



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

IGACSE: 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:
JUAN ÁLVAREZ
NELSON BALOIAN
FEDERICO MEZA

SANTIAGO DE CHILE
2024

Resumen

Los videojuegos educativos se utilizan como herramientas destinadas a motivar y entretenir durante el proceso de aprendizaje. Para respaldar estas afirmaciones, se han desarrollado marcos de trabajo y revisiones de literatura que analizan los enfoques utilizados en los estudios relacionados con los videojuegos educativos, como por ejemplo MEEGA+.

En este estudio se crea y pone a prueba un videojuego educativo en el campo de la computación, centrándose en el contenido de grafos. Para esto, se desarrolla IGACSE, un videojuego educativo diseñado para enseñar sobre grafos y los algoritmos Búsqueda en Anchura (Breadth First Search o BFS) y Búsqueda en Profundidad (Depth First Search o DFS). La elección de grafos se sustenta en que estas estructuras de datos tienen una representación visual directa, facilitando el diseño del videojuego.

Para la evaluación cuantitativa de IGACSE, se evalúan dos factores: la percepción del usuario y su aprendizaje. Para el primer aspecto, se utiliza como referencia metodológica el enfoque de MEEGA+, que permite medir la percepción general de los participantes respecto al juego a través de un valor numérico; para el segundo aspecto, se estableció una prueba académica para medir el aprendizaje adquirido en el juego. Como muestra se consideran dos grupos, el primer grupo ($N=15$) para la medición de percepción y aprendizaje; y un segundo grupo ($N=15$) para validar solo los resultados de percepción del videojuego.

Los resultados obtenidos sugieren que el juego facilitó el aprendizaje de los estudiantes, pero se identifican espacios de mejora a nivel de diseño de juego. Además, es necesario llevar a cabo un mayor número de pruebas empleando diferentes metodologías de estudio y muestras de tamaño más significativo, con el fin de otorgar una mayor validez científica a los resultados obtenidos. La percepción de los usuarios señaló posibles áreas de mejora en usabilidad, interfaz y ludificación. La arquitectura de software desarrollada permite su reutilización con otros algoritmos y estructuras de datos. Además, se proporcionan pautas para obtener resultados más precisos en futuras investigaciones relacionadas.

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, Mario, Cisneros, Dani, Nico, Lucas, Mati, Martín, Bea, Tommy, Diego y Vale.

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

Gracias a la Facultad por brindarme el espacio de trabajo, donde disfruté y aprendí tanto. Por permitirme trabajar de lunes a domingo. Por poder pasar la noche aquí y tener un ambiente de trabajo propicio.

Tabla de Contenido

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1. Trabajo Relacionado | 3 |
| 1.1.1. Videojuegos educativos o serios | 3 |
| 1.1.2. Análisis de otros autores | 3 |
| 1.2. Preguntas de Investigación | 4 |
| 1.3. Hipótesis | 4 |
| 1.4. Objetivo General | 5 |
| 1.5. Objetivos Específicos | 5 |
| 2. Marco teórico | 6 |
| 2.1. Metodología de validación de videojuegos educativos | 6 |
| 2.2. Teoría de Respuesta al Ítem (IRT): Intuición Cualitativa | 7 |
| 2.3. Motores de videojuegos: Godot | 9 |
| 3. Diseño de IGACSE | 10 |
| 3.1. Proceso de diseño de IGACSE | 10 |
| 3.1.1. Identificación del problema | 10 |
| 3.1.2. Diseño y propuesta de la solución | 12 |
| 3.1.3. Referentes | 13 |
| 3.1.4. Diseño de la interfaz de usuario | 13 |
| 3.1.5. Diseño de efectos de sonido y música | 15 |
| 3.1.6. Diseño de mecánicas de juego | 16 |

| | |
|---|-----------|
| 3.2. Descripción de IGACSE | 17 |
| 3.2.1. Flujo de juego | 17 |
| 3.2.2. Narrativa | 18 |
| 3.2.3. Menú principal | 19 |
| 3.2.4. Tutoriales | 20 |
| 3.2.5. Niveles | 20 |
| 3.3. Arquitectura de software | 20 |
| 3.3.1. Arquitectura de una aplicación en Godot | 20 |
| 3.3.2. Arquitectura de software de la aplicación IGACSE | 22 |
| 3.3.3. Cómo agregar una nueva CodeLine y ajustarse a otro algoritmo | 26 |
| 3.3.4. Arquitectura de software de un nivel de IGACSE | 30 |
| 4. Diseño experimental: Metodología | 33 |
| 4.1. Fases del trabajo de investigación | 34 |
| 4.1.1. Fase exploratoria | 34 |
| 4.1.2. Fase de evaluación académica y percepción de usuario | 34 |
| 4.1.3. Encuesta libre | 35 |
| 5. Resultados | 36 |
| 5.1. Resultados durante la fase exploratoria | 36 |
| 5.2. Resultados de la fase de evaluación académica | 36 |
| 5.2.1. Prueba académica | 36 |
| 5.2.2. Formulario MEEGA+: Percepción de usuario | 37 |
| 5.3. Resultados de la encuesta libre | 38 |
| 6. Discusión y Conclusión | 39 |
| 6.1. Fortalezas de la metodología aplicada | 39 |
| 6.2. Debilidades y mejoras para la metodología aplicada | 39 |
| 6.3. Fortalezas del software | 41 |

| | |
|--|----|
| 6.4. Aspectos a mejorar del software | 41 |
| 6.5. Trabajo futuro | 41 |
| 6.6. Conclusión | 42 |

| | |
|---------------------|-----------|
| Bibliografía | 47 |
|---------------------|-----------|

Capítulo 1

Introducción

La adopción de tecnologías digitales ha agilizado el acceso al conocimiento, permitiendo que cada vez más personas puedan acceder a la educación [44]. Sin embargo, este avance no está exento de desafíos, especialmente en el contexto de la educación en línea, donde surgen obstáculos adicionales que requieren soluciones innovadoras y adaptativas.

Un desafío presente en la educación contemporánea es mantener la atención de los estudiantes durante períodos prolongados. Aunque se plantea comúnmente que la capacidad de atención ha disminuido en las nuevas generaciones, diversos estudios contradicen dicha afirmación [38]. Este contraste de opiniones refleja la polarización entre dos perspectivas extremas: los "doomsters", quienes tienen una visión pesimista sobre el impacto de la tecnología en la sociedad, y los "boosters", que adoptan una visión optimista frente al avance de esta [53].

Existe consenso en que la tecnología y su portabilidad han incrementado notablemente la exposición de los estudiantes a distracciones [70, 61]. El término "multitasking" se refiere al intento de llevar a cabo múltiples tareas simultáneamente. Aunque las personas tienden a creer que son capaces de realizar varias actividades de manera concurrente, diversos estudios han demostrado que esta práctica conlleva una disminución en la productividad y en el proceso de aprendizaje [18]. Con la generalización de los dispositivos móviles y la adopción de clases en línea, se ha observado un aumento significativo en la tendencia al multitasking durante las clases [61].

Una forma de distracción reconocida en la literatura es la interferencia motivacional, concepto propuesto y explicado por Fries y Dietz [27]. El término indica que la motivación por los contenidos educativos disminuye ante la presencia de estímulos más atractivos, como los dispositivos móviles. Ante este panorama, surge la necesidad de explorar estrategias que mantengan la motivación y el enfoque de los estudiantes durante las clases.

Los videojuegos educativos surgen como estrategias válidas ante el escenario actual, como una potencial herramienta complementaria para la enseñanza, evaluación y entretenimiento de los estudiantes. Entre sus beneficios se destaca la motivación. De hecho, para disfrutar de una actividad, primero se debe permitir a la mente de un individuo percibir tal actividad como motivante [9].

Yu et al. [66] llevaron a cabo una revisión sistemática de la literatura sobre los efectos de los videojuegos educativos en el aprendizaje de los estudiantes y su motivación. En relación con este último aspecto, el estudio señala que la incorporación de videojuegos educativos como recurso complementario impacta positivamente en la motivación. Sin embargo, se menciona que la evidencia es insuficiente para concluir sobre la efectividad en los resultados del aprendizaje.

El equipo de Yu también menciona una ventaja asociada a los videojuegos educativos: pueden proporcionar servicios educacionales de alta calidad, flexibles, portátiles y de bajo costo, aumentando las interacciones entre materiales de aprendizaje, estudiantes y profesores [66].

En el estudio mencionado, se afirma que el diseño del videojuego tiene una gran incidencia en el resultado final. Concluyen que las mecánicas, elementos visuales y narrativos tienen un efecto significativo, sugiriendo que los juegos educativos deberían implementar varios elementos de ludificación, como rankings, sistemas de recompensa, entre otros aspectos [66].

En más del 50% de los estudios citados anteriormente, se señala la falta de certeza con respecto a la percepción de los usuarios al jugar videojuegos educativos, concluyendo que se necesita una investigación más profunda. Por lo tanto, es pertinente emplear una prueba estandarizada que permita evaluar diversos diseños de videojuegos, abarcando diferentes poblaciones objetivo, y cuyos resultados sean medidos conforme a un estándar uniforme [66]. Con el fin de estandarizar los resultados, Petri et al. desarrolló el modelo MEEGA+ para medir la percepción de los usuarios respecto a un videojuego educativo [47].

Esta investigación surge a partir de los antecedentes planteados inicialmente, y en relación con los contenidos del campo de las ciencias de la computación. Se observa que, en muchas ocasiones, los estudiantes de esta área creen entender cómo funciona cierto código o algoritmo, sólo haciendo una revisión preliminar de las instrucciones, pero no hay una ejecución y lectura del paso a paso en paralelo. Estas prácticas llevan a que no se aprenda adecuadamente los contenidos, y se crea erradamente que se entienden a cabalidad los contenidos, pero posteriormente no logran extrapolarlos ni reproducirlos con otras variables, arrastrando conceptos erróneos sin percibirse [71].

El trabajo busca tener un impacto positivo en el aprendizaje y la motivación de los estudiantes de computación con la aplicación del videojuego educativo IGASCE. Se enseñan contenidos de dos algoritmos vinculados a grafos. Estos son Búsqueda en Anchura (BFS) y Búsqueda en Profundidad (DFS). La idea del juego es representar las instrucciones de algoritmos de grafos en una forma visual, interactiva y lúdica.

Se eligieron grafos y los algoritmos BFS y DFS porque poseen una representación visual directa y más fácil de diseñar en un videojuego. Visualizar los algoritmos permite una comprensión mayor a las explicaciones basadas en texto e ideas abstractas [56].

Con esto, la investigación de tesis realiza una evaluación cuantitativa, mediante la medición de la motivación y percepción general del juego y el grado de aprendizaje adquirido, de los contenidos de algoritmos relacionados con grafos, en particular BFS y DFS. Además, existe una componente cualitativa basada en comentarios abiertos por parte de los participantes del estudio.

El público objetivo son estudiantes de primer año de ciencias de la computación, aunque también es aplicable a estudiantes de ingeniería con conocimientos de programación. El requisito principal es que no hayan estudiado grafos previamente. La evaluación se realiza mediante una prueba académica escrita, con un puntaje y nota; y el formulario MEEGA+, también con un puntaje asignable a cada criterio a evaluar.

1.1. Trabajo Relacionado

1.1.1. 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. [34] 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 [31] 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.

1.1.2. Análisis de otros autores

Bell y Gibson [31] 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 [37] 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 [69], se listan ejemplos de videojuegos diseñados para enseñar programación, como Wu’s Castle [19], CodeCombat [1], CodeSpell [20], y MiniColon [6]. Estos ejemplos emplean programación con texto. No obstante, también existen numerosos referentes que utilizan programación por bloques, como LightBot [33], Scratch [2, 40], y

RoboBuilder [62], los cuales simplifican el proceso de aprender la sintaxis relacionada con los lenguajes de programación.

A pesar de esto, el impacto de estos videojuegos no ha sido evaluado rigurosamente en muchos casos. En las instancias documentadas, las muestras suelen ser reducidas, y las evaluaciones se centran principalmente en aspectos cualitativos, sin evaluar un aprendizaje determinado ni entregar números que permitan la comparación de estos juegos educativos con otros [69, 30].

Petri y otros [47] coinciden en la falta de sistematización en la evaluación de los videojuegos educativos. Se ha observado que algunos juegos serios ni siquiera se autodenominan o se consideran como tales [31]. Además, muchos estudios carecen de resultados cuantitativos, lo que dificulta su comparación.

Para abordar estas deficiencias, Petri et al. [47] desarrollaron el modelo MEEGA+ (Modelo para la Evaluación de Juegos Educativos y Escala EGameFlow). Entre las deficiencias identificadas al momento de crear videojuegos según Petri et al. [47] se encuentran la falta 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. Una práctica común al analizar estas herramientas es utilizar comentarios informales por parte del estudiantado, en lugar de aplicar escalas psicométricas [47].

1.2. Preguntas de Investigación

RQ1: ¿Puede un grupo de estudiantes que no tenga conocimiento previo sobre grafos, identificar y construir un recorrido BFS y DFS solo habiendo sido expuesto a un videojuego educativo sobre grafos?

RQ2: ¿Cómo perciben los estudiantes de ciencias de la computación sin conocimientos de grafos un videojuego educativo sobre grafos?

1.3. Hipótesis

H1: Un videojuego que utilice conceptos relacionados a grafos puede enseñar sobre los algoritmos de BFS y DFS.

H2: Un videojuego que enseñe grafos será percibido de manera positiva por los estudiantes que todavía no aprenden sobre esos contenidos.

1.4. 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.

1.5. Objetivos Específicos

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

Capítulo 2

Marco teórico

2.1. Metodología de validación de videojuegos educativos

Petri y Gresse von Wangenheim, en su trabajo [46], llevaron a cabo una revisión de la literatura antes de desarrollar el modelo MEEGA+, cuya publicación tuvo lugar en 2018 [47]. En dicha revisión, analizaron la evaluación de videojuegos educativos relacionados con la computación a partir de una muestra de 3617 artículos. Clasificaron los estudios según la metodología empleada en verdaderos estudios, cuasi-experimentales, no experimentales y ad-hoc.

Los estudios que distribuían personas de forma aleatoria en distintos grupos se consideraron experimentales. Si un estudio emplea múltiples grupos o momentos de medida sin asignación aleatoria, se clasificaba como cuasi-experimental. Los estudios que no utilizaban múltiples grupos, pero que se llevaban a cabo de manera sistemática con estudios de casos, se consideraban no experimentales. Por último, los estudios que no se llevaban a cabo de manera sistemática, ni indican cómo se miden resultados, se clasificaban como estudios ad-hoc.

En una reseña de literatura realizada por Calderón y Ruiz [3] destacaron que la mayoría de los estudios sobre videojuegos educativos se centran en la evaluación del aprendizaje, la usabilidad y la experiencia del usuario. Además, observaron que la mayoría de estos estudios se llevan a cabo de manera ad-hoc, sin una sistematización adecuada. Se evidenció que los métodos de validación más comunes son cuestionarios y entrevistas. Asimismo, en su trabajo se propuso una categorización de características que indican la calidad de un juego, la cual fue posteriormente utilizada por los autores de MEEGA+ [47] en su cuestionario. Entre los aspectos evaluados se incluyen: el diseño del juego, la satisfacción del usuario, la usabilidad, la motivación y los resultados del aprendizaje, entre otros.

En su revisión sobre videojuegos serios, Calderón y Ruiz [3] también categorizaron los métodos utilizados para validar estos trabajos en tres tipos: simple, pre/post y pre/post/-post. En el primer enfoque, se llevaba a cabo una sesión de juego y luego se aplicaban los mecanismos de evaluación. En el segundo, se realizaban evaluaciones antes y después del

juego para medir el conocimiento previo y adquirido. En el tercero, además de las evaluaciones pre y post, se llevaba a cabo una prueba semanas después para evaluar la retención. En cuanto a las frecuencias, 50 de 89 de estos estudios utilizaron una metodología simple y el 55% lo hizo con tamaños de muestra inferiores a 40 [3].

Basándose en estas conclusiones, Petri y Christiane Gresse von Wangenheim, en su revisión de literatura sobre videojuegos educativos [46], afirman que la mayoría de los estudios sobre videojuegos educativos se realizan de manera ad-hoc en términos de diseño de investigación, medición, recolección de datos y análisis. Sin embargo, destacan el modelo MEEGA+ [58], indicando que ha sido utilizado en otras áreas distintas de la computación, proporcionando mayor sistematización y confiriendo mayor validez científica a los trabajos basados en este modelo.

2.2. Teoría de Respuesta al Ítem (IRT): Intuición Cuantitativa

La Teoría de Respuesta al Ítem (IRT) es un modelo estadístico utilizado en pruebas globales, como exámenes de inglés como lengua extranjera y programas de evaluación internacional de estudiantes. Este modelo destaca por su capacidad para evaluar detalladamente propiedades estadísticas de cada ítem (o pregunta) en términos de dificultad y capacidad de diferenciación [60, 54].

