Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE COMPUTADORES

Curso Académico 2023/2024

Trabajo Fin de Grado

USO DE MESA MULTI-TOUCH PARA EL APRENDIZAJE DE SCRATCHJR EN UN CONTEXTO COLABORATIVO

Autor: Guillermo J. Gª-Delgado Álvarez

Director: Maximiliano Paredes Velasco

RESUMEN

Con el fin de mejorar la enseñanza de conceptos básicos de programación a niños, este proyecto se desarrolla con el propósito de crear una herramienta interactiva que ayude a aprender a programar de forma colaborativa y sirva como una introducción pedagógica a las nuevas tecnologías.

En el proyecto se usa una mesa multi-touch como medio principal para usar la aplicación, con el objetivo de permitir que varios alumnos puedan interactuar con ella simultáneamente. Para ello, el software tiene las funciones de gestionar la cooperación entre los alumnos y la creación, composición y ejecución de scripts programados con bloques Scratch. Además, se incluyen funcionalidades complementarias que mejoran la experiencia de uso, como la capacidad de modificar aspectos estéticos, el flujo entre pantallas y otros elementos de apoyo que mejoran la usabilidad del sistema.

La estructura del trabajo se divide en varios capítulos que abarcan desde la motivación y los objetivos del proyecto hasta el desarrollo y la evaluación de la aplicación. Se incluye un marco teórico que explora la evolución histórica de la enseñanza de la programación, un análisis del estado del arte de diversas herramientas relacionadas, y casos de estudio sobre diferentes enfoques y tecnologías aplicadas en la enseñanza de programación. Posteriormente, se detalla el diseño del prototipo en *Figma* y *Unity*, describiendo el proceso de desarrollo, la interfaz diseñada, los bloques de programación y la dinámica de uso de la aplicación.

En conclusión, este proyecto aporta una herramienta diseñada y desarrollada con el objetivo de hacer el aprendizaje, concretamente el de la programación, más llamativo y lúdico que otras herramientas para los niños que tienen el mismo propósito gracias al aspecto colaborativo y las dinámicas de grupo, y la interactividad con el hardware, destacando el ser una mesa táctil de grandes dimensiones.

Índice

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En la era digital, el dominio de habilidades relacionadas con la programación se ha vuelto esencial para el desarrollo académico y profesional en la mayoría de los sectores laborales. Mientras la tecnología continúa transformando nuestra sociedad, la capacidad de comprender y trabajar con múltiples herramientas software se presenta como una competencia fundamental, no solo a nivel técnico, sino también como catalizador para la creatividad, el pensamiento lógico y la resolución de problemas, entre otras aptitudes. Por estos motivos es útil fomentar el interés en el aprendizaje de la programación desde edades tempranas, aprovechando la capacidad innata de los niños para asimilar conceptos nuevos con mayor facilidad.

Este trabajo se adentra en un enfoque particular de la enseñanza de programación para niños: la metodología colaborativa en grupos de alumnos haciendo uso de una herramienta común, una mesa *multi-touch*. El entorno cooperativo, en este contexto, se presenta como un vehículo efectivo para cultivar un entorno de aprendizaje que nutre no solo en el área del aprendizaje de la programación, sino también la capacidad de trabajo en equipo y la resolución autónoma de problemas.

1.1. Motivación

El desarrollo del prototipo se basa en el diseño y evaluación de una aplicación pensada específicamente para enseñar a programar de forma cooperativa a grupos de niños. La enseñanza de programación presenta varias dificultades inherentes, especialmente para los alumnos de educación primaria, ya que se trata de una actividad abstracta y difícil de comprender al principio. La programación textual, en particular, puede ser muy compleja para los más jóvenes debido a factores como la necesidad de comprender y manejar una sintaxis precisa y detallada, la memorización de términos nuevos o el entendimiento del flujo del código.

El problema que este proyecto busca resolver es la barrera inicial que enfrentan los niños al aprender programación, proporcionando una solución que facilite la comprensión y aplicación de conceptos de programación desde una edad temprana. La

motivación detrás de este proyecto radica en reconocer que la programación no solo es una habilidad técnica, sino también una herramienta para desarrollar habilidades sociales y cognitivas esenciales.

Para abordar estas dificultades, se ha recurrido a la programación por bloques para la herramienta, puesto que es un sistema que simplifica el proceso de entender cómo funciona la programación y que permite a los niños construir programas mediante la manipulación de elementos visuales, que son más fáciles de recordar y entender. Cada bloque representa una instrucción del código, y de esta manera se reduce la carga cognitiva y el aprendizaje se vuelve más accesible y atractivo para los niños.

El proyecto tiene como objetivo general el fomentar el aprendizaje de la programación de manera práctica y experiencial. Al tocar y manipular digitalmente el contenido educativo con la ayuda de imágenes visuales, sonidos y la interactividad táctil, se estimulan los sentidos de los niños. Esto refuerza la comprensión de conceptos abstractos y facilita la retención de información [1]. Además, se busca proporcionar una forma rápida y dinámica de poner a prueba los conocimientos del alumno, permitiéndoles ver los resultados de sus programas de inmediato, sin necesidad de esperar a una supervisión o corrección externa.

Es por esto anteriormente expuesto que el medio propuesto como espacio de aprendizaje donde los alumnos interactuarán con la aplicación sea una mesa interactiva o mesa multi-touch, que permite la experiencia cooperativa deseada gracias a su formato. El empleo de esta herramienta en la docencia infantil presenta numerosos beneficios, puesto que combina la tecnología táctil con el aprendizaje práctico [2]. Esto ofrece ventajas tanto para educadores como para los alumnos, facilitando y haciendo significativamente más dinámico y entretenido el proceso educativo; permitiéndonos llegar a los objetivos deseados.

En resumen, este proyecto pretende hacer que el aprendizaje de la programación sea más accesible y atractivo para los niños, utilizando una mesa multi-touch para crear un entorno de aprendizaje colaborativo e interactivo que no solo facilite la adquisición de habilidades técnicas, sino que también promueva el desarrollo de habilidades sociales y cognitivas desde una edad temprana [3].

1.2. Desglose de objetivos

El propósito principal de esta investigación es mejorar el aprendizaje de *Scratch Jr*. a través de la implementación del diseño funcional de una aplicación que facilite un entorno colaborativo, fomentando en todo momento interactividad entre los niños. Por ello se examinarán aspectos clave del desarrollo de esta herramienta, como la interfaz de usuario, la adaptabilidad a diferentes niveles de habilidad y la capacidad de promover la participación de los niños en actividades de programación en grupo.

Objetivos del Principales

Estos describen las tareas que se deben realizar durante el proyecto para obtener un producto final satisfactorio y obtener conclusiones útiles del mismo:

- Realizar una revisión histórica del aprendizaje de la programación.
- Revisar las herramientas de programación de bloques y el uso de tecnologías táctiles en las aulas.
- Diseño e implementación de la herramienta que será usada en la mesa *multi-touch*.
- Desarrollar un sistema para que la herramienta soporte la interacción colaborativa
- Llevar a cabo una evaluación heurística con expertos de la aplicación final.

Objetivos pedagógicos

En primer lugar se procede a describir los objetivos pedagógicos y personales que la herramienta debe asegurar. Estos cumplen la función de indicadores de calidad en cuanto al propósito de uso dentro del aula y como herramienta útil para los docentes:

- **Pensamiento lógico**: fomentar la resolución de problemas y ejercitar el raciocinio a través de la programación desde edades tempranas.
- **Creatividad**: estimulación de la imaginación a través de las múltiples combinaciones de bloques posibles, permitiendo dar vida a las ideas de los alumnos.

- **Habilidades de trabajo en equipo**: al ser una herramienta que permite un uso simultáneo por varios alumnos se fomentan aspectos como la colaboración y comunicación, mejorando las habilidades sociales y de gestión en grupo de los usuarios.

Objetivos de usabilidad

Por otro lado, las características que se deben tener como prioridad para que el diseño del proyecto sea usable son las siguientes:

- **Navegación sencilla**: crear una interfaz de usuario intuitiva y fácil de manipular, con menús y opciones simples y claros; evitando la complejidad innecesaria para que los niños puedan navegar por la aplicación de manera autónoma.
- **Interactividad**: la aplicación debe ser interactiva y atractiva mediante una estética lúdica para mejorar la experiencia de usuario, evitar el aburrimiento y convertir el aprendizaje en un elemento divertido.
- **Contenido educativo**: mantener el enfoque pedagógico de la aplicación mediante el desarrollo de actividades y juegos que fomenten el desarrollo cognitivo, emocional y social de los alumnos, a la vez que se aprende a programar.
- **Programación en equipo**: enfocar las dinámicas de uso de la aplicación para que todos formen parte de las decisiones que se tomen durante las sesiones de trabajo.
- **Accesibilidad**: desarrollar un prototipo orientado a que niños de un rango variable de edad y conocimientos técnicos lo encuentren fácil de usar y divertido.

1.3. Estructura de la memoria

El comienzo de este documento recoge el planteamiento y objetivos del trabajo. En los siguientes apartados trataremos estos temas:

1. Análisis de la evolución de la enseñanza interactiva y de la enseñanza de la programación hasta llegar a la actualidad.

2. Casos de estudio relevantes de proyectos relacionados con la enseñanza y nuevas tecnologías.
3. Planificación del diseño del prototipo según la información extraída de los apartados anteriores.
4. Desarrollo de la aplicación del prototipo en un motor gráfico
5. Análisis de la aplicación y posibles mejoras futuras.

2. FUNDAMENTOS Y TRABAJO RELACIONADO

En el este capítulo se abordará la evolución de la enseñanza de la programación y la incorporación de nuevas tecnologías en el entorno educativo. Desde los primeros intentos por introducir conceptos relacionados con el pensamiento computacional, pasando por la introducción de los ordenadores en las aulas, hasta la actualidad, donde el conocimiento sobre nuevas tecnologías y la informática ha adquirido un carácter esencial.

En los siguientes epígrafes se analizarán hitos reseñables que han configurado la interacción de los estudiantes con este campo de conocimiento. Se expondrá las diferentes filosofías y corrientes de pensamiento que han influenciado la docencia dentro de cada periodo en paralelo con los nuevos hitos tecnológicos coetáneos a estas. A través de la revisión del pasado y el presente, se sentarán las bases para comprender el contexto actual de la enseñanza de la programación y las nuevas tecnologías en las aulas.

2.1. Contexto histórico

A continuación, se expondrán unos análisis breves de los modelos de enseñanza relacionados con la enseñanza de la programación y el uso de nuevas tecnologías en el ámbito escolar clasificados cronológicamente en horquillas de fechas orientativas:

1960-1979

Durante este primer periodo, el sector de la enseñanza experimenta cambios notables en la forma de impartir el conocimiento. Un exponente de estos cambios son los nuevos lenguajes de programación como ***Logo***, creado principalmente por los educadores Seymour Papert, Cynthia Solomon y otros investigadores del Instituto de Tecnología de Massachusetts (MIT) en 1966 [4].

La premisa de este lenguaje era enseñar de una forma accesible y lúdica a niños conceptos fundamentales de programación mediante la introducción de elementos basados en el concepto del "construccionismo", concepto que desarrollaría más en profundidad en décadas posteriores, y cuyo principio propone que los individuos construyen su propio conocimiento a través de la interacción con el entorno aplicando

conceptos teóricos a proyectos prácticos; como la "tortuga gráfica" que se describe en la referencia [5] un recurso visual que Papert usaba con alumnos y que permitía controlar el movimiento de una tortuga virtual mediante comandos de programación, como se observa en la Figura 1 Diagrama de un ejercicio simple usando LOGO para controlar la tortuga,.

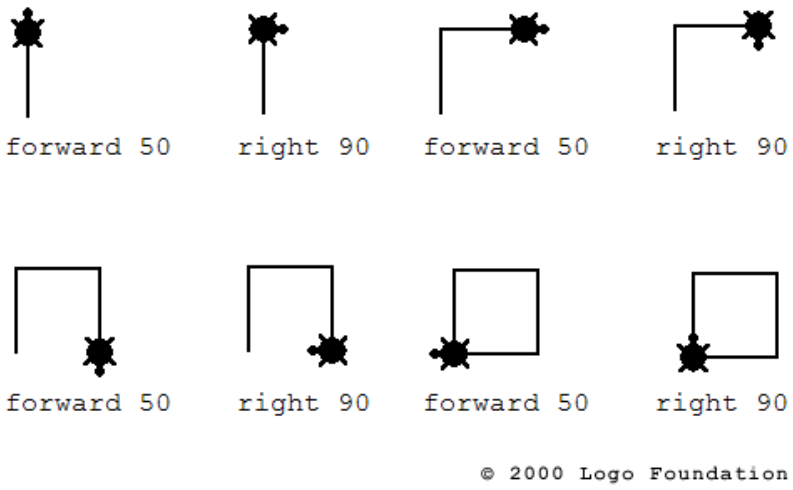

Figura 1 Diagrama de un ejercicio simple usando LOGO para controlar la tortuga

A parte de *Logo*, también se desarrollaron otros lenguajes de programación para principiantes como ***BASIC*** (*Beginner's All-purpose Symbolic Instruction Code* o Código de instrucción simbólica de propósito general para principiantes) por John G. Kemeny y Thomas E. Kurtz (Figura 2) en Dartmouth College, el cual revolucionó la programación de computadoras al proporcionar un lenguaje accesible y fácil de usar para una amplia variedad de usuarios [6]. Fue diseñado como un lenguaje interactivo, con instrucciones sencillas de leer para otros programadores y con soporte en la mayoría de los sistemas operativos.

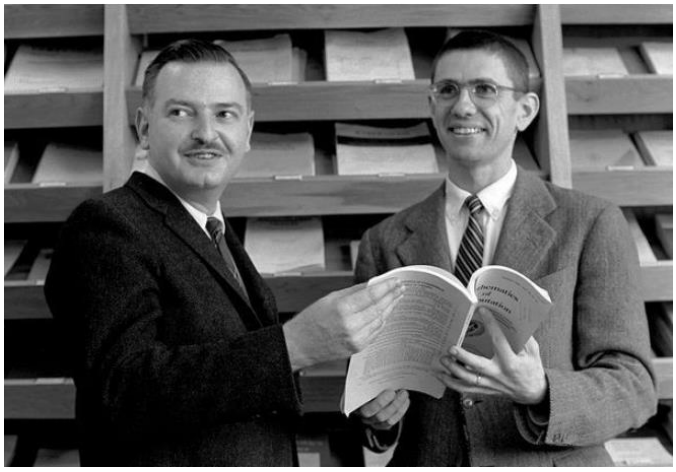

Figura 2 John G. Kemeny and Thomas E. Kurtz, desarrolladores de BASIC (1964)

Por otro lado, en 1972, se desarrolla en la Universidad de Illinois el sistema de enseñanza asistida por computadora llamado ***PLATO*** (Programmed Logic for Automatic Teaching Operations), que aunque no se planteó específicamente para niños, también se acabó usando en entornos educativos de edades más cortas (Figura 3), siendo uno de los primeros sistemas de enseñanza asistida por ordenador, y cuyo objetivo era proporcionar a los estudiantes acceso a lecciones interactivas de programación y matemáticas, haciendo un uso pionero de elementos multimedia en la educación, ya que ofrecía tecnología muy avanzada para esta época como gráficos vectoriales y capacidades de audio. [7] [8]

La herramienta fomentaba un aprendizaje que se ajustase al propio ritmo de cada alumno [8], como se observa en la Figura 3, y proporcionaba retroalimentación inmediata en diversas disciplinas, desde matemáticas y ciencias hasta idiomas y humanidades.

Figura 3 Estudiante interactuando con la pantalla táctil de PLATO (1960's)

Durante la década, el proyecto PLATO avanzó con el desarrollo del sistema con nuevas versiones, incluyendo mejoras y actualizaciones a la original [8], como la conexión simultánea entre varios usuarios para colaborar en proyectos comunes, comunicarse por mensajes o acceder a más aplicaciones educativas, ejercicios y juegos.

En estos años también se empieza a publicar libros educativos, juguetes y kits que introducían conceptos relacionados, basados en el aprendizaje de la lógica de la programación durante las sesiones de juego. Un exponente de este tipo de enseñanza lúdica fue el juego ***Think-a-Dot*** (Figura 4**)**, creado por Joseph Weisbecker [9], el cual, aunque no enseñe literalmente programación, sí que se centra en ejercitar el pensamiento lógico que se usa a la hora de programar.

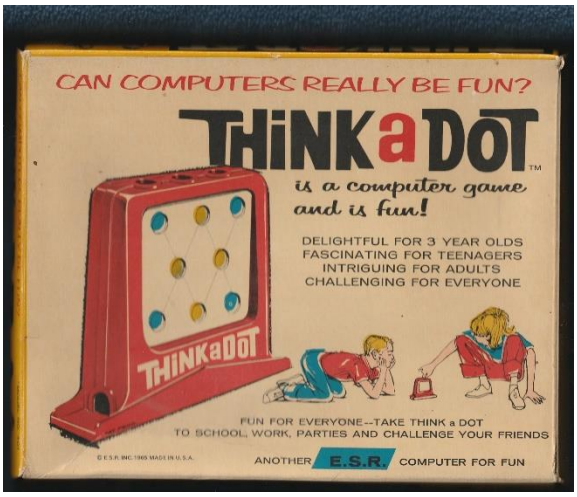

Figura 4 Kit del juego educativo Think-a-Dot (Editorial: E.S.R. INC., 1965)

1980-1999

Durante este periodo comienzan a popularizarse los ordenadores personales (*PC* en inglés) como *Commodore 64* o *Apple II*, fomentando el acceso a la programación en los hogares. *BASIC*, desarrollado en décadas anteriores, fue una de las herramientas comunes para la introducción a este campo durante este periodo de tiempo. No obstante, la enseñanza de la programación todavía se consideraba en gran medida una actividad dirigida a adolescentes y adultos jóvenes.

Los PC de esta etapa no solo tuvieron uso doméstico, también se introdujeron en las aulas de forma paulatina. Por ello se apuesta en esta etapa por el desarrollo y la implementación de programas interactivos, juegos educativos y softwares diseñado específicamente para facilitar el aprendizaje de diversas materias, junto con el uso de disquetes, enciclopedias en CD-ROM como ***Encarta*** [10], y otras tecnologías complementarias enfocadas a mejorar la experiencia de estudio o a una mayor interacción por parte de los alumnos durante las clases, como el uso de proyectores y pantalla, máquinas de escribir electrónicas o las calculadoras gráficas y científicas.

Por un lado, cabe destacar la publicación revistas con el objetivo de promover la difusión y enseñanza de la programación, como "***Compute!***" (Figura 5) o "***BYTE***". La primera comenzó en 1979, alcanzó su pico de popularidad durante la década de los 80's y estaba orientado a escribir programas para todos los equipos que utilizan alguna

versión del procesador de 8 bits "*MOS 6502 CPU*" y artículos sobre el Commodore PET, VIC-20, Atari 400/800, Apple II+ entre otros [11]; mientras que "***BYTE***" [12] se centró en temas relacionados con la informática, hardware, software, programación y tecnología en general, que a pesar de estar orientada a un público más adulto y profesional, también contenía secciones con código fuente completo de programas, lo que permitía a los lectores aprender programación y experimentar con nuevos proyectos.

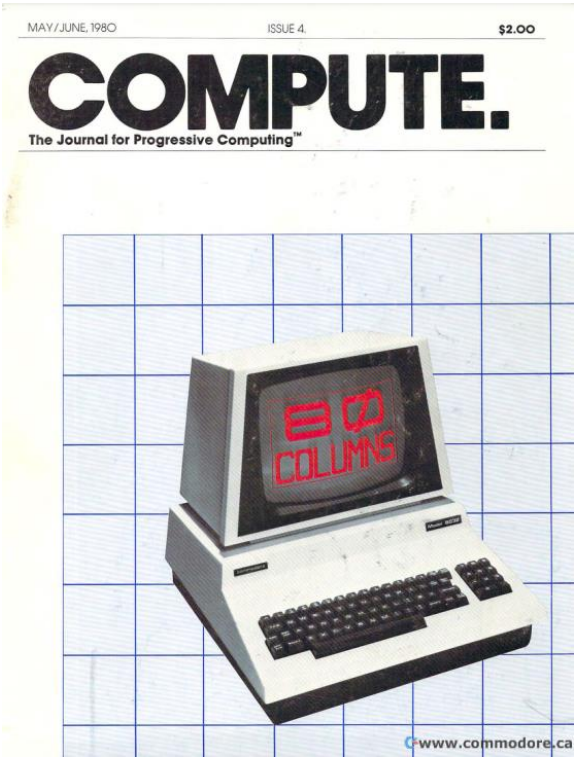

Figura 5 Revista compute (Edición de Mayo-Junio 1980)

Por otro lado, en este periodo de tiempo también se publican libros que defienden la enseñanza mediante el uso de la informática como "***Mindstorms: Children, Computers, and Powerful Ideas***" (1980), como se observa en la Figura 6, por Seymour Papert [13], quien continuó con su proyecto educativo de las décadas anteriores desarrollando sus ideas sobre el aprendizaje mediante la programación y el uso de LOGO como una herramienta educativa. En este libro defiende a la programación como forma alternativa y con potencial para el desarrollo cognitivo de los niños, y aboga por un modelo de aprendizaje que use la interacción persona-máquina como base. En este documento también es donde desarrolla en profundidad el concepto del construccionismo [14],

defendiendo que el aprendizaje es mayor durante el proceso de desarrollo de proyectos creativos más grandes, que van tomando forma a medida que los alumnos van aplicando múltiples conceptos teóricos a tareas prácticas menores que componen el total resultante.

