



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

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

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

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ALONSO UTRERAS MIRANDA

PROFESOR GUÍA:
IVÁN SIPIRÁN MENDOZA

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

SANTIAGO DE CHILE
2023

Resumen

Los videojuegos educativos son reconocidos como herramientas motivadoras para enseñar y entretenir a los estudiantes. En este proyecto, se presenta IGASCE, un videojuego educativo diseñado para enseñar conceptos de grafos y los algoritmos BFS y DFS. Se utilizó la metodología MEEGA+ [?] para evaluar la calidad de la aplicación, solicitando a dos grupos que probaran el juego y respondieran cuestionarios. El primero realizó una prueba académica después del cuestionario. Este grupo, sin conocimiento previo de grafos, obtuvo una puntuación perfecta ($n=15$) después de haber probado el videojuego y calificó al juego con $\theta = 0,613$, que corresponde según MEEGA+ a un juego bueno, pero no excelente [?]. El segundo grupo, aquel que no realizó la prueba académica, dio una puntuación de $\theta = 0,616$. Se concluye que el juego facilitó el aprendizaje de los estudiantes sobre grafos y los algoritmos BFS y DFS. La percepción de los usuarios señaló posibles áreas de mejora en usabilidad, interfaz y gamificación. La arquitectura de software desarrollada permite la reutilización para otros algoritmos y estructuras de datos. Además, se proporcionan pautas para obtener resultados más precisos en futuras investigaciones.

Abstract

Educational video games are acknowledged as motivating tools for teaching and entertaining students. In this project, IGASCE is introduced, an educational video game designed to impart concepts of graphs and BFS and DFS algorithms. The MEEGA+ methodology [?] was employed to assess the quality of the application, involving two groups to play the game and respond to questionnaires. The first group underwent an academic test following the questionnaire. This group, unfamiliar with graph concepts, achieved a perfect score ($n=15$) after testing the video game and rated it with $\theta = 0,613$, which, according to MEEGA+ [?], corresponds to a good but not excellent game. The second group, which did not take the academic test, gave a score of $\theta = 0,616$. It is concluded that the game facilitated student learning about graphs and BFS and DFS algorithms. User perception indicated potential areas for improvement in usability, interface, and gamification. The software architecture developed allows for reuse with other algorithms and data structures. At the end, guidelines are provided for obtaining more precise results in future research.

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

Capítulo 1

Introducción

Esta investigación tiene como objetivo transformar las instrucciones de algoritmos de grafos en una representación visual e interactiva a través de un videojuego. Esta representación se destina a estudiantes de informática, con lo que se puede evaluar si el uso de herramientas interactivas con feedback visual mejora tanto el aprendizaje como la motivación.

El trabajo implica presentar a estudiantes de ciencias de la computación (CS) un videojuego que exhiba grafos. En este juego, el usuario debe ejecutar las instrucciones de los algoritmos con la asistencia de elementos visuales y auditivos.

El objetivo de esta investigación de tesis es identificar posibles diferencias en los niveles de motivación y comprensión de los algoritmos entre los estudiantes. Esto se medirá mediante una prueba que evaluará su conocimiento de los algoritmos de grafos.

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.

1.1. Motivación

La adopción de tecnologías digitales ha agilizado el acceso al conocimiento, permitiendo que estudiantes previamente excluidos ahora puedan acceder a la educación. Sin embargo, la educación en línea presenta desafíos propios [?]. En este contexto, es crucial buscar metodologías que aceleren el aprendizaje con tecnologías adaptables, escalables y eficientes.

Se postula popularmente que la capacidad de atención de forma prolongada ha disminuido en las nuevas generaciones. Hay estudios que contradicen estas afirmaciones [?], indicando que las habilidades cognitivas de los estudiantes han cambiado, pero no necesariamente empeorado. Existen términos como “doomsters” y “boosters” [?], para describir la polarización entre estas miradas con respecto a la tecnología.

En lo que sí existe consenso, es que la tecnología y su portabilidad ha incrementado la

cantidad de distracciones a las que se exponen los estudiantes [?, ?]. Se le llama multitasking al acto de intentar realizar más de una tarea a la vez. Se ha demostrado que esta práctica disminuye la productividad y el aprendizaje, aunque las personas creen que pueden hacer más de una cosa a la vez [?]. Con la popularización de los smartphones y las clases en línea, el multitasking -desde hacer labores del hogar hasta sacar el celular y revisar alguna aplicación-durante clases se ha incrementado [?].

Una forma de distracción reconocida en la literatura es la interferencia motivacional, propuesta y explicada por Fries y Dietz [?]. Esta señala que la motivación respecto a la clase disminuye debido a la presencia de tentaciones más atractivas para la atención, como los smartphones. En este contexto, donde los estudiantes se ven tentados a distraerse, es importante buscar formas de mantenerlos motivados y enfocados durante las clases. Los videojuegos educativos destacan porque tienen potencial como una herramienta complementaria para la enseñanza, evaluación y entretenimiento para los estudiantes. Entre sus beneficios se destaca la motivación. En [?] se indica que para disfrutar una actividad, primero se debe permitir a la mente de un individuo percibir tal actividad como motivante.

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

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

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

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

1.2. Objetivo General

El objetivo principal de este trabajo es medir el aprendizaje y la motivación de los estudiantes de computación al utilizar un videojuego educativo para instruir en algoritmos

vinculados a grafos, incorporando las características expuestas en este documento.

1.3. Objetivos Específicos

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

1.4. Marco teórico

1.4.1. Metodología de trabajo para validar videojuegos educativos

Petri y Gresse von Wangenheim en [?], antes de crear el modelo MEEGA+ cuya publicación sucedió el 2018 [?], hacen una revisión de literatura de cómo son evaluados los videojuegos educativos relacionados con la computación a partir de una muestra de 3617 artículos. Clasifica los estudios como verdaderos estudios, quasi-experimentales, no experimentales y ad-hoc según la metodología empleada.

Aquellos estudios que tenían múltiples grupos con asignación aleatoria eran considerados experimentales. Si el estudio usa múltiples grupos o múltiples momentos de medida sin asignación aleatoria, es considerado cuasi-experimental. Si no se emplean múltiples grupos, pero se hace de una forma sistemática con estudios de casos, es no experimental. Estudios que no son llevados de forma sistemática, sin una definición explícita del estudio y la forma de medirlo de antemano, son clasificados como estudios ad-hoc.

En una reseña, Calderón y Ruiz [?], los autores identifican que la mayoría de los videojuegos educativos son evaluados en términos de aprendizaje, usabilidad y experiencia de usuario. Además, indican que la mayoría de los estudios se realizaron de una forma ad-hoc, sin sistematización. Este trabajo asevera que la mayoría de los estudios realizados utilizan cuestionarios y entrevistas como métodos de validación de resultados. Por otra parte, se crea una categorización de características que indican calidad en un juego, que posteriormente son utilizadas por [?] en su cuestionario. Ejemplos de items evaluados son: Diseño de juego, satisfacción del usuario, usabilidad, motivación, resultados del aprendizaje, entre otros.

