



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

IGASCE: UN VIDEOJUEGO EDUCATIVO PARA ENSEÑAR ALGORITMOS  
RELACIONADOS A GRAFOS A ESTUDIANTES DE CIENCIAS DE LA  
COMPUTACIÓN

TESIS PARA OPTAR AL GRADO DE  
MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL EN COMPUTACIÓN

ALONSO UTRERAS MIRANDA

PROFESOR GUÍA:  
IVÁN SIPIRÁN MENDOZA

MIEMBROS DE LA COMISIÓN:  
NOMBRE COMPLETO UNO  
NOMBRE COMPLETO DOS  
NOMBRE COMPLETO TRES

SANTIAGO DE CHILE  
2023

# Resumen

*A mi abuelo, que siempre me apoyó en mis estudios.*

# Agradecimientos

Gracias a Noemí por acompañarme en todo este proceso, de inicio a fin. A mi familia por su apoyo incondicional en todos los niveles.

A mí profesor guía y maestro, Iván Sipirán, quien siempre me recibió con una sonrisa, una conversación muy amena y dando excelentes consejos.

A mis profesores por su ayuda, por ayudarme a crecer no solo como profesional y como estudiante, sino también como persona. No sería quien soy de no ser por ustedes.

Gracias a mis amistades, por su apoyo, pero también a quienes me pidieron apoyo, porque en la ayuda mutua es donde surgen las mejores ideas. Gracias Alfonso, Gabriel, Ricardo, Felipes, Jeremy, Nancy, Isa, Enri, Fernanda, Dani, Nico, Mati, Martín, Bea, Tommy. Se pasaron, sin ustedes no estaría escribiendo la tesis.

Gracias a mis colegas, que me ayudaron sin esperar nada a cambio. Me dieron excelentes guías que jamás se me hubieran ocurrido.

Gracias a la Facultad por brindarme el espacio de trabajo, donde sufrí, disfruté, lloré de tristeza y felicidad más de una vez. Por permitirme trabajar de lunes a domingo, de sol a sol. Por poder pasar la noche aquí y vivir en un ambiente de trabajo propicio.

# Tabla de Contenido

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	1
1.2. Objetivo General . . . . .	3
1.3. Objetivos Específicos . . . . .	3
<b>2. Trabajo Relacionado</b>	<b>4</b>
<b>3. Diseño de la Solución</b>	<b>6</b>
3.1. Referentes . . . . .	6
3.2. Objetivo del diseño deseado . . . . .	6
3.3. Descripción de la aplicación al momento de la realización de los experimentos	7
3.3.1. Diagrama de flujo de juego . . . . .	8
3.3.2. Narrativa . . . . .	8
3.3.3. Menú principal . . . . .	9
3.3.4. Tutoriales . . . . .	9
3.3.5. Niveles jugables . . . . .	11
3.4. Proceso de diseño . . . . .	12
3.4.1. Diseño de la interfaz de usuario . . . . .	14
3.4.2. Diseño de efectos de sonido y música . . . . .	15
3.4.3. Diseño de mecánicas de juego . . . . .	15
3.5. Arquitectura de software . . . . .	16
3.5.1. Arquitectura de una aplicación en Godot . . . . .	16

3.5.2. Arquitectura de software de la aplicación IGASCE . . . . .	17
<b>4. Diseño experimental: Metodología</b>	<b>21</b>
4.1. Fases del trabajo de investigación . . . . .	21
4.1.1. Fase exploratoria . . . . .	21
4.1.2. Fase de evaluación académica y percepción de usuario . . . . .	22
4.1.3. Trabajo y encuesta libre . . . . .	22
<b>5. Resultados</b>	<b>23</b>
5.0.1. Prueba académica . . . . .	23
5.0.2. Formulario MEEGA+: Percepción de usuario . . . . .	23
<b>6. Discusión</b>	<b>24</b>
6.1. Fortalezas de la metodología aplicada . . . . .	24
6.2. Debilidades y mejoras para la metodología aplicada . . . . .	24
6.3. Trabajo futuro . . . . .	25
<b>7. Conclusión</b>	<b>27</b>
<b>Bibliografía</b>	<b>30</b>
<b>Apéndice A. Anexo A: Formulario basado en MEEGA+</b>	<b>31</b>
<b>Apéndice B. Anexo B: Prueba de conocimiento de grafos</b>	<b>33</b>
<b>Apéndice C. Anexo C: Respuestas de pruebas de usuario</b>	<b>35</b>
<b>Apéndice D. Anexo D: Aprobación de Comité de Ética</b>	<b>36</b>
<b>Apéndice E. Anexo E: Consentimiento de participación voluntaria</b>	<b>39</b>

# Capítulo 1

## Introducción

Esta investigación tiene como objetivo convertir las instrucciones de los algoritmos de grafos en una representación visual e interactiva con forma de videojuego. Esta representación será utilizada por estudiantes de computación para aprender, permitiéndoles determinar si el uso de herramientas interactivas con feedback visual mejora tanto su aprendizaje como su motivación.

El trabajo consiste en presentar a estudiantes de ciencias de la computación un videojuego donde se muestran grafos. En este juego, el usuario debe ejecutar las instrucciones de estos algoritmos con la asistencia de elementos visuales y auditivos.

El objetivo de esta investigación de tesis es determinar si se ven diferencias en los niveles de motivación percibidos por el estudiantado y en el entendimiento de los algoritmos, medido a través de una prueba donde se pregunta por algoritmos de grafos.

El público objetivo son estudiantes que estén en su primer año de ciencias de la computación (CS por sus siglas en inglés), aunque también es aplicable a estudiantes de otras carreras relacionadas a ingeniería que sepan de programación. El requisito principal es que no hayan aprendido de grafos todavía.

### 1.1. Motivación

El uso de tecnologías digitales ha permitido acelerar el acceso al conocimiento y su escabilidad, permitiendo que estudiantes que habrían sido excluidos en otros contextos, ahora puedan acceder a la educación. Sin embargo, la educación en línea tiene sus propios retos [25]. En este contexto, se vuelve importante buscar metodologías que permitan acelerar el aprendizaje de los estudiantes con tecnologías adaptables, escalables, y que permitan un aprendizaje más eficiente.

Se postula popularmente que la capacidad de atención de forma prolongada ha disminuido en las nuevas generaciones. Hay estudios que contradicen estas afirmaciones [23], indicando que las habilidades cognitivas de los estudiantes han cambiado, pero no necesariamente em-

peorado. Existen términos como “doomsters” y “boosters” [32], para describir la polarización entre estas miradas con respecto a la tecnología.

En lo que sí existe consenso, es que la tecnología y su portabilidad ha incrementado la cantidad de distracciones a las que se exponen los estudiantes [40, 33]. A intentar realizar más de una tarea a la vez se le llama multitasking, y se ha demostrado que disminuye la productividad y el aprendizaje, aunque las personas creen que pueden hacer más de una cosa a la vez [9]. Con la popularización de los smartphones y las clases en línea, el multitasking -desde hacer labores del hogar hasta sacar el celular y revisar alguna aplicación- durante clases se ha incrementado [33].

Una forma de distracción reconocida en la literatura es la interferencia motivacional, propuesta y explicada por Fries y Dietz [13]. Esta señala que la motivación respecto a la clase disminuye debido a la presencia de tentaciones más atractivas para la atención, como los smartphones. En este contexto, donde los estudiantes se ven tentados a distraerse, es importante buscar formas de mantenerlos motivados y enfocados durante las clases. Es aquí donde se proponen utilizar videojuegos educativos como una manera de mantener a los estudiantes motivados y enfocados en la tarea que se les pide realizar.

Los videojuegos educativos destacan porque tienen potencial como una herramienta complementaria para la enseñanza, evaluación y entretenimiento para los estudiantes. Entre sus beneficios se destaca la motivación. En [4] se indica que para disfrutar una actividad, primero se debe permitir a la mente de un individuo percibir tal actividad como motivante.

Yu hace una revisión sistemática de la literatura [38] sobre los efectos de los videojuegos educativos en el aprendizaje de los estudiantes y su motivación. En este estudio, relativo a la motivación, se señala que en un contexto de aprendizaje que utilice o se complemente con videojuegos educativos afecta positivamente la motivación e incluso los logros académicos, pero también señala que han habido estudios que contradicen esta afirmación, por lo que se requiere más investigación en el área.

En el estudio mencionado, se asevera que el diseño del videojuego afecta totalmente el resultado final. Por ejemplo, los juegos de acción o realidad aumentada tenían mejores resultados que un juego tradicional, y que las mecánicas, elementos visuales y narrativos tienen un efecto significativo en el resultado final.

Considerando estos antecedentes, de que un videojuego educativo puede enseñar, es escalable, repetible y flexible, es que se presenta una oportunidad para analizar la percepción de estudiantes de computación con respecto a un videojuego educativo que enseñe algoritmos relacionados a grafos.

En la mayoría de los estudios citados previamente se indica que falta tener certeza respecto de la percepción de los usuarios al jugar videojuegos educativos y que se requiere indagar más. Por lo mismo, resulta conveniente emplear una prueba estandarizada para probar distintos diseños de videojuegos, con distintas poblaciones objetivo, pero cuyo resultado sea medido con el mismo estándar. Por esta razón, se utilizó un formulario basado en el modelo MEEGA+ [26] para medir la motivación de los estudiantes, y una prueba de conocimiento para medir el aprendizaje de los estudiantes.

## **1.2. Objetivo General**

El objetivo principal de este trabajo es medir el aprendizaje y la motivación de los estudiantes de computación al utilizar un videojuego educativo para enseñar algoritmos relacionados a grafos con las características presentadas en este trabajo.

## **1.3. Objetivos Específicos**

- Diseñar una aplicación interactiva que muestre grafos y permita seguir los pasos relacionados a algoritmos que trabajen con grafos.
- Aplicar una prueba estandarizada para medir la percepción de los estudiantes sobre el videojuego creado.
- Proponer, diseñar e implementar una mecánica de juego y arquitectura de programación que permita ser transferida a otros videojuegos que enseñen materias relacionadas a la programación.

# Capítulo 2

## Trabajo Relacionado

Un videojuego es una forma de aprendizaje activo, pues el proceso de enseñanza no se desarrolla partiendo por un profesor exponiendo frente a un estudiante. En este caso, quien aprende debe ejecutar pasos y participar en alguna actividad a través de la cual se construye el conocimiento. En la reseña realizada por Hartikainen et al. [19] se enumeran justificaciones para el aprendizaje activo: mejores resultados, recomendaciones políticas y las nuevas demandas de la vida laboral actual, como capacidades de comunicación o descubrimiento por cuenta propia.