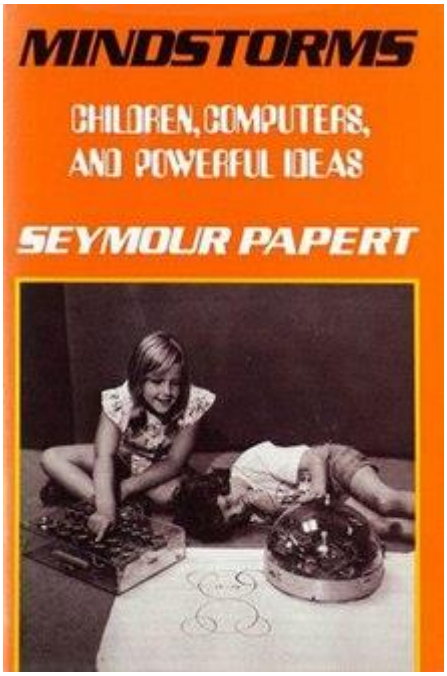

Figura 6 Mindstorms: Children, Computers, and Powerful Ideas (Publicado en 1980 por Harvester Press)

2000-2009

Durante este periodo de tiempo se acentúa la integración de la tecnología y da comienzo una nueva era digital, provocando un incremento en la importancia que tiene la impartición de contenidos sobre nuevas herramientas como la programación y otras tecnologías dentro del itinerario formativo, y su accesibilidad desde edades cada vez más tempranas. Un ejemplo emblemático de estos esfuerzos es Scratch[1], una plataforma desarrollada por el Instituto Tecnológico de Massachusetts (MIT) en 2003. Esta aplicación no solo simplifica el aprendizaje de la programación, sino que también proporciona un entorno visual intuitivo y herramientas interactivas diseñadas específicamente para la audiencia juvenil [15]. Así, Scratch, cuya primera versión la podemos observar en la Figura 7, se erige como una ventana lúdica que abre las puertas

[1] Scratch Jr.: https://www.scratchjr.org/

al mundo de la programación para los alumnos más jóvenes, facilitando la participación y creatividad de estos en este ámbito.

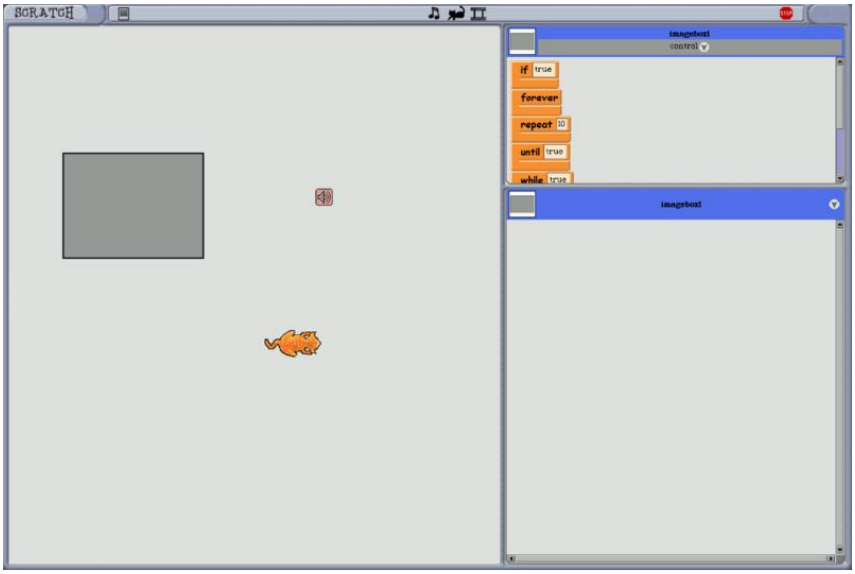

Figura 7 Primera versión de Scratch (Octubre de 2003)

En cuanto al uso de nuevas tecnologías enfocadas en el estudio durante esta década, destacan proyectos como el ***Proyecto OLPC*** (One Laptop Per Child), Figura 8, que consiste en proporcionar a cada niño un ordenador portátil de bajo coste de producción, específicamente diseñado para el uso en entornos educativos en países en desarrollo, y de esta forma, fomentar el acceso a la educación tecnológica [16]. Este modelo de portátil fue el *XO*, que contaba con una pantalla de bajo consumo, conexión innalámbrica y un sistema operativo basado en Linux.

El plan inicial este proyecto se sustentaba en vender estos ordenadores a gobiernos de países en desarrollo en grandes cantidades a un precio significativamente reducido para que pudieran distribuir las computadoras a los niños en edad escolar, no obstante, OLPC tuvo que explorar modelos alternativos más realistas, incluida la venta de dispositivos directamente a consumidores o a través de programas piloto más pequeños [17].

Otros dispositivos aparte de ordenadores portátiles que empiezan a tener más presencia en las aulas fueron las pizarras digitales, introduciendo dinamismo en la

presentación de información por parte de los educadores, estimulando la participación del alumnado y el aprendizaje colaborativo.

Figura 8 Nicholas Negroponte, fundador de 'One Laptop per Child', en el CES 2008 de Las Vegas. (Foto: AP)

No obstante, ninguno de estos elementos descritos anteriormente puede compararse con el hito sociocultural que supone el acceso a Internet en tiempo real, alzándose en ese periodo como la herramienta predilecta para el apoyo en los estudios hasta la actualidad. Foros, páginas web y otras plataformas en línea se vuelven fuentes comunes para la investigación y el aprendizaje autodirigido, permitiendo un acceso rápido y extenso a información educativa.

Algunos exponentes de plataformas educativas en línea creadas durante la década de 2000 son *Moodle*, *Blackboard* o *Khan Academy*, que se basan en la gestión de contenidos educativos provocando un cambio notorio en la interacción entre estudiantes y docentes. Pero internet y sus avances no solo cambiaron la manera en la que profesores interactúan con el alumnado y la accesibilidad y presentación del contenido, también la forma de trabajar, permitiendo desarrollar proyectos de manera colaborativa en tiempo real a distancia, lo que a su vez también derivó en sistemas de aprendizaje alternativos como las clases no presenciales.

A pesar de todos los beneficios que trae internet al ámbito de la enseñanza, también presenta nuevos retos como la brecha digital, la necesidad de proporcionar formación adecuada a los educadores y la revisión de las formas de evaluación de los conocimientos.

2010-Actualidad

Con el auge de los dispositivos móviles y la presencia de internet ya normalizada en la sociedad, han surgido diversas aplicaciones y plataformas interactivas que buscan introducir la programación de manera amena para los niños. Herramientas como ***Code.org***[2] (Figura 9), ***Tynker***[3] y ***Scratch Jr.*** desempeñan un papel fundamental al llevar la programación a un público más extenso, facilitando su integración tanto en entornos educativos estructurados como en ambientes informales. Estas iniciativas consiguen que el aprendizaje de la programación sea accesible y atractivo, transformando la percepción y participación de los niños en este campo de habilidades digitales.

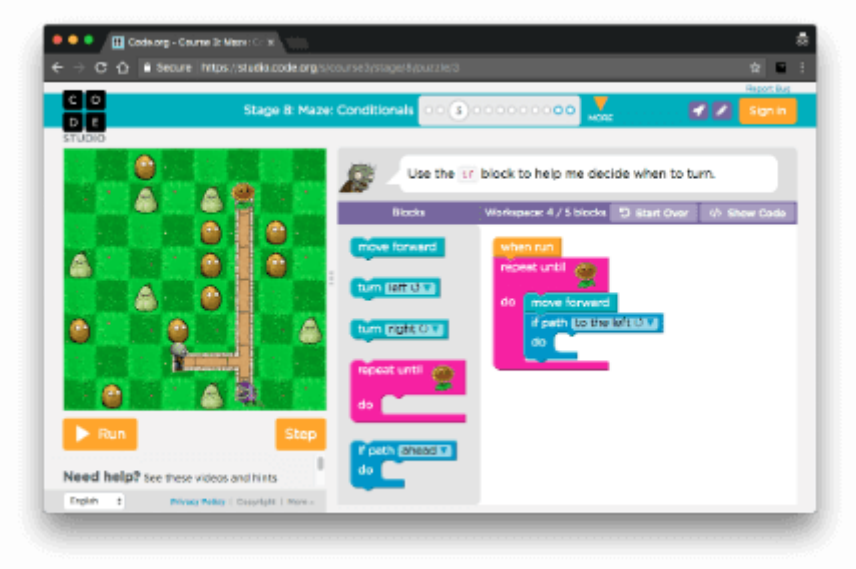

Figura 9 Interfaz de Code.org

En cuanto al uso de nuevas tecnologías en el sector de la educación, son remarcables el desarrollo las plataformas de aprendizaje en línea, heredadas de la década anterior y mejoradas en esta, con nuevos exponentes como *Coursera*[4] o *edX*[5], que aunque están enfocados a un público adulto, optimizan la manera en la que tanto alumnos como profesores consumen información mediante cursos de diversas disciplinas, no solo

[2] Code.org: https://code.org/
[3] Tynker: https://www.tynker.com/
[4] Coursera: https://www.coursera.org/
[5] edX: https://www.edx.org/

ofreciendo contenido de alta calidad, sino también democratizando la educación al hacerla accesible a una audiencia global.

Esto no solo cambia la forma en la que se accede a la información, sino también al ritmo de aprendizaje, con un enfoque más personalizado, según las necesidades individuales de los estudiantes. Ejemplos notables incluyen aplicaciones como *Duo Lingo*[6] (2011), cuyo estilo de interfaz se muestra en la Figura 10, ajustan el contenido y la dificultad de las lecciones según el progreso de cada estudiante, proporcionando una experiencia de aprendizaje más efectiva.

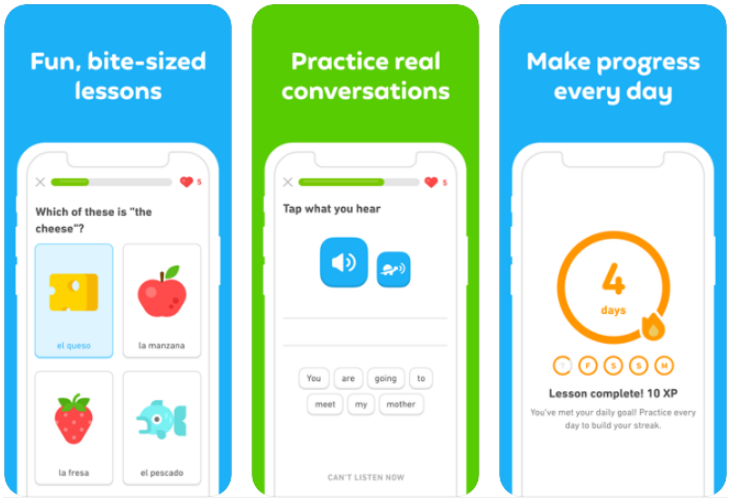

Figura 10 Interfaces de Duolingo

También está actualmente en incremento el interés en el desarrollo de tecnologías relacionadas con la Realidad Virtual, como ***Google Expeditions*** que es capaz de permitir a los estudiantes explorar entornos virtuales [18] , y plataformas de Inteligencia Artificial relacionadas con múltiples aspectos del estudio, como el caso del sistema Squirrel AI [19], que utiliza algoritmos para adaptar el contenido educativo de acuerdo con el estilo de aprendizaje de cada estudiante, mejorando la eficacia del mismo.

Este uso de nuevas tecnologías ha transformado la manera en que los docentes preparan las dinámicas de clase. Un ejemplo prominente de esta transformación es la filosofía del aula invertida (flipped classroom), donde los recursos multimedia en línea

[6] Duolingo: https://es.duolingo.com/

se utilizan para la entrega de contenido en casa, liberando tiempo en clase para actividades prácticas y discusiones [20].

Además del aula invertida, hay otras metodologías innovadoras como el aprendizaje basado en proyectos (Project-Based Learning), que consiste en que los estudiantes adquieran conocimientos y habilidades a través de la elaboración de trabajos que responden a problemas reales [21].

Por otro lado, la metodología del aprendizaje personalizado (Personalized Learning) adapta el contenido y los métodos de enseñanza a las necesidades individuales de cada alumno [22], permitiendo un ritmo de aprendizaje propio y enfocado en los intereses de cada uno.

En cuarto lugar, la metodología de aprendizaje colaborativo apoyado por computadoras (CSCL, por sus siglas en inglés: Computer-Supported Collaborative Learning) fomenta la interacción entre estudiantes mediante plataformas digitales, permitiendo que colaboren en proyectos y tareas de manera simultánea y remota [20]. Este enfoque no solo facilita la adquisición de conocimientos técnicos, sino que también desarrolla habilidades sociales y de trabajo en equipo.

Finalmente, cabe remarcar que el uso de tecnologías avanzadas como la inteligencia artificial (IA) también introduce nuevos retos. Estos incluyen la regulación de derechos de autor, derechos de imagen, y el manejo ético de la información proporcionada por estas tecnologías. Además, plantean nuevas formas de entender la evaluación de los conocimientos y obligan a los docentes a reinventar sus métodos de enseñanza, enfrentándose nuevamente a barreras tecnológicas.

2.2. Revisión de herramientas de programación de bloques

En esta parte de la memoria se analizará tanto el estado actual de las plataformas de enseñanza de la programación de bloques como los beneficios e inconvenientes que presentan sus diseños y la forma en la que imparten el contenido.

Blockly

La Programación Blockly[7] (Figura 11) es una biblioteca orientada a la creación de lenguajes de programación basados en bloques, en esta se basan numerosas plataformas online de aprendizaje de programación como Code.org, Scratch o Tynker, que serán analizadas más adelante. El objetivo de esta es ser una herramienta fácil y rápida para el aprendizaje de Javascript, Lua, Dart, Python, o PHP [1].

La interfaz gráfica de este programa permite a los usuarios programar manipulando bloques y conectándolos para generar código [23], haciéndolo accesible a un mayor número de tipos de estudiantes y también cuenta con una amplia comunidad de desarrolladores y educadores que contribuyen con recursos, ejemplos o foros para mejorar la experiencia de los usuarios.

Los puntos fuertes de esta forma de programar son principalmente que cuentan con interfaces y dinámicas de uso muy intuitivas, que dan acceso a los recursos sin necesidad de descargar ni instalar programas y que se adaptan fácilmente al ritmo de aprendizaje de cada alumno.

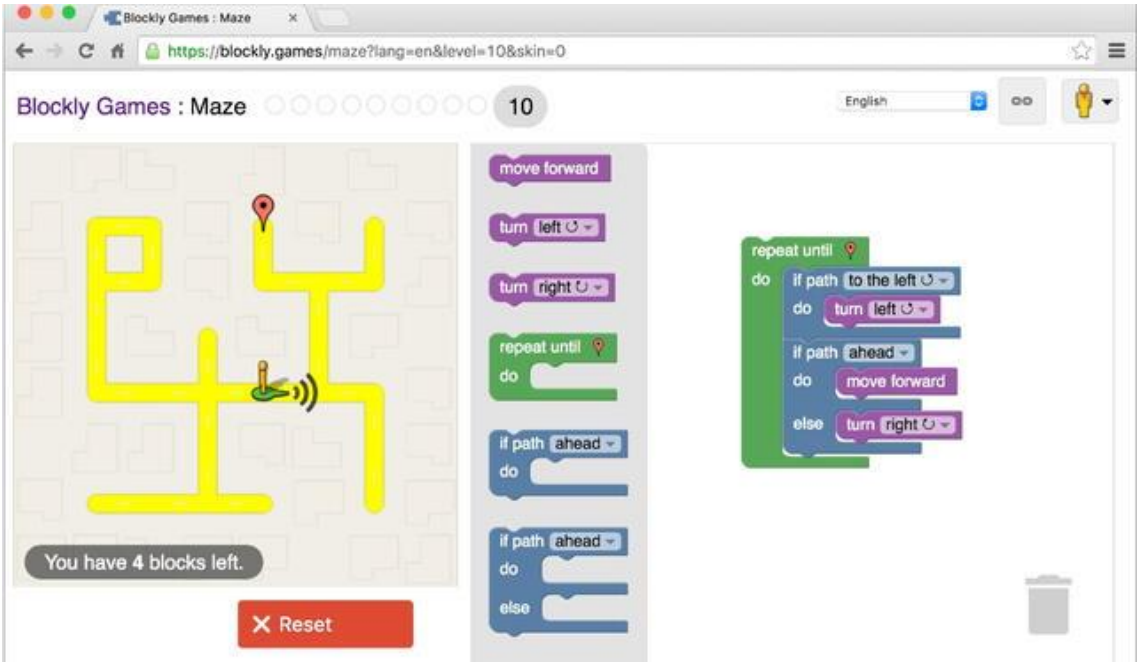

Figura 11 Interfaz de Blockly

[7] Blockly: https://blockly.games/?lang=es

Code.org

Esta es una organización sin ánimo de lucro cuyo proyecto tiene como objetivo hacer accesible conocimientos sobre informática y programación a los alumnos mediante una plataforma educativa accesible a través de su aplicación web [24], por lo que no hay que descargar ni instalar ningún programa, haciendo que su uso e introducción a la plataforma sea más rápida y sencilla para todo tipo de usuarios.

La forma en la que los alumnos aprenden los nuevos conceptos es mediante las acciones de arrastrar y soltar bloques para encadenar instrucciones y construir de esta forma algoritmos que resuelvan los retos que se les proponen [24].

Por ejemplo, la plataforma cuenta con los cursos "Computer Science Fundamentals" que a su vez se organizan por edades y se subdividen en lecciones online (como se puede observar en la Figura 12), presentadas de menor a mayor dificultad con videos introductorios en cada una [25]. Estas lecciones ofrecen ejercicios en los que se busca que el alumno encuentre la mejor solución a cada enunciado planteado, dándole un número máximo de bloques que poder usar, de tal forma que los ejercicios pueden calificarse, permitiendo en todo momento al alumno poder observar su progreso y repetir los ejercicios cuantas veces desee y mejorar sus resultados.

Figura 12 Lección del programa "The Computer Science Fundamentals" para alumnos de primaria

Otro punto para destacar de la plataforma es que se apoya mucho en la gamificación, no solo por el uso de personajes de otras franquicias de juegos para todos los públicos como *Angry Birds*[8] o *Plantas vs. Zombies*[9] para hacer más lúdica las sesiones de estudio,

[8] Angry Birds: https://www.angrybirds.com/
[9] Plantas vs. Zombies: https://www.ea.com/es-es/games/plants-vs-zombies

sino también por su diseño de interfaz, uso de sonidos, efectos visuales y el sistema de correcciones y recompensas.

Tynker

Se trata de una plataforma para enseñar programación de manera estructurada, principalmente Python [26] [27] como se observa en la Figura 13, aunque también cuenta con cursos de CSS y de programación orientada a *Minecraft*. Esta plataforma aborda las necesidades de tanto alumnos, padres y profesores mediante ejercicios que se plantean como acertijos y minijuegos. Al principio estos tienen una dificultad reducida y después cada vez son más difíciles, y fomentan que el estudiante los vaya resolviendo para ir creando historias y avanzar en los niveles.

La herramienta permite adaptar la configuración según las necesidades del educador, por ejemplo, creando aulas o agregando alumnos a estas durante la clase, pero también permite modificar el ritmo de la clase al contar con un selector de lecciones, por lo que no es necesario completarlas en un orden concreto.

Figura 13 Tynker enseñando a programar en el lenguaje Python a través de bloques

Scratch Jr.

Este programa consiste en un entorno virtual diseñado para introducir a niños de educación primaria a muchos de los conceptos de la programación. Se basa en la

creación de código por bloques con un diseño más minimalista y visual que los anteriormente expuestos.