En su reseña sobre videojuegos serios, Calderón y Ruiz [?] también tabulan los tipos de procedimientos utilizados para validar estos trabajos, identificando tres tipos: 1) simple; 2) pre/post y 3) pre/post/post. 1) En el primero, los autores llevan a cabo una sesión con un

juego serio, después de jugarlo, se aplican los mecanismos de evaluación a los jugadores; 2) En el pre/post, la prueba tiene dos etapas de evaluación. Uno antes del juego serio, y otro después, de manera que se establece el conocimiento anterior a la prueba; 3) Para el caso pre/post/post se hace, además de las pruebas pre y post, una prueba posterior semanas después para analizar los niveles de retención. Respecto a las frecuencias, 50 de 89 de estos estudios se hicieron con una metodología simple y el 55% lo hizo con tamaños muestrales inferiores a 40 [?].

Apoyándose en esta conclusión, Petri y Christiane Gresse von Wangenheim, en su reseña de literatura sobre videojuegos educativos [?] aseveran que la mayoría de los estudios sobre videojuegos educativos son realizados de una manera ad-hoc en términos de diseño de investigación, medición, recolección de datos y análisis. Sin embargo, destacan el modelo MEEGA [?], indicando que ha sido utilizado además en otras áreas distintas de la computación, proporcionando mayor sistematización y logrando que los trabajos basados en este modelo tengan mayor validez científica.

1.4.2. Teoría de respuesta al ítem, Item Response Theory o IRT, intuición cualitativa

Es un modelo estadístico utilizado utilizado por muchos tests globales como pruebas de inglés como lengua extranjera y el programa para la evaluación de estudiantes internacional. La cualidad que se le destaca es que permite evaluar propiedades estadísticas de forma detallada sobre cada ítem (o pregunta según sea el caso), viendo la dificultad y la capacidad de diferenciación de cada pregunta [?, ?].

Los modelos basados en IRT se basan en tres asunciones: Primero, existe una relación funcional entre la rasgo latente que se busca medir y la probabilidad de responder un ítem (o pregunta) en una cierta categoría, por ejemplo, en desacuerdo, de acuerdo o ninguno [?].

Un segundo supuesto es que existe una escala continua y unidimensional de habilidad denotada como θ (parámetro theta). Esta es una suposición fuerte, pues los datos nunca son completamente unidimensionales, sin embargo, existen técnicas para verificar la unidimensionalidad de los datos de testeo [?]. Si se quieren medir más características al mismo tiempo, existen otros modelos de IRT multidimensionales [?].

El tercer supuesto es la independencia local, la persona responde de forma independiente a cada uno de los ítems o preguntas de la prueba, sin influenciar la respuesta de una pregunta la respuesta de otra [?].

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

Existen distintos modelos logísticos para evaluar θ . El que se usa en este trabajo es el modelo logístico de 2 parámetros (2PLM o Two-Parameter Logistic Model) 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 [?]. 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* [?] y *mirtCAT* [?] 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 [?]. 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 [?].

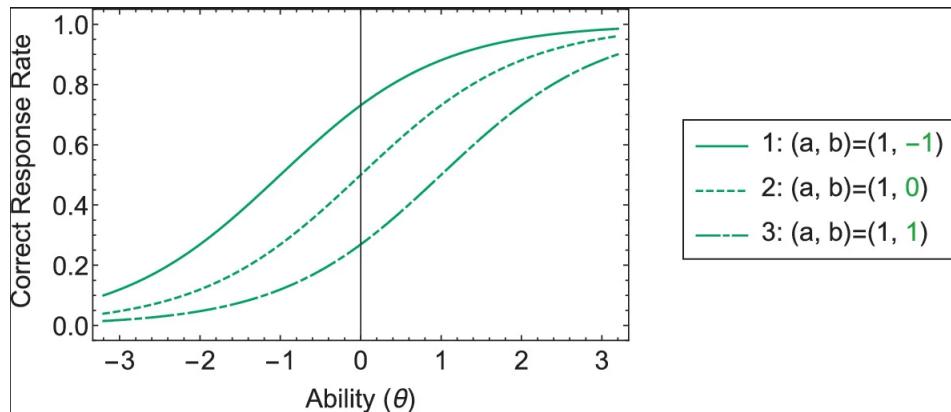


Figura 1.1: Probabilidad de responder correctamente para distintos valores del parámetro dificultad b

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 [?].

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 [?, ?]. De esta manera, a medida que θ , la calidad de juego es mejor según [?], las respuestas más tenderán a ser del formato Muy de acuerdo, a excepción de algunas preguntas que no se incluyen en la valoración. El script de R utilizado para el cálculo de θ se encuentra disponible en [?].

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.

1.4.3. Motores de videojuegos: Godot

Existen diversos motores de videojuegos, como Unreal Engine [?], Unity [?] y Godot [?]. Este último resalta por ser completamente gratuito, de código abierto y poseer una comunidad que también participa activamente en el desarrollo del motor [?]. La principal ventaja que ofrecen los motores de videojuegos es ofrecer un programa monolítico capaz de integrar diversas funcionalidades para distintos tipos de profesionales. Por ejemplo, ofrecen interfaces para integrar animaciones, así como trabajo con efectos visuales, de audio. Además, vienen con librerías integradas que se suelen utilizar en videojuegos, como colisiones, texturizado o composición de elementos, permitiendo unir la representación visual, auditiva y programática de un personaje. Los motores de videojuegos permiten exportar la aplicación a distintas plataformas sin necesidad de modificar el código [?].

Godot utiliza principalmente dos lenguajes de programación, GDScript, similar a Python, y C. Para este trabajo se utilizaron ambos. El primero permite prototipar rápidamente debido a su simpleza e integración con el motor. El último es más rápido, pero hay características del motor que no están integradas en la versión utilizada durante este trabajo, que es la 3.5.2 [?].

Capítulo 2

Trabajo Relacionado

2.0.1. Videojuegos educativos o serios

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

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

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

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

Kiili y su equipo [?] analizaron el uso de videojuegos en enseñanza y evaluación en matemáticas a través de los títulos “Semideus” y ”Wuzzit Trouble”. A través de estos, llegaron a la conclusión de que es posible utilizar videojuegos para enseñar y evaluar al mismo tiempo.

Además, caracterizaron estadísticamente las diferencias producidas por el uso de videojuegos entre resultados de un pre test y un post test.

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

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

Petri y otros [?] coinciden en la falta de sistematización en la evaluación de videojuegos educativos. De hecho, existen juegos serios que ni siquiera se autodenominan o se consideran como tales [?]. Además, muchos estudios eran cualitativos y carecían de resultados cuantitativos, por lo que no eran comparables. En vista de este antecedente, Petri et al. [?] desarrollaron el modelo MEEGA+ (Modelo para la Evaluación de Juegos Educativos y Escala EGameFlow).

