad de 5 segundos excepto el misil que no será acumulable.

Se han mostrado en la pantalla tres indicadores: la puntuación actual, el tiempo jugado y las vidas que le quedan al jugador. Estos indicadores se distribuyen por la parte superior e inferior de la pantalla como se puede observar en la figura 5.3

Por último, se ha realizado un menú principal similar al videojuego de Space Invaders. En la figura 5.4 podemos observar su diseño final. Este menú consta de las mismas opciones que el menú explicado en el capítulo anterior.

5.3 Implementación del videojuego

La mecánica seguida para empezar a implementar el videojuego Breakout se ha explicado en apartados anteriores. Se han encontrado versiones similares al videojuego original y se han encontrado las mecánicas más características del videojuego.

Para el videojuego implementado para este trabajo se han encontrado 8 *sprites* principales: los ladrillos, la pelota, la pelota y los 4 *power ups*. A parte de estos objetos debemos tener en cuenta que el escenario, al igual que los demás videojuegos, será el que controlará el inicio y final del juego además de otras acciones.

A continuación se explicarán cada uno de los objetos mencionados anteriormente:

Figura 5.4: Menú realizado para el videojuego Breakout realizado.

1. **Paleta:** La paleta será controlada por el jugador, con ella hará rebotar la pelota para destruir todos los ladrillos. Como se puede observar en la figura 5.5, la paleta será controlada por la posición horizontal (eje X) del ratón.

   En este *sprite* se han tenido que implementar ciertos controladores para los *power ups*. Una vez que el jugador consigue el *power up* del misil o el de reducción de tamaño, la apariencia de la paleta cambiará a su correspondiente disfraz. Además, este objeto será el encargado de restar una vida al jugador cuando pierde la pelota de la pantalla.

2. **Pelota:** La pelota ha sido el *sprite* más difícil de implementar. Este objeto será el encargado de rebotar por toda la pantalla hasta destruir todos los ladrillos. Se han construido dos nuevos bloques, para la posición inicial de la pelota y su movimiento de rebote. Estas dos funciones se utilizan dentro de otros *scripts* realizados para la pelota.

   El método más difícil de implementar ha sido el movimiento de la pelota. Hacer que rebotara de forma natural con la paleta o con los ladrillos no ha sido fácil. La fórmula utilizada ha sido encontrada en uno de los videojuegos probados para conocer la dinámica del juego. Esta fórmula realiza un cambio de sentido al tocar con alguno de los ladrillos. Sin embargo, al chocar con la paleta, la pelota cambiará de sentido y además se le añadirá a su dirección la diferencia de la posición X ente la pelota y de la paleta.

   En la figura 5.6 se pueden observar los métodos explicados anteriormente junto con el método que controla la pérdida de una vida del jugador. Si la pelota toca el *sprite* llamado *Death Line*, línea de muerte, y además solo existe una sola pelota, el método enviará un mensaje "*substractLife*" y así el jugador perderá una vida hasta acabar con 0.

   Además, se han implementado dos funcionalidades para los *power ups* de la pelota. En la figura 5.7 se puede observar los *scripts* correspondientes a este objeto.

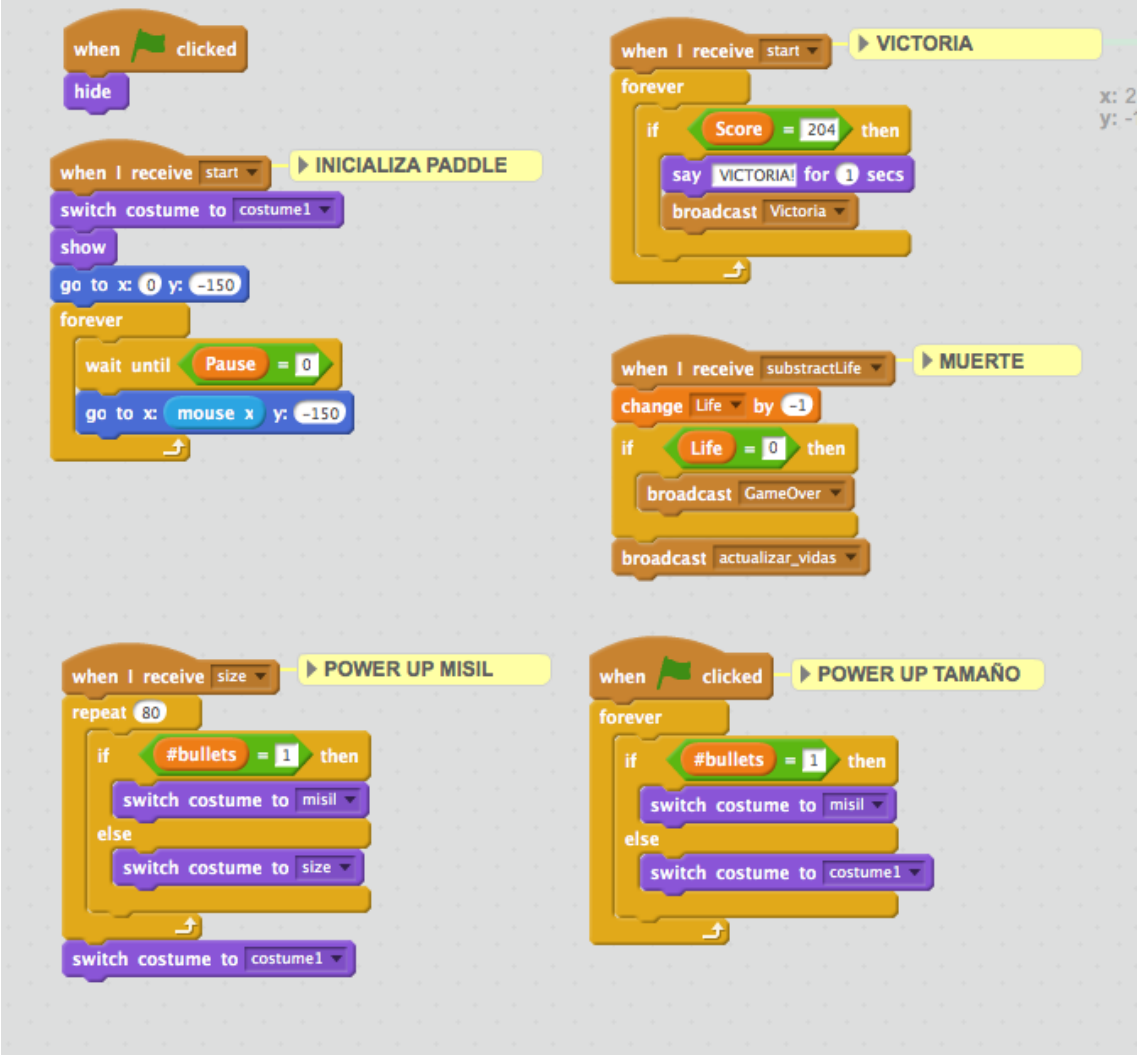

Figura 5.5: Métodos para el objeto Paleta de Breakout.

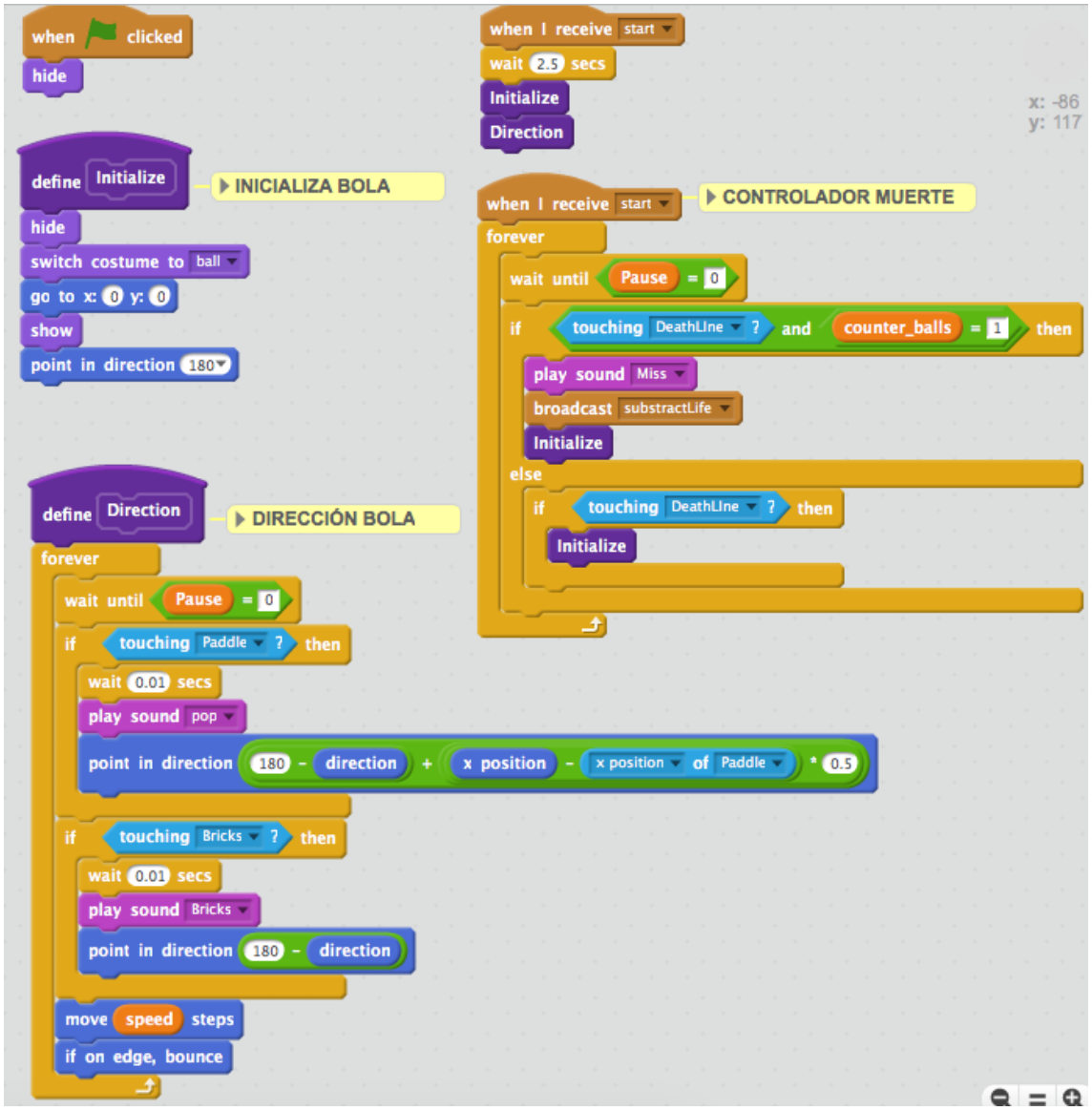

Figura 5.6: Inicialización y dirección de la pelota.