Los videojuegos serios (Serious Gaming) son una forma de aprendizaje activo. Un trabajo de Bell y Gibson [16] indican que los juegos educacionales son más efectivos que las clases, lecturas, videos y tareas. Se resume que el uso de juegos resulta en 1) mejor retención, 2) mejor conocimiento fáctico, 3) mejor conocimiento basado en habilidades y 4) mayor autoeficacia. Sin embargo, recomiendan que los juegos deben ser acompañados de otras formas de enseñanza, así como hacer actividades post juego donde se le pregunta al estudiantado cómo se relacionan los juegos a la materia.

Bell y Gibson [16] identificaron y clasificaron videosjuegos de Ciencias de la Computación (CS), considerando un total de 41 videojuegos. Uno de ellos, Map Coloring, se trata sobre grafos, tomando el tema de coloreo de grafos. Se realizó una búsqueda del juego a la fecha (2022), pero no se encontró ningún material al respecto.

Kiili y su equipo [22] analizaron el uso de videojuegos en enseñanza y evaluación en matemáticas a través de los títulos “Semideus” y ”Wuzzit Trouble”. A través de estos, llegaron a la conclusión de que es posible utilizar videojuegos para enseñar y evaluar al mismo tiempo. Además, caracterizaron estadísticamente las diferencias producidas por el uso de videojuegos entre resultados de un pre test y un post test.

En el trabajo realizado por Zhao y Shute [39] se enumeran ejemplos de videojuegos pensados para enseñar programación, tales como Wu’s Castle [10], CodeCombat [1], CodeSpell [11], MiniColon [3], tales ejemplos utilizan programación con texto. Sin embargo, también hay numerosos ejemplos que utilizan programación por bloques, como LightBot [18], Scratch [24], [2] y RoboBuilder [34], los cuales abstraen el trabajo de aprender una sintaxis relacionada a los lenguajes de programación.

Sin embargo, el impacto de estos videojuegos no ha sido evaluado en muchos casos. En los casos documentados, se cuenta con muestras pequeñas, además de que se trata principalmente de evaluaciones puramente cualitativas [39], [15].

Petri y otros [26] están de acuerdo en que no hay sistematización en la evaluación de videojuegos educativos. En efecto, hay juegos serios que ni siquiera se denominan o consideran como tales [16]. Por otra parte, no existe una forma estándar de evaluarlos, razón por la cual Petri et al. [26] crearon el modelo MEEGA+ (Model for the Evaluation of Educational Games and EGameFlow scale).

Entre las faltas mencionadas al momento de crear videojuegos, se mencionan las faltas de 1) Definición de un objetivo de evaluación; 2) Diseño de investigación; 3) Programa de medición; 4) Instrumentos de recolección de datos y 5) Métodos de análisis de datos. Un ejemplo de falta de sistematización: una práctica común al analizar estas herramientas son los comentarios informales por parte del estudiantado [26].

# Capítulo 3

## Diseño de la Solución

La aplicación creada, llamada IGACSE (Interactive Graph Algorithms for Computer Science Education) de ahora en adelante, es propuesta y diseñada en base a diversos requerimientos y necesidades de los estudiantes de computación.

### 3.1. Referentes

### 3.2. Objetivo del diseño deseado

En este apartado se justifican las decisiones de diseño que explican por qué se decidió la temática de grafos, cuáles son los elementos interactivos y la ubicación y responsividad de estos. Según Wiley en su libro sobre diseño interactivo [28], es relevante entender primero cuál es el problema que busca resolverse, qué usuarios se ven afectados por esto, qué características poseen y cómo podría solucionarse en base a prototipos.

Se tiene como supuesto de que muchas veces los estudiantes creen entender cómo funciona cierto código o algoritmo, pero no lo aplican paso a paso con papel y lápiz o en su imaginación propia. Cuando se asigna como ejercicio realizar el procedimiento siguiendo cada instrucción rigurosamente, los estudiantes no lo realizan, o lo hacen de forma incorrecta sin darse cuenta. Por ejemplo, en [41] se menciona que los estudiantes a menudo creen entender los contenidos, pero tienen conceptos erróneos sin darse cuenta.

En el trabajo de Zingaro et al. [41] se mencionan algunos ejemplos de malos entendidos por parte de los estudiantes, como errores al comprender los heaps y las formas en que pueden representarse o construirse.

Se revisaron canales educativos de YouTube como 3Blue1Brown [31], donde el propietario del canal, Grant Sanderson, habla repetidamente sobre cómo las visualizaciones ayudan a comprender la abstracción matemática subyacente.

Un requerimiento es que videojuego logrado debe evitar la mecanización por parte del

estudiante, obligándolo a revisar exhaustivamente cada paso relacionado con el algoritmo, de manera que no sé por sentado que se entiende el código o de que se comprendió a cabalidad.

Por otra parte, para enseñar conceptos relacionados con programación, lo ideal es buscar usuarios que sepan cómo funciona, pero que no tengan mucha experiencia y que no visualicen un código al nivel de entender cómo funciona instrucción por instrucción. Por lo mismo, se propone basarse en un modelo similar a los debuggers, donde se puede ver el estado de las variables en cada paso, así como la instrucción por ejecutarse y el resultado de la misma.

### **3.3. Descripción de la aplicación al momento de la realización de los experimentos**

El videojuego se separa en distintos niveles. Consta de un menú principal, tutoriales y niveles jugables. En el menú principal se puede seleccionar el nivel a jugar o un modo historia. En el modo historia, se juegan todos los niveles en orden, y se desbloquean a medida que se avanza.

En los tutoriales se enseñan los conceptos básicos de grafos, pero sin indicarle al usuario explícitamente qué es un grafo. Además, se enseñan conceptos de jugabilidad, cómo explorar o seleccionar un nodo, cómo navegar en el código ejecutando instrucciones, y cómo contestar preguntas del tipo Sí/No cuando el código tiene una instrucción que incluya un if.

Finalmente, se presentan los niveles jugables, que corresponden a los dos algoritmos que se buscan enseñar: BFS (Breadth First Search) y DFS (Depth First Search). En estos niveles no se presenta historia y hay menos ayuda para el usuario. Una vez terminados los niveles jugables, se mostrarán los créditos del juego.

El juego consiste en explorar planetas a través de rutas, lo que corresponde a una metáfora de los grafos, donde los planetas son nodos y los caminos entre ellos son aristas. El objetivo es rescatar pandas rojos que están esparcidos en los planetas. Para rescatarlos, hay que explorar todos los nodos en una galaxia. Cada galaxia corresponde a un nivel. En cada nivel, se enseña un algoritmo distinto.

El juego guía al usuario a través de instrucciones que aparecen al costado derecho de la pantalla. El jugador pasa a la siguiente instrucción apretando espacio. Cada vez que se pasa a la siguiente instrucción, ocurren efectos y animaciones que muestran qué está pasando en el código. Estas animaciones se ven reflejadas en el viewport del juego, donde se observa que a algún planeta se enfatiza, un camino se enfatiza o que una nave vuela hacia otro planeta.

Cada instrucción tiene una forma distinta de ser ejecutada, algunas requieren que el jugador haga click en un planeta, otras que haga click en un camino, otras instrucciones requieren contestar correctamente una pregunta del tipo Sí/No, y otras requieren que el jugador presione una tecla específica.

Al seguir las instrucciones en su totalidad, el jugador está repitiendo el algoritmo paso a paso, lo que le permite entender cómo funciona el algoritmo y cuál es su finalidad.

Las acciones del jugador pueden ser correspondidas con un sonido recompensante, además de permitirle avanzar con la siguiente instrucción. En caso de cometer un error, se le indicará al jugador, a través de una animación visual y un sonido. En caso de que el jugador no sepa qué hacer, en ciertos casos se mostrará un hint visual indicándole al jugador que debe presionar alguna tecla en específico o hacer click en algún planeta.

### 3.3.1. Diagrama de flujo de juego

El juego inicia mostrando el menú principal, donde se puede seleccionar el modo historia o los niveles jugables. Si se elige el modo historia, se mostrarán los tutoriales. Una vez terminados los tutoriales, se mostrará el primer nivel jugable, que es DFS. Si se eligen los niveles jugables, se mostrarán los niveles jugables, donde se puede seleccionar el nivel a jugar, BFS o DFS. Terminados los niveles BFS y DFS, se mostrarán los créditos.

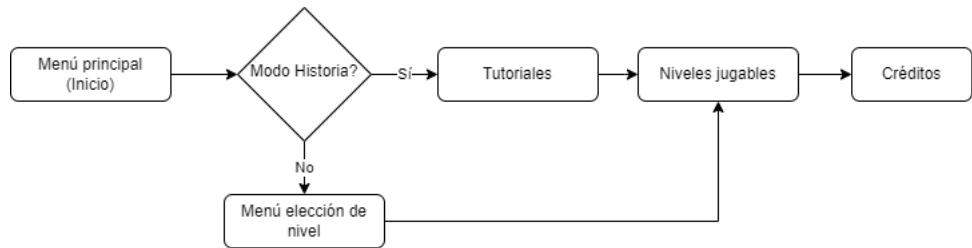


Figura 3.1: Flujo de juego de IGACSE

### 3.3.2. Narrativa

La historia se sitúa en el espacio exterior. Se muestra una nave en un inicio sosteniendo un diálogo con una estación llamada DCC. La nave está rescatando pandas rojos que están esparcidos en los planetas de la galaxia. El piloto indica que los niveles de combustible están bajos, a lo cual la estación le indica que debería seguir con las siguientes galaxias, pero sin utilizar el piloto automático, pues las redes neuronales que utilizadas son costosas y consumen combustible. A fin de ahorrar recursos, la nave deberá seguir instrucciones manuales para completar su misión.

Con cada galaxia visitada, el piloto rescata y encuentra otros pandas rojos. Se premia al jugador para que siga con los siguientes niveles y que termine el juego. Los pandas rojos son la mascota del Centro de Alumnos del Departamento de Ciencias de la Computación (CADCC). Cuando se realizan actividades de bienvenida a los nuevos estudiantes, se muestran afiches con esta mascota [5], por lo que se busca que el jugador se sienta identificado con el juego. Además, la estación se llama DCC para aumentar el sentido de identificación con el jugador, pues DCC son las siglas del Departamento de Ciencias de la Computación [8].

### 3.3.3. Menú principal

El menú principal permite seleccionar idioma apretando el botón en la esquina superior izquierda de la figura 3.2. Los idiomas disponibles son entre inglés y español. Además, proporciona las opciones para jugar el modo historia o los niveles jugables.

