UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MECANISMOS DE INTERACCIÓN NATURAL PARA REDUCIR BARRERAS DE ENTRADA EN ENTORNOS DE PROGRAMACIÓN BASADOS EN BLOQUE

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS MENCIÓN COMPUTACIÓN

SEBASTIÁN PATRICIO TORO GUAJARDO

PROFESOR GUÍA:
FRANCISCO GUTIÉRREZ FIGUEROA

MIEMBROS DE LA COMISIÓN:
ÉRIC TANTER
CRISTIAN PARRA OYARCE
VALERIA HENRIQUEZ NORAMBUENA

Este trabajo ha sido parcialmente financiado por Proyecto FONDECYT Nº 11190248

SANTIAGO DE CHILE
2024

Resumen

El desarrollo de habilidades de pensamiento computacional no sólo permite adquirir conocimientos técnicos, sino que también desarrolla habilidades como la descomposición de problemas o el pensamiento crítico. Sin embargo, aprender a programar puede resultar intimidante, especialmente para principiantes, debido a la complejidad percibida de la disciplina y a las influencias culturales que la presentan como difícil de entender. Este fenómeno crea una barrera de entrada, donde el miedo y la falta de confianza pueden disuadir a los niños y niñas de aventurarse en el aprendizaje de la programación. Aunque se han propuesto soluciones, como lenguajes visuales basados en bloques, como Scratch, el uso común de Mouse y Teclado no resuelve completamente el problema motivacional al no generar suficiente interés en los estudiantes.

En este contexto, surge Gesture Coding, un Lenguaje de Programación Visual basado en Scratch, cuyo mecanismo de interacción son los Joy-Con de Nintendo Switch, utilizando gestos y controles de movimiento para seleccionar y crear bloques. La idea principal de este trabajo es explorar cómo impacta en la percepción de los estudiantes el uso de Joy-Con de Nintendo Switch en el proceso de aprendizaje. En particular, se busca comprender el efecto que tiene el uso de gestos y controles de movimiento en la barrera de entrada de niños y niñas de entre 10 y 12 años al momento de programar.

Para esto, el desarrollo de este trabajo se dividió en tres espiras: (1) Refinamiento y prueba piloto para evaluar el estado inicial de Gesture Coding; (2) Extensión, que incorporó nuevos bloques, como Ciclos y Variables, además de mejoras de diseño; y (3) Caso de estudio, que consistió en talleres con el público objetivo para observar el comportamiento de la herramienta en un entorno controlado. Durante todo el proceso, se aplicó la Teoría Expectativa-Valor como referencia para desarrollar la motivación de los estudiantes.

La evidencia recolectada en este trabajo indica que el uso de gestos y controles de movimiento en Gesture Coding genera una mejor disposición y un ambiente más lúdico entre los participantes. En otras palabras, el uso de gestos o controles de movimiento mediante Joy-Con de Nintendo Switch es capaz de reducir la barrera de entrada a la programación, validando las hipótesis planteadas.

En conclusión, el uso de gestos demostró ser efectivo para aumentar la motivación de los programadores novatos, disminuyendo la barrera de entrada. Aunque se destaca la necesidad de replicar el estudio con un mayor número de participantes y se proponen áreas de investigación futuras, los resultados resaltan la importancia de enfoques creativos y motivadores en la enseñanza de la programación para hacerla más accesible y atractiva para los principiantes.

Dedicado a toda mi familia, cuya motivación fue clave en este trabajo.

Agradecimientos

Agradezco profundamente a todas las personas que me han acompañado en este proceso, en particular a mi familia. No importa cuán difícil pueda verse la situación, siempre han estado ahí brindándome su apoyo incondicional. A mi hermana, Catalina, cuyas conversaciones y ricas preparaciones siempre estaban ahí para levantarme el ánimo. A mi madre, Roxana, quién me acompañó en todo el proceso, dándome ánimos en los momentos más difíciles y complejos. A mi novia, Laura, quién siempre logró calmarme el estrés y la presión con su gran amor y paciencia, ayudándome a ordenar mis pensamientos y dar lo mejor de mí. A toda mi familia y seres queridos, fueron, son y serán siempre mi principal motivación.

Le agradezco también al Departamento de Ciencias de la Computación, especialmente a sus funcionarios, dispuestos a ayudar de manera atenta y cariñosa. Me gustaría mencionar específicamente a Sandra, Carolina, Ana y Paulette, quienes sin importar cuántas veces fuese a preguntar lo mismo, me respondían con una sonrisa en el rostro. Su orientación durante todo el proceso, a nivel académico y personal, fue vital para haber podido llegar a este punto. Por otro lado, me gustaría agradecer a mi Profesor Guía, Francisco, por la paciencia y dedicación al apoyarme en este trabajo. Finalmente, me gustaría mencionar a Bastián, Fabiola, Felipe y Nahuel, cuyos apoyos y aportes fueron fundamentales para el desarrollo de este trabajo de tesis.

Este trabajo de tesis ha sido parcialmente financiado por el proyecto Fondecyt Nº 11190248.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

El fomento del pensamiento computacional (PC) en niñas y niños se ha convertido en un tópico de interés mundial, dados los procesos cognitivos que promueve en los estudiantes [10]. Las habilidades que desarrolla el PC, como descomposición, abstracción, reconocimiento de patrones y creación de algoritmos, refuerzan habilidades orientadas a la resolución de problemas, como el pensamiento crítico, la creatividad y la colaboración [57].

La ACM (Association for Computing Machinery) promueve que los conocimientos en el área computacional sean enseñados a estudiantes desde etapas muy tempranas de su desarrollo, permitiendo que puedan, eventualmente, crear y entender sus propias tecnologías [14]. Según Wing [74], el pensamiento computacional está definido como los procesos que llevan a la concepción, desarrollo y resolución de problemas teniendo en cuenta el comportamiento humano. Es por esto que a nivel mundial se han realizado diversas propuestas para poder incluir planes de estudio y desarrollo de esta habilidad en colegios [10].

Un ejemplo de esto es la guía K-12 Computer Science Framework, que presenta una serie de modelos sobre cómo enseñar ciencias de la computación a niños y niñas desde la etapa preescolar a el equivalente a cuarto medio en Chile [14]. Además, existe la ONG CODE, cuyo fin es incluir a la informática como un ramo obligatorio en la educación primaria y secundaria [2]. Una de las iniciativas de esta ONG es "La Hora del Código" [1], que imparte cursos gratuitos de programación para estudiantes en etapa escolar.

Todas estas iniciativas surgen con el propósito de impartir cursos de pensamiento computacional a niños y niñas en etapa escolar, de tal manera que puedan desarrollar las habilidades antes descritas. Sin embargo, la gran parte de las herramientas computacionales que permiten desarrollar estas habilidades, usan lenguajes de programación basados en bloques y un paradigma de interacción principal basado en Point & Click [37]. Esto implica que la forma en la que interactúan con el computador sigue siendo la misma: Mouse y teclado.

Una forma de romper este paradigma es incorporar dispositivos de entrada no convencionales al momento de interactuar con el computador, puesto que permiten romper con la limitación impuesta por dispositivos tradicionales. Un ejemplo de esto son los controles de

[1] https://horadelcodigo.cl/

movimiento, que le permiten al usuario manejar un sistema usando sólo su cuerpo. Actualmente, estos se encuentran en diversos dispositivos ya usados por niños y niñas en etapa escolar, tales como smartphones o consolas de videojuegos. En el ámbito educativo, la implementación de controles de movimiento ha sido positiva, especialmente entre niños y niñas en etapa escolar [3]. Sin embargo, un problema común encontrado en estas iniciativas es la falta de precisión del dispositivo que permite captar los movimientos, dificultando así su uso [1].

1.1. Problema Abordado

En Chile, programación no es una materia que se encuentre siempre en la malla curricular de los colegios [28]. Esto deriva en que niños y niñas en etapa escolar tengan una visión distorsionada sobre cómo funcionan los computadores y cómo se pueden usar para abordar problemas [27]. Una forma de resolver esto ha sido impartir talleres de programación orientados a niños y niñas con el fin de desarrollar habilidades computacionales entre los estudiantes, en donde Scratch ha tomado protagonismo [28].

Sin embargo, se plantea la interrogante de cómo perciben los niños la programación, y si encuentran un valor en ello. De acuerdo a un estudio [27], los niños relacionan la programación como sólo computadores con estereotipos relativos al área presentes, como que la computación es sólo para niños o que *hackear* es lo mismo que *programar*. Aunque el tema despierta la curiosidad de los niños, generando interés en explorar más en el área, el aprendizaje de un nuevo lenguaje no solo implica abordar problemas lógicos complejos, sino también comprender una sintaxis y semántica nuevas para el estudiante. Esta percepción hace que aprender a programar se perciba como una tarea intimidante y compleja, incluso antes de comenzar [51]. A esto se le conoce como **barrera de entrada a la programación** [38]. Para reducir esta barrera, es necesario abordar la sensación de inseguridad e intimidación del estudiante, estimulando su motivación [61].

De acuerdo la Teoría de Expectativa-Valor, la motivación de una persona para realizar una tarea depende directamente de dos factores importantes: Expectativa y Valor [56]. El primero se refiere a la posibilidad de éxito en una tarea y la satisfacción que esta pueda traer. El segundo, en cambio, se refiere a: (1) la importancia de hacerlo bien, (2) el valor intrínseco, es decir, el interés y disfrute de la tarea como tal, (3) la utilidad y (4) el costo. En particular, la motivación depende directamente de ambos factores. Incluso, si uno de los dos es muy alto en comparación, puede llegar a influir en el otro. Esto significa que, por ejemplo, si una tarea es lo suficientemente entretenida (Alto valor), pero difícil (Baja expectativa), puede generar la motivación suficiente a la persona para cumplirla de todos modos. Del mismo modo, si a esto le sumamos que la actividad o tarea en cuestión se da en un ambiente de aprendizaje informal (Informal Learning) , puede aumentar no sólo el valor percibido por el estudiante, sino también su interés en seguir aprendiendo sobre el tema en un ambiente formal [5].

Actualmente, gran parte de las herramientas de programación orientadas a la educación en niños utilizan paradigmas basados en Point & Click [37]. Según la literatura, es posible potenciar el interés del estudiante interviniendo este el mecanismo de interacción. En particular, el uso de gestos y controles de movimiento ha demostrado ser bien aceptado por escolares en edades de 10 a 12 años en entornos educativos [3, 76]. Dado esto, existe un potencial para que la integración de controles de movimiento en ambientes educativos de programación au-

mente la motivación de los estudiantes para aprender estos temas, reduciendo así la barrera de entrada.

En este trabajo de tesis, se investigó el impacto del mecanismo de entrada como un primer acercamiento a la programación, centrándose específicamente en la comparación entre el uso de mouse y teclado frente a controles de movimiento. Para esto se hizo uso de *Gesture Coding* [43], un lenguaje de programación experimental basado en bloques controlado mediante gestos, desarrollado de manera interna. De manera similar a Scratch, Gesture Coding está basado en bloques, es decir, cada comando consiste en un bloque de color que se conecta a otro y permite escribir un programa. La principal diferencia es que este último es controlado por gestos, es decir, usa controles de movimiento como mecanismo de control dentro de la aplicación, específicamente para crear nuevos bloques.

Al comienzo de este trabajo, Gesture Coding se encontraba en etapa de prototipo funcional y sólo soportaba bloques de tipo secuencial y condicional [43]. Luego, no era posible abarcar el conjunto minimal de habilidades descrito anteriormente [52]. Es por esto que resultó necesario extender el soporte del lenguaje a bloques de variables y ciclos.

Dado esto, se propone que haciendo uso de controles de movimiento familiares en niños, como son los Joy-Con, sumado a la naturaleza informal del proceso de aprendizaje, es posible aumentar el valor intrínseco percibido por ellos; es decir, que la actividad resulte ser mas entretenida o lúdica. Luego, de acuerdo al Teoría Expectativa-Valor, la motivación puede aumentar a pesar de una baja expectativa. Esto significa que, de acuerdo a la teoría, un alto valor percibido, especialmente valor intrínseco, puede eventualmente sobreponerse a una baja expectativa.

El trabajo realizado presenta contribuciones en la línea de Interacción Humano Computador, mediante la evaluación y aplicación de metáforas de interacción (es decir, modelos cognitivos que pueden influir en el diseño de interfaces), y Educación en Ciencias de la Computación, al proponer y evaluar en terreno nuevas formas para potenciar la motivación de los estudiantes en ambientes educativos y de computación. Finalmente, este trabajo de tesis permitiría contribuir al estado del arte presentando y desarrollando una técnica alternativa que permita reducir la barrera de entrada al aprendizaje de pensamiento computacional, como son el uso de gestos y controles de movimiento.

1.2. Hipótesis

En base a lo planteado anteriormente, se presentan las siguientes hipótesis que se explorarán en este estudio:

H1. El uso de Joy-Cons es efectivo para reducir la barrera de entrada en lenguajes de programación basados en bloques.

Dado lo anterior, se plantean las siguientes dos hipótesis de trabajo:

H1.1. El uso de Joy-Con como control de un lenguaje de programación genera un mayor valor percibido por parte de niños y niñas en comparación con mecanismos de control tradicionales, como lo son el mouse y teclado en Scratch.

H1.2. El uso de Joy-Con como control de un lenguaje de programación genera una

mayor satisfacción de uso por parte de los niños y niñas en comparación a inputs tradicionales, a pesar de requerir mayor esfuerzo físico.

1.3. Preguntas de Investigación

En base a las hipótesis planteadas anteriormente, se derivan las siguientes preguntas de investigación:

RQ1 - ¿Cuál es el impacto del uso de Joy-Cons en la reducción de la barrera de entrada en lenguajes de programación basados en bloques? Esto da pie para estudiar la validez de la hipótesis H1.

RQ2 - ¿Cuál es el impacto del uso de Joy-Con como mecanismo de control de un lenguaje de programación en el valor percibido por parte de niños y niñas en etapa escolar? Esto da pie para estudiar la validez de la hipótesis H1.1

RQ3 - ¿Qué tan satisfactorio es el uso de un lenguaje de programación controlado por Joy-Cons en contraste con uno controlado por dispositivos de entrada tradicionales para niños y niñas de 10 y 12 años? Esto da pie para estudiar la validez de la hipótesis H1.2.

1.4. Objetivos

A continuación se describen los objetivos a cumplir en este trabajo de título.

1.4.1. General

Explorar el impacto de los mecanismos de control de navegación y entrada de tipo gestual en el valor percibido, facilidad de uso y potencial de adopción embebidos en el lenguaje Gesture Coding para suavizar la barrera de entrada para aprender a programar en niños y niñas de 10 a 12 años.

1.4.2. Específicos

1. Refinar metáforas de interacción de Gesture Coding de tal manera que sean adoptables por niños y niñas de entre 10 y 12 años en etapa escolar.
2. Extender el sistema de bloques de Gesture Coding, de tal manera que permita abarcar los conceptos de ciclos y variables.
3. Evaluar la adopción y valor percibido del uso de Joy-Cons como mecanismo de entrada de Gesture Coding en comparación a mecanismos de entrada tradicionales, por parte de niños y niñas en etapa escolar.

1.5. Solución

Este trabajo de tesis se enfoca en evaluar el impacto del mecanismo de entrada en la percepción de niñas y niños chilenos al programar, sin haber tenido experiencia previa. Para

llevar a cabo esta evaluación, se contrastan mecanismos de entrada convencionales, como el mouse y el teclado, con enfoques no convencionales, como los controles de movimiento. Utilizamos la aplicación Scratch para implementar los mecanismos de entrada tradicionales y Gesture Coding para los mecanismos de entrada no convencionales, dado que usa controles de movimiento como mecanismo de entrada.

Para esto, en primer lugar fue necesario realizar un estudio piloto exploratorio con la primera versión de Gesture Coding, es decir, usando sólo Secuencialidad y Condicionalidad. Esta se enfocó en medir la carga cognitiva que implica el aprender a programar en niños y el potencial de aceptación de la herramienta en escolares. Este estudio arrojó que los participantes que tuvieron su primera experiencia programando con Gesture Coding y controles de movimiento presentaron mayor interés en seguir con la actividad en comparación a sus pares que programaron con mecanismos de entrada tradicionales, sin diferencia significativa en cuanto a cargas cognitivas. Este estudio, si bien no generalizable dado el número de individuos estudiados, permite plantear que existe una oportunidad de desarrollo de valor percibido por los niños al usar un mecanismo de entrada no tradicional, en este caso, controles de movimiento.

Usando los resultados de este estudio como base, se procedió a refinar y extender la funcionalidad de Gesture Coding. Para esto se tomó de base el conjunto minimal de habilidades de desarrollo de pensamiento computacional planteado por Rich et al. [52]: Secuencialidad, Condicionalidad, Ciclos y Variables. Luego, para cubrir el conjunto minimal de habilidades desarrolladas por el pensamiento computacional fue necesario integrar Ciclos y Variables. Tomando como referencia las funcionalidades ya desarrolladas en el lenguaje, se diseñaron e implementaron nuevos gestos y bloques. Con esta extensión, un estudiante debe ser capaz de adquirir las habilidades antes desarrolladas de manera similar a cómo lo haría con otros lenguajes de bloques como Scratch.

Después de la ampliación del lenguaje, se llevó a cabo un piloto similar tras completar la extensión de Gesture Coding, que abarca el conjunto mínimo de habilidades de desarrollo de pensamiento computacional. El propósito de este piloto fue evaluar la plausibilidad de la metodología de trabajo y la funcionalidad de la extensión, verificando su correcto funcionamiento y su comprensibilidad. Este piloto se llevó a cabo con estudiantes de primer año del curso de Introducción a la Programación, ya que el objetivo del estudio no depende de la muestra utilizada para la evaluación. Los resultados indicaron que, aunque al principio resultaba complicado familiarizarse con los controles, la naturaleza lúdica del programa contribuyó a mantener el interés de los participantes y captar su atención.

Finalmente, tomando como base el piloto con Gesture Coding extendido, se lleva a cabo un caso de estudio [77] con el fin de evaluar el impacto del mecanismo de entrada de la herramienta en el valor percibido y la expectativa de escolares de entre 10 y 12 años. Esto se llevó a cabo con dos intervenciones. En primer lugar, se organizaron cuatro talleres de programación con niños de edades comprendidas entre los 10 y 12 años. En dos de estas sesiones, se utilizó Scratch con controles tradicionales, mientras que en las dos sesiones restantes empleamos Gesture Coding con controles de movimiento. Luego, se realiza una segunda intervención con la intención de profundizar los resultados obtenidos. Esta consistió en una segunda iteración del taller con un sólo participante por sesión siguiendo el protocolo Thinking Aloud. De este

caso de estudio se obtuvieron dos resultados principales: (1) El uso de gestos permite aumentar la Expectativa respecto a la programación; y (2) El uso de Joy-Con de Nintendo Switch permite aumentar el Valor Percibido de los estudiantes, al ser percibido como más lúdico y disfrutable. Esto, de acuerdo con la Teoría Expectativa-Valor, deriva en que el uso de gestos efectivamente potencia la motivación de los estudiantes e interés de escolares de entre 10 y 12 años.

1.6. Metodología

El desarrollo de este trabajo se llevó a cabo de manera incremental mediante la aplicación de la metodología Investigación-Acción [31]. Esta metodología propone un flujo de trabajo que, basado en una planificación inicial, se modifica según los hallazgos encontrados durante la investigación, refinando así el problema a medida que avanza el estudio. La elección de esta metodología se debe a la naturaleza exploratoria del trabajo, que busca comprender el desarrollo del fenómeno de percepción por parte de los estudiantes. Posteriormente, en torno a los Objetivos Específicos, se adaptó en tres espiras incrementales con un enfoque en (1) Análisis; (2) Implementación; y (3) Evaluación.

1.6.1. Espira 1 - Evaluación Preliminar y Familiarización

La primera espira de este trabajo consistió en evaluar y determinar el estado actual del lenguaje Gesture Coding, abarcando aspectos de diseño, implementación y su desempeño en la experiencia del usuario. Para llevar a cabo esta evaluación, se inició por familiarizarse con el código de Gesture Coding en Unity, al mismo tiempo que se revisó la documentación que facilita la detección de gestos libres mediante el uso de Joy-Cons.

Simultáneamente, se llevó a cabo un piloto inicial del lenguaje con niños de edades comprendidas entre los 10 y 12 años, centrándose exclusivamente en las habilidades de Secuencialidad y Condicionalidad, que son competencias ya presentes en Gesture Coding. Se midieron las cargas cognitivas y físicas, así como el tiempo que les llevó resolver una actividad. Este piloto se realizó con un total de 99 niños, divididos en tres grupos: uno que utilizó Gesture Coding sin controles de movimiento, otro con Gesture Coding que incluyó controles de movimiento y un tercer grupo que utilizó Scratch.

Del piloto inicial se desprende que Gesture Coding es competente al momento de enseñar pensamiento computacional, obteniendo resultados similares a los obtenidos usando Scratch en cuanto a tiempo de desempeño, a excepción de la versión de Gesture Coding que incluye controles de movimiento, la cual requirió más tiempo debido a la necesidad de adaptarse a dicho control. En cuanto a la comparación de cargas, el estudio arrojó que las cargas cognitivas se mantienen a un nivel similar entre las tres variantes. Sin embargo, la carga física resultó significativamente mayor en el caso de Gesture Coding con controles de movimiento, dada a su naturaleza.

1.6.2. Espira 2 - Diseño e Implementación de la Extensión

Después de completar el piloto inicial, se procedió al diseño e implementación de la extensión del lenguaje. Para lograr esto, se introducen dos nuevas categorías de bloques: variables

y ciclos. Esto implica determinar el gesto que caracterizará cada categoría, basándose en dos parámetros principales: la facilidad de ejecución y la coherencia con su significado. En el caso de las variables, se utiliza la metáfora de que funcionan como contenedores de valores, por lo que se eligió representarlas mediante el gesto de la letra "V", haciendo alusión a un contenedor visto de lado. En cuanto a los ciclos, se eligió representarlos con el gesto de la letra "Z", ya que dibujarla implica un movimiento horizontal repetitivo que se asemeja al concepto de un ciclo.

Además, para utilizar las variables, se hizo necesario diseñar una nueva interfaz que permitiera seleccionar las variables previamente definidas. Finalmente, la extensión se implementa utilizando Unity.

Una vez implementada la extensión, se lleva a cabo un nuevo piloto para realizar pruebas de funcionalidad y plausibilidad de la metodología de evaluación. Para esto, se reclutan 20 estudiantes de primer año de Universidad, a los que se les pide usar la plataforma para luego responder preguntas al respecto. Dados los objetivos particulares de esta intervención piloto, el hecho de que la muestra no forme parte del público objetivo de la herramienta no impacta en los resultados. Finalmente, en base a la retroalimentación obtenida durante esta fase, se refina y ajusta la plataforma en preparación para la evaluación final.

1.6.3. Espira 3 - Evaluación y Validación de la Herramienta con Usuarios

Finalmente, en la última espira de este trabajo se planifica y ejecuta la evaluación final de la herramienta. En esta evaluación, se compara Scratch, que utiliza métodos de entrada tradicionales como el mouse y el teclado, con Gesture Coding, que emplea la versión extendida que incorpora nuevos gestos. Se crea un taller de programación básica centrado en la herramienta a evaluar, poniendo un énfasis en la exploración de bloques para la resolución de problemas. El taller se lleva a cabo en cuatro ocasiones con grupos diferentes de niños de entre 10 y 12 años: dos utilizando Scratch y otras dos con Gesture Coding.

La evaluación tiene como objetivo medir las expectativas de los estudiantes y el valor percibido de la herramienta antes y después de su uso. Además, se miden las cargas físicas y cognitivas de los estudiantes al finalizar el taller. Las conclusiones obtenidas del taller permitieron analizar el comportamiento de los estudiantes frente a dos mecanismos de interacción distintos. Sin embargo, se decide refinar la evidencia encontrada haciendo una última ronda de talleres de manera particular con estudiantes utilizando la metodología Thinking Aloud. De esta evaluación se concluye que el uso de gestos y controles de movimeinto: (1) permite aumentar la Expectativa de los escolares al potenciar su confianza en los conocimientos adquiridos; y (2) permite aumentar el Valor Percibido por los escolares, al percibir la actividad como más lúdica y disfrutable. De esto se deriva que integrar gestos y controles de movimiento efectivamente permite aumentar la motivación de los estudiantes.

1.7. Estructura del Documento

La estructura del documento es la siguiente:

- Capítulo 2 - Marco Teórico: Describe el marco teórico general detrás del trabajo de tesis y discute trabajos relacionados a este, como otras instancias de Informal Learning, mecanismos de interacción no convencionales y la Teoría Expectativa-Valor.
- Capítulo 3 - Espira 1: Muestra el análisis preliminar de Gesture Coding y Scratch, contrastando sus interfaces y metáforas de interacción.
- Capítulo 4 - Espira 2: Presenta el proceso de extensión de la herramienta, así como la justificación detrás de decisiones de diseño tomadas para refinar la versión original de la plataforma. Además, se detalla el proceso del piloto original.
- Capítulo 5 - Espira 3: Describe el proceso de Caso de Estudio con usuarios y Thinking Aloud, junto a sus respectivos resultados.
- Capítulo 6 - Discusión: Se analizan los resultados obtenidos y sus implicancias.
- Capítulo 7 - Conclusión: Se presentan los principales hallazgos, se verifica la validez de la hipótesis inicial y posibles trabajos futuros.

Capítulo 2

Marco Teórico

La motivación de una persona se define como el conjunto de factores, ya sean internos o externos, que en parte determinan las acciones de un individuo, según la Real Academia Española (RAE) [1]. Una estrategia para reducir la barrera de entrada al desarrollo del pensamiento computacional es influir en la motivación del individuo, de manera que esta barrera se perciba menos como un obstáculo y más como un desafío que se puede superar.

Actualmente, la mayoría de los lenguajes de programación usan como mecanismo de entrada convencionales como Mouse y Teclado, por lo que se explora el impacto de mecanismos no convencionales en la motivación del individuo[37]. El marco teórico de este trabajo incluye teorías sobre la motivación y el impacto de los mecanismos de entrada no convencionales en el uso de la tecnología y en el proceso de aprendizaje, con un enfoque particular en el uso de gestos. Estos gestos han sido ampliamente utilizados como mecanismos de control en videojuegos [1, 3], por lo que se explora su influencia en la aceptación y el valor percibido por parte de los usuarios. Finalmente, se analizan tanto las metodologías tradicionales como los enfoques no convencionales para el desarrollo del pensamiento computacional en niños.

2.1. Motivación y Frustración

El aprendizaje del pensamiento computacional puede resultar intimidante para un estudiante, ya que, además de tener que resolver problemas lógicos complejos, deben adaptarse al lenguaje utilizado por la computadora [38]. Si bien las computadoras son muy eficientes en seguir instrucciones, no tienen la capacidad de determinar si lo que hacen es correcto o no. En este punto, la mente humana entra en conflicto con la forma en que las computadoras procesan la información, lo que puede provocar una disminución de la motivación y aumentar la frustración del usuario [48].

Parte de esta percepción por parte de los estudiantes de que programar resulta intimidante proviene de concepciones erróneas sobre lo que implica programar y cómo se lleva a cabo[28], a menudo influenciados por representaciones en los medios digitales o la ciencia ficción. Esto

[1]REAL ACADEMIA ESPAÑOLA: Diccionario de la lengua española, 23.ª ed., [versión 23.7 en línea]. `https://dle.rae.es`. [Noviembre de 2023].

impacta en la sensación de auto-eficacia del estudiante [46], ya que se generan expectativas muy altas al momento de desarrollar una nueva habilidad o conocimiento. Una de las principales aproximaciones que explica este fenómeno es la Teoría Expectativa-Valor[73], la cual identifica los factores que influyen en el desempeño y persistencia de un individuo al abordar una tarea específica.

Esta teoría, inicialmente adaptada al ámbito educacional por Eccles et al. en la década de los 80 [18, 19], explica cómo los estudiantes de edades K-12 (entre los 7 y 17 años) pueden tomar decisiones y cuál será su desempeño en ambientes académicos. Esta teoría indica que la motivación de una persona para desarrollar una tarea está íntimamente relacionada con la expectativa del estudiante y el valor que encuentre en la tarea.

La **expectativa** se refiere a la percepción que tiene el estudiante sobre su desempeño en una tarea. Esto significa que si un estudiante se siente capaz de llevar a cabo una tarea, tendrá una alta expectativa. Esto está relacionado con la auto-eficacia percibida por el individuo respecto a la habilidad que está desarrollando, es decir, qué tan probable es que cumpla la tarea de manera exitosa obteniendo un buen resultado [73]. La sutil diferencia radica en que la expectativa se relaciona con la auto-percepción de la persona y su capacidad para llevar a cabo la tarea, mientras que la auto-eficacia se centra en el resultado de la tarea en lugar de en la capacidad de la persona para realizarla.

Por otro lado, el **valor percibido** por el estudiante se refiere a la percepción subjetiva que tiene el individuo del beneficio de llevar a cabo la tarea. Este se compone de cuatro dimensiones:

- **Importancia**: Se refiere al valor percibido desde el punto de vista identitario o de comunidad. Puede ser, por ejemplo, el desarrollo de una habilidad en un contexto de prestigio o reputación, como ganar un torneo de programación.
- **Valor intrínseco**: Hace referencia al interés o disfrute al realizar la tarea. Por ejemplo, desarrollar una habilidad porque resulta divertido para la persona o disfruta de la actividad donde la aplica.
- **Utilidad**: Se relaciona con la utilidad percibida por el usuario al desarrollar esta habilidad o labor. Por ejemplo, saber que esta habilidad le será útil en el futuro en un trabajo u otra actividad asociada.
- **Costo**: Incluye el tiempo, esfuerzo y otros costos asociados al desarrollo de una habilidad o tarea. Por ejemplo, el costo monetario de realizar un curso, la duración de este y los elementos que se sacrifican para desarrollar esta habilidad.

Según la teoría, la expectativa y el valor percibido interactúan entre sí para predecir la motivación de una persona al realizar una tarea específica. Cuando tanto la expectativa como el valor percibido son altos, el individuo está altamente motivado para llevar a cabo la tarea. Por otro lado, si ambas están en niveles bajos, la motivación también será baja. Sin embargo, cuando la expectativa y el valor percibido no están en niveles similares, se presentan dos escenarios posibles:

- Si la expectativa es alta pero el valor percibido es bajo, la motivación se mantendrá baja. Esto puede asociarse a una tarea que es fácil de realizar, pero que no resulta muy

importante o interesante para el usuario, como una tarea aburrida. En consecuencia, la motivación para llevarla a cabo será baja.

- Si el valor percibido es alto pero la expectativa es baja, la motivación será baja a menos que el valor percibido aumente significativamente. Esto se interpreta como una tarea que es difícil de realizar pero que resulta útil o entretenida para el usuario, siendo percibida como un desafío. Sin embargo, existen casos en los que una expectativa demasiado baja puede disminuir incluso el valor percibido. Por ejemplo, si una tarea es divertida pero extremadamente difícil de realizar, puede dejar de ser percibida como divertida y pasar a ser frustrante.

Dados estos casos, existen aproximaciones para motivar estudiantes aumentando la expectativa o el valor percibido. Un ejemplo de esto es disminuir el costo de realizar la actividad, ya sea en tiempo o en dinero, o aumentar la utilidad percibida aplicando lo aprendido en la vida cotidiana del estudiante [7, 8, 35, 72].

2.2. Aprendiendo a Programar

Actualmente, el desarrollo de habilidades de pensamiento computacional es una práctica que va tomando fuerza entre las comunidades educativas, integrando programas de computación a las mallas curriculares en colegios [28]. En este contexto, se han desarrollado numerosas iniciativas para acercar la programación a estudiantes, suavizando la barrera inicial de entrada haciendo que el lenguaje sea más cercano a ellos [27].

Al momento de desarrollar habilidades de Pensamiento Computacional, una de las principales barreras de entrada es la baja expectativa de parte de los estudiantes[22]. En efecto, dadas las numerosas ideas erróneas sobre la disciplina adoptadas de la poca información y la cultura popular en general, los estudiantes tienen una percepción de que programar es una tarea difícil[48]. Esta sensación se potencia al momento de comenzar a aprender dado que no sólo deben enfrentarse a problemas matemáticos y lógicos complejos, sino que deben resolverlos mientras aprenden un nuevo lenguaje que el computador pueda entender. Esto implica que, además de entender este nuevo lenguaje, deben desarrollar una suerte de intuición para poder comunicarse con el computador. No basta con aprender el lenguaje, sino que además deben aprender a interpretarlo como lo haría una máquina[38]. Dado que el valor percibido de aprender habilidades de pensamiento computacional varía de individuo en individuo, la motivación es un tema necesario de abordar para superar la barrera de entrada antes mencionada.

Cuando se trata de aprender Pensamiento Computacional, los jóvenes programadores suelen cometer errores en alguna de las tres etapas: el Diseño, que implica conceptualizar y resolver el problema; la Generación, que es la implementación de la solución encontrada; y la Evaluación, que consiste en revisar errores y realizar pruebas[54]. De estas tres etapas, donde los estudiantes suelen encontrar más dificultades es en la primera[45], la etapa de Diseño. Esto se debe principalmente a que la principal barrera inicial para aprender a programar no es una sintaxis o semántica específica de los lenguajes, sino la habilidad para comunicarse con el computador y resolver problemas. Por lo tanto, se recomienda comenzar la enseñanza de programación a una edad temprana utilizando lenguajes que cuenten con una fuerte representación gráfica, es decir, aquellos que utilizan una interfaz que se ajusta a los modelos

mentales de los estudiantes [24, 53]. Estos son conocidos como lenguajes visuales o basados en bloques.

A lo largo de los años, se han propuesto diversas aproximaciones para acercar el proceso a los estudiantes, ya sea influyendo en la expectativa o en el valor percibido por parte de los estudiantes para aumentar la motivación [22]. Ejemplos de estas se describen a continuación.

2.2.1. Lenguajes Visuales

La programación visual o por bloques permite programar usando representaciones visuales de lo que se quiere implementar. Para esto, se hace una simplificación de la sintaxis para que los comandos sean más claros de entender [59]. Esto se complementa con representaciones visuales del comando utilizado, que puede ser una forma o color para categorizar las declaraciones. Esto es encapsulado en un bloque que permite generar una acción en particular. De esta forma, el estudiante es capaz de programar buscando el bloque del color o forma que necesite, y luego encontrar el comando específico que necesita usar. Finalmente, cada bloque puede conectarse entre sí para terminar formando código que luego es interpretado por el computador para poder ser ejecutado.

La principal ventaja del uso de bloques en programación es que permite enfocar el aprendizaje en la noción de programar, es decir, ayuda a reducir esa barrera de entrada al familiarizarse cómo piensa un computador y cómo procesa información. Por ejemplo, en el estudio de Weintrop y Wilensky [68] se analiza el desempeño de estudiantes contrastando lenguajes textuales con lenguajes visuales, en este caso Java y Snap!. Los resultados arrojaron que el lenguaje por bloques permitía un mejor desempeño de parte de los estudiantes al momento de entender cómo funcionaba su código. Del mismo modo, en un estudio posterior de los mismos autores, se identifican prestaciones específicas de los bloques que permiten desarrollar esa noción en los estudiantes. Por ejemplo, la forma de los bloques condicionales del estilo *if/else* ayuda a entender que se ejecutará sólo una rama dependiendo de la condición, cosa que no quedaba tan clara en lenguajes textuales [69].

A pesar de lo anterior, existen estudios que muestran que, si bien el desempeño inicial es mejor en lenguajes visuales basados en bloques, a largo plazo se hace más difícil entender conceptos más complejos de programación [81]. A esto se suma que, a largo plazo, la transición a lenguajes textuales resulte más compleja por el cambio de sintaxis [4]. Es por esto que los lenguajes visuales son ampliamente usados en estudiantes de rango etario K-12 como primera aproximación a desarrollar habilidades de pensamiento computacional, dado que ayudan a reducir la barrera de entrada.

Existen numerosos lenguajes que aplican este tipo de lenguaje en su sistema para subir la expectativa del estudiante, es decir, que perciba la tarea como algo más fácil o amigable [59]. Ejemplos de esto son Scratch[2], Blockly[3] o Snap![4]. Dado esto, es posible que Gesture Coding potencie la motivación de los estudiantes al aplicar lenguajes visuales en el proceso de aprendizaje.

[2] `https://scratch.mit.edu/`
[3] `https://developers.google.com/blockly/`
[4] `https://www.nintendo.co.uk/Nintendo-Labo/Nintendo-Labo-1328637.html`

2.2.2. Computación Física

Se entiende por computación física a programar sistemas interactivos que permiten interactuar con el mundo real [67]. Esto se logra gracias al uso de sensores y actuadores que nos permite tomar decisiones basadas en el contexto en el que nos encontremos. Por un lado, los sensores nos permiten entender diversos fenómenos que pueden ocurrir a nuestro alrededor, como ruidos, movimiento o ubicación. Con esta información, podemos programar nuestro sistema para que genere una acción e impacte en nuestro entorno, usando los actuadores. Estos van desde parlantes, pantallas a incluso ruedas y robots.

El uso de computación física en un ambiente de aprendizaje impacta en el valor percibido de los estudiantes, particularmente en el valor intrínseco dado que se percibe como una actividad más divertida o disfrutable [41]. Además, el hecho de ver el resultado del sistema impactar en el mundo real, ya sea hacer que un robot se mueva o encender alguna luz, aumenta en la expectativa del estudiante dado que se percibe como algo "lograble" [75]. Uno de los principales factores de esto es que el estudiante tiene algo tangible y una meta clara en el mundo real. Sin embargo, esto implica que el proceso puede ser mas lento debido a posibles problemas o ajustes mecánicos en el dispositivo, que resultan externos al código [42].

Una de las principales desventajas de la computación física radica en la necesidad de contar con dispositivos específicos para su implementación, lo cual implica un costo monetario significativo [40]. A su vez, la computación física no presenta una diferencia sustancial en el desarrollo de habilidades de pensamiento computacional en comparación con la computación puramente virtual [32]. Por lo tanto, una de sus principales ventajas recae en la capacidad de motivar a los estudiantes.

2.2.3. Metodologías Unplugged

Otra forma de fomentar el desarrollo de habilidades de pensamiento computacional es mediante la implementación de actividades lúdicas con los estudiantes, pero sin utilizar un computador. Estas actividades, conocidas como "Unplugged" [6]; generalmente se han aplicado con mayor frecuencia en niños muy pequeños, quienes aún no tienen la destreza para manejarse en un entorno informático, como lo es el piloto desarrollado por Garting-Daugs et al [25]. Aunque las actividades Unplugged suelen resultar atractivas y entretenidas para los estudiantes, es importante considerar que algunos pueden percibirlas más como una extensión del desarrollo matemático que desde una perspectiva computacional.

En el estudio de Taub et al. [63], los estudiantes informaron haber disfrutado de las actividades Unplugged y haber comprendido la materia sin dificultad aparente. Sin embargo, a pesar de haber adquirido habilidades de pensamiento computacional, estos presentaron un mayor interés en la componente matemática que en la de ciencias de la computación. Dado esto, el autor recomienda enfocar las actividades en conceptos fundamentales de las ciencias de la computación. En estudios posteriores, esta aproximación ha tenido una mejor recepción y desempeño por parte de los estudiantes [55].

En comparación con las metodologías más tradicionales de enseñanza, no se percibe una diferencia significativa en la asimilación de contenidos [33], lo que indica que el enfoque Unplugged es un método viable para desarrollar estas habilidades en los estudiantes. No

obstante, aún no existe un consenso claro sobre la viabilidad de incorporar estas metodologías Unplugged en los programas escolares, ya que pueden surgir diferencias culturales a la hora de aprender ciencias de la computación,[22] lo que dificulta la validación de los instrumentos de enseñanza.

Esta aproximación no solo maneja la expectativa de los estudiantes al aterrizar conceptos complejos como la abstracción y la resolución de problemas, sino que además aumenta el valor percibido por los estudiantes al desarrollar actividades lúdicas [50]. Esto les permite disfrutar de una experiencia más placentera, centrándose específicamente en el valor intrínseco de la materia, lo que se traduce en una mayor motivación por parte de los estudiantes al integrar los conocimientos adquiridos.

2.2.4. Aprendizaje Informal

Otras metodologías no convencionales para el desarrollo de Pensamiento Computacional son actividades enfocadas fuera del aula. Estas, al igual que las anteriores, también se enfocan en aumentar tanto la expectativa del estudiante como su valor percibido. Sin embargo, la principal diferencia es que, como son actividades fuera del aula, están menos estructuradas y menos rígidas, por lo que el estudiante puede explorar las herramientas y los desafíos planteados a un ritmo que le acomode [44]. En palabras de Seymour Papert [49], en el aprendizaje informal es el estudiante el que se encuentra en control de su aprendizaje, no el profesor. En ciencias de la computación, el aprendizaje informal puede implementarse mediante aprendizaje online o campamentos de programación, donde los estudiantes van siguiendo una guía general de contenidos manteniendo la "independencia de aprendizaje" [26].

Esta práctica permite que el estudiante maneje su propia expectativa [16], dado que las metas son auto-impuestas, en contraste a las metas que son entregadas por un docente. De esta manera, la motivación de aprender o desarrollar una habilidad en particular viene desde el estudiante. Esto también impacta en el valor percibido por el estudiante [13], especialmente en la Utilidad de lo aprendido, dado que es el propio estudiante quien elige qué aprender en función de sus intereses. Esto deriva en que el aprendizaje informal suele venir acompañado de una alta motivación de parte de los estudiantes [22]. Sin embargo, esto también implica que, si el estudiante no tiene interés en profundizar sus conocimientos, no lo hará hasta que sienta que le resulta útil [29]. Por esta razón, aplicar esta metodología para el desarrollo de habilidades de pensamiento computacional puede llevar a un estancamiento en el conocimiento del estudiante si no muestra interés en profundizar, quedándose así con el nivel de conocimiento alcanzado hasta el momento [11].

2.3. Mecanismos de Interacción

Al momento de desarrollar pensamiento computacional en niños, la gran mayoría de las veces se opta por metodologías tradicionales usando lenguajes visuales, es decir, programación en aula con actividades Plugged [69]. Gran parte de los lenguajes visuales usan como mecanismos de interacción exclusivamente el mouse y teclado del computador, por lo que la diferencia real entre uno y otro es en temas de interfaz y funcionamiento [37]. En esta arista, existe oportunidad de explorar cómo puede afectar el valor percibido y/o expectativa

del estudiante modificando el mecanismo de interacción, haciéndolo más interesante o más cómodo.

En un ambiente donde los mecanismos de interacción están constantemente diversificándose, observamos una oportunidad de explorar acercamientos novedosos. Un ejemplo de esto es AR-Maze [36], un entorno de Realidad Aumentada en el que los estudiantes pueden crear y conectar bloques para elaborar sus propios programas. Este proyecto está dirigido a niños de entre 5 y 9 años, brindando retroalimentación en tiempo real. La recepción del programa por parte de los estudiantes fue en su mayoría positiva, ya que mover los bloques directamente con sus manos les permitió experimentar una forma más intuitiva de programar.

Por otro lado, Yu et al. [78] implementaron CodeAttach, que consiste en un dispositivo interactivo, una aplicación móvil para interactuar con el dispositivo y materiales para diversas actividades. Su funcionamiento basado en Scratch está fuertemente apoyado por iconos, dado su enfoque en niños de entre 5 u 8 años. El sistema consta de instrumentos de juego que se pueden adherir a cualquier objeto utilizando una correa. Luego, haciendo uso de bloques de madera que simulan distintos comandos, los estudiantes pueden generar código que luego es traducido por la aplicación móvil. Este código les permite generar cambios en el dispositivo principal, como alternar luces de colores. El sistema hace una metáfora de los bloques de Scratch pero haciendo uso de computación física, con bloques tangibles.

Finalmente, BeadED Adventures [62] es un mecanismo que permite controlar software en una computadora utilizando contenedores de cuentas de diferentes colores. La idea principal es que el software narra una aventura en la que el usuario debe tomar decisiones, desarrollando su pensamiento computacional. Para tomar estas decisiones, el software presenta diferentes opciones al usuario, que se corresponden con el color de las cuentas físicas que se pueden ensartar en un collar o pulsera, simulando el camino recorrido. Este procedimiento es escaneado, generando comandos físicos para el software. Esto permite que el estudiante controle completamente la aplicación utilizando solo las cuentas que obtiene de los paquetes. La recepción del mecanismo por parte de los usuarios fue generalmente positiva, dado lo poco convencional de su control.

Aunque los mecanismos mencionados difieren en sus implementaciones y enfoques, todos parten de la misma base: desarrollar habilidades básicas de programación siguiendo un enfoque no convencional. Este enfoque termina generando un impacto ya sea en la expectativa del estudiante como en su valor percibido. En este caso en particular, lo poco convencional del mecanismo de control genera curiosidad por parte de los estudiantes, aumentando el interés o valor intrínseco.

Una forma de diversificar el mecanismo de interacción es mediante la integración de **Interfaces Naturales (NUI)**. La principal ventaja de este tipo de interfaces es que permiten controlar un sistema utilizando únicamente el cuerpo, prescindiendo de cualquier dispositivo mecánico [3]. Esto crea una sensación de interacción más natural para el usuario, ya que el sistema aprovecha acciones, movimientos y gestos que la persona ya conoce, como apuntar, mover los brazos o señalar con el dedo. Dentro de la categoría de gestos, destacan dos subtipos: Gestos Táctiles y Gestos Libres. Los Gestos Táctiles se refieren a los gestos realizados en una pantalla táctil, como la de un smartphone. La principal ventaja de estos gestos es que se llevan a cabo en un espacio concreto y delimitado, es decir, la propia pantalla, lo que

permite al usuario tener un control consciente sobre el alcance de sus gestos [1]. Por otro lado, los Gestos Libres no están restringidos a una pantalla, sino que pueden detectar todo el rango de movimiento del cuerpo del usuario. Estos gestos se pueden observar en cámaras de detección de movimiento, como Kinect, o en controladores que rastrean los movimientos de la persona, como los smartphones o los dispositivos de videojuegos. La principal ventaja de este tipo de gestos es que, al no estar limitados a una pantalla, resultan más intuitivos para el usuario, aunque requieren una mayor precisión [1].