Para el *power up* de múltiples pelotas, se han generado 3 clones de la pelota y se ha utilizado el método de dirección para el movimiento de cada uno de los clones. Una vez pasados 5 segundos, los clones desaparecen continuando con una única pelota. Para el *power up* de invisibilidad se ha creado un método que cambiará la apariencia de la pelota durante 3 segundos.

3. **Ladrillos:** Para este objeto se ha reutilizado código de los marcianos del videojuego Space Invaders. Se ha creado un *script* muy similar para formar 5 filas de 12 ladrillos cada fila. En la figura 5.8 se puede observar la implementación del código.

   Para la inicialización de los ladrillos se han utilizado dos métodos: uno para la posición de cada ladrillo en la fila y otro para su apariencia; según el número de fila su apariencia cambiará. Los ladrillos más cercanos a la paleta serán amarillos y los más alejados de color rojo, es decir, de menor a mayor puntuación.

   Como se ha querido añadir extras al videojuego, se ha implementado que al destruir un ladrillo pueda caer uno de los cuatro tipos de *power ups* creados. En la figura 5.9 se pueden observar los métodos.

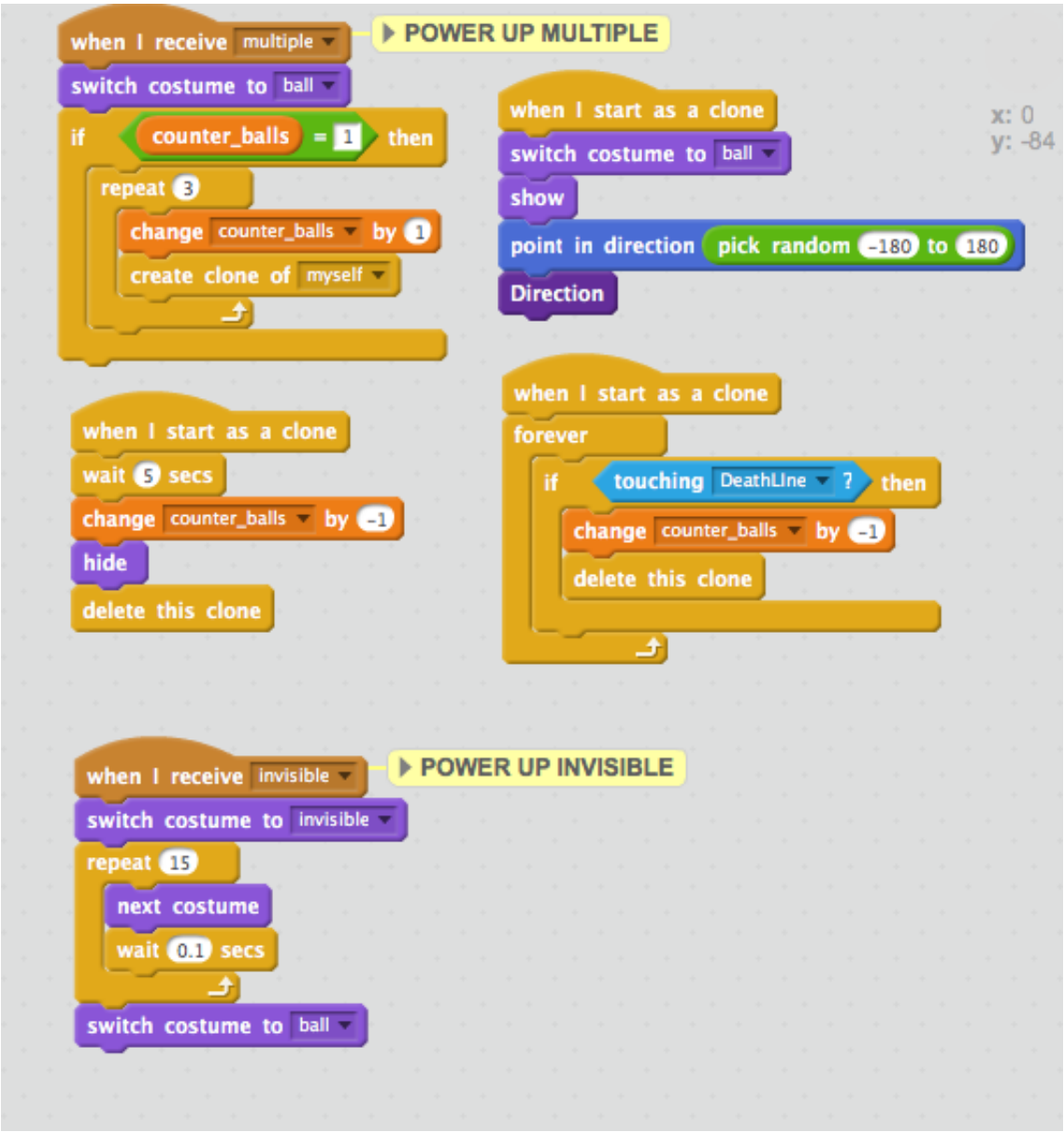

Figura 5.7: Métodos para los *power ups* correspondientes a la pelota.

Para la destrucción de los ladrillos se ha añadido un método que controlará si cada uno de los ladrillos toca o no la pelota. Una vez la pelota destruya un ladrillo, se sumarán los puntos correspondientes al jugador y llamará al método que se ha definido para la creación de uno de los *power ups* realizados. Estos objetos tienen un 25 % de probabilidad en caer de un ladrillo.

4. ***Power ups*:** Los *power ups* se han implementado como una funcionalidad extra para que sea más entretenido el videojuego. La creación de cada uno de estos objetos se ha definido en el *sprite* de los ladrillos explicado anteriormente.

   Se ha implementado un método similar al de los *power ups* del videojuego Space Invaders explicado en el apartado 4.3. Este método realiza la misma función que el realizado en Space Invaders. Una vez creado, el *power up* caerá del ladrillo destruido realizando un movimiento vertical hacia abajo. Si este objeto es tocado por la paleta mandará el mensaje correspondiente al *power up*. En el caso de la figura 5.10 mandará el mensaje del *power up* múltiple, creando múltiples pelotas.

   Se han creado 4 tipos distintos de *power ups*:

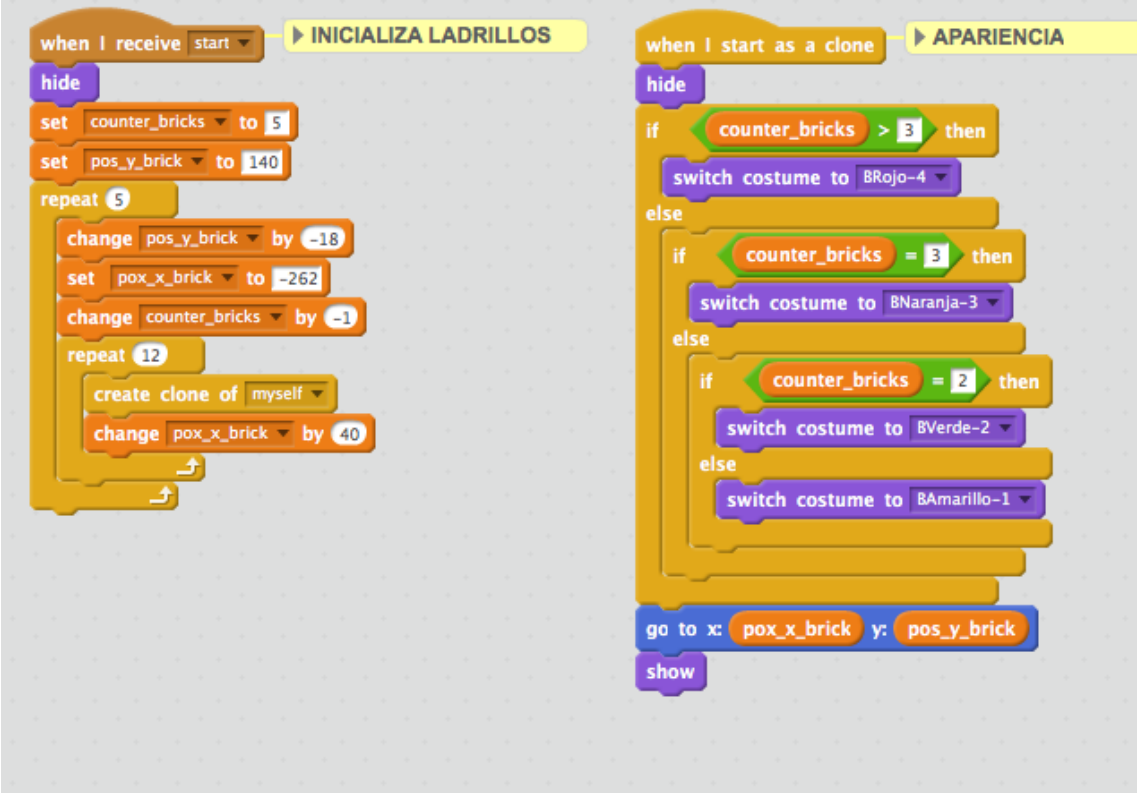

Figura 5.8: Inicialización y apariencia de los ladrillos.