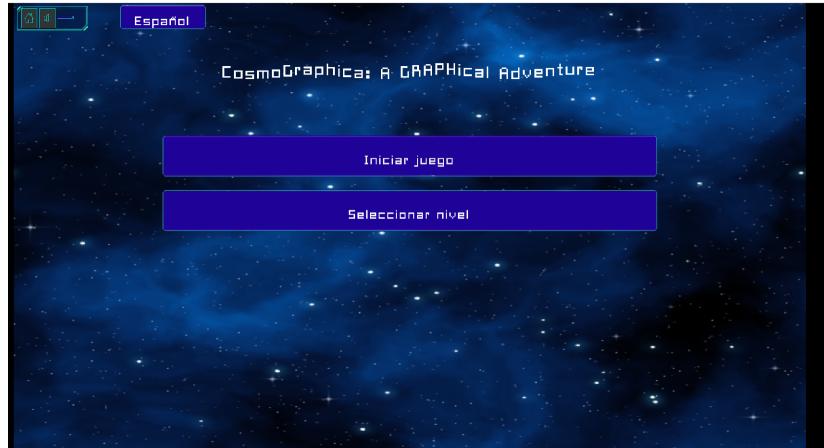


Figura 3.2: Menú principal de IGACSE, que se muestra al iniciar el juego.

### 3.3.4. Tutoriales

Los tutoriales buscan enseñarle al jugador las mecánicas básicas que serán utilizadas durante le resto del juego. Además, introducen al jugador a la historia, situada en una nave que está rescatando Pandas Rojos esparcidos en los planetas de la galaxia. La historia de fondo busca mostrar una aplicación de los grafos sin explicitar qué contenido se está enseñando. Las mecánicas se enseñan a través de una combinación de elementos, entre texto y hints visuales, como un mouse haciendo click izquierdo o una tecla R siendo presionada sobre el planeta. Cada nivel y tutorial es repetible para que el jugador pueda volver a probarlo en caso de no haber entendido algo.

#### Primer Tutorial

El primer tutorial busca enseñar la pantalla y que los planetas son navegables. Que hay caminos que conectan los planetas y se puede hacer click entre ellos. En esta instancia también se enseña sobre el diálogo y se muestran las pistas visuales

#### Segundo tutorial

El segundo tutorial introduce a la mecánica de instrucciones. Además, se introduce al ciclo for y al if en las instrucciones, que serán utilizadas posteriormente.



Figura 3.3: Primer tutorial. Se le indica al jugador a través de un diálogo que debe visitar cada planeta.

Al inicio del segundo tutorial, se le indica al jugador que tendrá que seguir las instrucciones del algoritmo, porque la exploración automatizada de pandas rojos es costosa y el combustible se está acabando. Las primeras dos instrucciones le piden al jugador apretar la tecla espacio para entender que al apretar espacio, se avanza a la siguiente instrucción (Ver figura 3.4).



Figura 3.4: Segundo tutorial. Se le muestra un diálogo de entrada al jugador introduciéndole el nuevo concepto. Este es el primer momento en que se le muestran las instrucciones al jugador.

Durante este segundo tutorial se introduce la lógica de responder sí o no cuando aparece una instrucción con un if en su instrucción. El jugador debe responder correctamente para avanzar a la siguiente instrucción. Si responde incorrectamente, perderá y deberá reiniciar el nivel. En la figura 3.5 se muestra la ventana que le aparece al jugador al llegar a una instrucción con un if.

### Tercer tutorial

El tercer tutorial introduce al jugador el concepto de Pilas (Stacks) y Colas (Queues) como estructuras de datos para almacenar nodos y luego obtenerlos en órdenes distintos. Además, se le introduce al jugador sobre las variables creadas y la variable seleccionada, la cuak se usa para señalar a qué objeto se le agregará el nodo que el usuario presione.



Figura 3.5: Segundo tutorial. Ventana de if que le aparece al jugador al llegar a una instrucción con un if. El jugador debe responder sí o no correctamente para avanzar

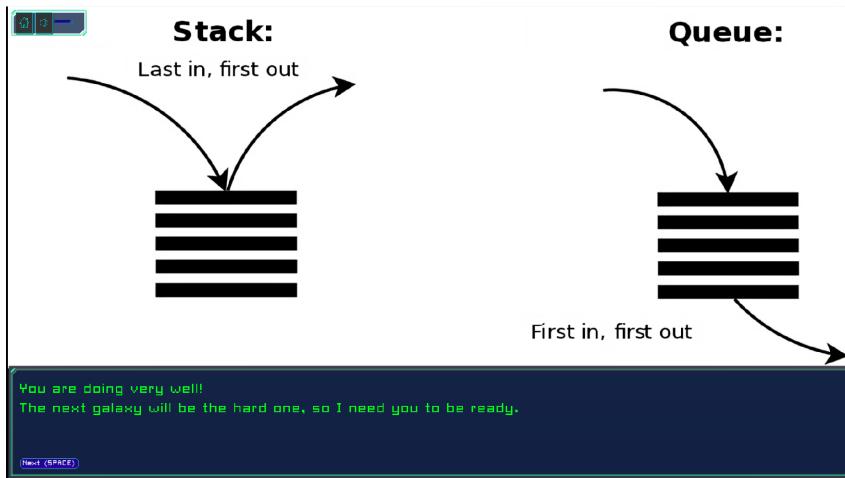


Figura 3.6: Tercer tutorial. Se le explica al usuario al inicio qué son las colas y stacks y cómo funcionan y en qué difieren.

El objetivo de las instrucciones en este tutorial es que el jugador aprenda la mecánica para agregar nodos a un stack o a una pila. Además, se refuerza la mecánica de las instrucciones y ejecutar cada acción paso por paso.

### 3.3.5. Niveles jugables

Los niveles jugables presentados son BFS y DFS, que enseñan esos algoritmos. Los planetas y sus conexiones son aleatorias, con la restricción de que el grafo formado es siempre conexo, es decir, todos sus nodos tienen al menos un camino hacia el resto de los otros nodos.

Se crearon 4 niveles para el juego, incluyendo los algoritmos de Kruskal y Prim. Sin embargo, estos últimos no fueron incluidos en la versión final del juego, pues el diseño y la



Figura 3.7: Tercer tutorial. Mostrando al usuario cómo funciona un stack a través de animaciones.

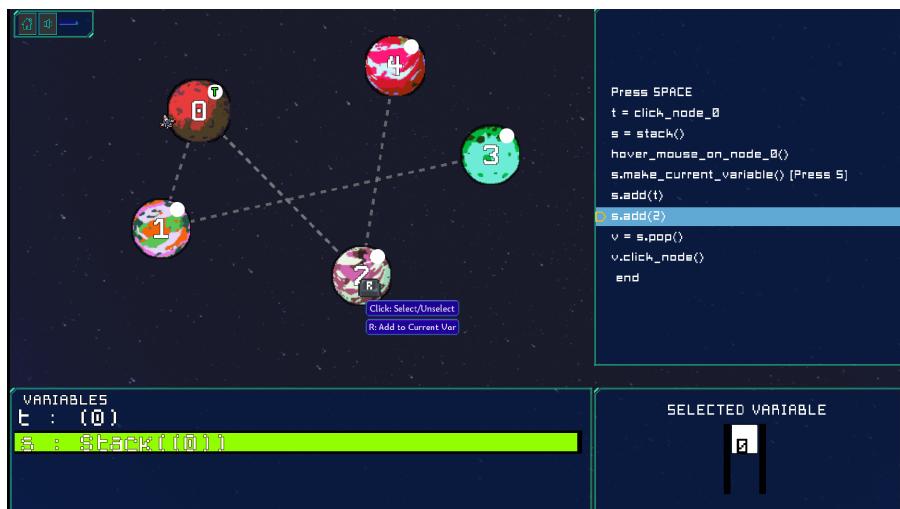


Figura 3.8: Tercer tutorial. El jugador debe presionar R sobre el planeta 2 para avanzar a la siguiente instrucción.

experiencia de usuario eran distintos por requerir otro tipo de acciones, como ordenar arcos y operar con conjuntos, obteniendo uniones e intersecciones. Además, el tiempo de la prueba de usuario era aproximadamente de 45 minutos, por lo que se prefirió no incluirlo en la prueba dado que no se contaba con el tiempo suficiente para probarlos.

### 3.4. Proceso de diseño

Para diseñar el videojuego, primero se determinaron problemas que se querían solucionar. En particular, el autor del trabajo hipotetiza que los estudiantes de computación no entienden a cabalidad los algoritmos que se enseñan en los cursos de programación, y que no se dan cuenta de que no los entienden. Por lo mismo, se busca que el videojuego sea una herramienta que permita a los estudiantes entender los algoritmos paso a paso, de manera que no se den

cuenta de que no los entienden.

En la Universidad de Chile, históricamente y al momento de realizar este trabajo, para entrar a computación se requiere aprobar anteriormente dos años de cursos de plan común, donde existe una práctica conocida como “pauteo”, la cual consiste en estudiar y revisar pautas de pruebas de semestres pasados para prepararse para los exámenes. Con esta práctica, el alumnado revisa las ecuaciones o procedimientos para responder una pregunta sin haber realizado los ejercicios o repetido el procedimiento paso por paso. Muchas veces revisan un algoritmo superficialmente, o lo reproducen mentalmente sin hacerlo en papel. Por lo mismo, se busca que el videojuego sea una herramienta que permita a los estudiantes entender los algoritmos paso a paso y acompañarlo con una visualización.

La idea principal es forzar al usuario a reproducir el algoritmo paso a paso. Esto se puede observar al utilizar el debugger. El debugger permite ver el estado de las variables en cada paso, y permite avanzar paso a paso en el código. El videojuego busca ser una herramienta que permita al usuario reproducir el algoritmo paso a paso, pero de una manera más lúdica. Por esto, el debugger que se utiliza en Visual Studio Code se utilizó como modelo.

Posteriormente, se realizaron distintos bocetos. Se le mostraron estos bocetos a un diseñador de videojuegos profesional con títulos ya liberados. Se eligió el que fue entendido a primera vista y que se veía más atractivo, además que se parecía más a Visual Studio Code. Se implementaron versiones del videojuego con Godot, que permitía agregar nuevas funcionalidades rápidamente, además de permitir cambiar el estilo. El autor del trabajo mostraba su trabajo de tesis semana a semana en un curso de 20 estudiantes, donde se recibían comentarios del resto de los alumnos.

Una vez definido cierto nivel de avance con todos los algoritmos definidos, se probó con 4 usuarios expertos que ya conocían el algoritmo. Sin embargo, no fueron capaces de seguirlo o comprenderlo a cabalidad, por lo que no se cumpliría el objetivo con personas neófitas en los grafos. Por esta razón, se decidió conseguir la asesoría directa de un diseñador de videojuegos y un diseñador de UX/UI.

Los diseñadores, en entrevistas separadas coincidieron en la necesidad de introducir tutoriales que mostraran los elementos del juego uno a uno, por lo que se decidió agregar tutoriales. Además, se decidió agregar una historia de fondo que permitiera al jugador sentirse identificado con el juego y comprendiera un uso inmediato de estos algoritmos y estructuras. Además, agregar una componente de premiación y gamificación aumentaría la motivación de los usuarios.