Este tipo de tecnología se está integrando cada vez más en la vida cotidiana de las personas gracias al uso de dispositivos inteligentes, como smartphones, relojes y consolas de videojuegos [10]. Además, gracias al avance de la tecnología, el sistema de detección de estos gestos es cada vez más preciso, lo que hace que la interacción se sienta más natural, reduciendo la tasa de errores y aumentando la satisfacción percibida por parte del usuario [71].

A pesar de que el uso de gestos está bien documentado en las Interfaces Naturales, es importante tener en cuenta que los niños no interactúan de la misma manera con este tipo de interfaces, ya que sus habilidades motoras aún se encuentran en desarrollo. Si esta tecnología no se adapta para su uso en niños, el uso de gestos complejos puede tener el efecto contrario al esperado, aumentando la frustración. Un estudio realizado por Rahman et al. [1] comparó el rendimiento y la percepción de niños de 10 años al usar gestos táctiles y gestos libres. Este estudio reveló que, a pesar de requerir una mayor precisión para ejecutarse, los gestos libres se percibieron como más entretenidos e intuitivos que los gestos táctiles, lo que influyó en la motivación para usarlos. Esto puede interpretarse como una diferencia en las expectativas de ambos tipos de gestos, ya que los gestos libres se perciben como más intuitivos y lúdicos, además de un alza en el valor intrínseco de la plataforma, dado lo lúdico de usar gestos como mecanismo de interacción.

2.3.1. Síntesis

En este trabajo, se presenta Gesture Coding, lenguaje visual basado en Scratch cuyo mecanismo de interacción son los Joy-Con de Nintendo Switch, uno para cada mano. Estos controles constan de un acelerómetro y un giroscopio que permiten detectar tanto el movimiento del dispositivo como su orientación espacial. Haciendo uso de esta tecnología, se pueden detectar gestos libres hechos por el usuario, por ejemplo, al agitar el control o realizar cierto movimiento con las manos. Además, como mecanismo de feedback, estos controles poseen Vibración HD (HD Rumble) que permite hacer que el dispositivo vibre a diferentes frecuencias, logrando que incluso se reproduzcan sonidos.

Actualmente, Nintendo ha lanzado dos herramientas para la consola Nintendo Switch que permiten a los niños tener su primer contacto con la programación utilizando estos controles. Por un lado, Toy-Con Garage de Nintendo Labo [5] permite a los usuarios programar videojuegos utilizando los sensores incorporados en cada Joy-Con a través de conexiones de nodos. Por otro lado, Game Builder Garage [6] permite a los usuarios programar videojuegos mediante conexiones visuales de nodos, pero no utiliza los sensores de Joy-Con. Dado que

[5] `https://www.nintendo.co.uk/Nintendo-Labo/Nintendo-Labo-1328637.html`

[6] `https://www.nintendo.co.uk/Games/Nintendo-Switch-download-software/Game-Builder-Garage-1964648.html`

estas herramientas se dan en un contexto de videojuego, el valor intrínseco percibido por los usuarios es alto, es decir, se percibe como una actividad lúdica. Esto va de la mano, además, con una alta expectativa dada la naturaleza informal del posible aprendizaje.

La herramienta desarrollada en este trabajo difiere de estas herramientas en dos aspectos clave. En primer lugar, Gesture Coding no se centra en la programación de videojuegos, sino que busca reducir la barrera de entrada a los lenguajes de bloques. En segundo lugar, tiene como objetivo explorar el uso de mecanismos de interacción naturales, como los controles de movimiento, como una forma de fomentar la aceptación y la adopción por parte de los usuarios en este ámbito.

Capítulo 3

Espira 1: Análisis

En este capítulo, se introduce Gesture Coding como herramienta y se describe su interfaz y funcionamiento. Se analiza esta herramienta desde la perspectiva de la teoría Expectativa-Valor, explorando cómo sus componentes pueden influir en la motivación de los estudiantes, en comparación con lenguajes visuales más tradicionales, como Scratch. Finalmente, se presenta una primera prueba piloto de Gesture Coding con niños y niñas de edades comprendidas entre 10 y 12 años, con el objetivo de medir la carga cognitiva que implica usar la herramienta, así como su potencial de aceptación. A partir de los resultados de esta prueba, se establece un camino a seguir para el refinamiento y la expansión de la herramienta.

Gesture Coding es un prototipo de lenguaje de programación desarrollado para el sistema operativo Windows, utilizando el motor de desarrollo Unity. Este lenguaje está dirigido a un público de entre 10 y 12 años.

El diseño general de Gesture Coding se inspira en lenguajes de programación visuales como Scratch, Blockly o Snap!, dado que el uso de bloques simplifica la comprensión de la sintaxis y semántica, como se menciona en el Capítulo 2. La principal diferencia entre Gesture Coding y estos lenguajes radica en su dispositivo de entrada y su mecanismo de interacción.

El dispositivo de entrada utilizado en este lenguaje son los Joy-Con de Nintendo Switch, que constan de dos controladores: uno para la mano derecha (Joy-Con R) y otro para la izquierda (Joy-Con L). Además de contar con los botones y palancas estándar presentes en los controles de videojuegos, estos controladores son capaces de detectar los movimientos del usuario, lo que puede integrarse en la experiencia de uso. La conexión entre los Joy-Con y la PC se realiza a través de Bluetooth, y se establece una conexión por control.

Gesture Coding aprovecha estos controles para integrar el uso de gestos libres en el entorno de desarrollo, asignando un gesto específico a cada tipo de bloque, categorizándolos según su funcionalidad. El principal objetivo de esta herramienta es ofrecer una experiencia lúdica para los estudiantes que les permita suavizar la barrera de entrada a la programación, siendo este un primer acercamiento a lenguajes visuales como Scratch. Inicialmente, esta herramienta estaba limitada sólo a bloques de Movimiento, Condicionales y Operaciones Matemáticas. Luego, al finalizar este trabajo, fueron agregados las categorías de Repetición y Variables.

3.1. Interfaz de Usuario

La pantalla principal de Gesture Coding, mostrada en la Figura 3.1 está dividida en dos secciones: el panel izquierdo que corresponde al *Área de Trabajo* donde se posicionan los bloques para programar; y el panel derecho que es el *Área del Personaje* [1] donde se ejecutará el código. Al igual que en otros lenguajes visuales, los bloques tienen los comandos que se ejecutarán en el panel derecho. las instrucciones pueden ser ejecutadas pulsando el botón + en el Joy-Con R, o pulsando en el ícono Start en el panel derecho.

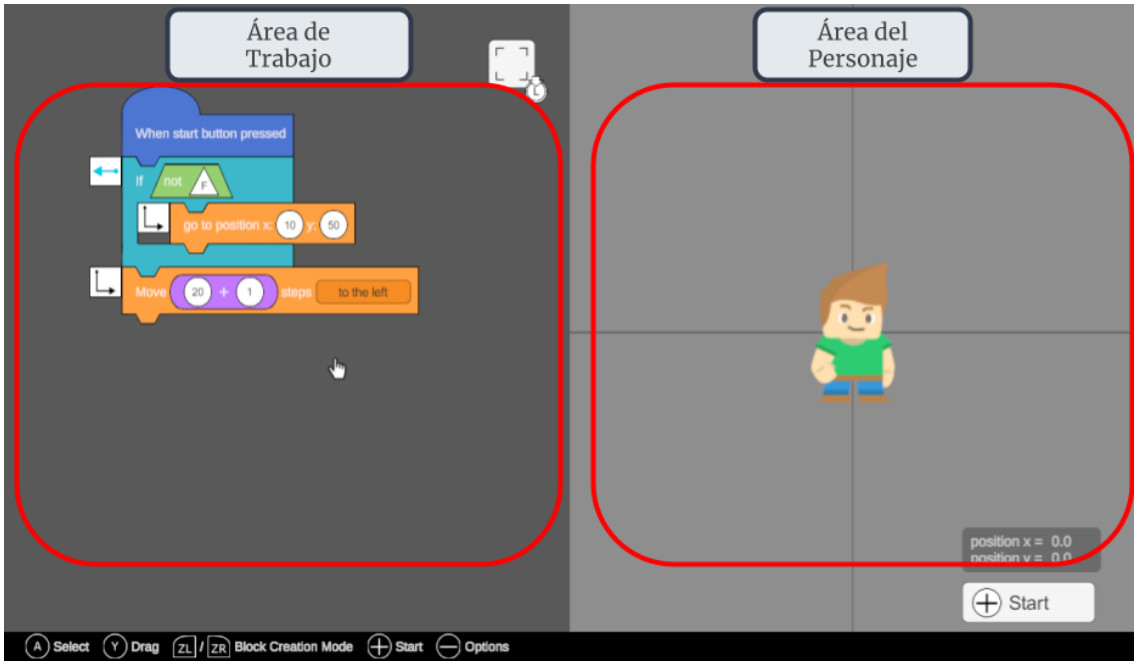


Figura 3.1: Pantalla principal de Gesture Coding.

Para interactuar con estas secciones, el usuario usa un *Cursos Virtual*. Este es controlado por la palanca análoga del Joy-Con L y, junto a los botones disponibles, permite que ejecutar acciones en el código. Este permite al usuario mover y encajar bloques haciendo uso de *Drag and Drop*, borrar bloques, seleccionar distintos valores y moverse por el Área de Trabajo. Además, en la sección de abajo de la interfaz se encuentra una barra con todos los comandos posibles a realizar por el usuario en este momento. Esta barra se actualiza al cambiar de interfaz, dependiendo de las acciones disponibles para este.

El personaje del panel derecho es quién recibe las instrucciones del código programado por los bloques. Este es capaz de moverse sólo en el plano derecho y sólo se puede interactuar con él ya sea ejecutando una serie de instrucciones con bloques, o arrastrándolo con el cursor. Este personaje se mueve bajo un sistema de coordenadas cartesianas en dos dimensiones, siento el centro del panel derecho el origen. Finalmente, en la parte inferior del panel derecho se muestran las coordenadas actuales del personaje para que el usuario tome de referencia al momento de programar.

[1] La imagen del personaje fue obtenida de `www.kenney.nl`. Recursos distribuidos bajo la licencia Creative Commons.

3.1.1. Controles y Gestos

El mecanismo de interacción de Gesture Coding son los Joy-Con de Nintendo Switch, siendo el Joy-Con L el que va en la mano izquierda y el Joy-Con R el de la mano derecha. Para poder conectar los controles al computador y al lenguaje, se usa la librería JoyConLib, que permite no sólo detectar los botones sino que además el acelerómetro y giroscopio de ambos controles a la vez. En el *Área de Trabajo*, el usuario interactúa usando los botones y palancas del control para mover, encajar y borrar bloques:

- **Palanca del Joy-Con L**: Mueve el cursor virtual.
- **Palanca del Joy-Con R**: Desplaza la zona visible del Área de Trabajo por la pantalla, haciendo *panning*.
- **Botón A**: Seleccionar objeto de un bloque. En caso de ser un input de tipo numérico, permite abrir la teclera virtual.
- **Botón Y**: Permite tomar o soltar un bloque para arrastrarlo por la pantalla. En caso de soltar su puerto hembra cerca de uno macho, estos se encajarán automáticamente.
- **Botón +**: Ejecuta el código desde el Bloque Inicial.
- **Botón -**: Abre el Menú de Opciones, desde donde se puede reanudar el progreso o salir del programa.
- **Botón ZR o ZL**: Abre el menú de creación de bloques.

Como se menciona arriba, para crear bloques se presionan los botones ZL (Joy-Con L) o ZR (Joy-Con R) en la parte posterior del control. Esto abre una nueva vista de la interfaz donde podemos seleccionar la categoría de bloque a abrir. Para elegir una categoría, el usuario debe realizar un gesto determinado ya sea con la mano izquierda (Joy-Con L) o derecha (Joy-Con R). Para realizar el gesto, el usuario controla un cursor virtual que responde a los movimientos realizados por el dispositivo, siendo un cursor virtual por control. Para registrar el gesto, el usuario debe mantener pulsado el botón ZR o ZL, dependiendo de la mano que esté usando, y dibujar en la pantalla el gesto deseado. Si este es detectado como un gesto válido, se pasará a la interfaz de selección de bloque, donde podrá elegir el bloque específico de la categoría elegida. Todos los gestos pueden realizarse con cualquiera de las dos manos, a excepción de los gestos relacionados a los Condicionales #1 y #2. En caso de que el sistema no logre detectar el gesto ingresado por el usuario, el control vibrará y le permitirá hacer un nuevo intento.

La detección de gestos del sistema se hace gracias al giroscopio incorporado en cada control. Esta información se obtiene mediante la librería JoyConLib [2] y permite detectar la velocidad angular del dispositivo en radianes en los tres ejes de rotación, mostrados en la Figura 3.3. La posición en el eje X del cursor virtual depende de la rotación del dispositivo en el eje Z, mientras que su posición en el eje Y se corresponde con la rotación del dispositivo en el eje Y.

De esta forma, si el usuario desea dibujar un círculo en la pantalla, debe realizar un movimiento giratorio con su muñeca mientras mantiene presionado el botón ZR. Al entrar al menú de creación de gestos, los cursores se centrarán en la pantalla automáticamente. Por lo

[2] `https://github.com/Looking-Glass/JoyconLib`

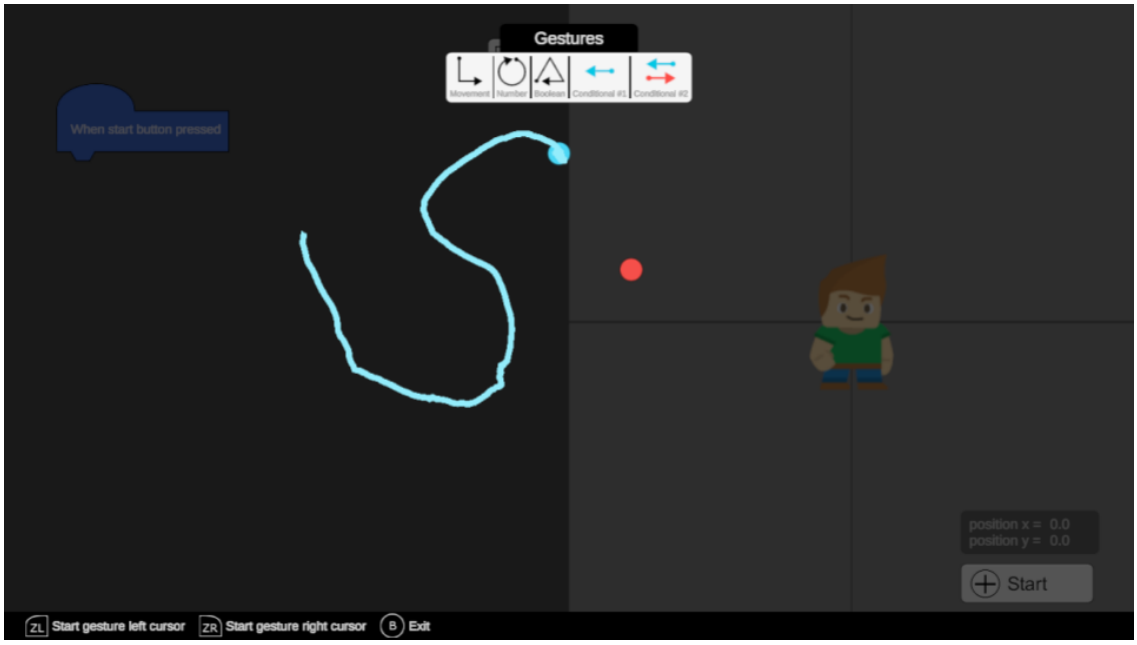


Figura 3.2: Menú de creación de bloques.

tanto, se recomienda apuntar a la pantalla con los controles antes de ingresar al menú para evitar sentirse descalibrado.

Detección de gestos

En Gesture Coding, se utiliza la detección de gestos en dos instancias: (1) Creación de bloques y (2) Elegir la dirección en la que se moverá el personaje. Para ambos casos la detección de movimiento es análoga, donde la interfaz del segundo caso es una variación de la primera. La detección de gestos se hace mediante la biblioteca de Unity *Gesture Recognizer*, orientada a detectar gestos dibujados con el mouse del computador. Un gesto está compuesto de un conjunto de líneas trazadas por el usuario en pantalla. Cada una de estas líneas son representadas por una lista de puntos en 2D que indican el camino que debe seguir el dibujo realizado por el mouse para ser reconocido como un gesto válido. Además, se puede restringir que el gesto sea reconocido sólo si este se realiza en una dirección determinada.

Para poder usar esta librería, en lugar de realizar el dibujo en pantalla con el mouse, se hace con el Joy-Con en la pantalla de Creación de Bloques. El usuario puede iniciar el proceso de dibujo manteniendo presionado el botón ZR o ZL, dependiendo de la mano que desee usar. Luego, sin soltar el botón, el cursor virtual del control comenzará a dejar un rastro por donde pasa. Finalmente, cuando el usuario suelte el botón, el proceso de dibujo habrá acabado y el sistema analizará el rastro dejado por el cursor. Si este coincide con el gesto de una categoría de bloques, se pasará al menú de selección de bloques de dicha categoría. Entonces, el proceso para poder crear un bloque desde la interfaz del *Área de Trabajo* es: (1) Presionar ZR o ZL para acceder al menú de creación de bloques, (2) Manteniendo presionado el botón ZL o ZR, dependiendo de la mano deseada, se realiza el gesto correspondiente a la categoría de bloques deseada; y (3) Se elige desde la ventana de selección el bloque específico que se desea usar, como se observa en la Figura 3.4.

Figura 3.3: Ejes de rotación de Joy-Con L.

Por simplicidad de uso, los gestos asociados a cada tipo de bloque consisten en una única línea contigua, sin restricciones de dirección; es decir, se pueden realizar en cualquier dirección, siempre y cuando la forma final coincida con la indicada por el sistema. Dado el funcionamiento del reconocimiento de gestos, es importante evitar gestos demasiado similares, ya que podrían llevar a falsos positivos o gestos tan precisos que resulten difíciles de detectar, lo que haría tediosa la creación de bloques. Por lo tanto, nos aseguramos de que todos los gestos sean lo suficientemente distintos entre sí para garantizar una experiencia agradable para el usuario. Sumado a esto, se procura que los bloques posean una etiqueta con el gesto que los representa, de tal forma que el usuario esté constantemente recordando qué gesto representa qué bloque. Cabe mencionar que basta con que el usuario haga un dibujo aproximado del gesto para que el sistema lo detecte, no es necesario que sea perfecto.

Para el caso de los gestos que nos permiten elegir la dirección de movimiento del personaje, el usuario debe trazar una línea en la dirección elegida. Entonces, si desea que el personaje se mueva hacia arriba, el usuario deberá dibujar una línea vertical en dirección ascendente, y si desea que se mueva hacia abajo, se traza una línea vertical en dirección descendente. Este proceso es análogo en caso de querer moverse hacia los lados, salvo que es una línea horizontal.

Finalmente, existe un tipo de gesto usado para una categoría de bloques que necesita una acción específica distinta de cada mano a la vez. A esto se le llama *Gesto Combo*, y lo que hace el sistema para detectarlos es verificar por separado el gesto dibujado por cada mano. Si la combinación de ambos movimientos es reconocido como un gesto válido por el sistema, entonces se procede a la siguiente interfaz, en caso contrario se le pide al usuario intentarlo de nuevo. De momento, este tipo de gestos sólo es usado en una categoría de bloque.

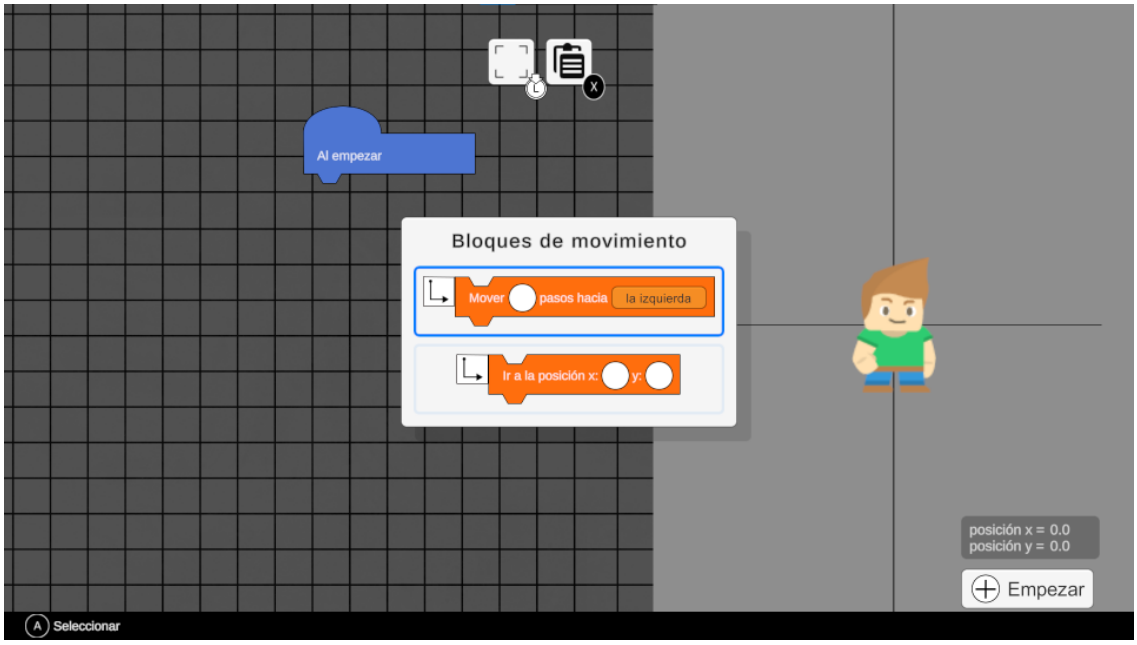


Figura 3.4: Menú de selección de bloques.

3.1.2. Bloques y Ejecución

Dado que Gesture Coding está fuertemente influenciado por otros lenguajes visuales como Scratch, sus bloques siguen convenciones de estos en la manera en que interactúan y se conectan entre sí mediante puertos macho o hembra. Dependiendo del comando a ejecutar, cada bloque puede poseer ya sea inputs numéricos, desplegables o espacios para encajar más bloques. Además, cada categoría de bloques está representado por un color en particular y un gesto libre a dibujar en pantalla.

Dentro del Área de Trabajo, cada bloque puede ser movido y posicionado en cualquier lugar sin necesidad de estar conectado a otro bloque. Sin embargo, estos no se ejecutarán a menos que estén conectados al *Bloque Inicial*.

Bloque Inicial

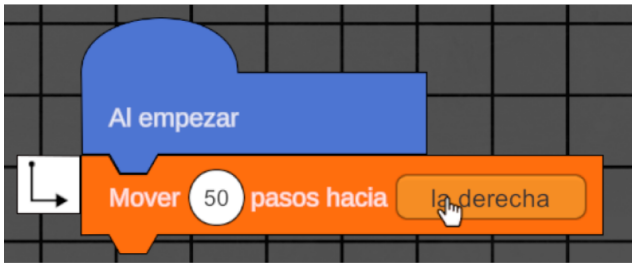


Figura 3.5: Bloque Inicial enlazado con uno de Movimiento.

Este bloque indica el comienzo del código. Al momento de ejecutar el programa, comienza ejecutando el primer bloque que esté enlazado a su puerto macho, siguiendo la cadena de

bloques hasta llegar al último. En un proyecto, sólo puede haber un *Bloque Inicial*, y es el único tipo de bloque que no se puede borrar. En la Figura 3.5 se le puede ver enlazado a uno de movimiento.

Bloques numéricos

Figura 3.6: Bloques de tipo numérico.

Estos bloques retornan un valor de tipo numérico, y consisten en bloques de operaciones matemáticas o variables con la posición actual del personaje en el Área del Personaje. Los bloques de operación contienen dos input donde se puede seleccionar un número con una teclera o poner otro bloque de tipo numérico, como se observa en la Figura 3.6. Este procedimiento se puede realizar de manera recursiva, obteniendo una operación tan compleja como se necesite. Además, como muestra la Figura 3.7 pueden ser insertados en otros bloques con un input de este tipo. Luego, al momento de ejecutar el código, este retornará el resultado de la operación que contenga el bloque, ya sea Suma, Resta, Multiplicación o División.

Por otro lado, los bloques de variable numérico son bloques especiales que contienen la posición actual del personaje en el eje X e Y. Estos bloques resultan especialmente útiles al ejecutar acciones dependiendo de la posición actual del personaje, combinando estos bloques con los de tipo condicional.

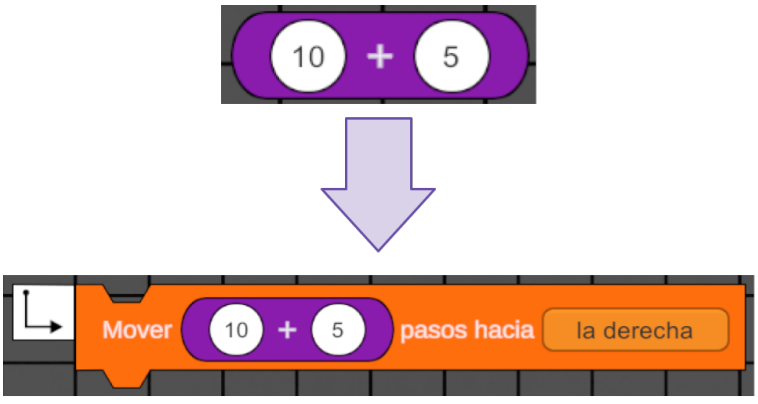


Figura 3.7: Bloque de tipo numérico insertado en uno de movimiento.

El gesto para crear un bloque de tipo numérico es un círculo. Dado esto, todo bloque que retorne un valor de tipo numérico tendrá forma circular, así como los inputs dónde se pueda ingresar un valor numérico con la teclera virtual. Para realizar el gesto, basta con que el usuario dibuje un círculo en pantalla en cualquier dirección y con cualquiera de las dos manos.

Bloques Booleanos

Figura 3.8: Bloques de tipo Booleanos.

Esta categoría de bloques abarca todos los comandos relacionados con booleanos o valores de verdad. Por un lado, se tienen bloques booleanos relacionados a operaciones entre números, como la operación *mayor (>)*, *menor (<)* o *igual (=)*. Estos bloques admiten dos inputs como los operandos, que pueden ser números ingresados por teclera u otro bloque de tipo numérico.

Por otro lado, tenemos los bloques booleanos relacionados a operaciones entre booleanos, como la *conjunción lógica (`and`)* y la *disyunción lógica (`or`)*. Los operandos en este caso son dos inputs de tipo booleanos. Finalmente, se tiene la *negación lógica (`not`)*, que tiene un sólo input de tipo booleano. Estos pueden observarse en la Figura 3.8

El gesto que representa esta categoría es un triángulo, por lo que todos los bloques con forma triangular o trapezoidal retornan un valor de tipo booleano. Además, todo input que admiten un valor booleano es de forma triangular. Para seleccionar esta categoría de bloque, el usuario debe dibujar un triángulo en el menú de creación de bloques con cualquiera de las dos manos y en cualquier dirección.

Bloques de Movimiento

Esta categoría de bloques se encarga de mover al personaje por el panel derecho. Dado que estos bloques sólo mueven al personaje, no retornan ningún tipo de valor y son cuadrados con un puerto macho y uno hembra. Como se observa en la Figura 3.9, existen dos bloques de este tipo.

El primer bloque es el que permite mover al personaje una cierta cantidad de pasos en una dirección. Este bloque admite un input de tipo numérico que determina cuántos pasos se moverá el personaje, y un input de tipo dirección que indica hacia dónde se moverá el

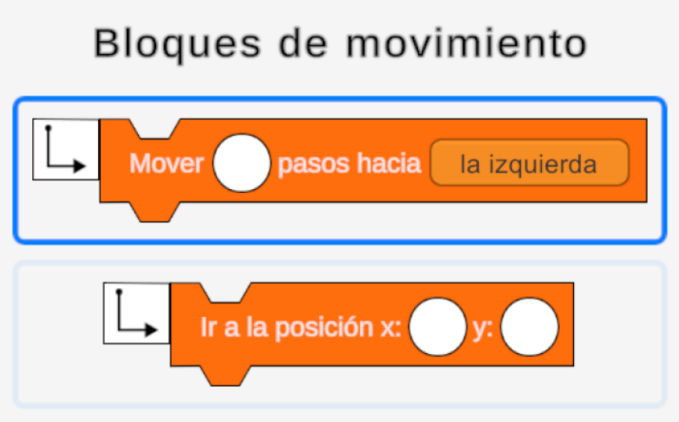


Figura 3.9: Bloques de tipo Movimiento.

personaje. Este último input es manejado exclusivamente con gestos libres y abarca cuatro direcciones: Arriba, Abajo, Izquierda y Derecha. Al ejecutar este bloque, el personaje se moverá dicha cantidad de posiciones en el eje mencionado.

Por otro lado, está el bloque que permite moverse a cierta posición específica. En este caso, se requieren dos inputs de tipo numérico, uno representando a la posición en el eje X y otro en el eje Y. Al ejecutar este bloque, el personaje aparecerá en la posición indicada.

El gesto seleccionado para esta categoría de bloques es una línea recta en forma de letra *L* mayúscula. Para realizar este gesto, el usuario debe, en primer lugar, dibujar una línea vertical descendente y luego una línea horizontal en un ángulo recto. Este gesto puede realizarse con cualquiera de las dos manos.

Bloques Condicionales

Estos bloques, mostrados en la Figura 3.10, permiten ejecutar código dependiendo del valor de una condición. Se presentan dos tipos de bloque: *If o Condicional #1* y *IfElse o Condicional #2*. En ambos casos, la condición es un bloque de tipo booleano, es decir, con forma triangulada. Además, ambos tipos de bloque poseen conjuntos de puertos macho-hembra en su interior para encajar código, además de los puertos externos que les permite conectarse al resto del código.

El bloque de tipo *If o Condicional #1* tiene un puerto hembra y uno macho para encajar código. De ser verdadera la condición, el código entre el primer conjunto macho-hembra se ejecuta, en caso contrario, el código se omite y se da por finalizada la ejecución del bloque.

Por otro lado, el bloque de tipo *IfElse o Condicional #2* tiene dos puertos hembra y dos machos, repartidos en dos conjuntos macho-hembra para encajar código. Si la condición es verdadera, el código del primer conjunto macho-hembra se ejecuta y se omite el código del segundo conjunto. En caso contrario, se omite el código del primer conjunto macho-hembra y se ejecuta el del segundo conjunto.

Este conjunto de bloques, si bien dentro de la misma categoría, poseen un gesto para cada uno. El gesto para crear un bloque Condicional # 1 es trazar una línea horizontal de derecha

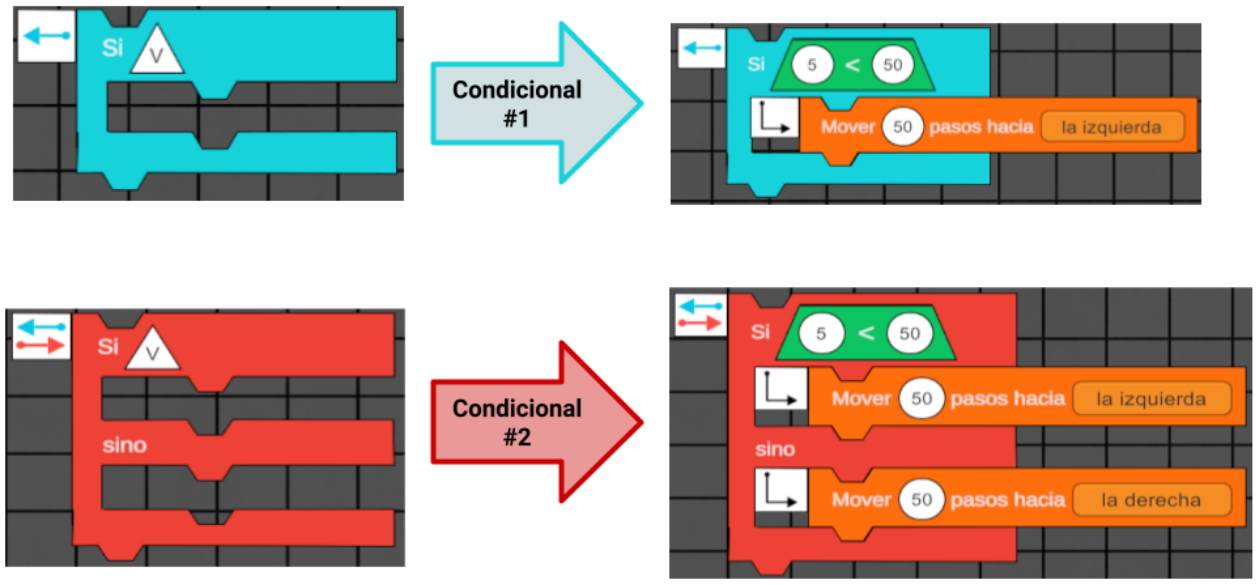


Figura 3.10: Bloques Condicionales simples y dobles.

a izquierda con la mano izquierda. Por otro lado, para crear un bloque Condicional #2, el gesto consiste en trazar dos líneas simultáneamente, una con cada mano, de manera reflejada. La mano izquierda repite el gesto del Condicional #1, mientras que la mano derecha realiza el mismo gesto en dirección contraria, es decir, de izquierda a derecha.

La elección de gestos para este par de bloques simboliza la naturaleza de una condición. En el Condicional #1, hay un solo código posible que puede o no ejecutarse según la condición. En cambio, en el Condicional #2, existen dos caminos posibles dependiendo del valor de verdad de la condición. Esto ayuda a desarrollar la intuición del usuario, donde un condicional doble presenta dos códigos posibles para ejecutar, mientras que el condicional simple presenta solo uno.

Ejecución de código

Luego de terminar de conectar los bloques en el orden deseado, el usuario puede ejecutar su código pulsando el botón + en el Joy Con R o haciendo click con el cursor virtual en el botón *Empezar* del panel derecho. Esto comenzará a ejecutar los bloques uno a uno desde el *Bloque Inicial*. Todo bloque que no forme parte de esta cadena no será ejecutado.

Consideraciones para la motivación

Como se mencionó en el capítulo anterior, aprender a programar puede ser un proceso intimidante para un estudiante novato, especialmente a una edad temprana. El uso de gestos libres al programar tiene un impacto significativo en cómo los estudiantes perciben el proceso de aprendizaje. Cambiar el mecanismo de interacción, especialmente uno utilizado principalmente en videojuegos, tiene el potencial de hacer que los estudiantes aborden la herramienta

de manera más abierta y vean la actividad como un desafío en lugar de una tarea tediosa. En particular, el uso de gestos específicos para cada categoría de bloque y colores distintivos no solo ayuda a los estudiantes a distinguir un bloque de otro, sino que también les permite relacionar las funcionalidades de los bloques entre sí y simplificar el proceso de creación de código. Por lo tanto, Gesture Coding afecta el valor percibido por los estudiantes debido a su mecanismo de interacción no convencional, al tiempo que reduce las expectativas de los estudiantes al simplificar y relacionar algunos contenidos para que sean más manejables.

A continuación se presenta un estudio piloto realizado con la primera versión de esta herramienta para evaluar cómo es recibida por estudiantes de entre 10 y 12 años.

3.2. Estudio de Usuarios

Con el objetivo de explorar el impacto de Gesture Coding como primera aproximación al aprendizaje de lenguajes de programación en niños, se realizó un estudio controlado. Este fue realizado respetando todos los protocolos sanitarios, dadas las condiciones en torno a COVID-19 al momento de llevar a cabo la experiencia.

Este estudio se enfocó en evaluar cómo Gesture Coding es percibida por un grupo de estudiantes, además de analizar el impacto del uso de controles de movimiento en las cargas físicas y cognitivas implicadas en su uso. Para esto, se evaluó el desempeño de tres lenguajes de programación visuales basados en bloques: (1) Scratch, (2) Gesture Coding con Joy-Con, y (3) Gesture Coding con Mouse y Teclado, lenguaje en el que se basa Gesture Coding.

Finalmente, usando esta información como base, se extendieron las funcionalidades de Gesture Coding para cubrir el conjunto minimal de habilidades de pensamiento computacional, para finalmente evaluar nuevamente su impacto en la barrera de entrada percibida por los estudiantes a programar en lenguajes visuales.

3.2.1. Participantes

Invitamos estudiantes de 5° año básico a participar en un taller informal de programación. Un total de 99 estudiantes participaron en la experiencia, de los cuales 47 eran niños y 52 eran niñas, todos entre 10 y 11 años. Cada participantes fue asignado aleatoriamente a uno de tres grupos asegurando balance en términos de edad, género y experiencia previa usando controles de movimiento de cualquier tipo. Ninguno de los participantes reportó tener conocimiento previo de programación.

Para poder participar en la actividad, los estudiantes debían brindar un asentimiento explícito, informado y libre al inscribirse. Del mismo modo, sus padres y/o tutores legales debieron proporcionar un consentimiento explícito, informado y libre dado que los estudiantes eran menores de edad al momento de realizar la experiencia. Todos los participantes tuvieron el derecho de abandonar la experiencia en cualquier momento sin consecuencias negativas. Los datos recopilados fueron anonimizados y utilizados únicamente en este análisis. El protocolo de investigación fue aprobado en términos éticos por el Comité de Ética de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile.

3.2.2. Procedimiento

El estudio consistió en un taller práctico informal de tres horas en el que a cada grupo se le asignó una de las tres herramientas: (1) Scratch, (2) Gesture Coding con Joy-Con y (3) Gesture Coding con Mouse y Teclado. Esto permitió evaluar comparativamente la aceptación y utilidad de cada una, explorando cómo los participantes percibieron Gesture Coding con Joy-Con como su primera aproximación a un lenguaje de programación.

Con el fin de mantener un escenario controlado durante la experiencia, la estructura del taller fue idéntica para los tres grupos, variando solo la herramienta utilizada por los participantes al momento de realizar las tareas de programación asignadas.

Al final de cada sesión, se llevó a cabo un focus group para recopilar información cualitativa sobre la herramienta utilizada, y se pidió a los participantes que completaran el cuestionario NASA Task Load Index (NASA-TLX). Este cuestionario consiste en una escala Likert de 10 puntos que mide la carga mental y física (*Task Load*, en inglés) experimentada por un individuo al interactuar con un dispositivo. Para este estudio, se realizó una adaptación del formulario original con el objetivo de que fuera comprensible para niños y niñas de entre 10 y 12 años. Previamente, se realizó un estudio piloto para asegurar la claridad y comprensión del instrumento por parte del público objetivo. El cuestionario evalúa los siguientes aspectos en una escala Likert de 10 puntos:

- *Carga Mental (MD)*: ¿Qué tanto tuviste que pensar mientras usabas la herramienta para resolver las tareas?
- *Carga Física (PD)*: ¿Qué tanto esfuerzo físico te tomó usar la herramienta para completar las tareas?
- *Carga Temporal (TD)*: ¿Qué tan apurado te sentiste mientras completabas las tareas?
- *Desempeño (P)*: ¿Qué tan seguro estás de las soluciones obtenidas en las tareas?
- *Esfuerzo (E)*: ¿Qué tan fáciles fueron las tareas que tuviste que completar?
- *Frustración (F)*: ¿Cómo te sentiste al completar las tareas?

3.2.3. Resultados y Discusión

Dado el tamaño de la muestra, no podemos asegurar que los resultados obtenidos sean del todo generalizables. Además, estos deben ser considerados e interpretados dado el contexto del estudio y de los grupos evaluados.

Ejecución de tareas

Para evaluar el impacto de la herramienta utilizada en el desempeño de los estudiantes al momento de desarrollar las tareas, se analizaron los tiempos promedios que le tomó a cada grupo ejecutar las tareas propuestas.

Con el fin de analizar el efecto del género del participante y la herramienta utilizada en la actividades del taller, se realizó un análisis de varianza (ANOVA) de dos vías tipo III no balanceado. Además, se llevaron a cabo pruebas post hoc con valores de significancia ajustados siguiendo el procedimiento de Benjamini-Hochberg. El efecto principal observado

Figura 3.11: Tiempo en segundos que le tomó completar las tareas a cada grupo.

por la herramienta utilizada, $F(2, 93) = 102{,}4153; p < 0{,}001$, fue tal que el tiempo dedicado por los participantes que utilizaron Gesture Coding con Joy-Cons ($M = 1256{,}029; SD = 126{,}9329$) fue significativamente mayor que el tiempo dedicado por los participantes que utilizaron Gesture Coding con un mouse y teclado ($M = 815{,}4848, SD = 91{,}05531$), así como Scratch ($M = 830{,}8438; SD = 133{,}3589$). El efecto principal del género también fue significativo, $F(1, 93) = 5{,}4672; p = 0{,}02$, aunque el tiempo dedicado por los niños ($M = 922{,}8085; SD = 213{,}7029$) no fue sustancialmente mayor que el tiempo dedicado por las niñas ($M = 1015{,}981; SD = 251{,}354$). Finalmente, no observamos un efecto de interacción, $F(2, 93) = 0{,}9066; p > 0{,}05$.

Los resultados obtenidos revelan que, en promedio, los participantes necesitaron más tiempo para completar las tareas al emplear controles de movimiento. Esta observación puede ser explicada por la ley de Fitts [23], la cual establece que el tiempo necesario para seleccionar un objeto es directamente proporcional a su distancia y tamaño. Por lo tanto, era de esperar que se requiriera un mayor esfuerzo físico y más tiempo para cumplir con la tarea, ya que: (1) El uso de controles de movimiento implica una fase de adaptación para que los usuarios se desplacen por la pantalla; y (2) Se necesita un mayor número de interacciones para crear un bloque. Respecto al género, no se observaron diferencias significativas atribuibles al uso de la herramienta específica para llevar a cabo la tarea.

Cargas físicas y cognitivas (NASA-TLX)

Con el fin de evaluar cómo los participantes percibieron la usabilidad de cada herramienta, se analizaron las respuestas proporcionadas por el cuestionario NASA-TLX aplicado al final del taller. En la Figura 3.12 se muestra la distribución de las puntuaciones asignadas a cada dimensión de la escala NASA-TLX.

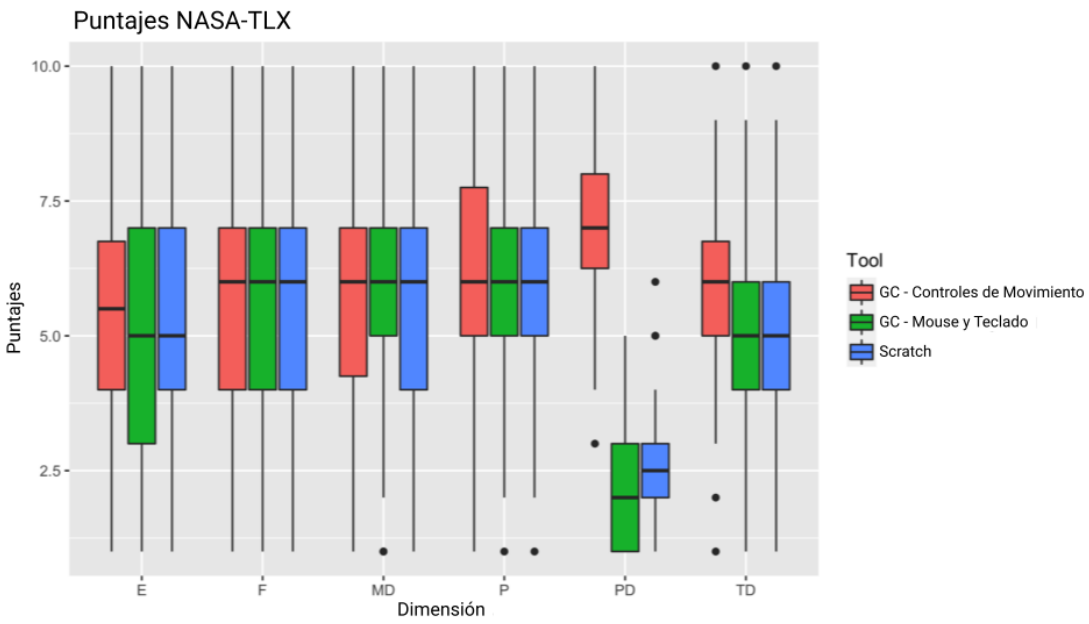


Figura 3.12: Puntuaciones en cada dimensión de la escala NASA-TLX.

Los datos obtenidos no cumplen con las hipótesis necesarias para ejecutar un ANOVA unidireccional paramétrico, según las pruebas de normalidad de Shapiro-Wilk. Luego, se realiza una prueba de Kruskal-Wallis para cada dimensión de la escala con el fin de estudiar si existen diferencias estadísticamente significativas en las medianas de las puntuaciones entregadas por los participantes. No se observan diferencias estadísticamente significativas al analizar los datos por género.

De todas las dimensiones del instrumento, sólo observamos un efecto principal para la dimensión de Carga Física (PD), $\chi^2(2) = 65{,}649, p < {,}001$, donde una prueba *post hoc* de Mann-Whitney indicó que el esfuerzo físico requerido para operar Gesture Coding con Joy-Cons es significativamente mayor que el de Scratch y Gesture Coding con mecanismos de interacción convencionales. Este resultado se condice con el tiempo promedio que los participantes pasaron interactuando con controles de movimiento. En consecuencia, al no observarse diferencias significativas en las otras cargas, se puede concluir que el principal elemento distintivo de Gesture Coding con respecto a otros lenguajes de programación visuales basados en bloques es su mecanismo de interacción basado en controles de movimiento.

Resultados de Focus Group

Siguiendo un análisis temático [12], la experiencia se enfocó en explorar las percepciones de utilidad y aceptación de las tres herramientas que se utilizaron en el taller. Los temas principales que se abordaron fueron el atractivo, la diversión y el potencial de adopción de Gesture Coding, todos apuntados a evaluar el valor percibido por el estudiante.

