Desarrollo de estrategias de Inteligencia Artificial para el entorno dinámico y colaborativo Geometry Friends

Development of Artificial Intelligence techniques for the dynamic and collaborative environment Geometry Friends

Trabajo de Fin de Grado
Curso 2022–2023

Autores

Alberto Almagro Sánchez
Juan Carlos Llamas Núñez

Directores

Belén Díaz Agudo
Antonio Alejandro Sánchez Ruiz-Granados

Doble Grado en Ingeniería Informática y Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

Desarrollo de estrategias de Inteligencia Artificial para el entorno dinámico y colaborativo Geometry Friends

Development of Artificial Intelligence techniques for the dynamic and collaborative environment Geometry Friends

Trabajo de Fin de Grado en Ingeniería Informática

Autores
Alberto Almagro Sánchez
Juan Carlos Llamas Núñez

Directores
Belén Díaz Agudo
Antonio Alejandro Sánchez Ruiz-Granados

Convocatoria: Junio 2023

Doble Grado en Ingeniería Informática y Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

28 de mayo de 2023

Dedicatoria

A nuestros padres, por su incondicional apoyo durante todos estos años.

Agradecimientos

A nuestros tutores, por los consejos y orientaciones que nos han dado durante toda la realización del trabajo. A las personas que han dedicado su tiempo a aprender a jugar y permitirnos comparar nuestros resultados con los suyos. Y a los profesores que tanto nos han enseñado durante toda la carrera, gracias a los cuales pronto podremos decir con orgullo que somos matemáticos e informáticos.

Resumen

Geometry Friends es un complejo juego de plataformas en el que dos figuras geométricas deben alcanzar diamantes prestando atención a los diferentes obstáculos de cada nivel. Cada año, participantes de distintas universidades desarrollan nuevos agentes inteligentes que son capaces de resolver correctamente una gran variedad de niveles, y estos participan en competiciones organizadas por los creadores del juego y que se celebran en importantes conferencias internacionales de Inteligencia Artificial. Existen tanto modos de juego individuales, en los que solo participa una de las figuras geométricas, como cooperativos.

Nuestro objetivo en este trabajo es el desarrollo de agentes que funcionen tanto de manera individual como cooperativa y que sean capaces de competir con los mejores agentes de competiciones pasadas, para poder participar en la competición de este año, que se celebrará en agosto. Para ello, exploramos qué técnicas de Inteligencia Artificial han sido más fructíferas en las competiciones pasadas y buscamos la forma de mejorar los resultados de estos agentes mediante diferentes estrategias. Además, desarrollamos un sistema que permite explicar las acciones que toman nuestros agentes. Finalmente, comprobamos mediante diferentes pruebas que nuestros agentes logran superar al jugador humano medio y a todos los agentes anteriores en todas las modalidades de juego.

Palabras clave

Geometry Friends, Inteligencia Artificial, aprendizaje por refuerzo, modelización física, IA explicable (XAI), sistema experto, entorno físico, cooperación, entorno multiagente.

Abstract

Geometry Friends is a complex platform game in which a pair of geometric figures must reach diamonds while paying attention to the different obstacles on each level. Every year, participants from different universities develop new intelligent agents that are capable of correctly solving a wide variety of levels, and they participate in competitions organized by the game's creators and held at major international Artificial Intelligence conferences. There are both individual game modes, in which only one of the figures participates, as well as cooperative ones.

Our goal in this work is the development of agents that work both individually and cooperatively and that are capable of competing with the best agents from past competitions in order to participate in this year's competition, which will be held in August. To do this, we explore which Artificial Intelligence techniques have had the most success in previous competitions and we look for ways to improve the results of these agents through different strategies. Besides, we developed a system that allows us to explain the actions that our agents take. Finally, we check through several tests that our agents manage to outperform the average human player and all previous agents in all game modes.

