Algoritmos genéticos para resolver "El Laberinto del Pingüino"

Elizabeth Méndez López

Resumen– Clasificación por niveles de mapas de un videojuego utilizando una estrategia de Inteligencia Artificial (IA). En este caso la estrategia utilizada es un algoritmo genético. Se describe la teoría de un algoritmo genético, y cómo se debe adaptar para resolver el videojuego propuesto. Además, se describen las características del juego y los resultados obtenidos con el algoritmo genético. Se utiliza el algoritmo genético para resolver el mapa como determinante del nivel de dificultad.

Palabras clave– IA, Algoritmos genéticos, Videojuego, Javascript, Clasificación por Nivel.

Abstract– Map level classification of a videogame using an Artificial Inteligence (AI). In this case, the AI used is a genetic algorithm. It is described the theory of genetic algorithms, and the steps taken to adapt it to the videogame prepared. Furthermore, the characteristics of this videogame are described and the results obtained from the genetic algorithm. The genetic algorithm is used to solve each map and as a metric for its difficulty.

Keywords– AI, Genetic Algorithms, Videogame, Javascript, Level Classification.

1 Introducción

En este proyecto se va a crear un juego en el que el objetivo es ir de un punto inicial del mapa a un punto final del mismo superando obstáculos, y a su vez medir de manera cuantitativa la dificultad que supone. Para medir la dificultad, se utilizarán algoritmos genéticos [1] con el fin de simular los movimientos del jugador. El juego está basado en el juego de Pacman [4]. Los movimientos de los obstáculos móviles, en el caso de Pacman, los Fantasmas [6]. Y en este caso, enemigos con forma de animal que se moverán con un patrón ya definido para simplificar el proyecto.

El juego será generado en *Javascript* [2] debido a la facilidad de adaptación de los algoritmos genéticos, además de la familiaridad con este lenguaje de programación. La utilización de algoritmos genéticos para conseguir el objetivo del juego, nos permitirá averiguar la dificultad de cada mapa.

• E-mail de contacto: 1361994@uab.cat
• Mención realizada: Tecnologías de la Información
• Trabajo tutorizado por: Ana Oropesa Fisica
• Curso 2022/23

Este trabajo se estructura de la siguiente manera, primero de todo se define la teoría de la diversión [Cap 4], seguido por el concepto de Inteligencia Artificial (IA) [Cap 5] y la teoría de la evolución [Cap 6]. Y por último, se describen los algoritmos genéticos [Cap 7]. Estos engloban los conceptos de evolución para completar el comportamiento de una IA. Se indica en qué consisten y los pasos para utilizarlos.

2 Objetivos

El objetivo principal de este proyecto es determinar qué mapa es el que se le debería presentar a un jugador una vez ha acabado un nivel del juego, con la finalidad de mantener su atención.

Lo primero a lograr es ordenar los mapas según su dificultad. Para hacer eso, se utilizará un algoritmo genético que recorra varios mapas y así determinar de manera cuantitativa su dificultad a través del número de generaciones que requiera el algoritmo para resolverlo.

El objetivo principal del videojuego propuesto es ir de un punto a otro del mapa del juego sin encontrarse obstáculos. El algoritmo genético efectuará los movimientos que puede hacer el jugador.

A continuación, se muestran todos los objetivos específicos y una breve descripción de ellos, para poder llevar a cabo este objetivo principal mencionado anteriormente:

1. **Generar mapas del juego** — Para poder generar los mapas del juego, hay diferentes pasos a realizar. Entre ellos, investigar la mejor manera de generar mapas con *Javascript*. Además, se va a querer mostrar el movimiento que efectúa el algoritmo genético, por lo tanto, investigar la representación disponible en este lenguaje de programación. Y por último la generación de los mapas con diferentes obstáculos, concretamente paredes y enemigos.

2. **Diseñar juego** — Se diseñará todo lo que conlleva generar el juego. Por ejemplo, un diccionario con las posiciones de las paredes en cada uno de los mapas, y otro que contenga las posiciones de los enemigos. Otro diccionario necesario para la comprobación de las decisiones de los algoritmos genéticos sería el de los movimientos correctos dentro del mapa.

3. **Adaptación de los algoritmos genéticos al juego** — Generar la representación de los movimientos de los algoritmos genéticos dentro de los mapas para poder observarlos y recoger los valores obtenidos una vez los algoritmos hayan resuelto todos los mapas.

4. **Ordenar los mapas según su dificultad** — Para saber si es el adecuado para mantener la atención del usuario mientras juega.

3 METODOLOGÍA

Se ha escogido la metodología Kanban [3] por su facilidad de uso y su visualización de las tareas. Al poder observar todas las tareas pendientes, las que están completadas y las que están en curso, permite una organización más clara sobre qué es lo que se tiene que hacer a continuación.

Se diferencian tres elementos en un tablero donde se representan las tareas: Pendientes, En curso y Realizadas. Para que sea más sencilla la organización y visualización de las tareas, debería determinarse un límite de tareas que se puedan mostrar a la vez en el tablero. Y así no acumular tareas en pendientes hasta que no se resuelvan otras antes. El límite de tareas puede cambiar según su utilidad, en el caso en el que necesite ver más tareas para organizar el trabajo actual, aumentaría ese límite.

Para la aplicación práctica de esta metodología se va a utilizar un programa llamado Notion [7], con una página personalizada y diseñada específicamente para el proyecto [8] con un apartado reservado para ser utilizado con la metodología mencionada.

4 LA DIVERSIÓN EN LOS JUEGOS

Existen varias teorías sobre lo que significa la diversión. Hasta las grandes empresas como Volkswagen [18] han efectuado experimentos para averiguarlo.

En 2009 en Estocolmo convirtieron las escaleras de una estación de metro en un piano para comprobar si el sonido de las teclas generaba suficiente diversión para conseguir que la gente las utilizara más que las mecánicas [18]. Y quedó demostrado que sí, un 66 % más de personas utilizaron las escaleras convencionales.

El Doctor en Psicología Mihali Csikszentmihalyi [19] a través de sus investigaciones desarrolló el término estado de flujo, porque muchas de las personas a las que entrevistó le describieron sus estados óptimos de desempeño como ejemplos en los que su trabajo simplemente fluyó de ellos sin mucho esfuerzo.

Su objetivo era descubrir cómo la creatividad conduce a una mayor productividad [20]. Para llegar a este estado, Mihaly describe varios puntos que deben darse a cabo:

- Concentración completa
- Claridad de objetivos y recompensa
- Transformación del tiempo (tanto aceleración como desaceleración del tiempo)
- Experiencia gratificante
- Equilibrio entre desafío y habilidades
- Las acciones y la conciencia se fusionan
- Sensación de control

La conclusión de Mihaly fue que la felicidad es un estado interno del ser, no externo. Y además, no es un estado rígido que no se puede cambiar, todo lo contrario, requiere un esfuerzo comprometido para manifestarse.