Una vez finalizados los tutoriales, agregados los estilos, música y elementos narrativos, se procedió a publicar el videojuego en la plataforma de itch.io IGASCE, una plataforma de videojuegos y elementos relacionados a videojuegos de creadores con bajos recursos o del mundo indie. El estilo se compró en itch.io a través de la tienda de assets <https://azagaya.itch.io/sci-fi-theme>. Los sonidos también se descargaron a partir de la misma tienda.

### 3.4.1. Diseño de la interfaz de usuario

La interfaz de usuario está basada principalmente en el debugger de Visual Studio Code (Ver figura 3.9). Se observa aquí que la instrucción actual está destacada por un puntero y en color. Además, las variables locales muestran su valor en un formato "nombre: valor". El usuario puede avanzar paso a paso en el código.

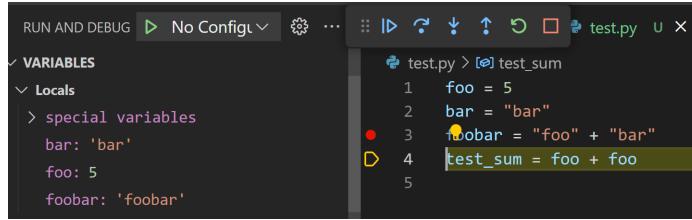


Figura 3.9: Visual Studio Code Debugger.

Hay otros juegos cuya interfaz sirvió de inspiración para la realización del juego final, como 7 Billion Humans, donde el código se ve a la derecha de la pantalla, y la pantalla con los elementos y eventos del juego está al lado izquierdo. Lo que ocurre en la interfaz de la derecha afecta al mundo.

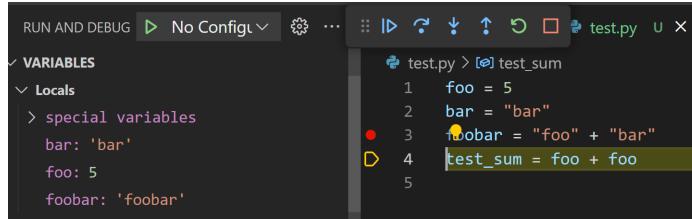


Figura 3.10: Videojuego: 7 Billion Humans

Otro videojuego con una interfaz similar es CodeCombat. Aquí el código también está a la derecha, mientras que abajo se ven diálogos y elementos seleccionables. En el caso de IGASCE, los diálogos también se muestran abajo, así como las variables actuales y la seleccionada. [1]. Además, el menú del juego se accede en la esquina superior izquierda (Ver figura 3.11).

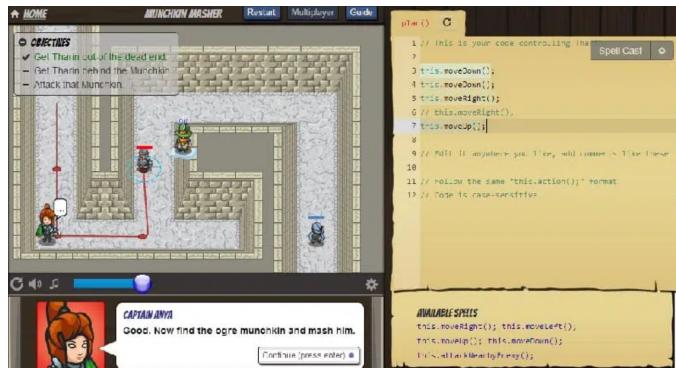


Figura 3.11: Videojuego: CodeCombat

### 3.4.2. Diseño de efectos de sonido y música

Los juegos retro suelen utilizar loops de música de 8 a 16 bits. Algunos ejemplos son: Golden Sun [7], ProtoCorgi [21], Pokemon Emerald [36], Space Invaders [37], o Final Fantasy IV [35].

En el caso de Pokemon [36], cada vez que se hace click en algún botón y se gatilla un evento, se emite un sonido de confirmación, como cuando el jugador selecciona un ataque. Lo mismo ocurre en Golden Sun [7]. El objetivo es confirmarle al usuario que se está ejecutando correctamente una acción. En el caso de IGASCE, se utilizan sonidos de confirmación cuando se hace click en un planeta o en un camino, o cuando se presiona una tecla correctamente, ya sea espacio o R. Los sonidos de confirmación en todos estos casos son agudos, para que el jugador no se distraiga con ellos, además de dar a entender que la acción tiene un efecto inmediato.

En el caso de los sonidos de error, se utilizan sonidos graves, con fade y que suelen dar la sensación de vibración (Ver figura 3.13), para que el jugador se dé cuenta que cometió un error y que debe corregirlo. En el caso de IGASCE, se utilizan sonidos de error cuando se hace click en un planeta o en un camino incorrecto, o cuando se presiona una tecla incorrecta, como apretar espacio cuando todavía no se finaliza una instrucción. Además, se utilizan sonidos de error cuando se contesta incorrectamente una pregunta del tipo Sí/No.



Figura 3.12: Espectro de Ondas del sonido de Confirmación de IGASCE

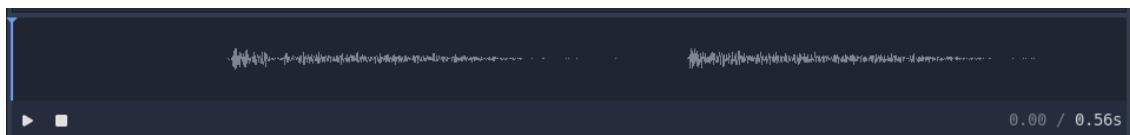


Figura 3.13: Espectro de Ondas del sonido de Error de IGASCE

### 3.4.3. Diseño de mecánicas de juego

La mecánica que determina si se gana un nivel o no es cumplir todas las instrucciones presentadas por el algoritmo. Para los niveles BFS y DFS, esto coincide con haber visitado todos los nodos del grafo. El puntero de instrucciones indica la instrucción actual. El jugador debe completar correctamente lo que pide la instrucción, ya sea hacer click, responder una pregunta o agregar un nodo a una estructura de datos, aunque hay instrucciones que no piden ejecutar nada por parte del jugador, como las instrucciones que contienen la palabra clave *for*. Una vez ejecutados correctamente los pasos exigidos, cambiará el color de la instrucción y el cursor de la interfaz se transformará en un ticket.

Para avanzar en las instrucciones, el jugador debe presionar la tecla espacio. Si se apreta la tecla espacio sin haber finalizado la instrucción, se emitirá un sonido de error y el color de la instrucción fluctuará entre rojo y amarillo por un segundo. Si se apreta la tecla espacio cuando se ha finalizado la instrucción, se emitirá un sonido de confirmación y se pasará a la siguiente instrucción. Si se apreta la tecla espacio cuando no hay más instrucciones, se emitirá un sonido de victoria y se permitirá pasar al siguiente nivel o los créditos según corresponda. Este proceso se resume en el diagrama 3.14.

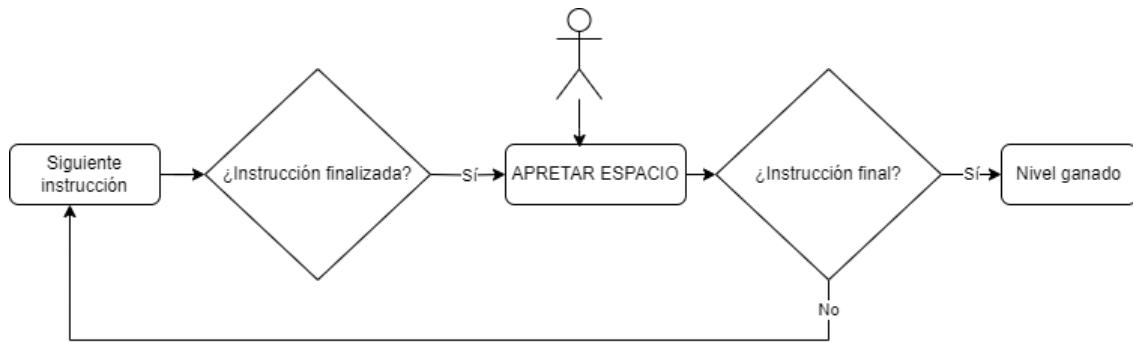


Figura 3.14: Flujo de mecánicas de Nivel. El jugador debe seguir las instrucciones paso a paso.

## 3.5. Arquitectura de software

Antes de explicar la arquitectura del programa en específico, es necesario explicar cómo se estructuran los programas hechos en Godot, pues esto justifica las decisiones de diseño tomadas durante el desarrollo de la aplicación.

### 3.5.1. Arquitectura de una aplicación en Godot

Una aplicación en Godot se construye en base a escenas. Una escena es un conjunto de objetos instanciados al mismo tiempo. Un programa creado en este motor se compone de escenas, que típicamente serán los niveles en un juego. Las escenas tienen un grafo de escena compuesto por uno o más nodos.

Los nodos en Godot son la unidad básica de construcción del programa. Un nodo puede representar un botón, una textura, un reproductor de sonido, áreas de colisión, entre otros. Para armar un personaje que funciona con varios niveles de lógica, tales como: habilidades, movimiento o colisiones, se deben componer más nodos.

La estructura del grafo de escena resulta conveniente en el desarrollo de videojuegos, porque basta mover el nodo padre que representa al personaje, para mover simultáneamente las colisiones, los sonidos y texturas asociadas al personaje. Esto facilita la manipulación de los elementos relacionados.

Otra gran ventaja de usar un motor de videojuegos como Godot es que permite utilizar variables exportadas. Estas son variables cuyo valor se puede definir desde el editor [6], lo que permite por ejemplo, instanciar objetos de la misma clase pero con distintas características.

En contraste, cuando se trabaja a un nivel más bajo utilizando bibliotecas como PyGame, que utiliza OpenGL, es necesario abordar cada nivel de representación de un objeto por separado. Por ejemplo, si se desea mover a un personaje con estas bibliotecas, se debe gestionar la textura y las colisiones de manera individual y separada en el código, lo que resulta en un aumento de los costos y la complejidad del desarrollo. En [17] se observa que para armar un personaje básico, se le agrega una componente de colisiones y una visual (Sprite2D) por separado.

### 3.5.2. Arquitectura de software de la aplicación IGASCE

La aplicación se desarrolla en diferentes fases, donde se pueden identificar 3 tipos de escenas: menús, tutoriales y niveles jugables. En cada fase, se utilizan módulos distintos, y cada nivel, menú o tutorial representa una escena.

Los menús se basan principalmente en nodos de control, que se utilizan típicamente para interfaces gráficas. En este caso, los menús consisten únicamente en botones que ejecutan códigos según su descripción, sujetos a ordenadores verticales o en forma de grilla (Ver figura 3.15).