Keywords

Geometry Friends, Artificial Intelligence, Reinforcement Learning, physical modeling, explainable AI (XAI), expert system, physical environment, cooperation, multi-agent environment,

Índice

Capítulo 1

Introducción

1.1. Motivación

En los últimos años, se han producido significativos avances en todas las ramas de la Inteligencia Artificial. En el campo de la teoría de juegos, se ha conseguido desarrollar agentes que alcanzan un nivel sobrehumano, con lo que se han podido descubrir nuevas técnicas aplicables a otras áreas. Ya en los años 90, se desarrollaba un motor de análisis de ajedrez capaz de derrotar a los mejores jugadores del mundo [10], y, desde entonces, en pocos juegos los humanos somos los que dominamos. Ejemplos notables incluyen juegos complejos como el go [37], juegos con información incompleta como el póquer [8] y juegos en tiempo real como el Starcarft [43] o el Dota 2 [5].

En este trabajo, tratamos Geometry Friends, un juego de plataformas bidimensional en el que participan un rectángulo verde y un círculo amarillo, cuyo objetivo es alcanzar una serie de diamantes (rombos morados) en el menor tiempo posible. En este sentido, es diferente de los juegos anteriores en tanto en cuanto no hay un adversario al que se deba superar. Existen niveles en los que participa solo un agente, pero también los hay en los que los dos personajes deben cooperar para alcanzar algunos diamantes que no pueden coger por separado, como el de la figura 1.1, en la que se muestra la apariencia del juego.

Además de plataformas con diferentes propiedades y posiciones desafiantes de los diamantes, que obligan a los agentes a resolver retos como decidir el orden correcto en el que alcanzarlos, el juego presenta muchas otras dificultades. En primer lugar, se trata de un entorno dinámico, en el que la física juega un papel significativo y la precisión de los movimientos es fundamental para lograr buenos resultados. Los movimientos son limitados (por ejemplo, el círculo salta siempre con el mismo impulso), pero muchas veces las trayectorias resultantes se antojan complejas. Por

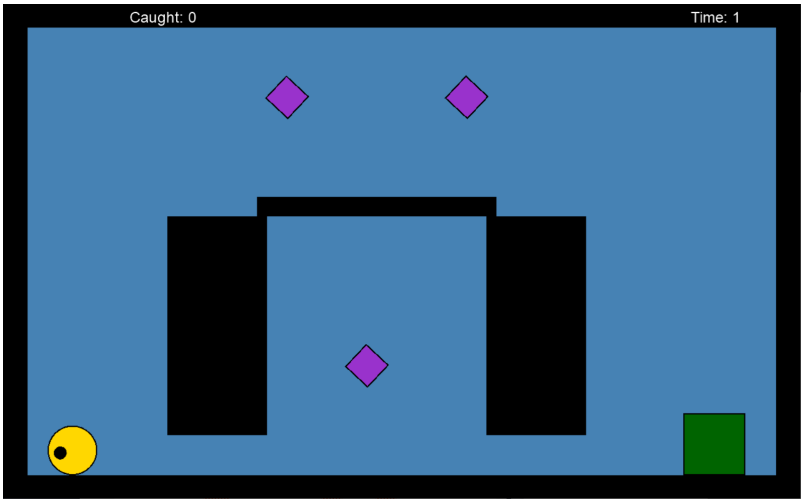


Figura 1.1: Ejemplo de un nivel cooperativo del juego.

otro lado, cada agente posee un conjunto de acciones propio, lo que provoca que sus comportamientos sean muy diferentes. Finalmente, la cooperación de agentes con roles distintos supone un reto de tal magnitud que los mejores agentes desarrollados hasta el momento aún distan del nivel de un jugador experto.

Estas características hacen que Geometry Friends sea un juego muy interesante desde el punto de vista de la Inteligencia Artificial. Tal es así, que en años recientes las competiciones que los creadores celebran anualmente, en las que los participantes presentan los mejores agentes que han podido desarrollar, han estado vinculadas a importantes conferencias internacionales, como son la *Conference of Games* (CoG) y la *International Joint Conference on Artificial Intelligence* (IJCAI). Estas competiciones tienen tres categorías: CircleTrack, RectangleTrack y CooperationTrack, y pretendemos superar los mejores resultados obtenidos hasta ahora en cada una de ellas. Las competiciones se celebrarán una vez más en agosto de este año, y volverán a estar asociadas a las conferencias CoG[1] e IJCAI[2].