En cuanto a las tareas que se presentaron a los participantes, no hubo diferencias sustanciales en la dificultad percibida para completarlas. En particular, los comentarios espontáneos de los participantes del grupo A (que utilizaron Scratch) elogiaron la interactividad y la paleta de colores de la interfaz de usuario. Los participantes de los grupos B y C (es decir,

aquellos que utilizaron Gesture Coding con cualquiera de los dos mecanismos de interacción) elogiaron la simplicidad y el atractivo de la interfaz de usuario. En particular, el uso de Joy-Cons en el grupo B fue muy apreciado. Los gestos propuestos se consideraron naturales e intuitivos, y la experiencia en general se percibió como lúdica y agradable, según lo reportado por los participantes: *"Fue divertido, cómodo y fácil de usar"* (P4, H, 10 años); *"A pesar de tener pocos bloques, me gustó mucho usarlo"* (P5, M, 11 años). Esto, posiblemente debido al contexto particular en el que los controles se utilizan en la práctica (es decir, los videojuegos). En este sentido, los participantes se mostraron visiblemente más entusiasmados al interactuar con Gesture Coding utilizando Joy-Cons y expresaron esta opinión en forma de comentarios espontáneos, como *"Me recuerda a Crash Bandicoot o Super Mario"* (P3, H, 11 años), o *"Está muy bueno para ser de Nintendo Switch"* (P8, M, 10 años).

Posible efecto del uso de controles de movimiento en el fomento del interés por la programación

Finalmente, y con el fin de evaluar el impacto del taller en el interés de los estudiantes en la programación sostenido en el tiempo, se contactó a todos los participantes del estudio tres meses después de que se hubiera llevado a cabo el taller. En estas reuniones, se le consultó a los participantes si habían tenido algún acercamiento adicional a la programación y, en caso afirmativo, qué herramientas utilizaron.

Tabla 3.1: Retención luego de 3 meses.

Grupo	N	Aún interesado	Proporción
A (Scratch)	32	4	12.5 %
B (Gesture Coding – Controles de Movimiento)	34	11	32.4 %
C (Gesture Coding – Controles Tradicionales)	33	1	3.0 %

Tal como se observa en la Tabla 3.1, los participantes del taller que tuvieron su primera experiencia en programación utilizando Gesture Coding con controles de movimiento mostraron un mayor interés en continuar con la actividad posteriormente, en comparación con la proporción de participantes inscritos en los otros dos grupos. Al realizar una prueba de dos muestras para la igualdad de proporciones, se observa que la diferencia es estadísticamente significativa ($p < ,01$) entre B y C, pero no entre A y B ($p = ,103$). Por lo tanto, los controles de movimiento efectivamente despertaron el interés por la programación sostenido en el tiempo entre los participantes, incluso después de concluir el taller.

El análisis del estudio realizado muestra que, a pesar de requerir un mayor esfuerzo físico para realizar tareas introductorias de programación, como conectar bloques y crear aplicaciones simples, los Joy-Con tienen el potencial de despertar un mayor interés y disfrute entre los participantes del estudio. Este compromiso ofrece oportunidades prometedoras para involucrar a jóvenes aprendices sin experiencia previa en programación en la creación de aplicaciones sencillas a través del uso de gestos naturales con las manos. Además, los participantes que utilizaron Gesture Coding con controles de movimiento mostraron más interés en la programación después del taller, por lo que se plantea que el mecanismo de interacción explorado tiene el potencial de reducir la barrera de entrada para aprender a programar. En términos de la teoría de expectativa-valor, el valor percibido por los estudiantes al programar

utilizando controles de movimiento fue tal que aumentó su motivación para aprender a programar, desarrollando un interés sostenido a lo largo del tiempo, algo que no se observó de la misma manera con otras herramientas que utilizan mecanismos de control tradicionales.

Usando este estudio de base, se propuso extender el rango de bloques del lenguaje de tal manera que permita cubrir el conjunto minimal de habilidades de pensamiento computacional integrando bloques de Ciclos y Variables, dado que actualmente sólo abarca Secuencialidad y Condicionales. Esto implica incorporar nuevos gestos y bloques, siguiendo el mismo paradigma de interacción actual. Además, se plantean modificaciones generales a la usabilidad del sistema basado en los comentarios de los participantes.

Capítulo 4

Espira 2: Diseño

En particular, en este trabajo de tesis se estudió el impacto del mecanismo de entrada en la percepción y motivación del estudiante respecto a la programación, específicamente, de un mecanismo tradicional como lo es mouse y teclado en comparación a controles de movimiento. Para esto se hizo uso de *Gesture Coding* [43], un lenguaje de programación controlado con gestos. De manera similar a Scratch, Gesture Coding está basado en bloques, es decir, cada comando consiste en un bloque de color que se conecta a otro y permite escribir un programa. La principal diferencia es que este último es controlado por gestos, es decir, usa controles de movimiento para seleccionar y crear dichos bloques. Específicamente, se buscó cubrir el conjunto minimal de habilidades de Pensamiento Computacional en el modelo de trayectorias de aprendizaje propuesto por Rich et al. [52]: Secuencialidad, Condicionalidad, Ciclos y Variables.

Inicialmente, Gesture Coding se encontraba en etapa de prototipo funcional y sólo soportaba bloques de tipo secuencial y condicional [43]. Luego, no era posible abarcar el conjunto minimal de habilidades descrito anteriormente [52]. Es por esto que resultó necesario extender el soporte del lenguaje a bloques de variables y ciclos.

El diseño de Gesture Coding se ve fuertemente influenciado por Scratch y otros lenguajes visuales, ya que permiten simplificar la sintaxis y semántica del proceso de programación, especialmente en la interacción de los bloques y su relación entre sí. Sin embargo, una gran limitante de la aplicación en un inicio es que la cantidad de bloques y funcionalidades de esta es poca en comparación a otros lenguajes visuales. Dado esto, se propone extender el alcance del lenguaje agregando nuevas categorías de bloques que permitan crear código más complejo y completo, siguiendo las trayectorias de aprendizaje planteadas por Rich et al. [52].

En un inicio, Gesture Coding disponía sólo de bloques de tipo Booleanos, de Movimiento y de Operaciones Numéricas. Sin embargo, estas categorías abordan exclusivamente las trayectorias de aprendizaje relacionadas con la Secuencialidad y la Condicionalidad. Por lo tanto, se propuso la incorporación de nuevas categorías de bloques destinadas a abarcar la trayectoria de Repetición. Con este fin, se introducen las categorías de Ciclos y Variables. Si bien, la implementación de Ciclos implicó agregar nuevos bloques, la integración de Variables al Lenguaje requirió un trabajo de diseño más extenso, aterrizando el concepto al público

objetivo, siguiendo las recomendaciones actuales de la literatura [65].

4.1. Arquitectura General

Los proyectos en Unity se componen principalmente de *Objetos de Juego* (`GameObject`), *Componentes* (`Components`) y *Escenas* (`Scenes`). Todo elemento dentro del proyecto es un `GameObject`, cuyas propiedades y lógica es determinada por sus componentes `Components`. Estos objetos son distribuidos en las escenas `Scenes`, generando el entorno de la aplicación.

En Gesture Coding en particular, existen dos escenas: *Menú Principal* y *Nuevo Proyecto*. La primera escena corresponde a la pantalla del título, y sólo contiene los botones para seleccionar un nuevo proyecto o salir de la aplicación. La segunda escena es la que maneja tanto las vistas de creación de bloques y gestos. En particular, en la escena de `Nuevo Proyecto` se encuentran la detección de gestos, cursores y bloques.

El comportamiento de los bloques depende de distintas clases o componentes, dependiendo del tipo de bloque que sea. Luego, para agregar un nuevo bloque basta con agregar un nuevo objeto `GameObject` con su respectivo componente `Component` que implemente la interfaz `Block`. Además, cada una posee el método `Calculate()` que evalúa y retorna la expresión específica del bloque. Finalmente, para ejecutar un código, se ejecuta la función `Calculate()` de todos los bloques enlazados con el Bloque Inicial.

4.2. Diseño de Bloques

El proceso de diseño e integración de nuevas categorías de bloques debe ser consistente con lo que ya existe, es decir, no debe romper con lo que ya está diseñado e implementado. Esto es crucial para evitar confusiones y garantizar una experiencia de usuario coherente. Para esto, en el proceso de diseño de nuevas categorías se hizo especial énfasis en mantener la consistencia respecto a las otras categorías de bloques, así como presentar gestos congruentes con lo que representa el concepto. Para esto, se tomaron de base los principios de diseño planteados por Jacob Nielsen [47]. En particular, los principales focos a tener en cuenta al momento de diseñar fueron:

- **Consistencia**: Las nuevas categorías y bloques integrados deben seguir el estilo tanto gráfico como funcional de los bloques ya implementados. De esta forma, estos serán percibidos como una parte de un todo y no como algo "extra".
- **Metáfora entre el sistema y el mundo real**: La forma de uso de los bloques, especialmente los gestos, debe reflejar el mundo real de tal forma que tanto la información como las acciones presentadas sean percibidas como familiares y/o intuitivas.
- **Reconocer en lugar de recordar**: El sistema debe minimizar la carga al usuario al no hacerlo recordar cómo usar los bloques, sino que estos deben ser lo suficientemente intuitivos, teniendo opciones y acciones a la vista en todo momento.
- **Diseño estético y minimalista**: No sobrecargar al usuario con información o acciones que no sean relevantes en un momento dado. Estas deben poder ocultarse o minimizarse a voluntad.

Otros principios de diseño como la prevención de errores, documentación y visibilidad del estado del sistema ya se encuentran integrados en el sistema original, por lo que se procura seguir los lineamientos ya planteados.

4.2.1. Repetición y Ciclos

La implementación de Ciclos y Repeticiones va de la mano con la noción que se espera desarrollar en los estudiantes. Para esto, todo el diseño de los bloques de esta categoría se ajustan a la metáfora de que una repetición es un proceso que se realiza muchas veces. Esto sigue la percepción del concepto de *Loop* o Repetición por parte de los estudiantes según la literatura [80]. Dado esto, tanto el lenguaje utilizado en los bloques como el gesto elegido debe reflejar esta situación.

Diseño de Bloques

Para la integración de Ciclos y Repeticiones en el lenguaje, en primer lugar se decidieron las funcionalidades a abordar, basadas en las trayectorias de aprendizaje [52] y su implementación en otros lenguajes visuales [80]. En particular, para la categoría de Ciclos se proponen los siguientes bloques:

- **Repetición una cantidad concreta de veces**: El usuario puede decidir un número exacto de veces que el código dentro del bloque se repetirá. Esta cantidad puede ser un bloque de tipo numérico o variable.
- **Repetición condicional**: El bloque repetirá el código mientras se cumpla una condición. Esta será representada por un bloque de tipo condicional, que puede integrar un bloque de tipo numérico o variable. Sin embargo, dado que los bloques de variables son inmutables, la única variable que se puede utilizar en este sentido es el de la posición del personaje ya sea en el eje X o Y.

Al ejecutar un bloque de repetición, el usuario debiera ser capaz de observar el movimiento del personaje en pantalla. Luego, si el usuario quisiera repetir un movimiento diez veces en pantalla, el programa debiese mostrar todos los estados intermedios en cada iteración, y no que se muestren sólo los estados iniciales y finales. De esta forma, el usuario es capaz de observar y procesar lo que ocurre dentro del ciclo en lugar de sólo ver el resultado final de la iteración.

Además, cada uno de estos bloques debe poder ser insertado en otros bloques, como lo son los de movimiento o condicionales. Del mismo modo, estos deben poder ser insertados dentro de otro bloque de tipo repetición, es decir, el lenguaje debe permitir la implementación de ciclos anidados.

El diseño de estos bloques busca mantener una consistencia respecto a los bloques ya integrados en la plataforma, siguiendo de cerca el diseño de los condicionales simples. Esto dado que ambos bloques se encargan de modificar la forma en que se ejecutan ciertas líneas de código, actuando como un contenedor para otros bloques.

Estos requisitos buscan ser consistentes con lo visto en otros lenguajes visuales según la literatura [80], y fueron validados con expertos de dominio.

Gesto

Dado que esta corresponde a una categoría de bloques, se elige un gesto que represente a ambos bloques diseñados como una sola categoría. La idea detrás de este gesto es que represente la naturaleza iterativa de los bloques de repetición, por lo que se prioriza un gesto que presente una iteración o un movimiento repetido.

Una primera aproximación a este diseño fue agitar el control como gesto. Luego, si el usuario agitaba el control en el menú de creación de bloques, se accedería a la categoría de repetición. Con esto, el usuario podría incluso decidir cuántas veces repetir el código agitando el control. La idea detrás de este gesto es que, al igual que en una repetición, el gesto de agitar el control es representado por una acción que se repite constantemente, que es mover el control de arriba hacia abajo. Sin embargo, una limitación de este gesto es que tanto la librería de reconocimiento de gestos como la librería que controla los Joy-Con permiten detectar si el control está siendo agitado con precisión, menos aún detectar cuántas veces ha sido agitado. Dada esta limitación, se descarta este gesto.

Otra aproximación al gesto es dibujar en la pantalla una figura o letra que requiera un movimiento iterativo del mismo modo que lo es agitar el teléfono. Con esto en mente, se evalúan gestos como dibujar una espiral en pantalla o una letra "Z" en pantalla. La principal limitación de la primera opción es que el dibujo de una espiral es similar a un círculo, gesto que representa los bloques de tipo numérico. Dado esto, el usuario debería ser demasiado preciso para que el sistema de reconocimiento de gestos evite dar un falso positivo al momento de detectar el gesto dibujado por el usuario. Usar una espiral y un círculo como gestos distintos implica propiciar errores de parte del usuario en lugar de evitarlos.

Por otro lado, el dibujo de una letra "Z" en pantalla implica un movimiento horizontal que se repite tres veces, uno por cada trazo. De esta forma, usar esta letra como gesto cumple con el requisito de representar la naturaleza, dado que el usuario debe mover su muñeca de lado a lado, gesto similar a agitar el control pero de manera más precisa. La ventaja de este gesto sobre el anterior es que la letra "Z" no induce error al ser similar a otro gesto. Sin embargo, posee la limitación de que el ángulo entre los trazos es muy preciso para el sistema de detección de gestos. Por lo que, si la letra "Z" dibujada por el usuario no es lo suficientemente bien hecha, esta no sería reconocida. Esto implica que gestos como dibujar una"Z"más aplastada o con los trazos superior e inferior de distinto tamaño no serían válidos. Este problema trasgrede el principio de diseño de Flexibilidad y Eficiencia de Uso, dado que, en lugar de darle facilidades al usuario para dibujar el gesto, se le complica.

Sin embargo, el objetivo del gesto no es dibujar una letra "Z" precisa, sino que hacer el movimiento iterativo con la mano, por lo que cualquier gesto que sea similar es válido, tomando como base el dibujo de la letra "Z". Es por esto que se agrega a la detección de gestos distintas variaciones de la letra "Z" para que el sistema de reconocimiento de gestos se perciba como más flexible. De esta forma, se hace énfasis en que el usuario haga el movimiento iterativo con el cursor en lugar de dibujar la letra "Z" de manera precisa. A pesar de esto, y por consistencia y claridad respecto a las otras categorías, se deja como gesto "oficial" de esta categoría a la letra Z. Con esto, se da la sensación en el usuario que el sistema de reconocimiento de gestos es más flexible al momento de dibujar la letra "Z" de lo que realmente es.

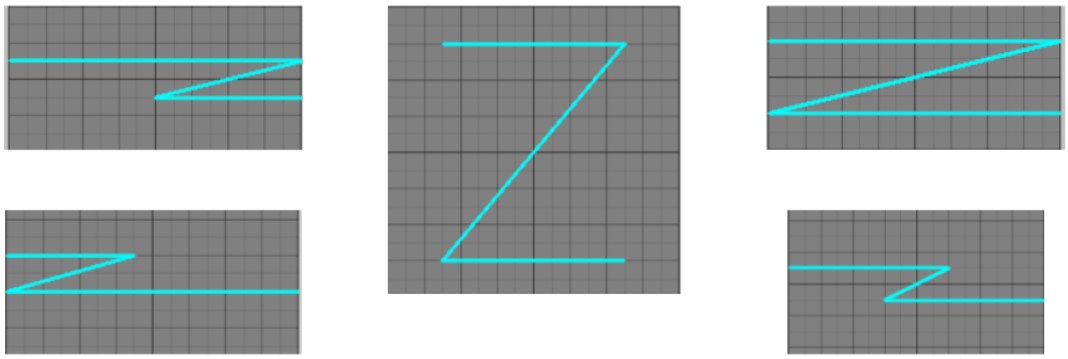


Figura 4.1: Posibles variaciones del gesto “Z” para Ciclos.

Implementación

La integración de nuevos bloques en el sistema implica modificar el menú de creación de gestos, agregar un nuevo menú de selección para los nuevos bloques de la categoría en particular y agregar los mismos bloques. Para implementar el gesto de la categoría, se generan e integran los nuevos patrones de gestos que van a representar a la repetición, siendo estos seis variaciones distintas de la letra “Z”, indicados en la Figura 4.1. Luego, a nivel de código se agrega un nuevo caso en el sistema de reconocimiento de gestos de tal forma que, si se detecta alguna de las variaciones de la letra “Z” antes mencionadas, se abrirá el menú de selección de bloques de repeticiones.

Los bloques de repetición toman de base el bloque condicional. Esto se debe en gran parte a que el diseño de esos bloques son análogos a los de ciclos, dado que ambas categorías permiten encapsular código e influir en cómo se ejecutarán dichos bloques. Sin embargo, al ejecutar el bloque de repetición, se observó un problema en la visualización de cada iteración del ciclo. En lugar de mostrar correctamente cada paso del ciclo, solo se mostraban el estado inicial y final del personaje.

Por ejemplo, supongamos que el usuario desea que el personaje se mueva desde la posición X = 0 hasta la posición X = 50 en 5 movimientos de tamaño 10. El comportamiento esperado sería que la pantalla mostrara cada iteración del personaje, comenzando en la posición 0 y mostrando cómo avanza a la posición 10, luego a la 20 y así sucesivamente hasta llegar a 50. Sin embargo, el bloque de repetición solo mostraba al personaje en la posición inicial X = 0, ejecutaba el ciclo completo y luego presentaba al mostraba en la posición final X = 50, como se observa en la Figura 4.2. Este problema surge porque el sistema ejecuta todos los bloques conectados al Bloque Inicial sin esperar a que el cambio se vea reflejado en la pantalla, por lo que sólo se termina por observar la posición final.

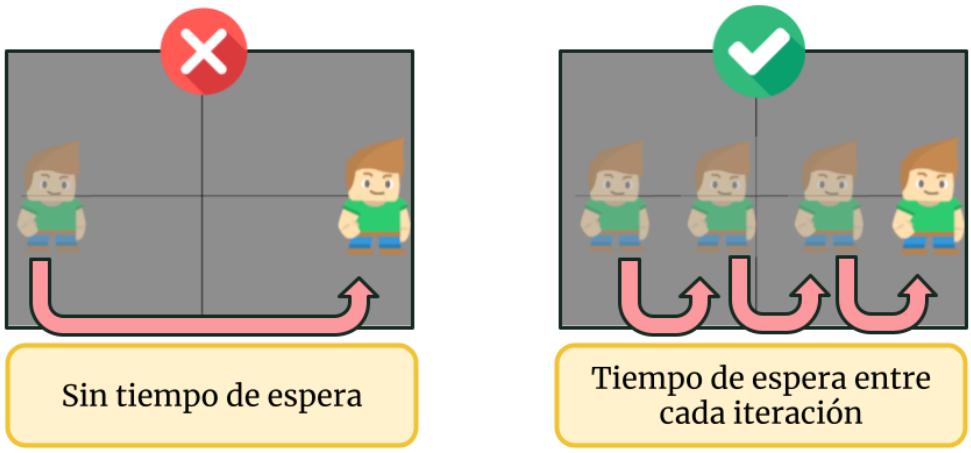


Figura 4.2: A la izquierda se ve el comportamiento inicial de Gesture Coding luego de tres desplazamientos, sólo se muestra la posición inicial y la final. A la derecha se observa el comportamiento deseado, con el personaje entre cada iteración.

Este problema es abordado integrando un tiempo de espera a cada bloque, de tal forma que el efecto del bloque se vea reflejado en el Área del Personaje antes de ejecutar la siguiente acción. Este cambio debió hacerse a todos los bloques, no sólo a los de repetición, dado que si el usuario ingresaba bloques de movimiento de manera secuencial se observaba el mismo efecto. Para esto, se implementa el método `invoke()`, que permite ejecutar una función luego de un tiempo de espera específico que es cuánto tomará el bloque en reflejarse en pantalla, es decir, cuánto tiempo estará el bloque "Ejecutándose" antes de pasar a la siguiente acción. Para el caso de bloques secuenciales basta con definir un delta específico; sin embargo, para los bloques de repetición se requirió la implementación de un sistema que permitiese calcular el tiempo que tomaría el ciclo completo en ejecutarse. Esto se debe a que cada bloque de repetición puede contener una cantidad indefinida de bloques dentro.

El manejo de cuánto dura cada bloque es manejado por la función `count()` dentro del Bloque Inicial, que entrega un vector con el "peso" de cada bloque. Este valor indica cuántas unidades de tiempo demora el bloque en ejecutarse. Por defecto, todos los bloques poseen un peso igual a 1, excepto los de tipo repetición, dado que pueden poseer bloques en su interior ejecutándose más de una vez.

Para poder definir el tiempo que demora un bloque de repetición, se calcula de manera recursiva el tiempo que demora cada bloque dentro del ciclo. De esta manera, se suma el peso de cada bloque y se multiplica por la cantidad de veces que se ejecuta el mismo dentro del ciclo. Para esto se definen las siguientes funciones dentro del bloque:

- **`Calculate`:** Ejecuta cada iteración del ciclo, estableciendo un delta de tiempo entre ellos.
- **`CalculateBehind`:** Ejecuta cada bloque de una iteración, indicando cuánto tiempo debe

quedar ejecutándose según lo indicado por su peso.

- Count: Retorna la suma total del tiempo del ciclo contando todas las iteraciones, además de un vector con los pesos de cada bloque. Esto permite que `CalculateBehind` permita ejecutar no sólo bloques de manera secuencial, sino que ciclos anidados.

De esta forma, el bloque de repetición puede determinar de manera recursiva cuánto demorará en ejecutarse. Es importante determinar este valor dado que pueden existir bloques conectados de manera secuencial luego del ciclo. Esto permite que el usuario vaya viendo el avance del ciclo en cada iteración.

Si bien esta implementación funciona de manera correcta para bloques que repiten código una cierta cantidad de veces, no funciona para repeticiones condicionales. Esto se debe a que no es posible predecir cuántas veces se va a iterar antes que la condición deje de cumplirse. Dado que no se puede temporizar la duración del bloque de repetición condicional, el sistema no puede ir pausando en cada iteración para que el usuario observe el progreso del ciclo. Esto rompe con el funcionamiento esperado del bloque, por lo que se decide no implementar el bloque de repetición condicional.

4.2.2. Variables

La integración de Variables a Gesture Coding está enfocada en desarrollar la noción de que existen valores que pueden ser almacenados por el computador para ser reutilizados luego. Para esto, se utiliza como metáfora el concepto de que una variable es un contenedor donde se guardan elementos. Este contenedor tiene un nombre y un valor, que puede ser reemplazado por otro. Esta metáfora es ampliamente usada al momento de enseñar el concepto de variable y, según Van Der Werf et al., cubre la definición completa de una variable [66]. La idea de que una variable es representada por un contenedor se mantiene a lo largo del diseño e implementación de los bloques de Variable, incluido el gesto que representa la categoría.

Diseño de Bloques

La categoría de Variables se divide en dos funcionalidades principales: Definición y Uso. La primera funcionalidad debe permitir guardar un valor en un bloque de variable que pueda usarse luego. Dado el estado del sistema, de momento las variables almacenarían sólo valores de tipo numérico. La segunda funcionalidad permitiría insertar los bloques de tipo variable en todo contenedor que sea compatible con bloques de tipo numérico, como son los bloques de operación o de repeticiones. Dado esto, se diseñan dos bloques específicos, ilustrados en la Figura 4.3:

- **Definición de Variable**: Bloque que permitiría asignar un valor a una variable. Dado que el sistema no maneja Strings, y hacer que el usuario escriba los nombres de variables a mano usando un teclado rompe con el propósito de Gesture Coding como tal, el usuario debe seleccionar de un DropDown una variable a definir. Este bloque también debe permitir redefinir una variable que ya poseía un valor asignado.
- **Uso de Variables**: Estos bloques almacenan el valor asignado a la variable, y no poseen ningún puerto macho o hembra, dado que la forma en que interactúan con otros bloques es insertándose en puertos específicos, como los habilitados para números en

bloques de operación.

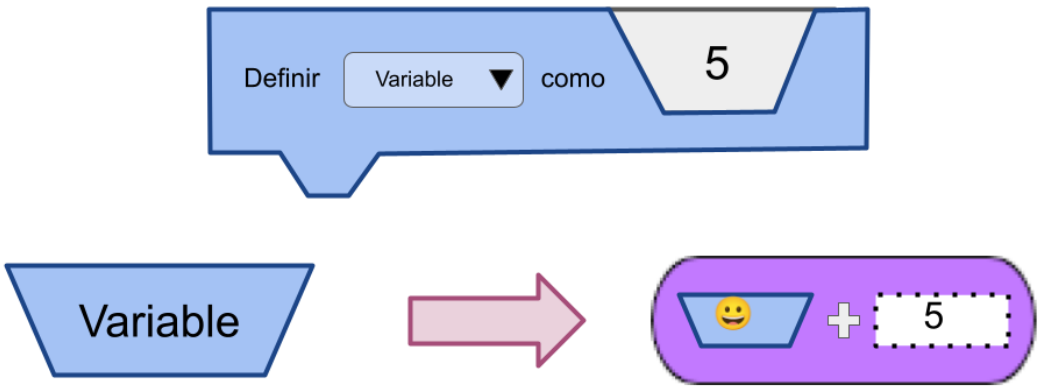


Figura 4.3: Diseño inicial del bloque de Definición y Uso de Variable, además de un ejemplo de una variable insertada en una operación.

La metáfora de interacción que guía el diseño de los bloques es que una variable representa un contenedor donde se pueden guardar valores. Dado esto, se integra en el diseño de ambos bloques un trapecio invertido, que hace el símil a un contenedor. De esta forma, el bloque de definición de variables posee un input de forma trapezoidal, haciendo el análogo con que el usuario está depositando en el contenedor el valor que desea guardar. A su vez, los bloques de uso de variables poseen esta forma trapezoidal, para desarrollar la intuición que están moviendo un contenedor que posee un valor dentro.

Los nombres de las variables son predefinidos anteriormente, y en lugar de utilizar texto, se emplean imágenes para representarlas, evitando así el uso de un teclado virtual para introducir los nombres de variables. Esta elección se justifica en la literatura, ya que el uso de imágenes en lugar de texto puede facilitar la comprensión, especialmente en un contexto educativo con niños [34]. Además, se evita que el usuario guarde sus datos en una variable con nombre "Variable X", sino que en la variable con un dibujo en particular. Esto permite que el usuario diferencie de mejor manera los valores, asegurando que cada variable posea un nombre o icono lo suficientemente distintos del anterior, respaldando la idea de que la diferenciación visual facilita el reconocimiento y comprensión de las variables [66]. Además, dado que el número total de variables predefinidas es limitado, los dibujos asignados a cada variable se mantienen constantes. Para implementar estos dibujos en los nombres de las variables, se utilizan emojis lo suficientemente distintos entre sí para evitar confusiones. La selección de emojis se basa en su color base, su representación y la potencial familiaridad del niño con el dibujo. Para esto, se consideraron aspectos cognitivos y estéticos respaldados por expertos de dominio.

Finalmente, para poder usar una variable fue necesario el diseño de una nueva interfaz que permitiese seleccionar la variable que se desea usar. Esta consiste en una ventana lateral que

Figura 4.4: Listado de variables, donde sólo el bloque con una cara sonriente tiene un valor asignado.

lista todas las variables disponibles actualmente. Para indicar qué variables tienen valores guardados, se utilizan colores en el menú de selección de variables. Si la variable tiene un valor almacenado, el color del bloque será azul. En caso contrario, este se verá de un color rojo claro, como se observa en la Figura 4.4.

La elección de utilizar emojis en lugar de texto para los nombres de las variables garantiza que todos los bloques de variable tengan un tamaño uniforme, eliminando la necesidad de ajustarse a nombres largos. Esto simplifica la disposición de los bloques en el espacio de trabajo y puede contribuir a una experiencia de programación más organizada y fácil de manejar para los usuarios, especialmente teniendo en cuenta que el propósito principal de esta herramienta es ser el primer acercamiento de los estudiantes a la programación. Además, esta decisión puede disminuir la carga cognitiva al eliminar la necesidad de ajustar y organizar bloques de diferentes tamaños, permitiendo a los usuarios enfocarse más en la lógica de programación y menos en la gestión del diseño.

Gesto

Siguiendo con la metáfora planteada para la interacción con variables, se plantea que el gesto a representar esta categoría de bloques siga esta noción de ser un contenedor. Para esto, se prueban distintos gestos que se pueden dibujar por el usuario, ilustrados en la Figura 4.5: trapezoide invertido, cuadrado, letra “U” o letra “V”. El primer gesto planteado, el trapezoide invertido, es un dibujo similar a la de un contenedor que, además, corresponde a la forma que tienen los bloques. Sin embargo, dibujar este bloque resulta complejo dado la precisión necesaria es alta. Si el usuario no resaltaba lo suficiente los vértices, o no hacía las líneas lo suficientemente rectas, el sistema interpretaba el gesto como un círculo, abriendo el menú de bloques de tipo numérico. Para evitar este problema, se prueba dibujando no dibujar el trazo superior, de mayor tamaño. De esta forma, se obtiene una forma de contenedor

o plato desde una vista lateral. Sin embargo, surge el mismo problema anterior pero esta vez con los gestos condicionales, el sistema interpretaba frecuentemente el gesto como una flecha horizontal. Finalmente, para evitar conflictos con otros gestos, se descarta el dibujo de trapezoide invertido.

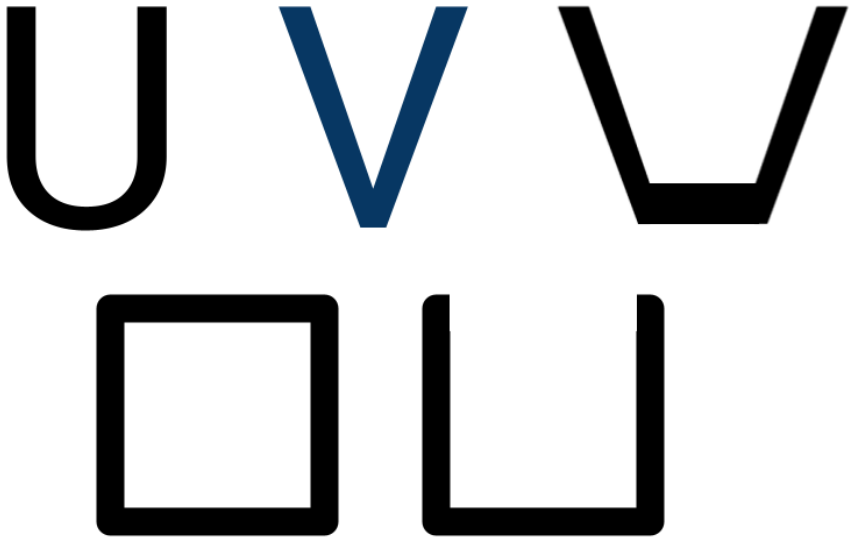


Figura 4.5: Gestos propuestos para variables, siguiendo la metáfora de que una variable es un contenedor. Finalmente, el elegido es la letra "V".

En el caso del gesto cuadrado, los resultados fueron similares. Después de realizar pruebas internas, se determinó que si el usuario no marcaba correctamente los vértices o no dibujaba líneas lo suficientemente rectas, el sistema interpretaba el gesto como un círculo. Asimismo, al omitir el trazo superior del cuadrado, el sistema lograba reconocer el gesto de manera más consistente, pero este gesto resultaba desafiante de mantener, ya que los vértices debían marcarse de manera clara. Luego, se exploró la posibilidad de utilizar el gesto con forma de letra "U" mayúscula, el cual era más fácil de dibujar de manera consistente. No obstante, debido a su curvatura, surgió un conflicto con el gesto circular utilizado en los bloques numéricos. Esto llevó a que el sistema interpretara de manera incorrecta tanto el gesto circular como el de la letra "U".

Finalmente, se llega al gesto con forma de letra "V" como una combinación de los gestos anteriores, es decir, una letra "U" pero con un vértice. Dado que este gesto sólo posee dos trazos rectos, resulta más fácil de dibujar. Además, no existe la restricción de que el ángulo sea recto, por lo que se puede programar una tolerancia alta respecto a la forma de la letra "V". Este gesto se logra dibujar de manera consistente, y no entra en conflicto con los otros gestos. Para evitar conflicto con el caso particular del gesto triangular, propio de los bloques condicionales, se cuida que la orientación del vértice sea apuntando hacia abajo.

Implementación

La integración de variables en el sistema aborda dos desafíos a solucionar: Cómo manejar variables en el sistema, e integrar el menú de variables a la interfaz. La principal dificultad

de esta implementación es que, una vez creados los bloques, estos no se pueden modificar en tiempo de ejecución. Esto implica que las variables creadas son inmutables. Para solucionar este problema, se tendría que hacer un refactoring de cómo el sistema ejecuta los bloques que, por tiempo, no se implementa en este trabajo.

Figura 4.6: Formulario para crear variables al que se accede luego de ingresar el gesto "V" de Variables.

Dado esto, el bloque de definición y modificación de variables cambia su funcionamiento. Como no se pueden crear o modificar variables mientras el código se está ejecutando, se rediseña el bloque para funcionar como un formulario dentro del menú de bloques de variables, ilustrado en la Figura 4.6. Este nuevo formulario sigue con el diseño de bloque para mantener consistencia dentro del sistema, y está compuesto de un Desplegable con los emojis disponibles para etiquetar la variable y un input de tipo numérico donde se puede asignar el valor deseado. De esta manera, para crear una variable el usuario deberá seleccionar de un DropDown el emoji que quiere que represente su variable, y un valor de tipo numérico de una teclera virtual.

Para almacenar valores en el sistema se implementa una clase caché, compuesta de una lista `Value` de valores igual a la cantidad de variables existentes en el sistema, todas inicializadas en 0. Para modificar las variables, se implementan dos funciones en la clase `NewProjectManager` que controla la escena `Nuevo Proyecto`, `saveValue` para asignarle un valor a la variable, así como cambiar el color del bloque, y `getValue` para obtener el valor asignado a una variable.

Actualmente, en el sistema existen seis variables predefinidas que pueden adoptar valores, siendo cada una un objeto individual de Unity dentro de la escena `Nuevo Proyecto`. Cada variable contiene un nombre `VariableBlocki` y un valor que indica la posición en la que se encuentra su valor asignado en la lista de la clase Caché. Para poder usar estas variables, el usuario debe presionar el botón X de su Joy-Con derecho en el Área de Trabajo, donde se abrirá una ventana tipo modal con las 6 variables disponibles en el sistema. Dependiendo del contenido de la variable, el bloque se mostrará de un color u otro. Si una variable tiene un

valor asignado por el usuario, entonces será de color azul y le permitirá al usuario crear una copia del código para ser usado. En el caso de que no tenga un valor asignado, entonces el bloque será de color rojo claro y no se le permitirá al usuario seleccionar el bloque. De esta forma, el usuario puede saber qué variable sacar, además de saber cuántas variables no han sido ocupadas, como se ilustra en la Figura 4.4.

Finalmente, el proceso de uso de variables se descompone en dos procesos: Asignarle un valor a la variable y crear los bloques de variable para integrarlos al código:

- **Asignar valores**: Se crea la ventana con el formulario para crear variables donde el usuario puede elegir un emoji desde un desplegable y el valor numérico a asignar, confirmando su decisión al pulsar el botón Aceptar. Al mismo tiempo, `NewProjectManager` le indica al Caché que debe hacer el cambio en la lista de valores y modifica el color del bloque a azul ahora que ha sido inicializado.
- **Crear bloque**: Al pulsar el botón X en el Joy-Con derecho, se abre la ventana `VariableListView` en donde podemos ver el estado actual de todas las variables, siendo azules las que ya tienen un valor asignado en el caché y rojas las que no. Al seleccionar un bloque, la función `CreateObject` obtiene la posición del bloque en la lista de valores de la clase Caché y le indica a `NewProjectManager` que cree el nuevo bloque con el valor asignado. Si el bloque es de color azul, se crea el objeto para ser usado, pero si es de color rojo entonces el sistema mostrará un mensaje de error.

Finalmente, para manejar el Desplegable del menú de creación de variables se utilizó DropDownTMP propio de Unity. Sin embargo, dado que este desplegable está programado para funcionar con mouse, el Scroll dentro de las opciones está limitado a ser con mouse, por lo que no reconoce los Joy-Con. Dado esto, se limitó a un máximo de seis variables, que es el número de elementos que permitía el desplegable antes de comenzar a usar el Scroll. Es posible modificar esta librería para compatibilizar el Scroll del desplegable con los controles, pero se dejó fuera del alcance del proyecto por tiempo.

4.2.3. Diseño General

Una vez completada la implementación, se realizaron ajustes generales a la usabilidad del sistema. En primer lugar, la implementación del menú de selección de variables implicó ajustar los tamaños de las Áreas de Trabajo y Personaje, dándole más espacio al usuario para programar. Además, se hizo un ajuste general de íconos y colores pensando en el público objetivo, aumentando el contraste para hacerlo más llamativo, como indica la Figura 4.7.

En particular, se revisó el color de todos los bloques del sistema para que los bloques de las nuevas categorías no se mezclaran con los bloques ya creados. En ese sentido, se procuró elegir colores que se distinguieran estando uno al lado del otro, además de que su combinación se vea armoniosa. Para esto, se tomó en cuenta una paleta de colores basada en triadas, tomando como inspiración los colores de los Joy-Cons disponibles en el mercado, según indica la Figura 4.8.

Figura 4.7: Ajuste realizado a los colores originales de Gesture Coding (Izquierda), aumentando la saturación de los mismos (Derecha).

Figura 4.8: Esquema de color de los bloques basado en los controles disponibles para Nintendo Switch, cuidando que sean distinguibles entre sí.

4.3. Evaluación Funcional

Una vez finalizada la extensión del sistema, se realizó un estudio de usuarios piloto para evaluar la plausibilidad de la metodología de trabajo y la funcionalidad de la extensión, verificando el correcto funcionamiento y su comprensibilidad. Para esto, se realiza una convo-

catoria abierta a estudiantes de 1er año de Ingeniería para evaluar la herramienta, dado que el objetivo del estudio no depende de la muestra utilizada para la evaluación. Los estudiantes se encontraban entre los 18 y 20 años de edad y se encontraban cursando una asignatura introductoria de programación en Python. El muestreo fue realizado por conveniencia, participando los estudiantes disponibles en los horarios de atención.

El principal objetivo de este estudio fue evaluar el desempeño de la aplicación frente a un usuario real. Cada individuo estuvo cinco minutos explorando libremente Gesture Coding con los Joy-Con, para luego asignarle un problema de programación que ocupara bloques de todas las categorías que debían resolver en un máximo de 15 minutos. Finalmente, se midieron las cargas cognitivas y físicas de los usuarios usando el cuestionario NASA-TLX, además de preguntas generales al final de la actividad.

Las tareas a realizar por los estudiantes fueron:

- Ingresar a pantalla de gestos
- Dibujar figuras (Puede ser una estrella, flechas, etc.)
- Evaluar qué tan intuitivos son los gestos. Reconocer los gestos de cada bloque. Interpretación de los gestos de manera cualitativa.
- Crear un bloque de cada tipo
- Eliminar todos los bloques creados
- Conectar bloques para que al correrlos hagan lo siguiente:
  - Crear una variable igual a 10. Este será el paso del personaje.
  - Mover al personaje a la derecha 20 veces con un ciclo.
  - Antes de cada paso, verificar la posición del personaje en el eje X. Si es mayor a 90, posicionar el personaje en las coordenadas x:-150 y: 0

Durante la realización de la actividad, el estudiante estaba acompañado en todo momento de un monitor que le resolviera dudas sobre el sistema o sobre la actividad en particular a realizar. Al mismo tiempo, el monitor se encargaba de reportar feedback no verbal del estudiante, como actitudes frente al código o estados de ánimo. Cada sesión constó de tres computadores con Gesture Coding, cada uno con un par de Joy-Cons para controlar el lenguaje.

Según lo reportado por los estudiantes en sus cuestionarios, todas las cargas tanto físicas como cognitivas obtuvieron un puntaje promedio bajo, siendo la carga mental y el esfuerzo los que obtuvieron el puntaje más alto, ambos con 4.5 puntos de 10. Por otro lado, en el focus group la mayoría de los usuarios encontraron la herramienta apropiada para aprender conceptos básicos de programación pese a sus limitaciones: *"Es un programa interesante y considero que bueno para aprender los principios de programación."* (P2, H, 18 años); *"Fue muy entretenido! Creo que a los niños les encantara y veo que puede servir mucho para comenzar a programar."* (P4, M, 18 años). Destacaron, además, la asociación de los bloques con gestos libres, puesto que facilitan la comprensión del programa, haciendo el lenguaje más intuitivo de usar: *"En mi opinión, las formas de los bloques permiten al usuario identificar de mejor manera el flujo del programa y el hecho de que uno pueda observar directamente en pantalla lo que hace el código, hace mucho más simple algo que puede llegar a ser bastante abstracto."* (P7, H, 18 años); *"Respecto a los controles, fueron agradables de*

usar y siento que el hecho de poder asociar cada función con un movimiento, puede llegar a servir bastante para internalizar ciertas cosas." (P9, M, 18 años). También comentaron que algunos experimentaron dificultades con la sensibilidad de los controles de movimiento y manipulación de bloques, sugiriendo mejoras a nivel de interfaz y detección de gestos: *"Me gustó, aunque tuve dificultades con los controles"* (P3, M, 18 años). En general, a pesar de algunas dificultades, los usuarios disfrutaron la experiencia y consideran que es una buena introducción a programación.

Por otro lado, los monitores reportaron que, inicialmente, los usuarios presentaban visible frustración para adaptarse a los controles y su sensibilidad, pero que luego de un par de intentos lograban controlarlos con seguridad: *"En un inicio los estudiantes se veían complicados usando los controles, haciendo hincapié en la sensibilidad de la palanca y en lo difícil que era dibujar los gestos, aunque después le agarraron el truco a los gestos"* (Monitor 2); *"Al principio hacían muchos comentarios sobre que les gustaba la idea, pero debían calibrarse mejor los controles. Después de intentar un rato, lograron adaptarse, pero mencionaban que aún así les gustaría que se ajustara la sensibilidad de la detección"* (Monitor 3). Una vez acostumbrados al mecanismo de interacción, su frustración pasó de la creación de bloques a la solución del problema. Los estudiantes que ya estaban acostumbrados a este tipo de controles al haberlos utilizado antes, no tuvieron mayor problema con el mecanismo de interacción: *"Al momento de plantearles el desafío de programación, muchos indicaban que no sabían dónde comenzar o qué bloques usar, y eso que los estudiantes supuestamente poseían experiencia previa programando. Sin embargo, luego de dar un par de pistas pudieron hacerlo"* (Monitor 1); *"Durante la etapa de exploración, y después que se acostumbraron a los controles de movimiento, no tuvieron mayor problema al crear y usar bloques. Les costó, eso sí, usarlos para resolver la tarea que les dejamos"* (Monitor 2); *En general, tuvieron dificultad para usar los bloques, no al momento de crearlos, sino que usarlos para programar algo. Sin embargo, un par de estudiantes destacaron al lograr el desafío de inmediato sin mayor tiempo de adaptación a los controles. Al consultarles qué les pareció, indicaron que tienen experiencia usando Joy-Cons y programando.* (Monitor 3). En relación al rendimiento de Gesture Coding, los estudiantes no experimentaron problemas técnicos significativos, siendo la desconexión de los controles el más común. Para esto, se dispuso de controles extra que se conectaron al computador en caso de algún error. Finalmente, se concluye que Gesture Coding permite a usuarios reales llevar a cabo programas sencillos sin mayor complicación.

Capítulo 5

Espira 3: Evaluación

En la tercera Espira de este trabajo, se diseña un caso de estudio con el público objetivo real, niños de entre 10 y 12 años, con la intención de verificar las hipótesis planteadas en este trabajo, es decir, que el uso de Joy-Cons de Nintendo Switch permite disminuir la barrera de entrada a programación. Este caso de estudio se compuso de dos actividades, una Prueba A/B Entre Sujetos y un estudio siguiendo el protocolo Thinking Aloud.

5.1. Caso de Estudio

El foco de esta evaluación es comprender qué ocurre y cómo afecta la percepción del estudiante por la programación al intervenir el mecanismo de control de la herramienta utilizada. Dado esto, se optó por aplicar una metodología de Caso de Estudio, puesto que su enfoque se acomoda con observar un fenómeno y explicar cómo o por qué ocurre [77].

El Caso de Estudio realizado es de carácter **Único** dado que su foco es evaluar el mecanismo de control de Gesture Coding en contraste con el de Scratch. Por otro lado, también consistió en un Caso de Estudio **Holístico** puesto que el enfoque estuvo en una unidad de análisis, la percepción del individuo en la programación. Finalmente, dado que no se posee un análisis previo en esta materia, se le otorga el carácter de Caso de Estudio **Exploratorio**.

Además, el diseño de esta evaluación fue de método mixto secuencial, dado que se levantan datos de tipo cuantitativos en una primera etapa, y se profundizan las observaciones realizadas con un segundo levantamiento de datos cualitativos. Dado esto, el análisis de los resultados será de carácter explicativo, relacionando los datos obtenidos de manera cuantitativa con las observaciones cualitativas de la segunda etapa.

Para este caso de estudio, la primera etapa consistió en una Prueba A/B Entre Sujetos, enfocada en medir las cargas físicas y cognitivas percibidas por los estudiantes, así como su satisfacción y aceptación de la herramienta. Luego, en la segunda etapa, se levantan datos de tipo cualitativo mediante el protocolo Thinking Aloud, profundizando en los conocimientos adquiridos durante la primera etapa.

5.2. Prueba A/B

Una vez Gesture Coding cubre el conjunto minimal de habilidades de pensamiento computacional, habiendo integrado Ciclos y Variables, se diseñó y aplicó un estudio de usuarios para evaluar el impacto del mecanismo de interacción usado por la herramienta en la motivación de los estudiantes. Para esto, se diseñó un experimento en el cual se comparan la expectativa y valor percibido de los estudiantes usando Gesture Coding (usando gestos), y Scratch (con mouse y teclado). La prueba realizada es de tipo A/B Entre-Sujetos, en la que un grupo ejecuta el experimento con Scratch, mientras el otro grupo lo hace con Gesture Coding.