- *Power up* misil: Este objeto otorgará al jugador un misil para destruir una columna entera de ladrillos. En la figura 5.11 se pueden observar los métodos para su movimiento vertical hacia arriba al pulsar el espacio.
- *Power up* reducción de tamaño: Si el jugador logra alcanzar este *sprite*, la paleta se hará más pequeña y así dificultar el rebote de la pelota.
- *Power up* múltiples pelotas: Como bien dice su nombre, este *power up* creará 3 pelotas más para destruir los ladrillos más rápidos.
- *Power up* invisibilidad: Al alcanzar este objeto, la pelota cambiará su apariencia a invisible y así dificultará mantener la pelota rebotando por la pantalla.

5. **Escenario:** Como en todos lo videojuegos, debe existir un objeto que controlará la inicialización tanto del juego como de las variables globales, la finalización del juego, ... El encargado de gestionar el estado del videojuego será el escenario. En la figura 5.12 se pueden observar los métodos implementados. Al igual que en Space Invaders se ha implemtado la opción de pausar el videojuego pulsando la tecla "P".

Por último, se puede observar en la figura 5.3 que se han implementado ciertas animaciones para ayudar al jugador a conocer su estado actual en el videojuego. En la parte superior de la pantalla está situada la puntuación actual y el tiempo que lleva jugando el jugador. Además, en la parte inferior izquierda de la pantalla se observa el número de vidas en forma de pelota.

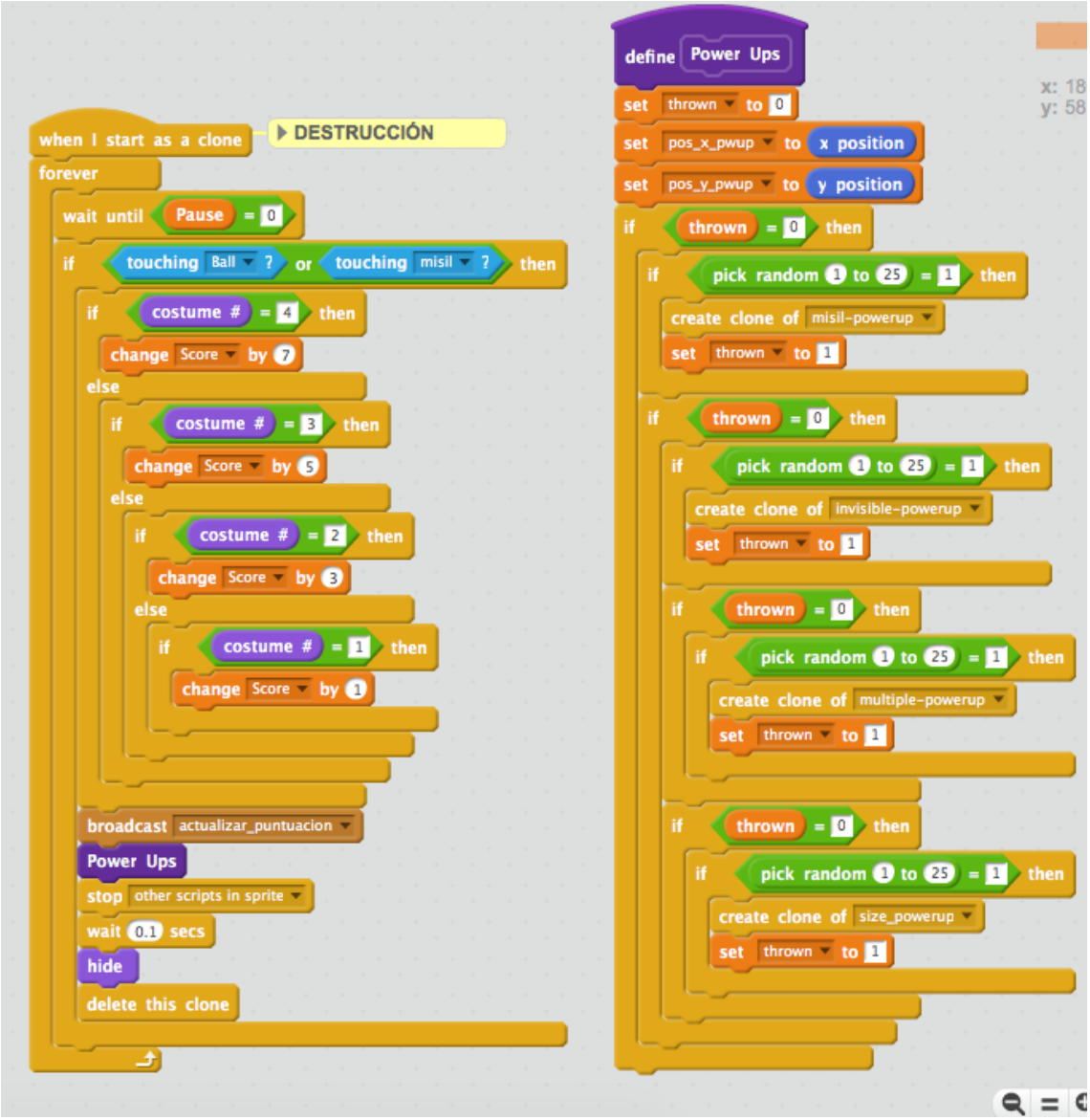

Figura 5.9: Métodos de la destrucción y creación de los *power ups* de los ladrillos.

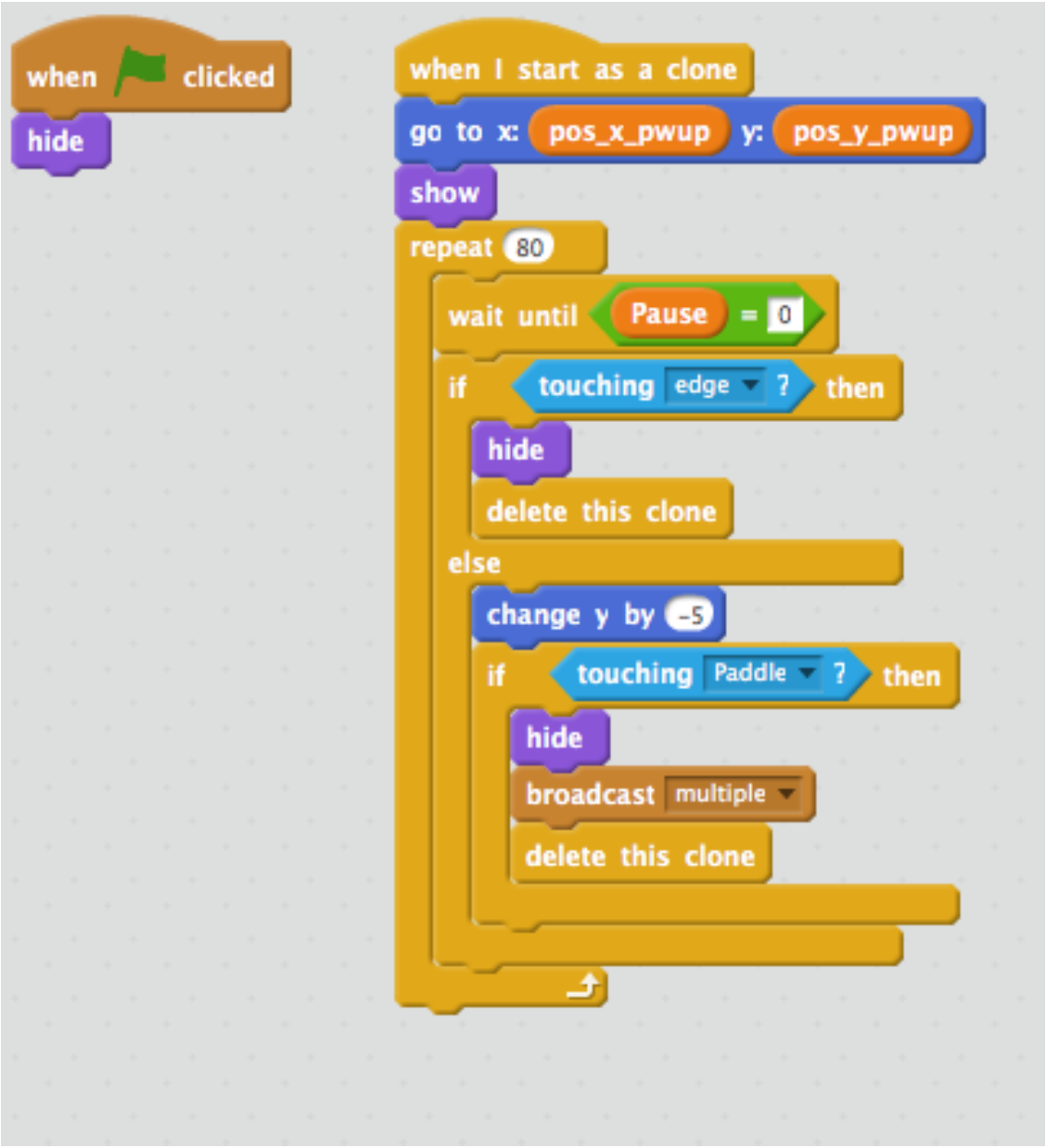

Figura 5.10: *Scripts* para el movimiento vertical de los *power ups*.