A diferencia de Mihaly, otros como Malone o Lepper se basaban en la motivación intrínseca para determinar qué era la diversión [21]. Específicamente, la diversión en un videojuego.

Una vez introducido el concepto de motivación, Malone y Lepper mencionaban dos tipos de motivaciones: la individual y la interpersonal. Según ellos, existían cuatro tipos de motivación individual: Reto, Curiosidad, Control y Fantasía. En cuanto a la motivación interpersonal existían tres tipos: Cooperación, Competición y Reconocimiento.

En conclusión, establecían como completamente necesario para el éxito en la motivación intrínseca, definir unos objetivos claros.

En este trabajo se va a crear un videojuego, con la intención de utilizar todos los conceptos anteriores sobre la diversión para hacerlo lo más entretenido posible. La intención es poder aplicar estas teorías de la diversión a un futuro jugador del juego. Para ello, se utilizaran los algoritmos genéticos para emular a un jugador y así poder evaluar la dificultad de todos los mapas.

5 Inteligencia Artificial

Se puede empezar a definir la inteligencia artificial como un concepto sin una definición única y exclusiva [16]. Eso se debe a su naturaleza cambiante, experimental y reciente. Asimismo tampoco existe una definición única y exclusiva de la inteligencia humana.

La inteligencia artificial es, en términos de ciencias de la computación, la disciplina que pretende replicar los procesos cognitivos humanos, es decir, todo aquello que se podría considerar inteligencia humana, a través de ordenadores.

En la actualidad, la inteligencia artificial se ve implicada en una gran variedad de campos de conocimiento, entre ellos el aprendizaje y percepción. Otros más concretos pueden ser la aplicación en el juego de ajedrez, la demostración de teoremas de cualquier disciplina, la escritura de poesía y incluso el diagnóstico de enfermedades.

Se pueden llegar a sintetizar y automatizar labores consideradas intelectuales, por lo tanto, es una disciplina relevante para cualquier aplicación que requiera de una capacidad intelectual humana. Por eso se considera un campo genuinamente universal.

6 Teoría de la evolución

Como se ha nombrado anteriormente, la inteligencia artificial puede tomar muchas formas y métodos. La que se va a utilizar en este trabajo se basa fundamentalmente en el concepto de de la teoría de la evolución [12].

La teoría de la evolución, en realidad una serie de hechos demostrados, definida por Charles Darwin en el libro "*Sobre el Origen de las Especies por medio de la Selección Natural*". La hipótesis planteada por Darwin y Wallace era la siguiente: "Pequeños cambios heredables en los seres vivos y la selección son los dos hechos que provocan el cambio en la Naturaleza y la generación de nuevas especies".

Pero Darwin desconocía el método por el cual se adquirían estos cambios heredables. Fue Lendel, el padre de la genética, quien descubrió que la manera de heredar las características se trataba de algo discreto, es decir, o se hereda del padre o se hereda de la madre. Y a su vez, depende de si se trata de un carácter dominante o recesivo. A los caracteres que pueden tomar diferentes valores se les llamó genes.

En los años 50, Watson y Crick descubrieron que la base molecular de los genes está en el ADN. El ADN se encuentra en los cromosomas, descubiertos años antes por el biólogo alemán Walther Flemming. Se determinó que los cromosomas están compuestos de ADN, y por lo tanto, los genes están en los cromosomas.

La macromolécula de ADN está formada por una secuencia única para cada ser vivo, llamada código genético, también llamado genotipo que es el encargado de codificar todas las proteínas que forman parte de un ser vivo. En cambio, al cuerpo que construyen esas proteínas, el cual se ve afectado por la presión ambiental, historia vital y otros mecanismos dentro del cromosoma, se le llama fenotipo.

Todo lo nombrado se engloba en la teoría del neodarwinismo, que indica que la historia de la mayoría de la vida viene dada por una serie de procesos que actúan en y dentro de las poblaciones, estos procesos son los mostrados en la figura 1:

Figura 1: Procesos de la evolución

La evolución por lo tanto, se puede definir como los cambios efectuados en el conjunto genético de una población. A continuación se va a determinar cómo se utilizará la definición de evolución en el entorno de la inteligencia artificial.

7 Unión entre Inteligencia Artificial y Teoría de la evolución

Se explica a continuación la unión entre el concepto de inteligencia artificial y el mecanismo de evolución definidos en los apartados anteriores, para crear los algoritmos genéticos.

7.1. Algoritmos genéticos

Los primeros algoritmos genéticos aparecieron a finales de los años 50 y a principios de los años 60. Pero no fueron creados por informáticos, sino por biólogos, y su objetivo era realizar modelos de aspectos de la evolución natural. En ese momento no se plantearon que pudieran ser utilizados para resolver otras situaciones como las que se pueden llegar a considerar en la actualidad [10].

Han existido mecanismos que se han ido modificando y mejorando partiendo de esos primeros algoritmos creados hasta construir lo que se usará en este trabajo. Consideraron la mecánica de generar un hijo a partir de un sistema o programa, y mantener el mejor de los dos. Posteriormente se hizo lo mismo pero con varios sistemas, lo que no consideraron fue el hecho de cruzar las máquinas para obtener lo mejor de cada una. Esta técnica fue llamada programación evolutiva [10].

Lo que realmente sentó unas bases fue el trabajo de John Holland sobre sistemas adaptativos, ya que introdujo el con-

cepto de cruzamiento, además de otros operadores de recombinación. Sin embargo, el paso definitivo fue el libro "Adaptación en Sistemas Naturales y Artificiales" basado en investigaciones y trabajos del mismo Holland. Fue el primero en presentar de manera sistemática y rigurosa los sistemas digitales adaptativos con los mecanismos de mutación, selección y cruzamiento. Simulando la evolución biológica como estrategia para resolver problemas [12].

7.1.1. ¿Qué es un algoritmo genético?

Un algoritmo genético es un método sistemático utilizado para la resolución de problemas de búsqueda y optimización que se basan en los métodos de la evolución biológica: selección basada en la población, reproducción y mutación [12].

Dado un problema específico a resolver, lo que se le aporta al algoritmo es un conjunto de soluciones potenciales al problema, además de una métrica llamada función de aptitud. Esta función indica cómo de afines son estas soluciones al resultado esperado, dando una evaluación cualitativa para cada una. Es posible que estas soluciones ya sean comprobadas y se espera una optimización, o pueden haber sido generadas aleatoriamente.

De acuerdo con esta afinidad, el algoritmo evalúa cada candidata. Se tiene que tener en cuenta que las primeras candidatas generadas serán las menos óptimas y eficientes, o no servirán en absoluto. Sin embargo, por evolución, repetición y cruce es posible generar alguna solución más favorable.