Los modelos basados en IRT parten de tres suposiciones fundamentales. Primero, establecen una relación entre el rasgo latente que se busca medir y la probabilidad de responder un ítem en una categoría específica, como “en desacuerdo” o “de acuerdo” [25].

El segundo supuesto es que existe una escala continua y unidimensional de habilidad, denotada como θ . Aunque esta suposición es fuerte, hay técnicas para verificar la unidimensionalidad de los datos de prueba [54]. Para medir múltiples características simultáneamente, existen modelos IRT multidimensionales [49].

La tercera suposición es la independencia local, que establece que las personas responden de forma independiente a cada ítem o pregunta, sin que la respuesta a una influya en la respuesta a otra [25].

Por definición, θ está en el rango $]-\infty, \infty[$, pero prácticamente la totalidad de los datos está en el rango aproximado de $]-3, 3[$. Un valor $\theta = 0$ se asume como un nivel promedio [54].

El modelo de 2 parámetros (2PLM o Two-Parameter Logistic Model) considera la dificultad y la discriminación (difficulty & discrimination) de cada ítem. La dificultad se refiere a qué tan difícil es un ítem en relación con otros; mientras que la discriminación se refiere a la capacidad del ítem para distinguir entre participantes con diferentes niveles de habilidad.

Existen distintos modelos logísticos para evaluar θ . El que se usa en este trabajo es el modelo logístico de 2 parámetros y otra variante de tres parámetros. Los parámetros en estos

casos buscan representar rasgos asociados a cada ítem o pregunta. El de 2 parámetros incluye dificultad y discriminación, aunque en español se podría interpretar mejor como distinción o graduación, asociado a indicar la probabilidad entre elegir un valor u otro [25]. El valor de θ se puede aproximar utilizando el método bayesiano EAP (Expected A Posteriori), el cual se calcula utilizando los paquetes de R *mirt* [11] y *mirtCAT* [12] aplicados en este trabajo.

El modelo 2PLM se compone del parámetro dificultad b_j y de discriminación a_j asignados para cada pregunta. Este último representa la pendiente de la curva e indica en qué medida el ítem diferencia a los examinados con un nivel en el rasgo latente por encima o debajo del parámetro de dificultad [5]. En un modelo con respuestas correctas e incorrectas, un ítem con un parámetro de dificultad mayor b_j será respondido incorrectamente con mayor probabilidad para casi cualquier nivel de θ [54].

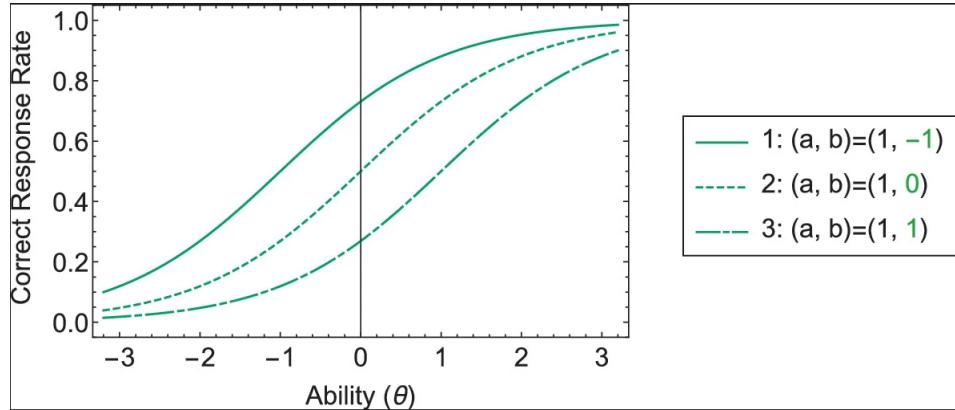


Figura 2.1: Probabilidad de responder correctamente para distintos valores del parámetro dificultad b . Fuente: [54].

Es importante destacar que cada ítem discrimina mejor en torno a valores de θ que sean cercanos al parámetro de locación b . Además, un parámetro a indica que el ítem es un mejor indicador para θ , pero teniendo en consideración que este poder discriminativo funciona mejor cuando $\theta \approx b$. Por esta razón, cada pregunta por separado entrega información acerca del valor θ que se quiere obtener, de manera que las preguntas deben estar formuladas de tal manera que permitan distinguir distintos niveles del rasgo a evaluar [5].

Aplicando este modelo a una escala de Likert de formato ordinal, para cada pregunta se indican 4 parámetros b , para diferenciar entre los niveles 1) Muy en desacuerdo; 2) En desacuerdo; 3) Ni en desacuerdo ni de acuerdo; 4) De acuerdo y 5) Muy de acuerdo [5, 58]. De esta manera, a medida que θ es mayor, la calidad de juego es mejor según [58]. Un juego de buena calidad tenderá a recibir respuestas positivas, como “Muy de acuerdo”. El script de R utilizado para el cálculo de θ se encuentra disponible en [58].

Las fórmulas empleadas, demostraciones relacionadas con IRT o mostrar cómo afectan los distintos parámetros al resultado final está fuera del alcance de este estudio, principalmente porque los paquetes de R se encargan de realizar este trabajo. Para entender mejor cómo funciona este modelo por debajo, puede consultarse [54, 25].

2.3. Motores de videojuegos: Godot

Existen diversos motores de videojuegos, como Unreal Engine [28], Unity [57] y Godot [23]. Estos programas ofrecen una ventaja significativa al proporcionar un entorno integrado que abarca diversas funcionalidades para diferentes profesionales. Por ejemplo, facilitan la integración de animaciones, efectos visuales y de sonido. Además, incluyen bibliotecas incorporadas comúnmente utilizadas en videojuegos tales como: colisiones, texturizado o composición de elementos; permitiendo la unificación de la representación visual, auditiva y programática de un personaje. Los motores de videojuegos también posibilitan la exportación de la aplicación a diversas plataformas sin necesidad de modificar el código [21]. Con todo, Godot destaca por ser completamente gratuito, de código abierto y poseer una comunidad activa en el desarrollo del motor [24].

Godot emplea principalmente dos lenguajes de programación: GDScript, similar a Python, y C#. Ambos fueron utilizados en este trabajo. GDScript permite prototipar rápidamente debido a su simplicidad y estrecha integración con el motor. Por otro lado, C# es más rápido, pero algunas características del motor no están completamente integradas en la versión de Godot utilizada durante este trabajo. Se utilizaron dos versiones, se partió en la 3.5.2 y se terminó usando la 3.6 [22].

Capítulo 3

Diseño de IGACSE

La aplicación IGACSE (Interactive Graph Algorithms for Computer Science Education) fue concebida en función de ciertos requisitos y necesidades identificados entre los estudiantes de informática.

3.1. Proceso de diseño de IGACSE

El proceso de diseño de IGACSE siguió una estructura basada en cuatro etapas: (1) Identificación del problema; (2) Diseño de la solución; (3) Programación de la solución y (4) prueba. Tras cada prueba, la propuesta de diseño se modifica, para ejecutar esos cambios posteriormente en el código y ponerlos a prueba nuevamente (Ver figura 3.1). Si las modificaciones generaban mejoras en la aplicación, se mantenían, de lo contrario se mantenía la versión original.

3.1.1. Identificación del problema

Previo a las decisiones de diseño que dan origen a las mecánicas de juego, Según Rogers en su libro sobre diseño interactivo [50], es crucial identificar el problema que se busca resolver, qué usuarios se ven afectados por este problema, qué características poseen y cómo podría resolverse en base a prototipos.

El diseño del videojuego surge a partir de la identificación de un problema recurrente entre los estudiantes de computación: la dificultad para comprender completamente los algoritmos enseñados. Esta falta de comprensión, a menudo no reconocida por los propios estudiantes, se manifiesta en la incapacidad para reproducir los pasos de los algoritmos de manera precisa, especialmente en situaciones como exámenes o tareas donde se requiere seguir instrucciones al detalle.

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. Al

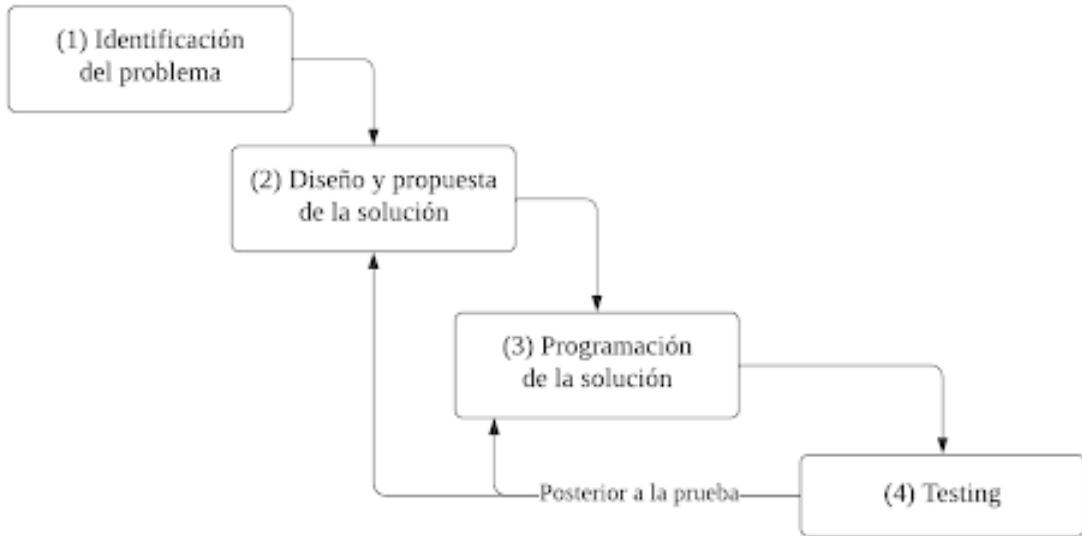


Figura 3.1: Proceso de desarrollo iterativo llevado a cabo para crear IGACSE.

asignarles como ejercicio realizar el procedimiento siguiendo cada instrucción rigurosamente, los estudiantes no lo llevan a cabo o lo hacen de forma incorrecta sin percatarse de sus errores. Un estudio de Zingaro et al. [71] corrobora esta situación, indicando que los estudiantes a menudo creen entender los contenidos, pero presentan conceptos erróneos sin ser conscientes de ellos.

El trabajo de Zingaro et al. [71] ejemplifica estos malentendidos, mencionando errores al comprender estructuras de datos como los heaps y las formas en que pueden representarse o construirse.

A un nivel más particular, en la Universidad de Chile, tanto históricamente como hasta la actualidad, para obtener la licenciatura en computación se exige aprobar dos años de cursos de plan común. Durante este período, los estudiantes practican una técnica conocida como “pauteo”, que implica estudiar y revisar pautas de pruebas y tareas de semestres anteriores para prepararse para los exámenes. Esta práctica lleva a los estudiantes a centrarse en la revisión de los procesos para responder preguntas, sin necesariamente realizar los ejercicios o repetir los pasos de manera detallada, lo que resulta en un aprendizaje superficial de los contenidos.

La práctica del “pauteo” resulta especialmente perjudicial en el área de grafos y sus algoritmos. Si bien los estudiantes pueden captar los conceptos básicos o memorizar las ideas generales explicadas en clase, a menudo no logran reproducir paso a paso algoritmos como BFS y DFS, lo que limita su comprensión profunda de la materia.

Para abordar este problema, se propone el desarrollo y evaluación de una herramienta que, mediante una interfaz visual interactiva, proporcione una experiencia lúdica y atractiva para el usuario, facilitando la comprensión y reproducción paso a paso de los algoritmos.

Cabe señalar que, para enseñar los conceptos relacionados con programación, el escenario

ideal consta de usuarios que tengan nociones de programación, pero que carezcan de años de experiencia, lo suficiente para que no puedan comprender un algoritmo a cabalidad con solo ver el código. Por lo mismo, se propone utilizar un modelo similar a los depuradores o debuggers, donde se puede ver el estado de las variables en cada paso, así como la instrucción por ejecutarse y el resultado que conlleva.

3.1.2. Diseño y propuesta de la solución

La fase inicial del diseño involucró la creación de diversos bocetos de interfaz, basándose en ciertos referentes mencionados en este trabajo. Se presentó semanalmente el trabajo de tesis al curso “CC7970 - Trabajo de Tesis I” compuesto por un grupo de 20 estudiantes de posgrado de ciencias de la computación. Los alumnos del curso daban sus comentarios en cada clase. Tras analizar las opciones, se seleccionó un diseño que destacaba por su claridad, atractivo y similitud con el entorno de desarrollo Visual Studio Code.

Como requisitos se establecieron los siguientes aspectos: el videojuego creado debe evitar la mecanización por parte del estudiante, instándolo a revisar de manera consciente y exhaustiva cada paso relacionado con el algoritmo, y que no dé por sentado que se entiende el código y saltar a la siguiente actividad; como punto de referencia visual, el trabajo se inspira parcialmente en el debugger (depurador) de Visual Studio Code. Esta herramienta permite visualizar los valores de las variables en cada momento, además de avanzar paso a paso a lo largo del código, facilitando la comprensión de los algoritmos de manera detallada.

Con lo anterior, se implementaron diversas versiones del juego utilizando Godot, un motor de juegos que permite agregar nuevas funcionalidades de manera rápida y flexible, manteniendo un diseño visual consistente.

Además, para validar la efectividad del juego, se realizaron pruebas con usuarios expertos. Estos experimentaron dificultades para seguir y comprender los algoritmos presentados, lo evidenció que el juego no alcanzaba su objetivo principal con personas sin experiencia previa en grafos. Ante este hallazgo, se recurrió a la asesoría directa de diseñadores de videojuegos, expertos en diseñar interfaces de usuario (UI) y de experiencia de usuario (UX).

Los diseñadores asesores sugirieron la inclusión de tutoriales que mostraran los elementos del juego de manera gradual. Adicionalmente, se propuso incorporar una narrativa de fondo que permitiera a los jugadores identificarse con el juego y comprender la utilidad práctica de los algoritmos y estructuras que estaban aprendiendo. Se enfatizó la importancia de agregar elementos de premiación y ludificación para incrementar la motivación y el compromiso de los usuarios con el juego.

Tras finalizar los tutoriales e integrar los estilos, la música y los elementos narrativos, se procede a publicar el videojuego en la plataforma Itch.io, un sitio web dedicado a la venta de videojuegos y elementos relacionados, especialmente de creadores independientes o con recursos limitados.

3.1.3. Referentes

Los referentes más importantes son aquellos videojuegos que tienen un rol educativo relacionado a la programación, como 7 Billion Humans [15] (Figura 3.2) y CodeCombat [1] (Figura 3.3).



Figura 3.2: Videojuego 7 Billion Humans

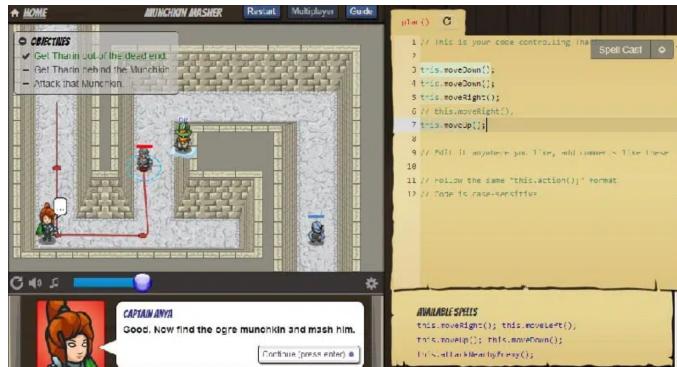


Figura 3.3: Videojuego CodeCombat

Estos juegos presentan ventanas de texto prominentes en la interfaz principal, que ocupan al menos el 20 % de la pantalla. Además, incluyen diálogos que contribuyen a una narrativa continua, situados en la parte inferior de la ventana gráfica al inicio y al final de cada nivel. En ambos casos, la interfaz de código se encuentra en el lado derecho, lo que permite al usuario observar cómo la ejecución de las instrucciones afecta al juego paso a paso.

A partir de este análisis, se concluye que es requisito del juego mostrar la ejecución del código instrucción por instrucción.

Adicionalmente, se revisaron canales de YouTube educativos como 3Blue1Brown [52] (Ver figura 3.4), donde el propietario del canal, Grant Sanderson, habla repetidamente sobre cómo las visualizaciones ayudan a comprender la abstracción matemática subyacente.