La principal motivación de Scratch Jr.,cuya interfaz se muestra en la Figura 14, es la introducción de la programación en edades muy tempranas intentando sacrificar el menor contenido posible. Este está adaptado a *tablets* principalmente y no necesita navegador web ni conexión a internet para su uso [28], lo cual se puede considerar un punto a favor por no presentar esas limitaciones y facilita que no se distraigan mientras aprenden. También cuenta con guías para ayudar a la navegación por pantallas, otra que explica el funcionamiento de los bloques con los que se construyen los programas y otra para el editor de imágenes de personajes y escenarios de la galería.

Figura 14 Interfaz de Scratcj Jr.

Una de las ventajas, pero que también puede llegar a ser un inconveniente de esta herramienta es el filtro de contenido de programación. Al simplificar tanto el sistema de bloques para hacerlo accesible a más edades, se pueden echar en falta conceptos como las variables, por ejemplo, puesto lo que se programa en una escena no puede reusarse en otras con una llamada o referencia, teniendo que volver a programar el mismo actor en cada escena que se quiera.

Otro punto positivo de Scratch Jr. es que, al presentarse como un primer contacto de la programación en niños, una vez dominada la herramienta puede dar paso más

fácilmente a un aprendizaje un poco más avanzado en otras plataformas como Scratch[10], Greenfoot[11] o App Inventor[12] como se describe en la referencia [28], ofreciendo mayor sensación de progreso y evitando la sensación de frustración al enfrentarse a nuevos sistemas que pueden ser demasiado complejos la primera vez que se usan.

Kodable

Kodable[13] es una aplicación educativa que ofrece múltiples planes de estudio para enseñar programación a estudiantes de primaria. Estos planes se apoyan en un incremento paulatino en la complejidad de los conocimientos que se trabajan, empezando por los temas fundamentales para las etapas de preescolar y los primeros cursos de primaria, y más tarde enseñando a leer e interpretar código a alumnos de cursos más avanzados.

La interfaz de este programa se basa en las acciones de arrastrar y soltar (Figura 15), lo que la hacer accesible para un mayor rango de edades; y su principal dinámica es la de dar instrucciones de movimiento a los personajes mediante icono de flechas para que cumplan un recorrido determinado, aprendiendo sobre secuenciación, bucles y condiciones, aunque también puede llegar a ser limitante a la hora de explorar otro tipo de instrucciones o desarrollar actividades variadas [2].

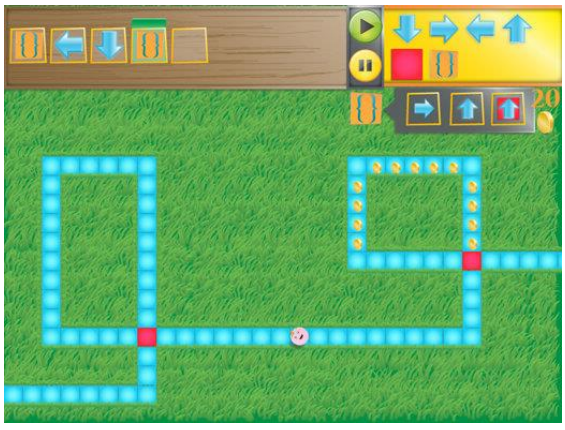

Figura 15 Interfaz de Kodable

[10] Scratch: https://scratch.mit.edu/
[11] Greenfoot: https://www.greenfoot.org/door
[12] App Inventor: https://appinventor.mit.edu/
[13] Kodable: https://www.kodable.com/

LightBot

Este juego se basa en la resolución de rompecabezas a través de los cuales se van impartiendo conceptos de programación un poco más complejos como bucles, recursiones, procesos o patrones a través de iconos visuales como flechas o dibujos, por lo que lo hace apto para estudiantes con edades un poco más altas.

Cada partida consiste en guiar a un robot, como se puede ver en la Figura 16, pero para ello el alumno primero debe pre-planificar qué tipo de algoritmo va a necesitar, el orden de secuencia de instrucciones correcto, si tiene espacio suficiente en el bloque principal o necesitará crear procedimientos a los que ir llamando durante la ejecución, probar y depurar el código con las herramientas de "debug", etc. [29]

Esta plataforma ofrece una experiencia más enriquecida que otras, no obstante, esto también perjudica la accesibilidad a aquellos alumnos que por edad no son capaces todavía de asimilar conceptos demasiado complejos como la recursividad dentro de un bucle.

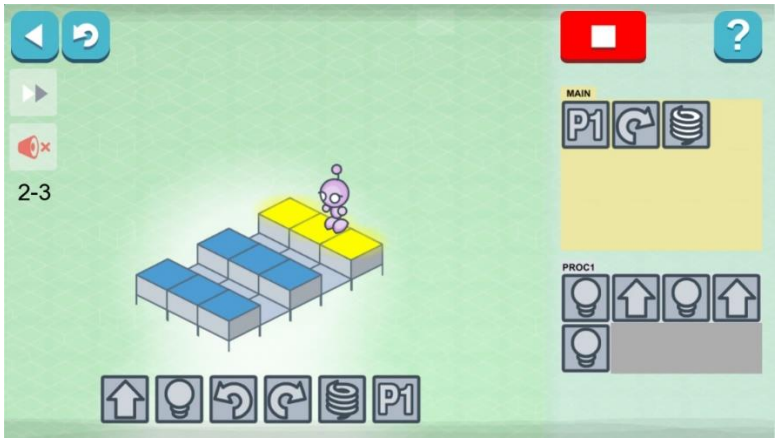

Figura 16 Interfaz de LightBot

2.3. Comparación de herramientas

Herramientas analizadas	Plataforma	Pros	Contras
Blockly	Web	Interfaz intuitiva, de código abierto, amplia comunidad y recursos educativos.	Puede ser más compleja para principiantes en comparación con otras plataformas.
Code.org	Web	Gran variedad de cursos y actividades, gamificación del aprendizaje, recursos para docentes, accesible desde cualquier dispositivo.	Puede ser repetitivo para usuarios avanzados. Además, algunas actividades requieren conocimientos previos.
Tynker	Web, iOS, Android	Permite transiciones de lenguaje de bloques a texto, también cuenta con módulos orientados a programación de hardware (robots, drones, etc.)	Requiere de suscripción para acceder a todo el contenido y algunas funciones avanzadas solo están disponibles en versiones de pago.
Scratch Jr.	iOS, Android	Interfaz muy sencilla y amigable, filosofía minimalista y muy visual. Fácil de entender para alumnos pequeños.	Permite solo comandos muy básicos y no permite transiciones a lenguajes de texto.
Kodable	Web, iOS	Es adecuado para edades tempranas, basa su aprendizaje en juegos. Se centra en ir mejorando la habilidad para resolver problemas a medida que avanzas en los niveles.	No permite crear código muy complejo y requiere de una suscripción para acceso completo.
LightBot	Web, iOS, Android	Está muy enfocado en aprender lógica de algoritmos para resolver los problemas y su interfaz es intuitiva para principiantes.	Los niveles son limitados y no enseña lenguajes de programación tradicionales.

2.4. Experiencias de uso con pantallas táctiles

Dado que la programación consiste en construir algoritmos a través de instrucciones y comandos dentro de un sistema o software, es importante tener en cuenta como se representan estas instrucciones, prestando especial atención a cómo pueden manipularse de tal forma que sean comprensibles para niños, ya que de la forma que se les presente estas tareas puede influenciar en su tolerancia a la frustración, su entendimiento de las herramientas que disponen y en definitiva, que el aprendizaje resulte exitoso y lúdico.

Como se explica en el estudio [2] sobre el aprendizaje de la programación en niños a temprana edad, aquellos que dedican tiempo a plataformas como Logo son capaces de comunicación y comprensión mejor con su entorno. Sin embargo, hay que tener en cuenta que muchos de estos niños están en edades en las que todavía no han aprendido a leer y escribir, incluso aprender a controlar con precisión un ratón de ordenador o entender el funcionamiento de un PC a edades muy tempranas puede ser un reto en sí. Es por este motivo que se llega a la conclusión que las aplicaciones basadas en pantallas táctiles pueden ser una solución debido a que son más fáciles de interactuar, y también admiten mejor un lenguaje basado en dibujos, iconos y símbolos simples más rápidos de comprender.

A continuación en los siguientes apartados se analizarán hitos concretos del uso de mesas y pantallas interactivas en un contexto educativo.

Caso de estudio de code.org

Este estudio [30] se basa en la búsqueda de mejores formas de integrar la tecnología en las actividades escolares, en este caso, a través del uso de pantallas táctiles como las de las tabletas, junto con aplicaciones apropiadas, es posible superar en cuanto a calidad de aprendizaje a otras metodologías ya establecidas.

Los conocimientos impartidos a los alumnos de este estudio estaban relacionados con la programación y el pensamiento computacional mediante dinámicas de gamificación, basándose en principios del conductismo, fomentando la interacción grupal y la participación en clase, por lo que no solo aprendieron y desarrollaron conceptos relacionados con la lógica y las matemáticas, sino que fomentaron la

comunicación grupal, el trabajo en equipo y el sentimiento de aprender jugando para los alumnos del aula.

Para contrastar y reforzar estos resultados, se hicieron pruebas en tres tipos de grupos de alumnos: los primeros siguieron el esquema de enseñanza habitual y estandarizado en la actualidad, el segundo grupo se le propusieron dinámicas de gamificación a través de juegos de mesa, y los últimos tuvieron acceso a las tabletas en parejas o grupos. El segundo grupo obtuvo mejores resultados en cuanto a la demostración de conocimientos adquiridos durante las sesiones, y a su vez, los del último grupo también tuvieron mejores resultados que los del grupo 2.

Finalmente, cabe resaltar que, aunque los resultados de este estudio [30] son muy positivos, no se puede asegurar con plena fiabilidad que sean replicables en otro tipo de contextos o alumnos, ya que la muestra fue relativamente pequeña y no se hizo en diferentes partes del mundo donde la cultura o el acceso a las tecnologías pueden ser factores importantes a la hora de obtener ciertos resultados.

Caso de estudio "The future of teaching programming is on mobile devices"

Este estudio [31], apoyado en el cambio tecnológico que está experimentando la forma en la que las personas se relacionan, consumen y crean contenido, y donde los dispositivos móviles basados en pantalla táctil, como smartphones y tabletas, superarán en ventas a los ordenadores convencionales, plantea nuevas oportunidades para la enseñanza de la programación.

A raíz de esta premisa, se recurre al entorno de aprendizaje virtual "TouchDevelop", enfocado a que los estudiantes desarrollen aplicaciones en sus *smartphones*. Este sistema presenta ventajas significativas, especialmente para estudiantes de cursos más avanzados, como secundaria o bachillerato. En este caso, durante las fases de pruebas, los estudiantes estuvieron expuestos a la aplicación durante varios plazos para que pudiesen desarrollar proyectos y ver la efectividad de esta metodología en programas a corto y largo plazo, probando con plazos de entre 9 meses, como sesiones aisladas de 90 minutos.

Las conclusiones de esta investigación fueron que, al contar los jóvenes con dispositivos móviles en su día a día y por la practicidad de estos, se incrementaron los resultados a la hora de asimilar conocimientos, por su mejor accesibilidad a través del *smartphone*.

Caso de estudio "Multi-touch Display Technology and Collaborative Learning Tasks"

Este estudio [3] trata sobre la tecnología de pantallas multi-touch y su aplicación en tareas colaborativas de aprendizaje en el aula, enfocandose en el diseño de software para el aprendizaje colaborativo a través de tecnología que fomente la interacción entre alumnos.

Durante las pruebas se usa una aplicación de alfabetización fonética, no obstante, estas dinámicas educativas pueden aplicarse a otros tipos de campos.

En el artículo se discuten cuestiones de diseño, como el análisis de tareas de aprendizaje, la descomposición de tareas y asignación de roles, además de mecanismos de comunicación colaborativa y problemas para diseñar y delimitar la interfaz para múltiples personas. También menciona posibles mejoras en la plataforma de hardware y software que se han implementado en la pantalla, por ejemplo con las actualizaciones en el software de código abierto Community Core Vision (CCV).

Como conclusión de este estudio se reitera en el potencial de las pantallas táctiles multitouch para mejorar la colaboración en el aprendizaje, con un enfoque específico en el diseño de software, ejemplificando esto en una aplicación de alfabetización fonética.

3. ESPECIFICAIÓN DE REQUISITOS Y ANÁLISIS

En este capítulo se estudiará la planificación y se desglosará la organización de la herramienta en los módulos necesarios para alcanzar los objetivos pedagógicos, de funcionalidades y del proyecto planteados anteriormente. Para ello se desarrollará una metodología que sirva de guía durante el proceso de diseño y desarrollo del sistema, se especificarán los requisitos funcionales y no funcionales y se profundizará en los módulos funcionales mencionados.

3.1. Metodología

Una vez planteados los objetivos del proyecto en el capítulo 1, se procederá a definir la metodología de trabajo. En este caso se planteará como varias etapas que consistirán en planificación, diseño e implementación para evitar en mayor medida de lo posible el rehacer elementos del prototipo o corregir errores no previstos, como se observa en la Figura 17 Diagrama de Grantt del desarrollo del proyecto.

En primer lugar, se analizarán los resultados del estudio de las múltiples herramientas, sistemas de aprendizaje y aplicaciones orientadas a la enseñanza de la programación desarrolladas en el Capítulo 2.

Tras esto se estudiará como diseñar una aplicación con una interfaz intuitiva y atractiva, adaptada a la mesa táctil, manteniendo en todo momento que sea fácil de entender y de utilizar por los niños, mediante el uso de personajes o elementos visuales que atraigan la atención y generen interés.

Con el prototipo de la aplicación montado, dará comienzo el análisis de interactividad y contenidos, para perfeccionar detalles y realizar modificaciones convenientes para mejorar la experiencia de aprendizaje de los alumnos.

La intención de la aplicación desarrollada es que esté lista para futuras pruebas prácticas con un grupo diverso de alumnos.

Etapas	Fecha de inicio	Duración en días	Fecha fin
Investigación	01-ene	15	16-ene
Diseño del prototipo	16-ene	68	24-mar
Memoria del proyecto	01-ene	145	25-may
Análisis del prototipo	15-feb	9	24-feb
Correcciones	01-mar	30	31-mar
Desarrollo de la aplicación	07-mar	61	07-may
Encuesta y Conclusiones	05-may	15	20-may

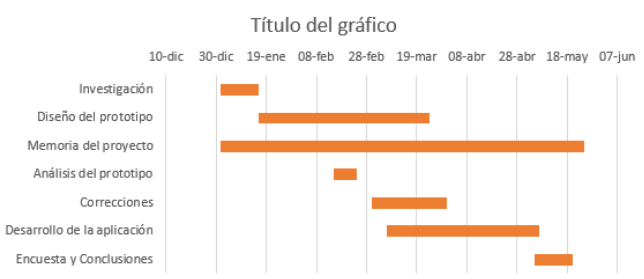

Figura 17 Diagrama de Grantt del desarrollo del proyecto

3.2. Especificación de requisitos

A continuación se describen los requisitos funcionales y no funcionales que debe cumplir la aplicación para cumplir con los objetivos propuestos.

Funcionales

Los requisitos funcionales se centran en definir lo que el sistema debe hacer, y sus objetivos tienen el propósito de cumplir con las necesidades del usuario, por lo que afectan principalmente a la lógica del diseño e implementación del sistema.

Categoría	**Requisito**
Multiusuario	El sistema debe permitir sesiones de entre 1 y 4 jugadores.
Manipulación de bloques	Los usuarios deben poder unir, desunir, configurar y mover bloques de código.
Ejecución de Código	El sistema podrá interpretar bloques o grupos de bloques como código cuando lo solicite el usuario.
Delimitar zonas por categorías	El sistema identificará zonas de trabajo propias para cada alumno, adaptando los bloques a su posición, y una zona común donde se podrá ejecutar el código.
Elección de bloques	El usuario tiene la opción de elegir cualquier tipo de bloque en todo momento.
Restaurar la ejecución	La aplicación da la opción de volver a un estado anterior a la última ejecución del código.
Cambiar número de usuarios	Se podrá cambiar la interfaz para adaptarla a un nuevo número de usuarios en cualquier momento.

Cambiar aspectos	Se podrá alterar el aspecto de los personajes que se programen y del escenario donde se ejecuten los bloques.
Sistema de coordinación entre usuarios	Para mantener cierto orden, el sistema deberá establecer una funcionalidad que arbitre lo que propone cada usuario a la zona de ejecución, y se encargará de controlar que se ejecuten solo los programas que han sido aceptados por el resto de los alumnos de la sesión.
Eliminar programas	Un usuario podrá eliminar programas que ya no desee en cualquier momento.
Programar varias secuencias a la vez	El sistema permitirá que los usuarios puedan programar múltiples secuencias de bloques independientes al mismo tiempo y proponer todas las que quiera.
Compartir programas	Los usuarios podrán pasarse los unos a los otros los programas que hayan diseñado, pudiendo manipular bloques que han sido creados por otros compañeros en su zona de trabajo individual.

No funcionales

Los requisitos no funcionales, por otro lado, están más enfocados en el comportamiento del propio sistema. Estos deben garantizar aspectos como el rendimiento, la seguridad y su usabilidad, entre otros. Los requisitos que se describen en este apartado influirán en decisiones como el motor gráfico, plataforma y especificaciones técnicas de la herramienta.

Categoría	**Requisito**
Detección táctil	Al pulsar la pantalla con un dedo se detectará como si se hubiera pinchado con el ratón. De la misma forma se puede pulsar y arrastrar.
Compatibilidad táctil con múltiples usuarios	El sistema debe permitir y diferenciar cuando más de un usuario interactúa con la pantalla, permitiendo que cada uno pueda pulsar o pulsar y arrastrar libremente y a la vez si es necesario.

Reproducción de efectos de sonido	La aplicación podrá reproducir sonidos ya sea mediante el bloque de sonido como al interactuar con ciertos elementos de la interfaz.
Representación de los programas	El sistema cambiará el aspecto del código programado por los usuarios dependiendo de si este se encuentra en las zonas de trabajo individual o si se encuentra en la zona de ejecución común.
Rendimiento	El programa no deberá bloquearse ni relentizarse cuando se interactúe con él.
Interfaz	La interfaz debe ser consistente y clara en todo momento.
Interactuabilidad	Todos los elementos del sistema deben ser interactuables a través de las acciones pulsar o pulsar y arrastrar.

3.3. Análisis funcional

Teniendo en cuenta los requisitos expuestos en el apartado anterior, a continuación profundizaremos en los módulos funcionales que conforman la estructura software de la aplicación educativa y cómo se relacionan entre ellos.

En el siguiente índice podemos ver cuáles son las funciones de cada módulo de manera más simple:

1. Gestor de escenas y personajes:
    a. Modificación del aspecto de escena y los personajes.
    b. Identificación del personaje seleccionado por cada jugador y de la escena seleccionada.
2. Máquina virtual:
    a. Generador de bloques.
    b. Sistema de enlaces de bloques.
3. Adaptador del espacio al número de alumnos:
    a. Representación gráfica de los bloques según la zona.
    b. Detección de zonas.
4. Visualizador de ejecución:
    a. Identificación y ejecución de comandos.

    b. Recuperar los estados pre-ejecución.
5. Gestor de colaboración:
    a. Sistema de notificaciones.
    b. Detección de permisos.

Todos estos módulos se comunican entre sí, por lo que ninguno tiene completa independencia. Por ejemplo, el módulo del visualizador de ejecución primero comprueba que el módulo de colaboración detecta que todos los permisos están en orden para poder ejecutar los comandos.

A continuación definiremos en más profundidad estos módulos y cómo interactúan sus funcionalidades:

- Gestor de escenas y personajes: se encarga de identificar el personaje que tiene cada jugador seleccionado para que cuando estos lo muevan a la zona de ejecución tengan una representación gráfica y un efecto de sonido asociado correctos. También se encarga de detectar modificaciones en el selector de los fondos de las escenas.
- Máquina virtual: es el módulo donde se agrupan las funciones de los bloques (creación, enlaces, desenlaces) además de ser quien asigna a cada bloque la información relevante para otros módulos, como qué tipo de bloque es, su posición y su orden dentro de una secuencia (entre otros).
- Adaptador del espacio al número de alumnos: modifica el número de zonas y la disposición y tamaño de estas según el número de alumnos seleccionado durante la sesión. También comunica esta información a otros bloques como el de colaboración, para saber cuántos permisos se necesita y cuántos mensajes mostrar.
- Visualizador de ejecución: se encarga de lo relacionado con la zona común de ejecución y las acciones de ejecutar comandos y restaurar programa. Identifica e interpreta la información de los bloques, qué deben hacer y almacena la información valiosa como los estados previos y post-ejecución.

- Gestor de colaboración: mantiene el orden en la sesión mediante un sistema de notificaciones y permisos que comunican a otros módulos qué deben permitir y qué no para cada jugador en todo momento.