Entre las deficiencias señaladas al momento de crear videojuegos según Petri et al. [?] se encuentran la carencia de: 1) Definición de un objetivo de evaluación; 2) Diseño de investigación; 3) Plan de medición; 4) Instrumentos de recopilación de datos; y 5) Métodos de análisis de datos. Una práctica que era común al analizar estas herramientas es hacer uso de comentarios informales por parte del estudiantado, sin la aplicación de escalas psicométricas [?].

Capítulo 3

Diseño de la Solución

La aplicación creada, llamada IGACSE (Interactive Graph Algorithms for Computer Science Education) de ahora en adelante, es propuesta y diseñada en base a algunos requerimientos y necesidades de los estudiantes de computación que se nombrarán en las siguientes secciones.

3.1. Referentes

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

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

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

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

3.2. Objetivo del diseño deseado

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

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

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

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

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

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

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

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

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

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

El juego se centra en explorar planetas a través de rutas, utilizando una metáfora de

grafos, donde los planetas representan nodos y los caminos entre ellos son aristas. El objetivo es rescatar pandas rojos dispersos en los planetas. Para lograrlo, es necesario explorar todos los nodos en una galaxia, correspondiente a un nivel. En cada nivel, se enseña un algoritmo distinto.

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

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

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

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

3.3.1. Diagrama de flujo de juego

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

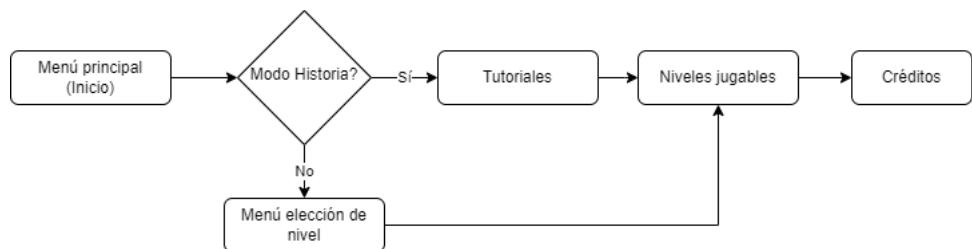


Figura 3.1: Flujo de juego de IGACSE

3.3.2. Narrativa

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

planetas de la galaxia. En un inicio, el piloto señala que los niveles de combustible son bajos, y la estación sugiere continuar con las siguientes galaxias, evitando el uso del piloto automático debido al alto costo y consumo de combustible de las redes neuronales. Para optimizar recursos, la nave deberá seguir instrucciones manuales para completar la misión.

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

3.3.3. Menú principal

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

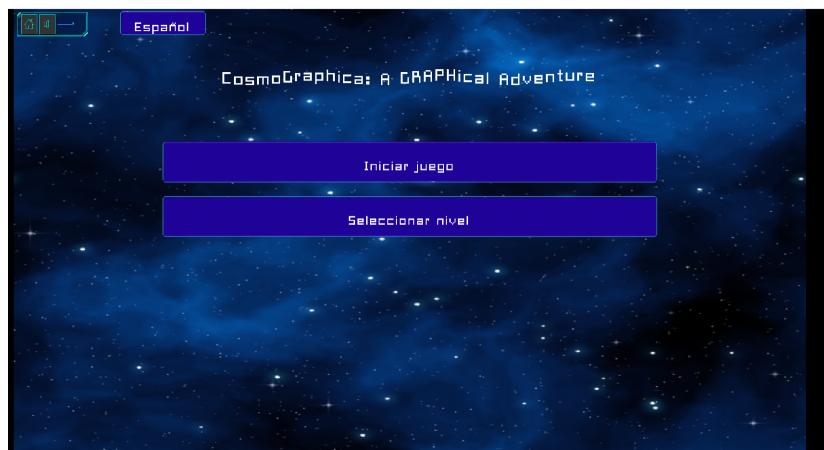


Figura 3.2: Menú principal de IGACSE, que se muestra al iniciar el juego.

3.3.4. Tutoriales

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

Primer Tutorial

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



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

Segundo tutorial

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

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



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

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



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

Tercer tutorial

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

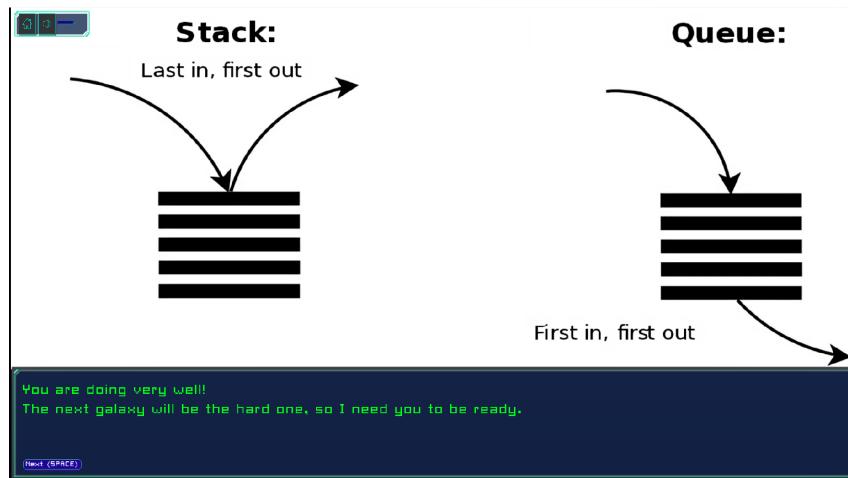


Figura 3.6: Tercer tutorial. Se le explica al usuario al inicio qué son las colas y stacks y cómo funcionan y en qué difieren.

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



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



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

3.3.5. Niveles jugables

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

Se crearon 4 niveles para el juego, incluyendo los algoritmos de Kruskal y Prim. Sin embargo, estos últimos no fueron incluidos en la versión final del juego, pues el diseño y la experiencia de usuario eran distintos por requerir otro tipo de acciones, como ordenar arcos y operar con conjuntos, obteniendo uniones e intersecciones. Además, el tiempo de la prueba de usuario era aproximadamente de 45 minutos, por lo que se prefirió no incluirlo en la prueba dado que no se contaba con el tiempo suficiente para probarlos.

3.4. Proceso de diseño

Para diseñar el videojuego, primero se determinaron problemas que se querían solucionar. En particular, el autor hipotetiza que los estudiantes de computación tienen dificultades para comprender completamente los algoritmos que se enseñan en los cursos de programación, y que no siempre son conscientes de esta falta de comprensión. Por lo tanto, se busca que este videojuego sea una herramienta que permita a los estudiantes entender los algoritmos paso a paso, guiándolos sutilmente a lo largo del proceso de aprendizaje.

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

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

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

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

Los diseñadores, en entrevistas separadas coincidieron en la necesidad de introducir tutoriales que mostraran los elementos del juego uno a uno, por lo que se decidió agregar tutoriales. Además, se decidió agregar una historia de fondo para lograr que el jugador se sienta identificado con el juego y comprenda un uso inmediato de estos algoritmos y estructuras. Por otra parte, los diseñadores enfatizaron la necesidad de agregar una componente de premiación y gamificación para despertar la motivación de los usuarios.