3.1.4. Diseño de la interfaz de usuario

La interfaz de usuario está basada principalmente en el depurador (debugger) de Visual Studio Code [43] (Ver figura 3.5). Se observa aquí que la instrucción actual está destacada a

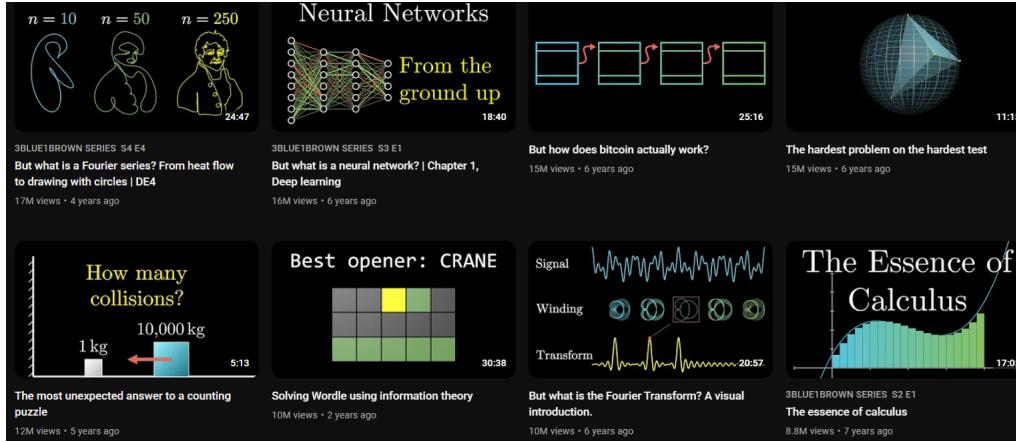


Figura 3.4: Thumbnail del canal de Youtube 3B1B de Grant Sanderson. Aquí se muestra cómo explica conceptos abstractos con visualizaciones y animaciones [52].

través de un puntero y un énfasis con color. Además, las variables locales muestran su valor en un formato “nombre: valor”. El usuario puede avanzar paso a paso en el código.

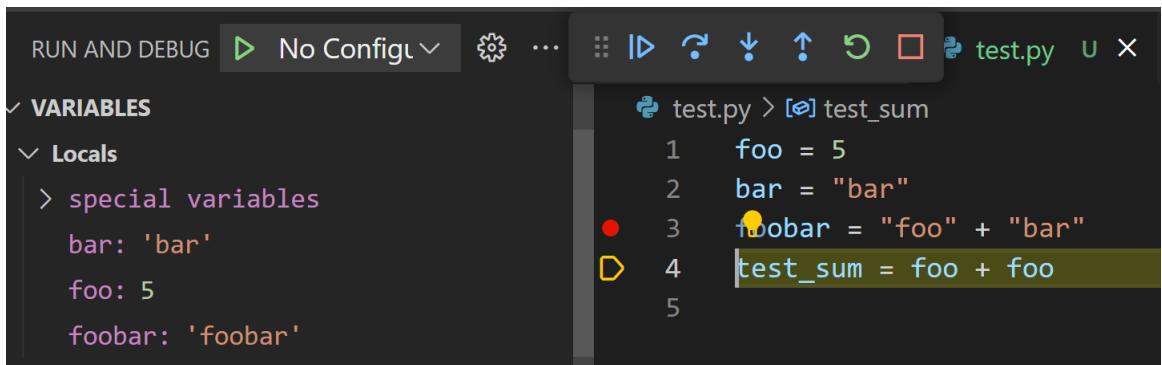


Figura 3.5: Visual Studio Code Debugger.

Hay otros juegos cuya interfaz sirvió de inspiración para la realización del juego final, como 7 Billion Humans [15], 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 IGACSE.

Otro videojuego con una interfaz similar es CodeCombat [1]. Aquí el código también está a la derecha, mientras que abajo se ven diálogos y elementos seleccionables. En el caso de IGACSE, 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 3.3).

En resumen, se observa que existen videojuegos educativos que dedican una gran parte de su interfaz a mostrar código al lado derecho, como CodeCombat y 7 Billion Humans, además de mostrar el estado del juego en la parte inferior. En el caso de IGACSE, se hizo lo mismo como se puede ver en la figura 3.6.



Figura 3.6: Interfaz del nivel BFS de IGACSE. A la derecha se observa el código del algoritmo, con un especial énfasis en la línea de código actual. Por otra parte, en el lado inferior de la pantalla se ve el estado de las variables.

3.1.5. 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 [14], ProtoCorgi [36], Pokemon Emerald [64], Space Invaders [65], y Final Fantasy IV [63].

En juegos como Pokemon [64] y Golden Sun [14], 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. IGACSE 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 IGACSE emplean tonos graves con desvanecimiento, generando una sensación de vibración (ver figura 3.8). 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 3.7: Espectro de Ondas del sonido de Confirmación de IGACSE

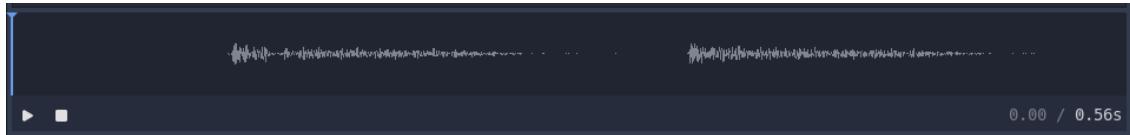


Figura 3.8: Espectro de Ondas del sonido de Error de IGACSE

3.1.6. 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 (Ver figura 3.9). 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 (Ver figura 3.10).



Figura 3.9: El jugador está en la instrucción `if u.is_not_explored():`: El usuario debe responder correctamente sí o no. En caso de respuesta correcta, la instrucción se desbloqueará y podrá pasar a la siguiente instrucción.

Para avanzar en las instrucciones, el jugador debe presionar la tecla espacio. Si esta se presiona sin completar la instrucción, se activará un sonido de error y el color de la instrucción



Figura 3.10: El jugador está en la instrucción `if u.is_not_explored()`: después de responder correctamente, la instrucción cambia de color. Ahora se puede pasar a la siguiente instrucción.

parpadeará entre rojo y amarillo durante un segundo. Presionar la tecla espacio después de completar la instrucción generará un sonido de confirmación y avanzar 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 3.11.

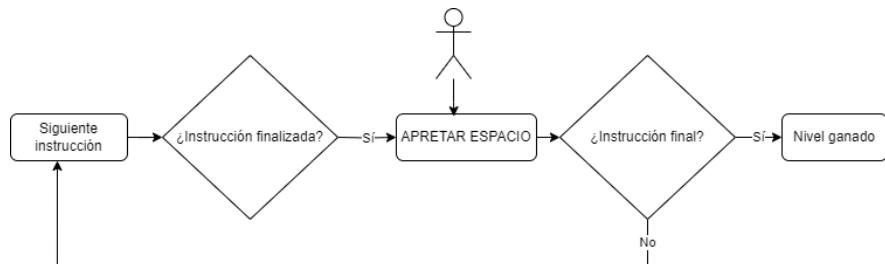


Figura 3.11: Flujo de mecánicas de un nivel jugable. El jugador debe seguir las instrucciones paso a paso. Fuente: Elaboración Propria.

3.2. Descripción de IGACSE

3.2.1. Flujo de juego

El juego comienza presentando el menú principal, donde se pueden elegir dos opciones: el modo historia o los niveles jugables. Si se opta por el modo historia, se accede a los tutoriales. Una vez completados los tutoriales, se desbloquea el primer nivel jugable, que corresponde

al algoritmo DFS. Por otro lado, si se eligen los niveles jugables desde el menú principal, se muestra una lista de niveles disponibles, entre los que se encuentra el algoritmo BFS y DFS. Al completar cada uno de estos niveles, se muestran los créditos del juego (Ver figura 3.12).

El videojuego se divide en distintas escenas, que incluyen el menú principal, los tutoriales y los niveles jugables. En el menú principal, los jugadores pueden seleccionar el nivel que desean jugar o acceder al modo historia. Los tutoriales están diseñados para introducir conceptos básicos de grafos sin mencionar explícitamente el término "grafo", con el objetivo de no interrumpir la narrativa. Además de los conceptos teóricos, los tutoriales enseñan habilidades de juego, como explorar o seleccionar nodos, ejecutar instrucciones de código y responder preguntas de tipo Sí/No relacionadas con las estructuras de control, como las instrucciones condicionales (if).

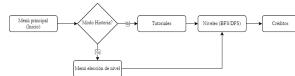


Figura 3.12: Flujo de juego de IGACSE. Fuente: Elaboración propia.

Finalmente, el juego presenta los niveles, que corresponden a los algoritmos que se desean enseñar: BFS (Breadth First Search) y DFS (Depth First Search). En estos escenarios, no hay avance en la narrativa ni guía para el usuario. Una vez que se completan, se muestran los créditos del juego.

El juego se enfoca en la exploración de 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, que corresponde a un nivel del juego. En cada nivel, se enseña un algoritmo diferente. Se utiliza una narrativa ficticia para ayudar en la comprensión de los conceptos de grafos.

Durante el juego, se proporcionan instrucciones en el lado derecho de la pantalla, a medida que el jugador avanza presionando la tecla espacio. Cada vez que se avanza en las instrucciones, se desencadenan efectos y animaciones que muestran visualmente lo que está sucediendo en el código. Estas animaciones se reflejan en la ventana de juego, donde se destacan los planetas, los caminos o las naves que se desplazan entre planetas.

Siguiendo las instrucciones paso a paso, el jugador repite el algoritmo, lo que le permite entender cómo funciona y cuál es su finalidad. Las acciones del jugador pueden ir acompañadas de sonidos de recompensa, que indican el avance a la siguiente instrucción. En caso de cometer un error, se notifica al jugador mediante una animación visual y un sonido. En ciertos casos, si el jugador no sabe qué hacer, se muestra un hint visual que le indica qué tecla debe presionar o qué acción debe realizar.

3.2.2. Narrativa

Para la visualización de algoritmos como el BFS y DFS, ampliamente conocidos como algoritmos de búsqueda de rutas, se realizan representaciones que requieren del enrutamiento

de redes, mapeo de ubicaciones, telecomunicaciones y la navegación [56]. Estas representaciones ayudan en la comprensión de los conceptos de grafos y establecen las bases para la narrativa ficticia que acompañará al usuario durante el desarrollo del videojuego.

Es por ello que el juego se enfoca en su utilidad más común, utilizando como metáfora narrativa la exploración de planetas a través de rutas, donde los planetas representan nodos y los caminos entre ellos son aristas. El objetivo final del juego es rescatar pandas rojos dispersos en los planetas. Para lograrlo, es necesario explorar todos los nodos en una galaxia, que corresponde a un nivel del juego. En cada nivel, se enseña un algoritmo diferente.

La narrativa se desenvuelve en el espacio exterior, donde una nave entabla un diálogo con una estación llamada DCC. La misión de la nave es rescatar pandas rojos dispersos en los planetas de la galaxia. Inicialmente, 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 continúe con los siguientes niveles y complete el juego.

Es relevante destacar que la estación se llama DCC para aumentar el sentido de identificación con el jugador, ya que DCC son las siglas del Departamento de Ciencias de la Computación. Además, el panda rojo es la mascota oficial 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 usuario se sienta identificado con el juego.

3.2.3. Menú principal

En el menú principal, se puede seleccionar el idioma presionando el botón en la esquina superior izquierda de la figura 3.13. Los idiomas disponibles son inglés y español. Además, ofrece las opciones para jugar en el modo historia o los niveles jugables.



Figura 3.13: Menú principal de IGACSE, que se muestra al iniciar el juego.

3.2.4. Tutoriales

Los tutoriales persiguen dos propósitos: familiarizar al jugador con las mecánicas básicas que se emplearán a lo largo del juego y sumergirlo en la trama narrativa. Las mecánicas se introducen a través de una combinación de elementos que incluyen texto descriptivo y pistas visuales, tales como la representación de un ratón realizando un clic izquierdo o la indicación de una tecla, como la “R”, siendo presionada sobre un planeta. La trama está diseñada para exemplificar la aplicación de conceptos relacionados con los grafos sin entrar en detalles explícitos sobre el contenido que se está enseñando. Es importante destacar que cada nivel y tutorial son concebidos como elementos repetibles, lo que permite al jugador revisarlos en caso de no comprender algún aspecto específico.

Se pueden ver imágenes y explicaciones de cada tutorial en el Anexo sobre tutoriales anexo H.

3.2.5. Niveles

Los niveles 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 BFS, DFS, 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.

Se espera que en posibles trabajos futuros se agreguen estos niveles, incluyendo más desafíos y estudios de usabilidad respecto a qué mecánicas podrían generarse para ordenar los arcos de un grafo por orden de pesos, o cómo operar con conjuntos. El código de estos niveles se encuentra disponible en el repositorio de GitHub [59].

3.3. 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.

3.3.1. 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 suelen ser los niveles en un juego. Las escenas tienen un grafo de escena compuesto por uno o más nodos (Ver figura 3.14).

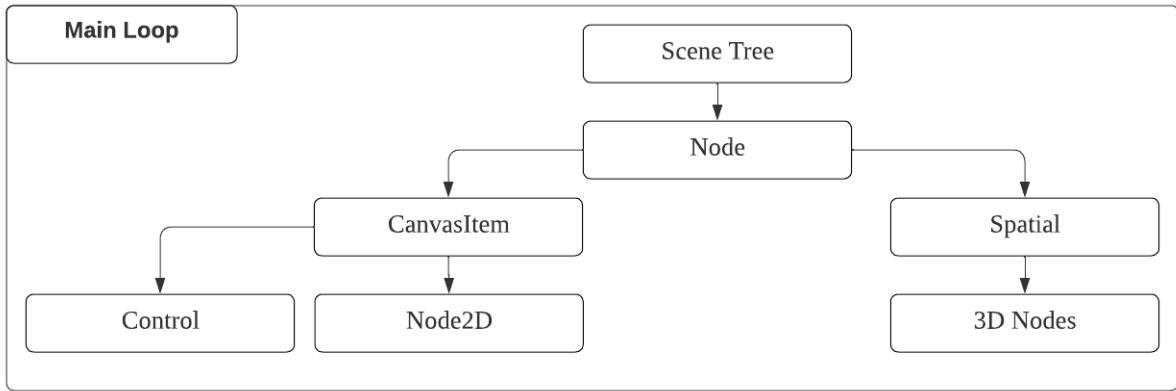


Figura 3.14: Estructura del Grafo de Escena de Godot. Cada elemento es un nodo, que puede ser de interfaz (control), 2D o espacial y 3D. En el caso de IGACSE, todos los elementos son de control o un Nodo 2D.

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 favorable 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.

En Godot existe un ciclo principal, que itera sobre cada nodo, permitiendo que estos manifiesten sus propiedades. Por ejemplo, un nodo de imagen se puede asociar a un nodo de posición 2D. Si el nodo de posición 2D se mueve, veremos que la imagen se mueve de la misma forma. Además, se puede asociar un nodo de sonido para hacer que se emitan sonidos desde esa posición.

Esta lógica de asociación de nodos permite crear componentes rápidamente, lo que permite reutilizar código y características asociadas, como imágenes, colores y sonidos. Como Godot es de código abierto, crear nuevos nodos con sus otras propiedades resulta accesible para programadores con distintos niveles de experiencia.

Una ventaja de usar Godot frente a bibliotecas de computación gráfica como PyGame u OpenGL 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 [13].

Al 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 en el desarrollo [32].

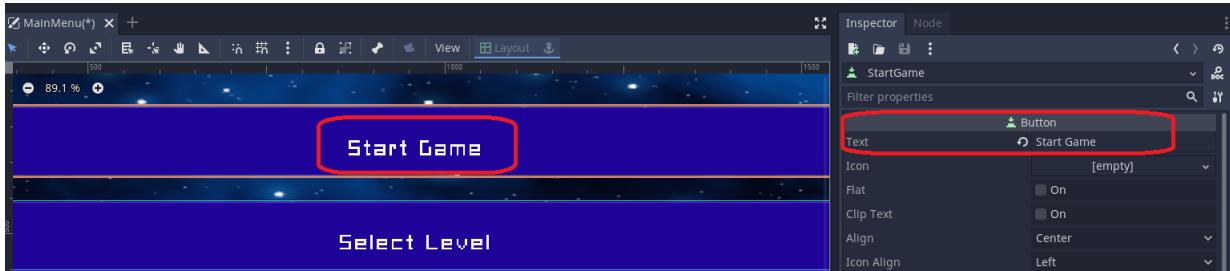


Figura 3.15: Variables exportadas de Godot. En este caso se muestra el menú principal al inicio del juego. Aquí, el botón destacado muestra el texto Start Game. A la derecha se muestra la variable exportada Texto. Esta se puede modificar desde el Editor de Godot.

