UNIVERSIDAD DEL BÍO-BÍO

Diseño e Implementación de un Lenguaje de Programación de Tipo Declarativo Inspirado en Prolog para Control de Agentes en Videojuegos

Christopher Cromer y Martín Araneda Acuña

12 de julio de 2022

Índice

Índice de figuras

Índice de cuadros

Composición del informe

El presente informe se encuentra dividido en cuatro capítulos. A continuación se describe brevemente el contenido de cada uno de ellos.

1. **Introducción:** Se presentan los conceptos básicos más importantes para este proyecto, como son el Desarrollo de Videojuegos y la Inteligencia Artificial. También se explican los motivos principales y específicos para el desarrollo de la investigación. Por último, se detalla la metodología de trabajo a aplicar.

2. **Marco Teórico:** Explicación de las herramientas a utilizar, análisis de rendimiento de software similares y descripción de conceptos abarcados en el proyecto.

3. **Estado del Arte:** Se describen las principales ideas de trabajos de investigación similares, contrastándolo con este trabajo. Se han dividido estos en dos categorías: basado en el lenguaje de programación Prolog y GOLOG, con descripciones acerca de metodología e implementación de agentes en videojuegos

4. **Descripción del Ambiente del software:** Se explica el software a utilizar, incluyendo la versión y el entorno de desarrollo.

Además, al final del informe se adjuntan las referencias con los artículos utilizados en el proceso de investigación.

1. Introducción

La Inteligencia Artificial (I.A) es una disciplina científica relativamente nueva que nació de los avances en las ciencias de la computación y/o la informática. De forma intuitiva, la I.A puede entenderse como la capacidad que tiene una máquina para tomar decisiones de forma autónoma a partir de los estímulos que recibe del exterior. Una de las primeras veces que se hablo de la I.A como disciplina fue en el año 1956 cuando John McCarthy, importante matemático e informático, la definió como *"La ciencia e ingeniería de hacer máquinas inteligentes, con mayor énfasis en programas computacionales inteligentes"* [1].

Actualmente, la I.A puede considerarse también una tecnología moderna que ha ido evolucionando y se ha ido incorporando a nuestra vida cotidiana. Un ejemplo bastante importante es el reconocimiento facial introducido en teléfonos inteligentes, lo que permite agregar ciertos efectos a las imágenes, ajustar automáticamente el enfoque o balancear la exposición en condiciones de poca luz.

En el campo del desarrollo de videojuegos, la I.A. ha tenido mucha presencia. Por ejemplo, se ha empleado con éxito para programar los personajes o las entidades de los mismos, haciéndolos más dinámicos y divertidos. Permitiendo que el juego aprenda del usuario y creando adversarios artificiales del nivel de un humano.

Dado lo anterior, tendrá un gran beneficio implementar una I.A. empleando un lenguaje declarativo ya que no indicamos el cómo se debe computar, solo indicamos el qué se quiere computar. Esto último es importante ya que conseguimos que la tarea de programación se aproxime a la lógica humana y el programador pueda implementar el comportamiento de los agentes de una forma más intuitiva y empleando reglas simples. En este sentido, uno de los lenguajes declarativos más importantes ha sido Prolog demostrando sus buenas propiedades en este ámbito. Por lo tanto, vamos a crear un lenguaje declarativo inspirado en Prolog que se pueda utilizar para programar el comportamiento de los personajes de un videojuego que tienen, al menos, que realizar las siguientes acciones: i) superar obstáculos, con la opción de saltar con velocidad; ii) recoger items que se agregan a la puntuación final; iii) evitar enemigos.

1.1. Objetivos

1.1.1. Objetivo General

Desarrollar un lenguaje de programación de tipo declarativo inspirado en Prolog que permita modelar el comportamiento de los agentes de un videojuego empleando reglas lógicas.

1.1.2. Objetivos Específicos

1. Revisar bibliografía sobre Prolog, el motor Godot y programación de videojuegos.
2. Analizar la información recopilada de la bibliografía investigada.
3. Crear el lenguaje de programación de tipo declarativo inspirado en Prolog.
4. Implementar un videojuego usando una inteligencia artificial basada en el lenguaje anteriormente creado.
5. Evaluar el desempeño del lenguaje inventado, verificando el cumplimiento exitoso de superación de obstáculos por parte del agente.