5.2.1. Participantes

Para este experimento, se reclutaron escolares de entre 10 y 12 años sin conocimiento previo en programación. Se selecciona ese rango etario porque, de acuerdo a la literatura, es una edad idónea para desarrollar conocimientos de Pensamiento Computacional en niños [52]. Por otro lado, se requiere que no tengan experiencia previa en programación debido a que se busca estudiar el impacto de la herramienta al romper la barrera de entrada a programar. Así, si han tenido contacto o experiencia programando, es probable que ya hayan traspasado esa barrera, provocando un sesgo en su evaluación.

Consideraciones éticas

A los participantes del estudio se les pide firmar un formulario de Asentimiento informado, indicando que están de acuerdo con participar en el estudio. Dado que los usuarios son menores de edad, se les pide el Consentimiento informado a los padres y/o tutores legales. Esto se hace al momento de ser seleccionados para el estudio, luego de la inscripción pero antes del comienzo del taller. El protocolo experimental siguió las recomendaciones de la APA para estudios empíricos con sujetos humanos. Asimismo, el protocolo experimental fue aprobado en términos éticos por el Comité de Ética Institucional para la Investigación y Bioseguridad de la Facultad de Ciencias Físicas y Matemáticas de la Unviersidad de Chile.

Reclutamiento

El reclutamiento de los participantes se hizo mediante un muestreo deliberado de contacto indirecto [9], el cual consistió en un llamado por las redes sociales del Departamento de Ciencias de la Computación de la Universidad de Chile a un taller de programación. Esto permite que el llamado pueda ser compartido y difundido por los mismos seguidores de las redes sociales, inclusive si no participan de la actividad. Dentro del llamado a inscripciones se especifica y hace énfasis en que el taller tiene como requisito no tener experiencia programando y el rango etario de 10 - 12 años.

Criterios de inclusión y exclusión

El criterio de inclusión en el estudio es que fueran niños y niñas entre 10 y 12 años que no reporten experiencia previa programando. Esto se controla mediante autorreporte el formulario de entrada, donde se les pregunta directamente si tienen experiencia previa programando y en qué lenguaje/plataforma.

El criterio de exclusión es que el individuo posea experiencia previa programando, o esté fuera del rango etario definido anteriormente.

Verificabilidad de la muestra

Dado que el experimento se realizó en formato de taller, esta información se verificó en el formulario de inscripción preguntando directamente cuál es su experiencia programando y la edad. Con esta información se hace el filtro en los participantes del taller. Además, se consultó sobre la experiencia utilizando Controles de Movimiento, especialmente Joy-Cons de Nintendo Switch. Se tomó en cuenta que un usuario con experiencia en el manejo de este sistema podría tener un mejor desempeño en la aplicación, considerándolo como un usuario "experto".

Posibles sesgos de la muestra

Al difundirse por las redes sociales de la universidad, los potenciales participantes pudieron estar relacionados de alguna forma con la misma. Por otro lado, es posible que los inscritos sean personas que, sin haber tenido experiencia, están interesadas en programación de antes. Esto puede introducir un sesgo de autoselección al momento de evaluar su expectativa, dado que podría ser más alta que el promedio. Además, es posible que la interpretación de "experiencia programando" difiera entre los inscritos, lo que podría sesgar la muestra al incluir escolares que ya han tenido un primer encuentro con la programación. Para abordar este posible sesgo, se solicita a los participantes que especifiquen el tipo de experiencia que poseen, permitiendo así refinar la muestra. Del mismo modo, en el caso de los controles, los estudiantes podrían tener un desempeño mejor o peor al utilizar la herramienta debido a su familiaridad con los controles utilizados.

Por otro lado, dado que el taller fue realizado en cuatro ocasiones, se debieron considerar factores para reducir la probabilidad de un posible sesgo: (1) Consistencia en la presentación de contenidos y actividades; (2) Distribución aleatoria de participantes; (3) Mantener el entorno y condiciones ambientales lo más consistentes posible.

Tamaño de la muestra

Para determinar el tamaño mínimo esperado de la muestra, se utilizó el análisis de potencia estadística, que permite indicar el mínimo necesario para detectar un efecto significativo, si es que existe. Con un nivel de significancia de 0.05, una potencia estadística de 0.8 y una magnitud de efecto esperado de 1.1, se estima que el tamaño total mínimo de la muestra es de 30 individuos por grupo. Estos parámetros son estándares comúnmente utilizados en el diseño de experimentos y permiten equilibrar la probabilidad de encontrar un falso positivo (error tipo I) y la probabilidad de no detectar un efecto real (error tipo II). Dado que el taller se llevará a cabo en varias ocasiones debido a la disponibilidad limitada de controles y computadores, la recolección de datos se distribuye en cuatro sesiones con grupos de 10 a 15 participantes por sesión. Esta decisión se toma para maximizar la participación en el estudio a pesar de las limitaciones de recursos, y se espera que esto contribuya a superar el tamaño estimado de la muestra para evaluar el efecto esperado.

5.2.2. Materiales

Independientemente de la herramienta utilizada para programar, cada estudiante necesitó un computador y un mouse, ambos provistos por la Universidad de Chile. En el caso del grupo control, se necesitó que tuviera instalada la aplicación de Scratch 3.0, mientras que para el caso del grupo experimental se necesitó que tuviese instalada la aplicación de Gesture Coding, además de un par de Joy-Con por participante.

Por otro lado, se necesitó una sala para desarrollar el taller de manera presencial. Estas fueron proporcionadas por la Universidad de Chile, equipadas con pupitres, un pizarrón y un proyector, con una capacidad máxima de 50 estudiantes. En cuanto a la distribución de participantes en el estudio, se procuró que estuvieran distanciados al menos por un puesto, de tal forma que tuvieran espacio para moverse sin entorpecer al compañero del lado, como se observa en la Figura 5.1.

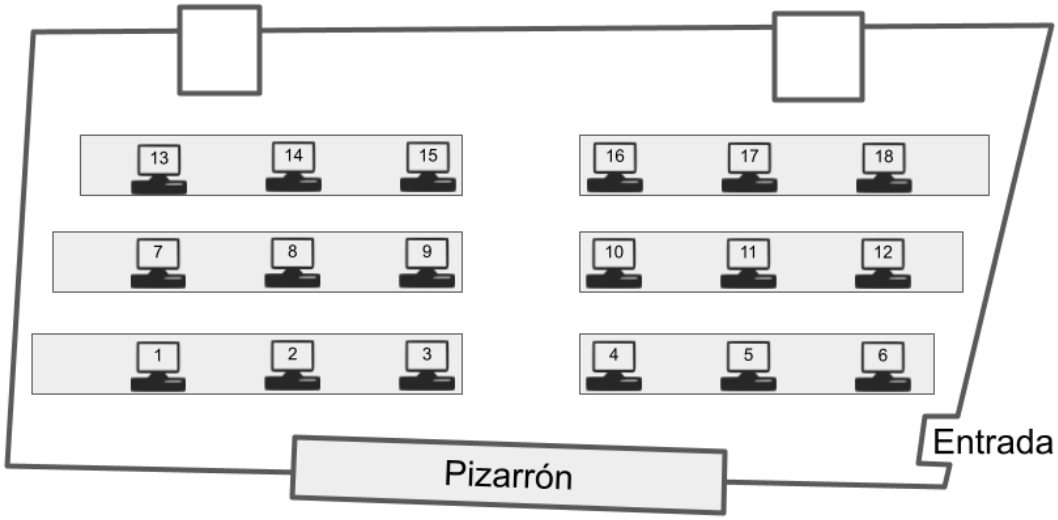


Figura 5.1: Distribución de puestos para estudiantes en la sala de clases.

Finalmente, dado el número de estudiantes en la sala, fue necesaria la presencia de ayudantes que monitorizaran a los estudiantes. Se hizo necesaria la presencia de un ayudante cada 5 niños por sesión, por lo que, incluido el investigador, fueron 3 monitores en total por sesión. Estos fueron estudiantes de último año de la Ingeniería Civil en Computación con experiencia en ayudantías, y fueron reclutados de manera voluntaria. Para este taller, fue necesario un entrenamiento previo donde se les enseñó a usar ambas plataformas y se acordó la distribución de estudiantes para cada uno. Además, dado que el estudio también consideró el lenguaje no verbal de los participantes durante la experiencia, se les indica a los monitores anotar y comentar su percepción sobre el desempeño y actitud de los niños frente a las herramientas. Para esto, se le proporcionó a los monitores una pauta de elementos a evaluar y considerar al momento de guiar estudiantes, reportado en el Anexo A.

Condiciones de control y de variación

Las condiciones de control y variación son idénticas en todas las sesiones del taller, salvo por la aplicación utilizada y su mecanismo de interacción. Estos corresponden a Mouse y Teclado en el caso de Scratch, y un par de Joy-Con para Gesture Coding:

- Computador con conexión a internet de las siguientes características:
  - **Modelo:** Notebook Hp Probook 445 G8
  - **Sistema Operativo:** Windows 11 Pro
  - **Memoria RAM:** 16 GB
  - **Procesador:** AMD Ryzen 7 5800U with Radeon Graphics 1.90 GHz
  - **Almacenamiento:** SSD PCI 512 GB Interno
  - **Software:** Office 365
- Par de Joy-Cons de Nintendo Switch

Plausibilidad del estudio

Los instrumentos de medición utilizados están validados por sus respectivos autores:

- Smiley-o-meter [79]: Basado en el trabajo de Zaman et al. para evaluar el nivel de disfrute de un sistema por niños.
- NASA-TLX[30]: Adaptado del instrumento desarrollado por Ames Research Center (AMC) en 1988 para evaluar carga cognitiva. Se aplica una versión simplificada del instrumento usada en el estudio piloto de este trabajo, formulando las preguntas de tal forma que pueda ser entendida por el lenguaje de los participantes de 10 a 12 años. El instrumento fue piloteado con niños del público objetivo así como expertos de dominio.
- Expectativa[15]: Adaptado del instrumento implementado por Katelyn M. Cooper et al. para evaluar expectativa en estudiantes.

El tamaño de la muestra está validado por el análisis de potencia estadística. Para su estimación se usó el software G*Power [21] con un nivel de significancia de 0.05, potencia estadística de 0.8 y magnitud de efecto esperado de 1.1, valores seleccionados en base a consideraciones estándar en el diseño de experimentos.

Replicabilidad del estudio

Se necesita un computador de, al menos, las características indicadas en el apartado de Condiciones de Control y Variación con conexión Bluetooth. Junto a esto se necesita un par de controles Joy-Con de Nintendo Switch. En cuanto a software, el código fuente de la aplicación puede construir un archivo ejecutable *.exe* que se puede correr en el computador. Para el caso de Scratch, el instalador de la aplicación se encuentra en su sitio web [1].

5.2.3. Definición del Experimento

Con el objetivo de evaluar el rendimiento de Gesture Coding en comparación con Scratch en términos de romper la barrera de entrada a la programación, se lleva a cabo una prueba

[1] https://scratch.mit.edu/download

A/B Entre-Sujetos, donde cada grupo utilizó una aplicación diferente, ya sea Scratch o Gesture Coding. Se usa Scratch como punto de comparación debido a que es una herramienta establecida para enseñar programación a principiantes. Se entiende por la expresión "Romper la barrera de entrada" a la programación como la capacidad de Gesture Coding de facilitar y hacer más accesible el proceso de aprendizaje de programación, especialmente para aquellos sin experiencia previa.

La asignación de los participantes se realizó de forma aleatoria, excepto para aquellos con experiencia en el uso de los controles Joy-Con, quienes fueron distribuidos equitativamente en ambos grupos. Además, se balanceó la muestra por edad y por género para evitar sesgos de género y procurar sesiones lo más homogéneas posible. La toma de datos en diferentes sesiones se realizó debido a limitaciones de recursos y disponibilidad de controles Joy-Con. Al distribuir las sesiones en días sábados distintos, se procuró que la muestra estuviera balanceada cada día de manera independiente para evitar posibles sesgos temporales y maximizar la participación en el estudio. De las cuatro sesiones, se asignó la primera como el grupo control con Scratch y las otras tres como el grupo experimental con Gesture Coding.

Se observa un desbalance en el tamaño de las muestras entre el grupo con Scratch y el grupo con Gesture Coding en la distribución de sesiones, donde este último reportó mayor cantidad de participantes. La literatura respalda la eficacia de Scratch en la enseñanza de programación a principiantes [20], lo que motivó la decisión de centrar la recolección de datos en Gesture Coding para obtener una comprensión más detallada de su impacto en la superación de la barrera de entrada a la programación. Sin embargo, es fundamental abordar cuidadosamente cualquier posible sesgo introducido por el desbalance en el tamaño de las muestras, asegurándose de que no afecte la validez de las comparaciones realizadas en el estudio.

Tipo de estudio

Se utilizó un enfoque de método mixto secuencial explicativo para el estudio, comenzando con la recopilación de datos cuantitativos, como los tiempos de desarrollo y la carga cognitiva. Posteriormente, se recopilaron datos cualitativos que incluían las opiniones de los estudiantes sobre la aplicación. Todos los cuestionarios fueron completados en línea por los participantes, y se tomaron notas por parte de cada monitor.

Proceso de experimentación

El reclutamiento se inició mediante un llamado abierto en redes sociales, solicitando a los interesados que indicaran su preferencia en relación con las fechas disponibles. Posteriormente, se procedió a la selección de los estudiantes inscritos que cumplían con los requisitos de edad y no tenían experiencia previa en programación.

Una semana antes de cada sesión, se realizaba un balance con los inscritos y se preparaba una nómina tentativa para la actividad. Luego, se enviaron instrucciones previas para confirmar la participación en la actividad. Estas indicaban que debían completar un formulario de Consentimiento Informado por parte de los padres y/o tutores legales, así como uno de Asentimiento por parte de los propios participantes. En caso de no recibir respuesta a estos formularios, se seleccionaba al siguiente inscrito que cumplía con los requisitos y se repetía el

proceso hasta llegar a los 15 inscritos por sesión. Esto se hacía con la intención de asegurar un mínimo de 10 participantes confirmados, considerando la posible ausencia de alguno.

El estudio consistió en un taller de 3,5 horas a realizarse en tres bloques separados por dos recesos de 10 minutos. En los minutos iniciales de los primeros dos bloques se les introduce un aspecto específico de la aplicación correspondiente a esa sesión, ya sea Scratch o Gesture Coding, para luego dejarlos explorarla de manera libre. En el tercer bloque se les entrega un desafío de programación que integra los conocimientos de los primeros dos bloques. Este proceso es análogo para todas las sesiones, siendo la única diferencia la herramienta que manejan los estudiantes.

Bloque 1: Se da la bienvenida a los estudiantes y se les aplica las preguntas de entrada, orientadas a evaluar su expectativa previa respecto a la programación. Luego, se introducen los bloques de Secuencialidad y Variables de tal forma que todos comiencen con una base común para explorar la herramienta.

Bloque 2: De manera similar al bloque anterior, se introducen los bloques de Condicionales y Ciclos, dado que presentan un nivel de complejidad mayor a los presentados anteriormente, para luego pasar a una fase exploratoria de dichos bloques. Además, se les indica que los combinen con los bloques Secuenciales y Variables.

Bloque 3: Se les entrega desafío de programación a resolver usando el conocimiento adquirido. Tanto el planteamiento como la resolución del desafío fue piloteado y probado anteriormente tanto con expertos como con programadores novatos. Con esta tarea se procura evaluar cómo se desenvuelve el participante frente a la herramienta de manera libre, pero con un objetivo a realizar.

El desafío consistió en lograr hacer que el personaje se mueva hacia la derecha constantemente. Si este se topaba con el borde derecho de la pantalla, este debía transportarse al borde izquierdo y seguir avanzando, repitiendo el proceso. Luego de indicarles el contexto a los participantes, se les entrega como ayuda las siguientes instrucciones:

- Conectar bloques de tal forma que "Programen un código que mueva al personaje hasta que se salga de la pantalla y se transporte al otro extremo". Para esto:
  - Crear una variable igual a 10. Este será el paso del personaje.
  - Mover al personaje a la derecha 20 veces con un ciclo.
  - Antes de cada paso, verificar la posición del personaje en el eje X. Si es mayor a 90, posicionar el personaje en las coordenadas x:-150 y: 0.

En la Fig 5.2 se muestra una solución posible al problema. Sin embargo, múltiples soluciones eran aceptadas, siempre y cuando cumplieran con lo propuesto en el desafío.

Luego, se les aplica un cuestionario de preguntas de salida, presente en Anexo B, que consiste en:

- Cuestionario NASA-TLX para medir la carga cognitiva y física de cada herramienta.
- Smiley-o-meter para evaluar qué tan disfrutable fue la experiencia del estudiante durante el experimento.

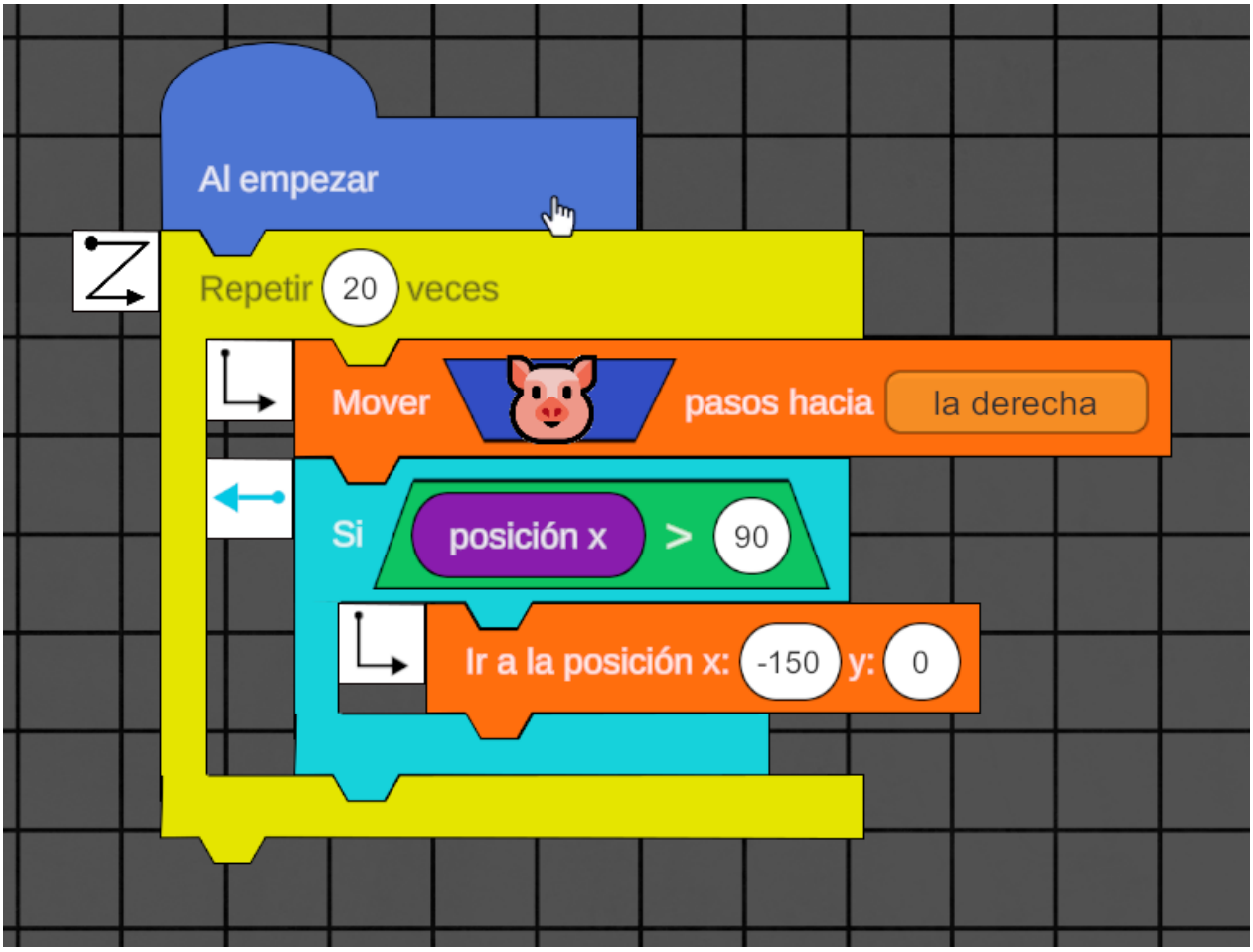


Figura 5.2: Ejemplo de respuesta esperada de la última tarea en Gesture Coding.

Finalmente, se concluye con preguntas generales del taller para evaluar percepciones generales de los participantes.

Luego de una semana después de terminada la actividad, se contacta a los estudiantes para completar un último cuestionario evaluando su expectativa respecto a la programación. Este se realiza una semana después para mitigar posibles sesgos. Si se aplica este cuestionario al final del tercer bloque, hay una alta probabilidad de que exista un sesgo por el *Novelty Effect* [64], es decir, la emoción por haber aprendido y hecho algo nuevo. Dado que se está evaluando el valor percibido por los estudiantes, este efecto podría provocar que se encuentre un falso positivo, es decir, que los estudiantes reporten un mayor valor percibido del real.

5.2.4. Instrumentos de Recolección de Datos

Los instrumentos de medición de la actividad corresponden a formularios online en Google Forms, simplificando el proceso de responder los cuestionarios al entrega. Además, se solicita a cada monitor tomar notas sobre el proceso en general ya sea en papel o de manera digital. Para esto se realizó una sesión de entrenamiento con los monitores donde se les entregó instrucciones operativas y a qué estar pendiente en cuanto a la actitud de los participantes del estudio. El tiempo de demora de cada tarea se toma desde que se termina la explicación hasta

que el estudiante nos indica que ha terminado y es registrado por cada monitor. Finalmente, el cuestionario de expectativa una semana después se hace de manera online con Google Forms, replicados en Anexo B.

Validez y confianza de los instrumentos

Los cuestionarios utilizados para este estudio están validados y estudiados por sus respectivas fuentes:

- **Smiley-o-meter [79]**: Basado en el trabajo de Zama et al., permite evaluar el nivel de disfrute de un sistema por niños. Consiste en una escala Likert de cinco puntos, donde cada uno es representado por el dibujo de una cara con distintas emociones. De esta forma, el niño o niña puede indicar cómo se sintió durante la experiencia o cómo lo hace sentir una pregunta en particular. En este experimento se usó para determinar si el participante se divirtió en la experiencia y la percibió como disfrutable, es decir, se evalúa el valor percibido. Las preguntas aplicadas en esta sección, junto al parámetro que evalúa son:
  - **Utilidad:** *Creo que gracias a la computación puedo resolver problemas de mi vida diaria.*
  - **Sesgo de Género:** *Los hombres y las mujeres pueden ser igual de buenos en computación.*
  - **Confianza:** *Cuando uso un computador, me siento seguro/a de lo que estoy haciendo.*
  - **Valor:** *Creo que es valioso para mí saber computación.*
  - **Engagement:** *Me sentí bien programando.*
  - **Satisfacción:** *Logré superar el desafío de programación planteado.*
- **NASA-TLX [30]**: Adaptado del instrumento desarrollado por Ames Research Center (AMC) en 1988 para evaluar cargas físicas y cognitivas. Dada la amplitud de tópicos abarcados en el instrumento original, se aplicó una versión simplificada del instrumento que acota los tópicos a Carga Mental (MD), Carga Física (PD), Carga Temporal (TD), Desempeño (P), Esfuerzo (E) y Frustración (F). Además, para poder ser aplicado a escolares de entre 10 y 12 años, se simplificó el lenguaje utilizado en las preguntas, agregando emojis a la escala likert para una mejor comprensión.
  - **Carga mental (MD):** *¿Te costó mucho entender la aplicación? ¿Cuánto tuviste que pensar cuando hacías el programa?*
  - **Carga física (PD):** *¿Te cansaste mucho al manejar la aplicación? ¿Cuánto esfuerzo físico requirió hacer el programa?*
  - **Carga temporal (TD):** *¿Sentiste que te quedaste con poco tiempo? ¿Qué tan apurado/a te sentiste al usar el programa?*
  - **Desempeño (P):** *¿Qué tan satisfecho/a te sientes con tu programa? ¿Cómo crees que fue tu desempeño al resolver el problema?*
  - **Esfuerzo (E):** *¿Cuánto tuviste que esforzarte para realizar el programa? ¿Fue muy difícil lograrlo?*
  - **Frustración (F):** *¿Qué tan cómodo/a y tranquilo/a te sentiste al hacer el programa?*

  - **Creación (Cr):** *¿Qué tan difícil fue crear nuevos bloques?*
  - **Conectar (C):** *¿Qué tan difícil fue conectar bloques?*
  - **Solución (S):** *¿Qué tan difícil fue llegar a la solución?*
- **Expectativa [15]**: Adaptado del instrumento implementado por Katelyn M. Cooper et al. para evaluar expectativa en estudiantes. Este consiste en una serie de preguntas orientadas a evaluar el desempeño y cómo se auto percibieron los participantes del estudio. Al igual que con el resto de instrumentos, fue necesario llevar a cabo una simplificación y adaptación de las preguntas para ser comprendidas por un niño o niña de 10 a 12 años. Además, para una mejor comprensión y dada su compatibilidad con el cuestionario, se integraron estas preguntas en el Smiley-o-meter ya mencionado.

Calibración de los instrumentos

Antes de cada sesión se hace una prueba en cada computador para verificar que todo funciona correctamente, especialmente los controles conectados por Bluetooth, dado que existe la posibilidad de se descalibren entre sesiones. Además, se verifica que los enlaces a cada formulario estén funcionando correctamente. Para asegurarse que todos los instrumentos son comprensibles por el público objetivo, es decir, niñas y niños de entre 10 y 12 años, se condujo una versión piloto del estudio con una muestra de dos niños. En base a sus observaciones se hicieron cambios en los formularios tanto en la forma como en la redacción de los mismos. A su vez, se realizó un pilotaje con expertos de dominio, en temas de educación y programación, para evaluar si las preguntas y los conceptos estaban correctamente aplicados.

Amenazas a la Validez

Existen limitaciones en los controles, puede que unos al estar un poco más gastados, estén más descalibrados que otros. Este es un problema recurrente con este tipo de controles y se le conoce como Drift [2]. Para evitar esto, se prueba cada uno de los controles con la aplicación para verificar que funcionen correctamente. Por otro lado, el costo de los mismos no permitió realizar el experimento con más estudiantes a la vez.

En referencia a las posibles amenazas a la validez del caso de estudio, se consideran:

Constructo: Con el fin de mitigar posibles amenazas al constructo del caso de estudio, se validan con expertos de dominio los instrumentos de medición, así como los protocolos operativos del taller en general y de los monitores. Además, se realizaron pilotajes del taller para evaluar posibles problemas operacionales o administrativos de la experiencia. Estos pilotajes estuvieron enfocados en evaluar si los instrumentos estaban bien calibrados pensando en el público objetivo, y verificar que los tiempos otorgados para realizar las actividades eran los adecuados. Esto permitió mitigar amenazas a la validez de constructo.

Interna: Para abordar posibles amenazas a la validez interna del experimento, durante la etapa de reclutamiento y distribución de los participantes, se procuró que la muestra de cada sesión estuviera lo más balanceada posible. Además, durante la realización de la actividad, se procuró adaptar los contenidos al público objetivo, poniendo especial énfasis en que sea comprendido por ellos. Mismo caso con los instrumentos, se adapta el lenguaje utilizado para

[2] https://en.wikipedia.org/wiki/Joy-Con#Joy-Con_drift_lawsuits - Visitado en Octubre de 2023

evitar problemas de comprensión por parte de los participantes, y que respondan correctamente las preguntas. Para evaluar que las adaptaciones al contenido y a los instrumentos sean efectivos, se realiza un pilotaje del experimento con escolares de entre 10 y 12 años donde se repasa y refina el material completo del taller. Otra amenaza a la validez interna del experimento tiene relación con los criterios de inclusión y exclusión al taller. En particular, la condición de no poseer experiencia previa programando no fue verificada más allá de un autorreporte por parte del participante en el formulario de inscripción.

Externa: En cuanto a posibles amenazas externas a la validez del experimento, los resultados obtenidos de este no pueden ser generalizados más allá del público objetivo, que son escolares de entre 10 y 12 años. Además, dado que el la actividad fue de carácter voluntaria, existe la posibilidad de que los participantes ya estuvieran interesados en programar, o con una mejor predisposición a hacerlo. Del mismo modo, dado que el llamado se hizo por las redes sociales de la universidad, es posible que los voluntarios estuvieran relacionados a esta, sesgando la muestra. Una forma de abordar esta amenaza sería replicar el experimento con una muestra más diversa de estudiantes, integrando la actividad en un ambiente escolar. De esta forma se podrían generalizar los resultados a escolares de entre 10 y 12 años en general.

Ecológica: La validez ecológica de este experimento se ve amenazada principalmente por la consistencia de las indumentarias utilizadas en cada sesión del taller. En particular, las salas proporcionadas por la Universidad, si bien de similares capacidades y funcionalidades, no poseían la misma distribución de pupitres y pizarrón. Si bien se procuró mantener consistencia en cuanto a la distribución de los estudiantes en la sala, así como el espacio disponible para moverse, los mismos pupitres no eran iguales. Dado que en el estudio se busca evaluar el valor percibido por los participantes de la herramienta, es posible que la diferencia de indumentarias utilizadas impacte en la percepción de las niñas y niños. Por otro lado, es posible que eventos externos realizados en la universidad a la vez que las sesiones experimentales impactaran en la percepción de los estudiantes.

5.2.5. Procedimiento de Recolección de Datos

Para evaluar el impacto del mecanismo de interacción en la percepción de los estudiantes fue necesario recolectar datos antes (para informar y dar consentimiento a participar en la actividad), durante (para evaluar el valor percibido de los participantes) y después (para verificar cómo ha evolucionado el valor percibido con el tiempo).

Pre intervención: Cada participante recibe los cuestionarios de Consentimiento, a ser rellenado por el padre y/o tutor legal del participante, y de Asentimiento, a ser rellenado por el estudiante. Además, se les entrega instrucciones operativas para el taller presencial, como fechas y horarios.

Durante la intervención: Durante el primer bloque los participantes completaron el cuestionario de entrada. Luego, durante cada espacio de exploración libre de la herramienta en los primeros dos bloque, los monitores se encargaron de orientar y registrar el comportamiento de los participantes, así como comentarios relevantes que hicieran. Durante el último bloque se toma el tiempo que toma a cada participante terminar la tarea indicada, así como registrar posible lenguaje no verbal destacable en el proceso. Al finalizar el tercer bloque del taller, se les pidió a los participantes responder el formulario de salida.

Post intervención: Es posible que los resultados del cuestionario de salida se vean sesgados dada la emoción del participante de haber acabado el taller, especialmente si se evalúa el valor percibido por el participante. Con la intención de mitigar el llamado *Novelty Effect* se contacta a los participantes una semana después de acabado el taller para completar un nuevo cuestionario análogo al de salida. De esta forma se puede comparar cómo ha evolucionado la percepción de los participantes respecto al taller y la herramienta.

Condiciones de observación y/o experimentación

Durante todo el proceso, los monitores tenían asignados un grupo específico de participantes a observar y apoyar, previamente acordados en una sesión de entrenamiento. Durante los espacios de exploración libre de la herramienta, los monitores debían dejar que los participantes explorasen la aplicación todo lo quisieran, acudiendo en su ayuda en caso de necesitarla. Para el caso del desafío durante el último bloque, se le indica a los monitores que pueden orientar a los participantes en la dirección correcta mas no indicarles la respuesta explícitamente.

Pilotaje del proceso

El proceso fue piloteado en tres ocasiones. En primer lugar, se pilotea el funcionamiento de la aplicación y las tareas a realizar por los usuarios con estudiantes de primer año de Ingeniería Plan Común de la Universidad de Chile. Este proceso fue análogo al tercer bloque del estudio y se realiza para detectar y solucionar a tiempo posibles errores técnicos de la plataforma que puedan impactar en el valor percibido por los participantes del estudio final.

Luego, se pilotean los cuestionarios con usuarios expertos para asegurar la comprensión del lenguaje y pertinencia de las preguntas realizadas. Además, se realiza un piloto del taller con dos individuos del público objetivo para asegurar que las tareas y cuestionarios son claros. Este piloto consistió en una versión más focalizada del experimento, realizando las mismas actividades que en el experimento final.

Verificabilidad del proceso

Los instrumentos y la metodología del estudio está respaldada por sus respectivas fuentes y validado con expertos en el área. Además, se reportan los cuestionarios en Anexo B y protocolos en Anexo A.

Replicabilidad del proceso

Los instrumentos de medición y protocolos seguidos por el investigador se encuentran en Anexos B y Anexo A respectivamente, permitiendo replicar el estudio a futuro.

Limitaciones del proceso

La principal limitación del proceso se relaciona con la cantidad de usuarios evaluados y el método de reclutamiento. Esto se debe, en parte, al costo de los controles utilizados, lo que requirió limitar la cantidad de estudiantes participantes por sesión. Esto derivó en tener que realizar múltiples sesiones para recolectar datos.

5.3. Thinking Aloud: Confirmación de Observaciones

Una vez finalizado el experimento descrito anteriormente, se observaron comportamientos y actitudes no verbales en los participantes que no fueron considerados durante la etapa de diseño, pudiendo no haberse capturado completamente con los instrumentos diseñados. Con el propósito de profundizar en estas observaciones, se llevaron a cabo cuatro sesiones adicionales del taller en formato individual e independientes entre sí, con nuevos participantes. En estas nuevas sesiones, se empleó la metodología Thinking Aloud [70], que implica solicitar a los participantes que expresen en voz alta sus pensamientos y decisiones en tiempo real mientras interactúan con la herramienta. Esta metodología proporciona una comprensión detallada del proceso cognitivo de los participantes, abordando aspectos que podrían no haberse capturado completamente a través de los cuestionarios diseñados.

En este estudio, se seleccionaron estudiantes de entre 10 y 12 años sin experiencia previa en programación. Es importante señalar que estos participantes difieren de aquellos involucrados en el experimento anterior, y, de manera similar, el reclutamiento se llevó a cabo mediante redes sociales. Esta actividad se llevó a cabo en el domicilio particular de cada participante en una fecha acordada con los padres y/o tutores legales del mismo. Dado que los participantes son menores de edad, fue necesario pedir su Asentimiento a participar en la actividad, así como el Consentimiento Informado de los padres y/o tutores legales. Toda la indumentaria necesaria para realizar la actividad fue proporcionada por el investigador, incluyendo un computador para que el participante desarrolle la actividad, un tablet para mostrar el material didáctico y, en caso de estar evaluando Gesture Coding, los controles Joy-Con. Dado que la actividad se llevó a cabo en el domicilio particular del participante, y con el fin de respetar la privacidad de cada participante y sus familias, se optó por no grabar audio o video. Todo registro fue hecho a mano por el investigador.

Cada sesión tuvo una duración aproximada de una hora y media, con el participante y el investigador presente. En algunas ocasiones, uno de los padres y/o tutores legales se quedaron a observar la actividad, sin intervenir en esta. La reducción en la duración de la actividad se debe a que, dado que la clase era de carácter focalizado, esta fluía de mejor manera. Un total de cuatro sesiones fueron llevadas a cabo, dos de las cuales utilizaron Scratch, y las otras dos Gesture Coding. Al comenzar cada sesión, el participante recibe instrucciones sobre el protocolo Thinking Aloud. En particular, se le indicó que debía verbalizar todo lo que pasara por su cabeza mientras usaba la herramienta, indicando la intención de uso y el resultado obtenido de cada acción. Un ejemplo de verbalización es: *"Quiero usar un bloque de movimiento, por lo que pulso el botón ZR para abrir el menú y, manteniendo pulsado el mismo botón, dibujo una L en pantalla, y luego selecciono el bloque que necesito"*. En caso de que el participante se quedara en silencio, se le consultaba qué estaba pensando o qué estaba intentando hacer.

La estructura de cada sesión fue una versión condensada del experimento descrito anteriormente, reutilizando los instrumentos y el material:

- **Bloque 1 - Secuencialidad y Variables** (40 minutos): Se comienza dando instrucciones iniciales al participante sobre el protocolo Thinking Aloud, para luego entregarle el formulario de entrada. Luego, se le enseña cómo usar la herramienta que le corresponda, ya sea Scratch o Gesture Coding, y se introducen los bloques de tipo movimiento y

variables. Finalmente, se le deja unos 10 minutos de exploración libre de la herramienta siguiendo el protocolo Thinking Aloud usando los bloques enseñados.

- **Bloque 2 - Condicionales y Ciclos** (20 minutos): Se introducen los bloques Condicionales y Ciclos, para luego dejar libre exploración de la herramienta usando todos los bloques, nuevamente siguiendo el protocolo Thinking Aloud.
- **Bloque 3 - Desafío** (30 minutos): Se plantea el mismo desafío del experimento anterior y se les asigna 15 minutos para resolverlo. Durante la resolución del ejercicio nuevamente se les pide seguir el protocolo Thinking Aloud, esta vez enfocándose, además, en cómo llegaron a la solución del desafío. Una vez resuelto el ejercicio o acabado el tiempo, se les entrega el cuestionario de salida análogo al del experimento anterior, que mide cargas físicas y cognitivas, y evalúa el valor percibido por el participante. Finalmente, se les pregunta a modo de cierre opiniones generales sobre la herramienta que utilizaron.

5.4. Resultados

Los experimentos apuntaron a evaluar el valor percibido por los estudiantes al usar un mecanismo de interacción en particular, ya sea controles de movimiento con Gesture Coding o mouse y teclado con Scratch con una participación de más de 30 estudiantes en total. Sin embargo, por razones operativas derivadas de la pandemia de COVID-19, no se alcanzó el tamaño de muestra deseado. Luego, no es posible concluir resultados estadísticamente significativos. Dado esto, los resultados y su análisis será de manera descriptiva. Asimismo, en este experimento se observaron múltiples comportamientos de los estudiantes que no fueron considerados al momento de diseñar el experimento.

Los múltiples comportamientos de los estudiantes, que no fueron considerados al momento de diseñar la intervención anterior, motivaron un segundo proceso de levantamiento de datos. Este se llevó a cabo como una forma de profundizar en dichos comportamientos observados, replicando la experiencia con cuatro estudiantes en sesiones individuales siguiendo el protocolo Thinking Aloud. Este enfoque se alinea con un diseño de método mixto secuencial-explicativo, donde se recopilan datos cuantitativos y cualitativos en dos etapas separadas. La primera etapa se enfocó en evaluar cargas físicas y cognitivas, mientras que la segunda se enfocó en explorar la percepción de los estudiantes. De esta manera, se busca confirmar o refutar si los fenómenos observados en el experimento anterior pueden generalizarse o si fueron un caso particular.

5.4.1. Pruebas A/B

En este experimento, se evaluó el valor percibido y las cargas físicas y cognitivas de los estudiantes al interactuar con Gesture Coding y Scratch. La muestra incluyó la participación de 36 niños en total, de los cuales 25 utilizaron Gesture Coding y 11 utilizaron Scratch, todos con edades comprendidas entre 10 y 12 años. Del total de participantes, el 88 % tenía experiencia previa en el uso de dispositivos como computadoras, tablets y consolas de videojuegos.

Cuestionario de Entrada

Este cuestionario estaba orientado a evaluar las percepciones de los estudiantes antes de realizar el taller. Las preguntas realizadas midieron la Utilidad, Sesgo de Género, Confianza y Valor percibidos por el participante. Este instrumento se encuentra en Anexo B.

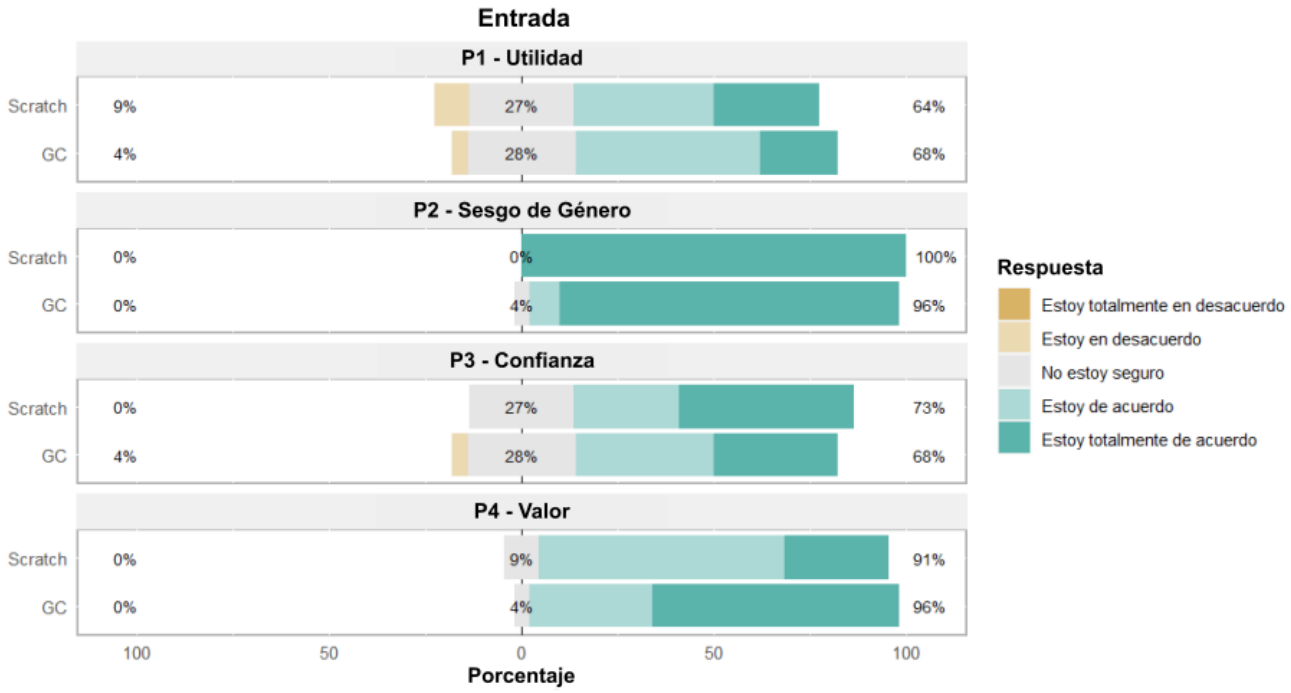


Figura 5.3: Puntuaciones en cada dimensión de la sección Smiley-o-meter del cuestionario de Entrada.

Tal como se observa en la figura 5.3, la percepción de los participantes de todos los campos fue similar, sin diferencia significativa entre ambos grupos. En cuanto a la utilidad percibida de la programación, la mayoría de los estudiantes expresó acuerdo o total acuerdo, con un 64 % en Scratch y un 68 % en Gesture Coding. Sin embargo, un porcentaje significativo reportó sentirse inseguro al respecto, con un 27 % con Scratch y 28 % con Gesture Coding.

En cuanto al sesgo de género, se observó que el 100 % de los participantes en Scratch y el 96 % en Gesture Coding reportaron bajos niveles de sesgo de género. Respecto a la confianza, el 73 % de los estudiantes en Scratch y el 68 % en Gesture Coding reportaron niveles altos de confianza en sus habilidades computacionales, mientras que un 27 % en Scratch y un 28 % en Gesture Coding indicaron sentirse inseguros al respecto. Además, en relación al valor percibido de la programación, el 91 % de los participantes en Scratch y el 96 % en Gesture Coding manifestaron percibir un alto valor en la habilidad de programar.

Cuestionario de Salida

Una vez acabado el taller, los participantes respondieron una versión extendida del cuestionario, agregando los campos de Engagement y Satisfacción a los antes mencionados, tal

como se observa en la Figura 5.4 y replicado en el Anexo B. Además, se agregan las preguntas derivadas del cuestionario NASA-TLX, para medir cargas físicas y cognitivas, cuyos resultados se observan en la Figura 5.5.

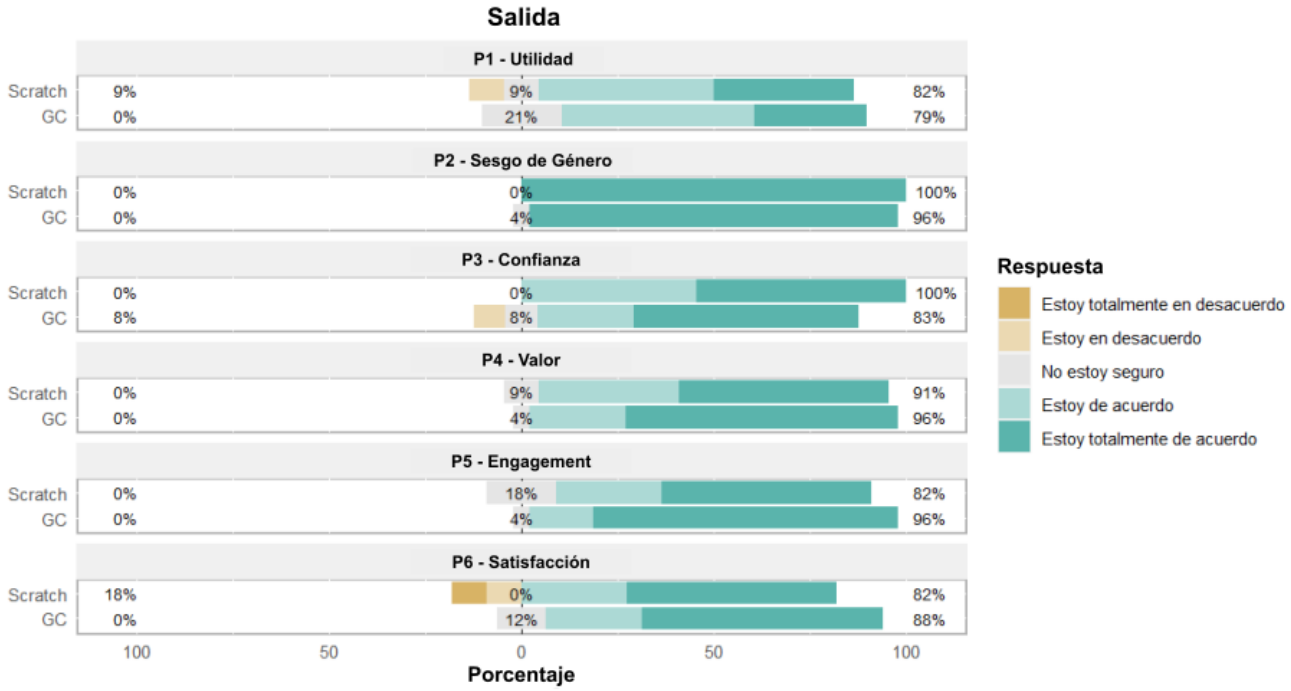


Figura 5.4: Puntuaciones en cada dimensión de la sección Smiley-o-meter del cuestionario de Salida.

