



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**

# **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 [?]. 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 [?], indicando que las habilidades cognitivas de los estudiantes han cambiado, pero no necesariamente em-

peorado. Existen términos como “doomsters” y “boosters” [?], 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 [?, ?]. Se le llama multitasking al acto de intentar realizar más de una tarea a la vez. Se ha demostrado que esta práctica disminuye la productividad y el aprendizaje, aunque las personas creen que pueden hacer más de una cosa a la vez [?]. 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 [?].

Una forma de distracción reconocida en la literatura es la interferencia motivacional, propuesta y explicada por Fries y Dietz [?]. 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 [?] 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 [?] 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 el uso de videojuegos educativos como complemento 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+ [?] 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 instruir en algoritmos vinculados a grafos, incorporando las características expuestas en este documento.

## **1.3. Objetivos Específicos**

- Concebir una aplicación interactiva que visualice grafos y permita seguir los pasos relacionados con algoritmos que operan en dichos grafos.
- Implementar una evaluación estandarizada para medir la percepción de los estudiantes respecto al videojuego creado.
- Idear, desarrollar e implementar mecánicas de juego y una arquitectura de programación transferibles a otros videojuegos que instruyan en materias relacionadas con la programación.

# **Capítulo 2**

## **Trabajo Relacionado**

### **2.0.1. Videojuegos educativos o serios**

Un videojuego representa una modalidad de aprendizaje activo, donde el proceso de enseñanza no sigue la estructura tradicional que consiste en tener a un profesor exponiendo frente a un estudiante. En este enfoque, el aprendizaje implica la ejecución de pasos y la participación activa del estudiante en actividades que contribuyen a la construcción del conocimiento. La revisión realizada por Hartikainen et al. [?] presenta argumentos a favor del aprendizaje activo, destacando mejores resultados, respaldo político y las demandas contemporáneas del entorno laboral, como habilidades de comunicación y la capacidad de aprendizaje autónomo.

Los videojuegos serios, también conocidos como “Serious Gaming”, constituyen una forma de aprendizaje activo. Bell y Gibson [?] sostienen que los juegos educativos son más efectivos que las clases tradicionales, lecturas, videos y tareas. Estos autores afirman que el uso de juegos conlleva a una mejora en la retención, conocimiento factual, habilidades basadas en el conocimiento, y autoeficacia. No obstante, recomiendan que los juegos deben complementarse con otras formas de enseñanza y actividades posteriores al juego que permitan a los estudiantes reflexionar sobre la relación entre los juegos y la materia.

### **2.0.2. Análisis de otros autores sobre el potencial y falencias al trabajar con videojuegos educativos**

Bell y Gibson [?] identificaron y clasificaron 41 videojuegos de Ciencias de la Computación (CS). Uno de ellos, “Map Coloring”, aborda el tema de grafos, en particular el coloreo de grafos. Aunque se realizó una búsqueda del juego hasta la fecha (2022), no se encontró material al respecto.

Kiili y su equipo [?] 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 de Zhao y Shute [?], se enlistan ejemplos de videojuegos diseñados para enseñar programación, como Wu's Castle [?], CodeCombat [?], CodeSpell [?], y MiniColon [?]. Estos ejemplos emplean programación con texto. No obstante, también existen numerosos referentes que utilizan programación por bloques, como LightBot [?], Scratch [?, ?], y RoboBuilder [?], los cuales simplifican el proceso de aprender una sintaxis relacionada con los lenguajes de programación.

A pesar de esto, el impacto de estos videojuegos no ha sido evaluado en muchos casos. En las instancias documentadas, las muestras suelen ser reducidas, y las evaluaciones se centran principalmente en aspectos cualitativos [?, ?].

Petri y otros [?] coinciden en la falta de sistematización en la evaluación de videojuegos educativos. De hecho, existen juegos serios que ni siquiera se autodenominan o se consideran como tales [?]. Además, no hay un método estándar para evaluarlos, razón por la cual Petri et al. [?] desarrollaron el modelo MEEGA+ (Modelo para la Evaluación de Juegos Educativos y Escala EGameFlow.

Entre las deficiencias señaladas al momento de crear videojuegos, se encuentran la carencia de: 1) Definición de un objetivo de evaluación; 2) Diseño de investigación; 3) Plan de medición; 4) Instrumentos de recopilación de datos; y 5) Métodos de análisis de datos. Un ejemplo de falta de sistematización es la práctica común al analizar estas herramientas, que implica comentarios informales por parte del estudiantado [?].

# **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**

Los referentes más importantes son aquellos videojuegos que tienen un rol educativo relacionado a la programación, como CodeCombat ?? y 7 Billion Humans ??.

Se observa que estos juegos tienen una componente textual que destaca en la interfaz del juego principal, ocupando al menos el 20 % de la pantalla. Por otra parte, hay texto que avanza en una línea narrativa al inicio y final de cada nivel.

En ambos casos la interfaz de código se encuentra a la derecha. Además, estos juegos permiten mostrar al usuario cómo la ejecución de las instrucciones afecta al juego paso a paso.

De este análisis se desprende que el juego tiene por requisito mostrar la ejecución del código instrucción por instrucción.

### **3.2. Objetivo del diseño deseado**

En este apartado se justifican las decisiones de diseño que llevaron a las mecánicas de juego, cuáles son los elementos interactivos y la ubicación y responsividad de estos. Según Rogers en su libro sobre diseño interactivo [?], 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 parte del supuesto de que, en muchas ocasiones, 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 les asigna como ejercicio realizar el procedimiento siguiendo cada instrucción rigurosamente, los estudiantes no lo llevan a cabo o lo hacen de forma incorrecta sin darse cuenta. Por ejemplo, en [?], se menciona que los estudiantes a menudo creen entender los contenidos, pero tienen conceptos erróneos sin percatarse.

En el trabajo de Zingaro et al. [?], se mencionan algunos ejemplos de malentendidos por parte de los estudiantes, como errores al comprender los heaps y las formas en que pueden representarse o construirse. Estos malentendidos pasan desapercibidos y no son detectables por los mismos estudiantes.

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

Un requisito esencial es que el videojuego logrado debe evitar la mecanización por parte del estudiante, obligándolo a revisar exhaustivamente cada paso relacionado con el algoritmo, evitando dar por sentado que se entiende el código y saltar a la siguiente actividad.

Por otra parte, para enseñar conceptos relacionados con programación, lo ideal es buscar usuarios que entiendan 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 que conlleva.

### **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 abordan los conceptos básicos de grafos sin indicar explícitamente al usuario qué es un grafo. Además, se enseñan conceptos de jugabilidad, como explorar o seleccionar un nodo, navegar en el código ejecutando instrucciones y responder preguntas del tipo Sí/No cuando el código incluye una instrucción con un if.