Finalmente, los scripts creados para contener estas funcionalidades se relacionan como se puede ver en el diagrama de la **¡Error! No se encuentra el origen de la r eferencia.**, aunque el código se explicará en mayor profundidad más adelante durante el capítulo 4:

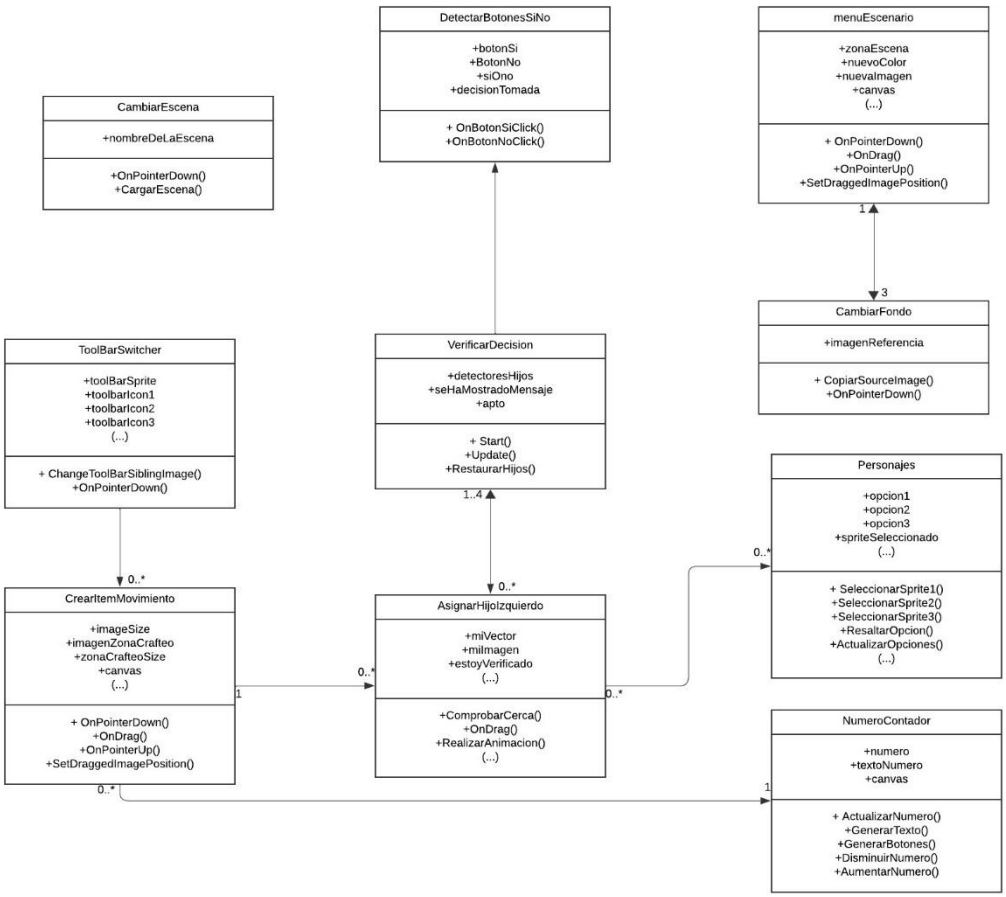

Figura 18 Diagrama de clases UML

4. DISEÑO DE LA INTERFAZ E IMPLEMENTACIÓN

Durante este capítulo se relatará el proceso de diseño de un prototipo de aplicación multiusuario, orientado a la enseñanza temprana de conceptos fundamentales de la programación, basado en el entorno de programación visual Scratch Jr. La singularidad de este proyecto reside en su adaptación para ser implementado en una pantalla táctil de dimensiones considerables, con el fin de que pueda ser usada en sesiones educativas para grupos de entre 1 y 4 alumnos simultáneos.

La elección de **Scratch Jr.** como la base para la implementación de este prototipo se fundamenta en su enfoque simple y accesible para usuarios no familiarizados con la programación. La interfaz gráfica compuesta por iconos simples y el lenguaje basado en bloques facilitan que los usuarios, sobre todo aquellos en sus primeras etapas de aprendizaje, puedan construir y comprender conceptos de programación de manera lúdica.

Durante esta fase del desarrollo se tuvo en mente combinar la versatilidad de Scratch Jr. con la adaptación y cooperación durante el uso de dicho hardware, para potencia la experiencia de aprendizaje y fomenta la participación y dinamismo durante las clases.

4.1. Fase de diseño

Esta fase de diseño del prototipo se llevó a cabo usando la herramienta **Figma**[14], como se muestra en la Figura 19, que está orientada al prototipado de aplicaciones en múltiples plataformas con la capacidad de generar interfaces con animaciones de alta fidelidad para poder previsualizar la aplicación real que se desarrollará más a delante. La elección de esta herramienta se basa en su capacidad para agilizar el proceso de diseño iterativo, posibilitando la retroalimentación continua y una visualización rápida de los resultados a medida que se va diseñando los diferentes flujos de la aplicación.

[14] Figma: https://figmsbusiness.com/

Figura 19 Conjunto de ítems diseñados en Figma para el prototipo visual de la aplicación

El flujo de pantallas comienza con aquellas que introducen al usuario a que pueda registrarse o introducir sus credenciales si ya está registrado. Esta funcionalidad se diseñó en un principio para contemplar la opción de poder tener en versiones futuras del desarrollo la aplicación subida en alguna plataforma para poder acceder a ella *online*.

Una vez pasadas estas pantallas iniciales, el usuario llegaría a la pantalla "home" donde puede seleccionar de cuantos alumnos va a hacer la sesión (Figura 20), con un máximo de 4 y un mínimo de 1 alumno, como se puede observar en la siguiente figura:

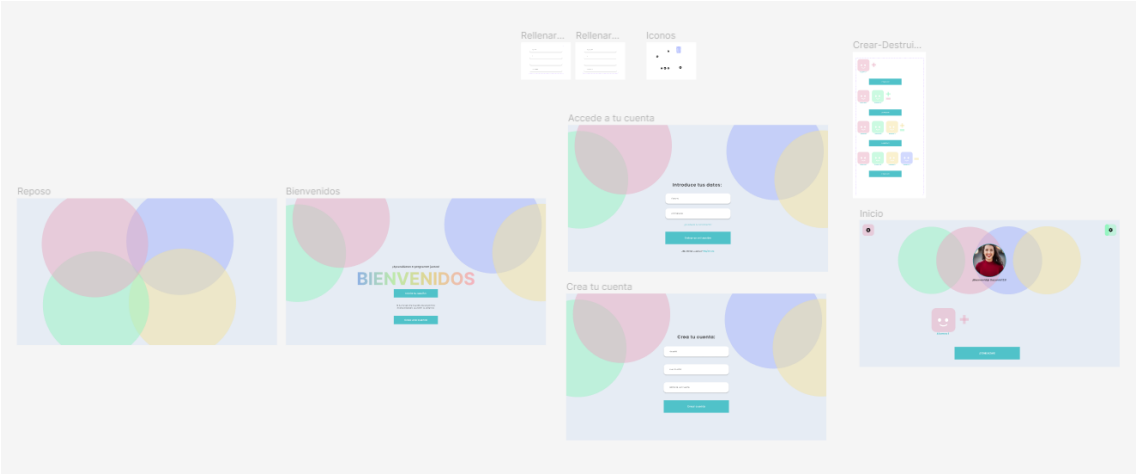

Figura 20 Pantallas iniciales previas a la pantalla de juego

A continuación, como se muestra la Figura 21, se diseñaron en el prototipo múltiples opciones interactivas, como la capacidad de cambiar de escenario, personaje y una previsualización de cómo usaría el alumno la aplicación para programar, compartir sus

creaciones y comunicarse con sus compañeros durante las sesiones. También se prepararon el comportamiento de los menús secundarios y las animaciones necesarias para las rutinas de uso durante la demostración.

Figura 21 Diseños interactivos para una mejor demostración de la funcionalidad de los elementos de cada pantalla

También se recreó un itinerario simulando cómo el usuario podría interactuar con el menú de bloques y como se mostraría la combinación de estos. Para ello se tuvo que desarrollar una actividad guionizada que se dividía en cuatro secuencias (Figura 22): primero uno de los usuarios creaba un programa de movimiento y lo proponía, después otro usuario lo llevaba a su zona para editar el programa propuesto y volverlo a proponer a la zona común. Esta se repitió varias veces para que se mostrase cómo podían interactuar cada uno de los usuarios.

Figura 22 Prototipo visual y funcional de posibles combinaciones de bloques

Finalmente, el itinerario de la simulación del funcionamiento de los bloques y otras animaciones previamente explicadas se introdujeron en la secuencia de uso que se usaría como demo de alta fidelidad de la aplicación:

Figura 23 Prototipo de uso de la aplicación mediante un guión de escenas que enseña la interactuabilidad de la futura aplicación final

4.2.Resultado de la fase de diseño

Finalmente, tras probar el funcionamiento del prototipo e iterando en pruebas prácticas de uso que se iban realizando en la mesa, se obtuvo la siguiente interfaz resultante:

Figura 24 Elementos de la interfaz principal indexados

En este mapa de interacciones de la interfaz se representa todos los elementos que la conforman y que participan en la experiencia de usuario:

1. Botón home: para cerrar la sesión y volver a la pantalla de bienvenida.
2. Menú para fondos de escenarios extra.
3. Icono de restaurar la escena a su estado de antes de haber ejecutado un programa.
4. Ejecutar programas en la escena de la zona común.
5. Menú rápido de fondos de escenarios.
6. Escena central de ejecución: zona común para los jugadores.
7. Iconos para desechar programas.
8. Selector de personajes.
9. Icono para abrir el menú de personajes extra.
10. Indicador de número de jugadores e icono para modificarlo.
11. Menú de tipos de bloques para programar.
12. Iconos que generan bloques para programar.
13. Zona individual de cada jugador donde pueden combinar bloques y editar programas.

Analizando más profundamente en las funcionalidades de la interfaz, dentro del elemento 11 de la lista anterior, encontramos estos 6 elementos, que consiste en un menú de los tipos de bloques con los que los alumnos pueden programar y que están basados en los de *Scratch Jr*:

Figura 25 Selector de tipos de Iconos

- Botón Amarillo: iniciador del programa, contiene un único bloque que indica que la secuencia de bloques empieza:
- Botón Azul: bloques de movimiento.
- Botón Morado: bloques de aumento o disminución de tamaño.
- Botón Verde: contiene el bloque para generar sonido.
- Botón Naranja: contiene el bloque de pausa temporal.
- Botón Rojo: contiene el bloque para cerrar la secuencia de ejecución.

Al pulsar los iconos anteriores, el usuario podrá visualizar un submenú con los bloques disponibles de cada tipo que están implementados en el prototipo funcional.

Bloque Amarillo

Figura 26 Bloque de inicio de ejecución

Este bloque le indica a la aplicación que debe iniciar la ejecución del programa cuando se pulse sobre el icono de la bandera verde, situada en la zona superior-derecha justo encima del escenario común. Si un jugador propone un programa pero olvida colocar este bloque al comienzo de este, no se realizará ninguna acción propuesta.

Bloques Azules

Figura 27 Selector de bloques de movimiento

Este grupo de bloques indican que debe ejecutarse un movimiento para cambiar la posición del personaje propuesto dentro de la escena. Cada uno se ejecuta individualmente, por lo que al combinarlos producirán un movimiento detrás de otro, pero nunca combinar sus direcciones en un solo movimiento.

Además, este tipo de bloques cuentan con la peculiaridad de que contienen indicadores numéricos en la zona inferior del icono. Estos representan la cantidad de desplazamiento que se quiera cuando se ejecute dicho bloque en cuestión, siendo posible manipularlos con los botones auxiliares que acompañan al indicador (botones "+" y "- ").

Figura 28 Multiplicador numérico en un bloque de movimiento a la derecha

Bloques Morados

Figura 29 Selector de bloques de cambio de tamaño

Los bloques morados tienen la funcionalidad de modificar el tamaño del personaje propuesto en escena. Cada bloque puede aumentar, disminuir o restaurar las dimensiones.

En adición, los bloques de aumento y disminución de tamaño también pueden editarse con el mismo tipo de indicadores previamente explicados en los bloques de movimiento.

Figura 30 Multiplicador numérico en un bloque de reducción de tamaño

Bloque Verde

Figura 31 Selector de bloque de reproducción de sonido

El bloque verde reproduce un sonido cuando es ejecutado. Este sonido varía dependiendo del personaje seleccionado, por lo que existe un total de seis sonidos posibles.

Bloque Naranja

Figura 32 Selector de bloque de pausa temporal

Este bloque tiene la utilidad de realizar una pausa temporal cuando se ejecute en escena. Por ejemplo, podría usarse entre un bloque azul y uno verde para que, tras moverse el personaje, dejase un margen de tiempo en el que se para, y una vez que ha transcurrido un tiempo pausado, reproduzca el sonido.

El bloque naranja, de la misma forma que los azules y morados, también cuenta con el indicador numérico que permite manipular la cantidad de tiempo que va a ocupar la pausa.

Figura 33 Multiplicador numérico en un bloque de pausa temporal

Bloque Rojo

Figura 34 Selector de bloque de finalización de ejecución

El bloque rojo es el encargado de indicarle al programa en ejecución que ya ha finalizado su ejecución. Al igual que el amarillo, es necesario incluirlo siempre, pues de lo contrario cuando se desee ejecutar el programa el sistema no reproducirá ninguna de las otras animaciones.

4.2. Dinámica de uso de la aplicación

En esta sección explicaremos brevemente a través de un ejemplo práctico cómo usar la aplicación, su flujo de pantallas y funciones, y cómo generar y ejecutar un programa sencillo. En el Anexo de este documento se encuentra una versión más extendida con más material gráfico sobre el uso de la interfaz.

En primer lugar, al abrir la aplicación, nos aparecerá el salvapantallas de inicio, al cual deberemos darle un toque para que se muestre el primer menú.

A continuación veremos la pantalla donde podremos elegir inicialmente cuantos jugadores o alumnos quieren usar la aplicación. Para ello podremos pulsar en los iconos “+” o “-” que van apareciendo a medida que añadimos o quitamos jugadores.

Una vez estamos conformes con el número de jugadores, debemos pulsar el botón en el que está escrita la frase “¡Comenzar!”. Si no estamos seguros de cuantos jugadores van a participar, o si se quiere modificar este factor más adelante, es posible hacerlo en las siguientes pantallas sin necesidad de volver a esta.

¿Cómo programar en el prototipo?

Una vez hecho los pasos anteriores, ya podremos visualizar la interfaz principal donde manipular los bloques y generar programas. Es este subapartado se explicará cómo interactuar correctamente con esta pantalla para poder ejecutarlos.

Primero podríamos seleccionar qué tipo de bloques queremos generar pulsando en los botones de la mitad izquierda de la barra de herramientas.

Al pinchar en uno de los bloques disponibles de cada sección se generará en el centro de la escena de trabajo, disponible para poder pulsarlo y arrastrarlo a donde se desee.

Para manipular el bloque que acabamos de generar, solo necesitamos pulsar encima de él, y manteniéndolo pulsado, arrastrarlo a donde deseemos. Por ejemplo, para borrar el bloque que acabamos de generar deberíamos arrastrarlo al icono de la basura, tal y como se muestra en la Figura 35 Mover un bloque por la pantalla.

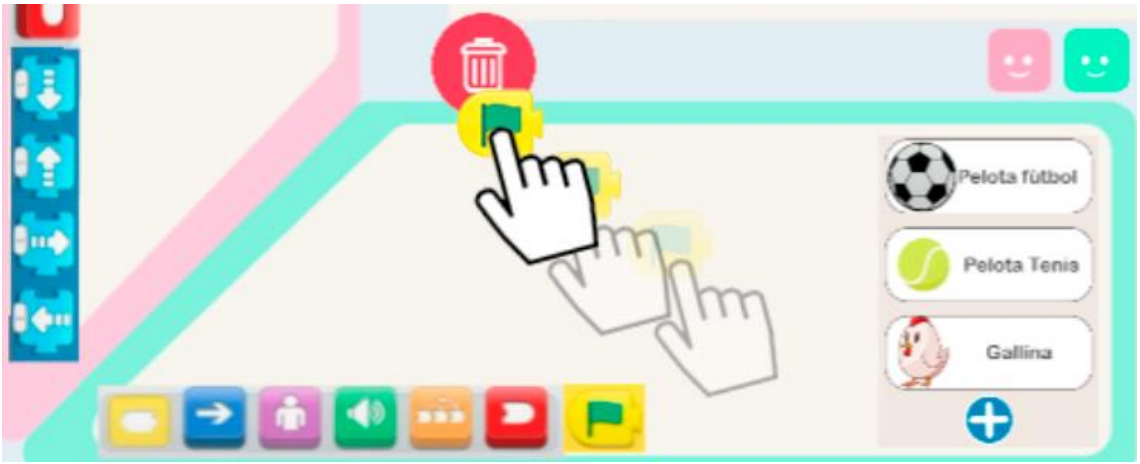

Figura 35 Mover un bloque por la pantalla

Si se quiere encadenar bloques, se deben generar y unir siguiendo el orden que se desee de izquierda a derecha. Por ejemplo, para una secuencia de movimientos simple debemos ir generando los bloques de inicio de ejecución (amarillo), los de movimiento (azules) y el de fin de ejecución (rojo). Pero primero debemos entender como conectar y desconectar los bloques de una secuencia.

Para conectar dos bloques solo deberemos acercarlos y cuando estén lo suficientemente cerca detectarán que quieres unirlos y, en consecuencia, se conectarán.

Ahora podemos mover la secuencia como si fuese una sola unidad o separarla. Para mover los bloques manteniéndolos unidos debemos arrastrar el bloque de mayor prioridad, en el ejemplo de la siguiente figura sería el amarillo. Es decir que si queremos

mover una secuencia sin romper una conexión debemos mover el bloque que se encuentre por el lado izquierdo de la conexión.

Figura 36 Cómo desplazar bloques enlazados

Si por el contrario queremos mover un bloque para romper una conexión concreta, debemos arrastrar el bloque de menor prioridad, es decir, el que se encuentra por al lado derecho de la conexión.

Dependiendo de los bloques que usemos, tendremos la opción de manipular la cantidad de efecto que queremos aplicar. Por ejemplo, en este caso podemos definir cuánto queremos que se desplace hacia la derecha pulsando en los símbolos de “+” o “-” en la zona inferior del bloque.

Una vez tenemos el programa que queremos, también podemos seleccionar el personaje que se desee en el momento, ya sea de los que están disponibles a simple vista como de los que están en el menú extendido con todos los personajes.

Para poner a prueba el programa que hemos creado debemos arrastrar la secuencia de bloques a la zona deseada del escenario y pulsar en la bandera verde. Al entrar en contacto con la escena, la secuencia de bloques se convertirá en el personaje preseleccionado.

Figura 37 Añadir un programa a la escena

Al colocar el personaje en el escenario, nuestros compañeros deberán seleccionar que están de acuerdo con nuestra propuesta, de lo contrario el programa creado volverá a nuestra zona individual.

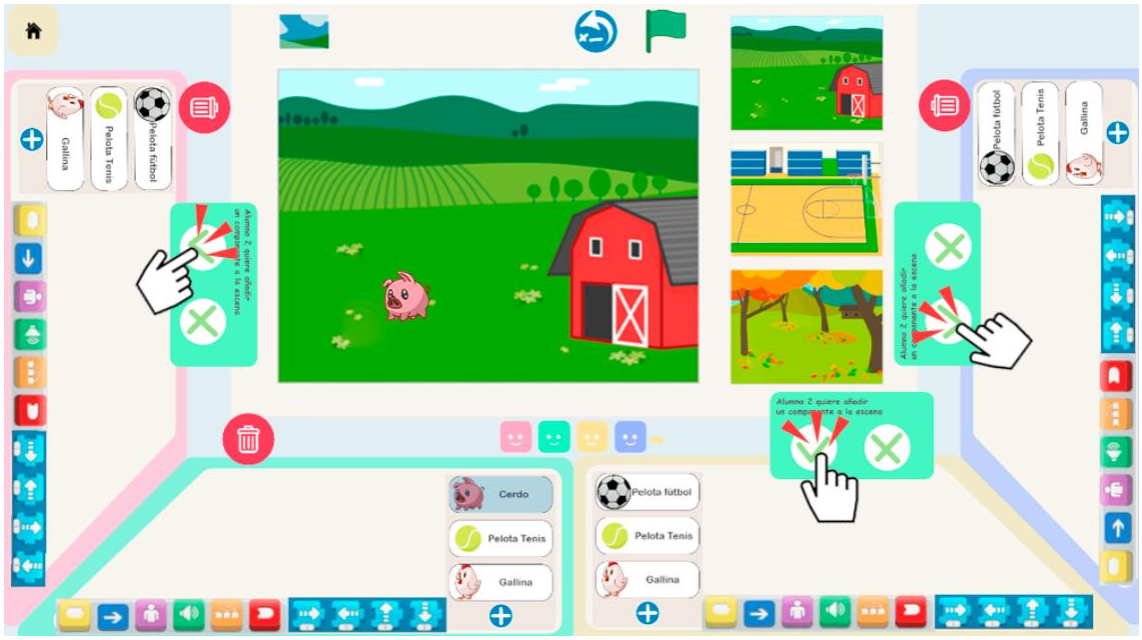

Figura 38 Notificaciones de intención de añadir un nuevo elemento en la escena

Una vez el personaje se encuentra en el escenario, podemos ejecutar el programa al pulsar en el icono de la bandera verde en la zona superior y comprobar que nuestra secuencia funciona correctamente.