En cuanto a la utilidad percibida, el 82 % de los estudiantes en Scratch y el 79 % en Gesture Coding expresaron estar de acuerdo o totalmente de acuerdo. No obstante, a pesar de observar una disminución en el porcentaje de participantes que reportaron sentirse inseguros, se observó que este porcentaje fue mayor en Gesture Coding, alcanzando un 21 %, en comparación con el 9 % en Scratch.

En relación al sesgo de género, los resultados se mantuvieron consistentes, con el 100 % de los participantes en Scratch y el 96 % en Gesture Coding mostrando bajos niveles de sesgo de género. En cuanto a la confianza en sus habilidades computacionales, el 100 % de los estudiantes en Scratch y el 83 % en Gesture Coding indicaron niveles altos de confianza, siendo un 8 % de los participantes en Gesture Coding quienes reportaron no sentirse seguros o con bajos niveles de confianza.

En lo que respecta al valor percibido de la programación, se observa un aumento en ambos grupos respecto al cuestionario anterior, donde el 91 % de los participantes en Scratch y el 96 % en Gesture Coding manifestaron percibir un alto valor. En términos de engagement, el 82 % de los estudiantes en Scratch y el 96 % en Gesture Coding expresaron alto interés en programación. Sin embargo, se observó un mayor nivel de incertidumbre en Scratch, con un 18 % reportando sentirse inseguro, en comparación con Gesture Coding, donde solo el 4 % indicó lo mismo.

Finalmente, en cuanto a la satisfacción percibida, el 82 % de los participantes en Scratch y el 88 % en Gesture Coding manifestaron estar satisfechos con su desempeño al resolver el desafío. No obstante, el 12 % de los estudiantes en Gesture Coding expresaron sentirse inseguros, mientras que el 18 % se mostró insatisfecho en Scratch.

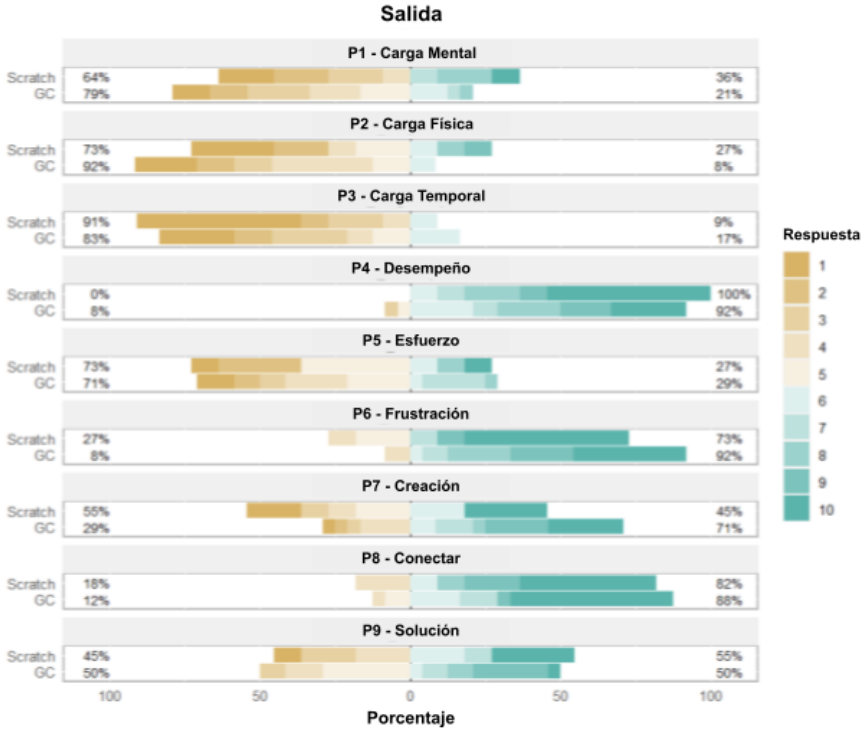


Figura 5.5: Puntuaciones en cada dimensión de la sección Nasa-TLX del cuestionario de Salida.

En cuanto a las cargas físicas y cognitivas reportadas por los estudiantes se observan distintas tendencias dependiendo de la herramienta, con poca diferencia significativa:

- **Carga Mental:** En Scratch, el 64 % de los participantes indicaron una carga mental baja, mientras que en Gesture Coding, este porcentaje aumentó a un 79 %. La tendencia es hacia una menor carga mental en Gesture Coding.
- **Carga Física:** La carga física fue percibida como baja por el 73 % en Scratch y un 92 % en Gesture Coding. Aquí, la tendencia es hacia una menor carga física en Gesture Coding.
- **Carga Temporal:** El 91 % de los participantes en Scratch percibieron una carga temporal baja, mientras que en Gesture Coding fue el 83 %. La tendencia es a una carga temporal baja en ambas herramientas, siendo ligeramente menor en Scratch.
- **Desempeño:** En Scratch, el 100 % de los participantes reportaron un desempeño alto, mientras que en Gesture Coding fue el 92 %. La tendencia es hacia un alto desempeño en ambas herramientas, siendo ligeramente menor en Gesture Coding.
- **Esfuerzo:** El 73 % de los participantes en Scratch indicaron un bajo esfuerzo, en comparación con el 71 % en Gesture Coding. La tendencia es hacia un bajo esfuerzo en ambas herramientas, siendo ligeramente mayor en Gesture Coding.

- **Frustración:** En Scratch, el 73 % de los participantes indicaron sentir baja frustración, mientras que en Gesture Coding, este porcentaje aumentó a 92 %. La tendencia es hacia una menor frustración en Gesture Coding.
- **Creación:** En Scratch, el 45 % indicó una baja dificultad al momento de crear bloques, mientras que en Gesture Coding, este porcentaje aumentó a 71 %. La tendencia es hacia una menor dificultad en la creación de bloques con Gesture Coding.
- **Conectar:** En Scratch, el 82 % indicó una baja dificultad al momento de conectar bloques, mientras que en Gesture Coding, el 88 % reportó lo mismo. La tendencia es hacia una baja dificultad en ambas herramientas, siendo ligeramente menor en Gesture Coding.
- **Solución:** En Scratch, el 45 % indicó una baja dificultad de la solución, mientras que en Gesture Coding el resultado fue de un 50 %. La tendencia es a una dificultad media.

En general, se observan resultados bastante similares en ambas herramientas, con diferencias poco significativas. En particular, la similitud de los resultados indicaría que, respecto a las cargas percibidas, no existe diferencia significativa para el usuario usar una herramienta u otra. Sin embargo, el resultado particular de la Carga Física resulta contra intuitivo, dado que la herramienta donde al usuario se le pide realizar movimientos fue reportada con menor carga. Este resultado se discute en profundidad en el Capítulo 6.

Cuestionario de Extensión

Finalmente, se contacta a los participantes una semana después de haber realizado la actividad para responder nuevamente el cuestionario anterior y evaluar cuánto ha cambiado su percepción respecto a la experiencia.

Como se puede observar en la Figura 5.6, en términos de utilidad percibida, el 75 % de los estudiantes en Scratch y el 75 % en Gesture Coding expresaron estar de acuerdo o totalmente de acuerdo, con un 25 % de inseguridad en Gesture Coding y un 12 % en Scratch. Sin embargo, este último grupo reportó un 12 % de baja utilidad percibida. En relación al sesgo de género, los resultados se mantuvieron consistentes, con el 100 % de los participantes en ambas plataformas mostrando bajos niveles de sesgo de género.

Respecto a la confianza en sus habilidades computacionales, el 88 % de los estudiantes en Scratch y el 85 % en Gesture Coding indicaron niveles altos de confianza. Sin embargo, se observó un mayor nivel de inseguridad en Gesture Coding, con un 15 %, en comparación con el 12 % en Scratch. En cuanto al valor percibido de la programación, se registró un 88 % en Scratch y un 85 % en Gesture Coding, con niveles de inseguridad del 12 % y 15 %, respectivamente. En términos de engagement, el 100 % de los estudiantes en Scratch y el 95 % en Gesture Coding expresaron alto interés en programación. Finalmente, en cuanto a la satisfacción percibida, el 88 % de los participantes en Scratch y el 80 % en Gesture Coding manifestaron estar satisfechos con su desempeño al resolver el desafío. Se observó un aumento en la inseguridad en Gesture Coding, con un 20 %, en comparación con el 12 % en Scratch.

En cuanto a las medidas de cargas físicas y cognitivas reportadas por los estudiantes, se observa que:

- **Carga Mental:** En Scratch, el 62 % de los participantes indicaron una carga mental

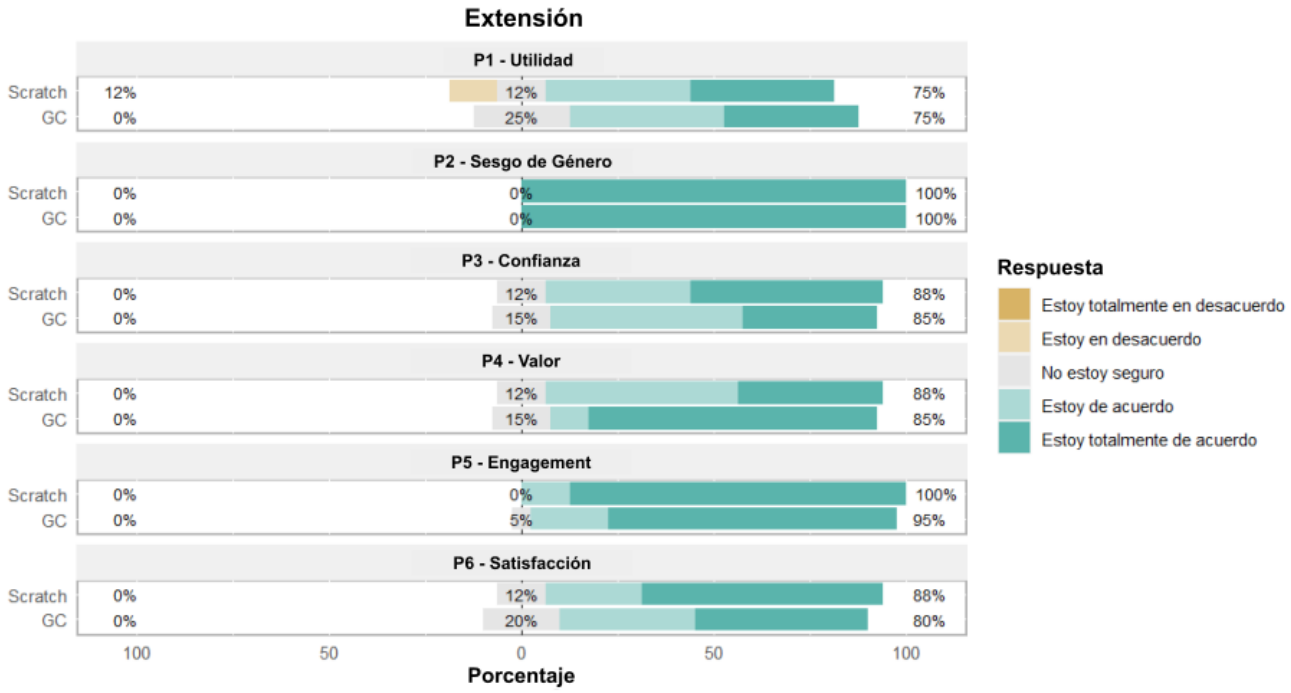


Figura 5.6: Puntuaciones en cada dimensión de la sección Smiley-o-meter del cuestionario de Extensión.

baja, mientras que en Gesture Coding, este porcentaje aumentó a un 85 %. La tendencia es hacia una menor carga mental en Gesture Coding.

- **Carga Física:** La carga física fue percibida como baja por el 100 % en Scratch y el 100 % en Gesture Coding. La tendencia es hacia una carga física baja en ambas herramientas.
- **Carga Temporal:** El 88 % de los participantes en Scratch percibieron una carga temporal baja, mientras que en Gesture Coding fue el 95 %. La tendencia es hacia una carga temporal baja en ambas herramientas, siendo ligeramente menor en Gesture Coding.
- **Desempeño:** En Scratch, el 100 % de los participantes indicaron un desempeño alto, mientras que en Gesture Coding, el 85 % reportó lo mismo. La tendencia es hacia un alto desempeño en ambas herramientas, siendo ligeramente menor en Gesture Coding.
- **Esfuerzo:** El 62 % de los participantes en Scratch indicaron un bajo esfuerzo, en comparación con el 90 % en Gesture Coding. La tendencia es hacia un bajo esfuerzo en ambas herramientas, siendo menor en Gesture Coding.
- **Frustración:** En Scratch, el 75 % de los participantes indicaron sentir baja frustración, mientras que en Gesture Coding, este porcentaje aumentó al 95 %. La tendencia es hacia una menor frustración en Gesture Coding.
- **Creación:** En Scratch, el 50 % indicó una baja dificultad al momento de crear bloques, mientras que en Gesture Coding, este porcentaje disminuyó al 25 %. La tendencia es hacia un menor nivel de dificultad en Gesture Coding.
- **Conectar:** En Scratch, el 88 % indicó una baja dificultad al momento de conectar

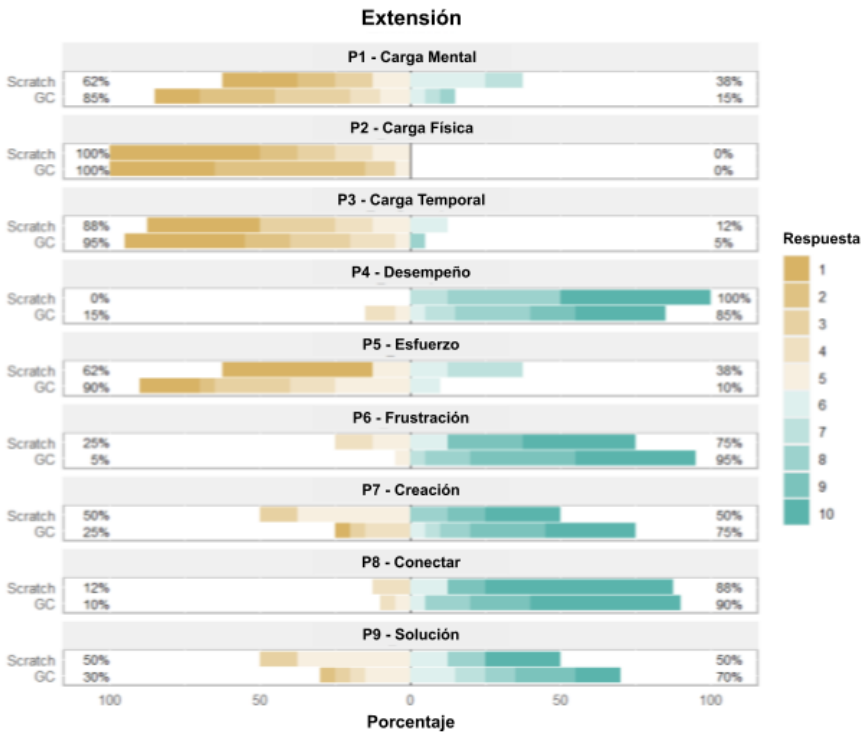


Figura 5.7: Puntuaciones en cada dimensión de la sección Nasa-TLX del cuestionario de Extensión.

bloques, mientras que en Gesture Coding fue el 90 %. La tendencia es hacia una baja dificultad en ambas herramientas, siendo ligeramente menor en Gesture Coding.

- **Solución:** En Scratch, el 50 % indicó una baja dificultad de la solución, mientras que en Gesture Coding aumenta al 70 % de los participantes. La tendencia es a una menor dificultad percibida en Gesture Coding.

En términos generales, los resultados presentan consistencia en las tendencias observadas, intensificándose en áreas específicas. No obstante, es importante destacar que las diferencias en el desempeño entre las dos herramientas aún se mantienen como poco significativas. En particular, luego de transcurrida una semana, se observa una leve tendencia a una menor carga percibida por los estudiantes con Gesture Coding. Además, las respuestas en general aumentaron su valoración, indicando que con el tiempo el estudiante se sintió más cómodo con la experiencia. Se discute más sobre este fenómeno en el Capítulo 6.

Comentarios generales

Además de los cuestionarios anteriores, los estudiantes también dieron su opinión general de la plataforma que les tocó trabajar. En el caso de Gesture Coding, la mayoría de los participantes destacó la facilidad de uso y la diversión que experimentaron al interactuar con el sistema. Algunos destacaron particularmente la satisfacción que sintieron al lograr implementar el desafío planteado y programas propuestos por ellos mismos: *"Me gustó completar el desafío y que el personaje se teletransporte"* (P3, H, 10 años); *"Estuvo difícil pero logré hacer que el monito se moviera por la pantalla"* (P5, M, 12 años). Aunque se señalaron problemas

técnicos o errores en la aplicación, estos parecieron no afectar significativamente la percepción general de la herramienta: *"Estuvo entretenido aunque a veces me costaba conectar bien los bloques"* (P2, H, 11 años); *"Me costó acostumbrarme a la sensibilidad de la palanca."* (P8, M, 11 años).

En cuanto al proceso de aprendizaje, señalaron que Gesture Coding inicialmente fue más complejo usar el sistema, pero que con el tiempo se fueron adaptando este, permitiendo que implementaran el código que ellos quisieran. Esto se condice con los resultados de los cuestionarios de NASA-TLX, donde la mayoría reporta una baja Carga Mental. En conjunto, los comentarios sugieren una respuesta positiva y favorable hacia Gesture Coding, destacando su impacto educativo y la satisfacción general de los estudiantes al utilizar la aplicación: *"Me gustó aprender cosas nuevas con la aplicación"* (P10, M, 11 años); *"Es una buena aplicación y fácil de usar, no me costó tanto usarla y entenderla como pensé"* (P13, H, 10 años). Al consultar a los participantes una semana después, los comentarios mantuvieron la misma tendencia positiva, haciendo más énfasis en los aspectos positivos de la experiencia por sobre posibles problemas o errores técnicos: *"Estuvo algo difícil pero muy entretenido, sólo no me gustó el bug que no me dejaba hacer nada, pero muy entretenido"* (P17, H, 10 años); *"El programa está muy bien hecho, sólo le encontré algunos fallos, me divertí mucho"* (P14, M, 10 años); *"Me sentí muy bien usando la aplicación, quiero volver a hacerlo"* (P16, M, 10 años).

En cuanto a Scratch, los comentarios muestran una diversidad de experiencias y opiniones sobre la aplicación. En general, los participantes expresaron que la experiencia con Scratch fue buena, pero con ciertos desafíos y dificultades para realizar ciertas tareas: *"Me gustó harto, pero costaba hacer algunas cosas"* (P3, H, 10 años); *"Es muy entretenida la aplicación pero no se cómo ocuparla tan bien"* (P7, M, 11 años). También se destaca la mención de aspectos lúdicos durante la clase, como pequeños desafíos espontáneos planteados por ellos mismos, que generaron una respuesta positiva *"Me gustó matar al gato"* (P4, H, 10 años); *"Me gustó aprender que el personaje camine fluido y moverlo con el ratón"* (P6, M, 10 años). En resumen, las opiniones son variadas, abarcando desde aspectos positivos hasta desafíos y experiencias mixtas. Esta percepción se mantiene al consultar a los participantes una semana después, destacando aspectos satisfactorios y entretenidos durante la ejecución de las tareas: *"Mi experiencia fue buena, pero noté un poco difícil usar la aplicación"* (P8, H, 11 años); *"Me gustó mucho porque aprendí a programar"* (P1, M, 11 años).

Observaciones de monitores

Finalmente, los monitores de cada sesión hicieron observaciones generales al comportamiento de cada estudiante, así como actitudes que podrían presentar durante los talleres que no necesariamente fueron reportadas por ellos mismos. Para esto, se acordó previamente qué aspectos tener en cuenta en estas observaciones, centrándose principalmente en el lenguaje no verbal de los participantes al momento de realizar las actividades, así como qué cosas les motivaba trabajar. Esto se consignó de manera estándar, presentada y acordada previamente con los monitores, en una plantilla para registrar los datos de manera controlada. Estas observaciones fueron realizadas al momento de atender a los participantes en caso de dudas o al revisar su desempeño durante la clase.

En las sesiones donde se utilizó Gesture Coding, los monitores comentaron que los parti-

cipantes se veían fuertemente frustrados durante el primer bloque, principalmente debido al uso de controles de movimiento: *"Noté a mis participantes complicados con los controles de movimiento durante la primera parte, con algunas caras de frustración"* (Monitor 2); *"En el segundo bloque vi a mis participantes más cómodos con el sistema, pidieron menos ayuda para crear bloques y se centraron en el código"* (Monitor 1). En primera instancia, los participantes se limitan a imitar lo visto en clases y recrear el código mostrado. Durante la sección libre del segundo bloque, notaban una actitud más cómoda y segura con el sistema. En particular, se les notaba más cómodos dibujando gestos y creando bloques. En este momento se observaron principalmente dos fenómenos simultáneos. En primer lugar, los estudiantes estaban enfocados en probar combinaciones de bloques para evaluar qué pasaba con el personaje, acudiendo a los monitores en caso de dudas: *"Los vi probando bloques y acumulándolos sin un objetivo claro, la mayoría de las preguntas eran para aclarar por qué ocurría un determinado suceso"* (Monitor 1); *"Me hicieron muchas preguntas enfocadas en qué se puede hacer, o qué pasa si conecto un bloque determinado con otro. La respuesta por lo general era indicarles que probaran"* (Monitor 3). Por otro lado, si algún estudiante lograba implementar algún tipo de programa, se generaba un foco de atención en el que sus compañeros se acercaban a ver y replicar el código: *"Durante una de las sesiones, un participante logró hacer que el personaje se moviera en círculos, y muchos otros participantes se acercaron a ver cómo lo hizo. Luego de un rato, todos tenían una variación de ese código, con distintas velocidades o trayectorias para el personaje"* (Monitor 2). A esta altura se les nota cómodos con el sistema y exploraban de manera libre el sistema, muchos estudiantes afirmaban querer explorar los límites de la plataforma: *"Vi participantes que expresaban querer encontrar el límite de la aplicación, acumulando muchos bloques para ver qué ocurría. Usaban mucho los bloques de repetición y de movimiento para esto."* (Monitor 3). Se observa un ambiente lúdico. Por otro lado, a pesar de poder realizar toda acción con los controles, los participantes recurrían ocasionalmente al uso del Mouse de manera inconsciente.

Finalmente, durante el último bloque, hubo un grupo de estudiantes que se enfocaron en resolver el desafío, mientras que otro grupo seguía en esta actitud lúdica probando combinaciones de bloques: *"A algunos les costó resolver el desafío, unos porque no entendían del todo qué hacer, y otros porque no sabían cómo hacerlo. Luego de dar un par de pistas lograban resolverlo"* (Monitor 1); *"A pesar de insistirles en que resolvieran el desafío, hubo participantes que prefirieron seguir extendiendo su código de la segunda parte de la actividad"* (Monitor 2). En general, se observa un ambiente lúdico durante la sesión, especialmente desde el segundo bloque.

En cuanto a la sesión donde se utilizó Scratch, los monitores reportaron un ambiente menos lúdico que en las otras sesiones: *"Con Scratch en particular, los participantes se notaban menos motivados a trabajar en la actividad y usar bloques. Algunos incluso usando el navegador para jugar"* (Monitor 3). En una primera instancia, la cantidad de bloques de la plataforma confundió a los participantes, quienes expresaron no saber dónde buscar los bloques requeridos. Una vez aprendían a utilizar un tipo específico de bloque, en pocos casos se observa un interés por explorar y probar otros tipos de bloques. Los participantes se veían más interesados en la parte gráfica de Scratch, especialmente en las funcionalidades de dibujo y sonido *"Si bien los participantes usaban Scratch, no se les veía con especial interés en los bloques o en la plataforma en general. Usaban mucho la pestaña de disfraces y sonidos."* (Monitor 1); *"Una vez un participante comenzó a jugar y reproducir sonidos muy fuertes con*

su computador, el resto le seguía y se perdía el foco en los bloques. Fue necesario darles la indicación de que cerraran la pestaña de sonidos. Esto provocó que el ambiente se tensara, dado que comentaban que los estábamos retando" (Monitor 2). En cuanto a la implementación de código, los participantes se limitaban a replicar lo visto en la parte expositiva de la clase y expandirlo. Esto significa que, durante las secciones de exploración libre, no presentaban interés en probar nuevos bloques que no fueran los vistos en clases: *"La mayoría de las dudas que recibí por parte de los participantes eran en torno a cuáles bloques se usaban y cuáles no. Por ejemplo, si querían mover al personaje me preguntaban qué bloque lo movía, en lugar de probar e intentar por su cuenta"* (Monitor 1).

Durante la última parte del taller, los participantes se vieron fuertemente frustrados con la resolución del desafío planteado. En particular, expresaron no entender cómo resolver el problema o qué bloques necesitar: *"Durante la última parte de la actividad, recibí muchas dudas sobre qué debían hacer porque no entendían el desafío."* (Monitor 3); *"Los participantes que intentaban resolver el problema me hicieron muchas consultas sobre los bloques, expresando que tenían una solución en mente pero no sabían implementarla"* (Monitor 2). A pesar de esto, y luego de entregar ayuda de parte de los monitores, lograron superar el desafío. Al igual que las sesiones con Gesture Coding, se observa un grupo particular que no se interesó en resolver el desafío, sino que en seguir explorando los bloques o las funcionalidades gráficas de Scratch *"Casi la mitad de los participantes a mi cargo no se veían interesados en resolver el desafío, y fue necesario llamarles la atención en más de una ocasión para que intentaran resolverlo"* (Monitor 3); *"Recibí preguntas de algunos participantes en torno a cómo cambiarle la velocidad o entonación a un sonido, o cómo animar al personaje cuando debieron estar resolviendo el problema planteado. Al mencionarles esto, comentan que no lo entienden del todo o que no saben cómo hacerlo"* (Monitor 1).

En general, la principal diferencia entre la actividad con Scratch y Gesture Coding es: (1) El ambiente, si bien lúdico en ambos, se notaba más distendido con Gesture Coding; (2) Durante la fase exploratoria de Scratch el interés estuvo en sus funcionalidades gráficas, mientras que en Gesture Coding fue la creación y combinación de bloques; y (3) El trabajo fue notablemente más cooperativo durante las sesiones de Gesture Coding que en Scratch, donde fue más personal.

5.4.2. Thinking Aloud

Una vez concluido el estudio anterior, y con la intención de ahondar en los comentarios hechos por los participantes, se realizaron cuatro sesiones adicionales. Cada sesión fue individual y se aplicó el mismo procedimiento de las pruebas A/B, usando Gesture Coding o Scratch. Además, se le pidió al participante verbalizar todo pensamiento mientras realizara la actividad. Este proceso fue facilitado por el autor de este trabajo de tesis. A continuación se presentan los resultados de estas intervenciones, clasificadas por la herramienta utilizada en cada sesión. Los nombres presentados en este trabajo son ficticios para proteger la privacidad de los participantes.

Gesture Coding

Al comenzar la actividad, a los participantes se les pidió llenar un formulario inicial con sus datos y su percepción respeto a la programación en general. Como se observa en la Tabla

Tabla 5.1: Caracterización de participantes en sesiones de Thinking Aloud con Gesture Coding.

Nombre	**Edad**	**Experiencia previa**	**Mano dominante**	**Experiencia con controles de movimiento**	**Tipo de Experiencia**	**Poseen Nintendo Switch**
Alberto	10	No	Diestro	Sí	Nintendo Switch	No
Bruno	10	No	Zurdo	Sí	Nintendo Switch	Sí

5.1, ambos participantes tenían 10 años al momento de realizar la actividad y carecían de experiencia previa en programación. Sin embargo, ambos reportaron tener experiencia con controles de movimiento, particularmente con Joy-Cons de Nintendo Switch. En el mismo cuestionario, también reportaron sus percepciones iniciales respecto a la computación y sesgos de género. Como se ve en la tabla 5.2, ambos participantes presentaron un bajo sesgo de género y alta confianza. Por otro lado, Alberto percibió la computación como más útil que Bruno, quien se mostró inseguro.

Durante la parte expositiva del primer bloque, ambos participantes se observaron atentos e interesados. Durante la parte exploratoria, se notaban entusiasmados con la experiencia de trabajar con controles de movimiento, al preguntarles el motivo ambos argumentaron que es porque lo perciben como algo divertido. El enfoque dado a la parte exploratoria fue distinta en ambos casos. Alberto, por un lado, se enfocó en hacer que el personaje se moviera de distintas maneras, acumulando bloques para lograr su objetivo. En un inicio presentó dificultades para dibujar los gestos, sin embargo menciona que lo percibe como un desafío y que debe acostumbrarse. Bruno, por el otro lado, se enfocó en los bloques, probando cada gesto disponible incluso cuando no se habían introducido todos. Al igual que Alberto, presentó dificultades al inicio para dibujar los gestos, pero se le veia más frustrado que Alberto. Al consultarle, Bruno menciona: *"La palanca está muy sensible y me cuesta moverla un poco, me gustaría bajarle a la sensibilidad"*.

Tabla 5.2: Percepción de participantes en sesiones de Thinking Aloud con Gesture Coding.

Participante	**Utilidad**	**Sesgo de género**	**Confianza**	**Valor percibido**
Alberto	De acuerdo	Totalmente de acuerdo	De acuerdo	Totalmente de acuerdo
Bruno	No estoy seguro	De acuerdo	De acuerdo	Totalmente de acuerdo

En la segunda sección de la actividad, durante la parte expositiva, Bruno se limitó a escuchar y responder las preguntas que se le hacían. En particular, al presentarle el bloque de repetición menciona: *"Ah sí, ese lo usé antes para mover al personaje, "* indicando que, como ya había usado y probado los bloques con anterioridad, entendía los conceptos presentados. Alberto, por otro lado, al presentarle los nuevos bloques de ciclos y condicionales, menciona

que estos bloques le habrían servido para mover al personaje de la manera que necesitaba: *"Entonces con el bloque de repetición puedo ahorrarme estar creando nuevos bloques de movimiento a cada rato"*. Durante la parte exploratoria de este bloque, los participantes aplicaron los conceptos nuevos aprendidos, nuevamente, con enfoques distintos. Por un lado, Alberto se dedicó a rehacer el código que tenía usando los nuevos bloques, en particular el de repeticiones. En estos momentos ya no le es complejo usar los gestos, comenta que se acostumbró a hacerlos. Por el otro lado, Bruno crea un bloque de cada uno sin un objetivo claro, encajando bloques para ver qué efecto tenía. Luego de un rato de apilar bloques, Bruno pregunta qué aplicación real pueden tener estos conceptos en la vida real, especialmente ciclos y repeticiones. Se le entregan ejemplos concretos e intenta aplicarlos en su código. Ya no se le observa frustrado al momento de crear bloques, al consultarle comenta: *"Al principio me costó usar la palanca porque estaba muy sensible, pero me acostumbré ya. Igual creo que debería bajarle la sensibilidad para que no sea tan difícil al inicio"*.

Finalmente, en el tercer bloque se les entrega un desafío de programación que integra todos los tipos de bloque vistos. Al igual que en el experimento anterior, esta consistió en lograr que el personaje se mueva a la derecha hasta tocar la pantalla, momento en el cuál debe aparecer por el costado izquierdo. Alberto intenta resolver el desafío siguiendo las pistas entregadas. En primer lugar, implementa el movimiento horizontal del personaje y se queda pensando en cómo resolver la parte de reposicionar al personaje cuando se llegue al borde. Al preguntarle qué pensaba, dice estar seguro de tener que usar condicionales pero no se le ocurre cómo integrar el bloque. Luego de un par de intentos, decide usar el bloque condicional doble para superar el desafío. Luego, se dedica a realizar variaciones del mismo, moviendo el personaje de manera vertical o en las esquinas. Mientras prueba trayectorias para el personaje, Alberto comenta: *"Es entretenido mover al personaje por la pantalla, pero siento que me gustaría poder moverlo de manera curva o hacerlo rebotar. ¿Podrían haber más bloques?"*, indicando que siente el sistema algo limitado.

Por otro lado, Bruno intenta resolverlo implementa el movimiento del personaje usando sólo repeticiones, contando a mano cuántas iteraciones se deben realizar antes de posicionar al personaje al lado izquierdo. Al preguntarle por el motivo de que no usó condicionales, comenta que es porque no entiende del todo como usarlos. Luego de una breve explicación, corrige su código aplicando correctamente el bloque condicional, resolviendo el desafío. Luego, se dedica a probar los límites del sistema, conectando un número elevado de bloques de manera secuencial y dentro de una repetición. Al preguntarle por el motivo de estas acciones, comenta que le da curiosidad saber qué pasa.

Finalmente, ambos participantes responden un formulario final para medir las cargas físicas y cognitivas ejercidas en la actividad: Carga Mental (MD), Carga Física (PD), Carga Temporal (TD), Desempeño (P), Esfuerzo (E) y Frustración (F). Además, se les consulta qué tan difícil fue crear bloques (Cr), conectarlos (C) y llegar a la solución (S). Este correspondía a una escala Likert de 10 puntos, donde 0 era el más bajo y 10 el más alto, salvo en Frustración donde es al revés.

En la tabla 5.3 se observa que, en general, Alberto reportó experimentar una baja carga cognitiva y física en el experimento. En particular, la Carga Mental y la Frustración fueron los valores más altos. Además, reportó tener un buen desempeño y estar satisfecho con la solución

Tabla 5.3: Respuestas a formulario de cargas físicas y cognitivas de participantes en sesiones de Thinking Aloud con Gesture Coding.

Nombre	**MD**	**PD**	**TD**	**P**	**E**	**F**	**Cr**	**C**	**S**
Alberto	3	1	1	10	2	8	10	9	10
Bruno	5	4	3	9	5	8	4	4	4

a la que llegó. Finalmente, menciona que Crear y Conectar bloques le resultó Fácil. Por otro lado, Bruno reportó que sus cargas físicas y cognitivas fueron más altas. En particular, tanto la Carga Mental y Física como el Esfuerzo tuvieron puntajes cercanos al centro. Del mismo modo, menciona que el esfuerzo para llegar a la solución fue medio, dejándolo medianamente insatisfecho. Asimismo, menciona que Crear y Conectar bloques fue medianamente difícil. De estos resultados se puede interpretar que para Alberto fue una experiencia desafiante pero quedó satisfecho con su desempeño, mientras que para Bruno fue más difícil tanto el uso de la plataforma como el proceso de programación como tal.

Tabla 5.4: Respuestas a formulario Smiley-o-meter de participantes en sesiones de Thinking Aloud con Gesture Coding.

Nombre	**Utilidad**	**Sesgo de Género**	**Confianza**	**Valor Percibido**	**Engagement**	**Satisfacción**
Alberto	No estoy seguro	Totalmente de acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	Totalmente de acuerdo
Bruno	No estoy seguro	De acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	Totalmente de acuerdo

Finalmente, en cuanto a la percepción de los participantes respecto a la herramienta, se observan resultados en la misma línea del cuestionario de entrada. En particular, tanto Alberto como Bruno reportaron un mayor nivel de Confianza, mientras que Alberto disminuyó su percepción de Utilidad de la programación. Respecto al Engagement y la Satisfacción respecto a su desempeño en la actividad, ambos declararon estar totalmente satisfechos. De esto se puede interpretar que, a pesar de haber reportado experiencias distintas tanto en la verbalización como en los cuestionarios de cargas, ambos percibieron la actividad como Satisfactoria y Entretenida.

Finalmente, se les consultó por opiniones generales respecto a la actividad y la herramienta que usaron. Alberto destacó lo divertido que es usar la aplicación, expresando que sintió un ambiente lúdico durante la actividad. Se le consulta sobre el uso de controles de movimiento, y menciona que le agrada usarlos aunque le pareció difícil en un inicio. Indica, además, que sintió satisfacción al momento de resolver el desafío. En particular comenta que: *"Me gustó el juego, aunque me costó usarlo bien al principio"*; *"Fue entretenido mover al personaje como quería, y me gustó hacer que el personaje se teletransportara"*. Finalmente, menciona que esta herramienta es algo que usaría en sus ratos libres para experimentar o implementar algún juego: *"Me gustaría que tuviera más bloques para hacer un juego más grande después del colegio, podría cambiarle el diseño también y crear niveles"*, indicando que le gustaría tener más bloques y opciones de personalización del personaje.

Por otro lado, Bruno destacó que resultó divertido usar la herramienta, en particular el uso

de controles de movimiento. Menciona que los gestos le ayudaban a recordar dónde estaba cada bloque. Se le consulta por el desafío y su solución, y menciona que sintió satisfacción al resolverlo, pero fue difícil y algo frustrante: *"Me divertí jugando con los bloques y viendo qué efecto tenían en el personaje."*; *"Al principio me costó entender un poco lo de dibujar en la pantalla, pero después creo que me ayudaba a recordar dónde estaban los bloques que necesitaba"*. Lo que más disfrutó de la actividad fue la posibilidad de explorar y probar el sistema para probar sus límites. En particular, menciona: *"Me gustó ver qué ocurría con el personaje cuando lo hacía volar por la pantalla cada vez más rápido, podría haber un modo cooperativo para que cada uno mueva a su personaje"*, indicando que le gustaría poder compartir su código con amigos, de una forma cooperativa.

Scratch

Una vez comenzaba la actividad, se le pidió al participante completar un formulario inicial con sus datos personales y su percepción respecto a la programación y sus aplicaciones. Como se observa en la Tabla 5.5, ambos participantes tenían 12 años y no presentaban experiencia previa programando. Además, declaran tener experiencia usando un computador y/o tablet para entretención y estudios. Por otro lado, como se observa en la Tabla 5.6, ambos participantes perciben la computación como algo útil y un alto valor. Sin embargo, en cuanto a la confianza al momento de usar tecnologías, Diego declaró sentirse inseguro de ello, mientras que Carlos indica que se siente seguro.

Tabla 5.5: Caracterización de participantes en sesiones de Thinking Aloud con Scratch.

Nombre	**Edad**	**Experiencia Previa**	**¿Para qué usas un computador o tablet?**
Carlos	12	no	tablet para jugar
Diego	12	nada	tengo las tres las uso para jugar y tareas

Una vez iniciada la parte expositiva del taller, ambos participantes se notaban atentos e interesados. En particular, Carlos se interesó en aplicaciones prácticas de los conceptos vistos en la clase haciendo preguntas al respecto. Al pasar a la parte exploratoria de la actividad, ambos participantes se dedicaron a replicar lo visto en clases, experimentando con esos bloques. Ningún participante tuvo la iniciativa de explorar más bloques de la misma categoría. Diego, por un lado, se interesó por mover el gato y rotarlo por el escenario. Al consultarle por lo que hacía, comenta: *"Es como Mario Maker donde puedo mover objetos por el escenario como yo quiera, pero puedo dejarlos programados"*, indicando que le recuerda a un videojuego y eso le llamó la atención. Carlos, por otro lado, comenzó a explorar las pestañas de sonido y disfraces. Además, menciona la intención de su proyecto, que es un concierto de personajes. Al recordarle el uso de bloques, vuelve a ellos pero usando sólo un tipo de bloque, moviendo al personaje que solía ser el gato y comenta: *"Aquí puedo hacer que él personaje cante y se mueva por el escenario, pero primero quiero terminar de decorar todo"*.

En la fase expositiva del segundo bloque, Carlos se veía distraído con la pestaña de disfraces de Scratch, mientras que Diego preguntaba cómo se aplicaba esto en videojuegos. Durante la parte exploratoria de esta actividad, los participantes se animaron a explorar los nuevos bloques, pero haciendo notar que no estaban seguros de cuál bloque necesitar. Carlos, en

particular, indicó que se confunde con los bloques y no recuerda cuál era que estaba usando antes: *"¿Cuál era el bloque para mover al personaje? Siento que se parecen mucho, no estoy seguro cuál era"*. Por otro lado, Diego integró sólo los bloques vistos en clases, preguntando en cada paso qué hacía cada bloque, y si era efectivamente el bloque que necesitaba, comentando: *"¿Este bloque qué hace? (...) ¿Entonces este necesito para mover al personaje derecho? ¿O era este? Creo que este no lo tengo que usar"*. Carlos se observa frustrado con los bloques y dice no entender cómo funcionan o cómo hacer que sus personajes se muevan. Diego, por otro lado, indica que necesita más tiempo para acostumbrarse. En ambos casos, el monitor les explica nuevamente los conceptos vistos en las clases pero a modo de trabajo dirigido, es decir, el estudiante iba seleccionando y conectando bloques mientras el monitor le indicaba cómo hacerlo. Al final de la fase exploratoria, ambos logran mover a sus personajes por la pantalla. Carlos en particular comenta: *"El cantante ya puede moverse de lado a lado por el escenario, pero no entendí mucho cómo lo hace, siento que me enredo mucho"*. Diego por su parte menciona: *"Es entretenido esto de los bloques, pero me cuesta recordar cuáles usar y cuáles no, me falta práctica"*.

Finalmente, en el tercer bloque de la actividad, se les entrega el desafío a resolver usando los conceptos vistos en el taller. Este consistió en mover el personaje a la derecha hasta tocar el borde de la pantalla, punto en el que debía posicionarse automáticamente al costado izquierdo para seguir avanzando. Diego, por un lado, intentó resolverlo en primer lugar usando sólo bloques de movimiento, sin condicionales ni repeticiones. Para esto, contó manualmente cuántos pasos debía dar el personaje antes de llegar al borde de la pantalla. Al mencionarle que esta manera está incorrecta y que debe usar tanto repeticiones como condicionales para lograr el objetivo, indica que no sabe cómo resolverlo y pide la respuesta. El monitor intenta orientarlo dejándole los bloques en particular que le servirán para la respuesta, logrando ensamblarlos correctamente. Al consultarle a Diego por su razonamiento, comenta: *"No se me había ocurrido usar el bloque de repetición, me parecía más fácil de la otra forma, pero esta se ve más corta y creo que se entiende mejor"*, indicando que que no se le había ocurrido usar esos bloques, a pesar de entender cómo funcionan. Por otro lado, a Carlos se nota frustrado, y luego de un tiempo sin decir nada, indica que no entiende del todo el desafío. Al repetirle la explicación con un ejemplo, Carlos indica que no se le ocurre cómo resolver el ejercicio, por lo que se adopta la medida usada con Diego, indicar qué bloques forman parte de la solución para luego conectarlos. Carlos finalmente logra completar el desafío pero siendo fuertemente guiado por el monitor. Al ver el resultado menciona: *"Ahí se mueve bien, pero no siento que podría haberlo hecho solo, me enredo con tantos bloques, siento que no los entiendo tan bien"*.

Tabla 5.6: Percepción de participantes en sesiones de Thinking Aloud con Scratch.

Nombre	**Utilidad**	**Sesgo de género**	**Confianza**	**Valor Percibido**
Carlos	De acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	De acuerdo
Diego	De acuerdo	De acuerdo	No estoy seguro	De acuerdo

Una vez completado el desafío, se les pide a los estudiantes que completen el formulario de salida, donde se evalúan sus cargas físicas y cognitivas. Como se observa en la Tabla

5.7, Carlos reportó una baja carga física y temporal, además de bajo nivel de frustración, siendo sus valores más bajos. Por otro lado, indica un nivel medio de carga mental y esfuerzo. En cuanto a la percepción de su desempeño, reportó la actividad como medianamente difícil, tanto el proceso de llegar a la solución como el de crear bloques nuevos. Los valores reportados por Diego siguen la misma línea, con una baja carga física, temporal y frustración. Del mismo modo, percibe el proceso de llegar a la solución como medianamente difícil, al igual que crear bloques y entender la herramienta en general.

En cuanto a la percepción de los participantes respecto a Scratch, las respuestas en la tabla 5.8 se mantienen en la línea del formulario inicial. En particular, Carlos mantiene su percepción inicial tanto en términos de Utilidad, Sesgo de Género, Confianza y Valor percibido. Además, expresa un alto nivel de Engagment de la herramienta y de Satisfacción por haber logrado el desafío. Diego, por otro lado, reafirma sus respuestas iniciales, incrementando particularmente el nivel de Confianza en sus habilidades computacionales. Asimismo, indica un alto nivel tanto de Engagement y Satisfacción al usar la herramienta y lograr el desafío.

Tabla 5.7: Respuestas a formulario de cargas físicas y cognitivas de participantes en sesiones de Thinking Aloud con Scratch

Nombre	**MD**	**PD**	**TD**	**P**	**E**	**F**	**Cr**	**C**	**S**
Carlos	5	2	1	5	4	10	4	8	4
Diego	6	1	1	6	6	8	4	8	4

Como última actividad, se les pregunta por opiniones generales respecto a la actividad y su evaluación de Scratch como herramienta. Ambos participantes expresaron haberse divertido durante la experiencia, destacando particularmente el juego con el gato de Scratch. Carlos destacó fuertemente las opciones de personalización, particularmente las pestañas de disfraces y sonido: *"Encontré muy entretenida la aplicación, sobretodo poder hacer dibujos y ponerle sonidos, es como hacer una película, y se puede programar para que mi personaje haga cualquier cosa"*. Diego, por el otro lado, destacó el uso de bloques, indicando que fue más cómodo de lo que pensaba antes de la actividad. Esto se debe a que, según Diego, tenía otras expectativas de lo que significaba programar, por lo que se sintió sorprendido al ver que bastaba con conectar bloques: *"Cuando pensaba en programar no me imaginaba que era sólo decirle al computador qué quiero hacer. Al final es sólo ir dándole ordenes de a poco. Me gustó mucho ir acumulando los bloques para que el gato se moviera por toda la pantalla, pero necesito más práctica porque aún no entiendo todos los bloques"*.