Una vez se obtiene una solución más cercana a la esperada, se conserva y se le permite reproducirse para transmitir ese conocimiento. Se realizan copias, no perfectas, sino que sufren cambios aleatorios a modo de mutaciones al crear sus considerados descendientes. De nuevo, se evalúan para comprobar su afinidad a la solución esperada y se repite el proceso las veces necesarias hasta obtener un resultado que nos parezca correcto.

El proceso descrito anteriormente se puede observar en la figura 2:

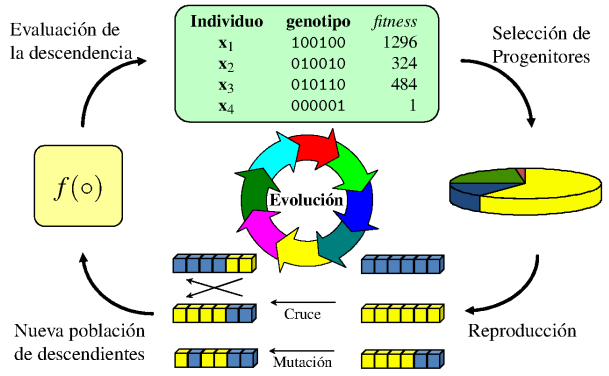

Figura 2: Proceso de un algoritmo genético [10]

Un algoritmo genético es independiente del problema, por lo tanto, se puede considerar robusto al poder utilizarlo para resolver cualquier problema. Sin embargo, también puede ser considerado todo lo contrario debido al hecho de no estar optimizado o especializado para ningún problema.

7.1.2. Ventajas de los algoritmos genéticos

A continuación se mostrarán las ventajas que aportan los algoritmos genéticos a la resolución de problemas:

- No tienen muchos requisitos matemáticos.
- Debido a su naturaleza evolutiva buscan soluciones sin que el algoritmo genético requiera de específico conocimiento del problema.
- Proporcionan una gran flexibilidad que implica unas implementaciones eficientes.
- Pueden manejar toda clase de funciones y restricciones definidas sobre un espacio de búsqueda discreto, continuo o mezclado.
- Su estructura los hace efectivos a la hora de realizar búsquedas globales.

7.1.3. Desventajas y limitaciones

Los algoritmos genéticos no son perfectos, por lo tanto, se deben considerar algunas limitaciones o desventajas que proporcionan:

- Si el problema a resolver es muy complejo, la función de evaluación es posible que sea demasiado costosa en cuanto a tiempo y recursos.
- Es posible que no se encuentre una solución óptima, o converger en una solución prematura con resultados no satisfactorios
- No son escalables con la complejidad.
- No existe un criterio real que seguir en cuanto a cuándo detener el algoritmo, ya que las soluciones se basan en otras soluciones, no a un resultado específico.
- No se recomiendan para el uso para problemas con respuesta simple, por ejemplo problemas de decisión, ya que el algoritmo difícilmente convergerá y el resultado será casi aleatorio.
- El desarrollador del algoritmo requiere de conocimiento previo del problema para la creación de las funciones que lo forman, para la función de afinidad y los criterios de cruce y selección.

7.1.4. Comportamiento de los algoritmos genéticos

Seguidamente se van a asociar los conceptos mencionados anteriormente en la Teoría de la evolución [Cap 6], con el proceso mencionado previamente en [Cap 7.1.1].

Los cromosomas, como ya hemos visto, son los que contienen los genes. En este caso, los genes serán los evaluados para poder cuantificar la afinidad de cada hijo. Para el juego que se está creando, los genes se consideran los movimientos que se han efectuado en el mapa.

El comportamiento se inicia generando aleatoriamente una población de cromosomas para poder empezar el proceso. A continuación:

1. **Selección:** se evalúan los genes de esos cromosomas con una puntuación
2. **Cruce:** Según esa puntuación, se les permitirá a esos cromosomas reproducirse
3. **Mutación:** Se emparejan los cromosomas de la nueva población, intercambiando material genético, siendo alguno de ellos alterado debido a una mutación espontánea.

Este proceso se repite tantas veces como se requiera para obtener el resultado esperado. Cada uno de esos pasos tiene la libertad de definirse de diversas maneras, que se desarrollan en los siguientes apartados.

7.1.5. Selección

En este paso se evalúa de manera cualitativa como de afín es un hijo para resolver el problema. Una vez se ha definido, se deben seleccionar los más afines para transmitir los buenos genes a la siguiente población. Pero esta selección puede efectuarse de muchas maneras:

- **Selección elitista:** los mejores hijos son reservados para la siguiente generación sin que entren en el proceso de selección. El número de hijos debe ser establecido con anterioridad. En el caso en el que no se encuentren hijos de élite, se seleccionan los más cercanos a lo esperado.
- **Selección proporcional a la afinidad:** los hijos más aptos tienen más probabilidad de ser seleccionados.
- **Selección por rueda de ruleta:** la probabilidad de selección de un hijo se mide según la diferencia de afinidad entre ese hijo y sus competidores.
- **Selección por torneo:** Se separan los hijos en subgrupos y se les hace competir entre ellos. Solo uno de cada subgrupo se selecciona.
- **Selección por rango:** Se le asigna un rango numérico a cada hijo según su afinidad y se seleccionan según ese ranking. Esto proporciona una clara ventaja a los que tienen más afinidad.
- **Selección generacional:** la descendencia de los hijos seleccionados en cada generación se convierte directamente en toda la siguiente generación. No se conservan hijos entre las generaciones.

7.1.6. Cruce

En cuanto ya se han seleccionado los componentes de la siguiente población, se deben alterar de manera aleatoria para mejorar su afinidad para la siguiente generación.

El cruce consiste en el intercambio de material genético entre dos cromosomas de dos individuos. Este mecanismo es el operador genético principal, tal es, que en el caso en el que no existiera, no se podría considerar un algoritmo genético. En cambio, sin mutación seguiría siendo un algoritmo genético, según Holland [10].

Para llevar a cabo este cruce, entrecruzamiento o recombinación, se escogen dos hijos de la población de manera aleatoria para que intercambien segmentos de su código genético y así producir una descendencia artificial. No resulta un inconveniente si esos dos hijos son de los mismos padres, ya que garantiza la perpetuación de genes ya demostrados como buenos. Sin embargo, si sucede con mucha frecuencia puede llegar a ser un problema, ya que la población queda sesgada por esos descendientes y puede pasar por alto genes buenos de otros padres.