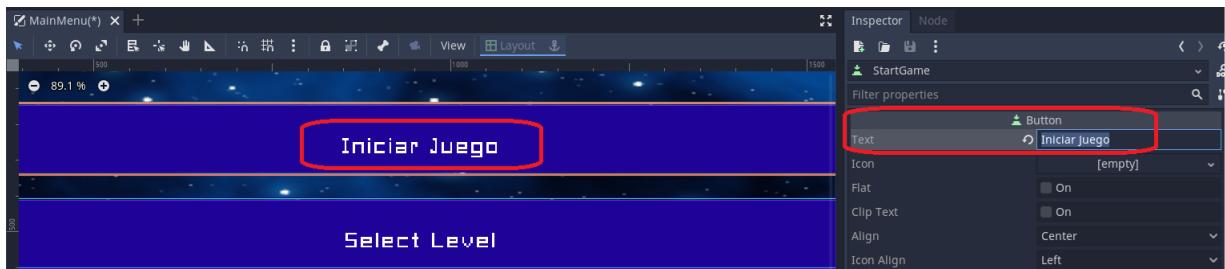


Figura 3.16: Siguiendo la imagen 3.15, el texto se modificó de Start Game a Empezar Juego. El cambio se observa de forma inmediata en la ventana, lo que permite anticipar cómo el usuario verá la escena.

3.3.2. Arquitectura de software de la aplicación IGACSE

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 3.17).

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 arcos ya están establecidos de antemano.

En cuanto a los niveles jugables, hay varios elementos destacables, como el código, las variables, la variable seleccionada y la ventana de juego. El grafo se representa combinando dos tipos de nodos de Godot: "Arcos" y "Nodos de Grafo".

A modo de ejemplo, en la figura 3.18, se muestra que el grafo del juego se representa

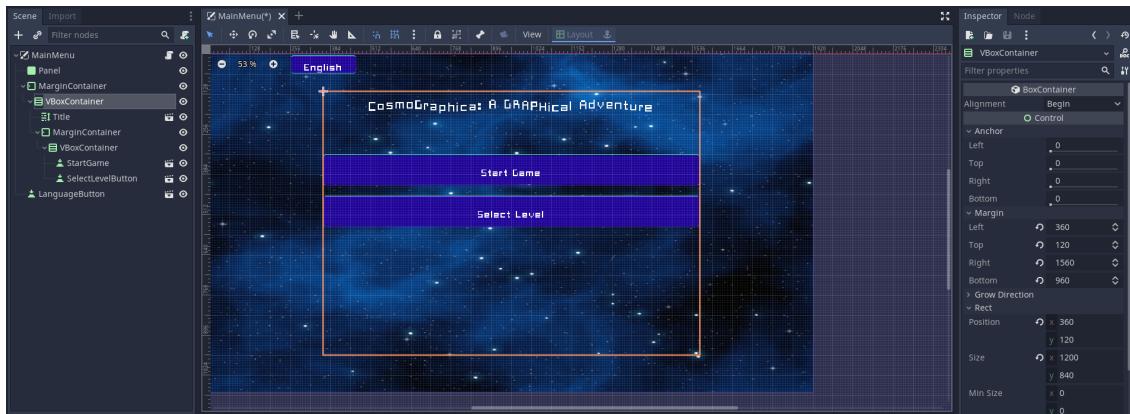


Figura 3.17: Interfaz de Godot describiendo la escena del menú principal de IGACSE. El juego públicamente se dio a conocer como CosmoGraphica, pero el proyecto completo se llama IGACSE.

en el grafo de escena en el lado izquierdo, con los nodos llamados Start, Planet2, Planet3 y StartingNode1. Los arcos se representan con los nodos Edge13, Edge1Star y Edge2S.

En la figura 3.18, el arco Edge1Star está seleccionado, por lo que se muestran sus variables exportadas en el lado derecho de la pantalla. En esta sección, sale destacada el área que indica: NodeA: StartingNode y NodeB: Star. Esto genera un arco entre estos dos nodos. De esta manera, se puede crear la disposición de arcos y nodos sin necesidad de generarlos en código directamente, lo que también permite posicionar de manera predeterminada cada nodo y ver estos cambios en tiempo real.

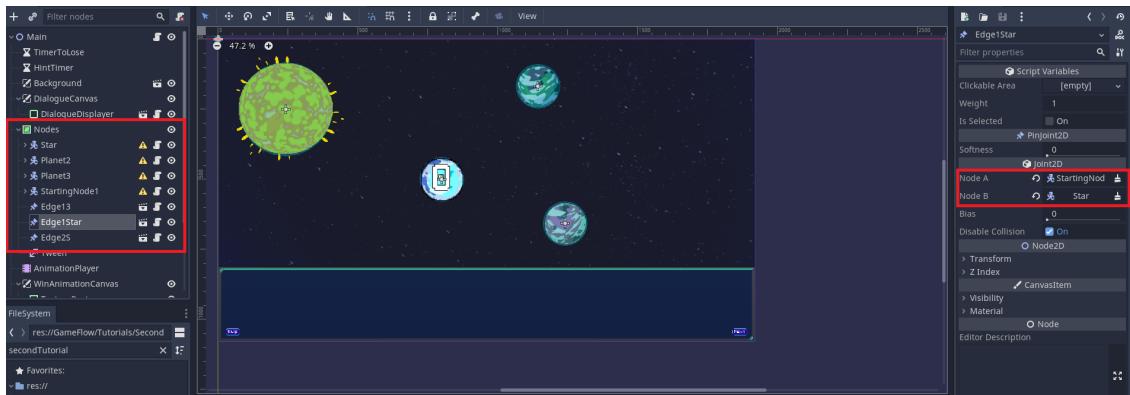


Figura 3.18: Interfaz de Godot describiendo el segundo tutorial. En rojo se observa el grafo de escena con los nodos y los arcos ya predefinidos a la izquierda. En la derecha se muestra cómo se unen los arcos, indicando los dos extremos del arco a través de variables exportadas de Godot.

Respecto a la arquitectura de software presente en los niveles jugables, se observan cinco secciones en el nivel (Ver figura 3.6). Estas son: (1) El menú de opciones arriba a la izquierda; (2) La ventana de juego al centro de la pantalla; (3) El bloque de código a la derecha de la pantalla (CodeContainer); (4) Las variables existentes y sus valores en la parte inferior izquierda (DebugBlock) y, (5) La representación visual de la variable seleccionada actualmente, presente en la esquina inferior derecha de la pantalla (ADTShower).

El bloque de 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. La clase CodeContainer posee un arreglo de objetos del tipo CodeLine llamado code_lines (figura 3.19).

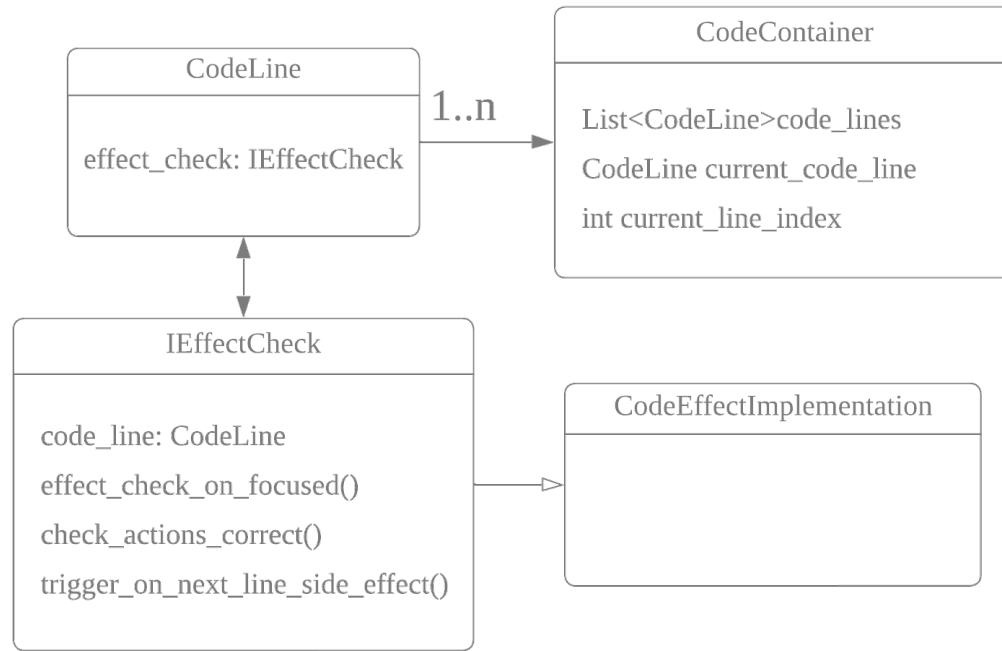


Figura 3.19: Arquitectura de software encargada de la mecánica de ejecución de código dentro del juego.

Durante el juego, respecto a cómo avanzar en el código, el usuario debe ejecutar las acciones requeridas por el código correctamente. Si estas son correctas, el usuario pasa a la siguiente instrucción, desbloqueando nuevas tareas.

La acción que debe ejecutar el usuario está contenida entre las funciones `effect_check_on_focused()` y `check_actions_correct()` (Tabla 3.1). Estas dependerán de cada línea de código. Por ejemplo, en la figura (3.6) que muestra el código de BFS, la primera línea dice `t = starting_node`. En este caso, el jugador debe hacer click en el nodo 0.

La clase CodeLine contiene la lógica de una única línea de código. Entre sus funciones se encuentran: dar énfasis visual a la línea de código actual, proporcionar feedback audiovisual al usuario cuando la instrucción de la línea se completa correctamente y verificar constantemente si la instrucción de línea se ha completado correctamente (ver figura 3.19).

Cada CodeLine tiene asociado un script que puede ser especificado 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 3.20, se observa que la variable “q” está seleccionada.

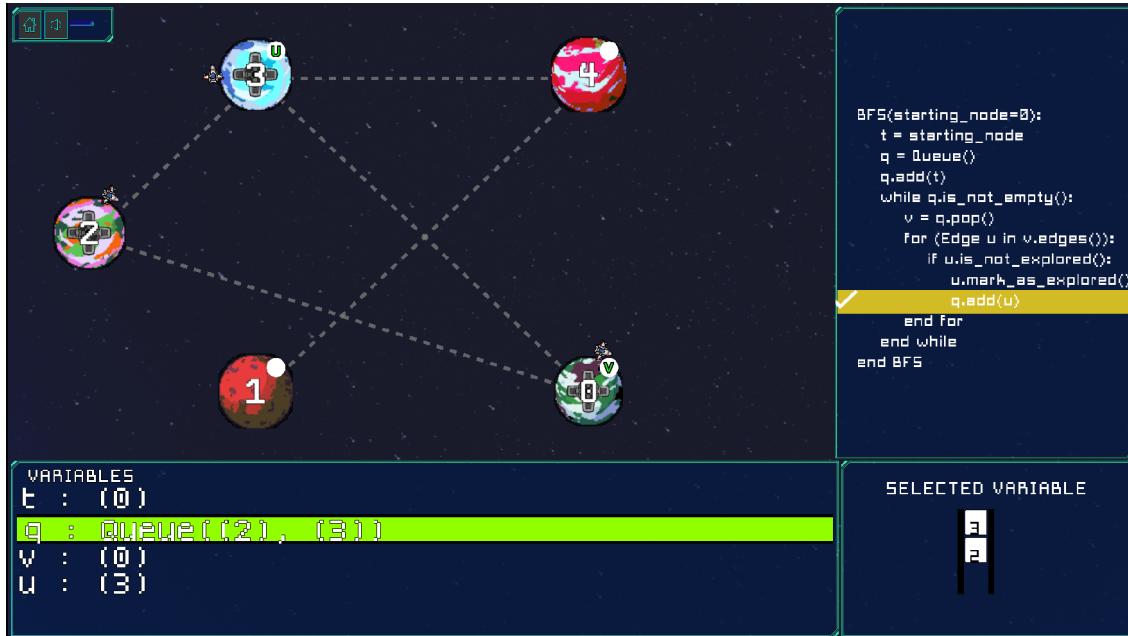


Figura 3.20: Nivel jugable BFS. Se observa el código a la derecha, el grafo en el centro, el DebugBlock en la parte inferior izquierda y el ADTShower en la esquina inferior derecha. La variable q está seleccionada.

La clase ADTShower se encarga de mostrar en la esquina inferior derecha la variable que ha sido seleccionada por el usuario. Esta visualiza al objeto, enfatizando cuando un nodo se agrega o elimina de una cola o pila, lo cual depende del algoritmo que se esté enseñando, BFS o DFS, respectivamente.

Inicialmente, cada vez que se creaba un nuevo script por una nueva línea de código se debían modificar las clases DebugBlock, ADTShower y StoredData (explicada más adelante). Además, estos efectos debían 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 debe reflejar 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 [26]. 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” [48]. Con esta arquitectura, se envía un mensaje a ADTMediator para solicitar la modificación, creación o eliminación de una variable. Poste-

riormente, esta clase ajusta sus variables internas y luego notifica a las clases DebugBlock y ADTShower qué información mostrar.

Una desventaja de esta arquitectura es que realiza copias innecesarias en cada paso, perdiendo eficiencia. Sin embargo, dado que se trabaja con menos de seis variables, este costo es despreciable a nivel de usuario.

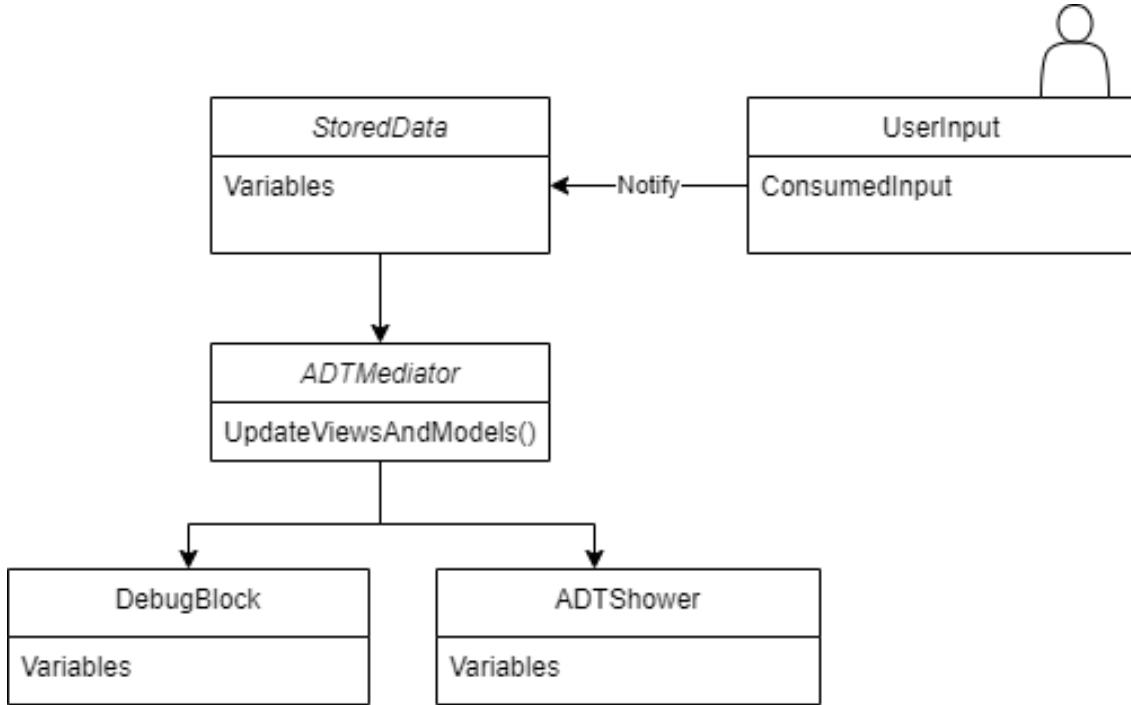


Figura 3.21: Diagrama de un nivel utilizando el ADTMediator, reduciendo el acoplamiento entre clases.