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

El juego se centra en explorar planetas a través de rutas, utilizando una metáfora de grafos, donde los planetas representan nodos y los caminos entre ellos son aristas. El objetivo

es rescatar pandas rojos dispersos en los planetas. Para lograrlo, es necesario explorar todos los nodos en una galaxia, correspondiente a un nivel. En cada nivel, se enseña un algoritmo distinto.

El juego guía al usuario mediante instrucciones que aparecen en el lado derecho de la pantalla. El jugador pasa a la siguiente instrucción presionando la tecla de espacio. Cada vez que avanza a la siguiente instrucción, se desencadenan efectos y animaciones que muestran qué está sucediendo en el código. Estas animaciones se reflejan en el viewport del juego, donde se destaca un planeta, un camino o una nave que se desplaza hacia otro planeta.

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.

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

Las acciones del jugador pueden ir acompañadas de un sonido recompensante, permitiéndole avanzar a la siguiente instrucción. En caso de cometer un error, se indicará al jugador mediante una animación visual y un sonido. Si el jugador no sabe qué hacer, en ciertos casos se mostrará un hint visual indicándole que debe presionar alguna tecla específica o hacer clic 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.

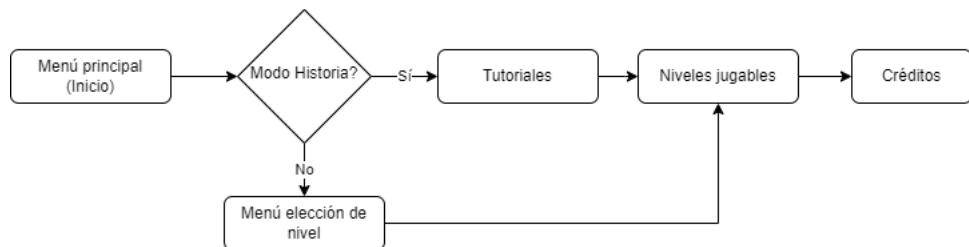


Figura 3.1: Flujo de juego de IGACSE

### 3.3.2. Narrativa

La narrativa se desarrolla en el espacio exterior, donde una nave inicia un diálogo con una estación llamada DCC. La misión de la nave es rescatar pandas rojos dispersos en los

planetas de la galaxia. En un inicio, el piloto señala que los niveles de combustible son bajos, y la estación sugiere continuar con las siguientes galaxias, evitando el uso del piloto automático debido al alto costo y consumo de combustible de las redes neuronales. Para optimizar recursos, la nave deberá seguir instrucciones manuales para completar la 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). Durante las actividades de bienvenida a los nuevos estudiantes, se muestran afiches con esta mascota [?], 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 [?].

### 3.3.3. Menú principal

En el menú principal, se puede seleccionar el idioma presionando el botón en la esquina superior izquierda de la figura ???. Los idiomas disponibles son inglés y español. Además, ofrece las opciones para jugar en el modo historia o los niveles jugables.

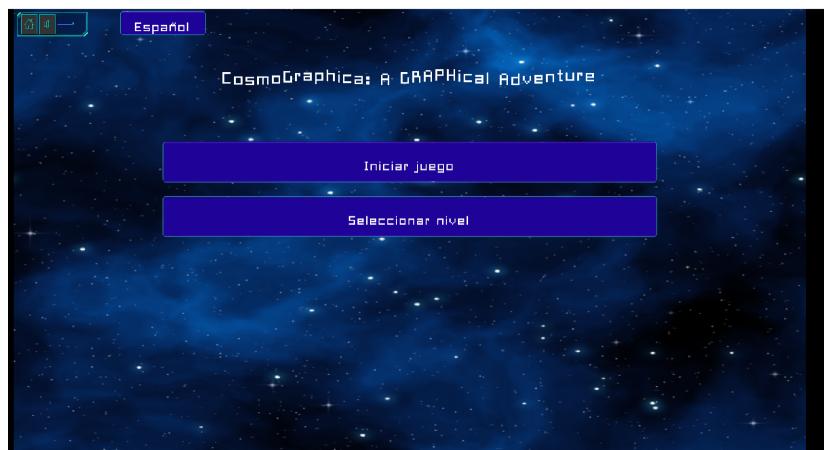


Figura 3.2: Menú principal de IGACSE, que se muestra al iniciar el juego.

### 3.3.4. Tutoriales

Los tutoriales tienen como objetivo familiarizar al jugador con las mecánicas básicas que se utilizarán a lo largo del juego. Además, introducen al jugador en la historia, ambientada en una nave que rescata pandas rojos dispersos en los planetas de la galaxia. La trama busca ilustrar la aplicación de los grafos sin detallar explícitamente el contenido que se está enseñando. Las mecánicas se presentan mediante una combinación de elementos, como texto y pistas visuales, que pueden incluir un ratón realizando clic izquierdo o una tecla R siendo presionada sobre el planeta. Cada nivel y tutorial son repetibles, permitiendo al jugador revisarlos en caso de no comprender algún aspecto.

## Primer Tutorial

En el primer tutorial, se pretende familiarizar al jugador con la pantalla, demostrando que los planetas son naveables y que existen caminos que los conectan, permitiendo hacer clic entre ellos. En esta etapa, también se proporciona información sobre el diálogo y se presentan pistas visuales.



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

## Segundo tutorial

En el segundo tutorial, se presenta la mecánica de instrucciones e introduce el ciclo for y el if en las instrucciones, elementos que se utilizarán más adelante.

Al inicio del segundo tutorial, se le indica al jugador que debe seguir las instrucciones del algoritmo debido a que la exploración automatizada de los pandas rojos es costosa y el combustible se está agotando. Las primeras dos instrucciones solicitan al jugador que presione la tecla espacio para avanzar a la siguiente instrucción (ver figura ??).



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. El jugador debe dar la respuesta correcta para avanzar a la siguiente instrucción. Si responde incorrectamente, perderá y deberá reiniciar el nivel. En la figura ??, se muestra la ventana que aparece al jugador al llegar a una instrucción con un if.



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

### 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 cual se usa para señalar a qué objeto se le agregará el nodo que el usuario presione.