1.2. Objetivos

En esta sección vamos a hablar sobre los objetivos que queremos lograr durante la realización de este trabajo. Nuestro principal objetivo es desarrollar agentes para participar en las competiciones individuales y cooperativas del juego Geometry Friends, cuyo rendimiento alcance el de los mejores agentes del estado del arte. Para ello, abordaremos los siguientes subobjetivos:

$\mathcal{O}1$ Estudiar las técnicas de Inteligencia Artificial empleadas en el desarrollo de los

[1] `https://2023.ieee-cog.org/competitions/`
[2] `https://ijcai-23.org/competitions/`

agentes de las competiciones de años anteriores, examinando las implementaciones concretas de los que hayan obtenido mejores resultados.

$\mathcal{O}2$ Identificar patrones en el juego que permitan abstraer y caracterizar el entorno, adquiriendo conocimiento experto sobre sus mecánicas.

$\mathcal{O}3$ Incorporar conocimiento experto para implementar agentes individuales que resuelvan la mayoría de los niveles.

$\mathcal{O}4$ Adaptar los agentes individuales para que sean capaces de colaborar y resolver niveles cooperativos.

$\mathcal{O}5$ Analizar cómo se puede utilizar el aprendizaje por refuerzo en el contexto del juego, comparándolo con otras aproximaciones.

$\mathcal{O}6$ Desarrollar un sistema de Inteligencia Artificial Explicable (XAI) que permita explicar las decisiones de los agentes en la interfaz gráfica del juego.

$\mathcal{O}7$ Adquirir los conocimientos necesarios sobre el lenguaje C#, en el que está escrito el juego y en el que se deben desarrollar los agentes.

$\mathcal{O}8$ Evaluar los agentes desarrollados comparándolos con los mejores agentes de competiciones anteriores y con jugadores humanos.

1.3. Plan de trabajo

Con el fin de completar los objetivos anteriores, describimos el plan de trabajo que hemos seguido durante el curso.

En primer lugar, estudiamos el estado del arte para este problema, explorando los agentes desarrollados para el círculo y el rectángulo, tanto individuales como cooperativos. Examinamos las diferentes propuestas, analizando los resultados en las competiciones de años anteriores. Dedicamos los primeros meses del curso a esto, hasta el comienzo del mes de diciembre.

A continuación, desarrollamos los dos agentes correspondientes a las competiciones individuales. Dado que tienen roles diferentes, el desarrollo de ambos requirió esfuerzo independiente, a pesar de que hubiese algunas características que logramos reusar. Para la implementación del primer agente, dedicamos los meses de diciembre y enero, y, para la del segundo agente, los meses de febrero y marzo.

Una vez implementados los agentes individuales, los adaptamos para la competición cooperativa, para lo que empleamos los meses de marzo y abril.

Con respecto a la organización general, hemos utilizado plataformas adecuadas para el trabajo colaborativo, como GitHub (para gestionar el código), Overleaf (para escribir este documento) y Google Drive (para almacenar los documentos que utilizamos como referencias).

Finalmente, hemos escrito esta memoria a medida que progresábamos en el proyecto. También hemos mantenido reuniones cada dos semanas con los tutores, de forma que pudiéramos comentar nuestros avances y los próximos pasos a seguir.

1.4. Estructura del documento

En el capítulo 2, proporcionamos una introducción más detallada al entorno con el que tratamos, explicando cómo se llevan a cabo las competiciones asociadas. También describimos los agentes que han participado en competiciones anteriores (es decir, el estado del arte), dando un resumen final que recapitula las estrategias seguidas.