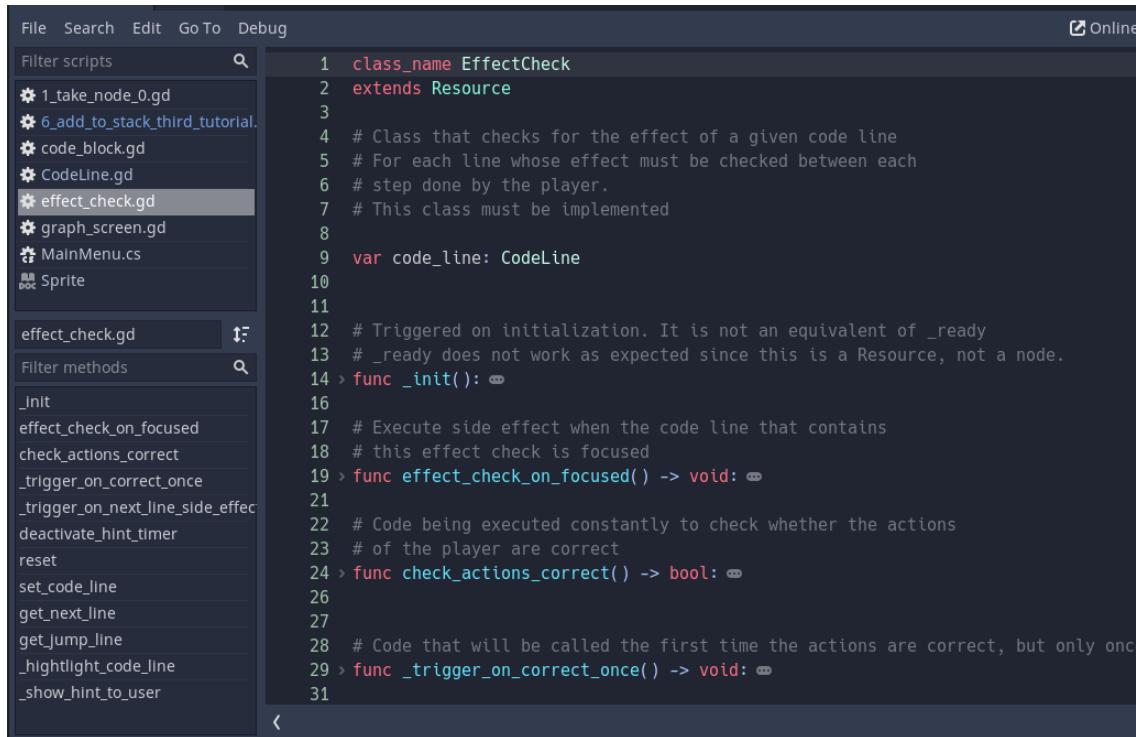
Una clase que sirve de puente para comunicarse con el ADTMediator es **StoredData**. Este es un singleton que contiene datos, tales como los avances del usuario en los distintos niveles y el estado del juego. Cuando una línea de código busca modificar variables dentro del juego, ejecuta un método de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

Otros singeltons 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.

3.3.3. Cómo agregar una nueva CodeLine y ajustarse a otro algoritmo

El framework creado en IGACSE permite la creación de líneas nuevas de código en menos de diez pasos. Facilitando la reproducción de algoritmos de manera precisa. En este capítulo se entrega un ejemplo y muestras de código. En primer lugar, se debe crear el script que ejecuta

una línea de código en el programa. Para esto, se debe crear un nuevo archivo GDScript que herede de la clase EffectCheck como se muestra en la figura 3.22.



The screenshot shows a GDScript editor interface. The top menu bar includes File, Search, Edit, Go To, and Debug. On the right, there's an "Online" button. The left sidebar has a "Filter scripts" search bar and lists several files: 1_take_node_0.gd, 6_add_to_stack_third_tutorial.gd, code_block.gd, CodeLine.gd, effect_check.gd (which is selected), graph_screen.gd, MainMenu.cs, and Sprite. Below this is a "Filter methods" search bar and a list of methods: _init, effect_check_on_focused, check_actions_correct, _trigger_on_correct_once, _trigger_on_next_line_side_effect, deactivate_hint_timer, reset, set_code_line, get_next_line, get_jump_line, _highlight_code_line, and _show_hint_to_user. The main code editor area displays the following GDScript code:

```
1 class_name EffectCheck
2 extends Resource
3
4 # Class that checks for the effect of a given code line
5 # For each line whose effect must be checked between each
6 # step done by the player.
7 # This class must be implemented
8
9 var code_line: CodeLine
10
11
12 # Triggered on initialization. It is not an equivalent of _ready
13 # _ready does not work as expected since this is a Resource, not a node.
14 > func _init(): @
15
16
17 # Execute side effect when the code line that contains
18 # this effect check is focused
19 > func effect_check_on_focused() -> void: @
20
21
22 # Code being executed constantly to check whether the actions
23 # of the player are correct
24 > func check_actions_correct() -> bool: @
25
26
27
28 # Code that will be called the first time the actions are correct, but only once
29 > func _trigger_on_correct_once() -> void: @
30
31
```

Figura 3.22: Script de la clase abstracta EffectCheck.

Los métodos de esta clase que pueden ser redefinidos según cada caso. Estos se pueden ver en la tabla 3.1.

| Método | Acción |
|-----------------------------------|--|
| effect_check_on_focused | Cuando el puntero de instrucciones alcanza esta línea de código, esta función ejecuta un efecto dependiendo del caso, por ejemplo, permitir que un nodo sea seleccionable. |
| check_actions_correct | Verifica constantemente si el jugador ha actuado acorde a la instrucción. Retorna verdadero si el usuario ha completado las acciones relacionadas con esta línea de código. |
| _trigger_on_correct_once | Acción que ejecuta la función cuando la instrucción se ha completado correctamente. Se ejecuta a lo más una vez. Por ejemplo, al presionar un planeta correctamente, deseamos visualizar una nave espacial volando hacia dicho planeta |
| _show_hint_to_user | Qué se muestra cuando el usuario tarda cierto tiempo en ejecutar correctamente las acciones requeridas por la instrucción, como mostrar un ratón haciendo click en un planeta. Cumple el objetivo de ayudar al jugador a avanzar. |
| _trigger_on_next_line_side_effect | Acción que ejecuta la función cuando se llega a la siguiente línea. De manera predeterminada, es apagar el timer para entregar un hint. |
| reset | Acción llevada a cabo cuando se llega la siguiente instrucción. Utilizado en ciclos y condicionales para asegurarse de que cuando se vuelvan a repetir tales instrucciones, la ejecución funcione como se espera. |
| get_next_line | Retorna el índice al que debería avanzar el puntero de instrucciones al terminar la función, si es que no corresponde un salto. |
| get_jump_line | Retorna el índice al que debería saltar el puntero de instrucciones al terminar la función. Utilizada en condicionales y ciclos. Esta función se diferencia de la anterior porque permite saltar líneas. |

Tabla 3.1: Descripción de los métodos que pueden ser redefinidos por las implementaciones de la clase abstracta EffectCheck.

A modo de ejemplo, en la figura 3.23 se muestra la instrucción *If v.is_not_explored()* usada en el algoritmo DFS. Esta instrucción asegura que un mismo nodo no sea explorado más de una vez. Se le pregunta al usuario si se ha explorado el nodo v. Una vez el usuario ha contestado correctamente, se le permite avanzar.

```

1  extends EffectCheck
2  # If v.is_not_explored()
3
4  # Ask the user whether answer is true or false
5 ~ func check_actions_correct() -> bool:
6    # Check if the user gave the right answer
7    > return StoredData.v_is_explored_right_answer
8
9  # Override from EffectCheck
10 ~ func effect_check_on_focused():
11    > ask_user()
12
13 ~ func ask_user() -> void:
14    # Show a popup that asks the user whether v is explored or not
15    # condition is true if not explored,
16    var v = StoredData.get_variable("v").get_node()
17    var explored = not if_condition_is_true(v);
18    NotificationManager.ask_user_if_graph_node_is_explored_dfs(v, explored)
19    # This sets the StoredData.v_is_explored_right_answer variable
20
21  # If v.is_not_explored()
22 ~ func if_condition_is_true(v) -> bool:
23    > if v in StoredData.iterated_nodes:
24      > return false
25
26    > return true
27
28  # If user presses enter

```

Figura 3.23: Script de la instrucción *If v.is_not_explored()*.

Para crear una nueva línea perteneciente a un algoritmo, es necesario generar un nuevo objeto que herede de CodeLine. Posteriormente, se deben redefinir los métodos de esta clase según el comportamiento esperado.

Por ejemplo, en el caso de *If v.is_not_explored()*, es esperable que al usuario se le pregunte si el nodo v está explorado una vez que se alcanza esta parte del código (Ver función *effect_check_on_focused* en la figura 3.23). La instrucción se finaliza cuando el usuario responde correctamente si el nodo v ha sido explorado o no (Líneas 5 a 7 en la figura 3.23).

Respecto a la implementación de una nueva línea de código a nivel de usuario, el proceso es como se indica en la figura 3.24.

Se debe tener un CodeBlock como raíz de un árbol jerárquico. Este CodeBlock debe contener un nodo del tipo VBoxContainer que se encarga de alinear los elementos de forma vertical. Dentro de este VBoxContainer, denominado LinesContainer en este caso, se encuentran los PanelContainers que, en realidad, representan las CodeLines (Ver figuras 3.24).

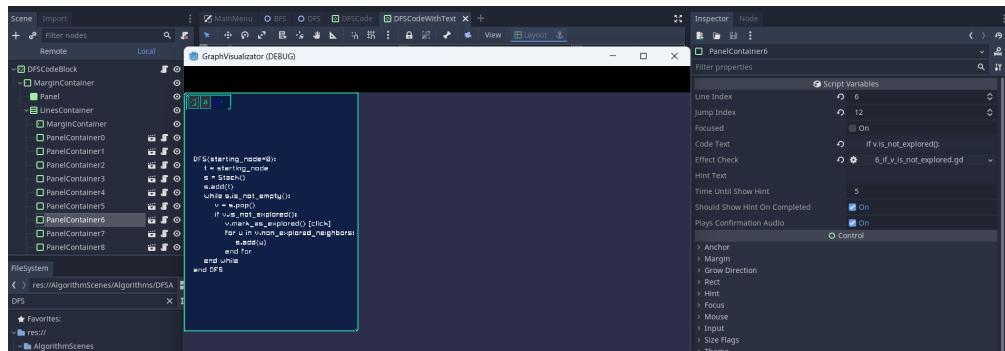


Figura 3.24: Ventana de Godot mostrando un nodo que contiene la instrucción *If v.is_not_explored()*.

Cada CodeLine tiene ciertas variables exportadas, como se observa en el lado derecho de la figura 3.24. Estas variables se pueden ver en la tabla 3.2

| Variable | Descripción |
|----------------------|---|
| line_index | Número de línea que representa la CodeLine. Sirve para indicar la siguiente línea, o a cuál línea saltar. |
| jump_index | Línea a la cual saltar en caso de que corresponda un salto. Utilizada en casos de condicionales y bucles. |
| focused | Su valor es verdadero cuando la línea de código está seleccionada. |
| code_text | El texto del código que se muestra en pantalla. |
| effect_check | Una referencia al script que hereda de EffectCheck y que determina la lógica de lo que hace la línea de código. |
| hint_text | El texto que se le mostrará al usuario si se demora en realizar alguna acción y está atascado. |
| time_until_show_hint | Tiempo que transcurre desde que el puntero de instrucciones apunta a la instrucción y que se muestra el hint. |

Tabla 3.2: Descripción de las variables exportadas de la clase CodeLine.

Cada efecto desencadena modificaciones en las estructuras de datos del nivel. Por ejemplo, al crear la variable v al recorrer un grafo, StoredData guarda una referencia a esta nueva variable, el ADTMediator es notificado y reenvía esta señal a las clases DebugBlock y ADTShower, permitiéndoles desplegar este objeto como un elemento visible en el juego.

3.3.4. Arquitectura de software de un nivel de IGACSE

Un nivel en IGACSE está compuesto por varios nodos de Godot. Los nodos más relevantes y específicos para esta aplicación se detallan en la figura 3.25, que muestra un ejemplo del nivel utilizando el algoritmo DFS.

Godot - Level Scene - DFS

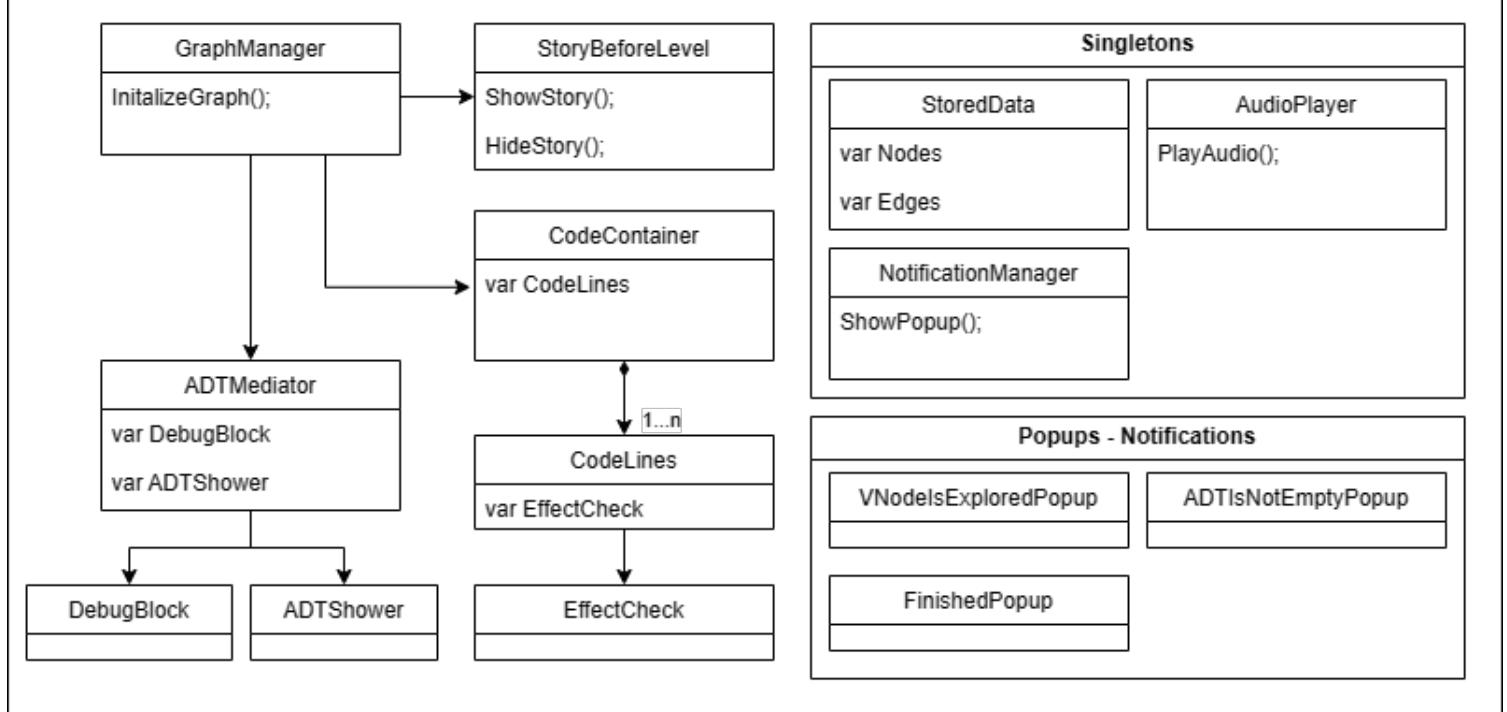


Figura 3.25: Diagrama simplificado de clases e interacciones del nivel DFS. Algunas clases y objetos no están agregados por motivos de simplicidad.

En este caso, los singletons son aquellas clases que se mantienen durante todo el juego. AudioPlayer administra los sonidos y música de fondo del videojuego, mientras que NotificationManager maneja las popups que se cierran y abren durante el juego, relacionadas con menús y preguntas del tipo if.

En cuanto al singleton StoredData, este almacena información sobre los niveles jugados, las acciones realizadas por el jugador, el estado de avance y las variables en juego actualmente. Además, esta clase proporciona una librería de funciones relacionadas con la manipulación de variables durante el nivel, como funciones para obtener los datos relacionados a una variable a partir de un nombre en forma de texto.

La clase GraphManager inicializa el grafo y proporciona las referencias de nodos y arcos a StoredData. También indica a StoryBeforeLevel que debe mostrar un diálogo específico antes de comenzar el nivel jugable.

Posteriormente, se inicializan el ADTMediator, DebugBlock y ADTShower, cuyos roles ya fueron explicados anteriormente. En conjunto con estos objetos, el CodeContainer aparece en el nivel, construyendo cada línea de código con sus respectivos scripts.

El jugador introduce inputs, lo que le permite avanzar a lo largo de las líneas de código hasta completarlas todas. Una vez finalizado el algoritmo, StoredData se encarga de realizar la transición al siguiente nivel.

Capítulo 4

Diseño experimental: Metodología

El propósito de este trabajo fue poner a prueba la hipótesis de que un videojuego educativo puede enseñar algoritmos relacionados con grafos y que puede ser percibido positivamente por parte de los estudiantes.

Para ello, se llevó a cabo un experimento en el que se solicitó a voluntarios que respondieran ciertas preguntas después de probar la aplicación. Este enfoque se considera una metodología simple según [46], es importante destacar que no es el más adecuado para este tipo de estudios, ya que la mayor validez científica se obtiene con una metodología pre/post y un tamaño muestral de 40 individuos o más.

Sin embargo, implementar este tipo de estudios requiere más tiempo por parte de los voluntarios, así como mayores incentivos para su participación y respaldo institucional. Para ejemplificar esto último, la participación en este estudio podría formar parte de una actividad pedagógica en un curso y ser obligatoria.

Se sugieren buenas prácticas metodológicas en [50, 45, 46], donde se establecen ideas primordiales para identificar las dificultades a resolver. Además, se destaca la importancia de realizar validaciones con tamaños muestrales significativos y representativos del usuario al que está destinado el trabajo. Asimismo, se sugiere aprovechar tecnologías como eye-tracking y telemetría. Sin embargo, la obtención de voluntarios para participar en las pruebas fue un desafío, sobre todo considerando que aquellos que participaron en una prueba inicial no podrían formar parte de las pruebas finales, que se consideraban más relevantes.