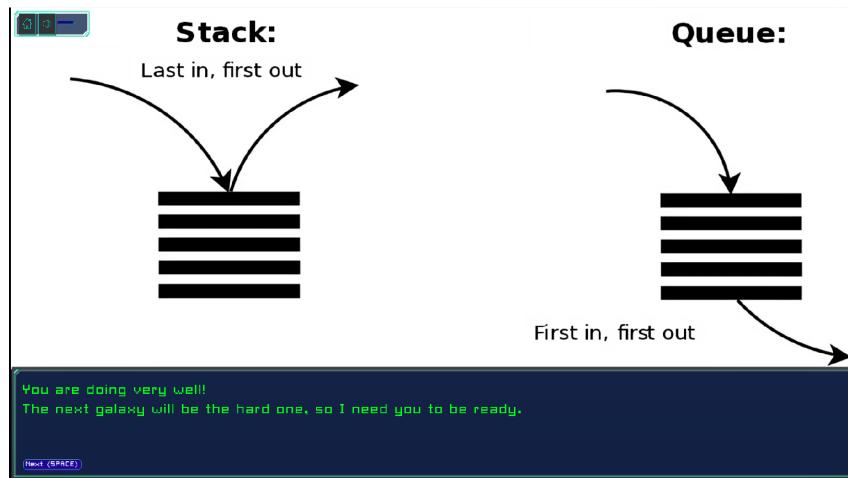


Figura 3.6: Tercer tutorial. Se le explica al usuario al inicio qué son las colas y stacks y cómo funcionan y en qué difieren.

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.



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.

### 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 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 hipotetiza que los estudiantes de computación tienen dificultades para comprender completamente los algoritmos que se enseñan en los cursos de programación, y que no siempre son conscientes de esta falta de comprensión. Por lo tanto, se busca que este videojuego sea una herramienta que permita a los estudiantes entender los algoritmos paso a paso, guiándolos sutilmente a lo largo del proceso de aprendizaje.

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 para lograr que el jugador se sienta identificado con el juego y comprenda un uso inmediato de estos algoritmos y estructuras. Por otra parte, los diseñadores enfatizaron la necesidad de agregar una componente de premiación y gamificación para despertar la motivación de los usuarios.

Una vez finalizados los tutoriales, junto con la incorporación de los estilos, música y elementos narrativos, se procedió a publicar el videojuego en la plataforma de itch.io IGASCE,

un servidor de venta de videojuegos y elementos relacionados 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 ??). 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.

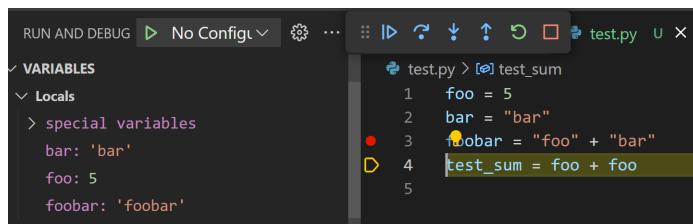


Figura 3.9: Visual Studio Code Debugger.

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. La porción que contiene los elementos y eventos del juego está al lado izquierdo. Lo que ocurre en la interfaz de la derecha afecta al mundo virtual, tal como en la aplicación IGASCE.



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. Además, el menú del juego se accede en la esquina superior izquierda (Ver figura ??).



Figura 3.11: Videojuego CodeCombat

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

Los juegos retro comúnmente emplean bucles de música de 8 a 16 bits. Ejemplos notables incluyen Golden Sun [?], ProtoCorgi [?], Pokemon Emerald [?], Space Invaders [?], y Final Fantasy IV [?].

En juegos como Pokemon [?] y Golden Sun [?], 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. El objetivo es confirmarle al usuario que se está ejecutando correctamente una acción. IGASCE sigue una estrategia similar, utilizando sonidos agudos de confirmación al hacer clic en planetas, caminos o al presionar teclas correctamente (espacio o R). Estos sonidos nítidos cumplen la función de validar la ejecución exitosa de una acción sin distraer al jugador.

En contraste, los sonidos de error en IGASCE emplean tonos graves con desvanecimiento, generando una sensación de vibración (ver figura ??). Estos sonidos más intensos se activan cuando el jugador comete errores, como hacer clic en un planeta incorrecto o un camino equivocado, presionar una tecla antes de completar una instrucción, o responder incorrectamente a preguntas del tipo Sí/No. La intención es que el jugador reconozca inmediatamente los errores y tome medidas correctivas.



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 condición de victoria en cada nivel implica completar todas las instrucciones proporcionadas por el algoritmo. En los niveles de BFS y DFS, esto implica visitar todos los nodos del grafo. El puntero de instrucciones señala la tarea actual, y el jugador debe ejecutar correctamente lo indicado en la instrucción, ya sea haciendo clic, respondiendo una pregunta, o agregando un nodo a una estructura de datos. Algunas instrucciones, especialmente aquellas que contienen la palabra clave *for*, no requieren acción por parte del jugador. Una vez que se han ejecutado correctamente los pasos requeridos, la instrucción cambiará de color y el cursor se transformará en un ticket.

Para avanzar en las instrucciones, el jugador debe presionar la tecla espacio. Si se presiona la tecla espacio sin completar la instrucción, se activará un sonido de error y el color de la instrucción parpadeará entre rojo y amarillo durante un segundo. Presionar la tecla espacio después de completar la instrucción generará un sonido de confirmación y avanzará a la siguiente instrucción. Si se presiona la tecla espacio cuando no hay más instrucciones, se emitirá un sonido de victoria, permitiendo al jugador pasar al siguiente nivel o visualizar los créditos según corresponda. Este proceso se resume en el diagrama ??.

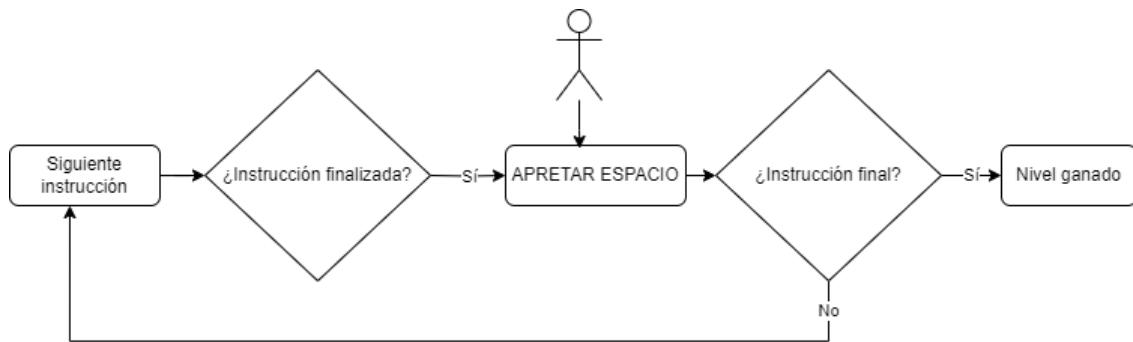


Figura 3.14: Flujo de mecánicas de Nivel. El jugador debe seguir las instrucciones paso a paso.