También es posible restaurar el estado inicial de los elementos de la escena pulsando en el icono de la flecha hacia atrás en la zona superior, para revertir los efectos del programa ejecutado y dejar la escena como antes de haber pulsado la bandera verde.

Como elemento "extra" meramente visual, también es posible modificar el fondo de la escena, ya sea con los escenarios que se encuentran en el mostrador del lateral de la escena, como los que están en el menú oculto de escenarios.

Para desplegar el menú extra de escenarios debemos pulsar en el icono superior con un paisaje y ahí seleccionar el deseado. Este menú cambiará el escenario que tengamos seleccionado por el nuevo, como se muestra en la siguiente figura.

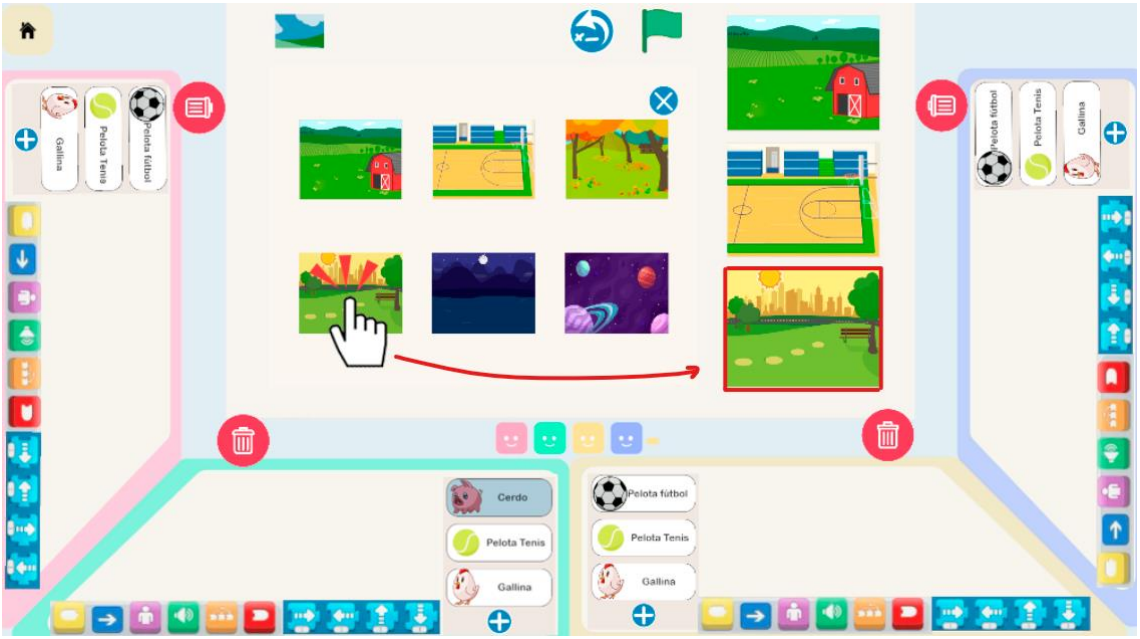

Figura 39 Acceso a más escenarios desde el sub-menú

Por último, si queremos modificar de cuantos jugadores es la sesión, solo se deberá pulsar los iconos de "+" y "-" que están junto a los iconos con emoticonos de caritas sonrientes.

4.2 Implementación

Una vez desarrollado el prototipo de alta fidelidad y teniendo claro cómo funciona la aplicación se procede a explicar cómo está construida.

Para esta tarea durante el desarrollo se barajó la idea de múltiples plataformas y motores gráficos, como Unreal Engine[15] 4, Unreal Engine 5 y Unity; y se decidió finalmente optar por **Unity**[16], un motor de videojuegos gratuito creado para soportar múltiples plataformas, incluyendo PC, consolas, dispositivos móviles y realidad virtual, por lo que permitía desarrollar una aplicación pensada para ser usada en una mesa táctil de grandes dimensiones.

Otros motivos de peso a la hora de tomar esta decisión fueron: la familiaridad con la interfaz, el estar basado en un lenguaje conocido y con el que se tiene experiencia como lo es C#, la existencia de una gran cantidad de tutoriales, foros y una comunidad de usuarios consolidada, que diera opción a trabajar en entornos 2D fácilmente, etc.

Componentes de la escena de Unity

En la siguiente imagen se pueden observar los elementos que componen la escena principal dentro del editor de Unity. Para poder explicarlos de forma más clara se van a agrupar por índices en esta explicación según los *scripts* que usan y su funcionamiento dentro del proyecto.

[15] Unreal Engine: https://www.unrealengine.com/en-US
[16] Unity: https://unity.com/

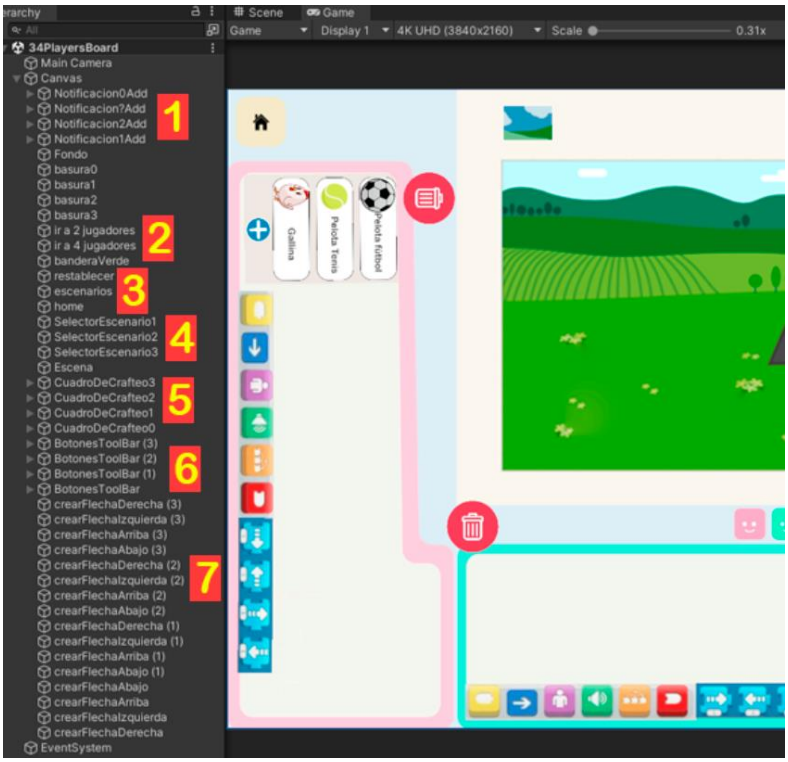

Figura 40 Índice de elementos de las escenas en el motor gráfico

Clasificación de elementos de la escena:

1. Notificaciones: son los cuadros de diálogos que deben mostrarse cada vez que se quiera proponer un nuevo programa a la escena. Estos contienen un archivo llamado “VerificarDecisión.cs”, encargado de controlar valores booleanos que indiquen si el cambio ha sido aceptado, si ya se ha realizado la votación, si alguno de los usuarios ha votado que no esta de acuerdo… El elemento notificaciones contiene hijos en la jerarquía que son cada uno de los cuadros de diálogo que deben aparecer, y a su vez, cada cuadro contiene un archivo llamado “DetectarBotonesSiNo” que recoge la acción del usuario y qué ha votado en cada momento para que el archivo padre “VerificarDecisión.cs” pueda controlar si la decisión ha sido a favor o en contra, como se en el fragmento de código a continuación:

```
// Verificar si algún hijo ha seleccionado el botón "No"
if (cantidadNoSeleccionados > 0 && !seHaMostradoMensaje)
{
```

```
            Debug.Log("Se ha seleccionado el botón 'No' en al menos uno
de los hijos.");
            seHaMostradoMensaje = true;
            apto = false;
            // Realizar acciones adicionales según sea necesario

        }else if (cantidadNoSeleccionados == 0 && !seHaMostradoMensaje &&
cantidadSiSeleccionados == detectoresHijos.Length){
            Debug.Log("Se acepta");
            apto = true;
        }
```

2. Iconos de cambio de escena: estos iconos son los del botón “home” y los de cambiar el número de usuarios, y contienen el archivo “CambiarEscena.cs”, que carga la escena que haya sido designada previamente.

3. Escenarios: se trata del icono superior con la miniatura de un paisaje. Este contiene el archivo “MenuEscenarios.cs”, el cual contiene una referencia de los escenarios que aparecen disponibles en la columna de la derecha de la escena y del sub-menu de escenarios extra que se genera al tocar el icono. Este *script* se encarga de gestionar los cambios de fondo de escenario, los cambios de los escenarios disponibles y de crear o destruir el menú de escenarios extra.

4. Selectores de escenario: estos elementos auxiliares facilitan el uso de la aplicación al evitar tener que navegar al sub-menú de escenarios extra para cambiar el fondo. En general estos selectores hacen referencia a los tres escenarios disponibles en la zona lateral derecha de la escena, y permiten hacer cambios rápidos de fondo mediante el archivo “CambiarFondo.cs”.

5. “Cuadros de crafteo”: Hacen referencia a las zonas donde cada jugador manipula los bloques y genera los programas. Dentro de estos elementos encontramos los selectores de personaje, que funcionan de manera similar a

los iconos de escenarios. Estos selectores contienen un archivo llamado “Personajes.cs”, encargado de gestionar qué personaje ha sido seleccionados, cuáles deben aparecer en la previsualización, cuando generar el submenú de personajes extra, etc.:

```
private void CambiarSprites(Sprite nuevoSprite)
{
    // Verificar si el nuevo sprite ya existe en el array spriteOpciones
    if (Array.Exists(spriteOpciones, sprite => sprite == nuevoSprite))
    {
        Debug.Log("El sprite ya existe en spriteOpciones.");
        DestruirMenuPersonajes();
        return; // Salir de la función sin realizar cambios
    }

    int indiceQueActualizar =
ObtenerIndiceSpriteSeleccionadoEnOpciones();
    // Cambiar el sprite seleccionado al nuevo sprite
    spriteSeleccionado = nuevoSprite;
    spriteOpciones[indiceQueActualizar] = nuevoSprite;

    // Obtener el índice del sprite seleccionado dentro de
spritesPersonaje
    int indexSubpersonaje = Array.IndexOf(spritesPersonaje, nuevoSprite);
    Debug.Log("indiceQueActualizar: " + indiceQueActualizar);
    Debug.Log("indexSubpersonaje: " + indexSubpersonaje);

    Image imageComponent =
transform.GetChild(indiceQueActualizar).GetComponent<Image>();
    if (imageComponent != null)
    {
        // Cambiar el sprite del GameObject hijo
        imageComponent.sprite =
spritesSelectorPersonaje[indexSubpersonaje];
    }
    else
    {
        Debug.LogError("No se encontró el componente Image en el hijo con
el índice " + indiceQueActualizar);
    }

    DestruirMenuPersonajes();
}
```

6. Botones “tool bar”: este objeto de la escena contiene el grupo de botones que cada jugador tiene y que son los encargados de cambiar los bloques disponibles cuando se pulsan. Dentro de cada uno de estos objetos están un botón de cada color con el código “ToolBarSwitcher.cs” que se encarga de comunicar que tipo de bloques se deben generar y qué imagen de menú usar según el color seleccionado.

7. Crear Flecha: estos iconos llamados crearFlechaDerecha, crearFlechaIzquierda, etc, son los botones que, en este caso como está seleccionado el botón azul son de movimiento, que se encargan de generar los bloques indicados al pulsarse mediante el código de “CrearItemMovimiento.cs”, el cual les otorga la capacidad a los bloques de detectar cuando son pulsados y arrastrados cuando se generan, permitiendo su mecanismo básico de uso. A su vez, este código hace que cuando se generen los bloques, estos contengan un archivo de código llamado “AsignarHijoIzquierdo.cs”, que es quien se encarga de la parte más compleja del funcionamiento de la aplicación, como el emparentamiento, qué sonido debe hacer en el bloque de sonido, en qué personaje debe transformarse en la escena, si está listo para ser ejecutado el programa, etc. Además, dependiendo del tipo de bloque, si contiene un indicador numérico, también contará con el código de “NumeroContador.cs”, encargado de notificar los cambios de dicho indicador, detectar los botones de aumento o disminución y también notificar a “AsignarHijoIzquierdo.cs” este valor cuando este active la ejecución del programa. A continuación se puede ver un fragmento del código encargado de la generación de bloques:

```
            // calcular la posición central de la imagen de la zona de
crafteo
            Vector3 centerPosition =
imagenZonaCrafteo.rectTransform.position;
            // asignar la posición inicial al centro de la zona de
crafteo
            rt.position = centerPosition;

            // agregar un Entry para el evento de drag
```

```
            EventTrigger eventTrigger =
currentDraggedImage.GetComponent<EventTrigger>();
            if (eventTrigger == null)
            {
                eventTrigger =
currentDraggedImage.AddComponent<EventTrigger>();
            }

            EventTrigger.Entry entryDrag = new EventTrigger.Entry();
            entryDrag.eventID = EventTriggerType.Drag;
            entryDrag.callback.AddListener((data) => {
OnDrag(currentDraggedImage, (PointerEventData)data); });
            eventTrigger.triggers.Add(entryDrag);

            EventTrigger.Entry entryUp = new EventTrigger.Entry();
            entryUp.eventID = EventTriggerType.PointerUp;
            entryUp.callback.AddListener((data) => {
OnPointerUp(currentDraggedImage, (PointerEventData)data); });
            eventTrigger.triggers.Add(entryUp);

            draggedImages.Add(currentDraggedImage);
            AsignarHijoIzquierdo asignarHijoScript =
currentDraggedImage.AddComponent<AsignarHijoIzquierdo>();
            asignarHijoScript.enabled = false;  // Desactivar
temporalmente el script, se activará cuando sea necesario
```

5. PRUEBAS Y EVALUACIÓN

En este capítulo se llevarán a cabo pruebas de funcionalidad y usabilidad para poder realizar un análisis de la aplicación desarrollada, para posteriormente poder estudiar los resultados y obtener una visión más global de sus fortalezas y defectos. Para ello se realizará una encuesta a través de *forms* del paquete *office365* con los aspectos que componen la evaluación para que los coautores de la publicación de este proyecto puedan analizar las fortalezas y debilidades de este programa.

5.1. Pruebas funcionales

En este apartado se someterá a la herramienta a una serie de pruebas de uso para verificar que todas las funcionalidades están programadas correctamente, con el fin de verificar que el sistema es apto para ser sometido a una evaluación más adelante.

Todas las funcionalidades programadas se describen en la siguiente enumeración, donde se describirán las pruebas realizadas para cada categoría:

1. **Interactuabilidad para la modificación de personajes y escenarios**: el sistema detecta correctamente cuándo el usuario realiza una interacción para realizar un cambio de personaje o el fondo de escenario, en ningún momento se bloquea o está deshabilitada esta función. Mediante el uso de “debugs” y la retroalimentación de los iconos de la interfaz se analiza que las interacciones son detectadas por el código siempre.
2. **Identificación correcta de personajes y escenario seleccionados**: comprobar que la aplicación detecta en todo momento las imágenes correctas seleccionadas por cada usuario y es consciente de la última modificación válida hecha por cada uno, probando a cambiar la apariencia de los personajes y escenarios seleccionados en todas las pantallas en todos los estados posibles de la herramienta (si está el código en la zona común, si primero se selecciona uno se ejecuta y se cambia, al crear un mismo usuario dos programas con dos personajes distintos, etc.).
3. **Generar bloques correctamente**: en todo momento al pulsar en un botón de bloque se debe generar ese bloque en la zona correcta dependiendo del

usuario. También se comprueba que este se genera con todas las subfunciones e información correcta, como la capacidad de arrastrarlo por pantalla o la de detectar qué usuario es su dueño, entre otras.

4. **Sistema de enlace de bloques**: comprobar que en todo momento se detecta correctamente cuando hay dos bloques cerca para unirlos, que se deshabilite cuando el bloque no deba unirse a otro, que todos los bloques unidos detecten al mismo dueño y las mismas zonas de la interfaz al unísono…
5. **Detección de zonas**: verificar durante la ejecución en Unity que todos los scripts y elementos de la escena detectan las mismas zonas de la misma forma, todos los elementos que lo necesitan tienen una referencia correcta de la zona común y sus delimitaciones, de las zonas individuales, de donde están las zonas de borrado, etc.
6. **Adaptación de bloques según la zona de la interfaz**: al arrastrar iconos de unas zonas a otras se debe comprobar que estos se giran, por ejemplo, si están en las zonas laterales, que deben deshacer este giro si se encuentran en la zona común o pasan a otra zona individual, que todos los bloques de un programa giran a la vez. También se comprueba que los iconos al entrar en la zona común detecten que deben cambiar su aspecto por la apariencia del personaje seleccionado.
7. **Identificación y ejecución de bloques**: se verifica que en todo momento el código cuando está bien programado y se interactúa con el icono de ejecutar, se ejecuta correctamente, según la configuración generada y no existen comportamientos inesperados al ejecutar ningún tipo de bloque.
8. **Recuperación del estado antes de su ejecución**: probar a recuperar el estado cuando se haya ejecutado muchos tipos de bloques, cuando no se haya ejecutado ninguno, o si en algún momento la aplicación detecta erróneamente un programa creado pero que no se ha llegado a ejecutar como parte del protocolo de recuperación por error…
9. **Detección de permisos**: usar la interfaz, proponer programas en diferentes estados del sistema, comprobar si a los bloques se les notifica correctamente qué pueden hacer o qué acciones no deben activarse según la situación… En

general se trata de comprobar que los elementos de la interfaz bloquean o desbloquean funciones según cada interacción.

10. **Sistema de notificaciones**: probar a proponer y retirar programas creados en todas las situaciones posibles y ver si el sistema de mensajes se activa cuando es necesario y este, según lo que decidan los jugadores, llaman a la detección de permisos para que cambien los permisos de estos elementos.
11. **Navegación entre pantallas**: comprobar que el flujo de uso funciona entre pantallas y que las funciones de menús en la interfaz están operativas siempre que se interactúen con ellas.

Estas funcionalidades se fueron probando a medida que se iban creando, pero sobre todo, se hicieron varias iteraciones de pruebas al final del proceso de desarrollo, para ir detectando errores de código e ir reprogramando los elementos necesarios y la interactividad que hay entre ellos. Por ejemplo, en una de las iteraciones de pruebas funcionales se detectó que en algunos casos los bloques no detectaban quien era su usuario dueño si algunos de estos entraban por error en la zona de otro usuario pero otros no, por lo que eso se corrigió dentro del código para que todos los bloques de un programa se pusieran de acuerdo en tener la misma información.

Tras cerrar la fase de prueba con todas las correcciones de *bugs* y fallos funcionales, la herramienta cumple con todos los requisitos para someterse a una evaluación de usabilidad por los expertos.

5.2. Planteamiento del análisis de usabilidad

El objetivo de esta evaluación realizada por expertos en usabilidad y el propio autor de este proyecto es mejorar la experiencia del usuario para futuras versiones detectando problemas de usabilidad, manejo de la interfaz por el usuario por primera vez, facilitar la identificación de problemas difíciles de detectar durante el desarrollo… mediante la **heurística de Nielsen** [32](puntuando del 1 al 10 en cada aspecto de esta, siendo 1 no la cumple y 10 la cumple completamente) de la aplicación desarrollada.

Antes de empezar a puntuar los aspectos de la evaluación, se desarrolló un itinerario para que los encuestados pudieran probar a realizar varias tareas, de tal modo que

puedan tener una visión clara de las opciones que brinda la aplicación y puedan analizar como de fácil es identificar dichas tareas y ejecutarlas.