1.2. Metodología de trabajo

La metodología utilizada es la de investigación experimental, la cual consiste en ir implementando ciertas acciones en la que el agente debe reaccionar de acuerdo a los estímulos que están presentes en un ambiente preestablecido, que en este caso es un nivel presente en el videojuego. Con esto, se harán pruebas de causa y efecto, recompilando información en su efectivo comportamiento racional frente a obstáculos para posterior comparación con un jugador real.

Por tanto, el plan de trabajo consistirá en las siguientes fases:

- **Fase 1:** Revisión y descarte de bibliografía relacionada al desarrollo de videojuegos con implementación de inteligencia artificial inspirado en Prolog y motor Godot.

- **Fase 2:** Estudio de la información recopilada para posible implementación en el software.
- **Fase 3:** Creación del lenguaje de programación lógico de tipo declarativo inspirado en Prolog.
- **Fase 4:** Implementación de un videojuego en el motor Godot de estilo plataforma formado por agentes inteligentes programados usando el lenguaje recientemente creado.
- **Fase 5:** Evaluación del correcto desempeño del agente en la superación de obstáculos, con el cumplimiento del nivel versus un jugador real y la comparación de la puntuación y tiempo total en completar el juego entre ambos jugadores.

2. Marco Teórico

Este capítulo tiene como propósito describir los conceptos más importantes en el desarrollo del lenguaje de programación para control de la Inteligencia Artificial, así como también su implementación en un juego estilo plataformas.

2.1. Desarrollo de Videojuegos

2.1.1. Motor de videojuego

El motor de videojuego tiene una alta importancia en el desarrollo de videojuegos, ya que es responsable de proveer facilidad en utilizar gráficos, sonidos, física, redes y varios otros aspectos que ayudan en el desarrollo de un videojuego. El motor que se utilizará se llama Godot. [2, 3, 4]

Figura 1: Godot Engine

El motor Godot tiene gran valor en el desarrollo del proyecto, principalmente por su capacidad para implementar arte de píxel, en comparación con motores similares como Unity o Unreal porque en Godot las distancias entre los elementos es medido en píxeles mientras que los otros motores utilizan metros, lo que facilita en gran medida la eficacia al trabajar en juegos 2D, que usualmente para este género de videojuegos se usan píxeles y no metros.

Otro factor importante es que Godot posee un elemento llamado GDNative. GDNative es una API que permite usar lenguajes de programación como C, C++ o Rust que son lenguajes que compilan el código a lenguaje máquina. [5]

Tiene especial importancia para el proyecto ya que permite implementar otro lenguaje de programación que hará interacción con GDNative. Por otro lado también per-

mite que la Inteligencia Artificial tome decisiones con mayor rapidez dado que el código objeto de máquina se ejecuta más velozmente que un lenguaje interpretado o que se compila a bytecode como se puede notar en los siguientes gráficos que compara GDScript(lenguaje interpretado similar a Python), C# y C++.