## 3.5. Arquitectura de software

Antes de abordar la arquitectura específica del programa, es crucial comprender cómo se estructuran los programas en Godot. Esta comprensión sirve como base para justificar las decisiones de diseño adoptadas 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 las unidades fundamentales de construcción del programa, representando diversos elementos como botones, texturas, reproductores de sonido o áreas de colisión. Para crear personajes con lógica compleja, como habilidades, movimiento y colisiones, se combinan múltiples nodos.

La estructura del grafo de escena es beneficiosa en el desarrollo de videojuegos, ya que mover el nodo principal de un personaje implica automáticamente el movimiento de elementos asociados, como colisiones, sonidos y texturas. Esto simplifica la manipulación de elementos relacionados.

Otra ventaja clave de Godot es la capacidad de utilizar variables exportadas. Estas variables, cuyos valores se definen desde el editor, permiten instanciar objetos de la misma clase con características distintas [?].

En comparación, trabajar a un nivel más bajo con bibliotecas como PyGame, que emplea OpenGL, requiere abordar individualmente cada aspecto de la representación de un objeto. Por ejemplo, mover un personaje implica gestionar texturas y colisiones por separado en el código, aumentando costos y complejidad de desarrollo [?].

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

La aplicación se desarrolla en diferentes fases, las cuales abarcan tres tipos de escenas: menús, tutoriales y niveles jugables. Cada fase hace uso de módulos distintos, y cada nivel, menú o tutorial representa una escena.

Los menús se construyen principalmente mediante nodos de control, comúnmente utilizados para interfaces gráficas. En este contexto, los menús consisten principalmente en botones que ejecutan códigos específicos según su descripción, dispuestos en formato vertical o en forma de grilla (Ver figura ??).

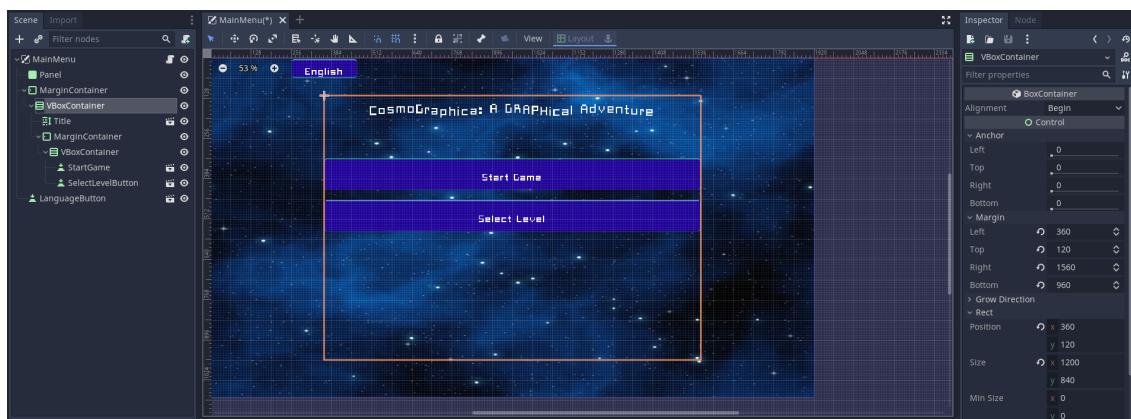


Figura 3.15: Interfaz de Godot describiendo la escena del menú principal de IGASCE. El juego públicamente se dio a conocer como CosmoGraphica, pero el proyecto completo se llama IGASCE.

Los tutoriales presentan una lógica más compleja y se asemejan a los niveles. A diferencia de las escenas finales, los tutoriales incorporan una ventana de diálogo que busca vincular la

historia del videojuego con los elementos interactivos. Además, en los tutoriales, el grafo ya está predefinido, lo que significa que los nodos y los arcos ya están establecidos de antemano.

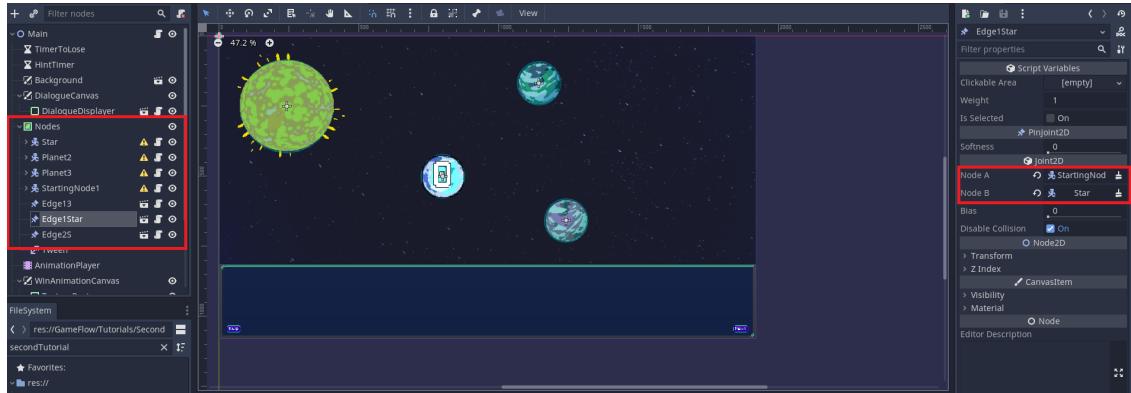


Figura 3.16: Interfaz de Godot describiendo el segundo tutorial. En rojo se observa el grafo de escena con los nodos y los arcos ya predefinidos a la izquierda. En la derecha se muestra cómo se unen los arcos, indicando los dos extremos del arco a través de variables exportadas de Godot.

En cuanto a los niveles jugables, hay varios elementos destacables, como el código, las variables, la variable seleccionada y la ventana de juego.

El código que se muestra a la derecha de la pantalla durante el juego está controlado por un nodo de la clase `CodeContainer`. Este objeto se encarga de recibir la entrada del jugador para avanzar en el código. Si las acciones ejecutadas son correctas, el usuario pasa a la siguiente instrucción, desbloqueando nuevas tareas. La clase `CodeContainer` posee un arreglo de objetos del tipo `CodeLine` llamado `code_lines`. La clase `CodeLine` contiene una única línea de código. El diagrama de clases ilustra la relación entre `CodeLine`, `IEffectCheck` y `CodeContainer`.