Los siguientes capítulos están dedicados a la explicación detallada de las implementaciones que hemos realizado para los agentes. En el capítulo 3, describimos la parte que es común a ambos agentes, y se presentan las herramientas de las que hemos dispuesto para llevar a cabo su desarrollo.

En los capítulos 4 y 5, explicamos los detalles y particularidades que hemos seguido durante el desarrollo de los agentes individuales, siendo el capítulo 4 el correspondiente al círculo y el capítulo 5 el del rectángulo. En ellos, cubrimos todas las áreas de la implementación que consideramos relevantes.

En el capítulo 6, explicamos las técnicas cooperativas que hemos desarrollado con el fin de que nuestros agentes colaboren en los niveles correspondientes. Para ello, introducimos las adaptaciones y modificaciones que tuvimos que implementar a las versiones individuales de los agentes. Además, dedicamos una sección al sistema de explicabilidad visual que hemos desarrollado para los agentes cooperativos, que permite entender su comportamiento de una manera más sencilla.

Tras estos capítulos, que constituyen el cuerpo principal de la memoria, tenemos el capítulo 7. Este es el capítulo de resultados, elaborado a partir de una selección de niveles correspondientes a competiciones de años anteriores, en los que comparamos a jugadores humanos y a los mejores agentes del estado del arte (tanto individuales como cooperativos) con los que hemos desarrollado nosotros.

Por último, se encuentra el capítulo 8, en el que resumiremos el trabajo realizado, extrayendo conclusiones a partir de los resultados que hemos obtenido. En ese mismo capítulo, detallamos múltiples vías de mejora de los agentes implementados, y

comentamos otras maneras en las que este trabajo puede ser útil para futuros desarrollos. Además, se explican los planes de cara a la competición que se celebrará en agosto de este mismo año.

Durante el documento, hay enlaces a breves animaciones o vídeos que permiten explicar conceptos de una manera más clara y visual, y se identifican por estar subrayados y en color azul, como en el siguiente ejemplo (este no es un enlace, solo un indicador de cómo es el formato que se utiliza). Si no se pudiera acceder a dichos enlaces, el conjunto de todas las animaciones y vídeos es accesible mediante la siguiente url, correspondiente a una lista de reproducción de YouTube: `https://bit.ly/3qay7Lr`. Esta lista contiene de manera ordenada los vídeos a los que se hace referencia durante el documento. Avisamos a que, debido a problemas de compatibilidad, la calidad de la mayoría de ellos no es tan alta como querríamos.

1.5. Código asociado al trabajo

Todo el código desarrollado y empleado en el proyecto se puede encontrar en un repositorio de Github de uno de los autores, que será accesible para la evaluación a través de `https://github.com/alberalm/TFG-Geometry-Friends`. Sin embargo, el repositorio no será público hasta que se celebre la competición en la que tenemos intención de participar, con el fin de evitar pérdida y/o robo de código. Una vez se celebre la competición, el código que hayamos subido será accesible a través de la página web, por lo que abriremos el repositorio entonces con una licencia MIT. Aún así, se dará acceso anticipado a los miembros correspondientes del tribunal evaluador del trabajo.

Advertimos de que, aunque a simple vista parece que la amplia mayoría de los *commits* han sido realizados por solamente uno de los miembros del grupo, esto se debe a que la mayoría del desarrollo se ha llevado a cabo mientras se utilizaba la herramienta Live Share de Visual Studio[3], que permite la programación simultánea del código por parte de varios contribuidores. Por tanto, dado que siempre era uno de los autores el que hacía de anfitrión para la sesión, el código se guardaba localmente en su ordenador, convirtiéndolo aparentemente en el principal contribuidor del repositorio.

[3]Véase `https://visualstudio.microsoft.com/services/live-share/`.