El itinerario consiste en los siguientes pasos:

1. Abre el programa y crea una sesión de 3 jugadores y sitúate como el usuario central.
2. Cambia el fondo del escenario.
3. Selecciona uno de los personajes y cámbialo por el cerdito.
4. Programa un desplazamiento arriba con valor 3 y un desplazamiento a la derecha de valor 7 encadenados.
5. Propón el programa a la zona común, ejecútala y haz que la escena vuelva al estado inicial.
6. Cámbiate de posición y colócate en la zona del usuario lateral-derecho.
7. Vuelve a cambiar de personaje desde la nueva posición.
8. Modifica el programa, añade nuevos bloques de efectos y vuelve a proponerlo a la zona común.
9. Añade un jugador desde esa pantalla para cambiar al modo 4 jugadores.
10. Vuelve a comprobar las funcionalidades en esta nueva modalidad si lo deseas.
11. Procede a realizar la siguiente evaluación.

Las sesiones de evaluación fueron cuatro, cuyo registro se recoge en la Tabla 1:

Usuario	**Fecha**	**Tiempo**
Evaluador1	14/05/2024	28:36
Evaluador2	17/05/2024	105:49
Evaluador3	21/05/2024	12:05
Evaluador4	23/05/2024	31:33

Tabla 1 Evaluadores

Cabe aclarar que la sesión de Evaluador2 se dividió en dos, primero una en la que probó la herramienta y luego otra en la que realizó la encuesta de la evaluación, por lo que su marca de tiempo solo recoge su tiempo de respuesta en la encuesta.

5.2. Evaluación de usabilidad

A continuación desglosaremos las puntuaciones dadas por cada usuario y desarrollaremos por categorías basadas en el decálogo de la heurística de Nielsen los resultados obtenidos.

Visibilidad del estado del sistema

Esta categoría valora si el usuario está informado en todo momento acerca de lo que está ocurriendo a través de una retroalimentación adecuada. El objetivo de este apartado es reducir la incertidumbre para obtener una mayor satisfacción de uso. La puntuación marcada por los evaluadores se muestra en la Figura 41.

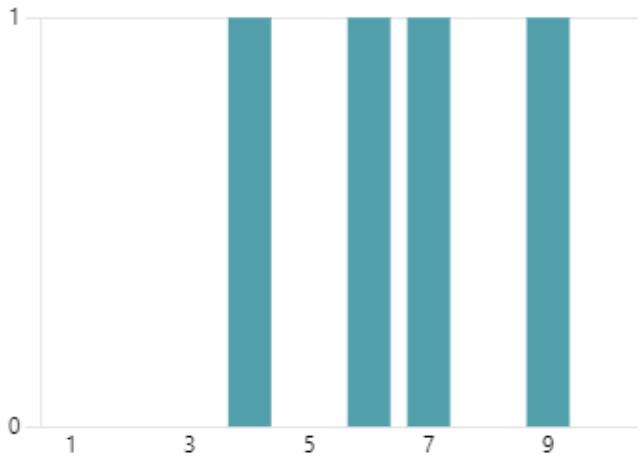

Figura 41 Puntuaciones de Visibilidad del estado del sistema

En este apartado, la puntuación promedio es de 6.50 sobre 10, con las siguientes justificaciones:

- "Cuando estoy cambiando un personaje no sé si es que estoy añadiendo o modificando. Tampoco se sabe qué personaje estoy programando (al menos al principio)"
- "(…) Entiendo que los personajes son los mismos en todas las áreas de trabajo. Cuando un usuario está trabajando con un personaje no se resalta (y bloquea) en el resto de áreas de trabajo (…) Cuando te traes un objeto para editarlo, no se marca el objeto que se está editando en tu área de trabajo (…)"
- "Se podría mejorar la visibilidad con algún mensaje de retroalimentación al usuario (acción aceptada por el resto de los participantes)"

- “Podría mejorarse con mensajes o notificaciones que expliquen por ejemplo por qué un programa diseñado no compila.”

Conclusión: podrían mejorarse más ciertos aspectos que no permiten una visibilidad del estado del sistema clara, generando confusión sobre las acciones realizadas, problemas con la duplicación de tareas debido a la falta de retroalimentación, y dificultades para identificar qué personajes u objetos están siendo editados. Por ello se propone una mejora como un resaltado en todo momento que permita identificar fácilmente qué personaje y escenario están seleccionados por defecto.

Coincidencia entre el sistema y el mundo real

Esta categoría evalúa si se usan conceptos familiares al usuario y se siguen las convenciones del mundo real, y que la información aparezca en un orden lógico para el público objetivo. La puntuación marcada por los evaluadores se muestra en la Figura 42.

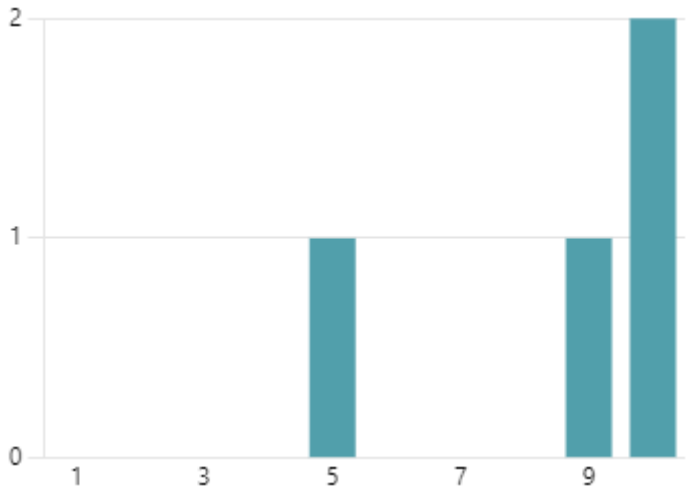

Figura 42 Puntuaciones de Coincidencia entre el sistema y el mundo real

En este apartado, la puntuación promedio es de 8.50 sobre 10, con las siguientes justificaciones:

- “Se entiende los mensajes que se muestran”
- “- En el enunciado de la tarea pone cambiar, esto se hace con el "+". Pero el "+" indica más bien añadir, no cambiar uno que ya existe. - Si mando un script de nuevo a la escena sin hacer cambios no tendría que pedir permiso de aceptación del resto de usuarios.”

- “Los iconos e imagenes que se usan son lo suficientemente representativos como para que cualquier usuario pueda entender su significado fácilmente.”

Conclusión: los mensajes y los íconos utilizados en la aplicación son claros en general salvo por los iconos de "+" que sugiere añadir en lugar de cambiar. Por esto se propone como futuros cambios el modificar la apariencia de estos iconos por flechas u otro tipo de símbolos que dejen más claro su función.

Control y libertad del usuario

Esta categoría evalúa que siempre haya una opción clara para dejar un estado no deseado sin tener que pasar por un diálogo extenso y fomentar la fluidez de uso. El sistema debe permitir a los usuarios explorar y utilizar diferentes funciones sin temor a cometer errores irreversibles. La puntuación marcada por los evaluadores se muestra en la Figura 43.

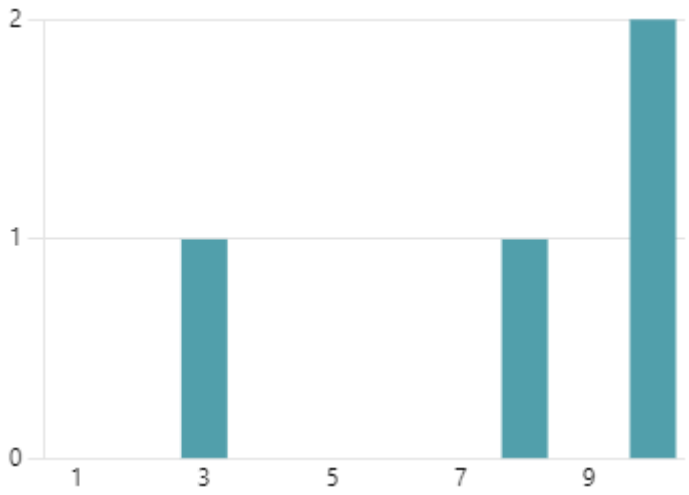

Figura 43 Puntuaciones en Control y libertad del usuario

En este apartado, la puntuación promedio es de 7.75 sobre 10, con las siguientes justificaciones:

- “(…) Evalúo si puede deshacer. No muestra mecanismo de deshacer acciones no deseadas”
- “Nada que decir, salvo la relación con otros aspectos mencionados.”

- “Ha sido difícil encontrar como descartar bloques que no quería utilizar. Igual se podría usar la misma mecánica que en ScratchJR, arrastrar a la zona de bloques de nuevo.”
- “En todo momento está la opción de borrar los programas no deseados, de volver al inicio de sesión o de rectificar el número de jugadores (por poner ejemplos).”

Conclusión: según los encuestados, puede haber problemas para descartar bloques, por lo que se propone modificar la herramienta para hacerla más similar a ScratchJR en cuanto a poder descartar bloques arrastrándolos a la zona de bloques. También podría incluirse en versiones futuras un botón con la funcionalidad de *ctrl+Z* para cada usuario.

Consistencia y estándares

Esta categoría evalúa si los elementos recurrentes son consistentes y si se siguen los estándares de la industria para que los usuarios pueden transferir su conocimiento previo de otras aplicaciones similares. La puntuación marcada por los evaluadores se muestra en la Figura 44.

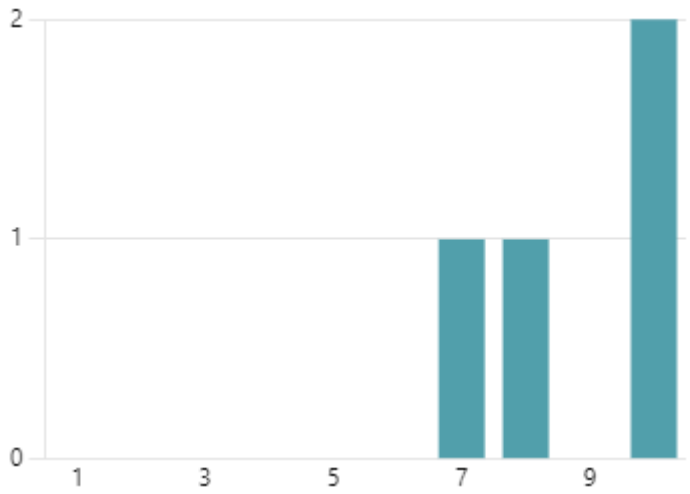

Figura 44 Puntuación en Consistencia y estándares

En este apartado, la puntuación promedio es de 8.75 sobre 10, con las siguientes justificaciones:

- “los diálogos siguen estándares y se entiende bien”
- “- Nada que decir, salvo la relación con otros aspectos mencionados.”

- “Los iconos y elementos interactuables en su mayoría guardan cohesión entre ellos para que el usuario entienda que forman parte de un mismo grupo de tipos de elementos. También se usa simbología y estructuras similares a otros programas del mismo tipo para que al usuario se le haga familiar cuando usa esta aplicación por primera vez.”

Conclusión: no son necesarias mejoras en este aspecto.

Prevención de errores

En esta categoría se evalúa si se proporcionan mensajes claros y comprensibles para eliminar o mitigar las opciones de cometer el error, siempre que sea posible. La puntuación marcada por los evaluadores se muestra en la Figura 45.

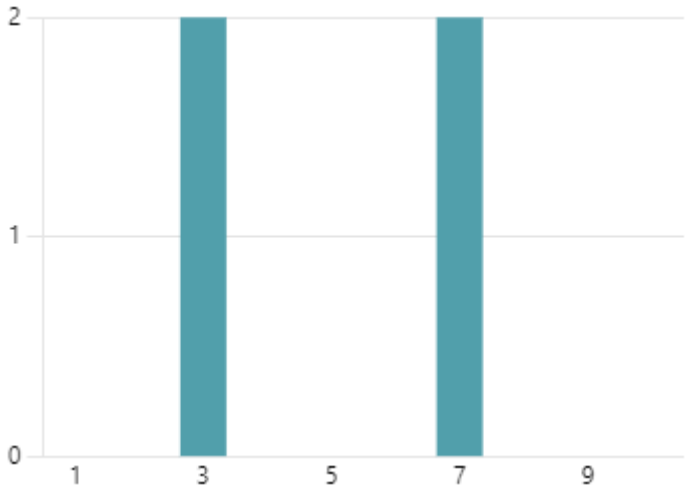

Figura 45 Puntuaciones en Prevención de errores

En este apartado, la puntuación promedio es de 5 sobre 10, con las siguientes justificaciones:

- “no protege errores. Por ejemplo, tengo un programa creado y añado un usuario y se pierde, no avisa que se va a perder. Si arrastro y borro por error no avisa que se va a borrar”
- “- Cuando se va a cambiar de escenario desde la paleta, deshabilitar de alguna forma las escenas que no se puedan seleccionar porque ya están a la derecha. - Cuando te traes un objeto(script) del escenario que no se quede parte fuera del área de trabajo, sino que entre todo. - Cuando llevas algo a la papelera, destacar

todo lo que vas a dejar en la papelera, por ejemplo con un reborde del mismo color que la papelera o una animación. Para que quede claro."

- "El sistema falla en varias ocasiones, se queda bloqueado, los bloques se enganchan con otros con los que no debería,..."
- "En la aplicación se suele dejar que el usuario cometa errores de programación para que aprenda a solucionarlos."

Conclusión: se podría mejorar la prevención de errores en la aplicación, ya que en el estado actual los usuarios pueden realizar acciones sin ser advertidos, como la eliminación accidental de elementos. Se sugieren mejoras para versiones futuras como deshabilitar las opciones que no sean válidas y corregir que los objetos puedan quedar parcialmente fuera del área de trabajo. También, aunque se permite que los usuarios cometan errores de programación para aprender, se propone como mejora el implementar un sistema que especifique más el porqué de un error a través de mensajes.

Reconocimiento en lugar de recuerdo

En este apartado se valora que la carga de memoria para el usuario sea mínima, haciendo que los objetos, acciones y opciones sean visibles sin necesidad de recordar información de una parte de la interfaz para utilizar otra parte. La puntuación marcada por los evaluadores se muestra en la Figura 46.

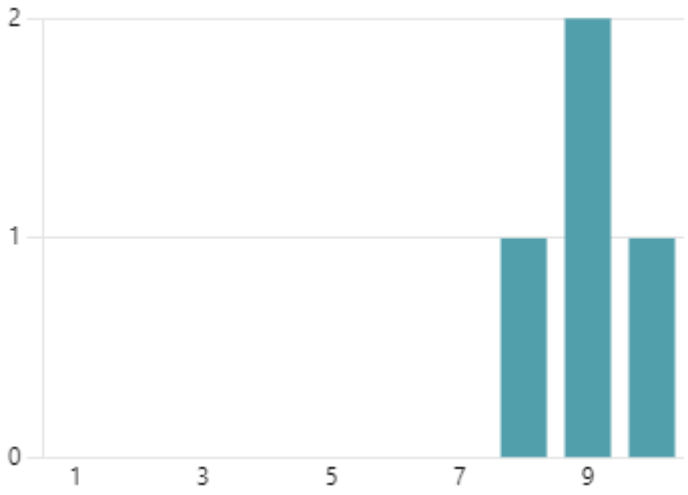

Figura 46 Puntuaciones en Reconocimiento en lugar de recuerdo

En este apartado, la puntuación promedio es de 9.00 sobre 10, con las siguientes justificaciones:

- “tengo que recordar qué personaje estoy programando”
- “- Lo mismo que para visibilidad del estado del sistema. - Que los objetos tengan alguna marca que haga referencia al último usuario que los editó. - Cuando traigo un personaje para editar, que se quede su silueta en la escena, para recordar dónde estaba al comienzo. Esta silueta se borrará cuando lo devuelva a la escena”
- “Es una aplicación con muy pocos cambios de pantallas por lo que el esfuerzo por recordar es mínimo. Lo único un poco más reseñable son los submenús de personajes y escenarios.”

Conclusión: aunque la aplicación presenta pocos cambios de pantalla, se propone como mejora hacer más clara la accesibilidad a los submenús con iconos o elementos visuales como “pestañas” que permitan interpretar mejor que son desplegables.

Flexibilidad y eficiencia de uso

En este apartado se valora que los diálogos no deberían fuercen al usuario a hacer cosas que no quieren hacer. También se valora la eficiencia del sistema de uso para todos los usuarios, no solo para usuarios expertos. La puntuación marcada por los evaluadores se muestra en la Figura 47.

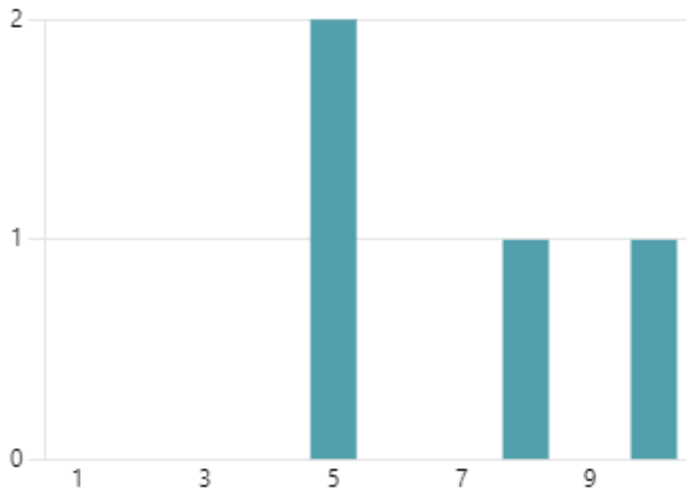

Figura 47 Puntuaciones de Flexibilidad y eficiencia de uso

En este apartado, la puntuación promedio es de 7.00 sobre 10, con las siguientes justificaciones:

- “es flexible porque permite varios programas a la vez por un usuario con varios usuarios pero no es eficiente en el manejo de los bloques en el espacio individual ya que cuando tengo muchos bloques se pegan unos a otros sin querer enlazarlos”
- “- Nada que decir, salvo la relación con otros aspectos mencionados.”
- “Algunas veces se arrastra y otras se pulsa para hacer acciones semejantes (mover bloques)”
- “No existe mucha diferencia entre el uso de un principiante y un experto. No hay atajos de teclado ni nada similar y la curva de aprendizaje es muy llana.”

Conclusión: aunque la aplicación es flexible al permitir varios programas a la vez y múltiples usuarios, el uso del sistema se ve entorpecido por problemas como el agrupamiento involuntario de bloques. Para mejorar la herramienta se propone optimizar el sistema de detección de bloques y aumentar el espacio individual.

Estética y diseño minimalista

En esta categoría se evalúa la utilidad de los diálogos y elementos del sistema, evitando la información irrelevante. El diseño minimalista se caracteriza por la simplicidad y la claridad en la presentación de la información. La puntuación marcada por los evaluadores se muestra en la Figura 48.

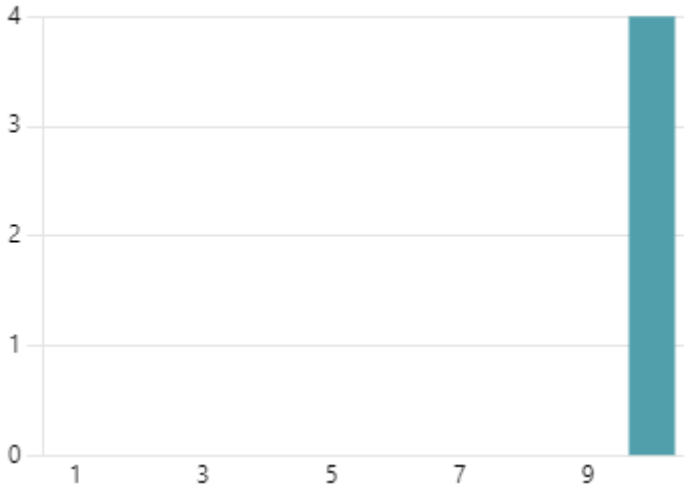

Figura 48 Puntuaciones en Estética y diseño minimalista

En este apartado, la puntuación promedio es de 10.00 sobre 10, con las siguientes justificaciones:

- "el diseño es minimalista teniedno los recursos e iconos solo los necesarios, con diseños simples y claros, colores apropiados, etc."
- "- Nada que decir, salvo la relación con otros aspectos mencionados."
- "La aplicación es muy visual y todas sus interacciones son directas."

Conclusión: se destaca el diseño minimalista en la aplicación, por lo que no hay sugerencias de mejora.

Ayuda al usuario a reconocer, diagnosticar y recuperarse de los errores

Se valora que los mensajes de error se expresen en un lenguaje claro, indicando claramente el problema y sugeririendo una solución constructiva. La puntuación marcada por los evaluadores se muestra en la Figura 49.