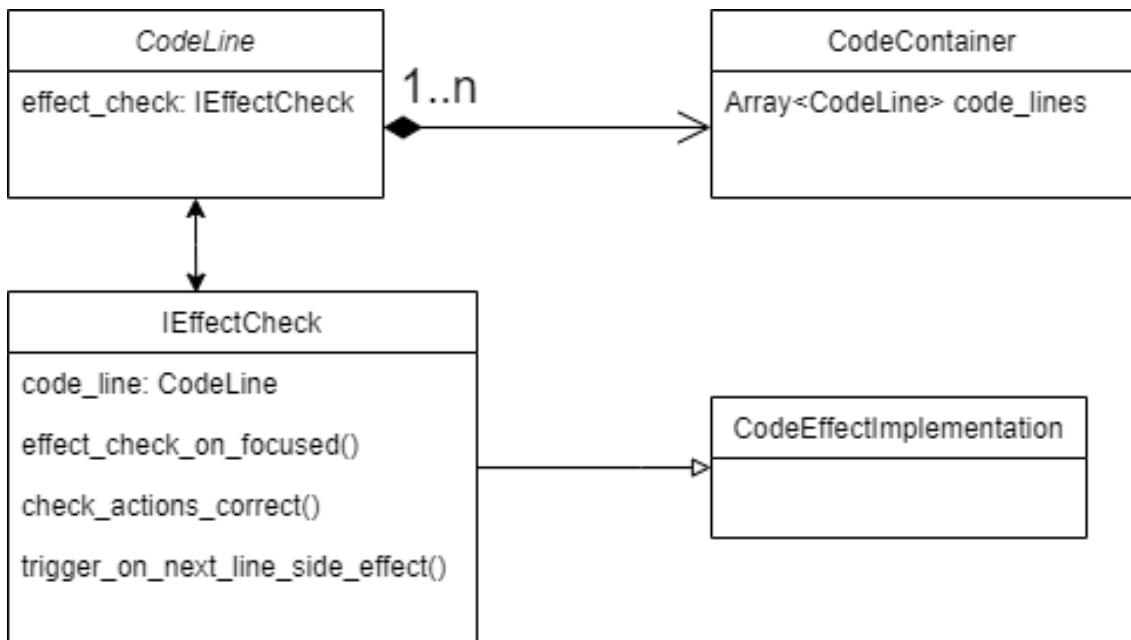


Figura 3.17: Arquitectura de la ejecución de código dentro del juego.

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: 1) cuando el puntero de instrucción llega a la línea del script; 2) verificar que la instrucción actual se ha realizado correctamente; y 3) si es necesario, manejar efectos colaterales de la instrucción. Este último se utiliza en los ciclos for, donde se mueve el índice utilizado para avanzar por los ciclos y recorrer arreglos, así como para la creación de variables.

En el panel inferior, se encuentran dos clases esenciales: DebugBlock y ADTShower. DebugBlock 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 ajusta sus variables internas. Hay una variable seleccionada que puede ser manipulada por el usuario de diversas maneras. Al presionar la flecha hacia arriba, se selecciona la variable que está encima de la actual y viceversa para la flecha hacia abajo. En la figura ??, se observa que la variable "q" está seleccionada.

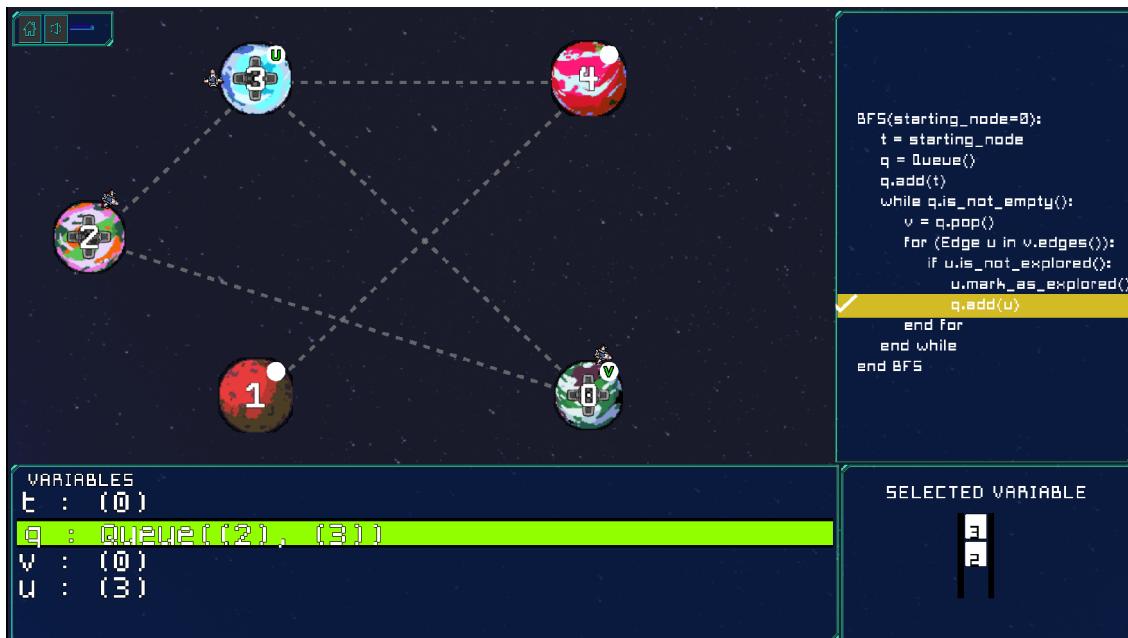


Figura 3.18: Nivel jugable BFS. Se observa el código a la derecha, el grafo en el centro, el DebugBlock en la parte inferior izquierda y el ADTShower en la esquina inferior derecha. La variable q está seleccionada.

La clase ADTShower se encarga de mostrar en la esquina inferior derecha la variable que ha sido seleccionada por el usuario. Busca añadir 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 debido al efecto de alguna CodeLine o línea de código del juego requería modificar las clases DebugBlock, ADTShower y StoredData (explicada más adelante), además de afectar la misma línea de código. Además, estos efectos deben reflejarse en las líneas de código siguientes para lograr cohesión. Por ejemplo, si la quinta línea crea la variable "u" y la sexta la modifica, esto se refleja correctamente en todas las estructuras mencionadas.

Para resolver el problema del alto costo de implementación, se optó por crear una clase llamada ADTMediator, siguiendo el patrón Mediator [?]. Esta clase se inspiró en la arquitectura de React, donde el código generado por React funciona como una única fuente de verdad o "Single Source of Truth"[?]. Con este tipo de arquitectura, se envía un mensaje a ADTMediator solicitando modificar, crear o eliminar una variable. Esta clase ajusta sus variables internas, luego informa a las clases DebugBlock y ADTShower sobre qué mostrar. Una desventaja de esta arquitectura es que realiza copias innecesarias en cada paso, perdiendo eficiencia. Sin embargo, dado que se trabaja con pocas variables, este costo no es notable a nivel de usuario.