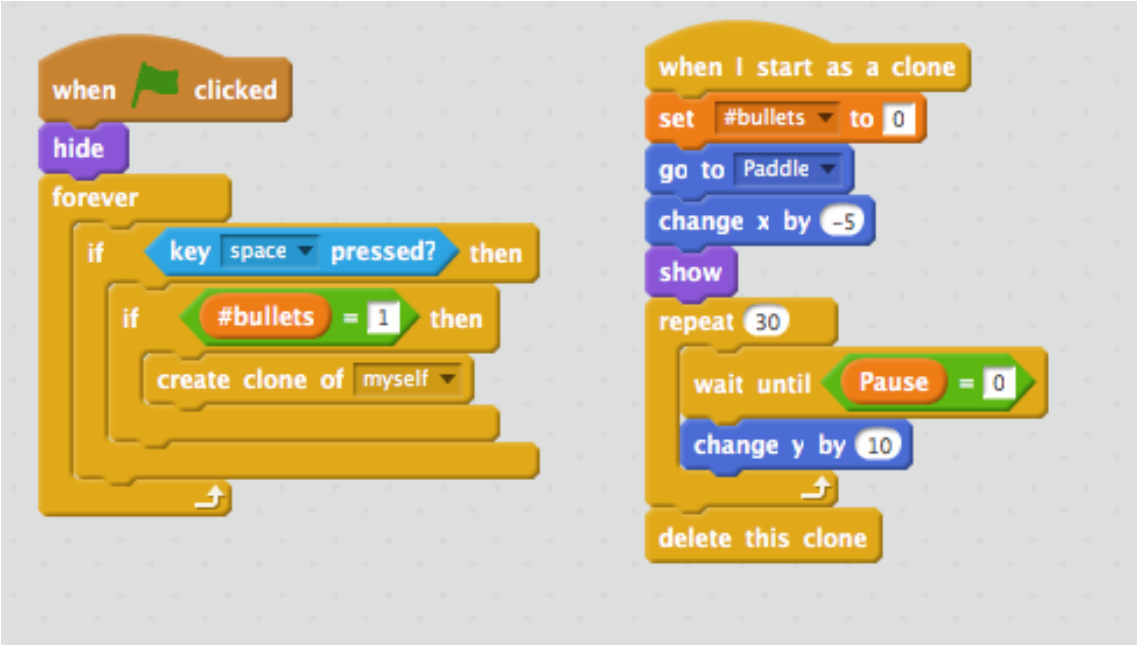

Figura 5.11: *Scripts* correspondientes al *power up* misil.

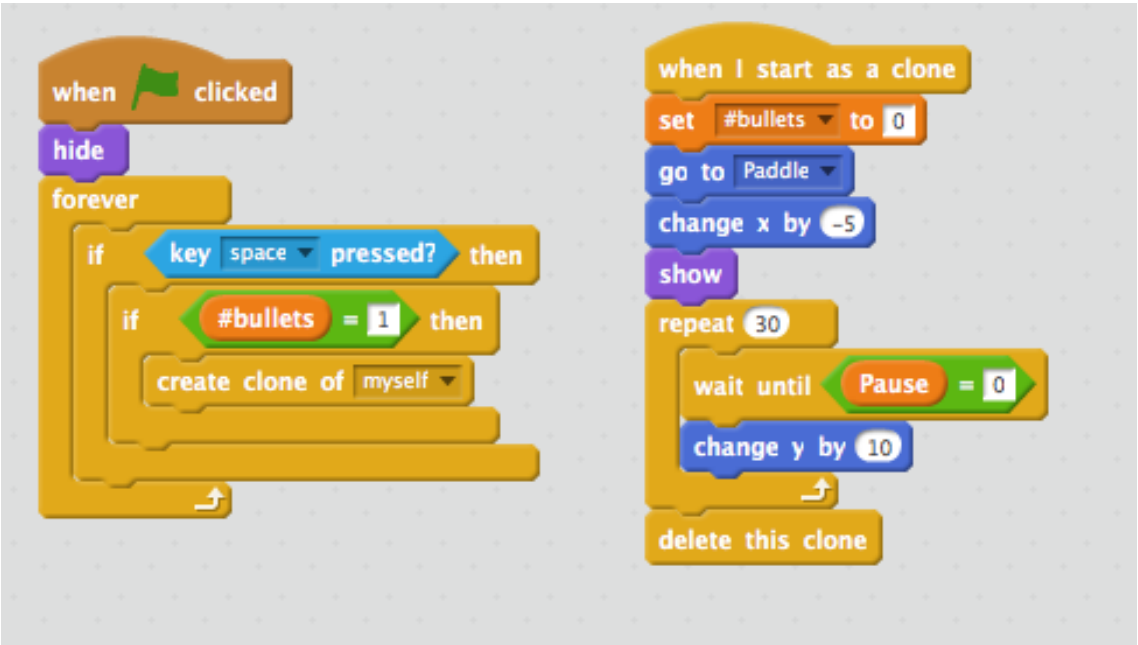

Figura 5.12: Métodos para la gestión del videojuego.

CAPÍTULO 6
Puzzle Bobble

El último videojuego realizado en este trabajo ha sido Puzzle Bobble. El juego consiste en explotar todas las burbujas que se muestran en la pantalla. En este capítulo vamos a explicar detalladamente el proceso de desarrollo para este videojuego en la plataforma de Scratch.

6.1 Diseño del videojuego

Para el diseño de este videojuego se ha utilizado la metodología de los videojuegos explicados anteriormente. En primer lugar, se ha procedido a una búsqueda de imitaciones al videojuego original en la plataforma de Scratch. Han sido pocos usuarios los que se han atrevido a realizar este videojuego.

Tras conocer las mecánicas del videojuego y las estructuras de los demás usuarios de Scratch, se han creado los principales *sprites*: la flecha controlada por el jugador y las burbujas. En la figura 6.1 se puede observar el resultado final de los objetos utilizados en el videojuego.

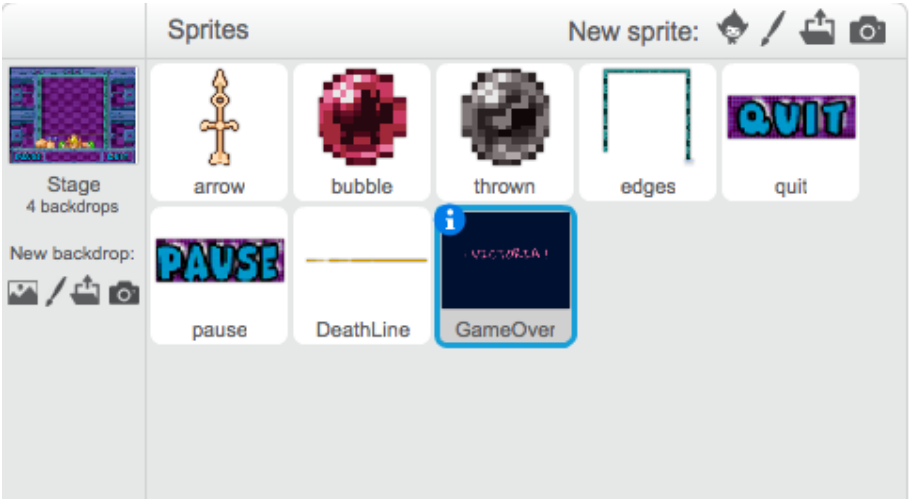

Figura 6.1: *Sprites* escogidos para el videojuego de Puzzle Bobble.

Finalmente, para seguir con la misma metodología de diseño de los demás videojuegos, se ha desarrollado un diagrama de estados para el objeto que manejará el jugador. Para ello, se ha hecho uso de la herramienta utilizada en el videojuego de Breakout.

El diagrama de estados de este videojuego es muy simple (figura 6.2). Se han encontrado 3 estados distintos para el objeto que controla el jugador: **mover** la flecha de

izquierda a derecha con las flechas del teclado, **disparar** con el espacio y aumentar su **puntuación** al explotar 3 o más burbujas del mismo color. El juego terminará cuando las bolas lleguen a la línea de muerte o cuando ya no queden bolas por explotar.