Se sugieren buenas prácticas metodológicas en [50, 45, 46], donde se establecen ideas primordiales para identificar las dificultades a resolver. Además, se destaca la importancia de realizar validaciones con tamaños muestrales significativos y representativos del usuario al que está destinado el trabajo. Asimismo, se sugiere aprovechar tecnologías como eye-tracking y telemetría. Sin embargo, la obtención de voluntarios para participar en las pruebas fue un desafío, sobre todo considerando que aquellos que participaron en una prueba inicial no podrían formar parte de las pruebas finales, que se consideraban más relevantes.

4.1. Fases del trabajo de investigación

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

4.1.1. Fase exploratoria

En esta primera fase, el objetivo fue recopilar retroalimentación y opiniones de usuarios de manera abierta, utilizando metodologías de pensamiento en voz alta con personas experimentadas en grafos. Se optó por esta metodología por dos razones. En primer lugar, si un usuario conoce el algoritmo y la estructura de datos pero no comprende el videojuego, entonces existen problemas de usabilidad, justificando el enfoque inicial con personas expertas.

En segundo lugar, los usuarios expertos tienden a expresarse de manera más natural cuando pueden emitir opiniones en el momento, por lo que se prefirió evitar encuestas o escalas posteriores a la experiencia. Resulta crucial que los expertos expresen sus impresiones a medida que los elementos del videojuego aparecen en pantalla y no después de la experiencia, para comprender mejor lo experimentado al usar la aplicación por primera vez. Hay animaciones que deben captar la atención en el momento, como la pista visual del ratón indicando al usuario que haga clic izquierdo en un planeta.

Este desarrollo fue iterativo. Primero se hacían cambios a la aplicación basados en los últimos comentarios recibidos. Luego, se le mostraba la aplicación a usuarios expertos, estos daban su opinión y se volvían a aplicar los cambios. Este proceso se hizo 8 veces con el curso “CC7970 - Trabajo de Tesis I”.

4.1.2. Fase de evaluación académica y percepción de usuario

En esta fase, se logró la participación completa de una muestra de 15 personas, a quienes se les ofreció un incentivo monetario para evitar sesgos asociados a la voluntariedad y para aumentar la participación [41, 16].

Para iniciar esta etapa, se realizó una convocatoria voluntaria en el foro de las tres secciones de Algoritmos y Estructuras de Datos de la Universidad de Chile durante las primeras dos semanas del semestre de primavera del año 2023. Se ofreció una remuneración a todas las personas que completaran la experiencia, la cual tenía una duración promedio de 45 minutos.

Las personas que rinden este curso suelen estar en su quinto semestre de la carrera, aunque algunas también lo rinden de forma más tardía en sus carreras, sobre todo cuando son de otras especialidades como Ingeniería Eléctrica o Industrial. Estos suelen tener nociones de programación, como declaración de variables y control de flujo, pero no necesariamente sobre estructuras de datos. Estos estudiantes rondan los 21 a 23 años de edad.

La convocatoria se realizó por dos medios. Por una parte, se escribió un mensaje en la comunidad de Telegram invitando a la gente del curso a participar del estudio, y por otra

parte, el equipo docente de cada sección del curso CC3001 - Algoritmos y Estructuras de Datos, escribió un mensaje en el foro invitando a sus estudiantes a participar.

La experiencia constaba de tres partes: realizar la prueba de usuario, completar un formulario que utilizaba la escala de Likert basado en el modelo MEEGA+ [47], y responder a una prueba escrita basada en exámenes previos del curso CC3001 antes mencionado.

El modelo sistemático MEEGA+ [47] está diseñado para evaluar videojuegos educativos, buscando 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 A.

La medición del rendimiento académico se lleva a cabo mediante 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 anexo B.

4.1.3. Encuesta libre

Con el objetivo de ampliar el estudio y aumentar el tamaño de la muestra, se llevó a cabo una tercera experiencia abierta al público en general con conocimientos en programación. Se emitió una invitación en diversas comunidades de videojuegos para probar el juego educativo y completar el formulario. Dado que las experiencias podían variar significativamente, se incorporó una pregunta sobre el nivel de experiencia en programación para permitir la segmentación. El formulario utilizado también se basó en el modelo MEEGA+, pero las respuestas recolectadas se almacenaron en una base de datos separada del grupo anterior. Aquí se mostraron dos versiones, una en español y otra en inglés para aumentar el tamaño de la muestra. En total, 12 personas participaron de esta experiencia.

Capítulo 5

Resultados

5.1. Resultados durante la fase exploratoria

Inicialmente, tras finalizar el nivel o jugar durante 12 minutos, se pidió a usuarios expertos que evaluaran la usabilidad del juego en una escala del 1 al 10. Se realizaron tres iteraciones utilizando esta metodología, junto con la recopilación de comentarios abiertos.

En la primera fase, la usabilidad promedio entre 4 usuarios fue de 2.5. Posteriormente, después de implementar las sugerencias previas, la usabilidad aumentó a 6 con una muestra de 4 usuarios. Sin embargo, los usuarios señalaron la facilidad para cometer errores y la monotonía del juego, así como la necesidad de leer mucho, lo cual generaba insatisfacción, a pesar de entender las acciones posibles.

Finalmente, antes de probar con el público objetivo, se llevó a cabo una última prueba con una muestra de 6 usuarios, obteniendo un promedio de calificación de 8 sobre 10. En este caso, los usuarios mencionaron problemas menores, principalmente relacionados con la interpretación de algunas animaciones o la falta de indicaciones, como la necesidad de presionar 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. La prueba realizada se encuentra en el anexo B.

5.2.2. Formulario MEEGA+: Percepción de usuario

Según [47] 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 5.1 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.

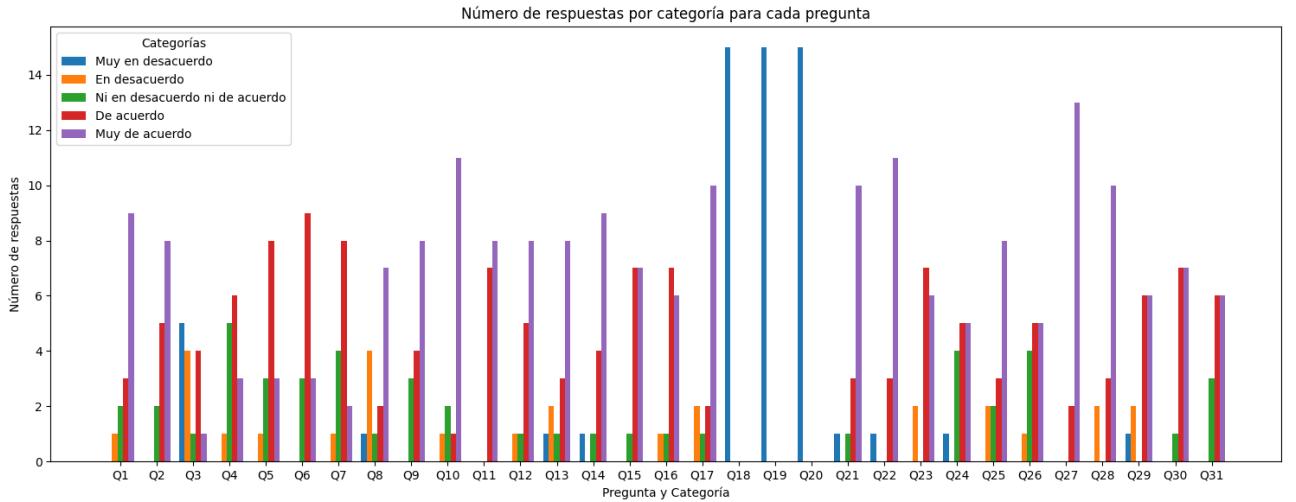


Figura 5.1: Agregación de respuestas por categoría y por pregunta

Por otra parte, en [47] se utiliza Item Response Theory (IRT) para asignar un valor θ 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 [58].

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 en desacuerdo, facilitando la distinción entre opiniones.

Aplicando el script en el anexo E con los parámetros entregados en la página con el manual de uso del modelo MEEGA+ [58], y las respuestas al formulario, se entrega un valor θ que indica la calidad del juego producido y permite clasificarlo. En este caso, el resultado obtenido fue de un $\theta = 0,613$, lo que significa que es un juego de buena calidad, mas no de excelente calidad según [47], pues para lograr la calidad excelente se requiere un valor $\theta \geq 0,65$.

En el anexo Anexo F se muestran los comentarios abiertos que se exigen en la metodología MEEGA+ [47].

5.3. Resultados de la encuesta libre

Esta encuesta permanece abierta en el momento de la redacción de esta tesis. Hasta ahora, se han recolectado 15 respuestas. El resultado de la percepción con la metodología [47] es un $\theta = 0,616$. Los resultados se encuentran en los anexos de forma análoga a las respuestas posteriores. Se observa un valor θ más alto que en el caso anterior. Además, los comentarios abiertos muestran una actitud de mayor aceptación con respecto al grupo que se expuso a la prueba académica. Esto puede atribuirse al sesgo del voluntario [51], puesto que la comunidad que participó de este trabajo es una comunidad de creación de videojuegos, por lo que aprecian cada aspecto de los videojuegos y reconocen la dificultad de crearlos.

Nuevamente, en el anexo Anexo F se muestran los comentarios abiertos que se exigen en la metodología MEEGA+ [47]. Las respuestas están separadas con respecto al grupo anterior.

Capítulo 6

Discusión y Conclusión

6.1. Fortalezas de la metodología aplicada

La principal fortaleza de la metodología empleada radica en la utilización de un formulario estandarizado y previamente validado. Se tiene una alta confianza en que dicho formulario mide de manera efectiva lo que busca evaluar, como se validó a través de la alfa α de Cronbach, medida en el trabajo de Petri et al. [45]. Esta medida indica que las preguntas evalúan efectivamente lo que el cuestionario buscan evaluar [45]. Esta metodología proporciona una valoración numérica de la calidad del juego según las opiniones de los usuarios, representando este valor numérico como un rasgo latente (latent trait) comúnmente denotado como θ en IRT [39, 4].

El proceso de diseño se considera robusto, ya que incorporó retroalimentación de expertos y la aplicación de la metodología de “Think Aloud” en cada iteración. Esto permitió que los jugadores se familiaricen adecuadamente con el juego, culminando en la realización exitosa de la prueba escrita. Se ha comprobado empíricamente como la mejor usabilidad en cada fase del desarrollo.

Además, el proceso está rigurosamente documentado, ya que la historia completa del desarrollo se encuentra disponible en un repositorio de Github [59]. Esto posibilita que cada iteración del juego sea reproducible, incluso los experimentos con usuarios, lo que permite a otros investigadores volver a versiones anteriores del videojuego y realizar pruebas con diferentes grupos de usuarios en cada instancia.

6.2. Debilidades y mejoras para la metodología aplicada

Una debilidad identificada es que la prueba académica no logró detectar diferencias entre los grupos, asignando un puntaje perfecto a todos los estudiantes. Se proponen dos metodologías para mejorar la precisión de los resultados:

1) Aplicar A/B testing: comparar la metodología del videojuego educativo con otra aplicación o la lectura de un artículo relacionado con la misma materia que busca enseñar el videojuego. Luego, evaluar a los estudiantes con un examen escrito más extenso para establecer una graduación y realizar una comparación directa entre distintos métodos de educación.

2) Tomar una muestra aleatoria de estudiantes: Idealmente seleccionar personas que estén cursando Algoritmos y Estructuras de Datos y mostrarles el videojuego. Posteriormente, realizar un examen y medir las diferencias entre el grupo que probó el videojuego y el grupo que no lo hizo. Sin embargo, esto requiere apoyo institucional y garantizar la inclusión de la materia de grafos en el curso.

Un aspecto a mejorar es que el videojuego no tiene componentes sociales, tales como multijugador, ya sea de forma competitiva o cooperativa. Estos elementos se descartaron debido a la dificultad para aplicar pruebas de usuario multijugador y la complejidad de agregar cooperatividad o competitividad a nivel de diseño. El modelo MEEGA+ [47] busca evaluar también esta dimensión social. Para objeto de la evaluación de la calidad de juego, las preguntas relacionadas con este ítem se omitieron en el formulario.

Por otra parte, no se puede garantizar la heterogeneidad de la muestra. Existe el sesgo del voluntario, que afecta según [51]. El caso ideal implica una separación aleatoria en los cursos como parte de las actividades de la clase. No obstante, es probable que aún existan sesgos en los cursos debido a la selección de estudiantes de carreras asociadas a computación o informática. Se recomienda acotar el grupo solo a estudiantes afines a la informática. Por esta misma razón, también se aconseja acotar estos estudios únicamente a estudiantes de carreras relacionadas con ciencias, tecnología e ingeniería. Además de agregar segmentación en la mayor cantidad de aspectos posibles, como edad, carrera, estudios cursados, entre otros.

Además, según libros como [50], es más efectivo realizar una prueba dos semanas después para medir aprendizajes efectivos. No obstante, esto fue difícil de realizar dadas las condiciones del trabajo, ya que conseguir voluntarios en un contexto donde los estudiantes prefieren dedicar su tiempo a preparar exámenes en lugar de participar en una actividad universitaria fue un desafío. Se sugiere buscar apoyo institucional y aumentar los incentivos a la participación, por ejemplo, mediante actividades que fomenten la comunidad al inicio del semestre académico.

Finalmente, 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 en los videojuegos, que deben ser atractivos 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 está completamente desarrollada; no presenta una historia con un fin, desarrollo de personaje ni antagonista. Estos elementos suelen despertar más el interés del jugador [68, 55].

6.3. Fortalezas del software

Se programó un software funcional, interactivo y percibido positivamente por los usuarios. La arquitectura construida es portable debido a los patrones que sigue y se entrega documentación para poder avanzar en la misma metodología, aunque se quiera modificar la estructura de datos subyacente y los algoritmos a enseñar. Por ejemplo, este trabajo podría transferirse a la enseñanza de arreglos.

El diseño que consiste en obligar al usuario a pasar instrucción por instrucción pudo ser comprendido satisfactoriamente por todos los usuarios. Sin embargo, se determinaron puntos de mejora en el diseño del videojuego. Cabe destacar que estos puntos de mejora son a nivel de diseño e interacción, no de código.

6.4. Aspectos a mejorar del software

A partir del anexo G se pueden identificar puntos de mejora. Entre estos, la legibilidad, colocación de títulos, elección de fuentes y colores se menciona en un total de 7 comentarios. Por otra parte, se sugieren cambios en la progresión de los desafíos

6.5. Trabajo futuro

En el futuro, esta aplicación o sus derivaciones podrían evaluarse en otros contextos, con muestras más extensas, para garantizar una validez estadística más sólida. Se recomienda la implementación de un diseño experimental que mitigue sesgos como el factor de novedad y el sesgo del voluntario. Además, se sugiere medir los resultados del aprendizaje en distintos intervalos temporales al utilizar esta aplicación. Considerando que la aplicación fue concebida como un complemento a la enseñanza tradicional y para ser utilizada en pausas activas, se propone estudiarla en grupos donde se haya empleado como herramienta complementaria y comparar los resultados con otro grupo que no la haya utilizado.

Una mejora en la recolección de datos sería la aplicación de principios de telemetría y el uso de tecnologías adicionales que profundicen en el estudio de usabilidad y experiencia del usuario. Existen frameworks que sirven como servidor para aplicaciones relacionadas con videojuegos y que ofrecen servicios de telemetría y análisis de datos, como PlayFab [42]. Tecnologías como eye-tracking o biofeedback podrían proporcionar una comprensión más detallada y estandarizada de la experiencia de los usuarios con el videojuego, identificando áreas de mejora [67]. Por ejemplo, se espera que los usuarios visualicen el código cada vez que realizan un paso, y esto podría confirmarse mediante eye-tracking. En el trabajo de [67], se observa cómo los usuarios interactúan con las interfaces en un videojuego educativo.

Una posible expansión de este trabajo implica la exploración de otras estructuras de datos y algoritmos que no necesariamente estén vinculados a los grafos. El software desarrollado está diseñado para ser adaptable a otros algoritmos, aunque requeriría trabajo adicional para

determinar visualizaciones, representaciones y mecánicas específicas para cada uno.

Desde la perspectiva del diseño de juegos, se podrían agregar más elementos de gamificación, como la adquisición de nuevos elementos, desbloqueo de mecánicas, acumulación de puntos, establecimiento de un ranking global, sistema de logros y finales alternativos.

6.6. Conclusión

El videojuego ha logrado su objetivo de enseñar los algoritmos de BFS y DFS, como evidencian los resultados obtenidos por estudiantes de informática de la Universidad de Chile. Estos estudiantes demostraron comprender los algoritmos, así como la naturaleza de los grafos, al responder de manera acertada a una prueba asociada al juego. Por lo tanto se responde positivamente RQ1.