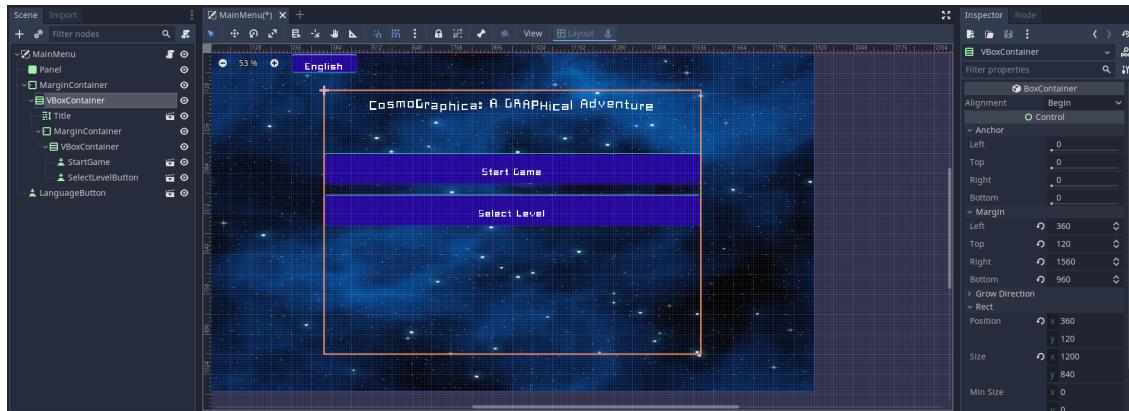


Figura 3.15: Interfaz de Godot describiendo la escena del menú principal de IGASCE. El juego públicamente se dio a conocer como CosmoGraphica, pero el proyecto completo se llama IGASCE.

Los tutoriales tienen una lógica más compleja y son similares a los niveles. La diferencia con las escenas finales es que los tutoriales incluyen una ventana de diálogo que busca conectar la historia del videojuego con los elementos interactivos, además de que el grafo ya está predeterminado, por lo que los nodos y arcos ya vienen predefinidos.

Respecto a los niveles jugables, existen diversos elementos destacables. Entre estos, se encuentran: El código, las variables, la variable seleccionada y la ventana de juego.

En relación al código que se muestra a la derecha de la pantalla durante el juego, este está

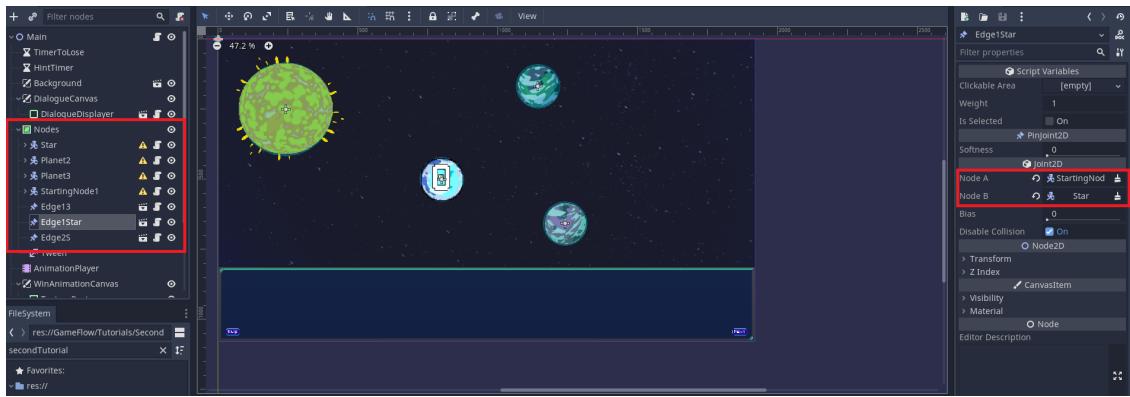


Figura 3.16: Interfaz de Godot describiendo el segundo tutorial. En rojo se observa el grafo de escena con los nodos y los arcos ya predefinidos a la izquierda. En la derecha se muestra cómo se unen los arcos, indicando los dos extremos del arco a través de variables exportadas de Godot.

controlado por un nodo de la clase CodeContainer. Este objeto se encarga de recibir el input del jugador para avanzar en el código. Si las acciones ejecutadas son correctas, el usuario pasa a la siguiente instrucción, desbloqueando nuevas tareas a realizar. La clase CodeContainer posee un arreglo de objetos del tipo CodeLine llamado code\_lines. La clase CodeLine contiene toda la lógica de una única línea de código. Entre sus funciones se encuentran: darle énfasis a la línea de código actual, darle feedback al usuario cuando la instrucción de la línea se completó correctamente y verificar constantemente si la instrucción de línea se ha completado correctamente (ver figura 3.17).

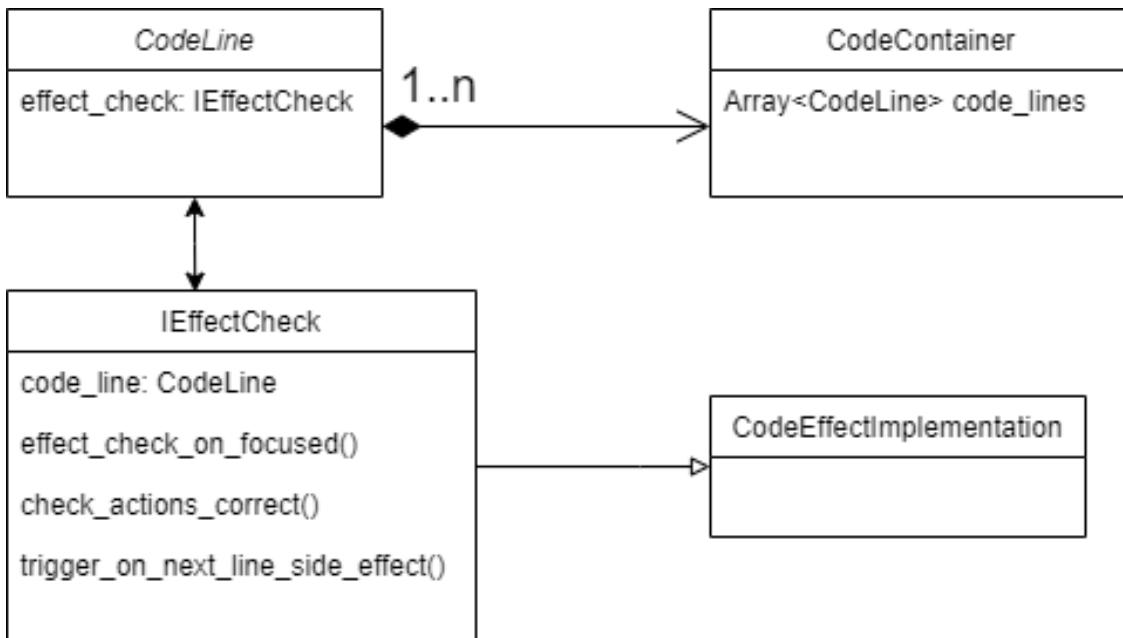


Figura 3.17: Arquitectura de la ejecución de código dentro del juego.

Cada CodeLine posee un script que se puede indicar desde el editor de Godot a través de una variable exportada. Este script debe heredar de la clase EffectCheck, la cual actúa como interfaz. Cada script que implemente la interfaz EffectCheck debe definir métodos para

cuando: 1) El puntero de instrucción llega a la línea del script. 2) Para verificar que la instrucción actual ha sido realizada correctamente. 3) Si corresponde, efectos colaterales de la instrucción. Este último se usa en los ciclos for, donde se mueve el índice utilizado para avanzar por los ciclos y recorrer arreglos. También se utiliza para la creación de variables.

En el panel inferior se encuentran dos clases: DebugBlock y ADTShower. El primero se encarga de mostrar las variables instanciadas, su nombre y su valor actual. Contiene la lógica para agregar, modificar o eliminar alguna variable. Cada vez que se pasa por un ciclo for o se declara una variable, este modifica sus variables internas. Existe una variable seleccionada, la cual puede ser manipulada por el usuario de distintas formas. Al apretar la flecha arriba se selecciona la variable que esté arriba de la actual y viceversa para la flecha de abajo. En la figura 3.18 se observa que está la variable “q” seleccionada.

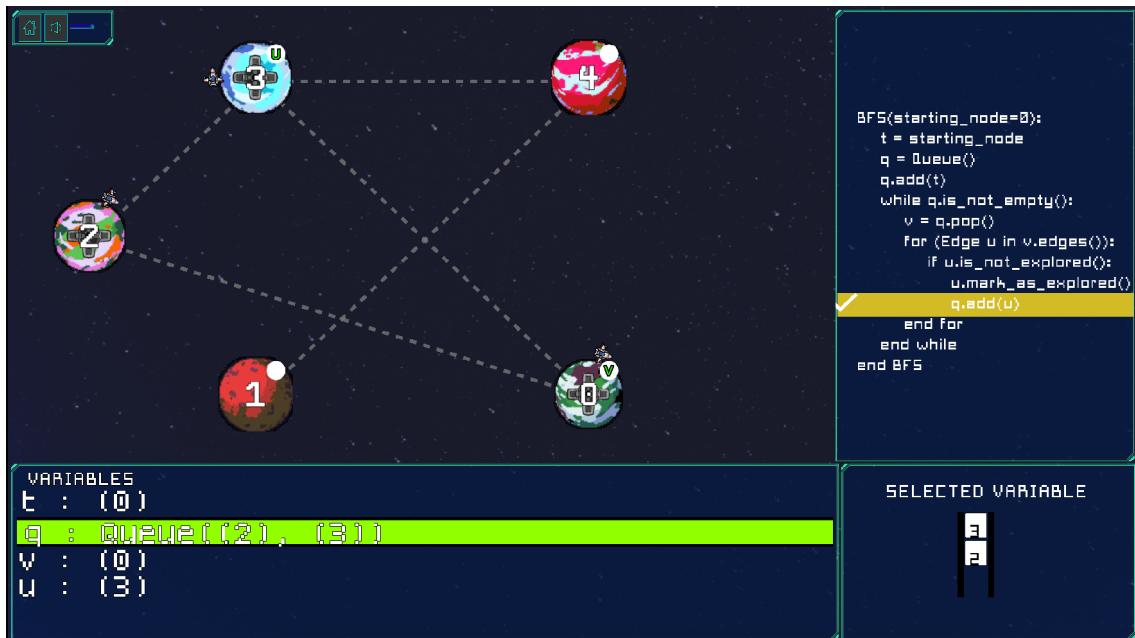


Figura 3.18: Nivel jugable BFS. Se observa el código a la derecha, el grafo en el centro, el DebugBlock en la parte inferior izquierda y el ADTShower en la esquina inferior derecha. La variable q está seleccionada.

La clase ADTShower se encarga de mostrar la variable seleccionada por el usuario en el DebugBlock. Este busca agregar una representación visual al objeto seleccionado, para permitir arrastrar nodos a la cola o la pila según el algoritmo que se esté enseñando, BFS o DFS, respectivamente.