Este intercambio genético puede producirse de muchas maneras, pero existen tres principales:

- **Cruce de n puntos:** se cortan los cromosomas por n puntos y el material entre ellos se intercambia. Lo más habitual es efectuar un cruce entre 1 o 2 puntos.
- **Cruce uniforme:** se genera un patrón en la serie de genes de cada hijo, y según el valor asociado se intercambia ese gen en concreto. Se puede utilizar una cadena de unos y ceros, o números aleatorios.
- **Cruces especializados:** en algunas ocasiones, el hecho de efectuar el cruce de manera aleatoria, genera soluciones inválidas. Por lo tanto, se requiere un cruce que genere siempre soluciones válidas, para eso se debe especificar una solución para el problema concreto.

7.1.7. Mutación

En el proceso evolutivo las mutaciones son poco comunes, en muchos casos pueden resultar letales, pero en promedio, contribuyen a la diversidad genética de la especie. En un algoritmo genético no serán diferentes.

La mutación se efectúa de manera simultánea al cruce, ya que debe analizarse cada cadena cuando se va a crear un nuevo hijo. La frecuencia de mutación debe estar establecida de antemano.

No es un mecanismo del que se deba hacer uso en exceso, sin embargo, puede ser la solución cuando un algoritmo se ha estancado. También es posible proporcionar más diversidad aumentando el tamaño de la población, o garantizando la aleatoriedad de la población inicial.

8 Desarrollo

En cuanto al desarrollo del primer objetivo, generar los mapas del juego, se han creado con la ayuda de un programa llamado Tiled [9]. Una vez se ha generado un mapa, se exporta como archivo JSON y se lee la información con las funciones definidas para crear los diferentes diccionarios. Un ejemplo de mapa se puede observar en el archivo JSON mostrado en el Apéndice A.2 y en la figura 4.

Actualmente, se ha conseguido automatizar la creación de los diccionarios que contienen la información de las posiciones de las paredes, los enemigos y los posibles

movimientos. Esta información va a ser útil a la hora de la comprobación de los movimientos efectuados por los algoritmos genéticos.

8.1. Componentes del juego

El juego contiene enemigos, representados por las imágenes que se muestran a continuación en la figura 3. Los números que contienen debajo son los números de identificación que están representados en el archivo JSON ejemplo en el Apéndice A.2.

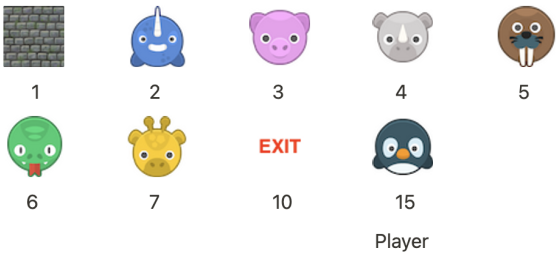

Figura 3: Identificadores mapa ejemplo

Un mapa ejemplo del juego puede verse en la figura 4.

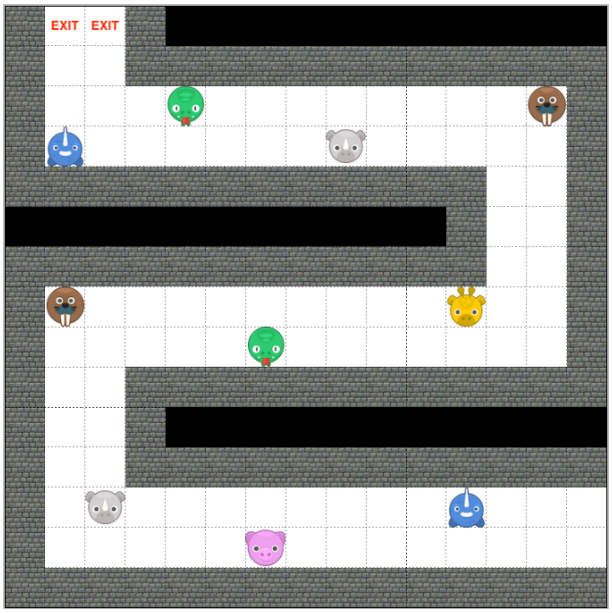

Figura 4: Ejemplo de mapa del juego

8.2. Movimiento de los enemigos

Los enemigos que se ven en la figura 3 no se desplazan libremente, sino que su movimiento es fijo como se puede observar en la figura 5.

Por lo tanto, todas las posiciones donde se halle un enemigo se considerarán como posición de muerte para ese momento.

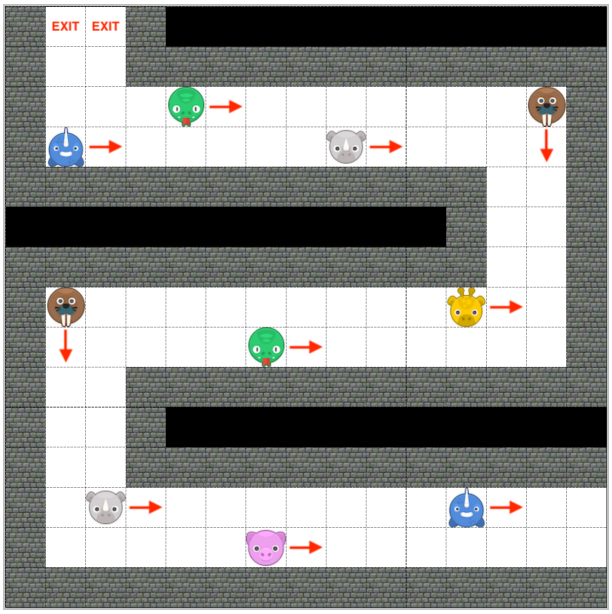

Figura 5: Movimientos enemigos

8.3. Movimiento del jugador

Para representar el movimiento de los algoritmos genéticos se ha utilizado el mismo ejemplo de mapa en la figura 4 mostrado anteriormente para observar cómo debería moverse el algoritmo genético. El movimiento del personaje principal que se puede efectuar es únicamente hacia delante, detrás, y los dos lados, no se permiten movimientos diagonales. Los movimientos posibles son las casillas blancas, las paredes impiden el movimiento pero no perjudican al personaje, los animales que representan el enemigo pueden eliminar al personaje principal.

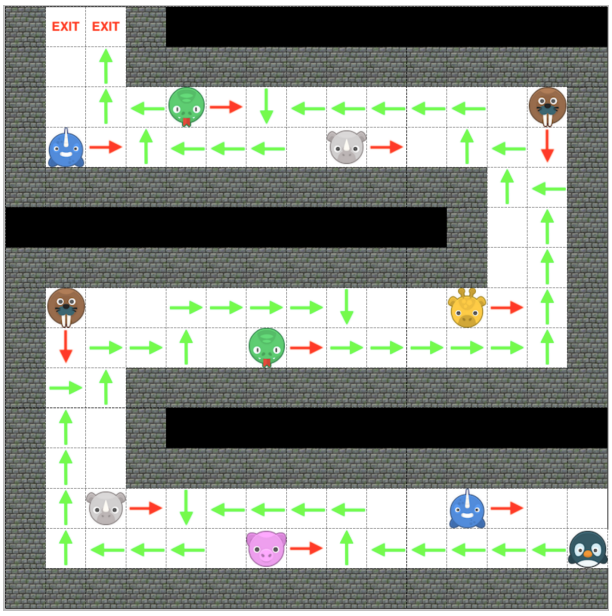

Figura 6: Movimientos del jugador

8.4. Finalidad del juego

El objetivo principal del juego es que el personaje principal llegue a la casilla de salida sin colisionar con ningún

enemigo, evitando las paredes. Tal y como se representa en la figura 6. Si el jugador sigue esos movimientos y llega a la salida, habrá conseguido pasarse el mapa. Y procederá a hacer lo mismo en otro mapa.