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

servidor de venta de videojuegos y elementos relacionados de creadores con bajos recursos o del mundo indie. El estilo se compró en itch.io a través de la tienda de assets. Los sonidos también se descargaron desde la misma tienda.

3.4.1. Diseño de la interfaz de usuario

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

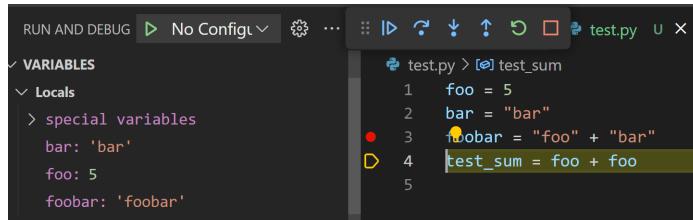


Figura 3.9: Visual Studio Code Debugger.

Hay otros juegos cuya interfaz sirvió de inspiración para la realización del juego final, como 7 Billion Humans[?], donde el código se ve a la derecha de la pantalla. La porción que contiene los elementos y eventos del juego está al lado izquierdo. Lo que ocurre en la interfaz de la derecha afecta al mundo virtual, tal como en la aplicación IGASCE.



Figura 3.10: Videojuego 7 Billion Humans

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

3.4.2. Diseño de efectos de sonido y música

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



Figura 3.11: Videojuego CodeCombat

Fantasy IV [?].

En juegos como Pokemon [?] y Golden Sun [?], cada vez que se hace click en algún botón y se gatilla un evento, se emite un sonido de confirmación, como cuando el jugador selecciona un ataque. El objetivo es confirmarle al usuario que se está ejecutando correctamente una acción. IGASCE sigue una estrategia similar, utilizando sonidos agudos de confirmación al hacer clic en planetas, caminos o al presionar teclas correctamente (espacio o R). Estos sonidos nítidos cumplen la función de validar la ejecución exitosa de una acción sin distraer al jugador.

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

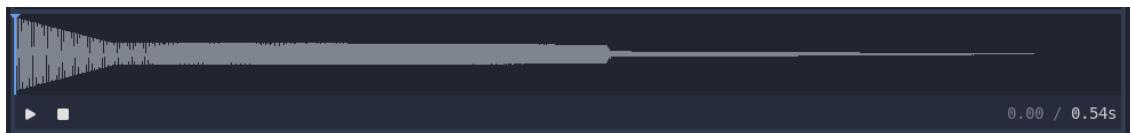


Figura 3.12: Espectro de Ondas del sonido de Confirmación de IGASCE



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

3.4.3. Diseño de mecánicas de juego

La condición de victoria en cada nivel implica completar todas las instrucciones proporcionadas por el algoritmo. En los niveles de BFS y DFS, esto implica visitar todos los nodos del grafo. El puntero de instrucciones señala la tarea actual, y el jugador debe ejecutar correctamente lo indicado en la instrucción, ya sea haciendo clic, respondiendo una pregunta, o

agregando un nodo a una estructura de datos. Algunas instrucciones, especialmente aquellas que contienen la palabra clave *for*, no requieren acción por parte del jugador. Una vez que se han ejecutado correctamente los pasos requeridos, la instrucción cambiará de color y el cursor se transformará en un ticket.

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

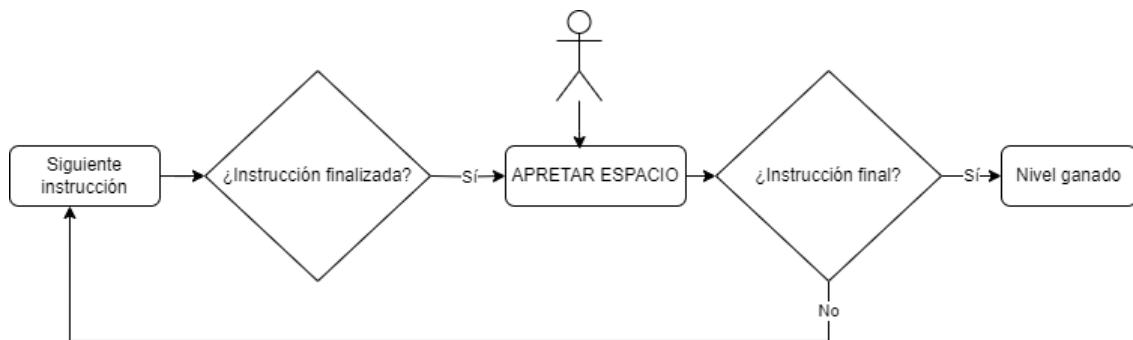


Figura 3.14: Flujo de mecánicas de Nivel. El jugador debe seguir las instrucciones paso a paso.

3.5. Arquitectura de software

Antes de abordar la arquitectura específica del programa, es crucial comprender cómo se estructuran los programas en Godot. Esta comprensión sirve como base para justificar las decisiones de diseño adoptadas durante el desarrollo de la aplicación.

3.5.1. Arquitectura de una aplicación en Godot

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

Los nodos en Godot son las unidades fundamentales de construcción del programa, representando diversos elementos como botones, texturas, reproductores de sonido o áreas de colisión. Para crear personajes con lógica compleja, como habilidades, movimiento y colisiones, se combinan múltiples nodos.

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

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

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

3.5.2. Arquitectura de software de la aplicación IGASCE

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

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

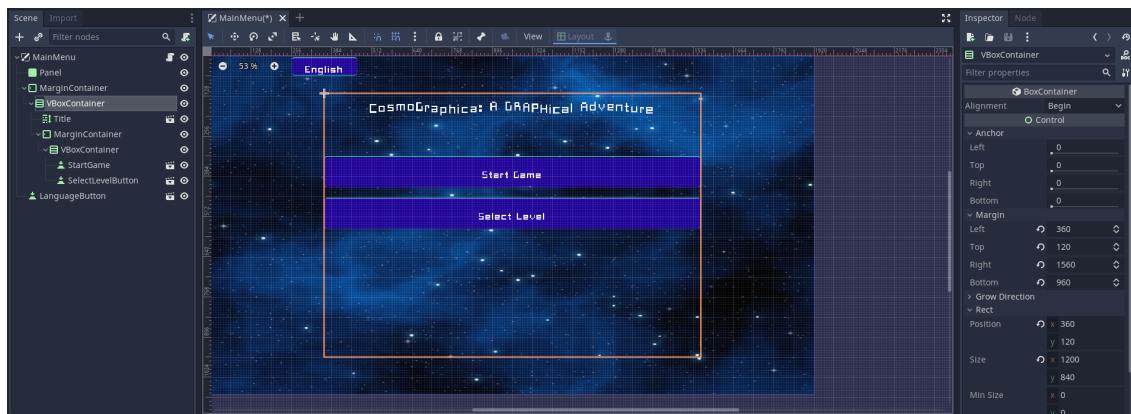


Figura 3.15: Interfaz de Godot describiendo la escena del menú principal de IGASCE. El juego públicamente se dio a conocer como CosmoGraphica, pero el proyecto completo se llama IGASCE.