Por otra parte, el juego ha recibido una evaluación positiva y se considera, según el modelo MEEGA+ [47], como un juego de buena calidad. A pesar de las mejoras identificadas durante el proceso, el juego ha sido bien recibido por los estudiantes, y su percepción positiva puede influir beneficiosamente en la motivación (RQ2) y, por ende, en los resultados académicos.

Las hipótesis planteadas fueron confirmadas. La primera hipótesis (H1) indica, que un videojuego que educa sobre grafos puede enseñar a los estudiantes sobre los algoritmos BFS y DFS se ha comprobado como verdadera. Lo mismo ocurre con la segunda hipótesis (H2), un videojuego que enseñe grafos será percibido de manera positiva por los estudiantes que todavía no aprenden sobre esos contenidos, también es correcta, medida a través del valor θ entregado por el modelo MEEGA+.

Este trabajo sienta bases para futuras investigaciones, especialmente en relación con el diseño de videojuegos educativos y su impacto en la educación en informática. La metodología utilizada aquí puede ser perfeccionada en trabajos futuros para evaluar mejor la comprensión de los conceptos enseñados mediante este enfoque lúdico.

Para casos futuros, se recomienda emplear una metodología de estudio pre/post/post como se indica en [46], aumentar el tamaño muestral y utilizar al menos dos grupos: uno de control y otro de tratamiento para medir mejor las diferencias.

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] Alejandro Calderón and Mercedes Ruiz. A systematic literature review on serious games evaluation: An application to software project management. *Comput. Educ.*, 87:396–422, 2015.
- [4] 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.
- [5] Attorresi, Horacio Félix Lozzia, Gabriela Susana Abal, Facundo Juan Pablo Galibert, María Silvia Aguerri, María Ester. Teoría de Respuesta al Ítem. Conceptos básicos y aplicaciones para la medición de constructos psicológicos. *Revista Argentina de Clínica Psicológica* , 2009.
- [6] 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.
- [7] Paulo Eduardo Battistella and Christiane Gresse von Wangenheim. Games for Teaching Computing in Higher Education. A Systematic Review. 2016.
- [8] Chris Bigum and Jane Kenway. New information technologies and the ambiguous future of schooling—some possible scenarios. *International Handbook of Educational Change: Part One*, pages 375–395, 1998.
- [9] 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.
- [10] CADCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [11] R. Philip Chalmers. mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, 48(6):1–29, 2012.

- [12] R. Philip Chalmers. Generating adaptive and non-adaptive test interfaces for multidimensional item response theory applications. *Journal of Statistical Software*, 71(5):1–39, 2016.
- [13] Godot Community. Godot export variables, 2023. Accessed on 5 October 2023.
- [14] Wikipedia contributors. Golden sun (video game) wikipedia, the free encyclopedia, 2023. Accessed 04-October-2023.
- [15] Tomorrow Corporation. 7 billion humans - now available!, 2023. Accessed: 2023-11-15.
- [16] Sören Dallmeyer, Christoph Breuer, and Svenja Feiler. To pay or not to pay? the effects of monetary compensation on volunteers in sports clubs. *Nonprofit and Voluntary Sector Quarterly*, 2023.
- [17] DCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [18] Sarah E. Domoff, Ryan P. Foley, and Rick C. Ferkel. Addictive phone use and academic performance in adolescents. *Human Behavior and Emerging Technologies*, 2019.
- [19] 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.
- [20] 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.
- [21] Godot Foundation. Exporting Projects with Godot. Accessed on 2023-12-06.
- [22] Godot Foundation. Godot CSharp Differences with GDScript. Accessed on 2023-12-06.
- [23] Godot Foundation. Godot Game Engine. Accessed on 2023-12-06.
- [24] Godot Foundation. Godot Github Repository. Accessed on 2023-12-06.
- [25] Francisca Loreto Calderón Maldonado. Statistical methods for the analysis of polytomous response data in non-cognitive tests.
- [26] Adam Freeman. The mediator pattern. 2015.
- [27] 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.
- [28] Epic Games. Unreal Engine: The most powerful real-time 3D creation tool. Accessed on 2023-12-06.
- [29] 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.

- [30] 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.
- [31] 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.
- [32] Godot Community. Godot official documentation, 2023. [Online; accessed 4-October-2023].
- [33] 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.
- [34] 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.
- [35] Ioannis Karatzas. and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, Berlin, 2nd edition, 2000.
- [36] Kemono Games. Protocorgi, 2023. [Online; accessed 4-October-2023].
- [37] Kristian Kiili, Keith Devlin, Artti 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.
- [38] 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.
- [39] Francisca Loreto Calderón Maldonado. Statistical methods for the analysis of polytomous response data. 2021.
- [40] 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.
- [41] Ioana E. Marinescu, Nadav Klein, Andrew Chamberlain, and Morgan Smart. Incentives can reduce bias in online reviews. *Economics of Networks eJournal*, 2018.
- [42] Microsoft. Playfab playstream: Real-time event processing and monitoring for games. Accessed: 2023-11-18.
- [43] Microsoft. Visual studio code, 2023.
- [44] United Nations. The impact of digital technologies, 2023. Accessed: 2023-08-04.
- [45] 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.

- [46] Giani Petri and Christiane Gresse von Wangenheim. How games for computing education are evaluated? a systematic literature review. *Comput. Educ.*, 107:68–90, 2017.
- [47] 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.
- [48] 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.
- [49] Mark D. Reckase. Multidimensional item response theory. *Handbook of Statistics*, 26:607–642, 2009.
- [50] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [51] 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.
- [52] Grant Sanderson. 3blue1brown youtube channel. <https://www.youtube.com/3blue1brown>, 2023. Accessed on 4 October 2023.
- [53] Neil Selwyn. Looking forward : Reimagining digital technology and the contemporary university. 2014.
- [54] Kojiro Shojima. *Item Response Theory*, pages 85–154. Springer Nature Singapore, Singapore, 2022.
- [55] Ismar Frango Silveira. Building effective narratives for educational games. *2019 XIV Latin American Conference on Learning Technologies (LACLO)*, pages 299–305, 2019.
- [56] Rohan Surti, Amey Desai, Sohendar Rana, Manthan Sankhe, and Yogita Mane. Neoroute: A pathfinding algorithm visualizer. In *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, pages 1–5. IEEE, 2023.
- [57] Unity Technologies. Unity Game Engine. Accessed on 2023-12-06.
- [58] GQS UFSC. Meega+ a model for evaluating educational games. <http://www.gqs.ufsc.br/quality-evaluation/meega-plus/>. Accessed: 2023-Nov-2.
- [59] Alonso Utreras. IGACSE GitHub Repository. <https://github.com/AlasAltum/Thesis-IGACSE>. Accessed: 2023-Nov-2.
- [60] Willem J. van der Linden. Handbook of item response theory. 2015.
- [61] Cheng-Hsin Wang. Comprehensively summarizing what distracts students from online learning: A literature review. *Human Behavior and Emerging Technologies*, 2022.

- [62] David Weintrop and Uri Wilensky. Robobuilder: A program-to-play constructionist video game. In *Proceedings of the constructionism 2012 conference. Athens, Greece*, 2012.
- [63] Wikipedia contributors. Final fantasy iv — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [64] Wikipedia contributors. Pokémon emerald — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [65] Wikipedia contributors. Space invaders — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [66] 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.
- [67] Nurul Hidayah Mat Zain, Fariza Hanis Abdul Razak, Azizah Jaafar, and Mohd Firdaus Zulkipli. Eye tracking in educational games environment: Evaluating user interface design through eye tracking patterns. In *International Visual Informatics Conference*, 2011.
- [68] Natalia Padilla Zea, Francisco Luis Gutiérrez Vela, José Rafael López-Arcos, Ana Abad-Arranz, and Patricia Paderewski. Modeling storytelling to be used in educational video games. *Computers in Human Behavior*, 31:461–474, 2014.
- [69] Weinan Zhao and Valerie J Shute. Can playing a video game foster computational thinking skills? *Computers & Education*, 141:103633, 2019.
- [70] B. Zimmerman and Dale H. Schunk. Handbook of self-regulation of learning and performance. 2011.
- [71] 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 6.1: Formulario basado en MEEGA+. Se eliminó la dimensión social y no se incluyeron los ítems que preguntan por los objetivos particulares del juego.

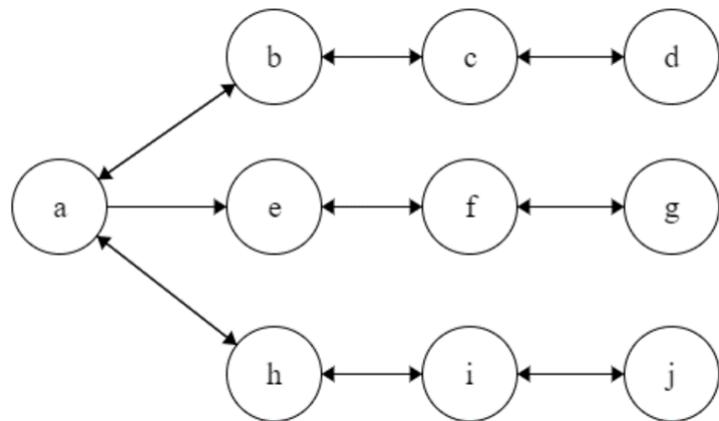
| Notación | Pregunta |
|----------|--|
| Q1 | El diseño de juego es atractivo |
| Q2 | La fuente de texto y los colores están bien combinados y son consistentes |
| Q3 | Tuve que aprender cosas antes de poder jugar |
| Q4 | Aprender a jugar se me hizo fácil |
| Q5* | Creo que la mayoría de la gente 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* | El juego permite customizar la apariencia (fuentes y color) según mis preferencias |
| Q11* | El juego previene que cometa errores |
| Q12* | Cuando cometo un error, es fácil recuperarse de él |
| Q13 | Cuando vi el juego por primera vez, tuve la sensación de que sería fácil para mí |
| Q14 | La estructura me ayudó a tener confianza en que aprendería con este juego |
| Q15 | Este juego es desafiante en la medida justa |
| Q16 | El juego ofrece nuevos desafíos a un ritmo adecuado |
| Q17 | El juego no se vuelve monótono a medida que se progresá |
| Q18 | Completar las tareas del juego me provocó satisfacción |
| Q19 | Gracias a mi esfuerzo personal pude avanzar en el juego |
| Q20 | Siento satisfacción con lo aprendido en este juego |
| Q21 | Recomendaría este juego a mis colegas |
| Q22* | Pude interactuar con otros jugadores mientras jugaba. |
| Q23* | El juego promueve la cooperación o competencia entre jugadores. |
| Q24* | Me sentí bien interactuando con otros jugadores durante el juego. |
| Q25 | Me entretuve con el juego |
| Q26 | Algún elemento del juego me hizo sonreír |
| Q27 | Había algo interesante al inicio del juego que llamó mi atención |
| Q28 | Estaba tan envuelto@ en la tarea propuesta por el juego que perdí la noción del tiempo |
| Q29 | Me olvidé de mi entorno físico mientras jugaba |
| Q30 | Los contenidos del juego son relevantes para mis intereses |
| Q31 | Es claro ver que los contenidos del juego se relacionan a cierta materia de la carrera |
| Q32 | Este juego es adecuado para enseñar el contenido |
| Q33 | Prefiero aprender con este juego en vez de aprender con otros métodos de enseñanza |
| Q34 | El juego contribuyó a mi aprendizaje |
| Q35 | 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 ni permitía customización.

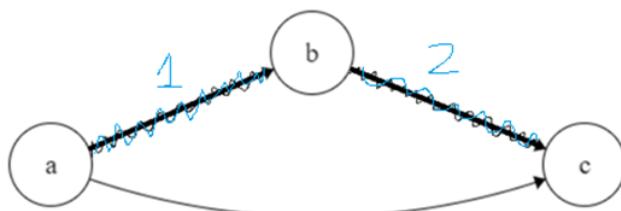
Anexo B: Prueba de conocimiento de grafos

Test para verificar aprendizaje de BFS y DFS

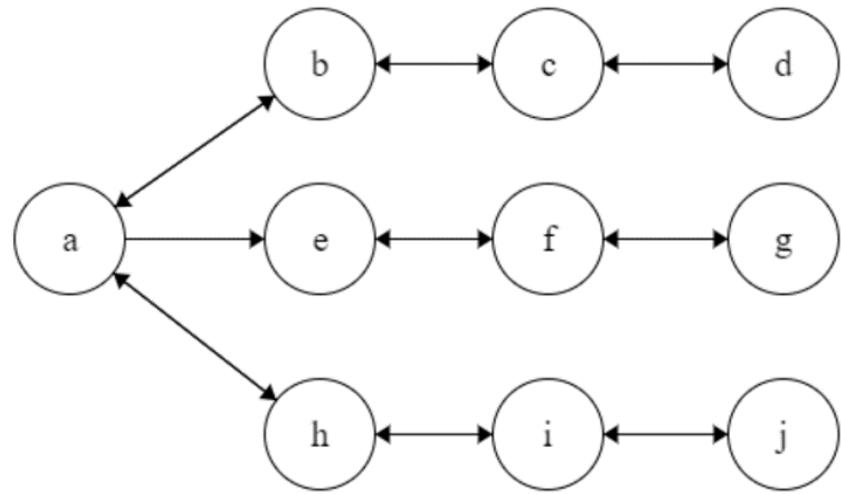
Consideré el siguiente grafo:



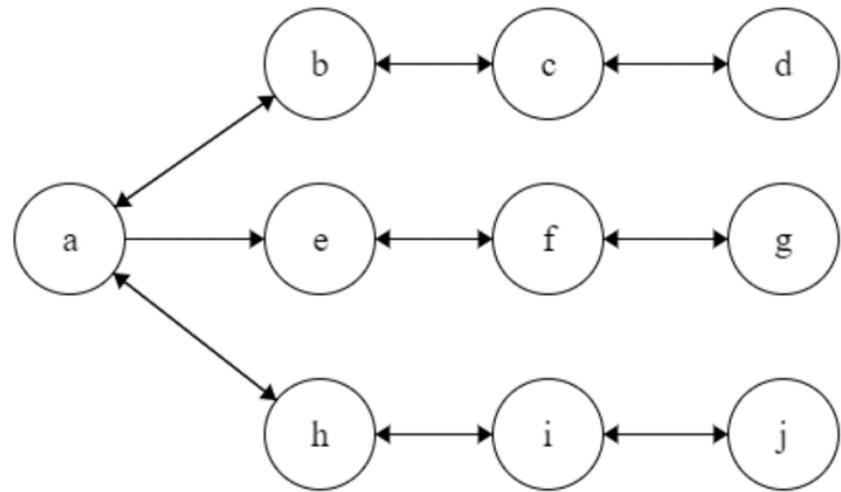
Muestre el recorrido de un grafo desde el nodo a utilizando distintos los dos algoritmos de búsqueda vistos. **Agregue una numeración al lado de sus arcos para indicar el orden en que se explorando.** Por ejemplo, en el siguiente grafo, si se recorre a, b y c partiendo desde a, pasando por b y llegando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

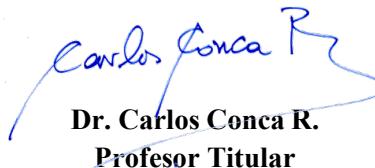
El proyecto busca crear y evaluar un videojuego educativo que enseñe una materia abstracta relacionada a la Ciencia de la Computación. Para su evaluación, se ofrecerá dicho videojuego la plataforma de U-Cursos cuando se imparta la materia respectiva.

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 D: 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

Anexo E: Script en R que permite obtener el valor indicador de la calidad de juego

Listing 6.1: Script en R para evaluar el videojuego

```
library(data.table)
library(mirt)
library(mirtCAT)
# CALCULATING THE SCORES
responses <- fread(input = "EXTRA.csv", header = T)
pars <- fread(input = "param.csv", header = T)
pars <- data.frame(
    a1=pars$a1, d1=pars$d1, d2=pars$d2, d3=pars$d3, d4=pars$d4)
modelo <- generate.mirt_object(
    parameters = pars, itemtype = "graded", min_category=1)
escore <- fscores(
    object = modelo, method = "EAP", response.pattern = used_responses)
SCORE_TRI <- escore[, 1]
# Theta is adjusted to improve interpretability.
final_theta <- SCORE_TRI * 15 + 50.0
mean(final_theta) # Theta is =
```

Anexo F: Comentarios abiertos entregados por los voluntarios

Las siguientes tablas de comentarios muestran las respuestas del primer grupo, aquel que realizó la prueba académica.