Inicialmente, cada cambio en el código por el efecto de alguna CodeLine o línea de código del juego requería modificar las clases DebugBlock, ADTShower, StoredData (Explicada más adelante) y afectar la misma línea de código. Además, estos efectos debían traspasarse a las siguientes líneas de código para lograr cohesión. Por ejemplo, si la línea cinco creaba la variable u y la sexta la modificaba, esto se debía reflejar correctamente en todas las estructuras mencionadas.

Para solucionar el problema del alto costo de implementación, se optó por crear una clase llamada ADTMediator, de acuerdo al Mediator Pattern [12]. Esta clase se inspiró en la

arquitectura de React, donde el código generado por React funciona como una Fuente de toda verdad o ”Single Source of Truth”[27]. Con este tipo de arquitectura, se le envía un mensaje a ADTMediator solicitando modificar, crear o eliminar una variable. Esta clase modifica sus variables internas, luego informa a las clases DebugBlock y ADTShower qué mostrar. Una desventaja de esta arquitectura, es que realiza copias innecesarias en cada paso, perdiendo eficiencia. Sin embargo, como se trabaja con pocas variables, este costo no se nota a nivel de usuario.

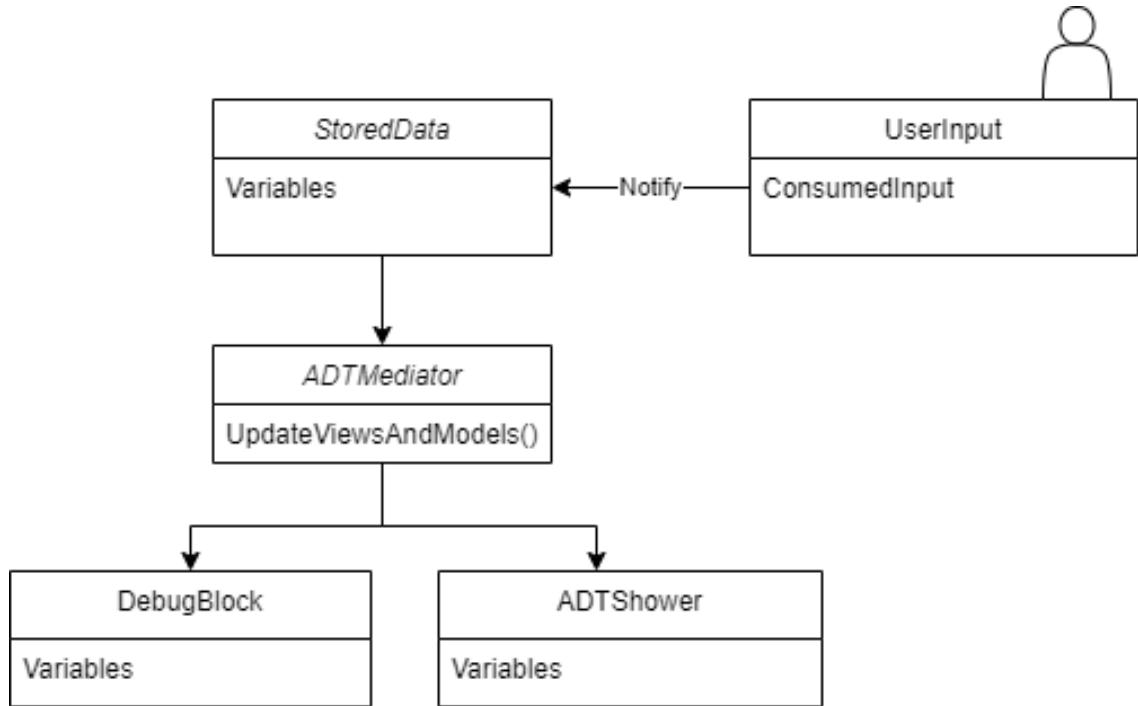


Figura 3.19: Arquitectura de software de un nivel utilizando el ADTMediator, reduciendo el acoplamiento entre clases.

Una clase que sirve de puente para comunicarse con el ADTMediator es **StoredData**. Este es un singleton que contiene datos como los avances del usuario en los distintos niveles y el estado del juego. Cuando una línea de código quiere modificar variables dentro del juego, ejecuta un método de este singleton, el cual reenvía esta información al ADTMediator. Esto se hace así porque es fácil obtener el puntero a **StoredData**, ya que usa el patrón singleton.

Otros singltons son: **AudioPlayer**, encargado de emitir sonidos específicos en cualquier momento del juego y **NotificationManager**, que genera ventanas emergentes para el usuario, como menúes, avisos o pestañas donde se debe responder sí o no. Además, se encarga del paso entre niveles.

# **Capítulo 4**

## **Diseño experimental: Metodología**

El objetivo de este trabajo era poner a prueba las hipótesis de que un videojuego educativo puede enseñar algoritmos relacionados a grafos, y que puede mantener a los estudiantes motivados y enfocados en la tarea que se les pide realizar.

La forma de comprobar tales hipótesis es mediante un experimento donde se le pide a personas voluntarias responder ciertas preguntas después de probar la aplicación.

### **4.1. Fases del trabajo de investigación**

El trabajo se dividió en tres fases, las cuales constaron de distintas preguntas después de probar la aplicación.

#### **4.1.1. Fase exploratoria**

En esta primera fase se buscaba acumular retroalimentación y opiniones de usuarios de la forma más libre posible, utilizando metodologías de loud thinking con usuarios experimentados en grafos. Las razones por las cuales se usó esta metodología son las siguientes.

Si un usuario conoce el algoritmo y la estructura de datos, pero no entiende el videojuego, entonces hay problemas de usabilidad, por lo que se justifica partir con gente experta.

Usuarios expertos pueden expresarse con más naturalidad cuando saben que no les espera un formulario al final y procuran guardar menos sus opiniones en el momento, por lo que se prefirió no dejar no utilizar una encuesta o escala post experiencia.

Se considera importante que los usuarios retraten sus impresiones a medida que los elementos del videojuego aparecen en pantalla y no después a la experiencia, para saber de mejor manera qué siente la persona cuando usa la aplicación por primera vez. Hay animaciones que deben llamar la atención en el momento, como la pista visual del ratón indicándole al usuario que haga click izquierdo en un planeta.

Se acabó con esta fase una vez los resultados de las pruebas de usuario mejoraron y se observó que cinco usuarios expertos pudieron terminar la prueba de inicio a fin sin estancarse. La condición es que estos usuarios no podían conocer el juego previamente.

#### **4.1.2. Fase de evaluación académica y percepción de usuario**

Se consiguió una muestra de 15 personas que participaron en esta fase de principio a fin.

Para iniciar esta fase, se hizo un llamado voluntario en el foro de las secciones de Algoritmos y Estructuras de Datos de la Universidad de Chile durante el semestre de primavera del año 2023. Se ofreció remuneración a todas las personas que completaran la experiencia. La experiencia duraba 45 minutos en promedio y consistía en tres partes: Realizar la prueba de usuario, llenar un formulario basado en el modelo MEEGA+ [26] y responder la prueba escrita.

El modelo sistemático MEEGA+ [26] está hecho para evaluar videojuegos educativo, el cual busca evaluar la percepción de la calidad de un videojuego desde la perspectiva del estudiante en el contexto de la enseñanza de la computación. El formulario utilizado en este estudio, basado en MEEGA+ [26] se encuentra en el anexo ??.

La medición de rendimiento académico se hace a través de una prueba escrita con dos preguntas, basadas en una pregunta de un examen del ramo de Algoritmos y Estructuras de Datos de la misma facultad. La prueba escrita se encuentra en el anexo ??.

#### **4.1.3. Trabajo y encuesta libre**

Para ampliar el estudio y aumentar el tamaño muestral, se realizó una tercera experiencia abierta a todo público que supiera programar. Se envió una invitación en distintas comunidades de videojuegos a probar el juego educativo y llenar el formulario. Como las experiencias podían variar de gran manera en este trabajo, se agregó una pregunta que pregunta por el nivel de experiencia en programación, para permitir la segmentación. El formulario utilizado también corresponde al modelo MEEGA+. Los resultados se anotaron aparte de la fase anterior.

# **Capítulo 5**

## **Resultados**

### **5.0.1. Prueba académica**

Las 15 personas que hicieron la prueba tuvieron ambas respuestas correctas, identificando correctamente el recorrido BFS y DFS en cada caso.

### **5.0.2. Formulario MEEGA+: Percepción de usuario**

Las respuestas al formulario se encuentran en: <https://docs.google.com/spreadsheets/d/1DlxUT5o-nRknYXGtFWqkXX2ZkuEZJZsow4DbW6bUTh8/edit?usp=sharing>

Debería poner aquí los gráficos o en un anexo?

# **Capítulo 6**

## **Discusión**

### **6.1. Fortalezas de la metodología aplicada**

La principal fortaleza de la metodología aplicada es que se basa en un formulario estandarizado y ya probado. Existe una confianza alta en que el formulario mide lo que efectivamente busca medir, lo cual se validó a través de la delta de Cronbach [26].

El proceso de diseño también se considera robusto, pues incorporó retroalimentación por parte de expertos y la utilización de la metodología de Loud Thinking en cada iteración, lo que permitió que los jugadores pudieran desenvolverse correctamente en el juego y finalmente responder la prueba escrita.

### **6.2. Debilidades y mejoras para la metodología aplicada**

Una debilidad identificada es que la prueba académica falló en detectar diferencias entre los grupos, al asignarle un puntaje perfecto a todos los estudiantes. Se proponen dos metodologías para mejorar la precisión de los resultados: 1.- Aplicar A/B testing, comparando la metodología del videojuego educativo con otra aplicación, o la lectura de un artículo relacionado a la misma materia que busca enseñar el videojuego. Luego, probar a los estudiantes con algún examen escrito que contenga más preguntas, de manera que se pueda establecer una graduación y hacer una comparación directa entre distintos métodos de educación. 2.- El caso ideal sería tomar una muestra aleatoria de estudiantes que estén cursando Algoritmos y Estructuras de Datos y mostrarles el videojuego. Posterior a un examen, medir las diferencias entre el grupo que probó el videojuego y el grupo que no lo hizo. Sin embargo, para esto se requiere apoyo institucional y asegurarse de que se vea la materia de grafos en el curso.

El trabajo hecho no mide una componente social asociada al videojuego. Esta se descartó por dificultades en la aplicación de las pruebas de usuario y porque era difícil implementar a nivel de diseño una componente social en el videojuego que esté justificada desde el punto de

vista del usuario. El modelo MEEGA+ [26] busca evaluar también una componente social.

No se puede asegurar la heterogeneidad de la muestra. De partida, existe el sesgo del voluntario. El sesgo del voluntario afecta de acuerdo a [30]. El caso ideal consiste en hacer una separación aleatoria en los cursos como parte de las actividades de la clase. Sin embargo, es probable que aún así existan sesgos en los cursos por la simple selección de estudiantes de carreras asociadas a computación o informática.