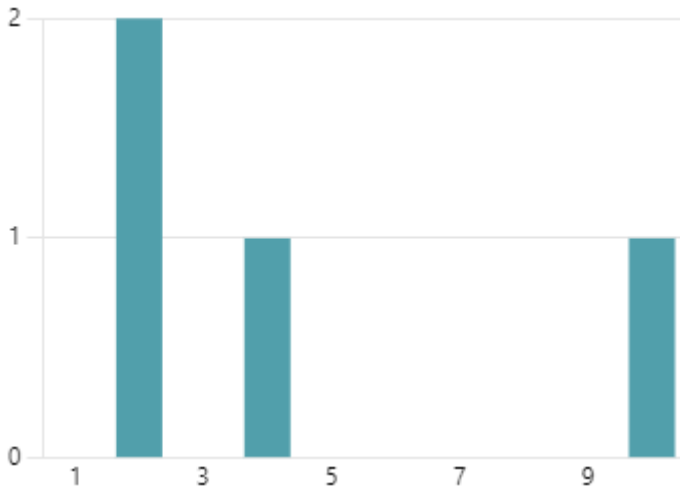

Figura 49 Puntuaciones en Ayuda al usuario a reconocer, diagnosticar y recuperarse de errores

En este apartado, la puntuación promedio es de 4.50 sobre 10, con las siguientes justificaciones:

- “no genera mensajes de errores”
- “- Si se borra un usuario no pide confirmaión y se borra el trabajo de todos. Es fácil de borrar sin querer. - No permite deshacer cambios.”
- “Podría mejorarse el apartado de notificación de errores.”

Conclusión: la falta de mensajes de error puede dificultar el uso de la herramienta, especialmente cuando se cometen acciones involuntarias como borrar usuarios sin confirmación o realizar cambios irreversibles sin la posibilidad de deshacerlos. En versiones futuras se propone aumentar y mejorar el sistema de notificaciones para que haya que confirmar acciones importantes antes de que se ejecuten.

Ayuda y documentación

Se valora que la documentación sea fácil de buscar y que esta se enfoque en las tareas del usuario. Esta debe ser clara, concisa y relevante. La puntuación marcada por los evaluadores se muestra en la Figura 50.

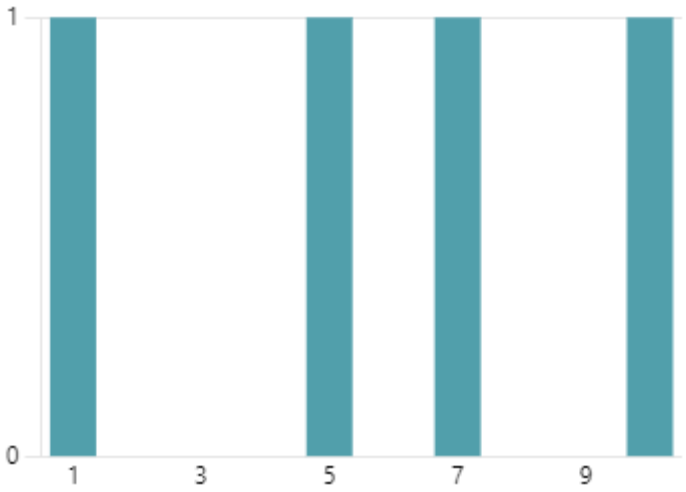

Figura 50 Puntuaciones en Ayuda y documentación

En este apartado, la puntuación promedio es de 5.75 sobre 10, con las siguientes justificaciones:

- "No hay ayuda ni recursos explicativos, no hay guía ni textos que indique para qué sirven las diferentes partes"
- "- Nada que decir, salvo la relación con otros aspectos mencionados."
- "No he visto ayuda en la aplicación"
- "Aunque está basado en scratch jr y hay bastante documentación y tutoriales accesibles de este programa, hay elementos que cambian con respecto a la aplicación desarrollada, por lo que no aplica del todo esta documentación."

Conclusión: existe una falta de ayuda y de recursos explicativos dentro de la aplicación, por lo que se propone desarrollar un submenú de guía del usuario o de ayudas que pueda servir de orientación y que se pueda consultar en cualquier momento.

6. CONCLUSIONES

A continuación se expondrán las conclusiones a las que se ha llegado tras la fase final del proyecto.

6.1. Objetivos alcanzados

A continuación repasaremos las secciones de objetivos expuestos en el capítulo 1 para analizar cuáles se han conseguido y qué se podría hacer para terminar de alcanzarlos:

Objetivos pedagógicos

- Pensamiento lógico: sí que se ejercita el pensamiento lógico mediante la programación de bloques en la aplicación. Sin embargo, es posible mejorar este aspecto añadiendo nuevos bloques con funcionalidades más complejas como los bucles o las condiciones.

- Creatividad: de la misma forma que el apartado anterior, se les da libertad creativa a los usuarios a realizar el programa que quieran, pero podría mejorarse este aspecto ampliando el abanico de posibilidades para una experiencia más enriquecida.

- Habilidades de trabajo en equipo: este objetivo es el que mejor se cumple en el estado actual del sistema. No obstante, se podría fomentar la interactividad entre los miembros de la sesión mediante un sistema de mensajes y emoticonos que amenicen la experiencia conjunta.

Objetivos de usabilidad

- Navegación sencilla: este aspecto está muy conseguido al tratarse de un sistema con poco flujo de interfaces.

- Interactividad: la aplicación es muy interactiva, aunque podría mejorarse introduciendo más efectos visuales y de sonido para hacerla más llamativa.

- Contenido educativo: la aplicación consigue mantener el foco principal en el aprendizaje de la programación, pero podría ampliarse con un sistema de ejercicios o actividades propuestas que lo enfatizase más.

- Programación en equipo: este objetivo es de los que más logrados está, las mejoras que podrían hacerse en este respecto serían muy residuales.

- Accesibilidad: este objetivo no se ha logrado del todo. Si bien la aplicación es usable por alumnos que no hayan tenido experiencia nunca con ella, creo que es necesario mejorarla con un sistema de niveles por dificultad o de poder ir desbloqueando bloques más complejos a medida que vas ganando experiencia con la herramienta para no introducir mucha información de golpe.

6.2. Opinión

Estoy muy satisfecho con el resultado del proyecto. Nunca había desarrollado una aplicación con un enfoque pedagógico, por lo que ha sido un nuevo reto que me ha ayudado a mejorar en mi faceta de diseñador de aplicaciones e interfaces. También el factor de que fuese para una plataforma tan peculiar como lo es una mesa *multi-touch* me ha parecido muy interesante, y aunque al comienzo del proyecto me sentía escéptico por no saber qué retos me podría presentar, al final no ha supuesto ningún problema adaptar la aplicación a este medio.

La parte más laboriosa del tfg ha sido, por un lado enfrentarme a la hoja en blanco durante la etapa de diseño y aprender a usar Figma, generar las interacciones que podría tener la aplicación y analizar si eran factibles o útiles antes de pasar a programarlas; y por otro lado la programación de ciertas funcionalidades más complejas como el sistema de notificaciones y permisos o las funciones de enlazar y mantener la coherencia de la información de los bloques cuando están enlazados.

También cabe destacar que me han servido de mucha ayuda el poder usar Inteligencia Artificial como apoyo a la hora de corregir errores sintácticos de código, algunos bugs, etc y para tareas fuera de la programación como la fase de “lluvia de ideas” o a la hora de redactar algunos textos de esta memoria.

6.3. Trabajo futuro

De cara a seguir expandiendo la herramienta en un futuro es importante analizar aspectos o funcionalidades que no han podido desarrollarse más durante el proyecto, pero que podrían ser interesantes para mejorar la experiencia con el sistema.

1. Dar más opciones y mejorar el aspecto de la accesibilidad para alumnos con capacidades distintas, hacer un sistema de niveles de dificultad que simplifique o haga más complejas las opciones de la interfaz para adaptarla a los niveles de cada clase. También mejorar aspectos como ayudas enfocadas a alumnos discapacitados como una guía de audio o un modificador visual para los iconos y textos, por poner algunos ejemplos.
2. Aumentar el abanico de habilidades que pueden enseñarse con la herramienta. Podría aumentar el número de bloques y fomentar que la herramienta se use para aprender a programar mientras se repasan otras materias mediante ejercicios o actividades generadas por la aplicación.
3. Crear un sistema de motivación para que los alumnos sientan como una recompensa el dedicarle tiempo extra a la plataforma. Fomentar el que usen la aplicación no solo cuando estén con el profesor sino también cuando se les dé tiempo libre.

Bibliografía

[1] P. D. P. a. A. K. Adi, «A review of the Blockly programming on M5Stack board and MQTT based for programming education,» *IEEE 11th International Conference on Engineering Education (ICEED),* 2019.

[2] S. A. F. S. K. J. L. A. R. &. W. E. A. Pila, «Learning to code via tablet applications: An evaluation of Daisy the Dinosaur and Kodable as learning tools for young children,» *Computers & Education,* vol. 128, pp. 52-62, 2019.

[3] M. M. D. A. A. &. M. M. Butler, «Multi-touch display technology and collaborative learning tasks,» *EdMedia+ Innovate Learning,* pp. 1441-1448, 2010.

[4] C. H. B. K. K. L. H. M. M. L. M. M. .. &. S. B. Solomon, «History of logo,» *Proceedings of the ACM on Programming Languages,* pp. 1-66, 2020.

[5] F. López, «Azul Web,» 13 Febrero 2019. [En línea]. Available: https://www.azulweb.net/logo-el-primer-lenguaje-de-programacion-disenado-para-ninos/. [Último acceso: Mayo 2024].

[6] J. Norman, «History of Information,» May 2024. [En línea]. Available: https://www.historyofinformation.com/detail.php?id=815.

[7] A. C. Kay, «History of programming languages---II,» *The early history of Smalltalk,* pp. 511-598, 1996.

[8] «The Grainger College of Engineering,» University of Illinois Urbana-Champaign, 2020. [En línea]. Available: https://grainger.illinois.edu/news/magazine/plato. [Último acceso: 2024].

[9] M. J. M. A. R. &. P. M. S. Fischer, «THINK-A-DOT,» 2004.

[10] P. Wasiak, «CD-ROM encyclopedias: Extending the Gutenberg galaxy to include computer multimedia technologies,» *Human Affairs,* pp. 382-392, 2013.

[11] «C64-wiki,» 22 Abril 2024. [En línea]. Available: https://www.c64-wiki.com/wiki/Compute!. [Último acceso: 2024].

[12] «VintageApple.org,» Google, [En línea]. Available: https://vintageapple.org/byte/.

[13] S. A. Papert, «Mindstorms: Children, computers, and powerful ideas.,» de *Basic books*, 2020.

[14] C. M. V. Solórzano, «Construccionismo. Referente sociotecnopedagógico para la era digital,» *Innovación Educativa,* pp. 45-50, 2009.

[15] «Scratch Wiki,» Scratch, [En línea]. Available: https://en.scratch-wiki.info/wiki/Scratch_11Oct03.

[16] J. M. López, «hipertextual,» 3 Enero 2023. [En línea]. Available: https://hipertextual.com/2023/01/one-laptop-per-child-olpc-notebooks.

[17] N. Shah, «Writing Program,» Boston University Arts & Sciences, [En línea]. Available: https://www.bu.edu/writingprogram/journal/past-issues/issue-3/shah/.

[18] A. S. K. &. Z. P. Parmaxi, «Leveraging virtual trips in Google expeditions to elevate students' social exploration,» de *Human-Computer Interaction–INTERACT 2017: 16th IFIP TC 13 International Conference*, Mumbai, India, 2017.

[19] K. Hao, «China has started a grand experiment in AI education. It could reshape how the world learns,» *MIT Technology Review,* pp. 1-9, 2019.

[20] N. a. M. T. Adhami, «Integrating inquiry-based learning and computer supported collaborative learning into flipped classroom: Effects on academic writing performance and perceptions of students of railway engineering.,» *Computer Assisted Language Learning,* pp. 521-557, 2024.

[21] M. a. R. S. Dooly, «Becoming little scientists: Technologically-enhanced project-based language learning,» 2016.

[22] T. a. E. P. Wanner, «Personalising learning: Exploring student and teacher perceptions about flexible learning and assessment in a flipped university course,» *Computers & Education,* vol. 88, nº 354-369, 2015.

[23] G. Ángeles, «¿Qué es Blockly de Google y cómo se utiliza?,» *La República,* 1 Julio 2023.

[24] R. L. J. A. S. S. &. V. A. Barradas, «Developing computational thinking in early ages: A review of the Code. org platform,» 2020.

[25] «Code.org,» Code, [En línea]. Available: https://code.org/.

[26] W. &. T. R. A. Elsawah, «The Effectiveness of Tynker Platform in Helping Early Ages Students to Acquire the Coding Skills Necessary for 21st Century,» de *International Conference on Information Systems and Intelligent Applications*, 2022.

[27] «Tynker Coding For Kids,» Tynker, [En línea]. Available: https://www.tynker.com/blog/from-block-coding-to-javascript-and-python-how-tynker-teaches-coding/.

[28] S. Goschnick, «App Review:,» Swinburne University of Technology, Melbourne, Australia, 2015.

[29] D. Yaroslavski, « How does Lightbot teach programming. Retrieved January,» 2014.

[30] S. C. T. P. A. B. M. &. B. M. Santos, «Innovative approaches in teaching programming: A systematic literature review,» de *Proceedings of the 12th International Conference on Computer Supported Education*, 2020.

[31] N. M. M. D. H. J. F. M. B. J. S. A. &. X. T. Tillmann, «The future of teaching programming is on mobile devices,» de *roceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, 2012.

[32] J. Nielsen, «Nielsen Norman Group,» Nielsen Norman Group, 1994. [En línea]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/.

Anexo I: Manual del usuario

En esta sección se desglosará con mayor detalle cómo usar la interfaz de la herramienta con una guía visual dividida en bloques.

Iniciar sección

Salvapantallas de inicio, al cual deberemos darle un toque para que se muestre el primer menú:

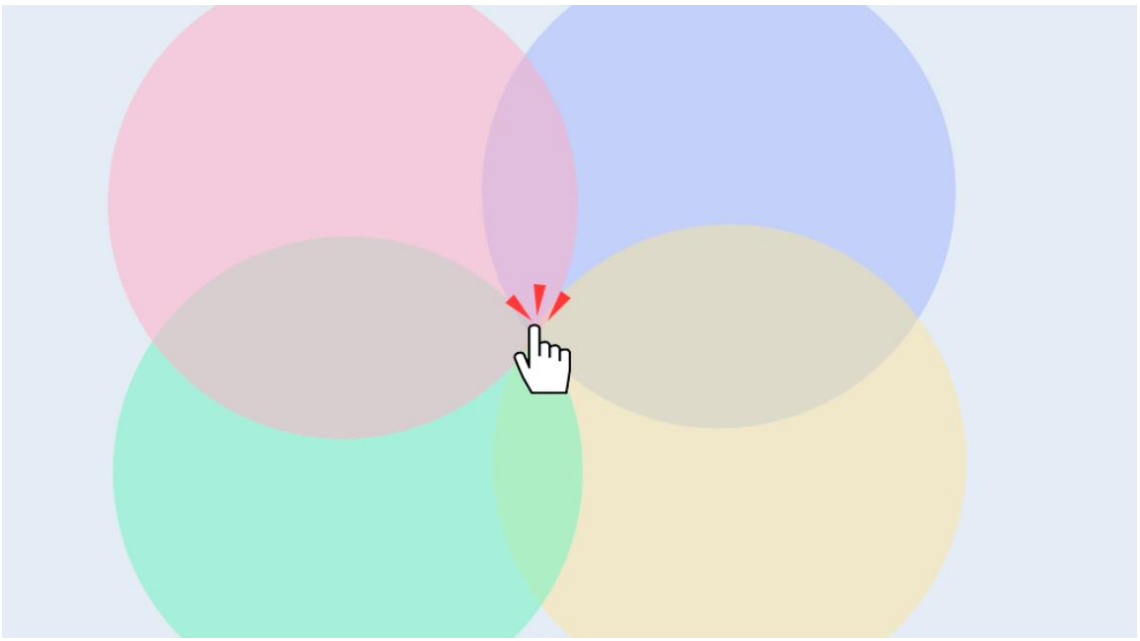

Figura 51 Interacción con la primera pantalla

Pantalla donde podremos elegir inicialmente el número de usuarios. Podremos pulsar en los iconos “+” o “-” que van apareciendo a medida que añadimos o quitamos jugadores:

Figura 52 Interacción con el menú de número de jugadores parte 1

Pulsar el botón en el que está escrita la frase “¡Comenzar!” para ir a la interfaz de programación:

Figura 53 Interacción con el menú de número de jugadores parte 2

Crear Script

En la interfaz principal podríamos seleccionar qué tipo de bloques queremos generar pulsando en los botones de la mitad izquierda de la barra de herramientas:

Figura 54 Interacción con el selector de tipos de bloques

Al pinchar en uno de los bloques disponibles de cada sección se generará en el centro de la escena de trabajo:

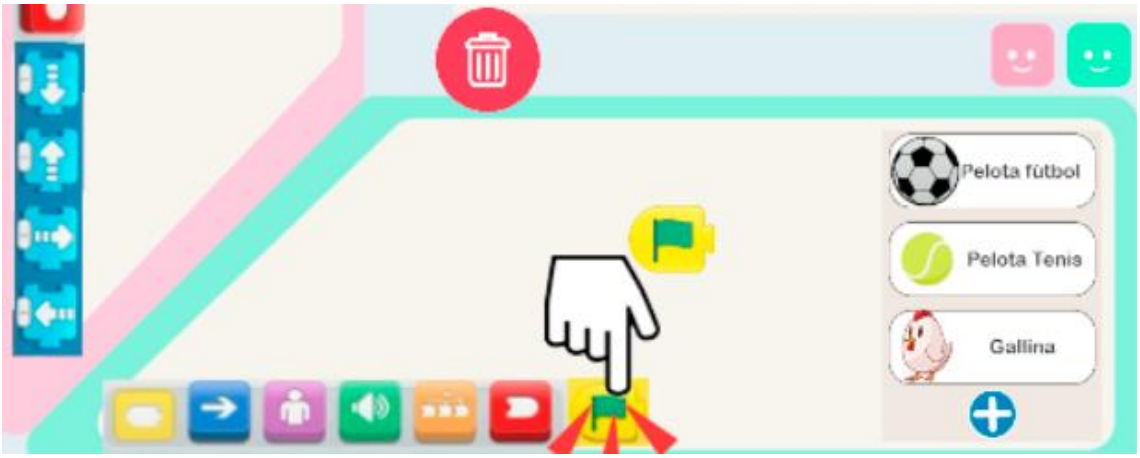

Figura 55 Generación de bloques nuevos

Para manipular el bloque que acabamos de generar, solo necesitamos pulsar encima de él, y manteniéndolo pulsado, arrastrarlo a donde deseemos. Por ejemplo, para borrar el bloque que acabamos de generar deberíamos arrastrarlo al icono de la basura, tal y como se muestra en la siguiente imagen:

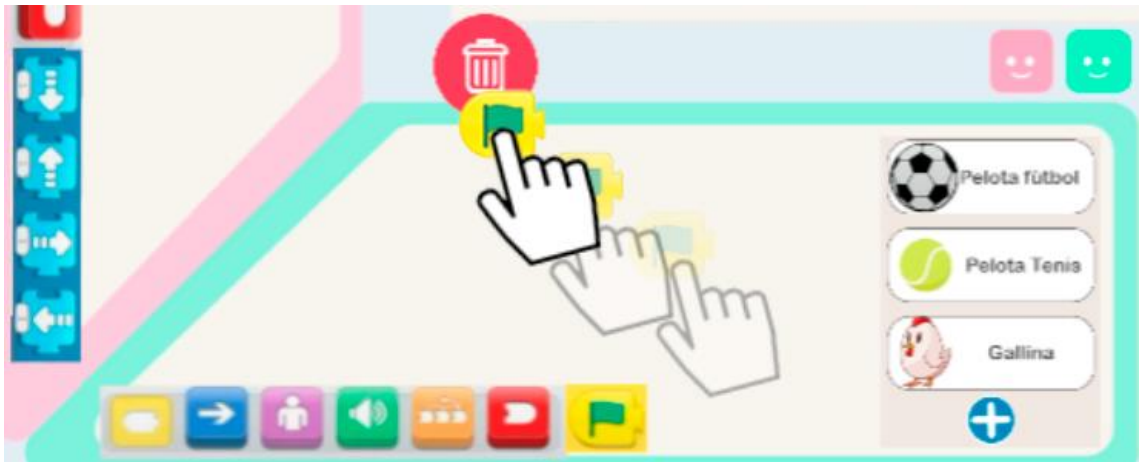

Figura 56 Mover un bloque por la pantalla

Si se quiere encadenar bloques, se deben generar y unir siguiendo el orden que se desee de izquierda a derecha. Por ejemplo, para una secuencia de movimientos simple debemos ir generando los bloques de inicio de ejecución (amarillo), los de movimiento (azules) y el de fin de ejecución (rojo). Pero primero debemos entender como conectar y desconectar los bloques de una secuencia.