Los tutoriales presentan una lógica más compleja y se asemejan a los niveles. A diferencia de las escenas finales, los tutoriales incorporan una ventana de diálogo que busca vincular la historia del videojuego con los elementos interactivos. Además, en los tutoriales, el grafo ya está predefinido, lo que significa que los nodos y 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.

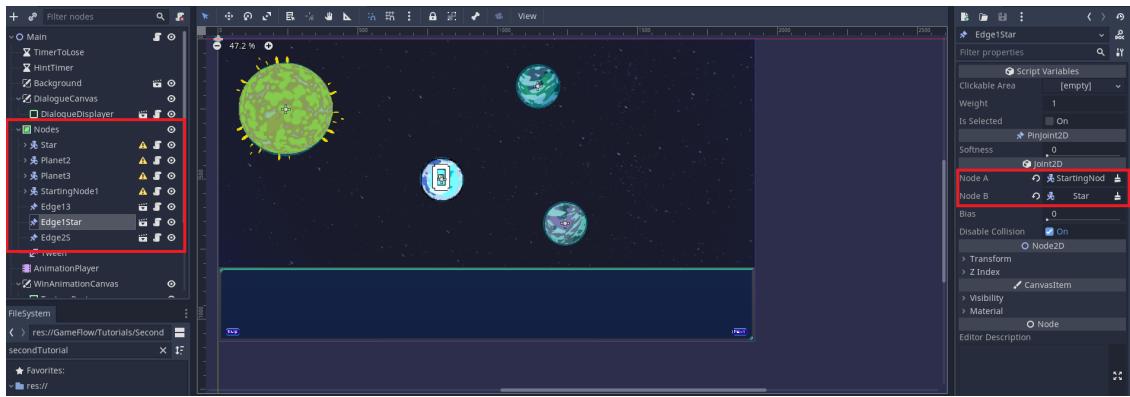


Figura 3.16: Interfaz de Godot describiendo el segundo tutorial. En rojo se observa el grafo de escena con los nodos y los arcos ya predefinidos a la izquierda. En la derecha se muestra cómo se unen los arcos, indicando los dos extremos del arco a través de variables exportadas de Godot.

El código que se muestra a la derecha de la pantalla durante el juego está controlado por un nodo de la clase CodeContainer. Este objeto se encarga de recibir la entrada del jugador para avanzar en el código. Si las acciones ejecutadas son correctas, el usuario pasa a la siguiente instrucción, desbloqueando nuevas tareas. La clase CodeContainer posee un arreglo de objetos del tipo CodeLine llamado code_lines. La clase CodeLine contiene toda la lógica de una única línea de código. Entre sus funciones se encuentran dar énfasis a la línea de código actual, proporcionar feedback 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 ??).

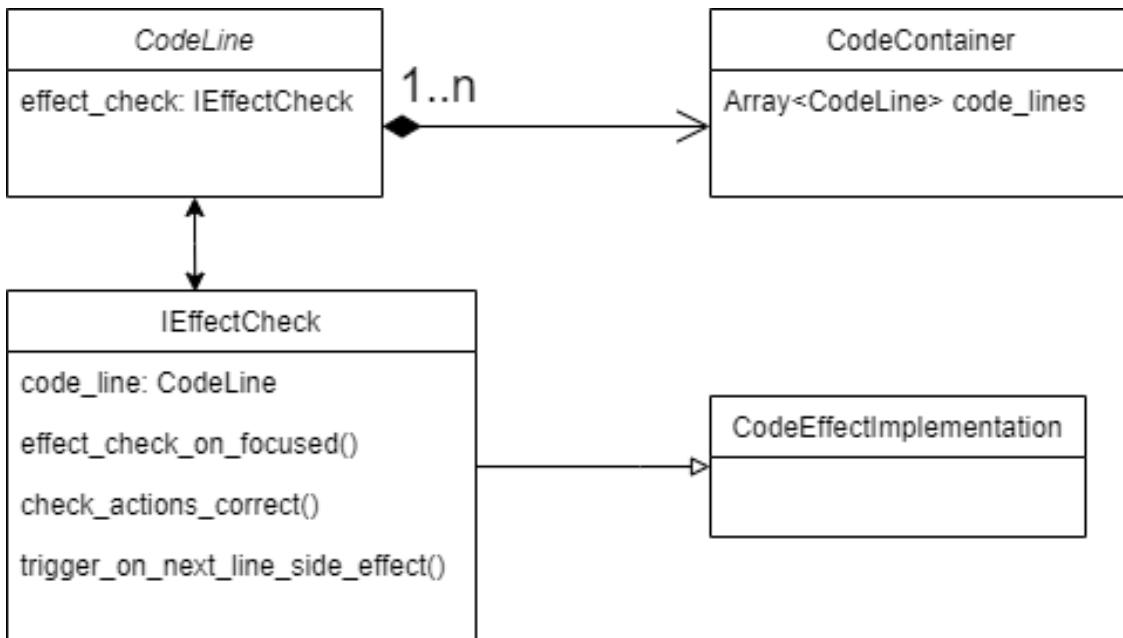


Figura 3.17: Arquitectura de software encargada de la mecánica de ejecución de código dentro del juego.

Cada CodeLine posee un script que se puede indicar desde el editor de Godot a través de

una variable exportada. Este script debe heredar de la clase EffectCheck, la cual actúa como interfaz. Cada script que implemente la interfaz EffectCheck debe definir métodos para: 1) cuando el puntero de instrucción llega a la línea del script; 2) verificar que la instrucción actual se ha realizado correctamente; y 3) si es necesario, manejar efectos colaterales de la instrucción. Este último se utiliza en los ciclos for, donde se mueve el índice utilizado para avanzar por los ciclos y recorrer arreglos, así como para la creación de variables.

En el panel inferior, se encuentran dos clases esenciales: DebugBlock y ADTShower. DebugBlock se encarga de mostrar las variables instanciadas, su nombre y su valor actual. Contiene la lógica para agregar, modificar o eliminar alguna variable. Cada vez que se pasa por un ciclo for o se declara una variable, este ajusta sus variables internas. Hay una variable seleccionada que puede ser manipulada por el usuario de diversas maneras. Al presionar la flecha hacia arriba, se selecciona la variable que está encima de la actual y viceversa para la flecha hacia abajo. En la figura ??, se observa que la variable “q” está seleccionada.