Comentarios del grupo que realizó la prueba académica frente a la pregunta: Nombra aspectos fuertes del juego

| Comentario |
|---|
| Es una forma distinta de aprender. Es interactivo. Me gusta que sea de prueba y error. |
| Enseña de manera correcta el algoritmo |
| Era intuitivo y fácil de entender |
| El aspecto que me llamó mucho la atención y lo destaco como un aspecto fuerte del juego, fue la forma interactiva de mostrar un código de programación y conceptos detrás de estos mismos. |
| De esa forma, se puede visualizar de mejor manera la dinámica detrás de un algoritmo (que muchas veces eso llega a ser un problema a la hora de analizar su funcionamiento, en especial si son algoritmos más complejos) |
| Me parece que al ser interactivo se entienden mejor los conceptos, además los movimientos de la nave ayudan a entender gráficamente cómo funciona el algoritmo |
| El diseño y la historia planteada son atractivos, como también la jugabilidad, es idónea para adquirir habilidades de programación. |
| Es intuitivo y no tan difícil de entender la idea, el diseño de niveles y la interfaz de usuario |
| Incorporar animales tiernos y buena música aporta mucho a la satisfacción y motivación de terminar los niveles |
| El acompañamiento visual de las líneas del código ayuda mucho al entendimiento del algoritmo sin tener la necesidad de explicar textualmente la función de cada uno |
| Me gustó poder saltarme las instrucciones porque no me gusta leerlas, menos si es un juego. Me gustaron los colores, llama la atención a pesar de ser un juego sin mucho detalle |
| La atmósfera del juego permite conectar fácilmente con la experiencia de aprendizaje, además su temática me parece bastante atractiva |
| Es entretenido y super fácil de seguir las instrucciones |
| Me gustó que agregue una imagen sobre el contenido de programación que se tratará en cada nivel, eso me facilita el aprendizaje y la idea de lo que estoy realizando. Además, el juego ayuda bastante si te equivocas o no realizaste una acción (como seleccionar el nodo u otros) |
| Es interesante que sea de programación y astronomía/ es desafiante |
| Es un juego entretenido, los gráficos llaman la atención, interactivo y uno puede quedarse pegado jugando |
| Las gráficas son muy bonitas y atractivas, invitan a jugar |

Comentarios a la pregunta “indica una o más sugerencias para mejorar el juego”. Grupo que realizó la prueba académica.

| Comentario |
|---|
| Mejorar colocación de títulos. |
| Hacerlo menos tedioso, quizás agregar alguna música acorde, poner el código de manera más amigable en vez de un código tan seco, que el grafo inicial sea más pequeño; |
| Quizás se podría explicar un poco más de qué significa el código de la derecha para gente que no tiene ni idea de programación. Que es un for, un while y demás, explicar cómo se recorre y por qué; |
| Agregar muchos más niveles que formen una gran escalera progresiva de contenidos a aprender. También podría sugerir que en el momento que la persona no interactúe con el juego en un tal momento del algoritmo, el juego diga mini hints que ayuden a pensar a la persona de lo que debería hacer en donde está estancado. Tal vez que sea más largo o que tenga más ejercicios por algoritmo; |
| La tipografía de la letra y los colores para facilitar la lectura, hay números que se confunden con letras y requieren contraste con el resto del juego, un buen ejemplo de esto sería el final fantasy donde el texto es solamente un fondo negro con letras blancas los controles, al momento de agregar los planetas yo pensaba que era apretar r no más pero había que poner el mouse sobre el planeta cambiar la letra no me gustaba; |
| Las palabras del texto explicativo deberían resaltar las instrucciones centrales. |
| Sería positivo incorporar un botón de ayuda, en algunas partes sentí que me quedé atrapado.; |
| Las instrucciones del principio deberían ser entregadas de una manera más simple y directa. Quizás un cuadro con una lista de todo lo que hay que hacer al inicio podría ayudar.; |
| Mejorar la fuente de texto, se me complicó en ciertos casos leer algunas letras.; |
| Podría ser la música que era un poco desesperante y se podría hacer un pequeño tutorial explicando las funciones que se utilizan; |
| Si el juego busca enseñar programación, sería bueno colocar un módulo donde se pueda explicar mejor el código o dar alguna explicación de ciertas funciones que no sabía qué hacían pero igual tenía que apretar espacio para pasárlas (podría ser una pequeña explicación al pasar el mouse por sobre el texto). Quizás no era de importancia para el juego, pero como no conozco mucho el lenguaje con el que se utilizó e igual sé programar, sería interesante poder conocer qué significan tales funciones.; |
| Encuentro que el juego se puede completar sin entender del todo la diferencia entre queues y stacks, solo mecanizando el seguir las instrucciones del panel derecho. Ambos niveles se diferencian en que el primero es mucho más rápido de recorrer y mecanizar, mientras que el segundo plantea más dificultad, pero no queda completamente claro en cuál se usan queues y en cuál stacks.; |
| Quizás faltaron planetas para que se notara más el hecho de acumular elementos en queues o stacks y la forma en que estos se extraen, o quizás sería más sencillo entender estos conceptos si el juego solo consistiera en clickear planetas para ir recorriendo los mapas, sin necesidad de apretar teclas como espacio, R, W o S, ya que a la larga esta mecánica distrae un poco de entender realmente cómo están operando los algoritmos. |

Comentarios a la pregunta “Algún otro comentario?”. Grupo que realizó la prueba académica.

| Comentario |
|--|
| Muy buen juego, es a mi gusta una forma divertida de aprender a manejar grafos. |
| Muy entretenido, viva los pandas rojos!! |
| Muy interesante y entretenida la propuesta de este juego. Espero que se logré desarrollar más y más :3 |
| No soy fan del tema espacial pero si me gusto el juego |
| muy bueno |
| Las palabras del texto explicativo deberían resaltar las instrucciones centrales. |
| Hay unos pasos en los que no hay que hacer ninguna acción, por ejemplo el comando s.pop, y al principio eso me confundió un poco por no tener la certeza de cuál era su función dentro el algoritmo. Creo que eso es parte del desafío del juego y se logra comprender gracias al acompañamiento visual y sonoro |
| creo que a pesar de ser estudiante de ingeniería tengo poca noción de la programación, así que para mi al no leer las instrucciones necesite ayuda para terminar el juego. Ayudaría quizás en el lado derecho no poner un código y si una explicación con palabras |
| En líneas generales una excelente temática y experiencia. |
| Buen juego, entretenido y didactico para aprender la materia de programación. Me gustaria que hubiesen más juegos así para abordar otros contenidos de programación. |
| Muy bueno el juego!!! Felicitaciones |
| aguanten los pandas! |

Comentarios del grupo que participó en el estudio libre sin restricciones

Comentarios a la pregunta: “Por favor, indica uno o más aspectos fuertes del juego”. Grupo libre.

| Comentario |
|--|
| 1. El ir paso a paso ejecutando los algoritmos ayuda mucho a entender cómo funcionan y cómo se diferencian entre sí. 2. El feedback visual de algunas cosas como la variable seleccionada o el estado actual del stack/queue que se está usando permite siempre saber lo que está ocurriendo, o incluso retomar la ejecución luego de perder la atención un momento. |
| La musica es bien llamativa y la tematica de la nave buscando los pandas rojos es atractiva. |
| La idea de representar lo importante de programar gráficamente es muy interesante, la idea de abstraerlo a una temática de exploración espacial también lo es. |
| Me gusta que te obligue a hacer las instrucciones una a una. |
| El aspecto visual es bueno, la forma retro del código mostrado al lado derecho me gusta. |

Comentarios a la pregunta “indica una o más sugerencias para mejorar el juego”. Grupo libre.

Comentario

1. El juego se beneficiaría de más feedback visual sobre las cosas seleccionadas/actuales en el algoritmo, como cuales son los nodos vecinos. 2. Además, un nivel inicial con nodos ordenados de forma que las líneas no se sobrepongan puede ser útil. 3. Para notar la diferencia entre BFS y DFS, tener una misma configuración de nodos puede ayudar a entender cómo ocurre que ciertos planetas se visitan antes que otros según el algoritmo que se esté usando. 4. Finalmente, animaciones in-game para explicar los algoritmos pueden funcionar mucho mejor que gifs a pantalla completa.

Algunos de los gifs de fondo que muestran el recorrido de grafos podrían tener mejor calidad, y creo que algunos dialogos se muestran muy lento en comparación a otros. Creo que hace falta niveles más profundos en los algoritmos de recorrido de grafos, para entender una dinámica mayor y quizás algún nivel que te dejen solo y por los requisitos de algún tipo, tengas que utilizar alguno de los algoritmos enseñados

Considero que una mejora en la selección de colores y distribuciones espaciales de los elementos en pantalla haría que gráficamente el juego mejorara mucho, y pienso que eso a su vez ayudaría a que fuera mucho más interesante aprender mediante él. Por otro lado, creo que se le debería dar más libertad al jugador, en especial la posibilidad de que se pudiera equivocar y este error afecte en el funcionamiento del programa (explicando gráficamente de alguna manera por qué se equivocó y qué consecuencias tiene eso), entiendo que es algo complicado de implementar pero creo que si es implementado de una manera inteligente puede potenciar mucho el aprendizaje.

Algunas cosas. Hay un bug que cuando aparece una de las imágenes de pandas rojo, el espacio deja de funcionar para avanzar.

Creo que el no explicar con anterioridad las instrucciones es un poco complicado. Me vi haciendo cosas que no entendía y simplemente siguiendo las instrucciones sin ver el algoritmo a gran escala, además estaba camuflado con todos los aspectos del juego. Creo que sería bueno poner un paralelo un poco más formal a medida que se desarrolla el juego, quizás más animaciones para mostrar que el planeta que seleccionas efectivamente se va a la cola, y quizás además del número debería haber una miniatura del planeta en la lista.

Otra cosa que me pasó un par de veces es que sin querer apreté la barra espaciadora y por coincidencia estaba en la opción correcta, por lo que no entendí qué hice y el juego avanzó. Quizás en cada iteración la consola debería reiniciar la posición para tener que mover el selector de manera voluntaria y entender el algoritmo.

Otro punto es que cuando haces algo bien, hay un sonido que uno termina relacionando con haber hecho bien el paso en el algoritmo. En algún momento el sonido deja de sonar al inicio de los for, lo que es raro porque no sabes si lo hiciste bien o no.

El aspecto de la cola, ver la forma de avanzar más rápido en esos pasos estaría bueno.

Comentarios a la pregunta “Algún otro comentario?”. Grupo libre.

| Comentario |
|---|
| Es un juego completamente necesario y una buena idea que puede facilitar mucho el aprendizaje de grafos. Gran trabajo PD: Tal vez tener alguna forma de configurar niveles custom podría hacer de esta una herramienta de debugging, que puede estar muy bien :D |
| Me parecio bien entretenido en general |
| Considero admirable el aportar en el aprendizaje de la computación mediante juegos, y muy valiente considerando que no es una opción ”popularçon mucha información al respecto |
| Me gusta la idea pero creo que hay que pulir los controles y las instrucciones |
| Me gustó el concepto. Se entiende bien el concepto de búsqueda en grafo y queda “grabado” al intentarlo varias veces. Eso sí, como mi objetivo era avanzar en el juego lo hice pese a que los ejercicios con la cola a veces eran más complejos en el sentido de avanzar al estar atento a pulsar el botón .espcio”, pero en general me gustó el juego y cómo se aprende en el proceso. |

Anexo G: Comentarios de algunos expertos

Estos comentarios han sido anonimizados y aleatoriazados para proteger la identidad de quienes los emitieron, pues se les garantizó anonimato en caso de que sus comentarios se publicaran. Muchos comentarios requieren más contexto para ser entendidos, puesto que se enmarcaban en medio de una prueba de usuario, frente a cierta interfaz. Por esta razón, se hizo una selección de comentarios y se muestran aquí.

Comentario

| |
|---|
| El juego debería ser consistente, si quieres avanzar con SPACE, apreta SPACE para continuar. Siempre. (Después de presionar un planeta, hace click en uno y no pasa nada) ¿Qué pasa al hacer click en un planeta? mmm, nada. Falta feedback ahí. |
| En un juego siempre hay que tener clara la condición de ganar. Cuál es el objetivo para ganar? La misión es siempre la misma? Anda subdividiendo la misión. |
| Te recomiendo hacer tutoriales de a poco, como Duolingo. AL principio te muestran las mecánicas con ejercicios muy simples, y luego te dejan andar. |
| Haz tutoriales, es como la técnica del triciclo. Al principio los ayudas, después los dejas andar solos |
| No recomiendo destacar las cosas solo con color. Agrega movimiento también. Para la gente con daltonismo puede ser complejo. |
| Trata de que el mouse cambie cuando pasas por un elemento clickeable o seleccionable. |
| (Respecto a la popup con un if) La fuente del Yes y No está re mala. Parece muy poco real. |
| Usa una única fuente constante de información. Si me quedo con una, que sea solo una popup. |
| Aprovecha el movimiento. Las cosas que se mueven llaman mucho más la atención. Si quieras que hagan click en algo, agrégale movimiento. Si quieras que el usuario lea algo, agrégale movimiento |
| Prueba testear conceptos de a poquito: Quiero testear este tutorial. Qué cosas ven primero? Después, prueba cambiar los colores de los planetas, resaltarlos, vé cómo eso afecta. Pero es importante probar paso por paso, si haces todos los cambios de una sin probar y falla algo, perderás mucho tiempo y no sabrás exactamente qué falló. |
| Busca conocer bien a tu público objetivo. Si son gente que está dado Algoritmos y Estructuras de Datos, ve qué edad tienen, qué tipo de animaciones llaman más su atención (...). |
| El núcleo del juego se debe tratar de que estás siguiendo las instrucciones (...). Lo ideal es no explicitarlo y que se dé a entender por sí solo. |
| Confía en el conocimiento de tu usuario. Gente universitaria que juega juegos. Selecciona este nodo y agrégalo a la variable. Que el usuario descubra a través de la interfaz cómo se hace eso Si la interfaz está bien hecha, el usuario debería encontrar cómo hacerlo. |
| Hay un concepto que se llama la ceguera del cambio. Uno no detecta los cambios en los que no está enfocado Si estoy enfocado en cierta parte de la pantalla, hay una parte que está cambiando y no la voy a ver si me pierdo esta información. Enfocar la atención del usuario en un solo lado, que es donde estoy dando la instrucción, procura que las instrucciones que se den con una sola forma. |
| Podrías simplificar mucho más el primer acercamiento. Tutoriales interactivos de lenguajes de programación: Esto es un if, lo que está dentro, así conectas lo que hiciste en el primer, paso con un conocimiento nuevo en el segundo paso |
| Algo importante en UX/UI es nunca sorprender al usuario. Mientras menos tenga que aprender, mejor. Lo ideal es usar estándares y colgarde de ellos. Piensa en la Ley de Jacob: Los usuarios gastan la mayor parte de su tiempo en otros sitios. Tu juego representa un porcentaje enano en la vida de los demás usuario. |

Anexo H: Tutoriales

Primer Tutorial

En el primer tutorial, se pretende familiarizar al jugador con la pantalla, demostrando que los planetas son navegables 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 6.1: 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 6.2).

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



Figura 6.2: 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.

instrucción. Si responde incorrectamente, perderá y deberá reiniciar el nivel. En la figura 6.3, se muestra la ventana que aparece al jugador al llegar a una instrucción con un if.



Figura 6.3: 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.

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.

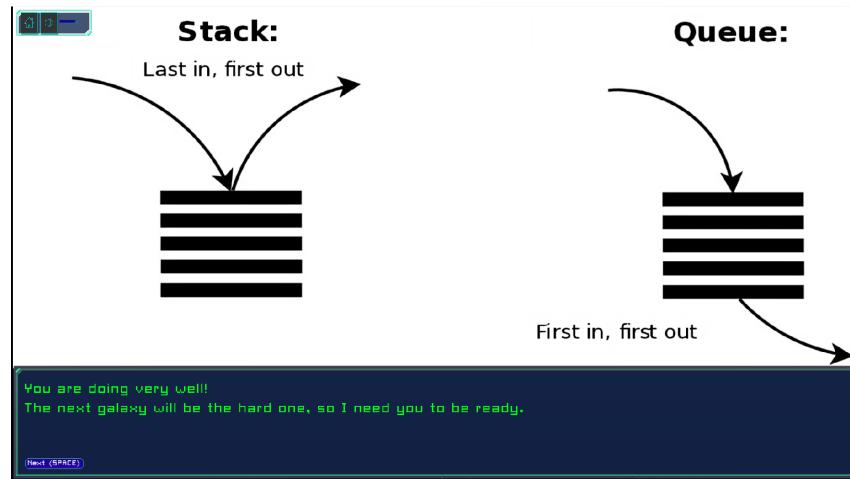


Figura 6.4: Tercer tutorial. Se le explica al usuario al inicio qué son las colas y stacks y cómo funcionan y en qué difieren.



Figura 6.5: Tercer tutorial. Mostrando al usuario cómo funciona un stack a través de animaciones.



Figura 6.6: Tercer tutorial. El jugador debe presionar R sobre el planeta 2 para avanzar a la siguiente instrucción.