Chapter 1

Introduction

1.1. Motivation

In recent years, there have been significant advances in all Artificial Intelligence branches. In the field of game theory, many agents that reach a superhuman level have been implemented, and the techniques in their development have been applied to other areas. Already in the 90s, a chess analysis engine capable of defeating the best players in the world [10] was being developed, and since then, there are only a few games in which humans still dominate. Notable examples include complex games like go [37], games with incomplete information like poker [8], and real-time games like Starcarft [43] or Dota 2 [5].

In this work, we deal with Geometry Friends, a two-dimensional platform game in which a green rectangle and a yellow circle participate, and whose objective is to reach a series of diamonds (purple rhombus) in the shortest possible time. In this sense, it is different from the games mentioned above as there is no opponent to beat. There are levels in which only one agent participates, but there are also those in which the two characters must cooperate to reach some diamonds that they cannot take separately, like the ones in Figure 1.1, in which we show how the game looks like.

In addition to platforms with different properties and tricky diamond positions, which force agents to solve challenges such as deciding the correct order in which to reach them, the game presents many other difficulties. First of all, it is a dynamic environment, in which physics play a significant role and precision in movements is essential to achieve good results. The movements are limited (for example, the circle always jumps with the same impulse), but many times the resulting trajectories are complex. On the other hand, each agent has its own set of actions, which results in their behaviors being very different. Finally, the cooperation of agents with different

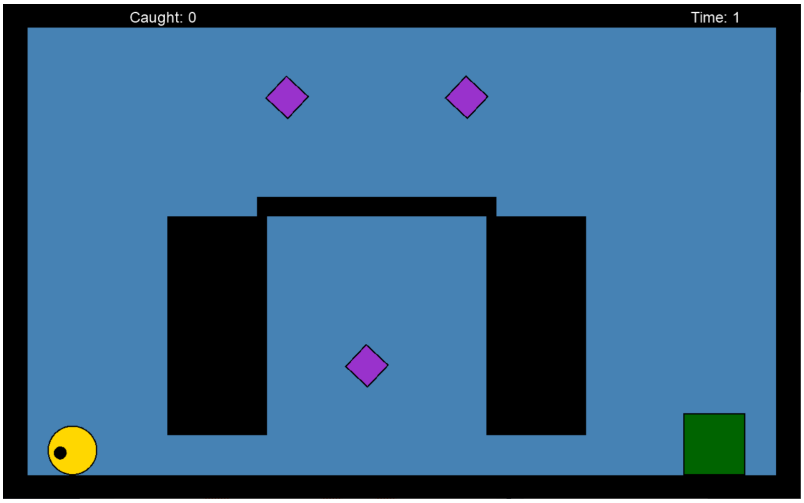


Figure 1.1: Example of a cooperative level of the game.

roles is such a challenge that the best agents developed to date are still far from the level of an expert player.

These features make Geometry Friends a very interesting game from the point of view of Artificial Intelligence. So much so, that in recent years the competitions that creators hold annually, in which participants present the best agents they have been able to develop, have been linked to important international conferences, such as the *Conference of Games* (CoG) and the *International Joint Conference on Artificial Intelligence* (IJCAI). These competitions have three categories: CircleTrack, RectangleTrack and CooperationTrack, and we intend to beat the best results obtained so far in each of them. The competitions will be held once more in August this year, being associated one more time with the CoG[1] and IJCAI[2] conferences.

1.2. Objectives

In this section, we are going to talk about the objectives that we want to achieve during the completion of this work. Our main goal is to develop agents to participate in the individual and cooperative competitions of the Geometry Friends game, whose performance reaches that of the best state-of-the-art agents. To do this, we will address the following objectives:

$\mathcal{O}1$ To study the Artificial Intelligence techniques used in the development of the agents of previous competitions, examining the specific implementations of those that have obtained the best results.

[1] `https://2023.ieee-cog.org/competitions/`
[2] `https://ijcai-23.org/competitions/`

$\mathcal{O}2$ To identify patterns in the game that allow abstracting and characterizing the environment, while acquiring expert knowledge about its mechanics.

$\mathcal{O}3$ To incorporate expert knowledge to implement individual agents that solve most levels.