8.5. Aplicación de los Algoritmos genéticos

A continuación se va a describir cómo se ha utilizado la información obtenida de la sección sobre la definición de los algoritmos genéticos [Cap. 7.1.1], para aplicarlos en la resolución de este proyecto. Como ya se ha indicado, hay muchas opciones a la hora de utilizarlos, crearlos y aplicarlos. Por eso, se especifica en los siguientes apartados cómo se han implementado en este proyecto.

8.5.1. Diseño de los genes

Es importante definir cómo se va a representar el movimiento en los genes de los individuos de la población, porque de esa manera será como lo transmitan a sus descendientes. Además, se utiliza parte de esos genes como representación de ese usuario, como su genética individual que lo defina.

Los genes se han separado en cromosomas y movimientos, donde los cromosomas son la representación única de ese individuo. Se utiliza una letra y un número aleatorios para representarlos, contienen la información de quién es el individuo.

Los movimientos son las acciones que debe efectuar ese individuo, para luego comprobar su afinidad y decidir si merece la pena preservarlos.

Para simplificar el proceso, se han determinado esos movimientos a partir de unas acciones. Por lo tanto, no se almacena la posición en la que ha estado el predecesor, sino qué acción ha hecho. Dentro de esas acciones, existen 4 posibilidades: arriba, abajo, izquierda y derecha, representadas por los números 0, 1, 2, y 3 respectivamente, las que se pueden observar en la figura 7.

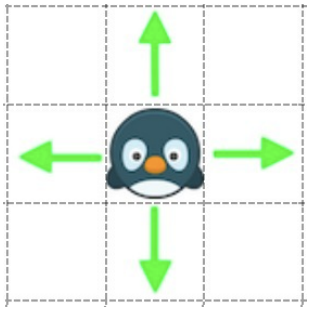

Figura 7: Movimientos posibles

Los genes se representan como se puede observar en la figura 8.

Por ejemplo, En el caso en el que el individuo tenga los movimientos que se pueden ver en la figura 9, el recorrido será el que se observa en la figura 10.

a1b2c3d4e5	01233210
cromosomas	movimientos

Figura 8: Representación de los Genes

a1b2c3d4e5	333300333003020
cromosomas	movimientos

Figura 9: Genes Ejemplo

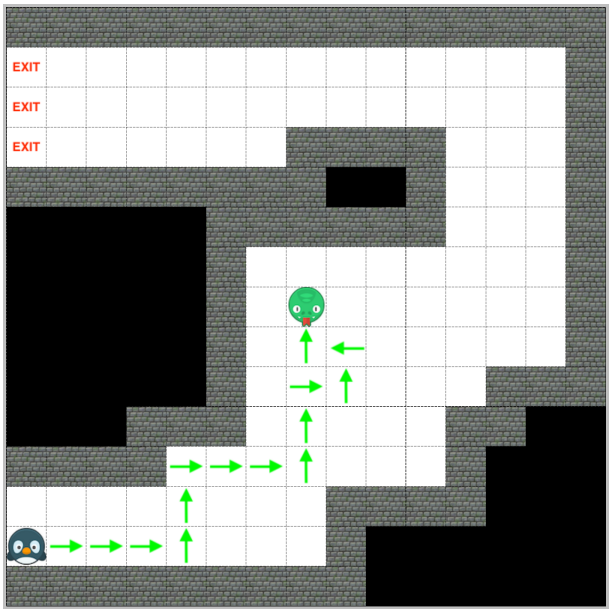

Figura 10: Movimientos Ejemplo

8.5.2. Creación de la población inicial

Es la parte más crítica [13], sobretodo la cantidad que individuos que contiene, ya que dependiendo de ese número obtendremos unos resultados u otros.

Lo mejor es crearla de manera aleatoria, cuanto más aleatoria sea la población inicial, mejores resultados obtendremos. La diversidad es importante, y el objetivo principal es que sean lo más diferente posibles, y que tiendan a moverse hacia el mejor. Por lo tanto, se comprueba que no existan duplicados dentro de la población inicial.

La población ideal depende del problema a tratar, es una solución difícil de encontrar. Se ha optado por generar poblaciones de 20 individuos.

8.5.3. Definición del movimiento de los algoritmos

Una vez definida la población inicial, con sus acciones iniciales, se determina la posición siguiente dentro del respectivo mapa según los criterios mostrados en la figura 7.

A continuación, se repite lo mencionado anteriormente, tantas veces como acciones estén almacenadas dentro de los genes del individuo. Para cada movimiento realizado, se comprueba si es correcto.

En el caso en el que se encuentre con uno de los enemigos, mencionados en la figura 3, ese individuo muere y no puede efectuar más movimientos.

En el caso en el que se encuentre con una pared también se considera una posición de muerte, ya que no queremos que ese movimiento se repita.

Si llega a la casilla de salida, se suma 1 a su afinidad y se añade esa acción a sus genes.

En el caso en el que la posición del individuo sea la casilla de salida o esté muerto, se deja de repetir este proceso para este individuo. Y se pasa al siguiente individuo. Esto se efectúa tantas veces como individuos en la población existan, en este caso, 20.

En el momento en el que todos los individuos han recorrido el mapa, se determinan los padres de la siguiente generación con los criterios mencionados [Cap.8.5.4]. Y se cruzan los padres.

8.5.4. Selección de padres

Se define un criterio simple de afinidad, donde un valor numérico determinado por el número de movimientos correctos del individuo, representa el nivel de afinidad de ese individuo.

Los 10 individuos que tengan el valor de afinidad más alto, serán seleccionados para crear descendencia. Se ha decidido ese valor a causa del número de individuos dentro de la población. En el caso en el que se use una población mayor, se consideraría un número mayor de predecesores.

En el momento en el que un individuo efectúa un movimiento correcto, se suma 1 a la variable de afinidad de ese individuo. Para saber qué individuos serán los padres de la siguiente generación, se ordenan según su afinidad de mayor a menor y se seleccionan los 10 primeros.

Para asegurar que los mejores padres son los seleccionados, se ha añadido una comprobación además de que el movimiento sea correcto [22]. Para aumentar su afinidad, es necesario que ese movimiento sea único, es decir, que no haya pasado por esa casilla anteriormente.