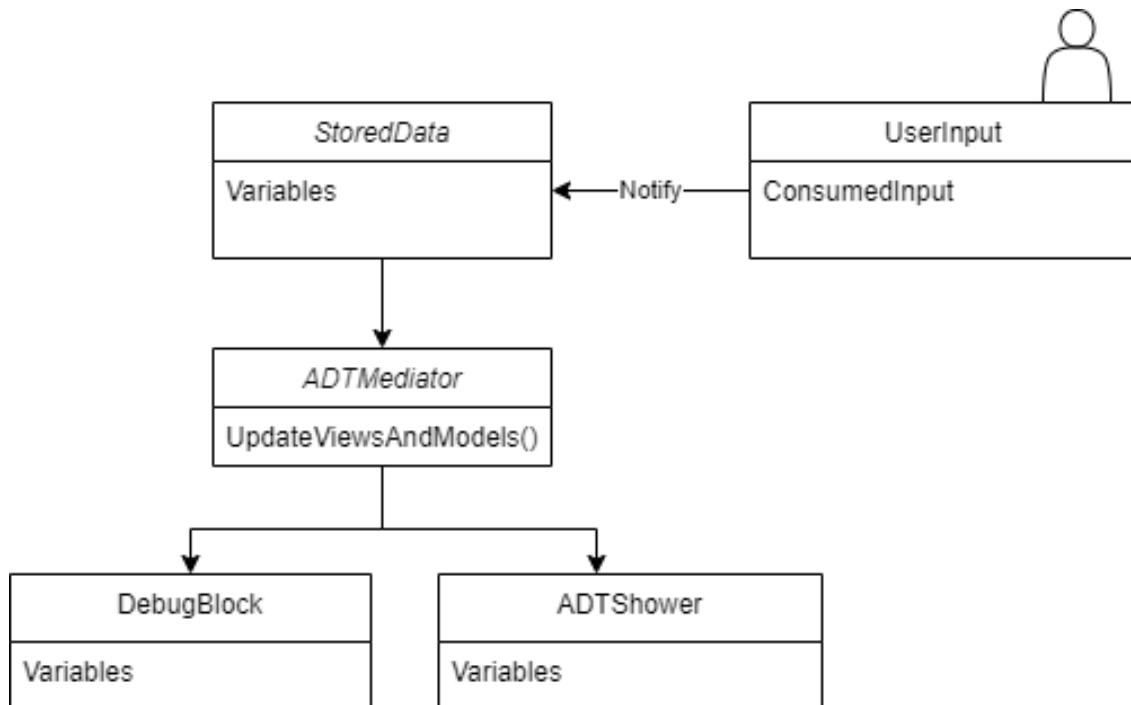


Figura 3.19: Arquitectura de software de un nivel utilizando el ADTMediator, reduciendo el acoplamiento entre clases.

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 singletons son: AudioPlayer, encargado de emitir sonidos específicos en cualquier momento del juego y NotificationManager, que genera ventanas emergentes para el usuario, como menús, avisos o pestañas donde se debe responder sí o no.

# **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+ [?] y responder la prueba escrita.

El modelo sistemático MEEGA+ [?] 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+ [?] 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. 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+, pero las respuestas recolectadas se guardaron en una base de datos separada del grupo anterior.

#### **4.1.4. Marco teórico que justifica el modelo MEEGA+**

TODO: Hablar aquí de MEEGA+, su metodología, por qué funciona lo que hicimos. Explicar un poco sobre IRT, la elección de parámetros que usan y cómo se justifica. [?].

# **Capítulo 5**

## **Resultados**

### **5.1. Resultados durante la fase exploratoria**

Inicialmente, una vez probado el videojuego hasta terminar el nivel o llevar 12 minutos jugando, se le preguntaba al usuario con conocimiento experto cuánta usabilidad le daban al juego en una escala del 1 al 10.

Se hicieron tres iteraciones con esta metodología, además de aceptar los comentarios abiertos. En la primera fase, el promedio de usabilidad con 4 usuarios fue 2.5.

Posteriormente, una vez añadidas las implementaciones sugeridas anteriormente, la usabilidad ascendió a 6 con un tamaño muestral de 4, pero los usuarios reportaban que era muy fácil cometer errores, además de que la monotonía del juego y el tener que leer mucho producían mucha insatisfacción, pese a que era comprensible qué se podía hacer.

Finalmente, antes de probar con el público objetivo, se hizo una última prueba de usuario con un tamaño muestral de 6, obteniendo en promedio un puntaje de 8 sobre 10. En este caso los usuarios reportaron problemas menores, principalmente asociados a la interpretabilidad de ciertas animaciones o a la falta de estímulos, como la indicación de apretar ENTER para avanzar en el código.

### **5.2. Resultados de la fase de evaluación académica**

#### **5.2.1. Prueba académica**

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

### 5.2.2. Formulario MEEGA+: Percepción de usuario

Según [?] un set de respuestas de un individuo se considera correcto y válido cuando tiene más del 85 % de las preguntas contestadas. En este caso no hubo omisiones.

En la figura ?? se muestran los resultados agregados para cada pregunta en cada categoría. Las preguntas se acortaron con acrónimos, pero la indexación está disponible en el anexo A ??.