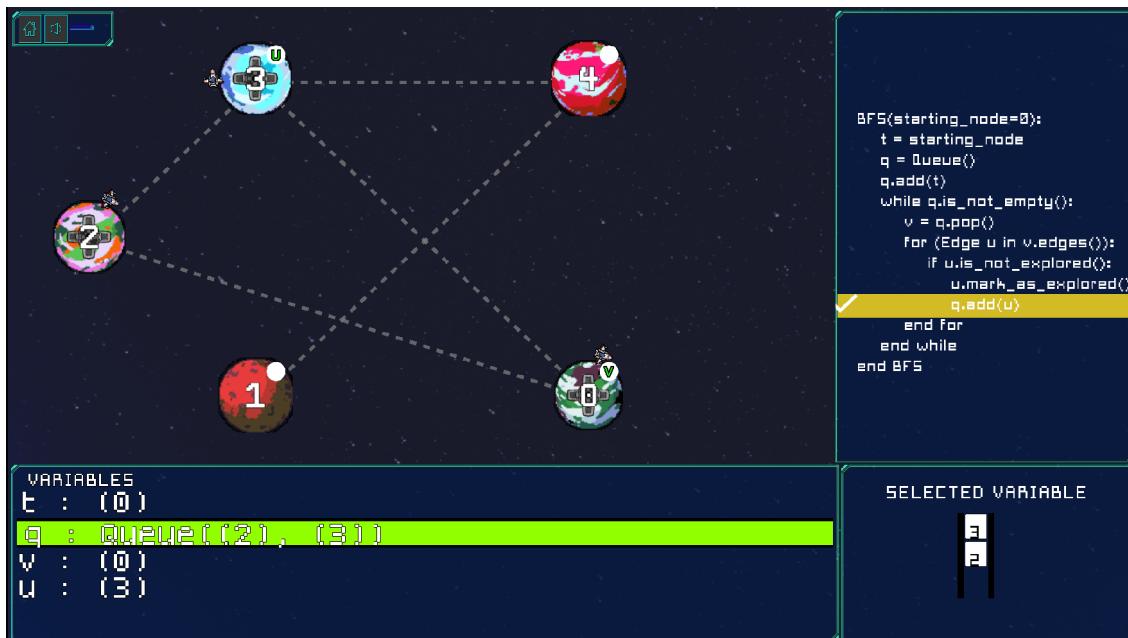


Figura 3.18: Nivel jugable BFS. Se observa el código a la derecha, el grafo en el centro, el DebugBlock en la parte inferior izquierda y el ADTShower en la esquina inferior derecha. La variable q está seleccionada.

La clase ADTShower se encarga de mostrar en la esquina inferior derecha la variable que ha sido seleccionada por el usuario. Busca añadir una representación visual al objeto seleccionado para permitir arrastrar nodos a la cola o la pila según el algoritmo que se esté enseñando, BFS o DFS, respectivamente.

Inicialmente, cada cambio en el código debido al efecto de alguna CodeLine o línea de código del juego requería modificar las clases DebugBlock, ADTShower y StoredData (explicada más adelante), además de afectar la misma línea de código. Además, estos efectos deben reflejarse en las líneas de código siguientes para lograr cohesión. Por ejemplo, si la quinta línea crea la variable “u” y la sexta la modifica, esto se refleja correctamente en todas las estructuras mencionadas.

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

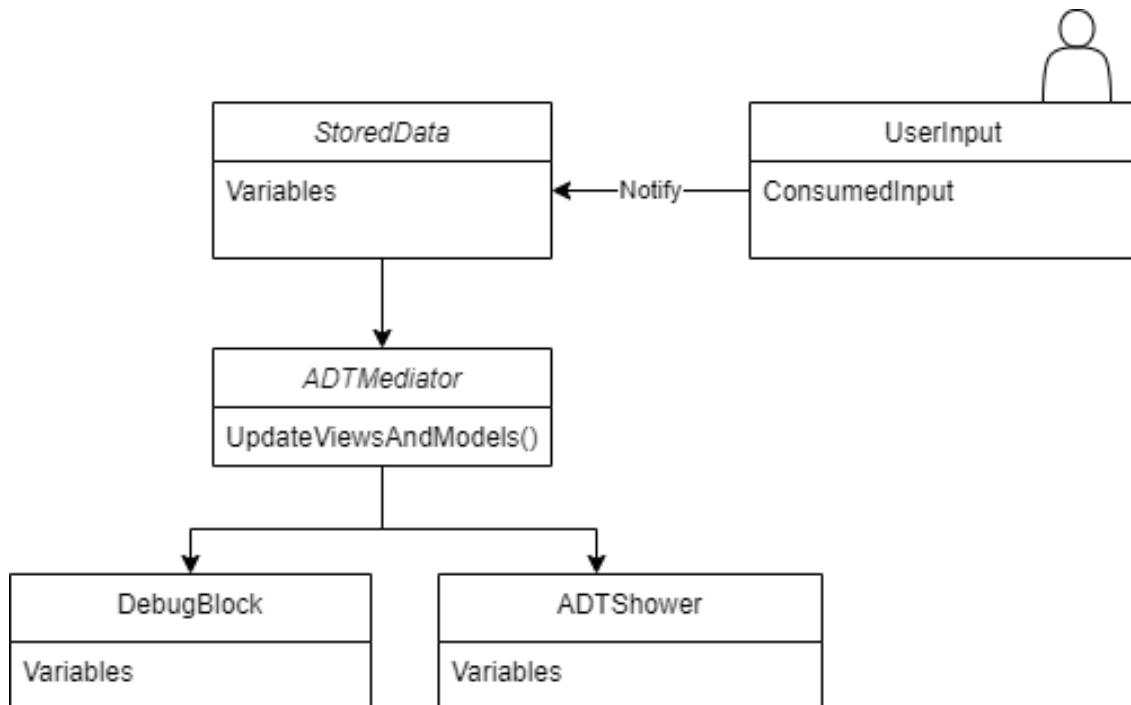


Figura 3.19: Diagrama de un nivel utilizando el ADTMediator, reduciendo el acoplamiento entre clases.

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

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

Capítulo 4

Diseño experimental: Metodología

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

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

4.1. Fases del trabajo de investigación

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

4.1.1. Fase exploratoria

En esta primera fase, el objetivo era 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 las siguientes razones: 1) Si un usuario conoce el algoritmo y la estructura de datos pero no comprende el videojuego, entonces existen problemas de usabilidad, justificando comenzar con personas expertas. 2) Los usuarios expertos tienden a expresarse de manera más natural cuando saben que no se espera un formulario al final, lo que permite obtener opiniones más espontáneas en el momento, por lo que se prefirió no utilizar una encuesta o escala posterior a la experiencia.

Es crucial que los usuarios expresen sus impresiones a medida que los elementos del videojuego aparecen en pantalla y no después de la experiencia, para comprender mejor lo que sienten 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.

Se dio por concluida esta fase una vez que los resultados de las pruebas de usuario mejoraron y se observó que cinco usuarios expertos pudieron completar la prueba de principio a fin sin inconvenientes. La condición era que estos usuarios no hubieran tenido conocimiento previo del juego.

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.

Para iniciar esta etapa, se realizó una convocatoria voluntaria en el foro de las secciones de Algoritmos y Estructuras de Datos de la Universidad de Chile durante el semestre de primavera del año 2023. Se ofreció una remuneración a todas las personas que completaran la experiencia, que tenía una duración promedio de 45 minutos y constaba de tres partes: realizar la prueba de usuario, completar un formulario que utilizaba la escala de Likert basado en el modelo MEEGA+ [?], y responder a una prueba escrita basada en exámenes previos del curso Algoritmos y Estructuras de Datos.