Para eso, se almacenan todas las posiciones únicas por las que pasa, añadiéndolas a la lista en el caso en el que la comprobación sea negativa. Es decir, que la posición actual no exista en la lista de posiciones.

En el caso en el que se consideraran las posiciones únicas como correctas y las repetidas como erróneas podría provocar bloqueos. Eso significa que en el caso en el que para poder continuar se necesite ir hacia atrás, al ya haber pasado por esa casilla, no se le permitiría pasar de nuevo, provocando un bloqueo donde no exista una solución.

Esa es la razón por la que únicamente la afinidad se ve afectada, para perjudicar a esos individuos que avancen casillas pero retrocedan otras. Permitiendo aún el retroceso en el caso en el que sea necesario.

Un ejemplo de ese problema se puede observar en la figura 11. Es necesario que el individuo retroceda para seguir avanzando, si no se le permite, no llegará a la casilla de salida.

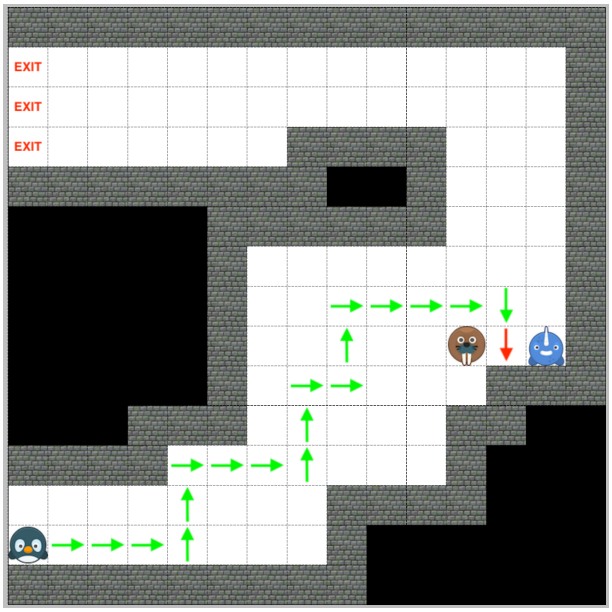

Figura 11: Movimientos hacia un bloqueo

Esta selección nos asegura escoger los mejores, y sobretodo, siempre tener individuos seleccionados. Si se utilizara un ranking o un baremo diferente para determinar qué individuos se usarán para crear la siguiente generación, existe la posibilidad de no tener ninguno.

8.5.5. Definición del cruce

Se ha determinado de la siguiente manera: En primer lugar, se separa en dos la lista de padres, para luego utilizar uno de cada sección para cruzar sus genes y obtener dos hijos por cada par de padres.

Con el par de padres seleccionados, se comparan las posiciones donde han muerto. El que tenga la posición de muerte más alta, será el que dará a los hijos sus movimientos hasta su muerte. El otro padre otorgará los movimientos después de su muerte.

Se puede observar en la figura 12 los movimientos verdes del padre, con mayor posición de muerte, que otorgará a los hijos. Y los movimientos rojos son los de después de la muerte del padre con menor posición de muerte.

8.5.6. Definición de la mutación

Una vez se han creado los hijos, los padres sufren una mutación. Se cambia la acción almacenada en la posición

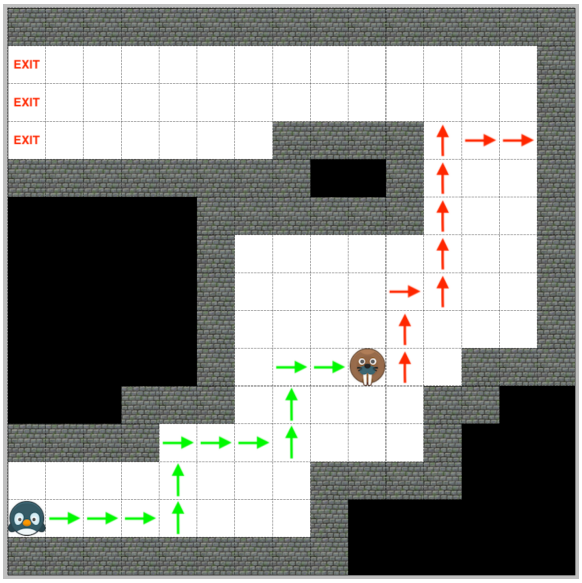

Figura 12: Movimientos de los padres

	Enemigos	Casillas	Promedio Generaciones
Mapa 1	6	38	50
Mapa 2	6	45	29
Mapa 3	6	60	67
Mapa 4	6	63	95
Mapa 5	6	97	202
Mapa 6	9	91	169
Mapa 7	10	64	182
Mapa 8	11	97	197
Mapa 9	11	106	138
Mapa 10	12	40	40

Figura 13: Resultados de la ejecución del algoritmo

de muerte, por una acción aleatoria dentro del rango mencionado [Cap. 8.5.1], comprobando que no sea la misma que lo llevó a la muerte. Para darle otra oportunidad.

Al finalizar las mutaciones, los padres y los hijos anteriores pasan de nuevo por la comprobación de afinidad para determinar si son aptos para ser padres y a su vez, comprobar si sus movimientos los llevan a la casilla de salida.

8.6. Resultados

A continuación se muestran las estadísticas obtenidas, como el número de generaciones necesarias para resolver cada mapa, cuántas casillas disponibles y enemigos contiene cada mapa.

En la figura 13 se puede observar el número de generaciones necesarias para resolver cada mapa. Este número se obtiene del promedio de 70 ejecuciones del código para cada mapa.

Estos resultados nos permiten cuantificar la dificultad de cada uno de los mapas y ordenarlos por nivel. Para poder ordenar los mapas según su dificultad se considera que cuantas más generaciones se han necesitado para resolver el mapa, más difícil es. En la figura 14 se puede observar el orden por dificultad de los mapas, del más difícil al más fácil.

Como se puede observar en la figura 14, el mapa más difícil es el número 5. Sin embargo, tiene menos enemigos y más casillas disponibles que otros mapas más fáciles. Eso se debe a la posición y movimientos de los enemigos y la distribución de las casillas. En la figura 15 se puede ver que la distribución y los movimientos de los enemigos limita el número de casillas por el que se puede pasar, incluso impidiéndolo.

En cambio, el mapa con menos dificultad, el mapa número 2, tiene más enemigos y menos casillas posibles que el

	Promedio Generaciones
Mapa 2	29
Mapa 10	40
Mapa 1	50
Mapa 3	67
Mapa 4	95
Mapa 9	138
Mapa 6	169
Mapa 7	182
Mapa 8	197
Mapa 5	202

Figura 14: Orden de los mapas según dificultad

más difícil. Eso es debido a las localizaciones de los enemigos y la existencia de un camino libre de ellos, que se puede ver en la figura 16.