Además, autores como mencionan que es mejor hacer una prueba dos semanas después para medir aprendizajes. Sin embargo, esto era difícil de realizar dadas las condiciones del trabajo, puesto que fue difícil conseguir voluntarios en un contexto donde los estudiantes prefieren dedicar su tiempo a preparar exámenes en vez de participar en una actividad universitaria [29].

Por otra parte, el autor de este trabajo no es un diseñador de videojuegos y su principal énfasis está en la programación. Se destaca el rol y la importancia del diseño del videojuego. Este tiene que ser llamativo y ofrecer mecánicas que recompensen al jugador, así como agregarle una dimensión social a través de un ranking, componentes cooperativas o competitivas. La narrativa tampoco es acabada, no es una historia que lleve un fin y no se presenta un desarrollo de personaje, tampoco se presenta un antagonista. Este tipo de elementos despiertan más el interés del jugador.

### 6.3. Trabajo futuro

En un futuro, esta misma aplicación o derivaciones de esta podrían probarse en otros contextos, con muestras más grandes, para asegurar de mejor manera la validez estadística. Por otra parte, se recomienda utilizar un diseño experimental que deje de lado sesgos como el factor de novedad y el sesgo del voluntario. Además, se recomienda medir en distintos intervalos temporales los resultados del aprendizaje al utilizar esta aplicación. Por último, esta aplicación fue creada con la intención de ser un complemento a la enseñanza tradicional y usarse en pausas activas, por lo que también se sugiere estudiarla en grupos donde se utilizó esta herramienta como complemento y compararla con otro grupo que no la haya utilizado.

Una mejora en la recolección de datos es aplicar principios de telemetría y uso de otras tecnologías que permitan profundizar el estudio de usabilidad y experiencia de usuario. Ejemplos de estas tecnologías son eye-tracking o biofeedback. Esto permitiría entender mejor y de una forma más estandarizada cómo experimentan los usuarios el videojuego y dónde hay espacio de mejora. Por ejemplo, se espera que los usuarios vean el código cada vez que realizan un paso, lo que podría verificarse utilizando eye-tracking.

Una posible expansión a este trabajo es usar otras estructuras de datos y otros algoritmos que no necesariamente tengan relación con los grafos. El software creado está hecho para ser extendible a otros algoritmos. Eso sí, esto requeriría trabajo de diseño, para determinar cómo serán las visualizaciones y representaciones de los algoritmos, así como las mecánicas.

A nivel de diseño de juego, podrían agregarse más elementos de gamificación, como adqui-

sión de nuevos elementos, desbloqueo de mecánicas, acumulación de puntos, establecimiento de un ranking global, sistema de logros y finales alternativos.

# Capítulo 7

## Conclusión

El videojuego cumple el propósito de enseñar el algoritmo de BFS y DFS en base a los resultados. Estudiantes que están iniciando en la carrera de computación en la Universidad de Chile fueron capaces de responder correctamente una prueba y entender la diferencia entre los algoritmos y de BFS y DFS, así como entender qué es un grafo.

Por otra parte, el juego posee una aprobación positiva y es considerado, según el modelo MEEGA+ [26], un juego de calidad. Se identificaron un conjunto de mejoras para realizar al proceso que están mencionadas en el capítulo de discusiones de este trabajo. Sin embargo, no queda esclarecido qué debilidades específicas puede tener esta metodología, puede que existan conceptos que no son capturados por parte de los estudiantes en este videojuego, o conceptos que son capturados erróneamente.

Se considera positivo que el videojuego sea percibido de buena manera por parte del estudiantado, puesto que esto afecta positivamente la motivación, lo cual a su vez mejora los resultados académicos, incluso si el videojuego no es un medio eficiente de aprendizaje, al menos puede constituir un complemento en la educación de la computación.

Se recomienda investigar más sobre este tipo de trabajos aplicando las sugerencias señaladas. Se necesita investigar más para entender de qué manera afecta el diseño de un videojuego, la narrativa, los efectos visuales y auditivos, la percepción subjetiva y el rendimiento académico en los estudiantes que aprendan con este videojuego.

# Bibliografía

- [1] Codecombat: Coding games to learn python and javascript. <https://www.codecombat.com/>. Accessed: 2022-May-20.
- [2] Scratch: Learn programming with blocks. <https://scratch.mit.edu/>. Accessed: 2022-May-20.
- [3] Reham Ayman, Nada Sharaf, Ghada Ahmed, and Slim Abdennadher. Minicolon; teaching kids computational thinking using an interactive serious game. In *Joint International Conference on Serious Games*, pages 79–90. Springer, 2018.
- [4] Christian Bisson and John L. Luckner. Fun in learning: The pedagogical role of fun in adventure education. *Journal of Experiential Education*, 19:108 – 112, 1996.
- [5] CADCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [6] Godot Community. Godot export variables, 2023. Accessed on 5 October 2023.
- [7] Wikipedia contributors. Golden sun (video game) wikipedia, the free encyclopedia, 2023. Accessed 04-October-2023.
- [8] DCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [9] Sarah E. Domoff, Ryan P. Foley, and Rick C. Ferkel. Addictive phone use and academic performance in adolescents. *Human Behavior and Emerging Technologies*, 2019.
- [10] Michael Eagle and Tiffany Barnes. Experimental evaluation of an educational game for improved learning in introductory computing. *ACM SIGCSE Bulletin*, 41(1):321–325, 2009.
- [11] Sarah Esper, Stephen R Foster, William G Griswold, Carlos Herrera, and Wyatt Snyder. Codespells: bridging educational language features with industry-standard languages. In *Proceedings of the 14th Koli calling international conference on computing education research*, pages 05–14, 2014.
- [12] Adam Freeman. The mediator pattern. 2015.
- [13] Stefan Fries and F. Meredith Dietz. Learning in the face of temptation: The case of motivational interference. *The Journal of Experimental Education*, 76:112 – 93, 2007.

- [14] Sarah Victoria Gentry, Andrea Gauthier, Beatrice L'Estrade Ehrstrom, David Wortley, Anneliese Lilienthal, Lorainne Tudor Car, Shoko Dauwels-Okutsu, Charoula K Nikolaou, Nabil Zary, James Campbell, and Josip Car. Serious gaming and gamification education in health professions: Systematic review. *J Med Internet Res*, 21(3):e12994, Mar 2019.
- [15] Andreas Giannakoulas and Stelios Xinogalos. A pilot study on the effectiveness and acceptance of an educational game for teaching programming concepts to primary school students. *Education and Information Technologies*, 23(5):2029–2052, 2018.
- [16] Ben Gibson and Tim Bell. Evaluation of games for teaching computer science. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, pages 51–60, 2013.
- [17] Godot Community. Godot official documentation, 2023. [Online; accessed 4-October-2023].
- [18] Lindsey Ann Gouws, Karen Bradshaw, and Peter Wentworth. Computational thinking in educational activities: an evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 10–15, 2013.
- [19] Susanna Hartikainen, Heta Rintala, Laura Pylväs, and Petri Nokelainen. The concept of active learning and the measurement of learning outcomes: A review of research in engineering higher education. *Education Sciences*, 9(4), 2019.
- [20] Ioannis Karatzas. and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, Berlin, 2nd edition, 2000.
- [21] Kemono Games. Protocorgi, 2023. [Online; accessed 4-October-2023].
- [22] Kristian Kiili, Keith Devlin, Arttu Perttula, Pauliina Tuomi, and Antero Lindstedt. Using video games to combine learning and assessment in mathematics education. *International Journal of Serious Games*, 2(4):37–55, 2015.
- [23] Jason M. Lodge and William J. Harrison. The role of attention in learning in the digital age. *The Yale Journal of Biology and Medicine*, 92:21 – 28, 2019.
- [24] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):1–15, 2010.
- [25] United Nations. The impact of digital technologies, 2023. Accessed: 2023-08-04.
- [26] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. Meega+: A method for the evaluation of educational games for computing education. *INCoD-Brazilian Institute for Digital Convergence*, pages 1–47, 2018.
- [27] React. Sharing state between components. <https://react.dev/learn/sharing-state-between-components#>  
a-single-source-of-truth-for-each-state, 2023. Accessed on 5 October 2023.

- [28] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [29] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [30] Ralph L. Rosnow, Robert Rosenthal, Roberta M. Mcconochie, and Robert L. Arms. Volunteer effects on experimental outcomes. *Educational and Psychological Measurement*, 29:825 – 846, 1969.
- [31] Grant Sanderson. 3blue1brown youtube channel. <https://www.youtube.com/3blue1brown>, 2023. Accessed on 4 October 2023.
- [32] Neil Selwyn. Looking forward : Reimagining digital technology and the contemporary university. 2014.
- [33] Cheng-Hsin Wang. Comprehensively summarizing what distracts students from online learning: A literature review. *Human Behavior and Emerging Technologies*, 2022.
- [34] David Weintrop and Uri Wilensky. Robobuilder: A program-to-play constructionist video game. In *Proceedings of the constructionism 2012 conference. Athens, Greece*, 2012.
- [35] Wikipedia contributors. Final fantasy iv — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [36] Wikipedia contributors. Pokémon emerald — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [37] Wikipedia contributors. Space invaders — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [38] Zhonggen Yu, Min Gao, and Lifei Wang. The effect of educational games on learning outcomes, student motivation, engagement and satisfaction. *Journal of Educational Computing Research*, 59:522 – 546, 2020.
- [39] Weinan Zhao and Valerie J Shute. Can playing a video game foster computational thinking skills? *Computers & Education*, 141:103633, 2019.
- [40] B. Zimmerman and Dale H. Schunk. Handbook of self-regulation of learning and performance. 2011.
- [41] Daniel Zingaro, Cynthia Bagier Taylor, Leo Porter, Michael J. Clancy, Cynthia Bailey Lee, Soohyun Nam Liao, and Kevin C. Webb. Identifying student difficulties with basic data structures. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 2018.

## Apéndice A

### Anexo A: Formulario basado en MEEGA+

Tabla A.1: Formulario basado en MEEGA+. Se eliminó la dimensión social y no se incluyeron los ítems que preguntan por los objetivos particulares del juego.