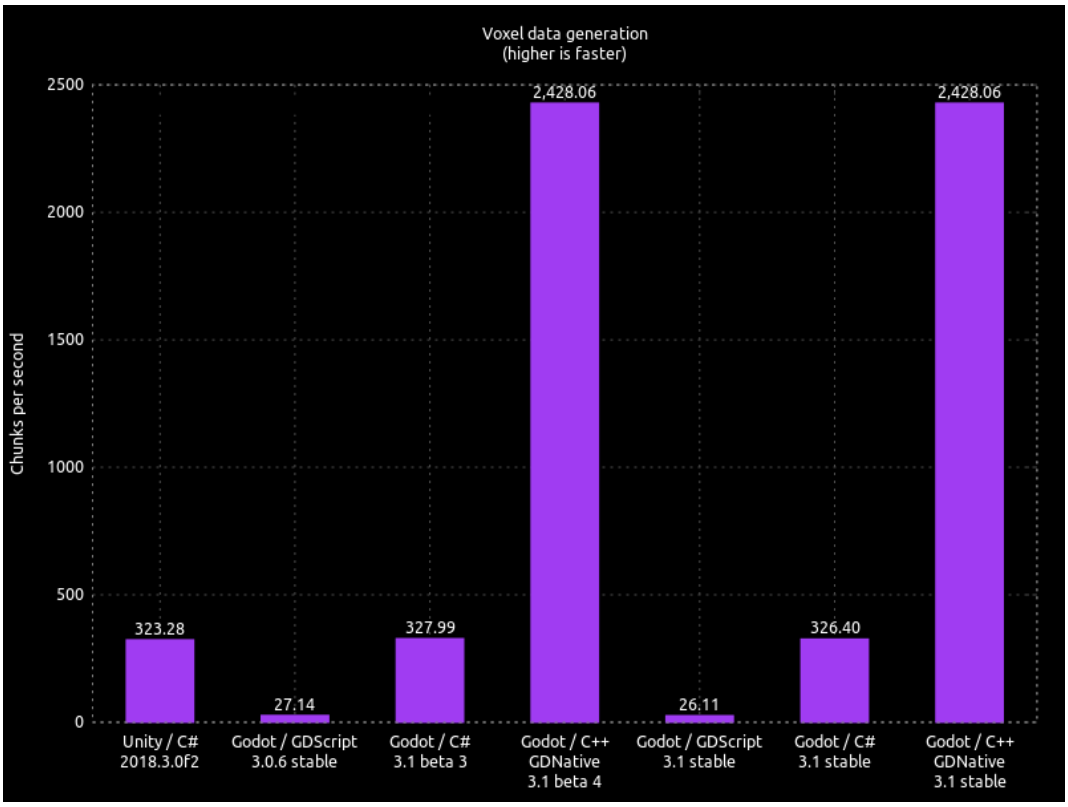

Figura 2: Voxel Data Generation
[6]

En el caso de generación de datos voxel, hay un aumento de 8,846.43 % en su rendimiento comparando GDNative con GDScript y un aumento de 651.07 % comparando GDNative con C#.