El modelo sistemático MEEGA+ [?] 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.

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

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

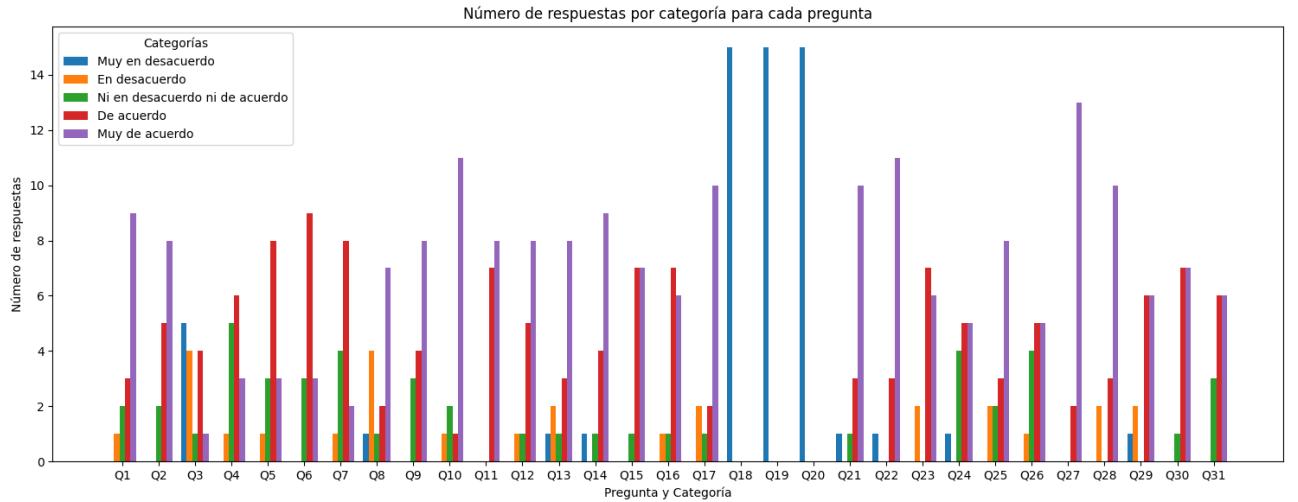


Figura 5.1: Agregación de respuestas por categoría y por pregunta

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

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+ [?], 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 [?], 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+ [?].

5.3. Resultados de la encuesta libre

Esta encuesta sigue libre al momento de la escritura de esta tesis. Hasta ahora, se han recolectado 11 respuestas. El resultado de la percepción con la metodología [?] 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 se puede deber al sesgo del voluntario [?], puesto que la comunidad que participó de este trabajo es una comunidad indie de creación de videojuegos, por lo que aprecian cada aspecto de los videojuegos y reconocen la dificultad de realizarlos.

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

Capítulo 6

Discusión

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. [?]. Esta medida indica que las preguntas evalúan efectivamente lo que el cuestionario buscan evaluar [?]. 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 [?, ?].

El proceso de diseño se considera robusto, ya que incorporó retroalimentación de expertos y la aplicación de la metodología de "Loud Thinking." en cada iteración. Esto permitió que los jugadores se familiarizaran adecuadamente con el juego, culminando en la exitosa realización de la prueba escrita. Se ha comprobado empíricamente cómo la usabilidad mejoró 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 [?]. 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, comparando 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) Idealmente, tomar una muestra aleatoria de estudiantes 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+ [?] 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 [?]. 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 STEM y agregar segmentación.

Además, según libros como [?], 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 [?, ?].

6.3. 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. Hay frameworks que sirven como servidor para aplicaciones relacionadas con videojuegos y ofrecen servicios de telemetría y análisis de datos, como PlayFab [?]. 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 [?]. 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 [?], 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.

Capítulo 7

Conclusión

El videojuego logra 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.

Además, el juego ha recibido una evaluación positiva y se considera, según el modelo MEEGA+ [?], como un juego de 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 y, por ende, en los resultados académicos. Aunque se reconocen las áreas de mejora, este trabajo sienta las 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 podría ser profundizada y perfeccionada en trabajos futuros para evaluar de manera más precisa la efectividad y comprensión de los conceptos enseñados mediante este enfoque lúdico.

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] 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.
- [6] Paulo Eduardo Battistella and Christiane Gresse von Wangenheim. Games for Teaching Computing in Higher Education – A Systematic Review. 2016.
- [7] 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.
- [8] CADCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [9] Godot Community. Godot export variables, 2023. Accessed on 5 October 2023.
- [10] Wikipedia contributors. Golden sun (video game) wikipedia, the free encyclopedia, 2023. Accessed 04-October-2023.
- [11] Tomorrow Corporation. 7 billion humans - now available!, 2023. Accessed: 2023-11-15.
- [12] DCC. Departamento de ciencias de la computación de la universidad de chile. Accessed on 2023-09-27.
- [13] Sarah E. Domoff, Ryan P. Foley, and Rick C. Ferkel. Addictive phone use and academic performance in adolescents. *Human Behavior and Emerging Technologies*, 2019.

- [14] 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.
- [15] 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.
- [16] Adam Freeman. The mediator pattern. 2015.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] Godot Community. Godot official documentation, 2023. [Online; accessed 4-October-2023].
- [22] 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.
- [23] 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.
- [24] Ioannis Karatzas. and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, Berlin, 2nd edition, 2000.
- [25] Kemono Games. Protocorgi, 2023. [Online; accessed 4-October-2023].
- [26] Kristian Kiili, Keith Devlin, Arttu Perttula, Pauliina Tuomi, and Antero Lindstedt. Using video games to combine learning and assessment in mathematics education. *International Journal of Serious Games*, 2(4):37–55, 2015.
- [27] 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.

- [28] Francisca Loreto Calderón Maldonado. Statistical methods for the analysis of polytomous response data. 2021.
- [29] 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.
- [30] Microsoft. Playfab playstream: Real-time event processing and monitoring for games. <https://phys.org/news/2023-07-korean-team-room-temperature-ambient-pressure-superconductor.html>. Accessed: 2023-11-18.
- [31] Microsoft. Visual studio code, 2023.
- [32] United Nations. The impact of digital technologies, 2023. Accessed: 2023-08-04.
- [33] 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.
- [34] Giani Petri and Christiane Gresse von Wangenheim. How games for computing education are evaluated? a systematic literature review. *Comput. Educ.*, 107:68–90, 2017.
- [35] 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.
- [36] 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.
- [37] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [38] Yvonne Rogers, Helen Sharp, and Jennifer Preece. Interaction design: Beyond human-computer interaction. 2002.
- [39] 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.
- [40] Grant Sanderson. 3blue1brown youtube channel. <https://www.youtube.com/3blue1brown>, 2023. Accessed on 4 October 2023.
- [41] Neil Selwyn. Looking forward : Reimagining digital technology and the contemporary university. 2014.
- [42] Ismar Frango Silveira. Building effective narratives for educational games. *2019 XIV Latin American Conference on Learning Technologies (LACLO)*, pages 299–305, 2019.