9 Conclusión

El problema más experimentado en este proyecto ha sido una de las desventajas ya mencionadas [Cap 7.1.3], el hecho de requerir conocimientos previos sobre el problema por parte del desarrollador. En el momento de definir cuál de los individuos son aptos para crear descendencia, o cómo se cuantifica esa afinidad, se requiere de una evaluación previa del problema algo compleja.

Ese fue el mayor obstáculo a la hora de desarrollar, decidir qué era lo óptimo sin saber exactamente cómo afectaría al problema, creándose así un escenario de ensayo y error que retrasa la solución del problema.

En un inicio, se decidió crear de manera dinámica los

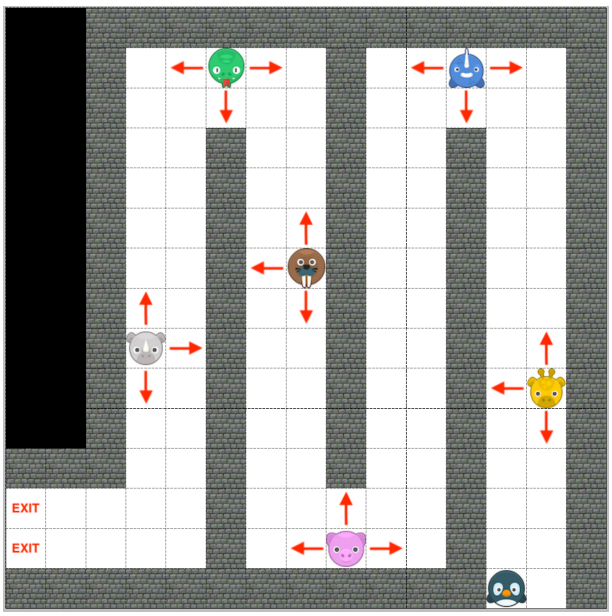

Figura 15: Mapa con más dificultad

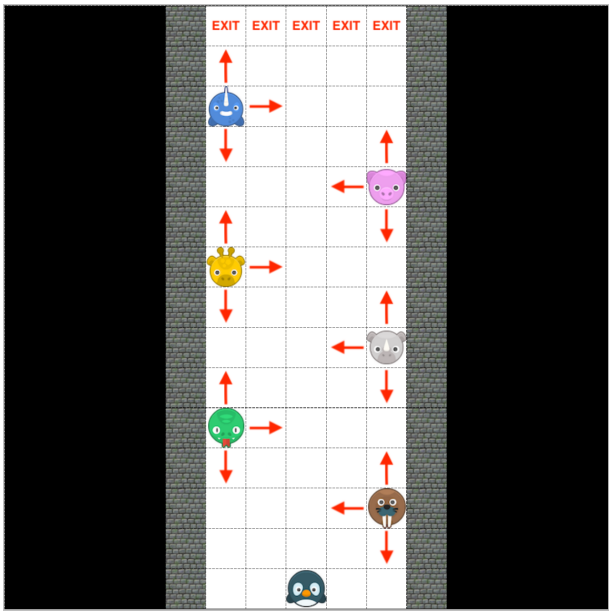

Figura 16: Mapa con menos dificultad

movimientos de cada individuo de la población inicial. Sin embargo, eso requería un número muy alto de condiciones para poder avanzar. Es decir, la comprobación de si el individuo estaba efectuando movimientos correctos, movimientos fuera del mapa, o movimientos que pudieran llevarlo a la muerte, se efectuaba a tiempo real mientras se generaban los movimientos de manera aleatoria.

En cuanto se cambió a la creación de una población inicial con todos los movimientos ya efectuados de manera aleatoria, el problema se simplificó, permitiendo comprobaciones más esporádicas en situaciones más específicas.

En el cruce de los individuos, se intentó hacer algo demasiado complejo, y utilizar valores aleatorios para determinar qué posiciones de los genes de los padres que almacenan los movimientos serían los que heredaran los hijos. Además, los movimientos al ser dinámicos en ese momento, provocaban el acceso a posiciones que aún no se habían generado. Como consecuencia, se han utilizado valores de referencia más eficaces para el cruce, en concreto: la posición de muerte del individuo.

La mutación se había considerado como un añadido que aportaba complejidad al código, pero esa interpretación era errónea. Ya que gracias a la mutación, que otorga una segunda oportunidad a los padres para finalizar el recorrido, el algoritmo ha sido capaz de encontrar soluciones válidas.

Este proyecto demuestra que el uso de los algoritmos genéticos resultan una herramienta válida para la cuantificación de la dificultad de un mapa de un videojuego.

Con respecto a la planificación, referenciada en el Apéndice [figura 17], los espacios de tiempo reservados para imprevistos han resultado esenciales para poder finalizar el proyecto a tiempo.

10 Agradecimientos

A la primera persona a la que querría agradecer es a Ivan, por su cuidado y sobretodo paciencia todas esas largas noches programando y buscando soluciones a errores que no se ven. A mi mejor amigo, Marc, por ser mi pato de goma particular y el más gracioso a la hora de buscar soluciones. A mi familia, por el apoyo y cariño. Y sobretodo, a la paciencia de mi tutora, Ana Oropesa. Por ayudarme en todo lo que he necesitado, dando manga ancha cuando no daba más de mí, y siendo la mejor guía, apoyo y correctora de este proyecto.

Referencias

[1] "¿Qué son los algoritmos genéticos?" , Laura Núñez, 6 Febrero 2019, https://bit.ly/2RKNhAj

[2] "Implementación de un algoritmo genético en JavaScript para solucionar el problema de asignación de docentes para un periodo académico en el IST Yavirac", Pablo Robayo, 4 Octubre 2021, https://bit.ly/3fz06iK

[3] "Metodologías ágiles: ¿qué diferencia hay entre Scrum, Kanban y XP?", Ana Junquera, 24 Octubre 2019, https://bit.ly/3LXJop7

[4] "The Pac-Mac Dossier", Jamey Pittman, 23 Febrero 2009, https://bit.ly/3UTRKlS

[5] "Juego de pacman en javascript", David Poza Suárez, 22 Enero 2019, https://bit.ly/3dVLCZE

[6] "Understanding Pac-Man Ghost Behaviour", Char Birch, 2 Diciembre 2010, https://bit.ly/2qBVCyf

[7] "One workspace. Every team.", Notion, https://bit.ly/2Kw2s0c

[8] "GEI: TFG - Smart decisions with AI", Elizabeth Méndez, https://bit.ly/3fOxiCX

[9] “Tiled — Flexible level editor”, Tiled, https://bit.ly/3sZ7pDR

[10] “ Algoritmos genéticos”, Fernando Sancho Caparrini, 4 Noviembre 2019, https://bit.ly/3DHobMn