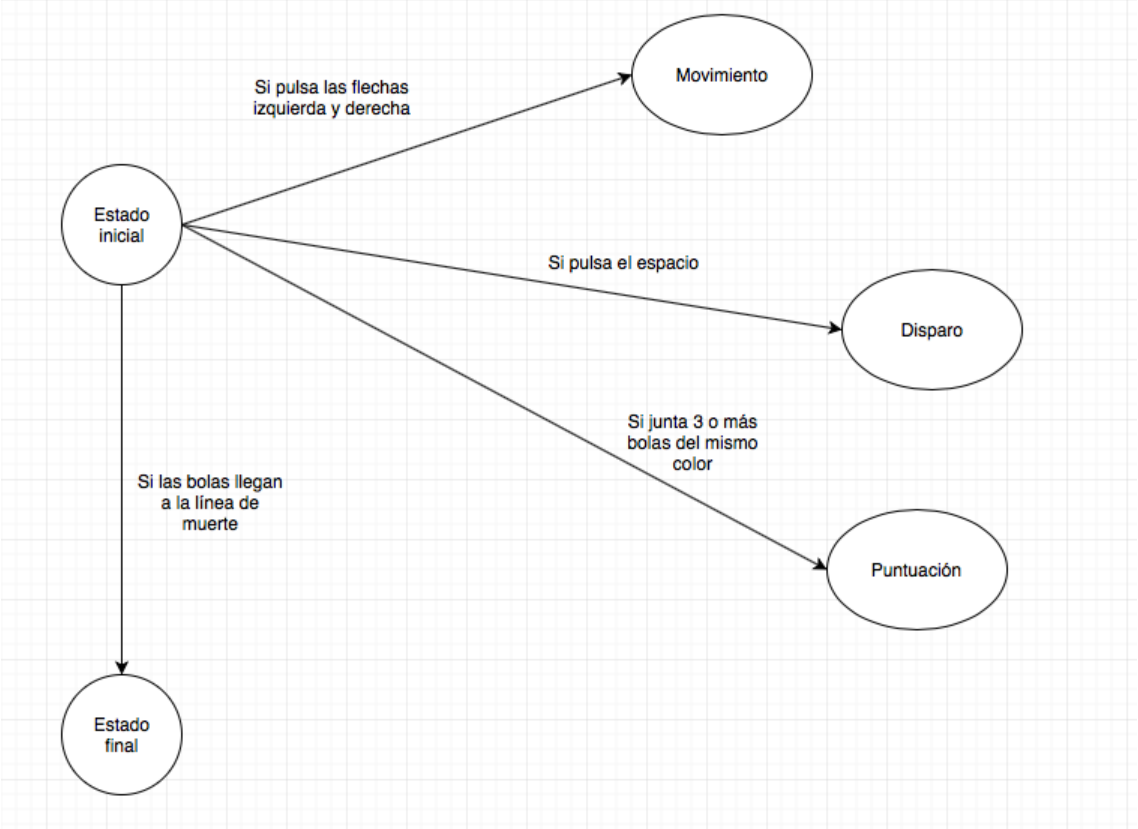

Figura 6.2: Diagrama de estados para el objeto flecha de Puzzle Bobble.

6.2 Modificaciones al videojuego original

El videojuego original consta de una sola pantalla principal, la pantalla de juego. En la parte superior de la pantalla se muestra la puntuación del jugador. Seguidamente se muestran 3 filas de burbujas de distintos colores encapsuladas dentro de un fondo. En la figura 6.3 se puede observar una captura del videojuego original. En la parte inferior, se encuentra el objeto que maneja el jugador y el número de ronda en la que se encuentra actualmente.

Las modificaciones para este videojuego han sido escasas, la fuente de la puntuación del jugador es distinta ya que se ha utilizado la misma para todos los videojuegos de este trabajo. Además, se ha escogido otro fondo para la pantalla del juego. En la figura 6.4 podemos observar el resultado final del videojuego.

Como se ha realizado en los demás videojuegos anteriormente explicados, se he diseñado un menú principal para Puzzle Bobble. En la figura 6.5 se puede observar el resultado final para este menú principal. Este menú dispondrá de las mismas opciones que los menús anteriores: jugar, instrucciones y créditos.

Figura 6.3: Pantalla de juego de la versión original de Puzzle Bobble.

Figura 6.4: Pantalla de juego de la versión realizada de Puzzle Bobble.

6.3 Implementación del videojuego

La metodología para implementar el videojuego ha sido similar al de los demás videojuegos. Tras conocer la mecánica y sus principales objetos, se ha procedido a implementar cada *sprite* junto con sus funcionalidades.

Se han encontrado 3 objetos principales en el videojuego implementado para este proyecto: la flecha controlada por el jugador, las burbujas a explotar y las burbujas lanzadas por el jugador. Como en los videojuegos anteriores, también se debe tener en cuenta el es-

Figura 6.5: Menú principal realizado para el videojuego Puzzle Bobble.

cenario ya que será el que controle el estado del videojuego. A continuación se explicará las funcionalidades de cada uno de estos *sprites*:

1. **Flecha**: La flecha es el objeto controlado por el jugador. Su único objetivo es explotar las burbujas que aparecen en la pantalla. Como se puede observar en la figura 6.6, la única funcionalidad de este *sprite* es mover la flecha en un sentido u otro según el jugador vaya pulsando las flechas del teclado.

2. **Burbujas a explotar**: Este objeto ha sido el más difícil de implementar. Este *sprite* es el encargado de comprobar si hay más de 3 burbujas tocándose del mismo color y si es así, explotarlas. Al iniciar el juego se crean 3 filas de burbujas en la parte superior del escenario eligiendo un color aleatorio. En la figura 6.7 se puede observar su inicialización.

   Conforme se vayan creando las burbujas, éstas llamarán a distintas funciones definidas para comprobar si se están tocando con otras de su mismo color.

   En Scratch no ha sido fácil de implementar estas funcionalidades ya que necesitamos hacer uso de un array multidimensional o de un HashMap. Se ha hecho el uso de listas para guardar la posición horizontal y vertical. Además, se ha creado una lista para llevar el contador de cuantas bolas se están tocando del mismo color.

   Por cada burbuja que se va creando comprueba si los de su alrededor (arriba, abajo y los lados) son del mismo color. Cuando esta función acaba, actualiza el contador de las burbujas más próximas y del mismo color. En las figuras 6.8 y 6.9 se muestra sus implementaciones.

   Estos métodos van comprobando de las listas de *row* y *column* si las burbujas próximas son del mismo color de la burbuja recién creada. En el método de comprobación de colores solo se irá incrementando un contador y en el método de actualización irá actualizando el contador de cada una de las burbujas que coincidan con el color con un método recursivo y así actualizarlas a todas. Además el contador se registrará en un lista.

Figura 6.6: Método para la dirección de la flecha del videojuego Puzzle Bobble.

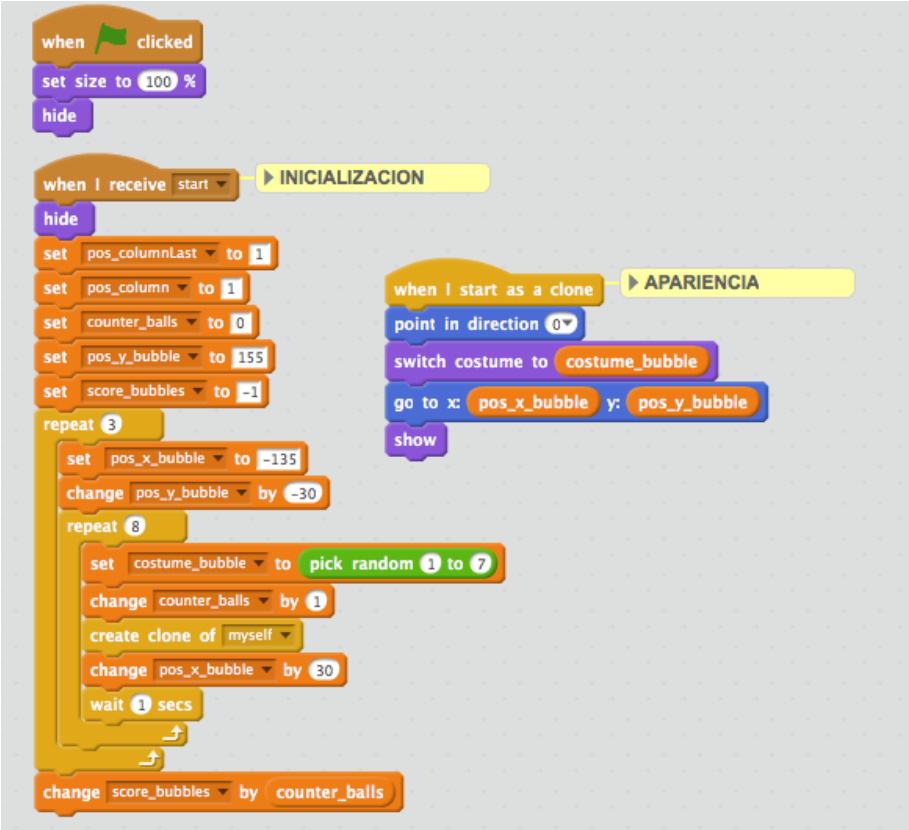

Figura 6.7: Método para la inicialización de las burbujas.

Además, se ha implementado un método destrucción. En la figura 6.10 se puede observar su implementación. Este método debería actualizar las listas de *row*, *column* y la del contador e inicializar la posición correcta a 0.

Debido a las limitaciones de Scratch no se ha podido realizar correctamente la destrucción de las burbujas ya que se ha observado muy complicado realizar los métodos explicados anteriormente. En un futuro se espera terminar el videojuego correctamente y poder compartir a los demás usuarios de este videojuego tan entretenido.

3. **Burbujas lanzadas**: Estas burbujas son las que el jugador lanzará al pulsar el espacio del teclado. Se han creado como un *sprite* independiente para poder controlar su movimiento rectilíneo y además para mostrar al jugador el color de la siguiente burbuja. En la figura 6.11 se puede observar los métodos utilizados para controlar la creación de las burbujas.

   Para el método de inicialización se ha declarado una variable llamada "*thrown*" para comprobar si el jugador ha pulsado el espacio y así crear una nueva burbuja para mostrar al jugador el color de la burbuja siguiente. Además, se ha definido una función denominada "*Direction Bubble*" para mover la burbuja de forma rectilínea y rebotar en los bordes del escenario. En la figura 6.12 se puede observar su implementación.