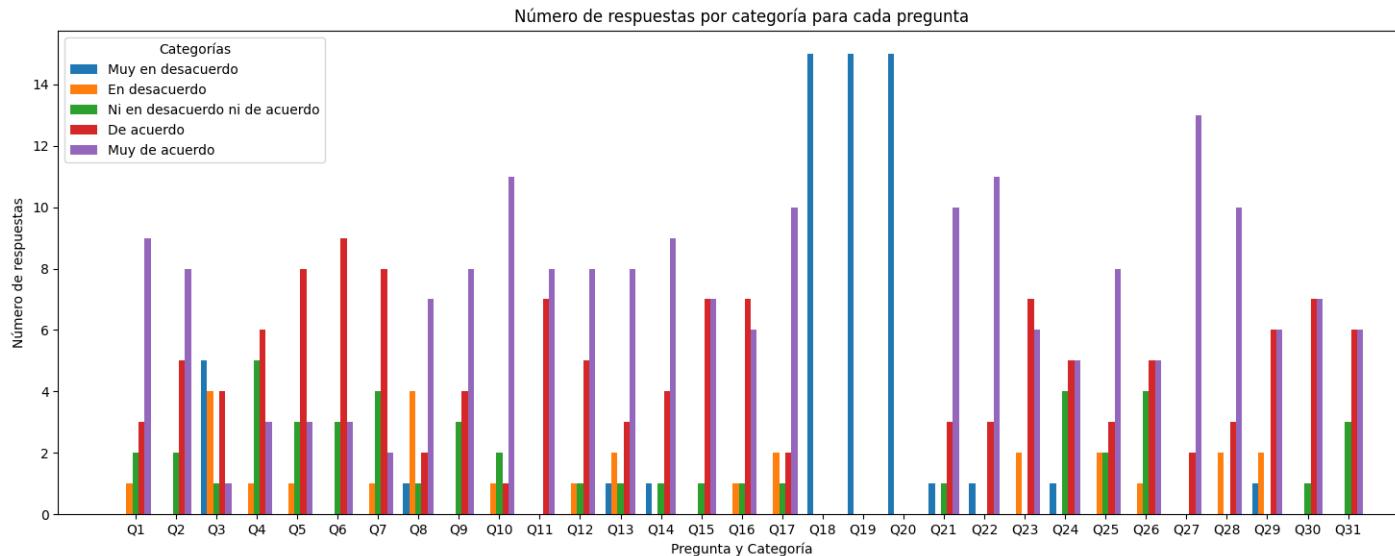


Figura 5.1: Agregación de respuestas por categoría y por pregunta

Por otra parte, en [?] se utiliza Item Response Theory (IRT) para asignar un valor  $\theta$  para la calidad del juego. Este se calcula a través de un script de R entregado en la página del Software Quality Group de la Universidad Federal Santa Catarina de Brasil [?].

En tal página, se entrega un archivo de parámetros que pondera cada pregunta, además de asignar un valor de dificultad de elección entre cada ítem. Es decir, indica qué tan difícil es que un usuario se encuentre entre responder, por ejemplo, muy en desacuerdo y de acuerdo, facilitando la distinción entre opiniones.

### 5.3. Resultados de la encuesta libre

Todavía en curso. Se mostrará otro gráfico como le de la figura ??.

# **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 [?]. Esta metodología permite asignar un valor numérico a la calidad del juego según las opiniones de los usuarios. Este valor numérico se considera rasgo latente (latent trait) y se suele designar con el signo  $\theta$  en IRT [?].

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. Se comprobó empíricamente cómo mejoró la usabilidad en cada paso.

El proceso además está fuertemente documentado, pues la historia de desarrollo está disponible en un repositorio de Github [?], de manera que cada iteración del juego es reproducible, e incluso los experimentos con usuarios son reproducibles, permitiendo a otro investigador volver a una versión anterior del videojuego y probarla con usuarios distintos en cada ocasión.

### **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+ [?] 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 [?]. 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 [?].

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 adquisición 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+ [?], 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] David Andrich and Ida Marais. A course in rasch measurement theory: Measuring in the educational, social and health sciences. *A Course in Rasch Measurement Theory*, 2019.
- [4] 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.
- [5] 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.
- [6] CADCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [7] Godot Community. Godot export variables, 2023. Accessed on 5 October 2023.
- [8] Wikipedia contributors. Golden sun (video game) wikipedia, the free encyclopedia, 2023. Accessed 04-October-2023.
- [9] Tomorrow Corporation. 7 billion humans - now available!, 2023. Accessed: 2023-11-15.
- [10] DCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [11] Sarah E. Domoff, Ryan P. Foley, and Rick C. Ferkel. Addictive phone use and academic performance in adolescents. *Human Behavior and Emerging Technologies*, 2019.
- [12] 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.
- [13] 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.

- [14] Adam Freeman. The mediator pattern. 2015.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] Godot Community. Godot official documentation, 2023. [Online; accessed 4-October-2023].
- [20] 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.
- [21] 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.
- [22] Ioannis Karatzas. and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, Berlin, 2nd edition, 2000.
- [23] Kemono Games. Protocorgi, 2023. [Online; accessed 4-October-2023].
- [24] 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.
- [25] 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.
- [26] 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.
- [27] Microsoft. Visual studio code, 2023.
- [28] United Nations. The impact of digital technologies, 2023. Accessed: 2023-08-04.

- [29] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. Meega+: A method for the evaluation of educational games for computing education. Technical Report INCoD/GQS.05.2018.E, INCoD - Brazilian Institute for Digital Convergence, 2018.
- [30] 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.
- [31] 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.
- [32] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [33] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [34] 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.
- [35] Grant Sanderson. 3blue1brown youtube channel. <https://www.youtube.com/3blue1brown>, 2023. Accessed on 4 October 2023.
- [36] Neil Selwyn. Looking forward : Reimagining digital technology and the contemporary university. 2014.
- [37] GQS UFSC. Meega+ a model for evaluating educational games. <http://www.gqs.ufsc.br/quality-evaluation/meega-plus/>. Accessed: 2023-Nov-2.
- [38] Alonso Utreras. Igasce github repository. <https://github.com/AlasAltum/Thesis-IGACSE>. Accessed: 2023-Nov-2.
- [39] Cheng-Hsin Wang. Comprehensively summarizing what distracts students from online learning: A literature review. *Human Behavior and Emerging Technologies*, 2022.
- [40] David Weintrop and Uri Wilensky. Robobuilder: A program-to-play constructionist video game. In *Proceedings of the constructionism 2012 conference. Athens, Greece*, 2012.
- [41] Wikipedia contributors. Final fantasy iv — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [42] Wikipedia contributors. Pokémon emerald — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [43] Wikipedia contributors. Space invaders — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].

- [44] 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.
- [45] Weinan Zhao and Valerie J Shute. Can playing a video game foster computational thinking skills? *Computers & Education*, 141:103633, 2019.
- [46] B. Zimmerman and Dale H. Schunk. Handbook of self-regulation of learning and performance. 2011.
- [47] 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.

## **Anexo A: Formulario basado en MEEGA+**

Tabla 7.1: Formulario basado en MEEGA+. Se eliminó la dimensión social y no se incluyeron los ítems que preguntan por los objetivos particulares del juego.