$\mathcal{O}4$ To adjust individual agents to be able to collaborate and solve cooperative levels.

$\mathcal{O}5$ To analyze how reinforcement learning can be used in the context of the game, comparing it with other approaches.

$\mathcal{O}6$ To develop an XAI system that allows the agents' decisions to be explained in the game's graphical interface.

$\mathcal{O}7$ To acquire the necessary knowledge about the C# language, in which the game is coded, and in which the agents must be developed.

$\mathcal{O}8$ To evaluate the developed agents by comparing them with the best agents from previous competitions and with human players.

1.3. Work plan

In order to complete the previous objectives, we describe the work plan that we have followed during the course.

First, we studied the state of the art for this problem, exploring the agents developed for the circle and the rectangle, both individual and cooperative. We examined the different proposals, analyzing the results in the competitions of previous years. We dedicated the first months of the course to this, until the beginning of December.

Next, we developed the two agents corresponding to the individual competitions. Since they have different roles, the development of both required independent effort, even though there were some features we were able to reuse. For the circle agent's implementation, we dedicated the months of December and January, and, for the rectangle agent, the months of February and March.

Once the individual agents had been implemented, we adapted them for cooperative competition, for which we used the months of March and April.

Regarding the general organization, we have used suitable platforms for collaborative work, such as GitHub (to manage the code), Overleaf (to write this document) and Google Drive (to store the documents we use as references).

Finally, we have written this memory as we progressed in the project. We have also held meetings every two weeks with the tutors, so that we could discuss our progress and the next steps to follow.

1.4. Document structure

In Chapter 2, we provide a more detailed introduction to the environment we deal with, explaining how the associated competitions are run. We also describe the agents that have participated in previous competitions (that is, the state-of-the-art), giving a final summary that recapitulates the strategies followed.

The following chapters are dedicated to the detailed explanation of the implementations we have made for the agents. In Chapter 3, we describe the part that is common to both agents, and we present the tools we had at our disposal to carry out their development.

In Chapters 4 and 5, we explain the details and particularities that we have followed during the development of the individual agents, being Chapter 4 the one corresponding to the circle and Chapter 5 the one to the rectangle. In these chapters, we cover what we consider relevant about their implementation.

In Chapter 6, we explain the cooperative techniques we developed that allow our agents to collaborate at the corresponding levels. To do so, we introduce the adaptations and modifications we implemented to the individual versions of the agents. In addition, we dedicate a chapter section to the visual explainability system we have developed for our cooperative agents, which makes it easier to understand their behavior.

After these chapters, which constitute the main body of the report, there is Chapter 7. This is the results chapter, made from a selection of levels corresponding to competitions from previous years, in which we compare human players and the best state-of-the-art agents (both individual and cooperative) with those that we have developed.

Finally, there is Chapter 8, in which we will summarize the work done, drawing conclusions from the results we have obtained. In that same chapter, we detail multiple ways to improve the implemented agents, and we comment on other ways in which this work can be useful for future developments. In addition, we explain our goals regaring this year's competition, which will be held in August.

Throughout the document, there are links to short animations or videos that allow concepts to be explained in a clearer and more visual way, and they are identified by being underlined and in blue, as in the following example (this is not a link, just

an indicator of how links are highlighted). If these links could not be accessed, the set of all animations and videos is accessible through the following url, which corresponds to a YouTube playlist: `https://bit.ly/3qay7Lr`. This playlist contains, in an ordered manner, the videos that are referred to throughout the document. Please note that, due to compatibility issues, the quality of most of them is not as high as we would like.

1.5. Associated code to this project

All the code we have developed and used in the project can be found in a Github repository of one of the authors, which will be accessible via `https://github.com/alberalm/TFG-Geometry-Friends`. However, the repository will not be public until the competition in which we intend to participate is held, in order to avoid loss and/or theft of code. Once the competition is held, the code that we will have uploaded will be accessible through the website, so we will open the repository with an MIT license. Even so, early access will be given to the corresponding members of the evaluation panel of the work.