   Este movimiento continuará hasta que el sensor detecte que se está tocando con el *sprite "Bubble"*, las burbujas anteriormente explicadas. Una vez se haya detenido esta burbuja se creará un objeto de tipo "*Bubble*" ya que se ha comprobado que Scratch no posee una funcionalidad de detectar si se están tocando dos objetos del mismo tipo.

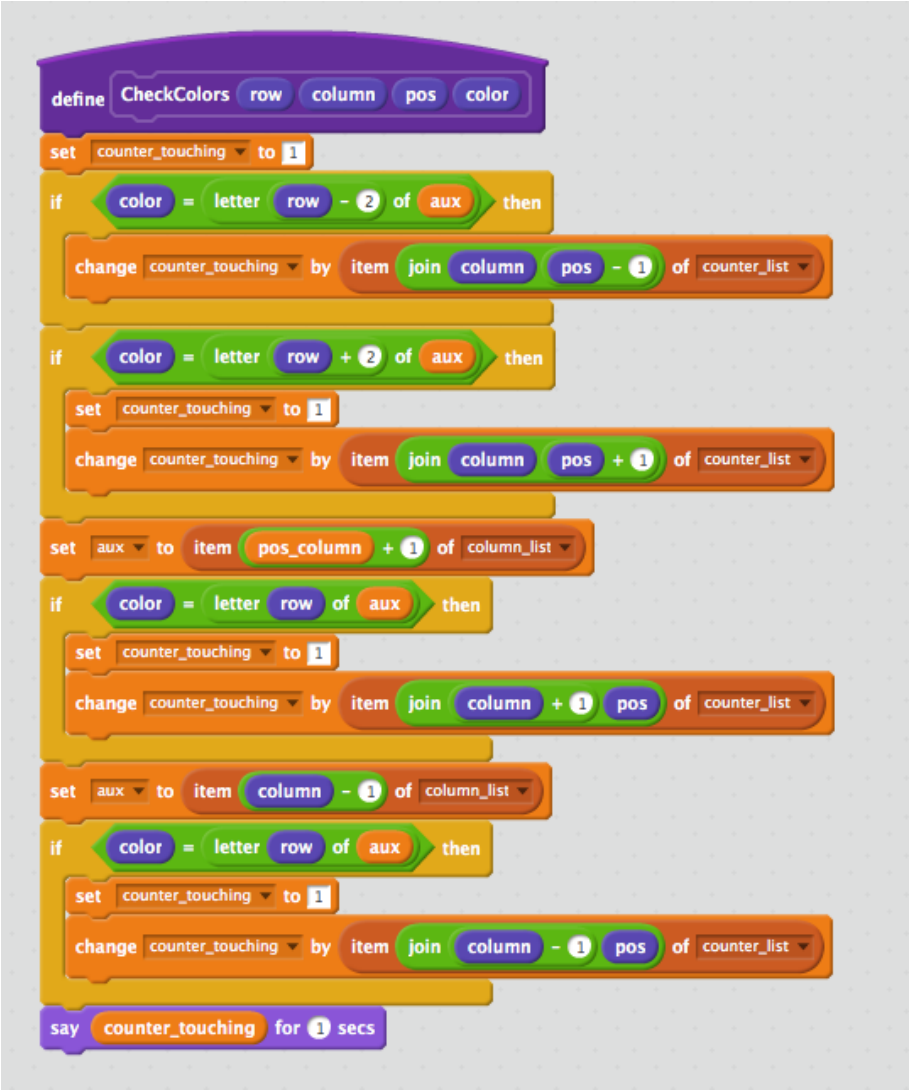

Figura 6.8: Implementación para la comprobación de burbujas del mismo color.

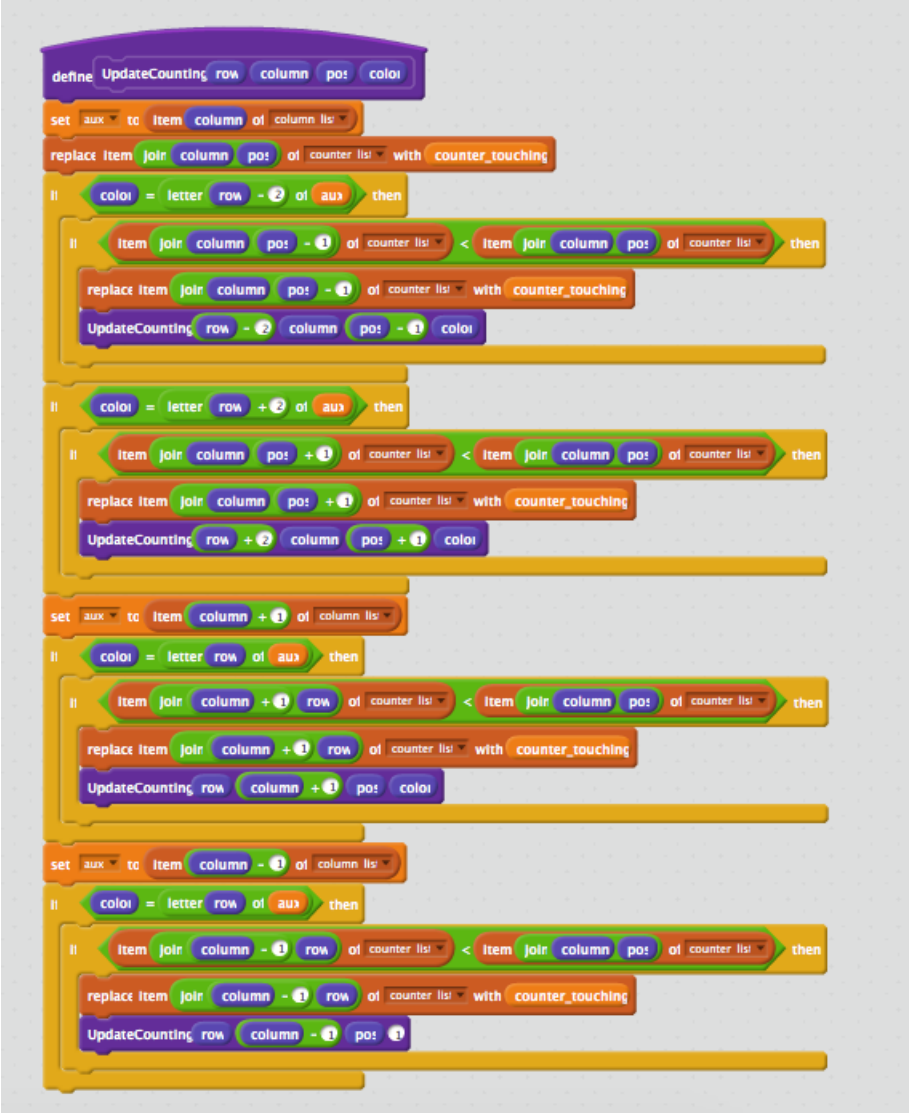

Figura 6.9: Función de actualización de burbujas tocándose del mismo color.

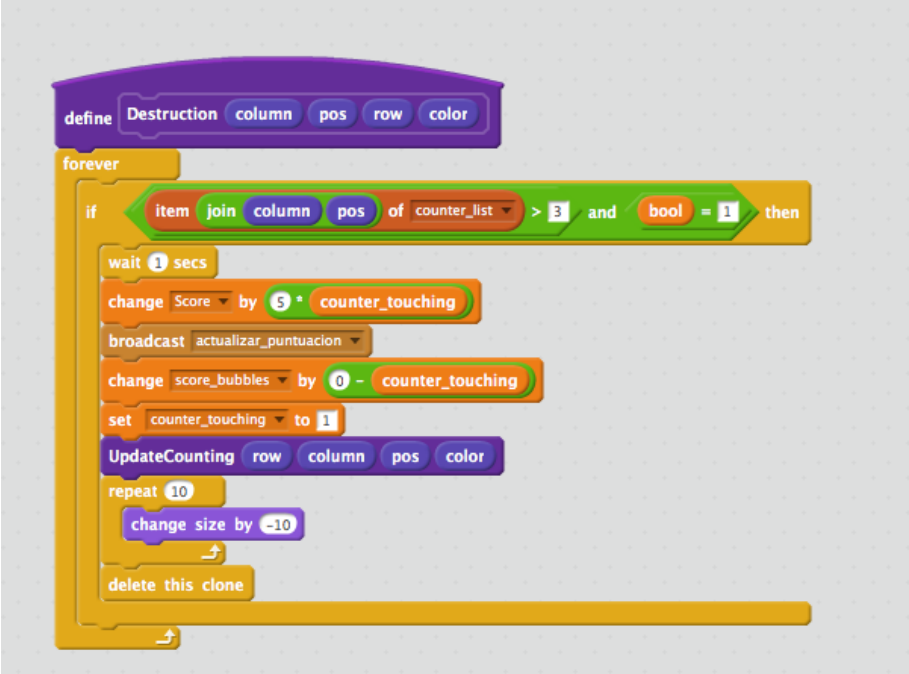

Figura 6.10: Implementación para la destrucción de las burbujas.