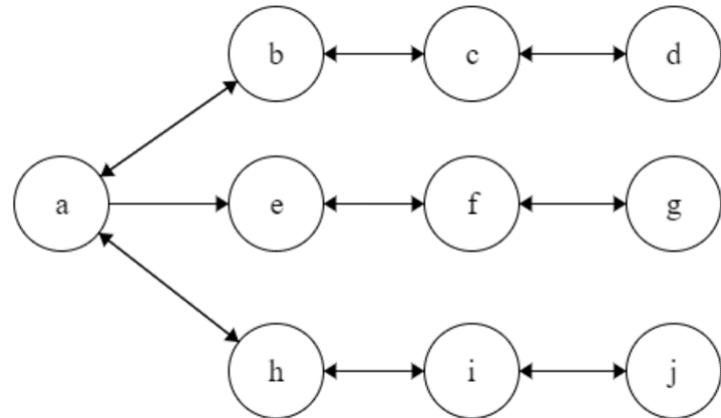
Notación	Pregunta
Q1	El diseño de juego es atractivo
Q2	La fuente de texto y los colores están bien combinados y son consistentes
Q3	Tuve que aprender cosas antes de poder jugar
Q4	Aprender a jugar se me hizo fácil
Q5	Las reglas del juego son claras y fáciles de entender
Q6	Las fuentes (su tamaño y estilo) usadas son fáciles de leer
Q7	Los colores usados en el juego son significativos
Q8	Cuando vi el juego por primera vez, tuve la sensación de que sería fácil para mí
Q9	La estructura me ayudó a tener confianza en que aprendería con este juego
Q10	Este juego es desafiante en la medida justa
Q11	El juego ofrece nuevos desafíos a un ritmo adecuado
Q12	El juego no se vuelve monótono a medida que se avanza
Q13	Completar las tareas del juego me provocó satisfacción
Q14	Pude avanzar en el juego gracias a mi esfuerzo personal
Q15	Siento satisfacción con lo aprendido en este juego
Q16	Recomendaría este juego a mis colegas
Q17	Me entretuve con el juego
Q18	Algún elemento del juego me hizo sonreír
Q19	Había algo interesante al inicio del juego que llamó mi atención
Q20	Estaba tan envuelto@ en la tarea propuesta por el juego que perdí la noción del tiempo
Q21	Me olvidé de mi entorno físico mientras jugaba
Q22	Los contenidos del juego son relevantes para mis intereses
Q23	Es claro ver que los contenidos del juego se relacionan a cierta materia de la carrera
Q24	Este juego es adecuado para enseñar el contenido
Q25	Prefiero aprender con este juego en vez de aprender con otros métodos de enseñanza
Q26	El juego contribuyó a mi aprendizaje
Q27	El juego me permitió aprender de forma eficiente en comparación con otras actividades

## Apéndice B

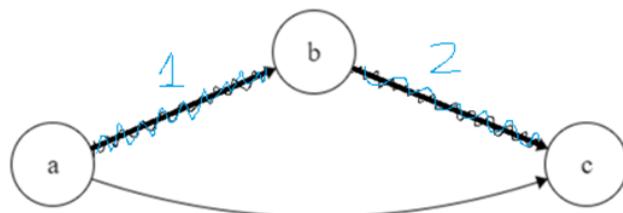
### Anexo B: Prueba de conocimiento de grafos

Test para verificar aprendizaje de BFS y DFS

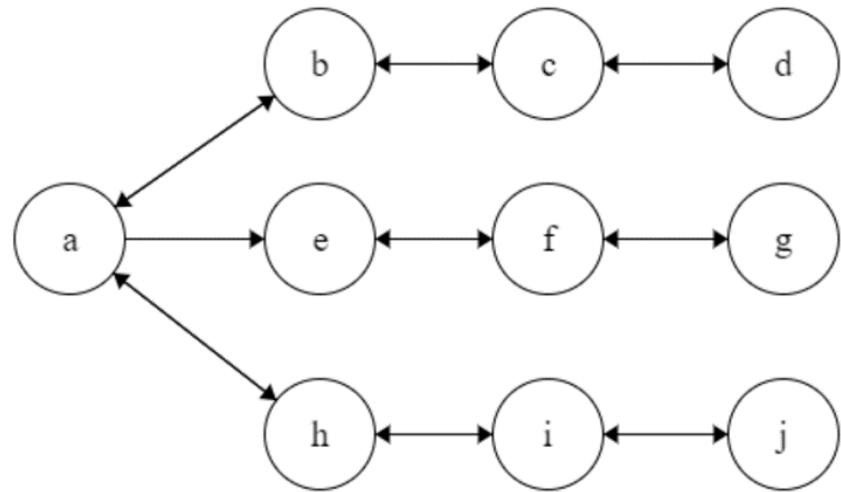
Considere el siguiente grafo:



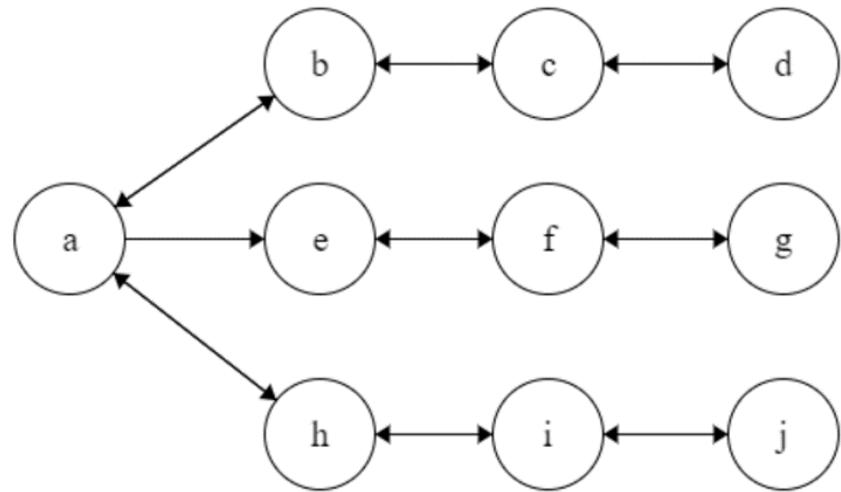
Muestre el recorrido de un grafo desde el nodo a utilizando distintos los dos algoritmos de búsqueda vistos. **Agregue una numeración al lado de sus arcos para indicar el orden en que se explorando.** Por ejemplo, en el siguiente grafo, si se recorre a, b y c partiendo desde a, pasando por b y llegando a c, el resultado esperado es el siguiente:



1. Aplique el algoritmo **Búsqueda en Profundidad** (DFS / Depth First Search) para mostrar cómo se recorre un grafo partiendo desde el nodo a.



2. Repita lo anterior usando el algoritmo de **Búsqueda en Achura** (BFS / Breadth First Search)



## Apéndice C

### Anexo C: Respuestas de pruebas de usuario

## **Apéndice D**

### **Anexo D: Aprobación de Comité de Ética**

Revisar siguiente página. Esta aprobación se dio como respuesta a una solicitud para realizar un trabajo de título con personas, para asegurarse que se enmarcara dentro del marco ético establecido por la facultad.

**CERTIFICACIÓN N° 022**  
**COMITÉ DE ÉTICA Y BIOSEGURIDAD PARA LA INVESTIGACIÓN**  
**FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS**

El Comité de Ética y Bioseguridad para la Investigación de la Facultad de Ciencias Físicas y Matemáticas (FCFM) de la Universidad de Chile certifica haber analizado el proyecto titulado **“VIDEOJUEGO EDUCATIVO PARA ENSEÑAR ALGORITMOS RELACIONADOS A GRAFO”** cuyo jefe de proyecto es el profesor **Iván Sipirán Mendoza**, del Departamento de Ciencias de la Computación, FCFM, Universidad de Chile y como estudiante de tesis de Magíster participará **Alonso Utreras Miranda**.

El proyecto busca crear y evaluar un videojuego educativo que enseñe una materia abstracta relacionada a la Ciencia de la Computación. Para su evaluación, se ofrecerá dicho videojuego la plataforma de U-Cursos cuando se imparta la materia respectiva.

Los participantes serán voluntarios y responderán un cuestionario. Tanto el cuestionario como el videojuego serán accedidos de forma online. Los datos y opiniones recolectadas serán anónimas

La metodología y los objetivos del proyecto permiten certificar que:

- i) El proyecto cumple con los estándares nacionales e internacionales de ética de la investigación, de acuerdo a la Declaración Universal de los Derechos Humanos, el Pacto de Derechos Civiles y Políticos, el Pacto de Derecho Económicos Sociales y Culturales, las leyes chilenas y el Documento oficial de ética para la investigación de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile.
- ii) El Comité de Ética considera que la investigación no vulnera la dignidad de los sujetos, no constituye una amenaza bajo ninguna circunstancia ni causa daño.
- iii) Asimismo, el Comité que suscribe, podrá auditar, al término del proyecto, el cumplimiento de los estándares arriba enunciados propios de la disciplina involucrada para asegurarse de su cumplimiento. Esta auditoría, además, tiene el propósito de asegurar la garantía del derecho a la privacidad, confidencialidad y el anonimato de los sujetos involucrados en la investigación.

iv) Dejamos constancia que el profesor Sipirán será responsable por eventuales daños causado a las personas por errores que puedan cometerse durante la investigación



**Dr. Carlos Conca R.**  
**Profesor Titular**  
**Departamento de Ingeniería Matemática**

**Dra. Viviana Meruane N.**  
**Profesora Titular**  
**Directora Académico, de Investigación e**  
**Innovación**



**Dr. Manuel Patricio Jorquera E.**  
**Secretario**

Santiago, 22 de agosto de 2023

GEV/CCR/PJE/rox.

## **Apéndice E**

### **Anexo E: Consentimiento de participación voluntaria**

Revisar siguiente página. Este consentimiento debió firmarlo cada participante antes de empezar con la actividad. Era requerido por parte de la facultad para asegurar que cada individuo voluntario estuviera de acuerdo con formar parte del trabajo.

## **Consentimiento para participar en estudio de formas**

### **Objetivo**

El propósito de esta investigación es ver los niveles de interés/motivación y aprendizaje con distintos métodos de estudio. En este caso, usted probará un videojuego educativo.

### **Actividad**

- 1) Ingresar al entregado en pantalla de itch.io y esperar a que cargue la aplicación. Si la página dice *Run Game*, presionar el botón. Si ve un menú de juego, pasar al paso 2.
- 2) Indicar el lenguaje de preferencia apretando el botón en el lado superior de la pantalla.
- 3) Presionar *Start Game* y continuar el juego hasta haber completado el nivel DFS. Una vez terminado, seguir con el nivel BFS. Si usted lo desea, puede completar más niveles.
- 4) Una vez terminado el videojuego, llenar el formulario en un link que será entregado. Aquí **se le pedirá el dato personal de cuánta experiencia tiene en programación. No se asociará este dato a su nombre.**
- 5) Una vez terminado el formulario, proceda a responder las preguntas en la prueba escrita. Si usted lo desea, puede recibir feedback de sus respuestas. Para esto debe indicar su nombre en la prueba e indicarle explícitamente al instructor que quiere saber de sus resultados.

El tiempo estimado de toda la experiencia es de 45 a 60 minutos en total.

Si completa todos los pasos, recibirá una remuneración monetaria.

Cualquier duda, puede consultarle a la persona a cargo de las instrucciones.

Ante cualquier emergencia o problema, puede salir en cualquier momento de la experiencia e incluso hablar con el instructor para participar en otra ocasión.

### **Consentimiento explícito**

¿Acepta participar en la siguiente actividad?

---

Firma