Please note that, while at first glance it appears that the vast majority of *commits* have been made by just one of our group members, this is because most of the development has been made while using the Visual Studio Live Share tool[3], which allows simultaneous code programming by multiple contributors. Therefore, since one of us was always the session host, the code was stored locally on his computer, apparently making him the repository's main contributor.

[3]See `https://visualstudio.microsoft.com/services/live-share/`.

Capítulo 2

Descripción del entorno Geometry Friends

En este capítulo, realizamos una introducción al entorno Geometry Friends en el que se desarrolla el juego, así como las competiciones en las que se puede medir el rendimiento de los agentes. Para finalizar el capítulo, revisamos el estado del arte y estudiaremos las técnicas que utilizan los mejores agentes desarrollados hasta el momento.

2.1. Introducción a Geometry Friends

Geometry Friends es un juego de plataformas cooperativo de dos jugadores desarrollado por GAIPS INESC-ID[1]. El juego está compuesto por una serie de niveles en un mundo bidimensional y dirigido por un motor físico en el que dos agentes, un círculo y un rectángulo, deben conseguir el mayor número de diamantes en el menor tiempo posible. Los agentes participan individualmente o cooperando, según el modo de juego. Los niveles finalizan cuando se alcanzan todos los diamantes o se acaba el tiempo disponible.

Cada personaje tiene un conjunto propio de acciones de acuerdo a sus características. El círculo puede rodar hacia la derecha o la izquierda y también cuenta con la posibilidad de saltar. Por su parte, el rectángulo puede deslizarse hacia la izquierda y la derecha y puede redimensionarse. Esto quiere decir que puede cambiar su forma de manera continua y manteniendo constante su área (y su masa). El rango de formas varía desde una en la que su base es ancha y tiene poca altura a otra en la que su base es estrecha y tiene altura máxima, pasando por la forma de un cuadrado. Sin embargo, el rectángulo no puede saltar, haciendo que llegar a los puntos más elevados del nivel con este agente suela suponer un reto.

[1] `http://gaips.inesc-id.pt/`

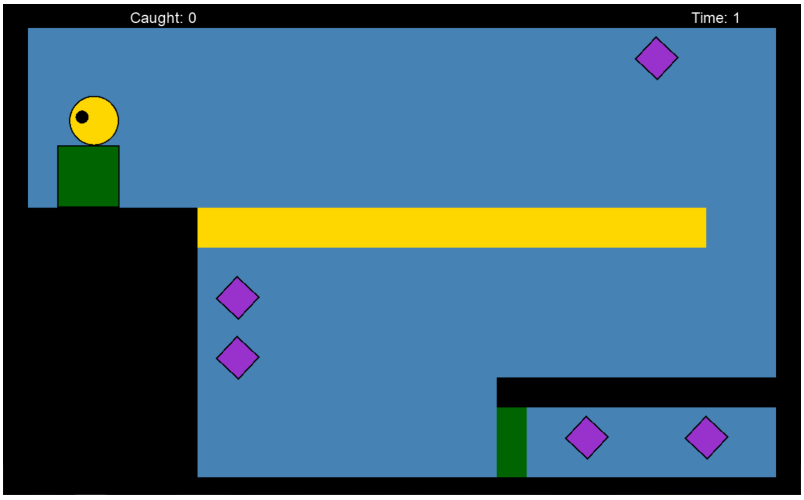


Figura 2.1: Ejemplo de un nivel del juego con diferentes tipos de plataformas.

Los niveles del juego están definidos por la disposición de los obstáculos o plataformas y los diamantes, así como la posición inicial de cada uno de los personajes. Las plataformas pueden ser de tres colores: amarillo, verde y negro. El círculo, que es de color amarillo, y el rectángulo, que es de color verde, atraviesan las plataformas que son de su mismo color, mientras que colisionan con las que son de colores distintos. Es decir, el círculo ignora las plataformas amarillas, mientras que rebota contra los obstáculos verdes y negros, y el rectángulo tiene un comportamiento análogo. Los personajes también chocan entre sí. Un ejemplo de la apariencia de estas plataformas se encuentra en la figura 2.1. Nótese que se muestra información del tiempo transcurrido en el nivel y del número de diamantes recogidos hasta el momento.