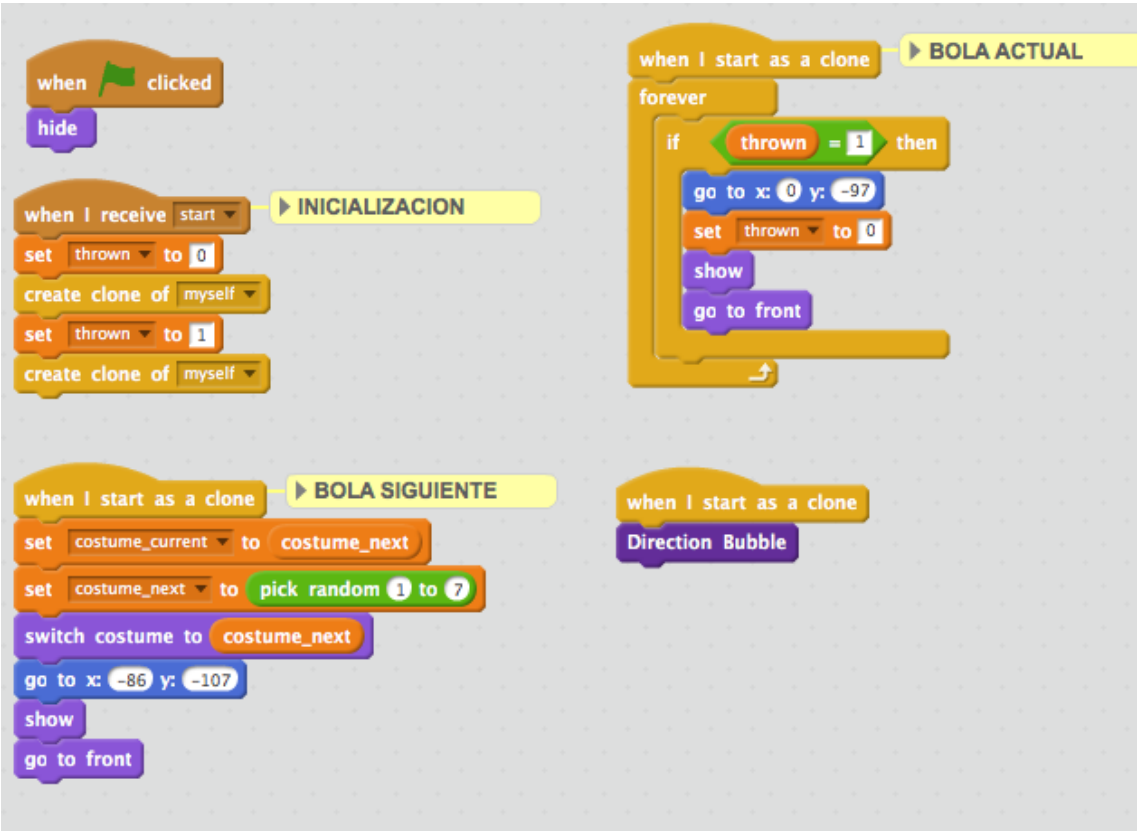

Figura 6.11: Inicialización de las burbujas lanzadas por el jugador.

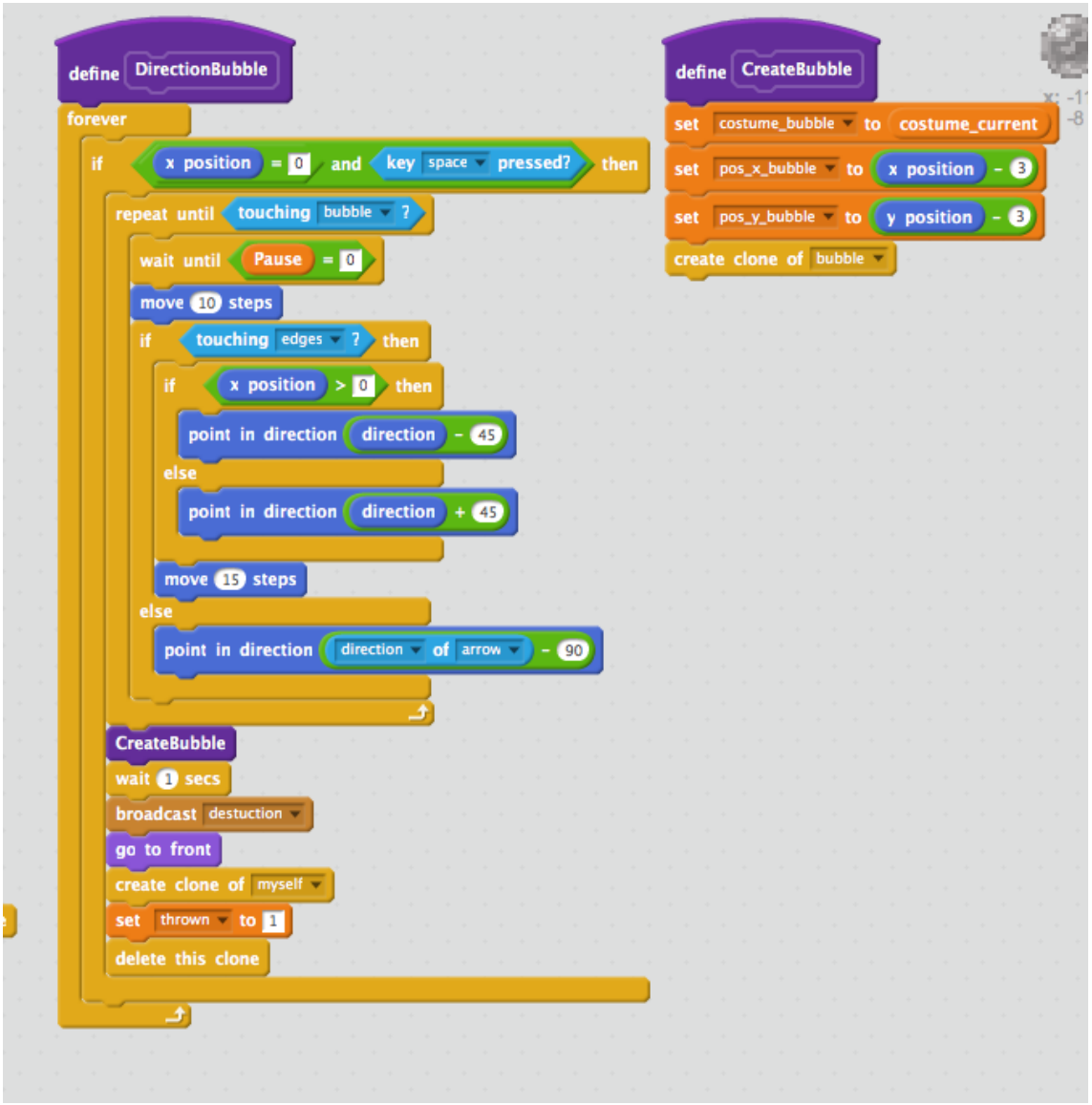

Figura 6.12: Definición de la función de movimiento de las burbujas lanzadas por el jugador.

4. **Escenario**: El escenario, como se ha mencionado en los anteriores videojuegos, será el encargado de comprobar el estado de videojuego además de inicializar las variables globales creadas para el inicio del juego. En la figura 6.13 se puede comprobar las implementaciones realizadas.

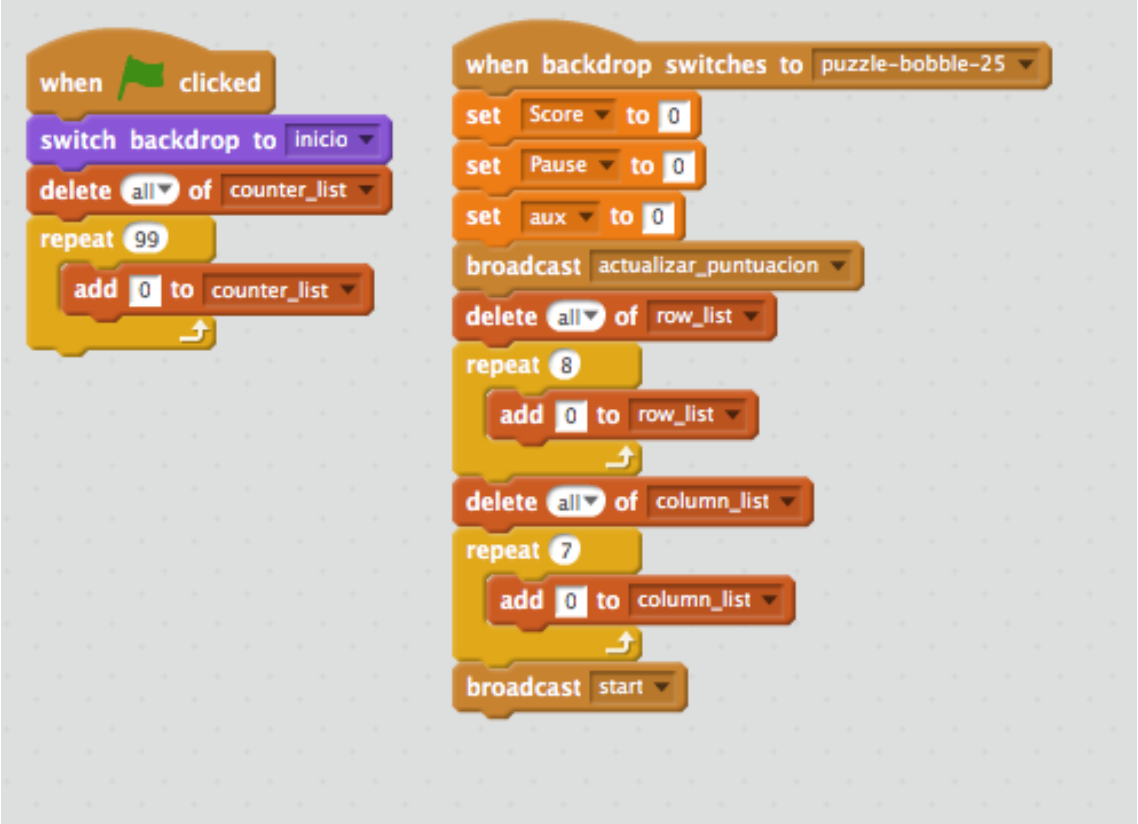

Figura 6.13: *Scripts* definidos para el escenario de Puzzle Bobble.