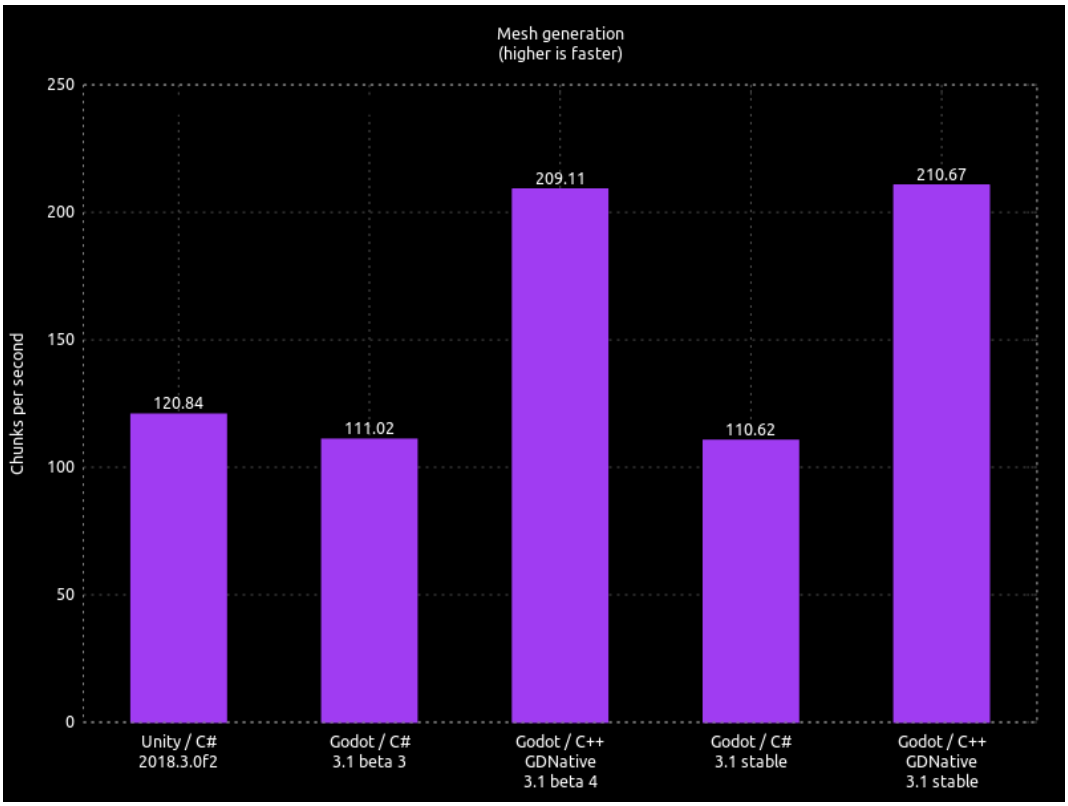

Figura 3: Mesh Generation
[6]

En la prueba de generación de mallas el GDNative tiene un aumento de 651.07 % en su rendimiento comparando con C#.

2.1.2. Game I.A.

Para la toma de decisiones de la Inteligencia Artificial, se va a utilizar un lenguaje propio inspirado en Prolog. El Prolog es un lenguaje declarativo y lógico. [7]

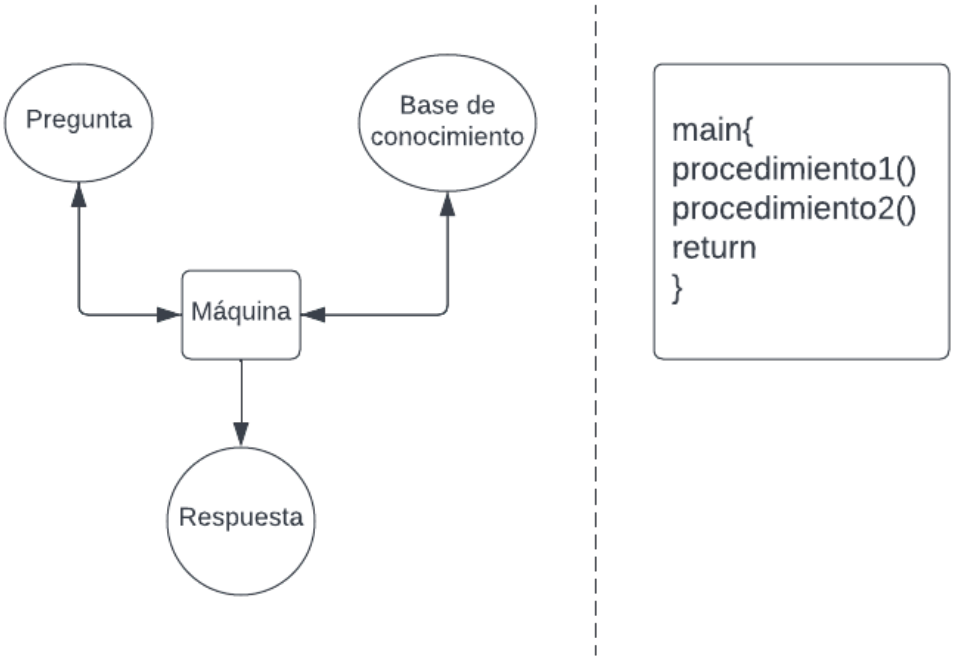

Figura 4: Programación Lógica vs. Funcional

Cuando se trata de programación funcional, la idea principal es ver cómo se puede resolver un problema. Eso se hace de forma de paso por paso para resolver el problema. En programación lógico se usa un base de conocimiento para hacer preguntas y recibir respuestas que se utilizaran para resolver el problema. [8]

El lenguaje tipo Prolog debe tener 3 elementos importantes para funcionar.

1. **Hechos:** Son datos verdaderos. Por ejemplo ”el español es una idioma”.

2. **Reglas:** Son cláusulas condicionales que conectan los hechos. Por ejemplo ”si vives en Chile hablas el español”.