Para conectar dos bloques solo deberemos acercarlos lo suficiente y cuando estén lo suficientemente cerca detectarán que quieres unirlos y, en consecuencia, se conectarán.

Figura 57 Cómo enlazar bloques

Ahora podemos mover la secuencia como si fuese una sola unidad o separarla. Para mover los bloques manteniéndolos unidos debemos arrastrar el bloque de mayor prioridad, en este ejemplo sería el amarillo:

Figura 58 Cómo desplazar bloques enlazados

Si por el contrario queremos mover un bloque para romper una conexión concreta, debemos arrastrar el bloque de menor prioridad, es decir, el que se encuentra por al lado derecho de la conexión. En el siguiente ejemplo podemos ver que al arrastrar el bloque de en medio, el azul, la conexión entre el amarillo y el azul se rompe porque estamos moviendo el bloque de la derecha de esa conexión, pero la conexión entre el azul y el rojo se mantiene intacta ya que estamos moviendo el bloque de la izquierda con respecto a esa conexión:

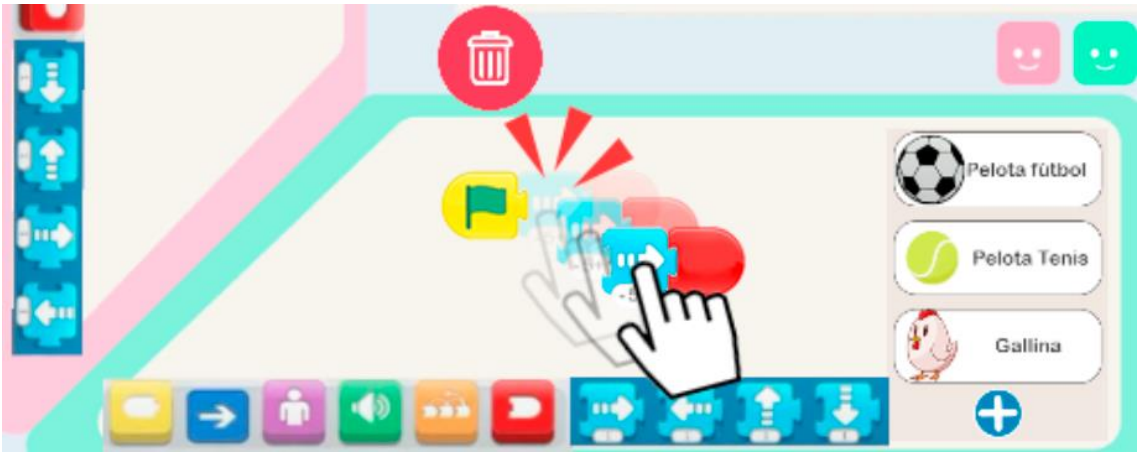

Figura 59 Separar bloques enlazados

Dependiendo de los bloques que usemos, tendremos la opción de manipular la cantidad de efecto que queremos aplicar. Por ejemplo, en este caso podemos definir cuánto queremos que se desplace hacia la derecha pulsando en los símbolos de "+" o "-" en la zona inferior del bloque:

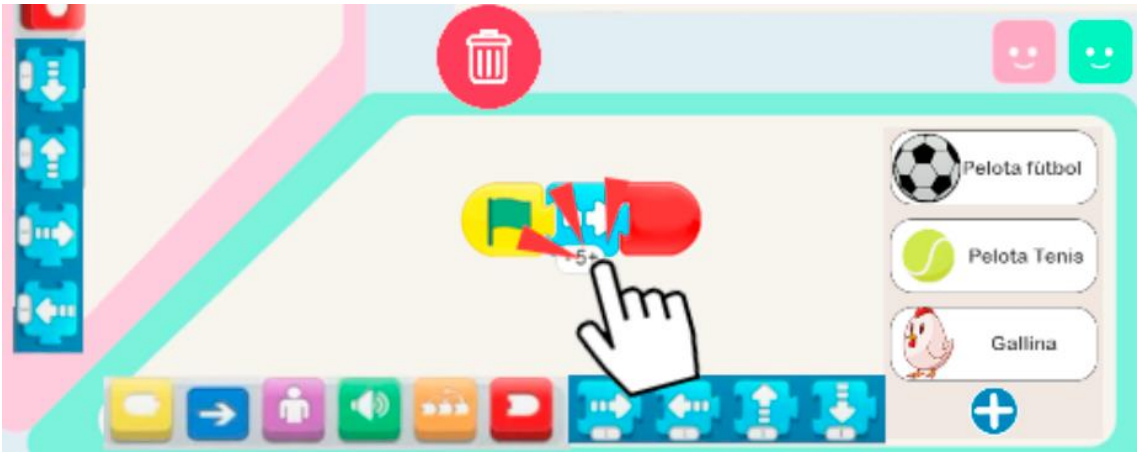

Figura 60 Modificar el contador de un bloque

Una vez tenemos el programa que queremos, también podemos seleccionar el personaje que se desee en el momento, ya sea de los que están disponibles a simple vista como de los que están en el menú extendido con todos los personajes:

Figura 61 Uso del menú de cambio de personaje

Proponer un script

Para poner a prueba el programa que hemos creado debemos arrastrar la secuencia de bloques a la zona deseada del escenario y pulsar en la bandera verde. Al entrar en

contacto con la escena, la secuencia de bloques se convertirá en el personaje preseleccionado:

Figura 62 Añadir un programa a la escena

Al colocar el personaje en el escenario, nuestros compañeros deberán seleccionar que están de acuerdo con nuestra propuesta, de lo contrario el programa creado volverá a nuestra zona individual:

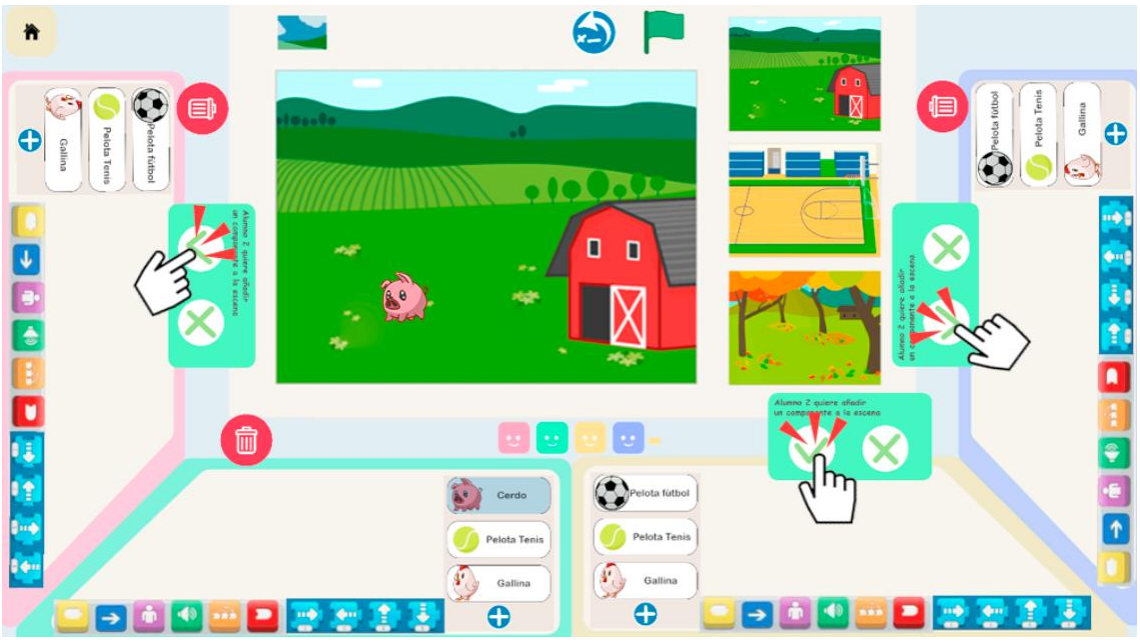

Figura 63 Notificaciones de intención de añadir un nuevo elemento en la escena

Una vez el personaje se encuentra en el escenario, podemos ejecutar el programa y comprobar que nuestra secuencia funciona correctamente:

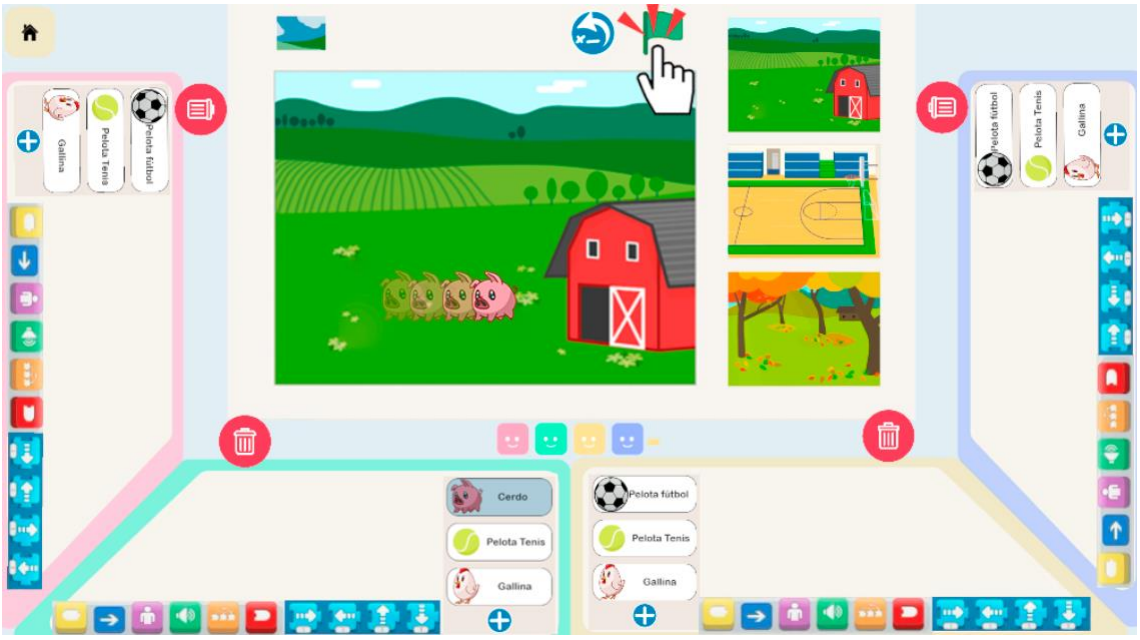

Figura 64 Ejecutar un programa en escena

También es posible restaurar el estado inicial de los elementos de la escena para revertir los efectos del programa ejecutado y dejar la escena como antes de haber pulsado la bandera verde:

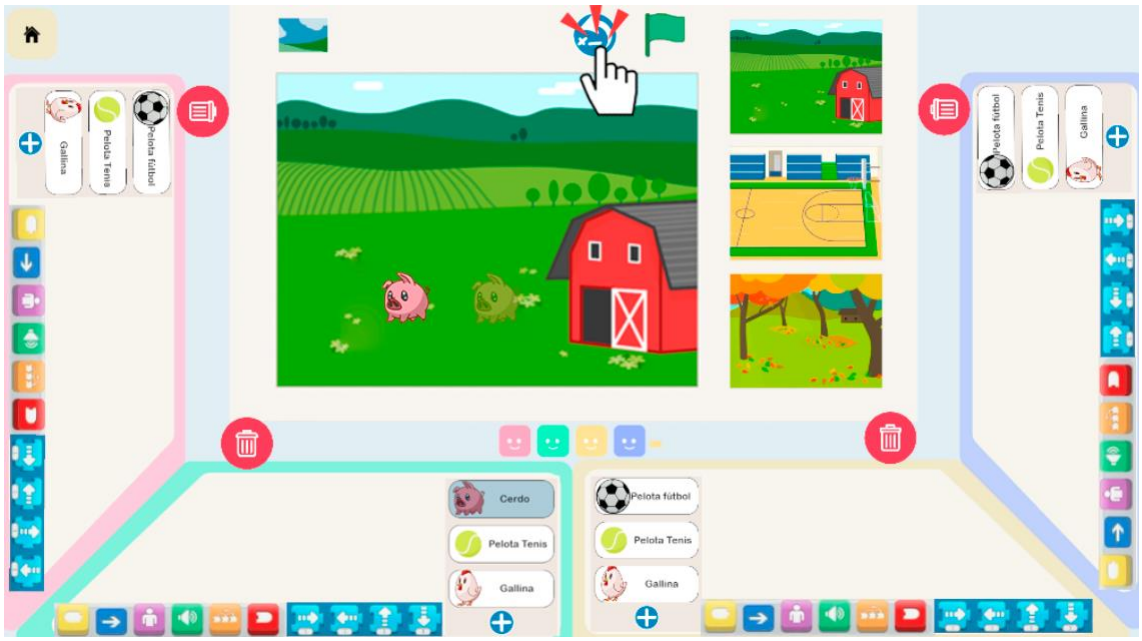

Figura 65 Restaurar un programa ejecutado a su estado inicial

Configurar escena

Como elemento “extra” meramente visual, también es posible modificar el fondo de la escena, ya sea con los escenarios que se encuentran en el mostrador del lateral de la escena, como los que están en el menú oculto de escenarios:

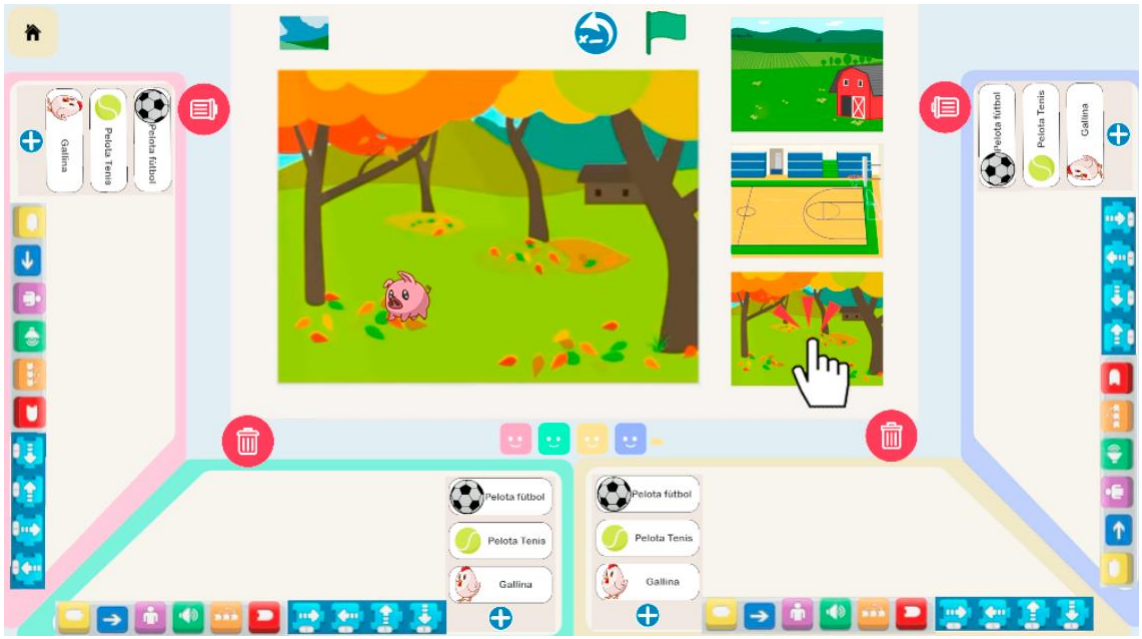

Figura 66 Modificar el escenario desde el menú en pantalla

Para desplegar el menú extra de escenarios debemos pulsar en el icono superior con un paisaje y ahí seleccionar el deseado. Este menú cambiará el escenario que tengamos seleccionado por el nuevo:

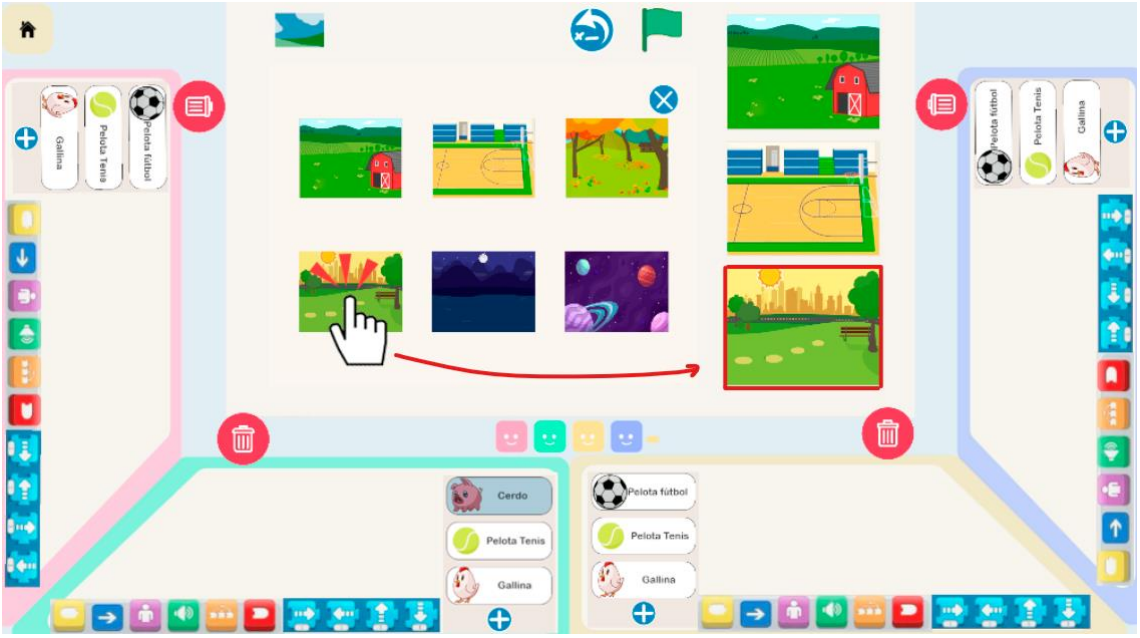

Figura 67 Acceso a más escenarios desde el sub-menú

Por último, si queremos modificar de cuantos jugadores es la sesión, solo se deberá pulsar los iconos de “+” y “-” que están junto a los iconos con emoticonos de caritas sonrientes:

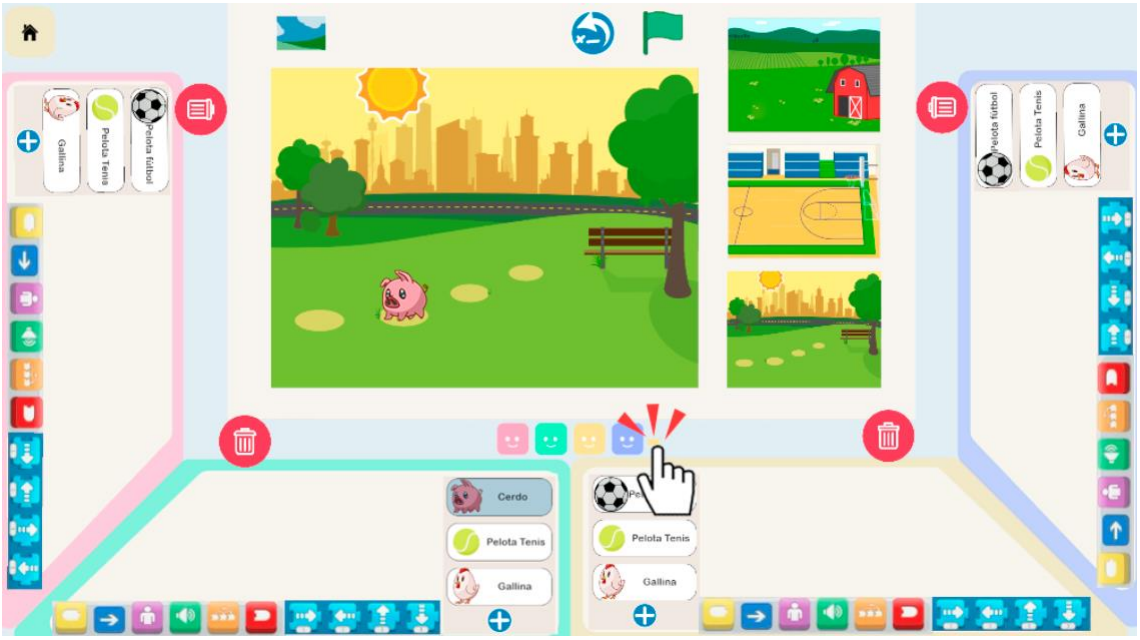

Figura 68 Modificar el número de jugadores durante la sesión