CAPÍTULO 7
Página Web

Este capítulo se centra en la elaboración de una página web para el museo de la escuela. En esta página se han incluidos los tres videojuegos realizados en este trabajo. Gracias a la plataforma de Scratch ha sido tan fácil como incluir el código HTML que ofrece la plataforma para compartir los proyectos en otras páginas web.

7.1 Implementación

Como ya se ha comentado, el objetivo de este trabajo ha sido crear tres videojuegos de la edad de oro de informática en el lenguaje de programación de Scratch. Una vez terminados, se ha realizado un apartado en la página web del Museo de Informática de la Escuela Técnica Superior de Ingeniería Informática para que sus visitantes disfruten jugando y motivarlos a empezar a programar con este divertido lenguaje.

Para comenzar a implementar la página web, se ha inspeccionado las ediciones anteriores realizadas por Samuel Villaescusa Vinader y Miguel Marqués Moros. Una vez conocida la estructura y diseño de la página web se ha comenzado a construir un archivo HTML en un editor de texto, Sublime Text. Se ha copiado la estructura HTML de las ediciones anteriores y se ha modificado el texto a mostrar junto con los enlaces de los videojuegos. En la figura 7.1 se puede observar parte de código realizado para la página web.

```
<!-- Start Page Content -->
<div class="row-wrapper-x"><div class="row-wrapper-x" style="text-align: left;">
<p style="text-align: center;"><span style="color: #d60066;"><strong><span style="font-size: 2em; line-height: 1.5em;">
Videojuegos clásicos con Scratch (III)</span></strong></span></p>
<h3 style="text-align: center; color: #404040;"><strong>Patricia Ruá Lozano</strong></h3>

<p>Tercera edición de la creación de videojuegos clásicos que incluye tres grandes videojuegos de la historia de la
informática con el objetivo de disfrutar con ellos a la vez que aprender.</p>

<p>Los videojuegos han sido implementados en el lenguaje de programación <strong>Scratch</strong>. Esta plataforma posee
una comunidad en la cual los usuarios comparten sus videojuegos. Existe una opción llamada "reinventar" que permite a los
usuarios conocer el código fuente de los videojuegos y así no partir de cero en la programación. Gracias a la comunidad
de <strong>Scratch</strong> y al estudio previamente realizado, se ha logrado realizar estos videojuegos.</p>

<h3 style="text-align: center;"><strong>Space Invaders</strong></h3>

<h3><strong>Historia</strong></h3>
<p>Space Invaders es uno de los primeros videojuegos arcade diseñado por Toshihiro Nishikado y lanzado al mercado por
primera vez en 1978. Fue vendido por la empresa Taito Co. en Japón, y posteriormente distribuido en Estados Unidos por
Midway Games.</p>
<p>Space Invaders fue uno de los primeros juegos shoot 'em up. Fue uno de los precursores de los videojuegos y de gran
ayuda a la expansión de este sector. Un juego exitoso y popular desde su lanzamiento.</p>
```

Figura 7.1: Parte de código realizado para la página web del museo.

7.2 Organización de la página web

La organización que se ha seguido es similar a las páginas web realizadas para ediciones anteriores. Al comienzo de la página se muestra el título que da nombre a nuestro apartado: “Videojuegos clásicos con Scratch (III)”. Se ha escogido el mismo título cambiando solamente el número de edición. Seguidamente, se ha mostrado el nombre de la encargada en realizar los tres videojuegos en Scratch y una breve introducción. Posteriormente, se muestran los videojuegos realizados para este proyecto.

Para cada videojuego se explica cada poco de su historia para poner en contexto a los usuarios que visiten la página. Acto seguido se ha mostrado los objetivos de cada videojuego para conocer la misión de cada uno de los videojuegos realizados. Por último se muestran las instrucciones básicas. Además, se ha embebido el videojuego para poder jugar desde la página web y también se muestra un enlace de cada videojuego dentro de la comunidad de Scratch. Para poder añadir el videojuego en la página web Scratch ofrece una opción dentro de cada proyecto que incluyo una porción de código HTML para poder copiarlo en nuestra implementación.

Por último, se han añadido notas adicionales ya que no se ha podido implementar del todo bien ciertos aspectos en Scratch. En la figura 7.2 se puede observar el aspecto final de la página realizada. Actualmente la página web del museo está realizando cambios, con lo cual no estará disponible este apartado hasta que finalicen los cambios.

Figura 7.2: Página web realizada en la que se incluye una breve introducción y parte de la explicación de uno de los videojuegos realizados.

CAPÍTULO 8
Conclusión

Último capítulo de la memoria, en él se realiza una opinión personal sobre la plataforma Scratch obtenidas a partir del trabajo realizado y un análisis de los objetivos alcanzados.

8.1 Consideraciones finales

En el trabajo realizado se demuestra que, con la utilización de un lenguaje de programación orientado al público infantil como Scratch, se puede realizar la imitación de videojuegos que en su época supusieron un reto tecnológico. Para la realización de los videojuegos se han consultado ciertos artículos que pueden consultarse en la bibliografía al final del trabajo.

Lo primero que se ha realizado es un estudio histórico de cada videojuego explicando al lector lo más característico de la época en la que se creó cada videojuego. Posteriormente se explica detalladamente el desarrollo de cada uno de los videojuegos escogidos para este trabajo, haciendo hincapié en el diseño previo, las modificaciones realizadas al videojuego original y su implementación en Scratch.

Tras la realización de cada videojuego, se ha diseño un apartado para la página web del museo mostrando su contexto histórico, las instrucciones de juegos y notas adicionales por cada uno de los videojuegos. El objetivo de esta página web es divertir a sus visitantes además de enseñar un poco más sobre el contexto histórico de los videojuegos.

Este trabajo pone especial énfasis en el lenguaje de Scratch. Como ya se explicó, este lenguaje introduce a usuarios noveles en el mundo de la programación, para ello ofrece un lenguaje intuitivo basado en bloques que no requiere escribir ninguna línea de código. Esta manera interactiva facilita la programación con respecto a otros lenguajes, pero puede resultar algo limitada ya que las funcionalidades que ofrece Scratch, pueden resultar insuficientes para recrear ciertas aspectos de los videojuegos. En el capítulo 6 se puede observar que la falta de uso de un array multidimensional dificulta la comparación entre burbujas y el uso de clones no es nada fácil a la hora de esta comparación. Para un usuario más avanzado, es mucho más fácil esta implementación en un lenguaje como el de Java.

En resumen, Scratch ha demostrado ser lo bastante potente para poder desarrollar videojuegos. Este lenguaje se caracteriza por ser sencillo y visual además de fomentar el aprendizaje de la programación. A pesar de todas las funcionalidades que ofrece Scratch, no se ha podido realizar ciertos aspectos de los videojuegos ya que se han observado ciertas limitaciones. No obstante, se puede decir que Scratch es un lenguaje bastante intuitivo

y fácil de usar pero para usuarios más avanzados este lenguaje puede ser limitado para recrear ciertas acciones.

Bibliografía

[1] CASTELLS Y NEL (2015). *¿Qué es Scratch y qué beneficios tiene para el aprendizaje?* España: CEF NAC. Noticias Tecnología en la Educación.

[2] COHEN D.C. (2016) *The History of Classic Video Games: The Golden Age.* https://www.lifewire.com/the-golden-age-and-first-generation-729751 [Consulta: Marzo 2017]

[3] COLLADO L (2011). *Análisis Puzzle Bobble Universe.* Japón: Esfera Fantástica, S.L.

[4] GUTIÉRREZ OROZCO,JORGEA.(2007). Máquinas de estados finitos. Ciudad de México: Escuela Superior de Cómputo.

[5] LÓPEZ GARCÍA, J.C. (2013). *Guía de referencia de Scratch 2.0.* http://www.eduteka.org/pdfdir/ScratchGuiaReferencia.pdf [Consulta: Abril 2017.]

[6] MARQUÉS MORO, MIGUEL (2016). *Diseño e implementación de videojuegos clásicos con Scratch.* Trabajo final de grado, Escuela Técnica Superior de Informática, Universidad Politécnica de Valencia.

[7] STRATEGYWIKI (2014). *Space Invaders' guide.* https://strategywiki.org/wiki/Space_Invaders [Consulta: Marzo 2017.]

[8] SCRATCH (2017). *Versión 2.0 de Scratch.* https://wiki.scratch.mit.edu/wiki/Scratch_2.0 [Consulta: Abril 2017].

[9] SCRATCH (2017). *Lifelong Kindergarten Group.* https://wiki.scratch.mit.edu/wiki/Lifelong_Kindergarten_Group [Consulta: Abril 2017]

[10] RESNICK, MITCHEL (2013). *Lifelong Kindergarten.* Cultures of Creativity, LEGO Foundation. http://web.media.mit.edu/~mres/papers/CulturesCreativityEssay.pdf [Consulta: Abril 2016.]

[11] VELASCO JJ (2011). *Historia de la tecnología: Breakout, otro gran éxito de Atari.* España: Hipertextual: Tecnología Videojuegos.

[12] VILLAESCUSA VINADER, SAMUEL (2015). *Diseño de videojuegos clásicos en Scratch.* Trabajo final de grado, Escuela Técnica Superior de Informática, Universidad Politécnica de Valencia.

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