Además, Carlos indica que, si bien logra entender qué hacen los bloques, le costó entender cómo llegar a la solución y cómo usarlos en conjunto. Además, comenta que la cantidad de bloques lo confundían, dado que muchas veces usaba un bloque que no hacía lo que él estaba buscando: *"Lo que sí me costó fue hacer que el personaje se teletransportara, no entendía muy bien qué tenía que hacer ni cómo hacerlo, además me costó entender bien los bloques al haber tantos"*. Diego, por su lado, hizo hincapié en que la cantidad de bloques lo abrumó, por lo que agradeció la ayuda brindada por el monitor al resolver el desafío, indicándole qué bloques usar: *"Siento que sabía cómo resolver el desafío, pero no cómo programarlo. Me perdí mucho con los bloques y no estoy seguro de cuál me servía o hacia lo que yo quería. Cuando me dijo qué bloques eran parte de la solución fue mucho más fácil"*. Finalmente, ambos indican que

están interesados en aprender más de esto. Carlos en particular indica que le gustaría tener más tiempo para probar cada bloque y entenderlos, mientras que Diego señala su intención de implementar un juego más grande con sus amigos. Al consultarles en particular por el ambiente de la actividad y cómo se sintieron, Carlos señaló que se sintió cómodo usando la herramienta, pero que no es algo que usaría en su tiempo libre: *"Lo encontré entretenido, y me gustó la aplicación, pero no me llamó mucho la atención como otros juegos"*. Diego, por su lado, señala que le gustó aprender jugando, recordándole sus clases de computación en el colegio: *"El juego es muy divertido, sobretodo ir probando cosas con los bloques para mover al personaje. Sería entretenido que enseñaran esto en el colegio con mis compañeros"*.

Tabla 5.8: Respuestas a formulario Smiley-o-meter de participantes en sesiones de Thinking Aloud con Scratch

Nombre	**Utilidad**	**Sesgo de Género**	**Confianza**	**Valor Percibido**	**Engagement**	**Satisfacción**
Carlos	De acuerdo	Totalmente de acuerdo	Totalmente de acuerdo	De acuerdo	Totalmente de acuerdo	De acuerdo
Diego	De acuerdo	De acuerdo	De acuerdo	Totalmente de acuerdo	De acuerdo	De acuerdo

Capítulo 6

Discusión e implicancias

6.1. Discusión

En esta sección se discuten los resultados obtenidos en los experimentos, evaluando qué tanto impactó el mecanismo de interacción en la percepción de los estudiantes y cómo esto afectó su motivación. Además, se comentan posibles nuevas aristas de investigación que no fueron consideradas en primera instancia, que pueden dar pie a nuevas espiras de trabajo.

6.1.1. Expectativa, Valor Percibido y Motivación

Según la teoría Expectativa-Valor, la motivación de los estudiantes para aprender a programar es directamente proporcional a la Expectativa del estudiante y el Valor Percibido por éste en la actividad de programar. La Expectativa se refiere a qué tan seguro se siente de que logrará desenvolverse en la actividad con éxito. Del mismo modo, el Valor Percibido es la percepción subjetiva que tiene el individuo del beneficio obtenido de llevar a cabo la tarea, como puede ser la Importancia de la tarea, el Valor Intrínseco o disfrute, Utilidad de realizar la tarea o su Costo.

En la Prueba A/B, los participantes de entrada reportaron que, tanto para Scratch como para Gesture Coding, la Utilidad percibida era medianamente alta, con un promedio de 64 % y 68 % por herramienta, respectivamente. Esto indica que existe un porcentaje importante de la muestra que no está segura de la Utilidad de la computación en su vida cotidiana, con un porcentaje reducido que estaba en desacuerdo de esta aseveración. Sin embargo, la gran mayoría de los participantes de ambos grupos (91 % y 96 % respectivamente) declaró que sí es valioso para ellos aprender programación. Esto implica que, en un inicio, el Valor Percibido de aprender a programar no estaba en la Utilidad percibida por los participantes, sino en la Importancia de la tarea.

Por otro lado, la Confianza declarada por los participantes en un inicio estaba en un nivel medio-alto, pero nuevamente con un porcentaje importante de la muestra que se presentaba insegura de sus capacidades frente a la computación. Esto implica que la Expectativa de los participantes se encontraba en un nivel medio-alto. Esto se explica por las concepciones previas que pueden tener los participantes de lo que significa programar. En particular,

comentarios de los participantes indican que esto se debía a elementos de la cultura popular y la forma en la que se muestra la disciplina. Por ejemplo, eran comunes comentarios de los participantes preguntando si los monitores son capaces de entrar a sistemas de bancos o intervenir redes sociales. Esto también se relaciona con la Utilidad percibida del participante, dado que no veían la programación como algo que podía solucionar problemas de su vida cotidiana, sino problemas a gran escala o a futuro.

En el cuestionario final, se observa un aumento considerable en la percepción de Utilidad de los participantes. En particular, el grupo que utilizó Gesture Coding sólo presenta participantes que estén neutrales o de acuerdo con la Utilidad de la programación en su vida cotidiana, y el porcentaje de participantes que son neutrales también disminuye. Para el grupo con Scratch, el porcentaje de participantes que no están de acuerdo con que la programación sea útil se mantuvo, pero el porcentaje de participantes neutros disminuyó considerablemente. A pesar de que existe diferencia en el impacto entre ambos grupos, estas no son lo suficientemente significativas como para atribuirle el efecto encontrado al uso de una herramienta en particular. Esto implica que, si bien la actividad impactó positivamente en la percepción de Utilidad de los participantes, no hubo mayor diferencia en la herramienta utilizada.

Del mismo modo, se observa un aumento considerable en la Confianza de los participantes en sus habilidades computacionales. Para el caso del grupo Scratch, la totalidad de los participantes indicó estar de acuerdo o totalmente de acuerdo con sentirse seguros de lo que hacen frente a un computador. Para el caso de Gesture Coding, existe un 8 % que declara no sentirse seguro con sus habilidades computacionales, y un 8 % que permanece neutro, disminuyendo en relación al cuestionario inicial. Esto implica que, el haber participado en el taller de programación impactó en la autopercepción de seguridad de los participantes, es decir, ahora sienten mayor confianza en sí mismos para desenvolverse en el área. Esto significa que la Expectativa de los participantes aumento considerablemente respecto a su valor inicial. Sin embargo, y al igual que en el caso de la Utilidad percibida, la diferencia entre los grupos no es estadísticamente significativa como para atribuir el efecto a una herramienta por sobre la otra.

Por otro lado, el Valor Percibido por los participantes se mantuvo constante, es decir, la participación en el taller no impactó en la Importancia percibida por los participantes. Los resultados del formulario de extensión, aplicado una semana después de terminada la experiencia, se mantuvieron en el tiempo sin diferencia significativa entre una versión y la otra. Esto implica que el aumento en la percepción de Utilidad y Confianza de los participantes no son derivados del "Novelty Effect", es decir, no son resultado de la emoción del momento al haber terminado el taller.

En cuanto al Valor Intrínseco de aprender a programar, es decir, el disfrute o interés en realizar la actividad, se observa que se encuentra en un nivel alto al final de la actividad. En efecto, las respuestas de los participantes al campo de Engagement indica que percibieron la actividad como entretenida y disfrutable. Más aún, considerando que este efecto se ve mantenido en el tiempo una semana después del taller. Caso similar con la Satisfacción percibida por el participante respecto a su desempeño. Estos dos parámetros permiten concluir que, luego de realizada la actividad, los participantes no sólo consideraron útil y valioso el

aprender a programar, sino que también lo percibieron como una actividad placentera. Luego, hubo un aumento en el Valor Intrínseco percibido por el participante respecto a aprender a programar.

Este aumento considerable de Utilidad y Valor Intrínseco percibidos implican un aumento en el Valor Percibido en general del participante respecto a aprender a programar. A esto se suma un aumento en la Confianza del participante en sus habilidades computacionales, que derivan en un aumento de la Expectativa del mismo respecto a la tarea de programar. Luego, según la Teoría Expectativa-Valor, este aumento de ambos campos implica un aumento en la motivación del participante. Este aumento puede verse mantenido en el tiempo gracias a los resultados del formulario de extensión, donde los resultados mantuvieron la tendencia positiva de las respuestas de los participantes.

Sin embargo, dado que la diferencia entre ambos grupos no es estadísticamente significativa, no se puede concluir que una herramienta en particular impacte en la motivación del participante de mejor o pero manera que la otra. Esto significa que ambas herramientas, con su respectivo mecanismo de interacción, impactan en la motivación de la persona de manera similar. No obstante, dado el tamaño de la muestra obtenida, es necesario replicar la actividad para poder asegurar esta hipótesis.

Dados estos resultados, se enfoca el análisis en cómo se produce este aumento de Valor Percibido y Expectativa. Esto se debe a que durante la ejecución de la actividad, se notaron distintos comportamientos de los participantes que no se consideraron al momento de diseñar el primer experimento.

Durante el segundo experimento, cuatro nuevos participantes fueron seleccionados para realizar la actividad, esta vez siguiendo el protocolo Thinking Aloud. De estos cuatro participantes, dos lo realizaron con Scratch, y dos con Gesture Coding.

En el caso de Gesture Coding, los datos reportados por los participantes en los cuestionarios siguen la tendencia encontrada en la prueba A/B. Por un lado, se ve un aumento en la confianza de los participantes respecto a sus habilidades computacionales, con un alto valor percibido. Sin embargo, la utilidad percibida de la herramienta baja un nivel, donde ambos participantes se ven inseguros de ello. Al igual que en las pruebas A/B, esto se explica dada su participación en la actividad, puesto que esto les permitió aclarar concepciones previas erróneas de lo que significa programar. Luego, al ver que lograron hacer un programa con la herramienta, aumentó su confianza respecto a sus habilidades. Sin embargo, al estar envueltos en un ambiente lúdico, es posible que no vean la utilidad en la programación así como la vieron, razón por la que este valor se ve disminuido en el cuestionario final.

Para las sesiones con Scratch, uno de los participantes se mantiene constante en sus respuestas del cuestionario, mientras que el otro ve aumentada su confianza y valor percibido. Al igual que con Gesture Coding, este aumento de valores es consecuencia de haber realizado el taller con una herramienta, y no puede ser atribuido a Scratch en particular.

Durante la fase exploratoria de cada bloque de la actividad, se le pidió a los participantes que verbalizaran cómo se sentían y cuáles eran sus intenciones en cada minuto. En este sentido, las actitudes de cada participante se pueden resumir en su objetivo. Dado que los par-

ticipantes solían hacer referencia a Gesture Coding como un *Juego*, estos objetivos o actitudes pueden relacionarse con la Taxonomía de Jugadores de Bartle [82]. Esta teoría caracteriza a los jugadores de videojuegos según sus objetivos y comportamientos en este mundo. En particular, se mencionan cuatro personajes que caracterizan al jugador: (1) *Achiever*, que busca lograr objetivos y superar metas dentro del mundo virtual; (2) *Explorador*, que busca interactuar con el mundo a su propio ritmo sin necesidad de un objetivo fijo, suelen encontrar errores o bugs dentro de los videojuegos; (3) *Socializer*, que busca interactuar con otros jugadores más por el aspecto social que por el videojuego como tal; y (4) *Killers*, que se enfocan en la competencia con otros jugadores o personajes, buscando objetos o habilidades poderosas.

Para el caso con Gesture Coding, Alberto tenía en mente un objetivo o una tarea en particular que quería realizar, que es mover al personaje por el escenario. Por el otro lado, Bruno estaba motivado por probar la aplicación y ver qué ocurría al apilar bloques, sin un objetivo claro en frente. En este sentido, Alberto presenta rasgos de un jugador de tipo Achiever, con un objetivo claro e intención de superar desafíos. En palabras del mismo participante *"Me gustó haber completado el desafío, pero me gustaría tener más bloques para hacer que el personaje se mueva de otras formas"*. A pesar de no poseer experiencia previa, el participante comenta sus intenciones de integrar lo aprendido en un proyecto más grande, como un videojuego completo. Esto implica un aumento en el Valor Intrínseco percibido por el participante debido al alza de interés.

Bruno, por otro lado, presenta rasgos de un jugador de tipo Explorer. Estos jugadores se caracterizan por su tendencia a salirse del camino establecido y explorar fuera de los límites. En particular, Bruno presentó este tipo de comportamiento con Gesture Coding al explorar los tipos de bloques y ver qué ocurría. Incluso menciona su poco interés en realizar el desafío: *"Me gustó haber resuelto el desafío, pero me costó y no me gustó mucho esa parte, prefiero conectar bloques y ver qué pasa."*. En este sentido, Bruno disfrutó explorando Gesture Códing y probando sus límites, aumentando el Valor Intrínseco percibido por este.

Si bien ambos participantes que usaron Gesture Coding vieron su Valor Intrínseco aumentado, estos se interesaron en aspectos distintos de la plataforma. Esto, sumado al hecho de que ambos participantes presentaron rasgos de un tipo de jugador según la Taxonomía de Bartle, indica que la actividad y la herramienta fueron percibidas como algo lúdico y disfrutable.

En cuanto a las sesiones con Scratch, ambos participantes declararon haberse divertido con la experiencia, pero nuevamente se observan focos distintos de interés. Por un lado, Carlos se enfoca en las opciones de personalización de la plataforma y, a pesar de entender los bloques de manera independiente, indica no entender cómo usarlos. Diego, en cambio, se interesa en mover al personaje por la pantalla usando los bloques disponibles. En este caso, sólo uno de los participantes presenta interés en el uso de bloques, Diego, mientras que Carlos se veía frustrado al momento de usarlos. Usando la Taxonomía de Jugadores de Bartle, Diego presenta rasgos de un jugador tipo Achiever, mientras que Carlos presenta más rasgos de tipo Explorador. El caso de Diego es similar al de Alberto, en el sentido de que ambos estaban intentando avanzar en un objetivo similar, que era mover al personaje por un espacio. La principal diferencia entre ambos es que Diego, al no comprender y encontrar los bloques que

necesitaba, terminaba abrumado. Carlos, por el otro lado, se diferencia del caso de Bruno en el sentido de que su interés por explorar se alejaba de los bloques.

Al igual que con Gesture Coding, los intereses de los participantes que usaron Scratch también se diferencian en el enfoque presentado. Esto implica que, a pesar de ver incrementado el Valor Intrínseco de cada participante, la razón de este aumento difiere.

Comparando los casos de herramientas, para Gesture Coding se observa que el foco de interés de ambos participantes son los bloques, ya sea explorándolos o planteándose desafíos. Sin embargo, con Scratch el enfoque de sólo uno de los participantes son los bloques, mientras que el otro está más interesado en funcionalidades externas, como lo son las opciones de personalización.

El resultado observado con el estudio Thinking Aloud confirma las observaciones hechas por los monitores en las pruebas A/B. En particular, estos comentaron que gran parte de los participantes del grupo con Scratch enfocaron su atención en opciones de personalización y no de programación con bloques, haciendo énfasis en la edición de sonidos y dibujo. Mientras que con Gesture Coding, los participantes presentaron comportamientos similares a los de Alberto y Bruno. Por lado se observaban participantes enfocados en el desafío, intentando lograr un objetivo, y por el otro participantes que exploraban la herramienta de manera libre, sin seguir del todo las indicaciones del monitor asignado.

Esta diferencia es importante porque, a pesar de que la gran mayoría de los participantes reportó un aumento en su Valor Percibido, este incremento se debió a distintos factores. En conclusión, si bien en ambas herramientas se vio una alta Satisfacción de uso, esta se debió principalmente al uso de bloques y programación en mayor medida con el grupo de Gesture Coding.

Finalmente, los datos sugieren que ambas herramientas son tienen el potencial de aumentar el Valor Percibido del estudiante y su Expectativa. Esto lo logran captando el interés del participante en alguna actividad, ya sea programar con los bloques o funcionalidades adyacentes, como manejo de sonidos y dibujo en el caso de Scratch. Este fenómeno permite concluir que ambas herramientas permiten aumentar la motivación por la programación desde distintos enfoques. Sin embargo, el tamaño de la muestra de ambos experimentos no es suficiente para asegurarlo.

Con la intención de profundizar este efecto encontrado, se procede a analizar los datos cualitativos encontrados desde el punto de vista del entorno de aprendizaje percibido en las actividades

6.1.2. Entorno de Aprendizaje

El siguiente análisis está basado en los comentarios hechos por los monitores durante las pruebas A/B, y los comportamientos observados por los participantes de las sesiones con el protocolo Thinking Aloud. Para efectos de este análisis, se entiende entorno de aprendizaje como la percepción de los participantes respecto al ambiente de la actividad en general. Este ambiente puede ser percibido como lúdico, es decir, algo poco estructurado cuyo principal objetivo es disfrutar el uso de la herramienta; o de aprendizaje, un ambiente más estructurado

donde el principal objetivo es aprender a programar y usar los bloques por sobre el disfrute del participante.

Esta diferencia es importante porque afecta la percepción de los estudiantes. Siguiendo la metodología de Informal Learning, un ambiente menos estructurado y que se perciba como más lúdico le permite al participante desarrollar conocimiento a su ritmo. Sin embargo, si el participante no está interesado en aplicar nuevos conocimientos, no necesariamente profundizará en ellos. Dado que el principal objetivo de Gesture Coding es romper la barrera inicial de entrada a la programación usando controles de movimiento, un ambiente de aprendizaje lúdico resulta mejor opción. Además, un ambiente lúdico o informal abre paso a un aumento en la Expectativa de los participantes, al desarrollar conocimiento a su propio ritmo.

Según lo reportado por los monitores, durante las sesiones de Gesture Coding los participantes eran mucho más cooperativos entre sí que con Scratch. Esto significa que los participantes constantemente se acercaban a ver lo que hacían sus compañeros para imitar o comentar el código. Este comportamiento se puede atribuir a la naturaleza informal de la actividad, por lo que se sentían con libertad de explorar la herramienta asignada como quisieran. Por otro lado, el trabajo en Scratch fue descrito como más personal, con cada participante preocupado de su trabajo individual.

Por otro lado, los monitores también comentan que en las sesiones de Scratch los participantes se presentaban más resistentes a usar bloques, enfocándose en las funcionalidades gráficas de la herramienta, como el dibujo y el sonido. Además, se observó una notable frustración en los participantes al momento de realizar el desafío. El ambiente poco estructurado de la actividad derivó en que los participantes eligieran no trabajar con bloques y no profundizar en el conocimiento. Esto es precisamente lo indicado por la metodología de Informal Learning, que indica que el conocimiento se estanca si el participante no presenta interés. Dado esto, para lograr que los participantes se enfocaran en el desafío planteado, fue necesario establecer más estructura y guiarlos a usar los bloques de Scratch.

Dada la evidencia descrita anteriormente, el ambiente observado con Gesture Coding se asemeja más a un ambiente lúdico, donde los participantes podían explorar libremente los contenidos a su propio ritmo. El ambiente en Scratch, por otro lado, al ser más individual y estructurado, se percibe como un ambiente de aprendizaje más tradicional.

Durante las sesiones con Thinking Aloud se observa un fenómeno similar. Los participantes que usaron Gesture Coding se refirieron frecuentemente a la herramienta como algo que les gustaría explorar de manera libre, disfrutando de su experiencia. Por otro lado, los participantes de las sesiones con Scratch indican que, si bien se divirtieron durante la actividad, no es algo que usarían de manera independiente. Es posible que esto se deba a que para resolver el desafío planteado, todos los participantes que utilizaron Scratch requirieron algún tipo de guía, dada la cantidad de bloques que están disponibles. Esto tiene el potencial de disminuir la expectativa del participante, dado que observa la cantidad de bloques que no conoce y no entiende.

Finalmente, se puede concluir que Gesture Coding tiene el potencial de romper la barrera de entrada inicial de mejor manera. Esto dado que el ambiente lúdico que es percibido por los participantes se traduce en un aumento del Valor Intrínseco e Importancia percibida,

al potenciar la cooperación entre participantes. Sin embargo, esto no puede ser atribuido directamente al uso de mecanismos de control en particular, dado que puede ser resultado del diseño de la herramienta como tal. Es por esto que se procede a analizar los resultados desde el punto de vista de los mecanismos de interacción.

6.1.3. Mecanismo de Interacción

En esta sección se discuten los resultados del experimento desde el punto de vista del mecanismo de interacción y su incidencia en la motivación del participante. Los mecanismos de interacción siendo evaluados son mouse y teclado, con metáfora de interacción tradicional Point & Click; y los Joy-Cons de Nintendo Switch, cuya metáfora de interacción son gestos libres realizados con controles de movimiento.

En el cuestionario de cargas de Nasa-TLX, tanto las cargas cognitivas como físicas resultaron similares para ambas versiones de la actividad, con Gesture Coding y Scratch. Sin embargo, en teoría, la carga física debió ser mayor con Gesture Coding por el uso de controles de movimiento. Este efecto se puede atribuir a la calibración del instrumento, dado que los participantes no comprendieron del todo la pregunta o no existía un criterio uniforme entre ambos grupos de lo que significa "cansarse al usar la herramienta". Por otro lado, la similitud en cuanto a cargas cognitivas indica que ambas herramientas representan una carga similar al momento de usarse, por lo que la única carga que efectivamente se está alterando es la física con los controles de movimiento.

Respecto a comentarios cualitativos, los monitores observaron frustración en los participantes al usar Gesture Coding en un inicio. Esto dado que usar la herramienta conlleva un proceso de adecuación. Sin embargo, luego de acostumbrarse, los participantes disfrutaban del uso de la herramienta. Por otro lado, el uso de mouse y teclado no representó problemas en los participantes dado que poseían experiencia previa en su uso cotidiano. Esto puede afectar directamente la barrera de entrada a la programación dado que una herramienta requiere un proceso de adecuación, mientras que la otra usa un mecanismo de interacción al que ya están familiarizados. Según la Teoría Expectativa-Valor, esto se reflejaría en una mayor expectativa para el uso de controles de movimiento. Sin embargo, este mecanismo de interacción también posee un mayor valor percibido por los participantes [1], por lo que el efecto en la motivación dependería de qué tan difícil es comparado con Mouse y Teclado. En particular, en este Caso de Estudio los participantes de Gesture Coding reportaron, en general, acostumbrarse a este mecanismo de interacción luego del proceso de adecuación. Luego, en este aspecto, el impacto en la motivación y en la barrera de entrada sería positivo por parte de los controles de movimiento.

Como se menciona en la sección anterior, el ambiente de aprendizaje informal en las sesiones de Scratch llevó a los estudiantes a estancarse en cuanto al conocimiento adquirido, evitando usar los bloques de programación. Esto, sumado a que no tenían incentivo adicional para usarlo, los llevó a quedarse enfocados en las secciones de sonido y dibujo que percibieron como más lúdicas. En este sentido, el mecanismo de interacción Point & Click con mecánicas Drag and Drop de los bloques no llamaron la atención de los participantes como para ser usados. Por otro lado, los bloques al estar organizados por colores y funcionalidad, los participantes se guiaban por estas características para buscar el bloque deseado. Esto

conlleva a que, si hay dos bloques en la misma categoría con el mismo color y forma, pero que hagan cosas distintas, el participante termina confundiendo y abrumado. En efecto, como menciona Diego en el protocolo Thinking Aloud *"No se me había ocurrido usar los bloques de esa manera, no los encontraba"*. Una forma de abordar este problema sería usar el principio de diseño Progressive Disclosure [60], o revelación progresiva, de tal manera que ciertas funcionalidades estén ocultas hasta que el usuario haya aprendido a usar lo más básico. Sin embargo, ese rasgo encontrado no necesariamente está ligado al mecanismo de interacción de Scratch, sino a su diseño de interfaces y arquitectura de información.

En el caso de Gesture Coding, si bien algunos estudiantes prefirieron usar el mouse para seleccionar algunas opciones en lugar del control, la única forma de seleccionar bloques era usando gestos libres. Esto provocó que los participantes agruparan los bloques por color, forma y gesto. En efecto, según un comentario de un monitor *"Los participantes solían decir que necesitaban un bloque de tipo L, Circular o de flecha, haciendo alusión al gesto en lugar de su color o categoría"*. Según la teoría de Embodied Cognition [17], que sostiene que el aprendizaje puede ocurrir mediante interacciones del cuerpo con el entorno, el uso de gestos puede complementar y mejorar el aprendizaje, mientras que restringirlos puede afectarlo negativamente. Dado esto, el hecho de usar gestos ayudaba a los estudiantes a interiorizar de mejor manera los conceptos, dado que generar un bloque como tal requería esfuerzo cognitivo, para seleccionarlo, y físico, para crearlo. Esta mejora en el aprendizaje se ve reflejada en la actitud de los participantes frente al desafío. En particular, durante las sesiones de Thinking Aloud, los participantes que usaron Gesture Coding se vieron más dispuestos a enfrentarse al desafío que los participantes con Scratch. Estos últimos necesitaron una guía más explicita de parte del investigador para resolver el desafío.

Asimismo, el fenómeno observado impacta directamente en la barrera de entrada a la programación. Según la Taxonomía de Bloom [39], existen seis niveles de habilidades cognitivas para el aprendizaje: (1) Recordar, (2) Comprender, (3) Aplicar, (4) Analizar, (5) Evaluar y (6) Crear. Según lo observado anteriormente, el uso de gestos impacta directamente en los primeros dos niveles de la Taxonomía. En particular, el uso de gestos en el proceso de aprendizaje ayuda al participante a recordar dónde estaba cada bloque, mediante el dibujo del gesto, y a comprender lo que hace mediante el significado del gesto que está realizando. Además, mediante la realización del taller y el planteamiento del desafío, se logra extender este impacto al tercer nivel de la Taxonomía que es aplicar lo aprendido. Dado esto, es posible deducir que, en términos de aprendizaje, el uso de gestos en programación ayuda a introducir conceptos y romper la barrera de entrada inicial a la programación. Esto también impacta en la motivación del estudiante dado que, al ayudar a desarrollar habilidades básicas en computación, la expectativa del estudiante aumenta. Sin embargo, es necesario estudiar más a fondo esta hipótesis replicando el estudio como un caso confirmatorio con un mayor número de participantes.

Si bien los datos parece indicar que integrar gestos en el proceso de programación mejora cómo los estudiantes aprenden los conceptos, no tenemos suficiente evidencia para asegurarlo. Además, es posible que el fenómeno observado responda a la diferencia en las interfaces de Scratch y Gesture Coding, así como al hecho de que Scratch posee más funcionalidades, que actúan como distractores.

Por otro lado, en la prueba A/B se observó una diferencia en el ambiente de aprendizaje de cada herramienta. Este fue percibido como más lúdico con Gesture Coding que con Scratch. Según comentarios de los monitores, los participantes solían hacer la observación de que Gesture Coding les recordaba a un videojuego. Este comentario se repite una vez más en las sesiones con Thinking Aloud, donde mencionan que, dado los controles usados, relacionan el sistema a estar en un videojuego. En el caso de Scratch, los participantes indicaban que la aplicación era lúdica y entretenida, pero hacían más mención a que les gustaba dibujar y aplicar sonidos. Dada la evidencia recolectada, es posible que esta diferencia en la percepción de los estudiantes se deba al uso de gestos o al control como tal. En este sentido, el uso de gestos o controles de movimiento es una mecánica que rompe con los mecanismos de controles tradicionales como Point & Click, y, según Embodied Cognition, el uso de gestos no sólo potencia el aprendizaje de los conocimientos por parte de los estudiantes, sino que aumenta el interés del participante al usar una Interfaz Natural. En este sentido, una Interfaz Natural tiene la característica de ser más intuitiva para el usuario, siendo los gestos libres especialmente intuitivos y disfrutables en niños. Por otro lado, el uso de mecanismos tradicionales como mouse y teclado no representa un cambio para el participante respecto a otras actividades. Esto es mencionado por uno de los participantes de la actividad con Thinking Aloud: *"Lo encontré entretenido, y me gustó la aplicación (Scratch), pero no me llamó mucho la atención como otros juegos, no es algo que usaría en mis tiempos libres"* (Carlos, H, 12 años), mientras que el otro participante menciona que le gustaría integrarlo en el colegio: *"El juego es muy divertido, sobretodo ir probando cosas con los bloques para mover al personaje. Sería entretenido que enseñaran esto en el colegio con mis compañeros"* (Diego, H, 12 años). Los datos parecen indicar que el mecanismo de interacción efectivamente tuvo incidencia en el valor percibido por los participantes, motivándolos a programar. Sin embargo, la evidencia no es suficiente para generalizar este fenómeno.

Por otro lado, también es posible que esta diferencia en el ambiente detectado entre ambas herramientas sea algo más lúdico y relacionado con videojuegos. Gesture Coding utiliza como mecanismo de interacción y como medio para detectar movimiento los controles Joy-Con de Nintendo Switch, una consola de videojuego. Dado esto, es posible que los participantes se hayan mostrado más abiertos a aprender en la plataforma dado que el control utilizado está asociado con una actividad lúdica, como lo es jugar videojuegos.

Finalmente, en el estudio se observó una diferencia en la percepción de los participantes respecto a la programación influenciado por el mecanismo de interacción utilizado. Ya sea por el tipo de control utilizado o por el uso de gestos, el valor percibido por los estudiantes se vio mejorado, así como la utilidad y confianza percibidas. Esto implica que, (1) el uso de mecanismos de interacción no convencionales, en este caso gestos controlados con el Joy-Con de Nintendo Switch, permite que los participantes perciban una sensación de satisfacción mayor, aumentando su motivación; (2) Dado el uso de gestos como interfaz natural adaptada a los participantes, los estudiantes se presentan más abiertos a interactuar con el sistema y aprender a programar, ayudando a reducir la barrera de entrada, y (3) Los estudiantes fueron capaces de obtener el conocimiento entregado en la actividad, resolviendo el desafío final de la actividad aplicando el conjunto minimal de habilidades de pensamiento computacional.

6.2. Implicancias

Dados los resultados discutidos en la sección anterior, a continuación se presentan implicancias teóricas y prácticas de lo aprendido en este caso de estudio. Para esto, analiza la integración de gestos con distintas teorías y/o metodologías para enseñar programación.

6.2.1. Teóricas

Según Embodied Cognition, la integración del uso de interfaces naturales en un ambiente educativo puede ser beneficioso para el estudiante, dado que mejora la retención de conceptos e impulsa el valor percibido de los participantes en la actividad [17]. En particular, la literatura menciona que los gestos libres en particular son los preferidos por los niños de entre 10 y 12 años, por sobre gestos táctiles [1]. Actualmente existen diversas aproximaciones a este concepto en todas las áreas de aprendizaje. Un ejemplo de esto es *Word Out!* [76], que potencia el aprendizaje del alfabeto mediante el uso de gestos a cuerpo completo. A pesar de que el público objetivo de esta aplicación son niños de 4 a 7 años de edad, el concepto detrás es similar al de Gesture Coding: integrar corporalidad en el aprendizaje. La principal diferencia que tiene Word Out! con Gesture Coding en términos de mecánicas de interacción, es que en Word Out! el participante usa el cuerpo completo para generar explícitamente el contenido aprendido. Por ejemplo, si se desea generar la letra A, el estudiante debe formar este carácter con su cuerpo. En cambio, en Gesture Coding, los gestos no son directas traducciones del contenido en pantalla, sino que son interpretaciones de la noción que se quiere generar en el estudiante. Esto se debe a lo abstracto de los contenidos presentados. Un ejemplo de esto son los gestos para generar Condicionales, si bien el estudiante no escribe o dibuja un Condicional en pantalla, sí dibuja una línea con cada mano en sentidos opuestos, representando la bifurcación de un Condicional en un código. La principal contribución de este estudio radica en explorar cómo la incorporación de gestos libres en el proceso de enseñanza de programación puede mejorar la retención de conceptos y la motivación de los estudiantes, proporcionando una perspectiva valiosa sobre el uso de interfaces naturales y corporalidad en educación.

Esta diferencia obedece a la Teoría Educativa de Piaget [58], que propone centrar el proceso de aprendizaje en el entorno del estudiante según las habilidades físicas y cognitivas del mismo. Este plantea cuatro etapas de desarrollo cognitivo con creciente grado de abstracción:

1. **Sensoriomotriz:** Desde el nacimiento hasta los dos años, la persona reconoce su entorno y el ambiente.
2. **Pre-Operacional:** El niño, de 2 a 7 años, se enfoca en representaciones del mundo real como dibujos y juegos.
3. **Operaciones Concretas:** El niño, de 7 a 11 años, puede resolver problemas lógicos pero aún no en términos abstractos.
4. **Operaciones Formales:** La persona, de 11 años hasta su adultez, es capaz de pensar de manera abstracta y lidiar con situaciones hipotéticas.

El público objetivo de Word Out! se encuentra en la etapa de pre-operacional, por lo que los gestos realizados son una representación fiel del concepto, en este caso, el alfabeto. Por otro lado, el público objetivo se encuentra entre las etapas Pre-Operacional y de Operaciones Concretas, es decir, en proceso de desarrollo de su pensamiento abstracto. Dado esto, se

representan los conceptos de una manera un poco más abstracta. Dado esto se puede indicar que el uso de gestos no sólo debe ser comprendido por los participantes, sino que deben ser correctamente diseñados según el rango etario del público objetivo. Además, el uso de gestos diseñados y adaptados a estudiantes permite reducir la barrera de entrada a la programación. En efecto, según la Taxonomía de Bloom, las dos primeras etapas del aprendizaje son el Recordar y Comprender, habilidades fuertemente apoyadas por el uso de gestos [39]. Una vez superadas estas dos etapas, podemos decir que el estudiante ya está en proceso de aprendizaje y de adquirir la nueva habilidad.

Por otro lado, el uso de gestos no sólo potencia el aprendizaje de conceptos abstractos. Según Embodied Cognition, el uso de gestos libres es percibido como más intuitivo y usable por los estudiantes que mecanismos de aprendizaje más tradicionales [17]. Esto se debe principalmente a que los gestos son una forma de Interfaz Natural, que le permite al usuario interactuar con un sistema de manera más intuitiva. Además, es percibido por los participantes como más disfrutable que otros tipos de gestos, impactando directamente en su Valor Intrínseco percibido.

Luego, el uso de gestos impacta positivamente en el aprendizaje de conceptos y el valor intrínseco de los estudiantes. Según la Teoría Expectativa-Valor, esto se puede interpretar como un aumento en la Expectativa, por facilitar el aprendizaje de conceptos, y un aumento en el Valor Percibido, por el aumento del Valor Intrínseco. Esto deriva en un aumento de la motivación del estudiante por aprender programación.

6.2.2. Prácticas

El hecho de que la integración de gestos a un lenguaje visual de programación haya ayudado a subir la motivación y a internalizar conceptos abstractos, abre la puerta a integrarlo junto a otras metodologías o prácticas de aprendizaje. Por ejemplo, el uso de gestos o de corporalidad en una actividad de aprendizaje, como fue el caso de estudio evaluado en este trabajo, impactó en cómo los estudiantes perciben el ambiente de aprendizaje. En particular, el uso de gestos o de corporalidad en el proceso de aprendizaje hace que la actividad sea percibida más como un ambiente lúdico y menos como un ambiente rígido de aprendizaje. La ventaja de esto es que la motivación del estudiante es más alta, y es más resiliente a errores. Esto debido a que, si no logra realizar una tarea o no entiende algún concepto, lo ve como un desafío a superar, y no un obstáculo en su aprendizaje.

Esto va de la mano con lo planteado por el Aprendizaje Informal, que indica que un ambiente más informal o lúdico tiene la característica de ser más distendido, y por lo tanto, menos estresante y abrumador para el estudiante [26]. Esto le permite aprender a su propio ritmo e integrar los conocimientos que desee. Sin embargo, existe el potencial riesgo de que el estudiante no desee explorar más allá de lo que sienta seguro, llegando a un estancamiento del aprendizaje. El uso de controles de movimiento ayuda a mitigar este riesgo al asignar un gesto en particular a cada concepto importante que se desea enseñar. Dado que el dibujar gestos y uso de controles de movimiento corresponde a la parte lúdica de la experiencia, se usa como incentivo para explorar la herramienta y reconocer los conceptos vistos en la sesión expositiva.

Sin embargo, el uso de gestos para el aprendizaje no es algo sustentable en el tiempo. Tal

como indica la teoría de Aprendizaje Informal y de Lenguajes Visuales, este tipo de prácticas tiene un límite en cuánto a conceptos representables. Es por esto que, tarde o temprano, el estudiante tendrá que hacer una migración a lenguajes más tradicionales si desea expandir más su conocimiento en programación. Esto también ocurre con el uso de gestos, estos no son sustentables para códigos más complejos o con muchos bloques, por lo que resultan ideales para incentivar el aprendizaje temprano de programación en niños.

En conclusión, tanto en la teoría como en la práctica, la integración de gestos en la programación impacta positivamente en el aprendizaje de los estudiantes. Esto dado que aumenta su motivación, ayuda a internalizar mejor conceptos abstractos y genera un ambiente distendido e informal donde los estudiantes están más abiertos a aprender conceptos. Sin embargo, resulta importante que los gestos se limiten a ser usados sólo para introducir a los estudiantes a la programación, dado que no es una práctica sustentable en el tiempo, por cómo va aumentando la complejidad de los conceptos.

6.3. Revisitando las Preguntas de Investigación

Según los resultados y aprendizajes de este trabajo, es posible dar respuesta a las preguntas de investigación planteadas en el Capítulo 1. Las respuestas a las preguntas de investigación derivan directamente del cumplimiento de los objetivos general y específicos presentados en el mismo capítulo. Dado que las preguntas RQ2 y RQ3 son derivadas de RQ1, esta será respondida al final.

En cuanto a la pregunta RQ2: *"¿Cuál es el impacto del uso de Joy-Con como mecanismo de control de un lenguaje de programación en el valor percibido por parte de niños y niñas en etapa escolar?"*, se puede concluir que el valor percibido por los estudiantes se ve aumentado respecto a un mecanismo de control tradicional, como lo es el Mouse y Teclado. El uso de Joy-Cons en un ambiente de aprendizaje permite el uso de gestos y controles de movimiento, mecanismo de interacción que es percibido como lúdico y divertido por parte del estudiante. Además, dado que los controles son usados en un contexto de juego, al pertenecer a una consola de videojuego, el estudiante percibe un ambiente de aprendizaje más distendido e informal. Dado esto, el estudiante ve aumentado especialmente el Valor Intrínseco de programar, percibiendo la actividad como algo más lúdico que complejo e intimidante.

En cuanto pregunta RQ3: *"¿Qué tan satisfactorio es el uso de un lenguaje de programación controlado por Joy-Cons en contraste con uno controlado por dispositivos de entrada tradicionales para niños y niñas de 10 y 12 años?"*, se puede concluir que el uso de un lenguaje controlado por Joy-Cons es igual o más satisfactorio que uno controlado por mecanismos de entrada tradicionales. En efecto, el uso de Joy-Cons en un ambiente educativo permite integrar gestos y controles de movimiento al proceso de aprendizaje, ayudando al estudiante a comprender conceptos más abstractos. Por otro lado, realizar gestos con un control de videojuego como son los Joy-Con de Nintendo Switch implica experiencia más disfrutable por parte del estudiante. Esto deriva en que el estudiante se sienta satisfecho tanto en su desempeño frente a la aplicación, como a su experiencia de uso.

Finalmente, respecto a la pregunta RQ1: *"¿Cuál es el impacto del uso de Joy-Cons en la reducción de la barrera de entrada en lenguajes de programación basados en bloques?"*, se

puede concluir que el impacto del uso de Joy-Cons en la reducción de la barrera de entrada es positiva. Esto dado que, como se menciona en las preguntas anteriores, el uso de Joy-Cons permite aplicar gestos y controles de movimiento en el lenguaje. Como esto es percibido por el estudiante como algo lúdico, su Valor Intrínseco aumenta, según la RQ2. Del mismo modo, el estudiante se siente más satisfecho con su desempeño al haber internalizado los conceptos de mejor manera, mejorando la confianza en sus habilidades, según RQ3. Luego, de acuerdo con la Teoría Expectativa-Valor, el uso de Joy-Cons como mecanismo de control de un lenguaje de programación visual permite aumentar el Valor Percibido y la Expectativa del estudiante, lo que deriva en un aumento de su motivación. Finalmente, este aumento en su motivación deriva en que perciba la programación como un desafío a cumplir en lugar de una actividad intimidante, disminuyendo así su barrera de entrada.

Es importante mencionar que, dada la metodología de caso de estudio, estas respuestas a las preguntas de investigación pueden asegurarse sólo para el caso específico estudiado. Para generalizar y expandir los aprendizajes de este estudio es necesario replicar el estudio con un mayor número de participantes.

6.4. Espacios de Mejora

En esta sección se detalla el trabajo futuro respecto a la herramienta, así como a su evaluación y análisis. Si bien Gesture Coding fue bien recibida por los participantes, uno de los comentarios más comunes era sobre nuevas funcionalidades, como personalización o cantidad de bloques. Es por esto que, en una futura iteración del proyecto, es posible integrar nuevas funcionalidades como personalización del personaje y fondo, además de agregar bloques que permitan controlar sonidos y/o música, de manera análoga a Scratch.

Sin embargo, implementar esto puede transformarse en un arma de doble filo dado que puede terminar siendo un distractor de la funcionalidad principal de la herramienta, que es programar con gestos. Es por esto que se propone una implementación que aplique el concepto de Progressive Disclosure, es decir, que permita desbloquear nuevos bloques al ir superando desafíos o ciertas metas. De esta forma, el participante se verá limitado inicialmente de la cantidad de bloques y funcionalidades existentes. Además, al ir incorporando funcionalidades de manera gradual, el estudiante no se sentirá tan abrumado con las opciones, dado que las irá conociendo de a poco. Durante el caso de estudio se observó este fenómeno con Scratch, donde los participantes se veían notablemente abrumados por la cantidad de bloques y funcionalidades del sistema.

En cuanto al diseño de Gesture Coding, es posible maximizar lo más posible el valor percibido respecto a la herramienta por los participantes. Para esto, sería necesario llevar a cabo un estudio enfocado en dos etapas: Interfaces Tradicionales e Interfaces de Gestos. Para el primero, sería necesario recolectar la opinión de participantes y expertos de dominio en cuanto a la arquitectura de la información y organización general del sistema. Esto incluye iconografía y colores. En este sentido, resulta fundamental ajustar la sensibilidad del Joystick respecto al cursor al momento de programar. Esto incluye ajustar la velocidad y tamaño tanto del cursor como de los bloques y botones, según lo indique la Ley de Fitts. En el caso de Interfaces de Gestos, se pueden realizar ajustes a la lectura de gestos, o a la sensibilidad de los controles de movimiento para que dibujarlos sea más sencillo. Además, es posible agregar

nuevas versiones de gestos para que la tolerancia al error por parte del sistema sea mayor.

Finalmente, una opción de mejora es aislar de mejor manera las variables a medir para tener más claro cuánto es el impacto efectivo del uso de controles de movimiento en el aprendizaje de programación. En este sentido, durante este caso de estudio, se notó que las funcionalidades extra de Scratch que no poseía Gesture Coding impactaron negativamente en el desempeño del mismo Scratch, dado que el enfoque de los participantes iba en las opciones de personalización en lugar de los bloques. Para mitigar este efecto, es posible adaptar Gesture Coding a una versión que use mecanismos de control tradicionales como mouse y teclado. De esta forma, el único cambio que efectivamente se estaría evaluando es el mecanismo de control. Esto permitiría obtener un respaldo estadístico de los resultados obtenidos en este trabajo, permitiendo abrir nuevas aristas para esta investigación.

6.5. Posibles Futuras Líneas de Investigación

De los resultados de este trabajo se abren nuevas interrogantes respecto al uso de gestos en el aprendizaje. En particular, esta práctica es aún poco común en la literatura, especialmente en el aprendizaje de lenguajes de programación. Dado esto, se proponen nuevas líneas de investigación que se desprenden de los resultados de este trabajo.

Si bien se observa que Gesture Coding logró aumentar la motivación de lo participantes y generar un ambiente lúdico, es posible este se deba al controlador específico utilizado. En este caso, se usaron los Joy-Con de Nintendo Switch, controles diseñados y usados en ambientes netamente lúdicos, como lo son los videojuegos. Dado esto, es posible que los estudiantes se presentaran más abiertos a aprender los conceptos y usar la herramienta en particular por el controlador utilizado. Es por esto que se plantea implementar variantes de Gesture Coding que use otros controladores más accesibles y que estén presentes en otros tipos de contextos. En particular, se propone adaptar el uso de gestos a teléfonos inteligentes que permitan la detección de movimiento. De esta forma, se podría evaluar si el resultado se debe específicamente al uso de gestos y/o controles de movimiento o a que el controlador utilizado corresponde a una consola de videojuegos. Otra opción es implementar variantes de Gesture Coding que permita usar otros controles de videojuegos con detección de gestos, como Wiimote, PSMove, DualShock, etc. De esta forma se puede evaluar específicamente qué aspecto del uso de gestos es el que más aporta a generar valor en los estudiantes. Por otro lado, se podría explorar el impacto del uso de otro tipo de Interfaces Naturales, como son los comandos de voz, similar a los de un asistente virtual, o seguimiento de la mirada, permitido por un dispositivo EyeTracker. Estos, además, pueden ser incorporados en entornos de realidad extendida, ya sea usando Realidad Aumentada (AR) o Realidad Virtual (VR).

6.6. Limitaciones

Si bien los resultados obtenidos en este caso de estudio son prometedores respecto al uso de gestos en el aprendizaje de la programación, es necesario resaltar que los resultados son válidos para este entorno en particular. Es decir, los resultados obtenidos no son generalizables a otros entornos u otros participantes, consecuencia directa de la metodología de Caso de Estudio, dado su carácter exploratorio y explicativo.

Esto se debe principalmente a que los resultados obtenidos no poseen poder estadístico claro debido al tamaño reducido de la muestra. Esto se debió en gran parte a limitaciones operativas causadas por pandemia causada por COVID-19 que impidieron ejecutar y replicar en condiciones óptimas los escenarios empíricos diseñados. El impacto fue que, al ser el público objetivo parte del grupo de riesgo, no se pudo realizar demasiadas evaluaciones en vivo que permitieran refinar las hipótesis. Es por esto que el análisis es en su mayoría descriptivo y aplicado, basado en observaciones.

Para poder generalizar estos resultados, es necesario replicar la actividad con nuevos participantes hasta llegar a saturación.