3. **Preguntas:** Son necesarias para tener una respuesta de la base de conocimiento. Por ejemplo ”¿El español es una idioma?”.

2.1.2.1. Lenguaje de Programación Compilado

El proyecto LLVM es un conjunto de tecnologías de compilador y toolchain, lo cual permite crear un lenguaje propio de programación. [9]

Figura 5: LLVM

El LLVM consiste de varios sub-proyectos, pero el que será utilizado principalmente es LLVM Core. Este sub-proyecto contiene un optimizador y generador de código, siendo este último llamado LLVM Intermediate Representation(LLVM IR). La funcionalidad es similar a una Virtual Machine de bytecode que es portátil y se puede correr en cualquier sistema que posee el LLVM.

Otro aspecto importante de LLVM es que se puede utilizar el LLVM IR que fue generado para después compilarlo a lenguaje máquina para la arquitectura computacional que se desee. Luego, el código objeto generado se puede utilizar con un linker para crear librerías y binarios, lo que tendrá importancia al querer integrar el código que compila nuestra compilador en el motor Godot.

2.1.2.2. Máquinas de Estados

Una máquina de estados se utiliza para controlar el estado de un objeto que tiene una ramificación compleja y estados mutables. [10] La máquina de estados finita es parte de una rama de la ciencia de la computación llamado *"teoría de autómata"*, la cual incluye la famosa máquina de Turing. La máquina de estados es usualmente usada en programación de I.A., videojuegos y en la creación de compiladores de código, sin embargo fuera de estas 3 áreas, posee muy poca adopción y uso.

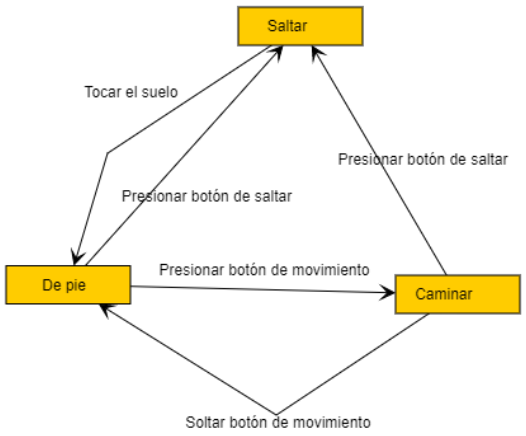

Figura 6: Máquina de Estado Finito

La máquina de estado finita posee un conjunto concreto de estados en la que es capaz de estar, como ”saltar” o ”caminar”, la cual también posee una restricción importante que consiste en que esta máquina solo puede presentar un solo estado en un instante de tiempo, es decir, el jugador no es capaz de saltar y caminar en un mismo momento.

El funcionamiento de la máquina de estados finita consiste en una secuencia de entradas y eventos que son enviadas a esta, los cuales se usan para cambiar entre estados. Cada máquina tiene un conjunto de transiciones y cada una de ellas está asociada con una entrada o un evento, lo que finalmente apunta a otro estado. Por tanto, cuando llega la entrada o evento y este coincide con una transición dentro del estado actual, la máquina cambiará al estado al que apunta la transición, por lo que si un jugador se encuentra ubicado en el estado ”caminar”, se puede presionar el botón de saltar para cambiar al estado de ”saltar”.

3. Estado del Arte

La programación del comportamiento de agentes en videojuegos se ha realizado desde diferentes puntos de vista. Los trabajos de investigación que hemos estudiado se pueden agrupar en dos categorías: (i) basados en el lenguaje de programación lógico Prolog; (ii) basados en el lenguaje de programación lógico GOLOG. A continuación se describirán cada una de las propuestas de forma más detallada.

3.1. Basado en el lenguaje de programación lógico Prolog

3.1.1. Prolog-Scripted Tactics Negotiation and Coordinated Team Actions for Counter-Strike Game Bots

Este proyecto de investigación consiste en la creación e implementación de un script basado en el lenguaje Prolog dentro de un juego estilo FPS (Disparos en primera persona), con la finalidad de crear diferentes tácticas de comportamiento en un equipo de bots compuesto por agentes. [11]

La idea fue inspirada principalmente en la observación de tácticas de equipo presentes en torneos reales de Counter-Strike. Por otro lado, la construcción de este framework está basado en un proyecto anterior hecho por los autores para crear scripts para bots.