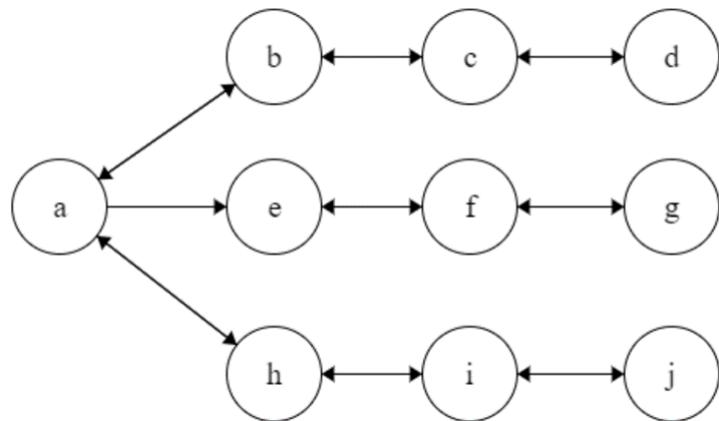
Notación	Pregunta
Q1	El diseño de juego es atractivo
Q2	La fuente de texto y los colores están bien combinados y son consistentes
Q3	Tuve que aprender cosas antes de poder jugar
Q4	Aprender a jugar se me hizo fácil
Q5	Creo que la mayoría de la gente aprendería a usar este juego rápidamente.
Q6	Creo que el juego es fácil de jugar.
Q7	Las reglas del juego son claras y fáciles de entender
Q8	Las fuentes (su tamaño y estilo) usadas son fáciles de leer
Q9	Los colores usados en el juego son significativos
Q10	La estructura me ayudó a tener confianza en que aprendería con este juego
Q11	Este juego es desafiante en la medida justa
Q12	El juego ofrece nuevos desafíos a un ritmo adecuado
Q13	Cuando vi el juego por primera vez, tuve la sensación de que sería fácil para mí
Q14	Completar las tareas del juego me provocó satisfacción
Q15	Siento satisfacción con lo aprendido en este juego
Q16	Siento satisfacción con lo aprendido en este juego
Q17	Recomendaría este juego a mis colegas
Q18*	Pude interactuar con otros jugadores mientras jugaba.
Q19*	El juego promueve la cooperación o competencia entre jugadores.
Q20*	Me sentí bien interactuando con otros jugadores durante el juego.
Q21	Me entretuve con el juego
Q22	Algún elemento del juego me hizo sonreír
Q23	Había algo interesante al inicio del juego que llamó mi atención
Q24	Estaba tan envuelta/o en la tarea propuesta por el juego que perdí la noción del tiempo
Q25	Me olvidé de mi entorno físico mientras jugaba
Q26	Los contenidos del juego son relevantes para mis intereses
Q27	Es claro ver que los contenidos del juego se relacionan a cierta materia de la carrera
Q28	Este juego es adecuado para enseñar el contenido
Q29	Prefiero aprender con este juego en vez de aprender con otros métodos de enseñanza
Q30	El juego contribuyó a mi aprendizaje
Q31	El juego me permitió aprender de forma eficiente en comparación con otras actividades

Preguntas con \* es porque no estaban en el formulario. Se omitieron y se llenaron con 0 porque el juego no incluía interacción con otros jugadores.

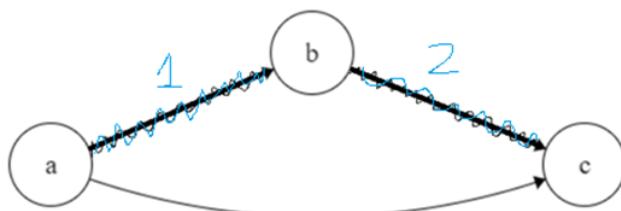
# Anexo B: Prueba de conocimiento de grafos

Test para verificar aprendizaje de BFS y DFS

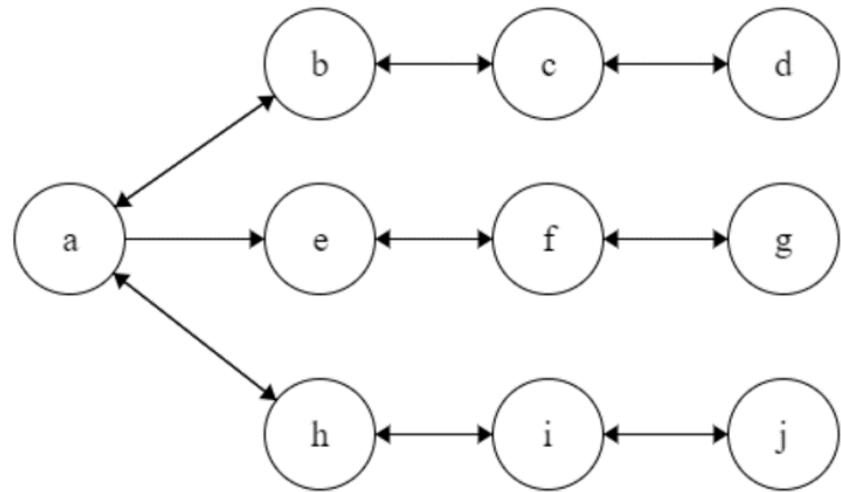
Consideré el siguiente grafo:



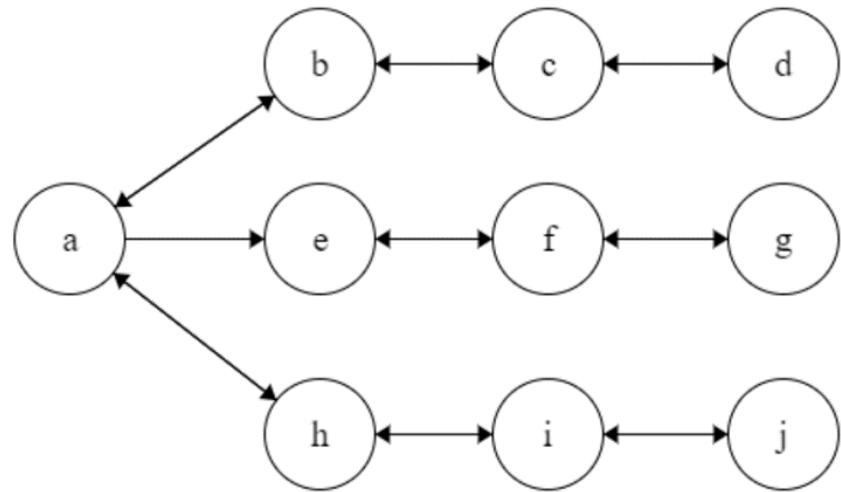
Muestre el recorrido de un grafo desde el nodo a utilizando distintos los dos algoritmos de búsqueda vistos. **Agregue una numeración al lado de sus arcos para indicar el orden en que se explorando.** Por ejemplo, en el siguiente grafo, si se recorre a, b y c partiendo desde a, pasando por b y llegando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



# **Anexo C: Respuestas de pruebas de usuario**

## **Prueba de usabilidad en conjunto con prueba académica**

Q1;Q2;Q3;Q4;Q5;Q6;Q7;Q8;Q9;Q10;Q11;Q12;Q13;Q14;Q15;Q16;Q17;Q18;Q19;Q20;Q21;Q22;Q23;Q

## **Prueba de usabilidad en formato libre para cualquier usuario**

## **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**.

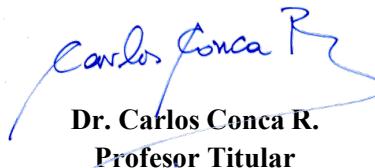
El proyecto busca crear y evaluar un videojuego educativo que enseñe una materia abstracta relacionada a la Ciencia de la Computación. Para su evaluación, se ofrecerá dicho videojuego la plataforma de U-Cursos cuando se imparta la materia respectiva.

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.

## **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