Capítulo 7

Conclusión y Trabajo Futuro

El desarrollo de habilidades de pensamiento computacional corresponde a una de las llamadas Habilidades del Siglo XXI, pues permite desarrollar otras competencias en el estudiante, aplicables a su vida cotidiana. Entre estas habilidades se encuentran la descomposición de problemas, la habilidad de resolverlos y pensamiento crítico. Existen numerosas aproximaciones para desarrollar estas habilidades, como lenguajes de programación tradicionales o adaptadas para niños, entre otras, pero comenzar a aprender a programar es una tarea compleja que puede resultar intimidante para un principiante.

Aprender una nueva habilidad puede resultar atemorizante y complejo, y precisamente ese sentimiento puede generar resistencia por parte del estudiante. Este fenómeno en particular ocurre en el proceso de aprendizaje de un lenguaje de programación, donde influencias de la cultura popular muestran a la disciplina como algo complejo y difícil de entender. A esto, se le suma el hecho de que aprender a programar implica resolver problemas lógicos complejos mientras se aprende un lenguaje nuevo, con sintaxis y semánticas nuevas. Esto termina generando una sensación de inseguridad y falta de confianza en niñas y niños que desean aprender, por lo que se muestran poco abiertos a desarrollar estas habilidades. A esto se le conoce como barrera de entrada, es decir, perder el miedo y dar el paso inicial para poder aprender y desarrollar una nueva habilidad.

Una forma de abordar dicho problema es utilizar lenguajes de programación adaptados para principiantes, en este caso, Lenguajes Visuales basados en bloques. Estos permiten simplificar la sintaxis de la tarea de programar. Ejemplos de este tipo de lenguajes son Scratch, Blockly o Snap!; sin embargo, todos poseen en común el mismo mecanismo de entrada que es Mouse y Teclado. Si bien los lenguajes son distintos, para ojos de un niño principiante en programación pueden ser análogos, por lo que el problema motivacional aún persiste.

Dado este contexto, se presenta Gesture Coding, un Lenguaje de Programación Visual cuyo mecanismo de control son los gestos y controles de movimientos. Para esto, Gesture Coding hace uso de los Joy-Con de Nintendo Switch, permitiendo que el usuario controle el lenguaje con una mezcla de botones y gestos libres. Gesture Coding está fuertemente basado en Scratch, siendo la principal diferencia la manera de elegir bloques. Estos están

organizados en categorías que son representadas por un gesto en particular. De esta forma, si el estudiante desea obtener un bloque de cierta categoría, basta con que realice el gesto asociado a esa categoría de bloques y podrá crearlo. Esto tiene el potencial de ser percibido por el estudiante como una tarea disfrutable en lugar de intimidante, lo que ayudaría a bajar la barrera de entrada. Para esto, se toma de base la Teoría Expectativa-Valor, que indica que si el Valor Percibido, o interés en la herramienta, aumenta junto con la Expectativa, o confianza en que se logrará tener un buen desempeño, la motivación del estudiante aumentará. En un inicio, Gesture Coding sólo cubría bloques de tipo Booleano, Numérico y Secuencial o Movimiento.

Para realizar esto, este trabajo se dividió en tres espiras: (1) Refinamiento y prueba piloto de Gesture Coding, para evaluar su estado actual y poder extenderlo; (2) Extensión, donde se implementan nuevos bloques para cubrir el conjunto minimal de habilidades de pensamiento computacional, así como mejoras de diseño; y (3) Caso de estudio, donde se lleva a cabo un taller con el público objetivo para observar el comportamiento de la herramienta en un entorno controlado. Durante todo el trabajo se toma como base la Teoría Expectativa-Valor para desarrollar motivación, siendo referente para evaluaciones, diseños y talleres.

La primera espira de este trabajo, arrojó como resultado que Gesture Coding es capaz de desarrollar habilidades de pensamiento computacional de manera similar a Scratch, a pesar de que la carga física de usar los controles sea mayor. Esto significa que, si bien Gesture Coding posee menos bloques y un mecanismo de interacción distinto a Scratch, tiene el potencial de desarrollar las habilidades de pensamiento computacional de manera similar a este. Además, se toma nota de que los participantes mencionan lo disfrutable que es utilizar controles de movimiento para programar.

Durante la segunda espira se implementan bloques de Ciclos y Variables, así como modificaciones generales de diseño y estabilidad. En particular, se agregan dos nuevos gestos al sistema para las nuevas categorías y se crea un sistema de variables que permita almacenar valores. Al final de esta espira se obtiene una versión de Gesture Coding que cubre el conjunto minimal de habilidades de pensamiento computacional planteadas por Rich et al. [52].

Finalmente, durante la última espira se lleva a cabo un caso de estudio exploratorio en dos etapas: (1) Prueba A/B, donde se realizan cuatro talleres de programación con un aproximado de 10 niños cada uno, para comparar Scratch con Gesture Coding en términos de Cargas Cognitivas y Físicas, Valor Percibido y Expectativa, así como comentarios generales; y (2) Thinking Aloud, que consistió en una repetición de la actividad anterior pero esta vez con un sólo estudiante por sesión siguiendo el protocolo Thinking Aloud.

La primera etapa del caso de estudio mostró que entre Gesture Coding y Scratch no existe una diferencia significativa en términos de aprendizaje o Valor Percibido por los participantes. Sin embargo, durante las sesiones se observan actitudes particulares de los participantes que llevan a pensar que, si bien no existe diferencia estadística significativa entre ambas herramientas, sí existe una diferencia en la percepción. Esto significa que el mecanismo de interacción, en este caso, el uso de gestos, sí impacta en la percepción del participante.

Con la intención de profundizar este fenómeno, se realiza la segunda etapa del caso de estudio siguiendo el protocolo Thinking Aloud. En esta versión del taller, se observa que

efectivamente con Gesture Coding los participantes tienen una mejor disposición a aprender y desarrollar habilidades de pensamiento computacional que con Scratch. Además, se observa un ambiente más lúdico y más distendido al usar controles de movimiento. Como conclusión de las observaciones se puede indicar que:

- Usar controles de movimiento con los Joy-Con de Nintendo Switch efectivamente es percibido como más lúdico y disfrutable por los estudiantes que usar Mouse y Teclado. Esto provoca una mejor disposición a aprender y genera un ambiente más distendido durante los talleres. Esto valida la Hipótesis 1.1.
- El uso de gestos permite internalizar mejor conceptos de programación complejos y abstractos, permitiendo que los participantes aumenten su Expectativa respecto a la programación. Luego, al aumentar la confianza de los participantes en sus habilidades, el estudiante percibe una mayor satisfacción de uso de la herramienta. Esto valida la Hipótesis 1.2.

A partir de estas dos observaciones y, según la Teoría Expectativa-Valor, se puede concluir que el uso de gestos es efectivo al momento de elevar la motivación de los programadores novatos. Esto implica que integrar controles de movimiento al ambiente educativo efectivamente es capaz de bajar la barrera de entrada a la programación, validando la Hipótesis 1 de este trabajo.

Sin embargo, dada la cantidad de datos recolectados y la naturaleza de la metodología de Caso de Estudio utilizada en este trabajo, no es posible generalizar este conocimiento sin antes replicar el estudio con un mayor número de participantes.

Finalmente, es posible continuar desarrollando Gesture Coding agregando más categorías de bloques y más gestos. En este sentido, es posible seguir una línea de implementación y diseño similar a la de Scratch, agregando opciones de personalización o multimedia. Es importante tener en cuenta que, de implementarse dichas funcionalidades, estas no deben ser el centro de atención del estudiante. Dado esto, existe el riesgo de que estas nuevas funcionalidades terminen siendo una distracción en lugar de un complemento. Se propone implementarlo usando Progressive Disclosure, de tal forma que estas funcionalidades queden disponibles una vez el estudiante haya superado ciertas metas.

Por otro lado, se plantea como trabajo futuro replicar la evaluación de la herramienta para estudiar la validez externa de los resaltados obtenidos. Para esto, se propone implementar una versión de Gesture Coding con mecanismos de control tradicionales, Mouse y Teclado. Con esto, es posible realizar una comparación donde la única variable modificada sea el mecanismo de interacción. Este estudio tomaría de base la Prueba A/B realizada en este Caso de Estudio, haciendo énfasis en el valor percibido del participante y aplicando cuestionarios de Pensamiento Computacional para evaluar el aprendizaje del mismo.

Además, se proponen nuevas líneas de investigación abiertas por los resultados de este caso de estudio. En particular, se plantea estudiar cómo afecta en el valor percibido y el desempeño del estudiante el haber usado un controlador que proviene específicamente de un contexto lúdico, los videojuegos. Para esto se propone implementar versiones de Gesture Coding con diversos controladores usados en distintos contextos, ya sean teléfonos inteligentes, controles de otro tipo de consolas u otros ambientes de interacción, como Realidad Aumentada (AR)

y Realidad Virtual (VR). Todo esto con la intención de maximizar el valor percibido por el estudiante, aumentando su motivación.

El panorama educativo actual destaca la importancia del pensamiento computacional como una habilidad esencial para enfrentar los desafíos del siglo XXI. Sin embargo, la tarea de introducir a los estudiantes en el mundo de la programación sigue siendo un desafío significativo debido a la percepción de los estudiantes del área. Motivar a un estudiante novato a aprender un concepto complejo es una tarea desafiante. Sin embargo, si se logra que el estudiante perciba la tarea como algo disfrutable, la motivación viene de la mano. En este trabajo se exploró el impacto del uso de gestos y controles de movimiento en la motivación de los estudiantes para aprender a programar. Se observó que aplicar este mecanismo de control impacta en la percepción de los estudiantes de la tarea, aumentando su motivación y disminuyendo la barrera de entrada. Estos hallazgos resaltan la importancia de considerar enfoques creativos y motivadores en la enseñanza de la programación, no solo para fomentar el pensamiento computacional, sino también para superar los obstáculos iniciales y hacer que el aprendizaje sea más accesible y atractivo para los principiantes.

Bibliografía

[1] Ab Rahman, M. S., Ali, N. M., and Mohd, M. Natural user interface for children: From requirement to design. In *International Visual Informatics Conference* (2017), Springer, pp. 612–624.

[2] Barradas, R., Lencastre, J. A., Soares, S., and Valente, A. Developing computational thinking in early ages: A review of the code. org platform.

[3] Barron-Estrada, M. L., Zatarain-Cabada, R., and Cardenas-Sainz, B. A. A natural user interface implementation for an interactive learning environment. In *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)* (2020), IEEE, pp. 341–343.

[4] Bau, D., Gray, J., Kelleher, C., Sheldon, J., and Turbak, F. Learnable programming: blocks and beyond. *Communications of the ACM 60*, 6 (2017), 72–80.

[5] Begel, A., and Ko, A. J. Learning outside the classroom.

[6] Bell, T., Rosamond, F., and Casey, N. Computer science unplugged and related projects in math and computer science popularization. *The multivariate algorithmic revolution and beyond: Essays dedicated to Michael R. Fellows on the occasion of his 60th Birthday* (2012), 398–456.

[7] Blackwell, A. F. First steps in programming: A rationale for attention investment models. In *Proceedings IEEE 2002 Symposia on Human Centric Computing Languages and Environments* (2002), IEEE, pp. 2–10.

[8] Blackwell, L. S., Trzesniewski, K. H., and Dweck, C. S. Implicit theories of intelligence predict achievement across an adolescent transition: A longitudinal study and an intervention. *Child development 78*, 1 (2007), 246–263.

[9] Blandford, A., Furniss, D., and Makri, S. *Qualitative HCI research: Going behind the scenes*. Morgan & Claypool Publishers, 2016.

[10] Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., et al. Developing computational thinking in compulsory education-implications for policy and practice. Tech. rep., Joint Research Centre (Seville site), 2016.

[11] Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., and Klemmer, S. R.

Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2009), pp. 1589–1598.

[12] Braun, V., and Clarke, V. Using thematic analysis in psychology. *Qualitative Research in Psychology 3*, 2 (2006), 77–101.

[13] Cho, M.-H., Demei, S., and Laffey, J. Relationships between self-regulation and social experiences in asynchronous online learning environments. *Journal of Interactive Learning Research 21*, 3 (2010), 297–316.

[14] Committee, K.-. C. S. F. S., et al. *K-12 computer science framework*. ACM, 2016.

[15] Cooper, K. M., Ashley, M., and Brownell, S. E. Using expectancy value theory as a framework to reduce student resistance to active learning: A proof of concept. *Journal of microbiology & biology education 18*, 2 (2017), 10–1128.

[16] Dabbagh, N., and Kitsantas, A. Personal learning environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning. *The Internet and higher education 15*, 1 (2012), 3–8.

[17] Danish, J. A., Enyedy, N., Saleh, A., and Humburg, M. Learning in embodied activity framework: A sociocultural framework for embodied cognition. *International Journal of Computer-Supported Collaborative Learning 15* (2020), 49–87.

[18] Eccles, J. Expectancies, values and academic behaviors. *Achievement and achievement motives* (1983).

[19] Eccles, J. S. Gender roles and women's achievement-related decisions. *Psychology of women Quarterly 11*, 2 (1987), 135–172.

[20] Fagerlund, J., Häkkinen, P., Vesisenaho, M., and Viiri, J. Computational thinking in programming with scratch in primary schools: A systematic review. *Computer Applications in Engineering Education 29*, 1 (2021), 12–28.

[21] Faul, F., Erdfelder, E., Buchner, A., and Lang, A.-G. Statistical power analyses using g* power 3.1: Tests for correlation and regression analyses. *Behavior research methods 41*, 4 (2009), 1149–1160.

[22] Fincher, S. A., and Robins, A. V. *The Cambridge handbook of computing education research*. Cambridge University Press, 2019.

[23] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental PSychology 74* (1954), 381–391.

[24] Garner, S., Haden, P., and Robins, A. My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42* (2005), pp. 173–180.

[25] Gärtig-Daugs, A., Weitz, K., Wolking, M., and Schmid, U. Computer science experimenter's kit for use in preschool and primary school. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education* (2016), pp. 66–71.

[26] Grover, S., Pea, R., and Cooper, S. Designing for deeper learning in a blended computer science course for middle school students. *Computer science education 25*, 2 (2015), 199–237.

[27] Gutierrez, F. J., Simmonds, J., Casanova, C., Sotomayor, C., and Hitschfeld, N. Coding or hacking? exploring inaccurate views on computing and computer scientists among k-6 learners in chile. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), pp. 993–998.

[28] Gutierrez, F. J., Simmonds, J., Hitschfeld, N., Casanova, C., Sotomayor, C., and Peña-Araya, V. Assessing software development skills among k-6 learners in a project-based workshop with scratch. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)* (2018), IEEE, pp. 98–107.

[29] Hao, Q., Wright, E., Barnes, B., and Branch, R. M. What are the most important predictors of computer science students' online help-seeking behaviors? *Computers in Human Behavior 62* (2016), 467–474.

[30] Hart, S. G. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting* (2006), vol. 50, Sage publications Sage CA: Los Angeles, CA, pp. 904–908.

[31] Hayes, G. R. The relationship of action research to human-computer interaction. *ACM Transactions on Computer-Human Interaction (TOCHI) 18*, 3 (2011), 1–20.

[32] Heiner, C. A robotics experience for all the students in an elementary school. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), pp. 729–734.

[33] Hermans, F., and Aivaloglou, E. To scratch or not to scratch? a controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12th workshop on primary and secondary computing education* (2017), pp. 49–56.

[34] Hermans, F., Swidan, A., Aivaloglou, E., and Smit, M. Thinking out of the box: comparing metaphors for variables in programming education. In *Proceedings of the 13th workshop in primary and secondary computing education* (2018), pp. 1–8.

[35] Hulleman, C. S., and Harackiewicz, J. M. Promoting interest and performance in high school science classes. *science 326*, 5958 (2009), 1410–1412.

[36] Jin, Q., Wang, D., Deng, X., Zheng, N., and Chiu, S. Ar-maze: a tangible programming tool for children based on ar technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children* (2018), pp. 611–616.

[37] Kakavas, P., and Ugolini, F. C. Computational thinking in primary education: A systematic literature review. *Research on Education and Media 11*, 2 (2019), 64–94.

[38] Kelleher, C., and Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR) 37*, 2 (2005), 83–137.

[39] Kiesler, N. Towards a competence model for the novice programmer using bloom's revised taxonomy-an empirical approach. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (2020), pp. 459–465.

[40] Kurniawan, O., Lee, N. T. S., Datta, S., Sockalingam, N., and Leong, P. K. Effectiveness of physical robot versus robot simulator in teaching introductory programming. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (2018), IEEE, pp. 486–493.

[41] Levy, R. B.-B., and Ben-Ari, M. Robotics activities–is the investment worthwhile? In *Informatics in Schools. Curricula, Competences, and Competitions: 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2015, Ljubljana, Slovenia, September 28-October 1, 2015, Proceedings 8* (2015), Springer, pp. 22–31.

[42] Liu, A. S., Schunn, C. D., Flot, J., and Shoop, R. The role of physicality in rich programming environments. *Computer Science Education 23*, 4 (2013), 315–331.

[43] Lizama, E., and Gutiérrez, F. J. DiseÑo de un entorno de programaciÓn interactiva controlado por joy-cons. Tech. rep., Universidad de Chile, 2021.

[44] Marsick, V. J., and Watkins, K. E. Informal and incidental learning. *New directions for adult and continuing education 2001*, 89 (2001), 25–34.

[45] McCall, D., and Kölling, M. Meaningful categorisation of novice programmer errors. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings* (2014), IEEE, pp. 1–8.

[46] Muenks, K., Wigfield, A., and Eccles, J. S. I can do this! the development and calibration of children's expectations for success and competence beliefs. *Developmental Review 48* (2018), 24–39.

[47] Nielsen, J. Ten usability heuristics.

[48] Oram, A., and Wilson, G. *Making software: What really works, and why we believe it*. O'Reilly Media, Inc., 2010.

[49] Papert, S. Mindstorms: children, computers, and powerful ideas january 1980.

[50] Polat, E., and Yilmaz, R. M. Unplugged versus plugged-in: examining basic programming achievement and computational thinking of 6th-grade students. *Education and Information Technologies 27*, 7 (2022), 9145–9179.

[51] Repenning, A., and Basawapatna, A. Smacking screws with hammers: Experiencing affordances of block-based programming through the hourglass challenge. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (2021), pp. 267–273.

[52] Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., and Franklin, D. K–8 learning trajectories derived from research literature: sequence, repetition, conditionals. *ACM Inroads 9*, 1 (2018), 46–55.

[53] Robins, A., Haden, P., and Garner, S. Problem distributions in a cs1 course. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (2006), pp. 165–173.

[54] Robins, A., Rountree, J., and Rountree, N. Learning and teaching programming: A review and discussion. *Computer science education 13*, 2 (2003), 137–172.

[55] Rodriguez, B., Kennicutt, S., Rader, C., and Camp, T. Assessing computational thinking in cs unplugged activities. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (2017), pp. 501–506.

[56] Rosenzweig, E. Q., Wigfield, A., and Eccles, J. S. 24 expectancy-value theory and its relevance for student motivation and learning.

[57] Rotherham, A. J., and Willingham, D. T. 21st-century" skills. *American Educator 17*, 1 (2010), 17–20.

[58] Sanghvi, P. Piaget's theory of cognitive development: a review. *Indian Journal of Mental Health 7*, 2 (2020), 90–96.

[59] Schanzer, E., Krishnamurthi, S., and Fisler, K. Blocks versus text: Ongoing lessons from bootstrap. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (2015), IEEE, pp. 125–126.

[60] Springer, A., and Whittaker, S. Progressive disclosure: empirically motivated approaches to designing effective transparency. In *Proceedings of the 24th international conference on intelligent user interfaces* (2019), pp. 107–120.

[61] Starrett, A. *Integrating self-determination and expectancy-value theories in examining the achievement of first-generation college students: A latent profile analysis examining relations between perceived choice, school valuing, and perceived competence and academic achievement*. PhD thesis, University of South Carolina, 2018.

[62] Sullivan, A., and Johnson, E. K. Beaded adventures: Crafting stem learning. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction* (2019), pp. 351–358.

[63] Taub, R., Armoni, M., and Ben-Ari, M. Cs unplugged and middle-school students' views, attitudes, and intentions regarding cs. *ACM Transactions on Computing Education (TOCE) 12*, 2 (2012), 1–29.

[64] Tsay, C. H.-H., Kofinas, A. K., Trivedi, S. K., and Yang, Y. Overcoming the novelty effect in online gamified learning systems: An empirical evaluation of student engagement and performance. *Journal of Computer Assisted Learning 36*, 2 (2020), 128–146.

[65] van der Werf, V., Aivaloglou, E., Hermans, F., and Specht, M. (how) should variables and their naming be taught in novice programming education? In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2* (2022), pp. 53–54.

[66] Van Der Werf, V., Zhang, M. Y., Aivaloglou, E., Hermans, F., and Specht, M. Variables in practice. an observation of teaching variables in introductory programming moocs. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (2023), pp. 208–214.

[67] Waite, J. Pedagogy in teaching computer science in schools: A literature review. *London: Royal Society 253* (2017).

[68] Weintrop, D., and Wilensky, U. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children* (2015), pp. 199–208.

[69] Weintrop, D., and Wilensky, U. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE) 18*, 1 (2017), 1–25.

[70] Whalley, J., Settle, A., and Luxton-Reilly, A. A think-aloud study of novice debugging. *Computing Education 23*, 2 (2023), 1.

[71] Wigdor, D., and Wixon, D. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[72] Wigfield, A., and Eccles, J. S. Expectancy–value theory of achievement motivation. *Contemporary educational psychology 25*, 1 (2000), 68–81.

[73] Wigfield, A., and Eccles, J. S. 35 years of research on students' subjective task values and motivation: A look back and a look forward. In *Advances in motivation science*, vol. 7. Elsevier, 2020, pp. 161–198.

[74] Wing, J. M. Computational thinking: What and why. In *Presentation slides from Trippel Helix Conference on Computational Thinking and Digital Competencies in Primary and Secondary Education Stockholm, Sweden. https://pdfs. semanticscholar. org/presentation/d20a/a49744877f2bb98d6ad303742be7bd025fcd. pdf* (2017), pp. 1580695435–1378800312.

[75] Wu, C.-C., Tseng, I.-C., and Huang, S.-L. Visualization of program behaviors: Physical robots versus robot simulators. In *Informatics Education-Supporting Computational Thinking: Third International Conference on Informatics in Secondary Schools-Evolution and Perspectives, ISSEP 2008 Torun Poland, July 1-4, 2008 Proceedings 3*

(2008), Springer, pp. 53–62.

[76] Yap, K., Zheng, C., Tay, A., Yen, C.-C., and Do, E. Y.-L. Word out! learning the alphabet through full body interactions. In *Proceedings of the 6th augmented human international conference* (2015), pp. 101–108.

[77] Yin, R. K. *Case study research and applications*, vol. 6. Sage Thousand Oaks, CA, 2018.

[78] Yu, J., Zheng, C., Tamashiro, M. A., Gonzalez-millan, C., and Roque, R. Codeattach: Engaging children in computational thinking through physical play activities. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction* (2020), pp. 453–459.

[79] Zaman, B., Abeele, V. V., and De Grooff, D. Measuring product liking in preschool children: An evaluation of the smileyometer and this or that methods. *International Journal of Child-Computer Interaction 1*, 2 (2013), 61–70.

[80] Zeevaarders, A. *An Exploratory Study of the Learning Progression of Scratch Users.* PhD thesis, Master's thesis, Open Universiteit, 2020.

[81] Zhang, L., and Nouri, J. A systematic review of learning computational thinking through scratch in k-9. *Computers & Education 141* (2019), 103607.

[82] Zuchowska, L., Kutt, K., and Nalepa, G. J. Bartle taxonomy-based game for affective and personality computing research. In *MRC@ IJCAI* (2021), pp. 51–55.

Anexos

A. Instructivo de Evaluación para Monitores

A.1. Planificación por sesión:

3 Bloques por sesión, con receso de 10 minutos en medio.

Bloque 1:

- Bienvenida y Preguntas de entrada.
- Introducción del sistema de programación de bloques con instrucciones generales de Scratch y Gesture Coding junto a secuencialidad y variables.
- Se deja el tiempo restante para actividades exploratorias por cada estudiante.

Bloque 2:

- Introducción general de Condicionales y Ciclos.
- Se deja el tiempo restante para actividades exploratorias por cada estudiante.

Bloque 3:

- Se les entregan tareas a realizar a modo de actividad práctica, en azul solo para Gesture Coding:
  - Ingresar a pantalla de gestos.
  - Dibujar figuras (Puede ser una estrella, flechas, etc.).
  - Evaluar qué tan intuitivos son los gestos. Reconocer los gestos de cada bloque. Interpretación de los gestos de manera cualitativa.
  - Crear un bloque de cada tipo.
  - Eliminar todos los bloques creados.
  - Conectar bloques para que, al ejecutarlos, hagan lo siguiente:
    - Crear una variable igual a 10. Este será el paso del personaje.
    - Mover al personaje a la derecha 20 veces con un ciclo.
    - Antes de cada paso, verificar la posición del personaje en el eje X. Si es mayor a 90, posicionar el personaje en las coordenadas $x : -150$ y $y : 0$.
- Se les entrega formulario de preguntas de salida junto a preguntas generales del taller.

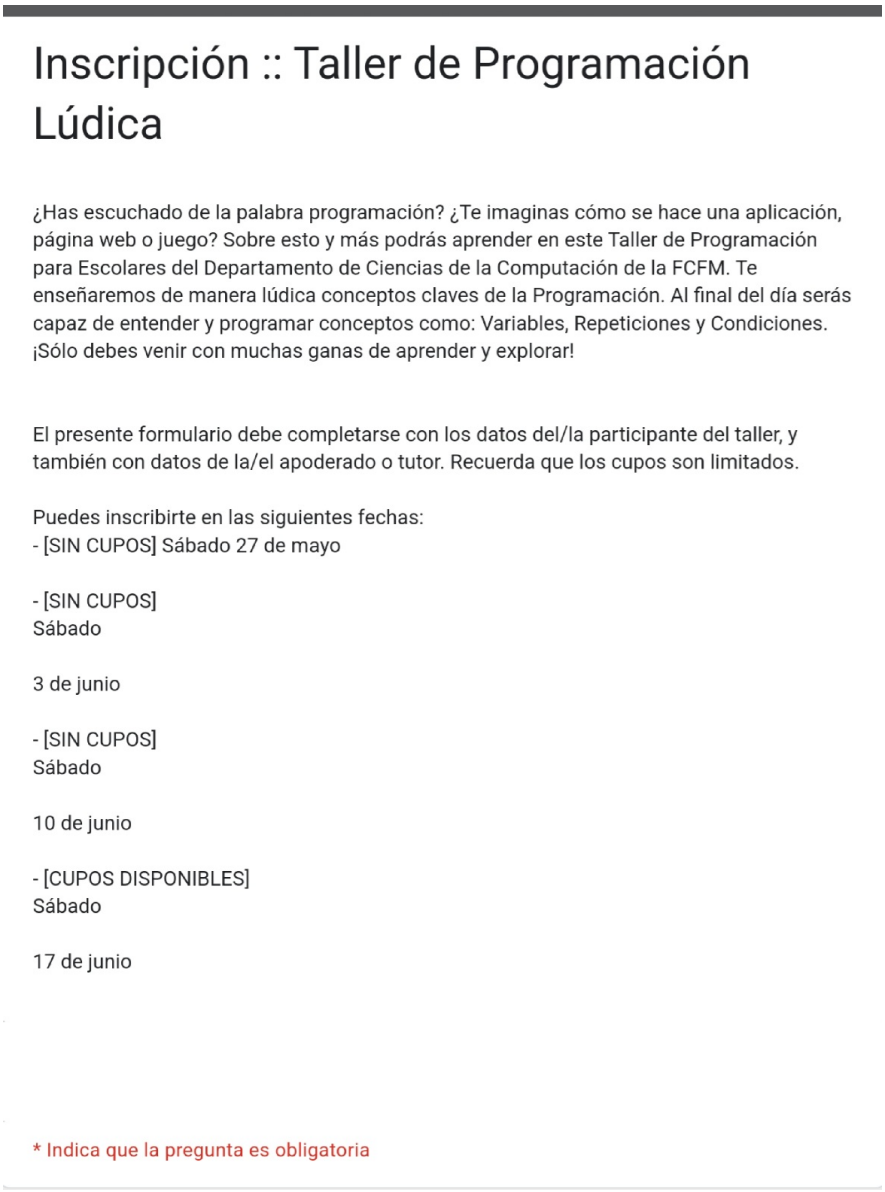
A.2. Post-Intervención

Se contacta a los participantes una semana después del taller para que completen un formulario para medir su expectativa después del taller.

A.3. Labores a monitorear

- Recibir y despedir a los estudiantes del taller. Para esto, se debe llegar media hora antes que comience la actividad, y retirarse una vez se hayan ido todos los participantes.
- Hacerse cargo de, a lo más, 5 estudiantes en paralelo. En caso de pedir permiso para ir al baño, uno de los monitores llevará a todos los estudiantes que lo deseen, sean o no de su grupo.
- Orientar y ayudar a los estudiantes en caso de que se pierdan en la aplicación. Idealmente sin dar la respuesta, sino empoderando a explorar.
- Recordar a los participantes que son ellos los que están evaluando la plataforma y no al revés.
- Si un estudiante logra completar una tarea antes de que termine el tiempo asignado, empoderarlo a seguir explorando la aplicación pero sin adelantar contenidos. Por ejemplo, si durante el bloque 1 el estudiante dice ya haber terminado de explorar, se le puede plantear un desafío en base a lo que construyó SIN USAR condicionales y ciclos.
- En caso de algún problema técnico, se reinicia la aplicación o la conexión de los controles.
- Anotar observaciones de los estudiantes tanto verbales como no verbales. Ejemplos de observaciones pueden ser:
  - El estudiante menciona que es muy difícil tomar un bloque.
  - Se le nota abrumado al estudiante.
  - En el último bloque, cuando un estudiante indique que terminó la actividad, revisar si cumple correctamente lo pedido y anotar la hora en la que avisó.

B. Formularios de Caso de Estudio



Inscripción :: Taller de Programación Lúdica

¿Has escuchado de la palabra programación? ¿Te imaginas cómo se hace una aplicación, página web o juego? Sobre esto y más podrás aprender en este Taller de Programación para Escolares del Departamento de Ciencias de la Computación de la FCFM. Te enseñaremos de manera lúdica conceptos claves de la Programación. Al final del día serás capaz de entender y programar conceptos como: Variables, Repeticiones y Condiciones. ¡Sólo debes venir con muchas ganas de aprender y explorar!

El presente formulario debe completarse con los datos del/la participante del taller, y también con datos de la/el apoderado o tutor. Recuerda que los cupos son limitados.

Puedes inscribirte en las siguientes fechas:
- [SIN CUPOS] Sábado 27 de mayo

- [SIN CUPOS]
Sábado

3 de junio

- [SIN CUPOS]
Sábado

10 de junio

- [CUPOS DISPONIBLES]
Sábado

17 de junio

* Indica que la pregunta es obligatoria

Figura 7.1: Formulario de Inscripción al Taller de Programación Lúdica - Parte 1

Datos de la o el participante
Necesitamos la siguiente información de quién asistirá al taller

¿Cuál es tu nombre? *

Tu respuesta

¿Correo electrónico? *

Tu respuesta

¿Cuántos años tienes? *

Tu respuesta

¿Cuál es tu género?

- ◯ Masculino
- ◯ Femenino
- ◯ Prefiero no decirlo
- ◯ Otros:

Figura 7.2: Formulario de Inscripción al Taller de Programación Lúdica - Parte 2

Un requisito para este taller es que el participante no tenga conocimiento en programación. ¿Confirma? *

- ◯ Sí
- ◯ No

Datos del apoderado/a

Necesitamos también algunos datos la persona adulta a cargo de el o la participante

Parentesco (ejemplo madre, padre, tutor, tía u otro) *

Tu respuesta

Nombre y Apellidos *

Tu respuesta

Correo electrónico del Apoderado *

Tu respuesta

Figura 7.3: Formulario de Inscripción al Taller de Programación Lúdica - Parte 3

Asentimiento Informado

Estimada/o participante del taller,

Como investigadores de la Universidad de Chile, estamos comprometidos con apoyar la educación de nuestro país. Para ello, dentro del marco de este taller realizamos investigación activa acerca de cómo los/as niños/as en edad escolar pueden adquirir y desarrollar habilidades de pensamiento computacional y programación de aplicaciones de software.

Para nosotros es muy importante entender cómo los/as estudiantes perciben y entienden las actividades que les proponemos. Es por eso que continuamente realizamos revisiones a las actividades propuestas y sintetizamos de manera agregada y anonimizada los resultados obtenidos.

A través de este documento, solicitamos a usted colaborar con el proyecto "Taller de Programación Lúdica". El autor de este trabajo es Sebastián Toro G. junto al profesor guía, el académico Francisco J. Gutiérrez, Profesor Asistente del Departamento de Ciencias de la Computación de la Universidad de Chile. Las únicas personas autorizadas a recopilar y procesar la información recogida en este proyecto son las que conforman el equipo de investigación del académico responsable.

Al completar este formulario manifiesto que voluntariamente deseo participar en este estudio y declaro conocer los términos de participación y el alcance de los datos recogidos en este proyecto de investigación.

* Indica que la pregunta es obligatoria

Figura 7.4: Formulario de Asentimiento - Parte 1

Nombre del/a participante: *

Tu respuesta

Correo del/a participante *

Tu respuesta

Con fecha: *

Fecha

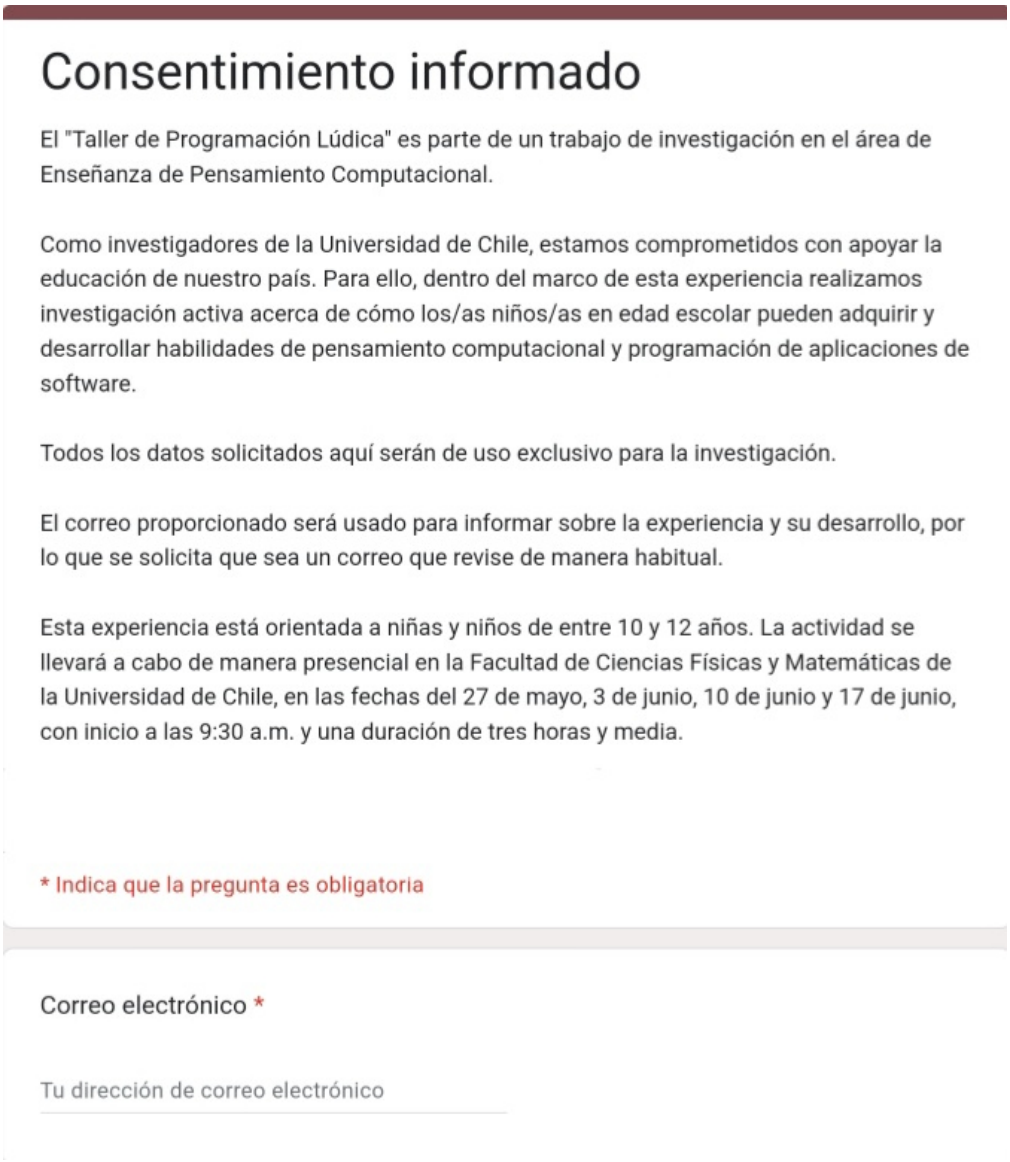
¿Acepta participar en esta actividad? *

- Sí
- No

Me comprometo a responder los cuestionarios entregados antes, durante y después de realizada la actividad. *

- Sí
- No

Figura 7.5: Formulario de Asentimiento - Parte 2



Consentimiento informado

El "Taller de Programación Lúdica" es parte de un trabajo de investigación en el área de Enseñanza de Pensamiento Computacional.

Como investigadores de la Universidad de Chile, estamos comprometidos con apoyar la educación de nuestro país. Para ello, dentro del marco de esta experiencia realizamos investigación activa acerca de cómo los/as niños/as en edad escolar pueden adquirir y desarrollar habilidades de pensamiento computacional y programación de aplicaciones de software.

Todos los datos solicitados aquí serán de uso exclusivo para la investigación.

El correo proporcionado será usado para informar sobre la experiencia y su desarrollo, por lo que se solicita que sea un correo que revise de manera habitual.

Esta experiencia está orientada a niñas y niños de entre 10 y 12 años. La actividad se llevará a cabo de manera presencial en la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile, en las fechas del 27 de mayo, 3 de junio, 10 de junio y 17 de junio, con inicio a las 9:30 a.m. y una duración de tres horas y media.

* Indica que la pregunta es obligatoria

Correo electrónico *

Tu dirección de correo electrónico

Figura 7.6: Formulario de Consentimiento Informado - Parte 1

Para nosotros es muy importante entender cómo los/as estudiantes perciben y entienden las actividades que les proponemos. Es por eso que continuamente realizamos revisiones a las actividades propuestas y sintetizamos de manera agregada y anonimizada los resultados obtenidos.
A través de este documento, solicitamos a usted colaborar con el proyecto "A Reference Framework of Software Development Practices Calibrated for Very Novice Programmers", patrocinado por ANID a través del Proyecto FONDECYT #11190248. Este proyecto está dirigido por el académico Francisco J. Gutiérrez, Profesor Asistente del Departamento de Ciencias de la Computación de la Universidad de Chile. Como asistentes de investigación participan profesionales y estudiantes de pregrado, Magíster y Doctorado de la misma Universidad. Las únicas personas autorizadas a recopilar y procesar la información recogida en este proyecto son las que conforman el equipo de investigación del académico responsable.

Al firmar este documento manifiesto que voluntariamente deseo participar en este estudio y autorizo a mi pupilo/a a participar en este proyecto. Asimismo, declaro conocer los términos de participación y el alcance de los datos recogidos en este proyecto de investigación.

Nombres y apellidos de quien asistirá *

Tu respuesta

Nombres y apellidos de la persona adulta responsable *

Tu respuesta

Teléfono de contacto *

Tu respuesta

Fecha de hoy *

Fecha

Figura 7.7: Formulario de Consentimiento Informado - Parte 2

Yo, apoderado/a del/a estudiante arriba indicado y con fecha arriba indicada autorizo al Departamento de Ciencias de la Computación de la Universidad de Chile a utilizar los resultados de aprendizaje (tales como actividades parciales, avances de proyectos, proyecto final) de mi pupilo/a como insumos de investigación en la mejora continua del taller realizado u otros fines que el equipo estime pertinente. Esta información será procesada de manera individual o agregada, siempre manteniendo en reserva la identidad del/a alumno/a, respetando su integridad y autoría, pudiendo utilizarse estos datos en reportes técnicos, artículos de investigación (por ejemplo, en conferencias y revistas especializadas sometidas a referato). *

- ◯ Sí
- ◯ No

Yo, apoderado/a del/a estudiante arriba indicado y con fecha arriba indicada autorizo al Departamento de Ciencias de la Computación de la Universidad de Chile a solicitar la participación de mi persona o la de mi pupilo/a en entrevistas o grupos focales, con la finalidad de mejorar continuamente el taller realizado y otros fines estrictamente académicos y de investigación que el equipo considere pertinente. Entiendo que dar mi autorización no implica necesariamente mi participación activa en estas instancias, y que tengo el derecho de desistir de participar en el momento que lo considere pertinente sin tener que dar necesariamente explicaciones para ello. *

- ◯ Sí
- ◯ No

Yo, apoderado/a del/a estudiante arriba indicado y con fecha arriba indicada autorizo al Departamento de Ciencias de la Computación de la Universidad de Chile a registrar en forma de audio, imágenes y/o video, la participación de mi pupilo/a en las actividades que componen el taller. Entiendo que la utilización de estos registros se realizará de manera individual o colectiva, y doy mi consentimiento para su publicación con fines estrictamente académicos y de difusión no comercial, tanto en prensa escrita, digital como redes sociales institucionales. *

- ◯ Sí
- ◯ No

Me comprometo a responder los cuestionarios entregados antes, durante y después de realizada la actividad. *

- ◯ Sí
- ◯ No

Por favor, contacte al académico responsable de este proyecto si tiene alguna pregunta sobre el estudio, sobre sus derechos, o sobre el uso de la información recopilada durante la investigación. Los datos de contacto se indican más abajo. El Prof. Francisco J. Gutiérrez suscribe el compromiso de respetar cabalmente las condiciones detalladas más arriba.

Figura 7.8: Formulario de Consentimiento Informado - Parte 3

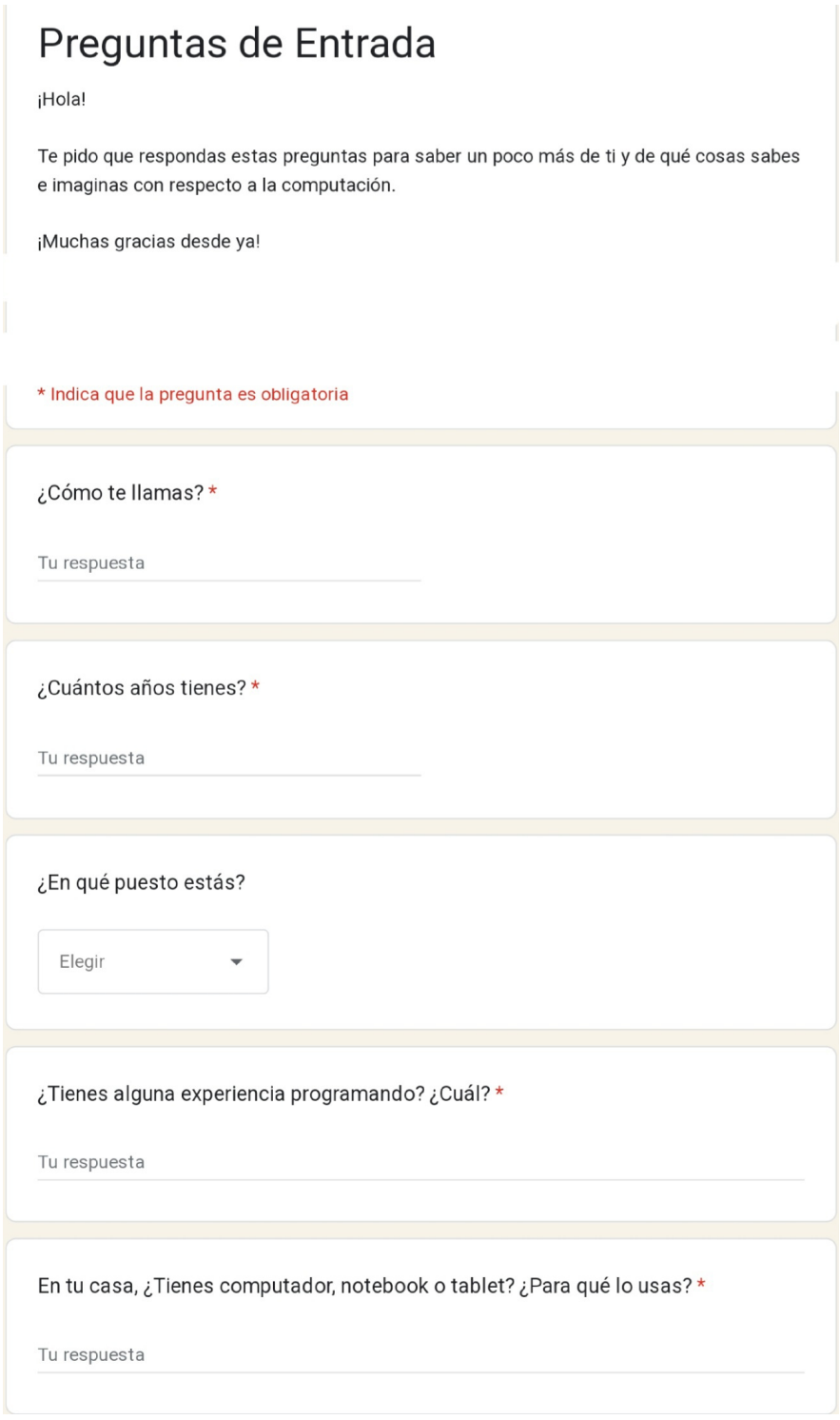


Figura 7.9: Preguntas de Entrada con Gesture Coding - Parte 1

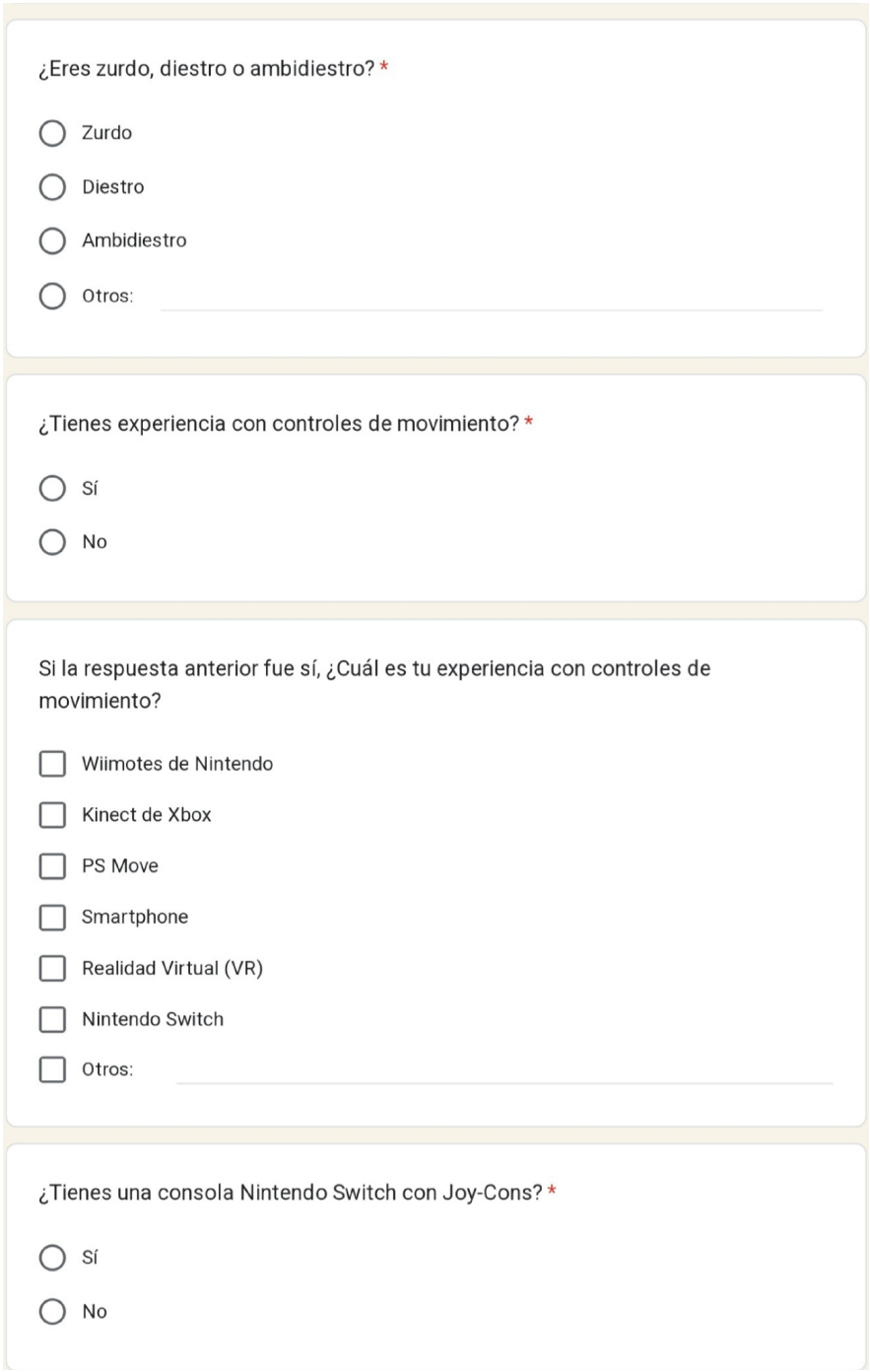


Figura 7.10: Preguntas de Entrada con Gesture Coding - Parte 2

Figura 7.11: Preguntas de Entrada con Gesture Coding - Parte 3

Figura 7.12: Preguntas de Entrada con Scratch - Parte 1

¡Última parte!