El pilar de esta investigación se construye en la premisa de diseñar el framework para utilizar un creador de mapas y así personalizar el comportamiento del bot para mapas nuevos, pues con esto se le entregaría al agente conocimiento necesario del ambiente que le rodea para adquirir una ventaja táctica frente a oponentes humanos.

3.1.1.1. Desarrollo del razonamiento lógico del agente

Cada bot contiene dos pilas, una de acciones y otra de tareas. Estas siempre ejecutan el bloque que esta en el tope de la pila. Para que un bot sea capaz de cumplir una tarea de manera exitosa, hará ciertas acciones asociadas a esa tarea con tal de cumplirla. Algunas condiciones causarán que se agreguen o quiten acciones dentro de esta pila.

Por tanto, cuando se complete una tarea o una acción, se borrará el bloque y luego el bot intentará ejecutar la acción o tarea siguiente en la pila.

Uno de los mecanismos de razonamiento utilizado es uno llamado ”Razonamiento de reflejos”, el cual sucede en cada actualización del estado actual del juego, prestando gran apoyo cuando ocurren cambios repentinos en un nivel, como en el ambiente o cuando el bot este siendo atacado. En consecuencia, este mecanismo posee una característica poderosa para agregar nuevas tareas o limpiar la pila de tareas, adaptándose a cada situación.

3.1.1.2. Tipo de I.A.

El agente usado en este proyecto es de tipo racional basado en objetivos, con base en un árbol de decisión con instrucciones condicionales.

3.1.1.3. Planificación de camino

En los trabajos previos hechos por el autor, se implementó una base de datos que contiene puntos de navegación, con el fin de calcular y planificar el camino que debe seguir al moverse el bot. También se usan los puntos de navegación para tomar decisiones racionales de naturaleza táctica.

3.1.1.4. Agente omnisciente

El bot adquiere total conocimiento del mapa, lo que le permite tener una ventaja táctica y así aumentar sus posibilidades para ganar la partida.

3.2. Basado en el lenguaje de programación GOLOG

3.2.1. Controlling Unreal Tournament 2004 Bots with the Logic-based Action Language GOLOG

El proyecto describe el uso de una variante del lenguaje de programación lógico llamado GOLOG para implementar agentes en un videojuego estilo FPS (Disparos en Primera Persona) y enfrentarlos a los oponentes originales controlados por computador.

El videojuego en cuestión, llamado ”Unreal Tournament 2004”, es de tipo multi-jugador, donde los jugadores humanos compiten para alcanzar objetivos. Los oponentes pueden ser tanto humanos o controlados por el computador. [12]

3.2.1.1. Control de oponentes dirigidos por la computadora

Originalmente, la toma de decisiones de los bots fue hecha usando un lenguaje orientado a objeto llamado ”UScript”, que es propio del juego y es de tipo script, con lógica basada en un *”state machine”*, constando de nueve estados controlados por el motor del juego.

Sin embargo, el manejo de los bots se realizó en *”ReadyLog”*, un lenguaje basado en GOLOG, que permite el razonamiento basado en acciones.

Con esta información se agregó una interfaz al motor del juego que le permite transmitir información importante relacionada al mapa, tal como items (munición, vida y armas), ubicación de los jugadores y estilo del ambiente en general.

Por tanto, la información de estos elementos sera transmitida al bot si este es capaz de percibirlos, lo que da posibilidad a cambiar el comportamiento del bot dentro del framework.

3.2.1.2. Tipo de I.A.

Este proyecto de investigación implementa un agente inteligente racional basado en objetivos, usando técnicas de planificación.

3.2.1.3. Planificación de camino y Detección de Colisión

La interfaz entre el juego, junto con el motor y los bots no permite modificar los algoritmos de la planificación de los caminos ni de la detección de colisiones para permitir el cumplimiento de los objetivos.