[11] “SimpleGA Algoritmos genéticos en Js / NodeJS”, Angel Lopez, 27 Septiembre 2012, https://bit.ly/3fDmNCI

[12] “Implementación de Juegos usando Algoritmos Evolutivos”, Facultad de Informática Universidad Complutense de Madrid, Abril 2004, https://bit.ly/3WANmZE

[13] “Videojuego RPG (Role-Playing Game) Basado en algoritmos evolutivos”, Escuela Politécnica Superior de Jaén: José JAvier Pérez Cruz, Junio 2018,https://bit.ly/3VCJVAS

[14] “Juego”, https://bit.ly/2m8vF6Y

[15] “iocus”, 3 Mayo 2017, https://bit.ly/3NWBc9V

[16] “Inteligencia artificial: qué es, cómo funciona y para qué se utiliza en la actualidad”, Juan Antonio Pascual Estapé, 24 Agosto 2019, https://bit.ly/2U6LsQF

[17] “Inteligencia artificial”, https://bit.ly/3X2ECvU

[18] “Innovar, la teoría de la diversión de Volkswagen”, Daniel Goldman, 28 Enero 2013, https://bit.ly/3UQRIuC

[19] “Mihaly Csikszentmihalyi: biografía del padre de la teoría de flujo”, Sonia Budner, 31 Enero 2019, https://bit.ly/3gknh0N

[20] “Teoria del flow”, Sonia Castro, 26 Septiembre 2022, https://bit.ly/3Gv7O8H

[21] “Making learning fun: A taxonomy of intrinsic motivations for learning, Chapter 27”, Malone, T.W. & Lepper, M.R., 1987, https://bit.ly/3V9gxlk

[22] “Optimización de circuitos digitales mediante algoritmos genéticos”, Miriam Caravaca y Mariona Fernández, 2018-2019, Universitat Pompeu Fabra

APÉNDICE

A.1. Planificación

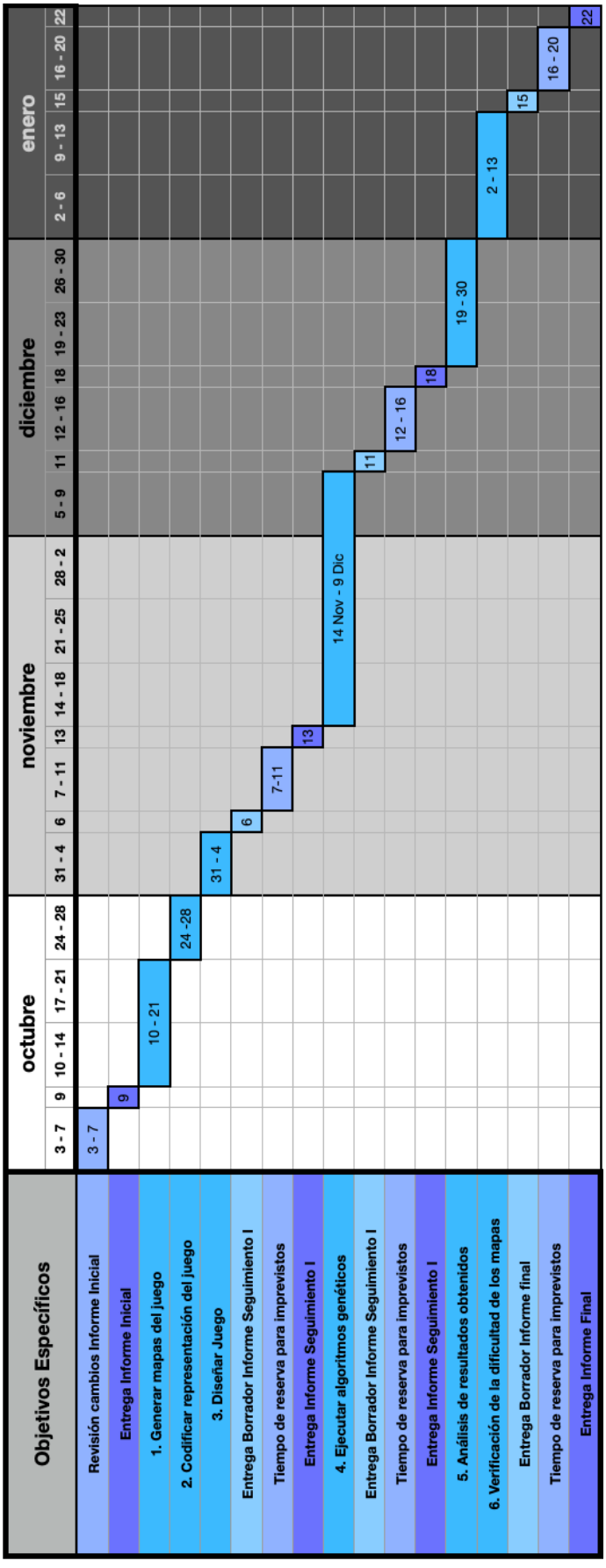

Figura 17: Planificación objetivos

A.2. Ejemplo Mapa formato JSON


```
{ "compressionlevel":-1,
 "height":15,
 "infinite":false,
 "layers":[
        {
         "data":[1 , 10, 10, 1 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9,
                 1 , 8 , 17, 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1,
                 1 , 8 , 17, 11, 6 , 12, 18, 16, 16, 16, 16, 16, 11, 5 , 1,
                 1 , 2 , 17, 16, 16, 16, 16, 11, 4 , 12, 8 , 17, 16, 14, 1,
                 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 17, 16, 1,
                 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 1 , 8 , 17, 1,
                 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 13, 17, 1,
                 1 , 5 , 12, 8 , 19, 19, 19, 19, 18, 11, 11, 7 , 12, 17, 1,
                 1 , 14, 19, 19, 17, 11, 6 , 12, 19, 19, 19, 19, 19, 17, 1,
                 1 , 19, 17, 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1,
                 1 , 17, 13, 1 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9 , 9,
                 1 , 17, 13, 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1,
                 1 , 17, 4 , 12, 18, 16, 16, 16, 16, 8 , 11, 2 , 12, 8 , 8,
                 1 , 17, 16, 16, 16, 11, 3 , 12, 17, 16, 16, 16, 16, 16, 15,
                 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1],
         "height":15,
         "id":1,
         "name":"Capa de patrones 1",
         "opacity":1,
         "type":"tilelayer",
         "visible":true,
         "width":15,
         "x":0,
         "y":0
        }],
 "nextlayerid":2,
 "nextobjectid":1,
 "orientation":"orthogonal",
 "renderorder":"right-down",
 "tiledversion":"1.9.2",
 "tileheight":50,
 "tilesets":[

 "tilewidth":50,
 "type":"map",
 "version":"1.9",
 "width":15
}
```