En esta última sección te preguntaremos sobre tu experiencia personal con la computación. ¡Responde de la manera más honesta posible!

En esta última sección te preguntaremos sobre tu experiencia personal con la computación. ¡Responde de la manera más honesta posible! *

	Estoy totalmente de acuerdo	Estoy de acuerdo	No estoy seguro	Estoy en desacuerdo	Estoy totalmente en desacuerdo
Creo que gracias a la computación puedo resolver problemas de mi vida diaria.	○	○	○	○	○
Los hombres y las mujeres pueden ser igual de buenos en computación.	○	○	○	○	○
Cuando uso un computador, me siento seguro/a de lo que estoy haciendo.	○	○	○	○	○
Creo que es valioso para mí saber computación.	○	○	○	○	○

Figura 7.13: Preguntas de Entrada con Scratch - Parte 2

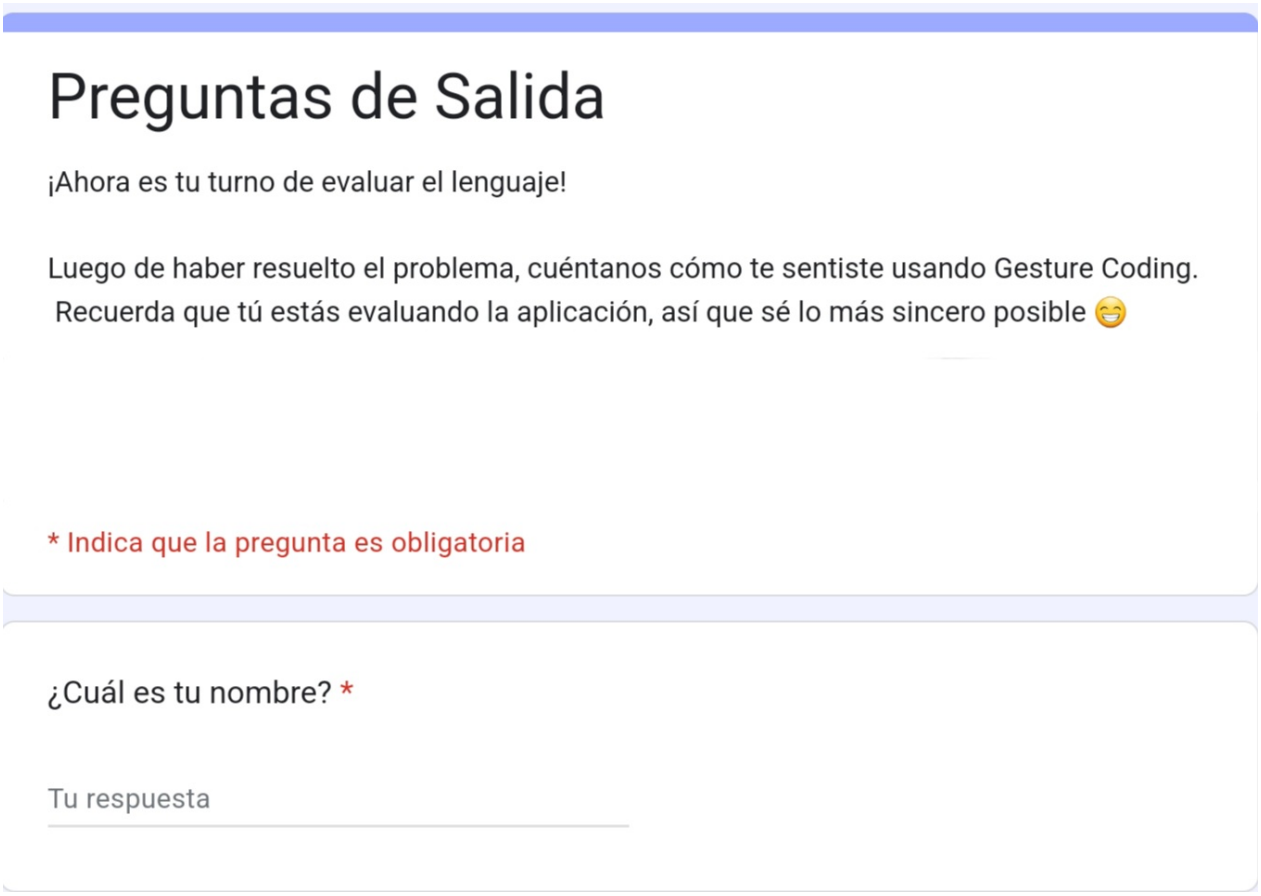


Figura 7.14: Preguntas de Salida con Gesture Coding - Parte 1

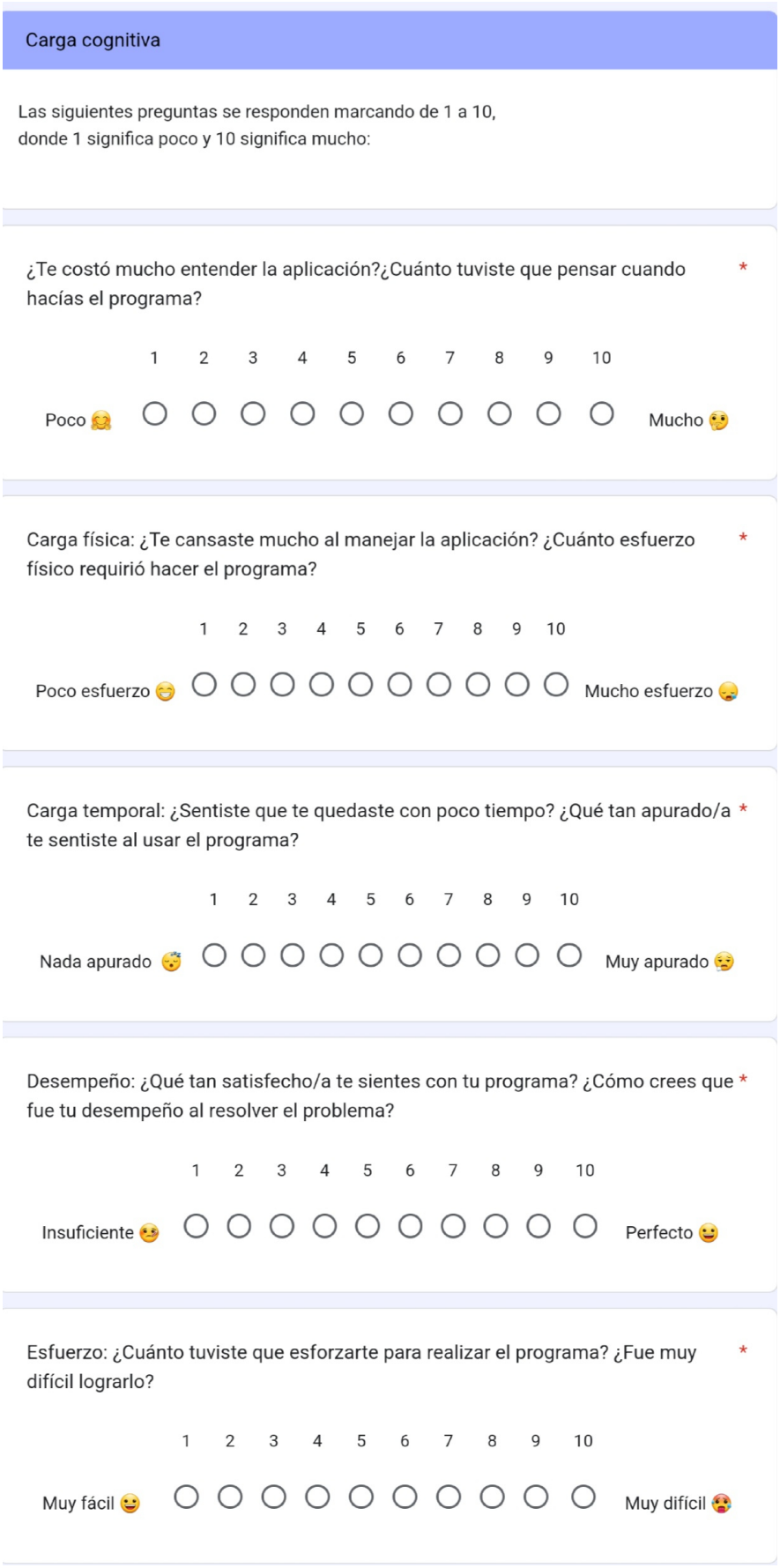


Figura 7.15: Preguntas de Salida con Gesture Coding - Parte 2

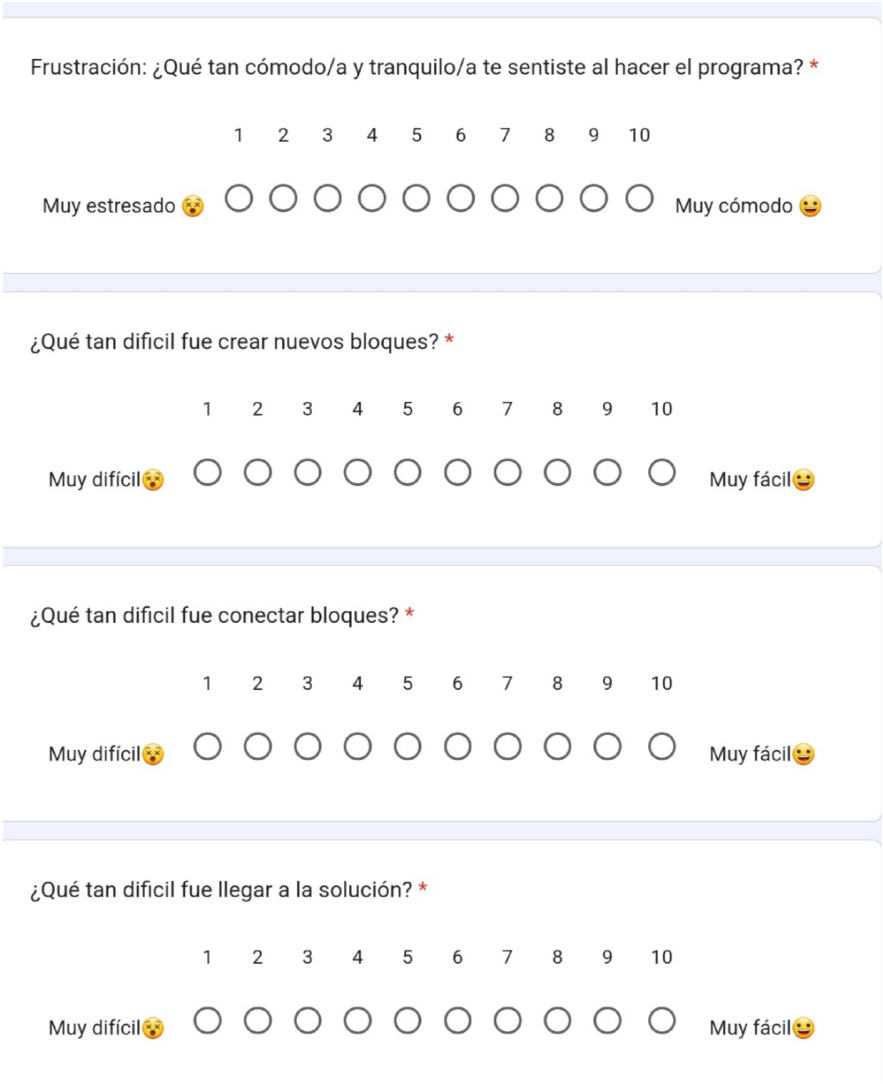


Figura 7.16: Preguntas de Salida con Gesture Coding - Parte 3

¡Última parte!

En esta última sección te preguntaremos sobre tu experiencia personal con la computación. ¡Responde de la manera más honesta posible! *

	Estoy totalmente de acuerdo	Estoy de acuerdo	No estoy seguro	Estoy en desacuerdo	Estoy totalmente en desacuerdo
Creo que gracias a la computación puedo resolver problemas de mi vida diaria.	○	○	○	○	○
Los hombres y las mujeres pueden ser igual de buenos en computación.	○	○	○	○	○
Cuando uso un computador, me siento seguro/a de lo que estoy haciendo.	○	○	○	○	○
Creo que es valioso para mí saber computación.	○	○	○	○	○
Me sentí bien programando.	○	○	○	○	○
Logré superar el desafío de programación planteado.	○	○	○	○	○

Finalmente ¿Qué opinas de Gesture Coding? Cuéntanos cuál fue tu experiencia trabajando con esta nueva herramienta *

Tu respuesta

Figura 7.17: Preguntas de Salida con Gesture Coding - Parte 4

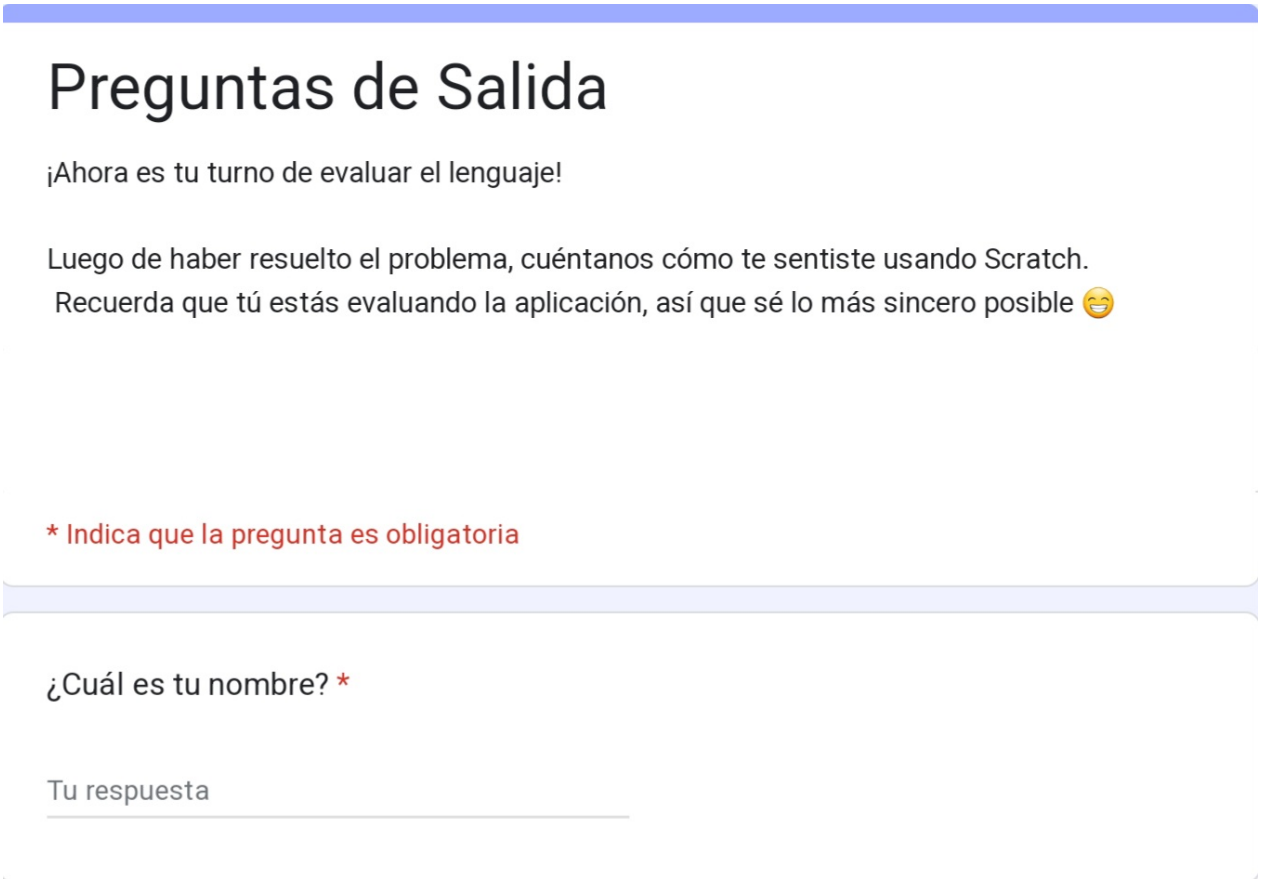


Figura 7.18: Preguntas de Salida con Scratch - Parte 1

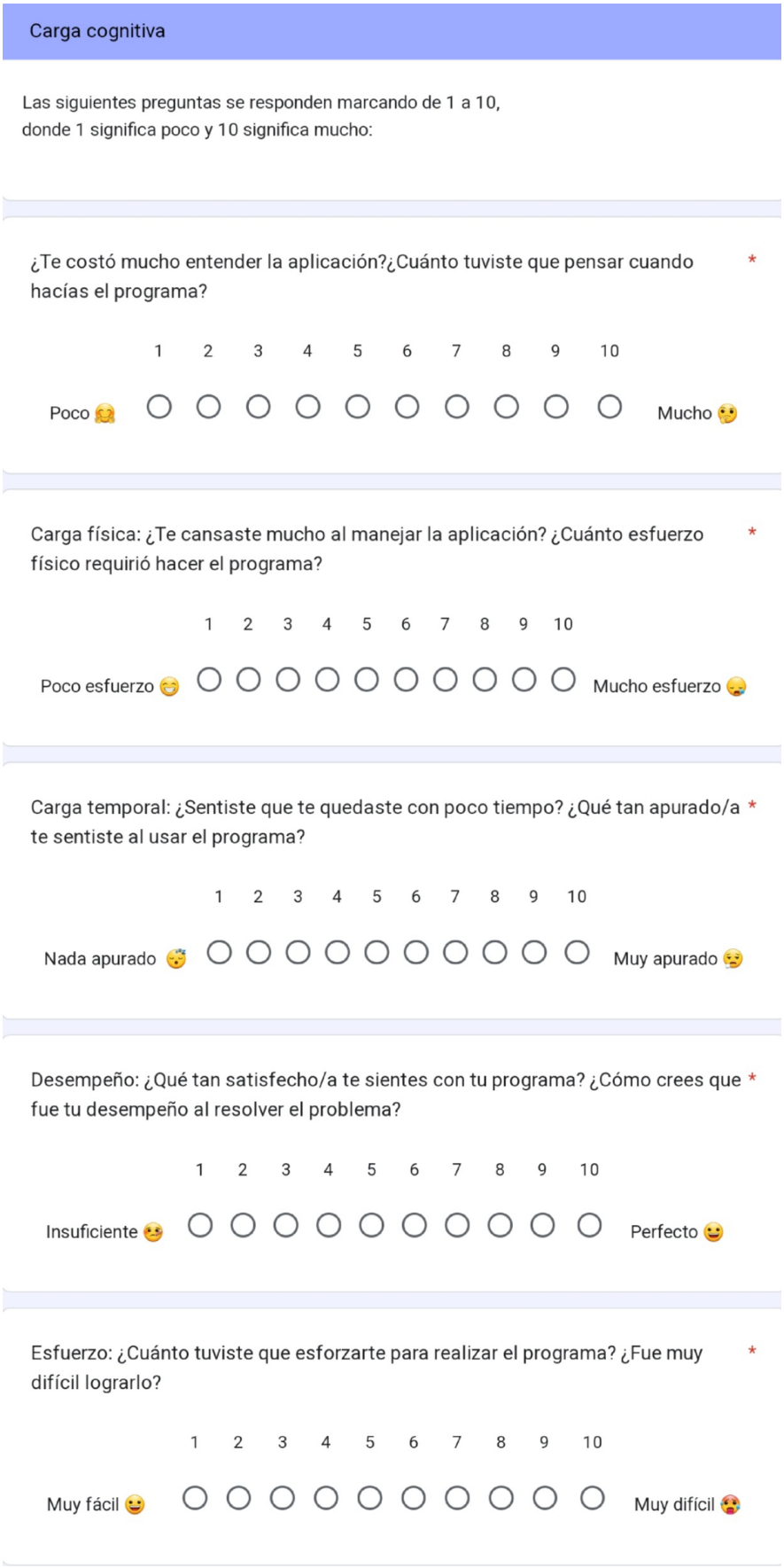


Figura 7.19: Preguntas de Salida con Scratch - Parte 2

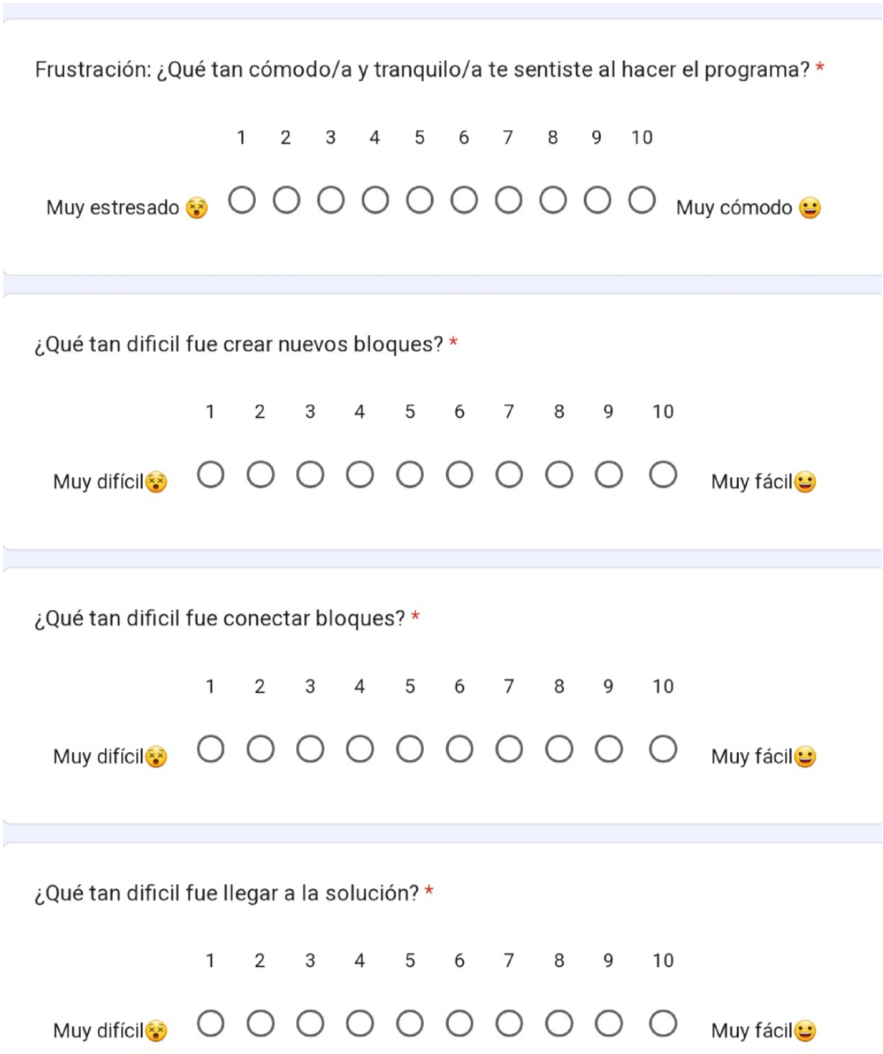


Figura 7.20: Preguntas de Salida con Scratch - Parte 3

¡Última parte!

En esta última sección te preguntaremos sobre tu experiencia personal con la computación. ¡Responde de la manera más honesta posible! *

	Estoy totalmente de acuerdo	Estoy de acuerdo	No estoy seguro	Estoy en desacuerdo	Estoy totalmente en desacuerdo
Creo que gracias a la computación puedo resolver problemas de mi vida diaria.	○	○	○	○	○
Los hombres y las mujeres pueden ser igual de buenos en computación.	○	○	○	○	○
Cuando uso un computador, me siento seguro/a de lo que estoy haciendo.	○	○	○	○	○
Creo que es valioso para mí saber computación.	○	○	○	○	○
Me sentí bien programando.	○	○	○	○	○
Logré superar el desafío de programación planteado.	○	○	○	○	○

Finalmente ¿Qué opinas de Scratch? Cuéntanos cuál fue tu experiencia trabajando con esta herramienta. *

Tu respuesta

Figura 7.21: Preguntas de Salida con Scratch - Parte 4

Figura 7.22: Preguntas de Extensión con Gesture Coding - Parte 1

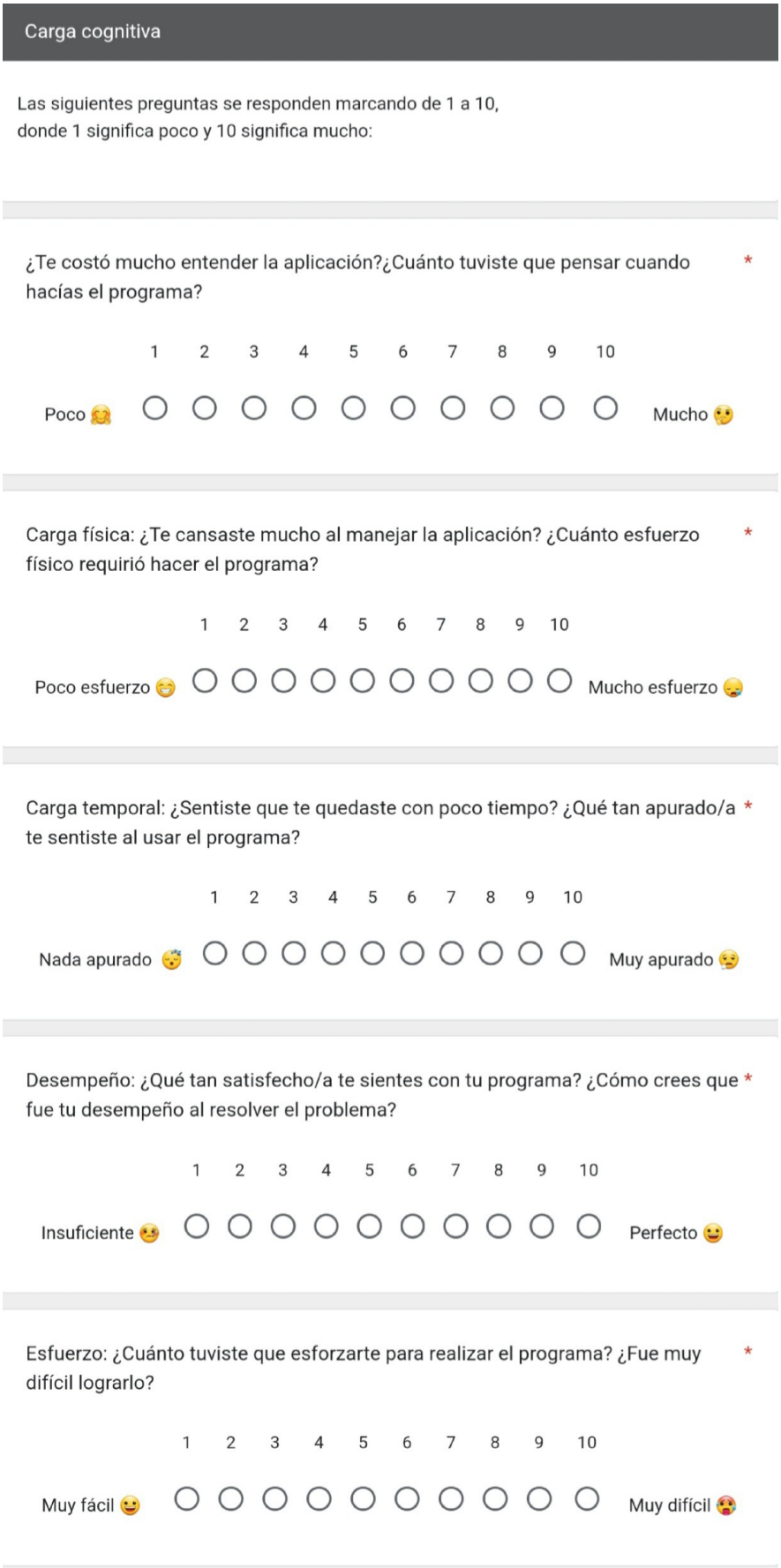


Figura 7.23: Preguntas de Extensión con Gesture Coding - Parte 2

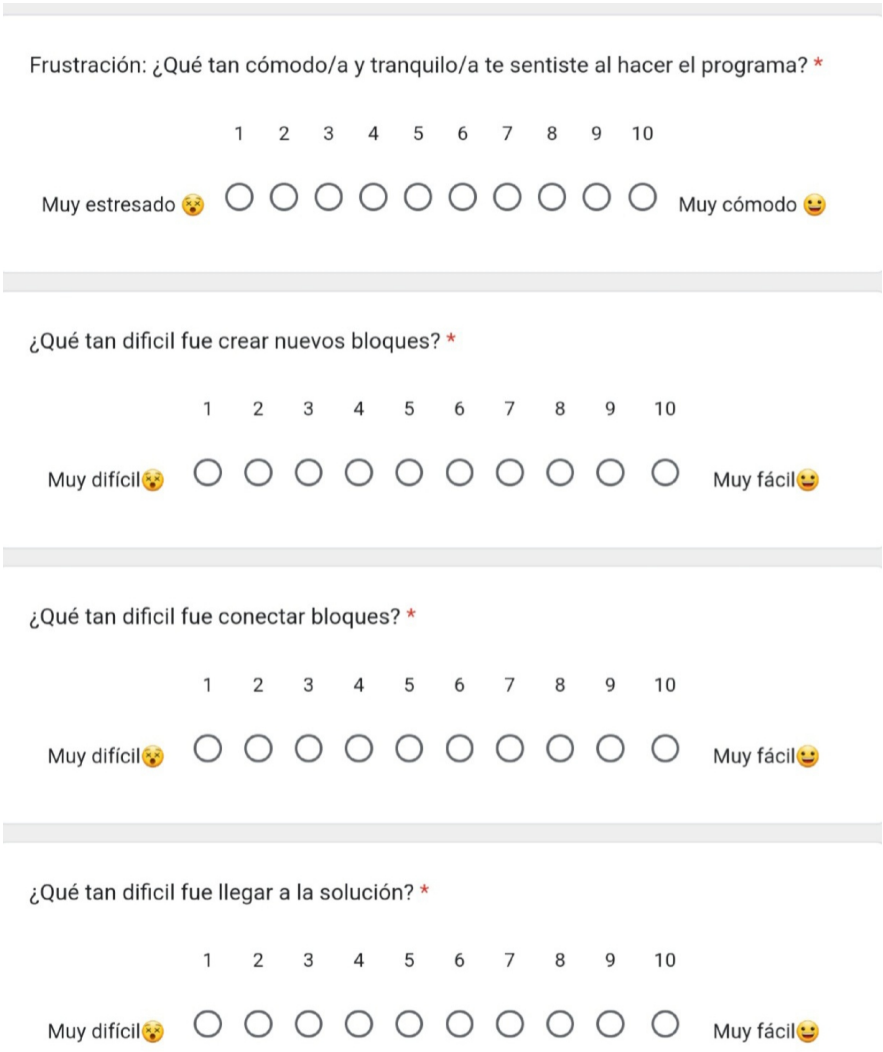


Figura 7.24: Preguntas de Extensión con Gesture Coding - Parte 3

Figura 7.25: Preguntas de Extensión con Gesture Coding - Parte 4

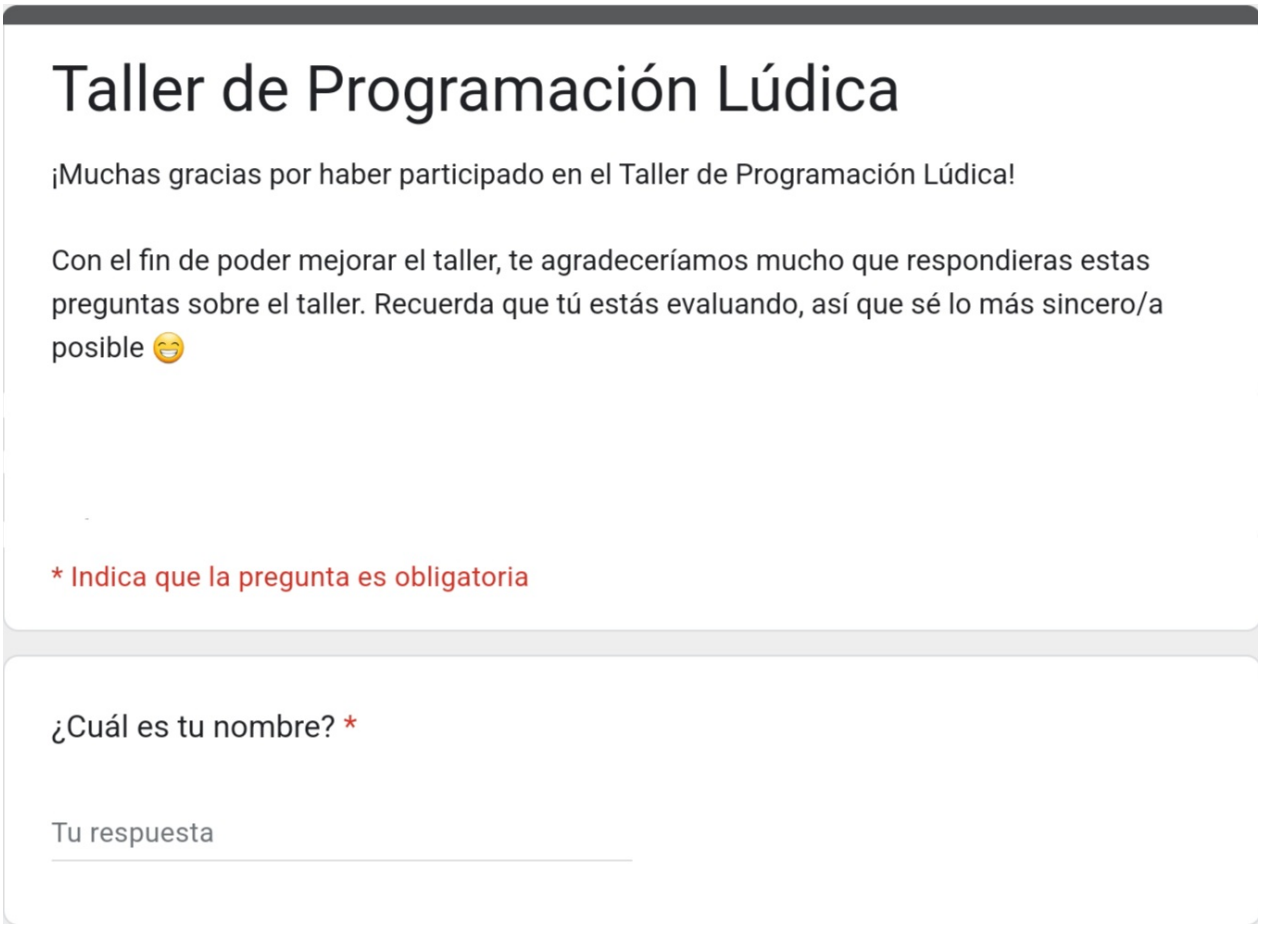


Figura 7.26: Preguntas de Extensión con Scratch - Parte 1

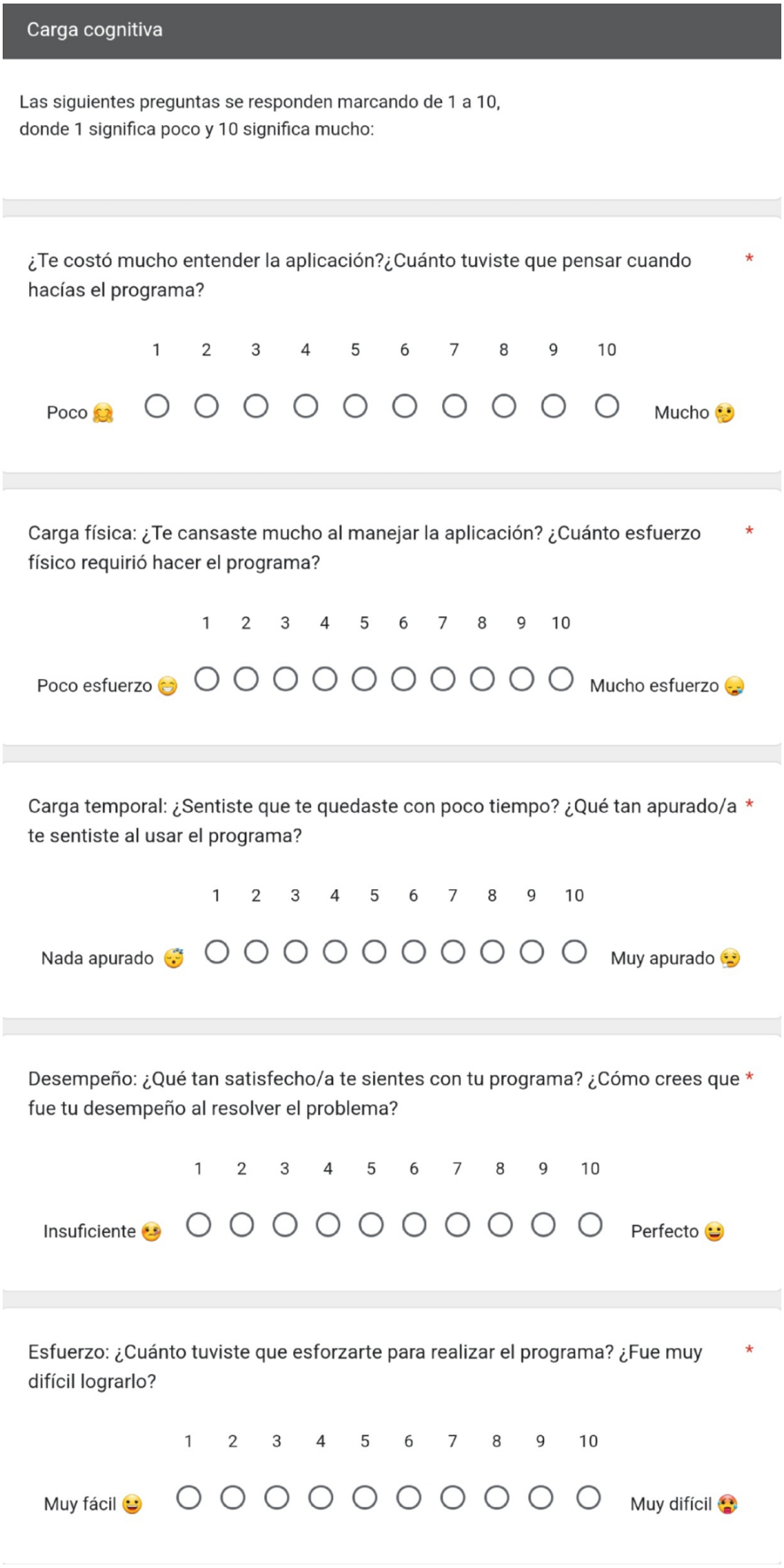


Figura 7.27: Preguntas de Extensión con Scratch - Parte 2

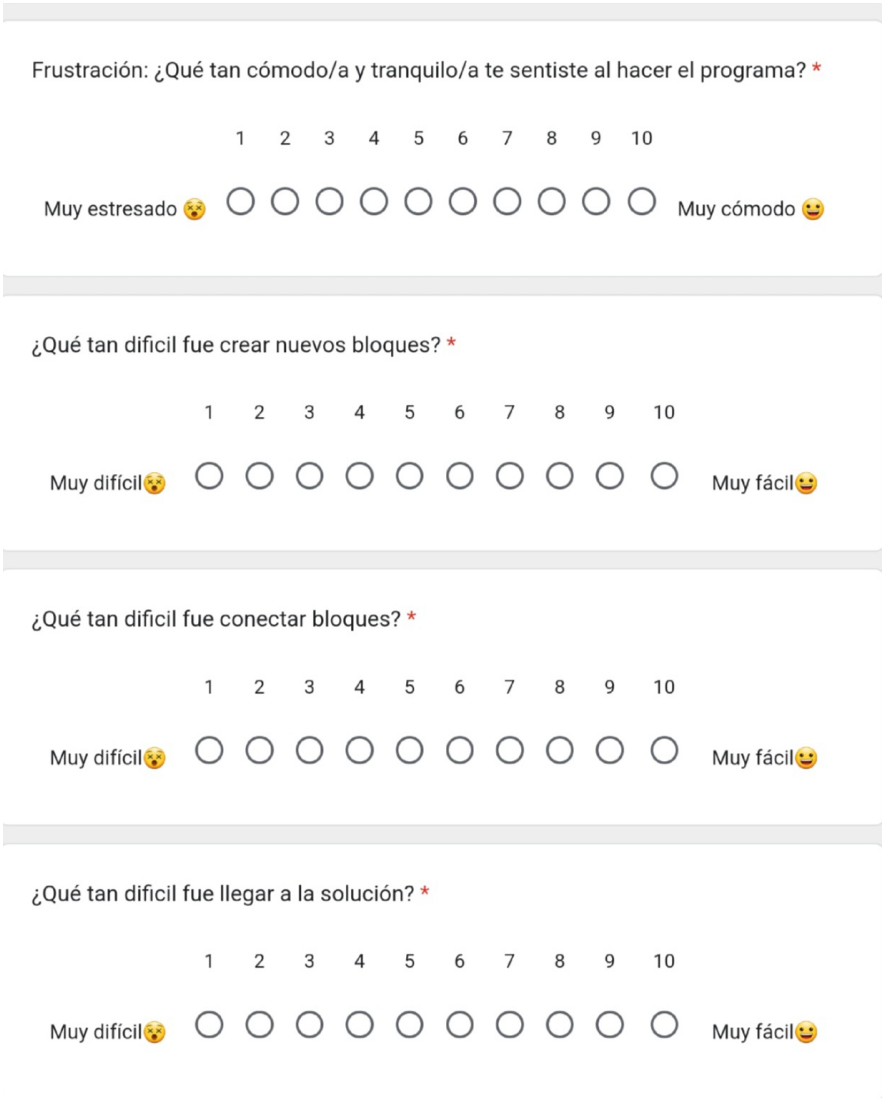


Figura 7.28: Preguntas de Extension con Scratch - Parte 3

¡Última parte!

En esta última sección te preguntaremos sobre tu experiencia personal con la computación. ¡Responde de la manera más honesta posible! *

	Estoy totalmente de acuerdo	Estoy de acuerdo	No estoy seguro	Estoy en desacuerdo	Estoy totalmente en desacuerdo
Creo que gracias a la computación puedo resolver problemas de mi vida diaria.	○	○	○	○	○
Los hombres y las mujeres pueden ser igual de buenos en computación.	○	○	○	○	○
Cuando uso un computador, me siento seguro/a de lo que estoy haciendo.	○	○	○	○	○
Creo que es valioso para mí saber computación.	○	○	○	○	○
Me sentí bien programando.	○	○	○	○	○
Logré superar el desafío de programación planteado.	○	○	○	○	○

Finalmente ¿Qué opinas de Scratch? Cuéntanos cuál fue tu experiencia trabajando con esta herramienta. *

Tu respuesta

Figura 7.29: Preguntas de Extensión con Scratch - Parte 4