3.2.1.4. Agente omnisciente

El agente solo recibe información si está en su campo de visión, por tanto, no es omnisciente y no tiene información suficiente que le permita obtener ventaja contra otros agentes o jugadores.

3.3. Comparación de Trabajos

	Obelisk	**Prolog**	**ReadyLog**
Basado en	Prolog	Prolog	Golog
Tipo de Lenguaje	Compilado	Scripted	Scripted
Juego Propio	Sí	No	No
Tipo de Juego	Platformer	FPS	FPS
Tipo de I.A.	Racional	Racional	Racional
Uso de *BOT*	No	Sí	Sí
Planificación de Camino	Posible	Sí	Imposible
Detección de Colisión	Posible	Desconocido	Imposible
Agente Omnisciente	No	Sí	No

Cuadro 1: Comparación de Trabajos

En el cuadro anterior, se realiza un resumen general de las principales características de proyectos de investigación similares al nuestro, contrastando las diferencias de cada uno de ellos.

Cabe destacar que una de las mayores diferencias presentes en relación a los otros trabajos es el tipo de lenguaje y si es un juego propio del autor.

Lo primero es porque el uso de lenguaje compilado trae consigo una enorme ventaja en aspectos como por ejemplo la velocidad de toma de decisiones complejas en distintos ambientes en que el agente pueda encontrarse.

El desarrollo de un juego propio permite una mejor integración entre la inteligencia artificial y el juego en cuestión, porque nos da mejor control en todos los aspectos que permiten el control del agente.

4. Descripción del ambiente del software

Para la realización del proyecto se necesita los software siguientes.

4.1. Lenguaje de programación

Para compilar el lenguaje de programación se necesita:

- GCC 5 o clang 5
- LLVM 14

4.2. Videojuego

Para compilar el videojuego se necesita:

- GCC 5 o clang 5
- Godot 3.5

4.3. Entorno

El entorno usado para el desarrollo es:

- Linux Mint 21 o Ubuntu 22.04
- Visual Studio Code 1.68

Referencias

[1] John McCarthy. «What is artificial intelligence?» En: (2007).

[2] David Vallejo y Cleto Martín. *Desarrollo de Videojuegos: Un Enfoque Práctico*. CreateSpace, 2015. ISBN: 978-1-51730-955-8.

[3] Ernest Adams y Joris Dormans. *Game Mechanics: Advanced Game Design*. New Riders, 2012. ISBN: 0-32182-027-4.

[4] Chris Bradfield. *Godot Engine Game Development Projects*. Packt, 2018. ISBN: 978-1-78883-150-5.

[5] Ariel Manzur y George Marques. *Sams Teach Yourself, Godot Engine Game Development in 24 Hours*. Pearson, 2018. ISBN: 0-13483-509-3.

[6] Royal Donut Games. *CPU Voxel Benchmarks of Most Popular Languages in Godot*. Abr. de 2022. URL: http://www.royaldonut.games/2019/03/29/cpu-voxel-benchmarks-of-most-popular-languages-in-godot/.

[7] Max Bramer. *Logic Programming with Prolog*. Springer, 2013. ISBN: 978-1-44715-486-0. DOI: 10.1007/978-1-4471-5487-7.

[8] tutorialspoint. *Prolog Tutorial*. Abr. de 2022. URL: https://www.tutorialspoint.com/prolog/.

[9] Mayur Pandey y Suyog Sarda. *LLVM Cookbook*. Packt, 2015. ISBN: 978-1-78528-598-1.

[10] Robert Nystrom. *Game Programming Patterns*. Genever Benning, 2014. ISBN: 0-99058-290-6.

[11] Grzegorz Jaśkiewicz. «Prolog-Scripted Tactics Negotiation and Coordinated Team Actions for Counter-Strike Game Bots». En: *IEEE Transactions on Computational Intelligence and AI in Games* 8.1 (2016), págs. 82-88. DOI: 10.1109/TCIAIG.2014.2331972.