- [43] GQS UFSC. Meega+ a model for evaluating educational games. <http://www.gqs.ufsc.br/quality-evaluation/meega-plus/>. Accessed: 2023-Nov-2.
- [44] Alonso Utreras. Igasce github repository. <https://github.com/AlasAltum/Thesis-IGACSE>. Accessed: 2023-Nov-2.
- [45] Cheng-Hsin Wang. Comprehensively summarizing what distracts students from online learning: A literature review. *Human Behavior and Emerging Technologies*, 2022.
- [46] David Weintrop and Uri Wilensky. Robobuilder: A program-to-play constructionist video game. In *Proceedings of the constructionism 2012 conference. Athens, Greece*, 2012.
- [47] Wikipedia contributors. Final fantasy iv — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [48] Wikipedia contributors. Pokémon emerald — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [49] Wikipedia contributors. Space invaders — Wikipedia, the free encyclopedia, 2023. [Online; accessed 4-October-2023].
- [50] 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.
- [51] 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.
- [52] 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.
- [53] Weinan Zhao and Valerie J Shute. Can playing a video game foster computational thinking skills? *Computers & Education*, 141:103633, 2019.
- [54] B. Zimmerman and Dale H. Schunk. Handbook of self-regulation of learning and performance. 2011.
- [55] Daniel Zingaro, Cynthia Bagier Taylor, Leo Porter, Michael J. Clancy, Cynthia Bailey Lee, Soohyun Nam Liao, and Kevin C. Webb. Identifying student difficulties with basic data structures. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 2018.

Anexo A: Formulario basado en MEEGA+

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

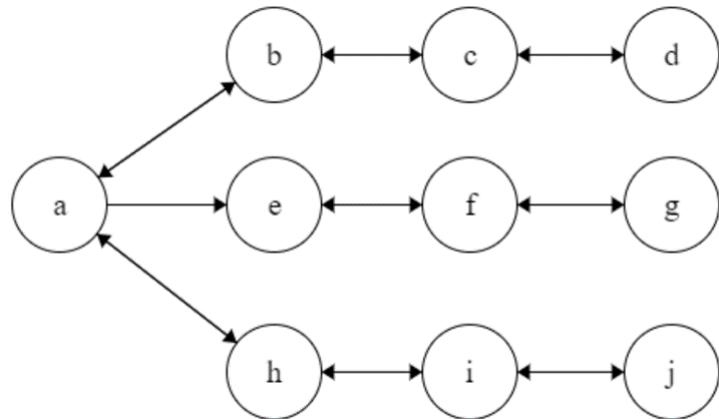
Notación	Pregunta
Q1	El diseño de juego es atractivo
Q2	La fuente de texto y los colores están bien combinados y son consistentes
Q3	Tuve que aprender cosas antes de poder jugar
Q4	Aprender a jugar se me hizo fácil
Q5*	Creo que la mayoría de la gente aprendería a usar este juego rápidamente.
Q6	Creo que el juego es fácil de jugar.
Q7	Las reglas del juego son claras y fáciles de entender
Q8	Las fuentes (su tamaño y estilo) usadas son fáciles de leer
Q9	Los colores usados en el juego son significativos
Q10*	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.

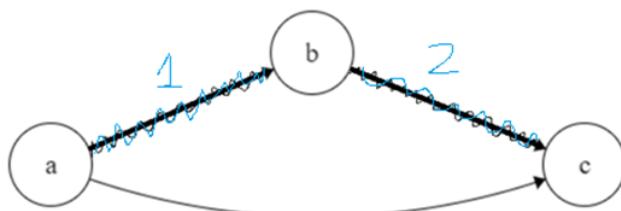
Anexo B: Prueba de conocimiento de grafos

Test para verificar aprendizaje de BFS y DFS

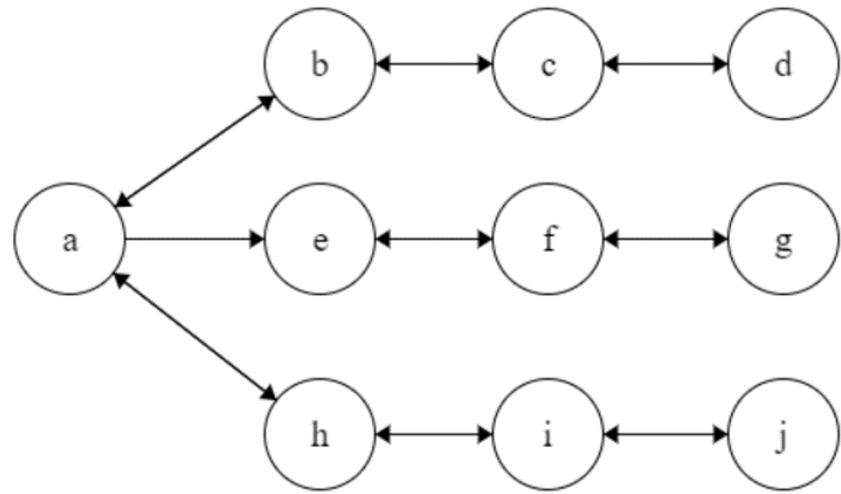
Consider el siguiente grafo:



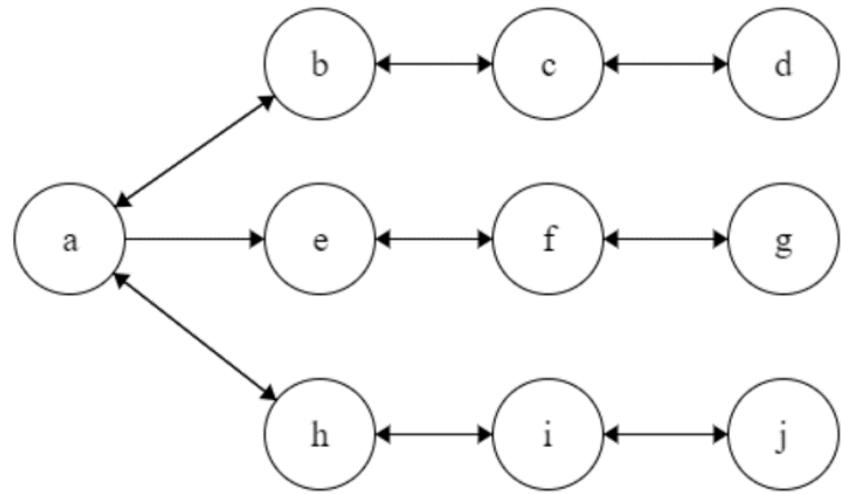
Muestre el recorrido de un grafo desde el nodo a utilizando distintos los dos algoritmos de búsqueda vistos. **Agregue una numeración al lado de sus arcos para indicar el orden en que se explorando.** Por ejemplo, en el siguiente grafo, si se recorre a, b y c partiendo desde a, pasando por b y llegando 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 7.1: 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 .“espacio”, 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.