[12] «Controlling Unreal Tournament 2004 Bots with the logic-based action language Golog / Jacobs, Stefan ; Ferrein, Alexander ; Lakemeyer, Gerhard». En: *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE).* (2011), págs. 151-152.

Prolog

[7] Max Bramer. *Logic Programming with Prolog*. Springer, 2013. ISBN: 978-1-44715-486-0. DOI: 10.1007/978-1-4471-5487-7.

[8] tutorialspoint. *Prolog Tutorial*. Abr. de 2022. URL: https://www.tutorialspoint.com/prolog/.

[11] Grzegorz Jaśkiewicz. «Prolog-Scripted Tactics Negotiation and Coordinated Team Actions for Counter-Strike Game Bots». En: *IEEE Transactions on Computational Intelligence and AI in Games* 8.1 (2016), págs. 82-88. DOI: 10.1109/TCIAIG.2014.2331972.

[12] «Controlling Unreal Tournament 2004 Bots with the logic-based action language Golog / Jacobs, Stefan ; Ferrein, Alexander ; Lakemeyer, Gerhard». En: *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE).* (2011), págs. 151-152.

Inteligencia Artificial

[1] John McCarthy. «What is artificial intelligence?» En: (2007).

[11] Grzegorz Jaśkiewicz. «Prolog-Scripted Tactics Negotiation and Coordinated Team Actions for Counter-Strike Game Bots». En: *IEEE Transactions on Computational Intelligence and AI in Games* 8.1 (2016), págs. 82-88. DOI: 10.1109/TCIAIG.2014.2331972.

[12] «Controlling Unreal Tournament 2004 Bots with the logic-based action language Golog / Jacobs, Stefan ; Ferrein, Alexander ; Lakemeyer, Gerhard». En: *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE).* (2011), págs. 151-152.

LLVM

[9] Mayur Pandey y Suyog Sarda. *LLVM Cookbook*. Packt, 2015. ISBN: 978-1-78528-598-1.

Godot

[4] Chris Bradfield. *Godot Engine Game Development Projects*. Packt, 2018. ISBN: 978-1-78883-150-5.

[5] Ariel Manzur y George Marques. *Sams Teach Yourself, Godot Engine Game Development in 24 Hours*. Pearson, 2018. ISBN: 0-13483-509-3.

[6] Royal Donut Games. *CPU Voxel Benchmarks of Most Popular Languages in Godot*. Abr. de 2022. URL: http://www.royaldonut.games/2019/03/29/cpu-voxel-benchmarks-of-most-popular-languages-in-godot/.

Videojuegos

[2] David Vallejo y Cleto Martín. *Desarrollo de Videojuegos: Un Enfoque Práctico*. CreateSpace, 2015. ISBN: 978-1-51730-955-8.

[3] Ernest Adams y Joris Dormans. *Game Mechanics: Advanced Game Design*. New Riders, 2012. ISBN: 0-32182-027-4.

[4] Chris Bradfield. *Godot Engine Game Development Projects*. Packt, 2018. ISBN: 978-1-78883-150-5.

[5] Ariel Manzur y George Marques. *Sams Teach Yourself, Godot Engine Game Development in 24 Hours*. Pearson, 2018. ISBN: 0-13483-509-3.

[6] Royal Donut Games. *CPU Voxel Benchmarks of Most Popular Languages in Godot*. Abr. de 2022. URL: http://www.royaldonut.games/2019/03/29/cpu-voxel-benchmarks-of-most-popular-languages-in-godot/.

[10] Robert Nystrom. *Game Programming Patterns*. Genever Benning, 2014. ISBN: 0-99058-290-6.

[11] Grzegorz Jaśkiewicz. «Prolog-Scripted Tactics Negotiation and Coordinated Team Actions for Counter-Strike Game Bots». En: *IEEE Transactions on Computational Intelligence and AI in Games* 8.1 (2016), págs. 82-88. DOI: 10.1109/TCIAIG.2014.2331972.

[12] «Controlling Unreal Tournament 2004 Bots with the logic-based action language Golog / Jacobs, Stefan ; Ferrein, Alexander ; Lakemeyer, Gerhard». En: *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE).* (2011), págs. 151-152.