El entorno en el que se desarrolla el juego es dinámico y controlado por un simulador físico. Sobre los personajes actúan la fuerza de la gravedad y el rozamiento con las superficies. Estas fuerzas no tienen el mismo impacto sobre los dos personajes, ya que, por ejemplo, el rectángulo y el círculo no se desplazan con la misma aceleración. Además, se simulan colisiones inelásticas con los demás objetos. El estado del juego viene dado, por tanto, por las posiciones y velocidades de los agentes, la altura (o anchura) del rectángulo y los diamantes que todavía no se han recogido.

Este entorno físico genera limitaciones naturales. El círculo, por ejemplo, aunque es el único agente que puede saltar, solo puede hacerlo con una velocidad vertical inicial determinada. Esto quiere decir que no se puede regular un salto según la altura a la que se quiere saltar, sino que se deben buscar alternativas para llegar a esa altura deseada con precisión. Asimismo, no puede cambiar su velocidad horizontal durante el vuelo, lo que hace imposible corregir un salto que no sigue la trayectoria ideada. Es por tanto imprescindible una alta precisión en la toma de acciones.

Por otro lado, el rectángulo también posee sus dificultades. Por ejemplo, el agente solo puede ejecutar las acciones de crecer y decrecer si su orientación es paralela al suelo, como en la figura 2.1. En consecuencia, algunos movimientos para cambiar de

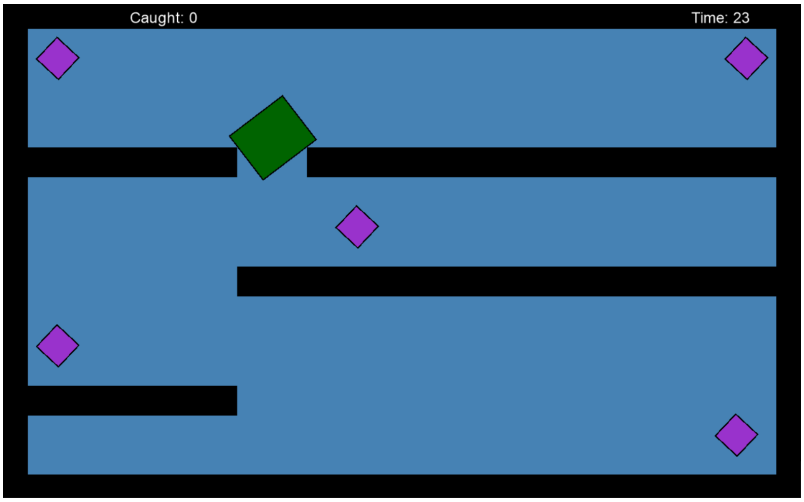


Figura 2.2: Ejemplo de una configuración en la que el rectángulo se ha quedado estancado.

plataforma solamente son posibles cuando se realizan con una forma específica. En caso contrario, el agente tiene la posibilidad de quedarse estancado, es decir, en una configuración de la que el rectángulo no puede salir, como en la figura 2.2. Pero la mayor limitación del rectángulo es su impedimento de saltar y su gran dificultad para ascender a plataformas que estén a una altura mayor que la actual. Profundizaremos más en las dificultades de cada personaje en los capítulos correspondientes.

Las distintas configuraciones de obstáculos y diamantes proporcionan una amplia variedad de retos y dificultades que los personajes deben sortear para completar el nivel. Entre estos desafíos, hay niveles en los que es imprescindible la cooperación entre los personajes y la coordinación de las acciones, mientras que en otros la estrategia óptima es la división de tareas para que cada personaje consiga los diamantes que tiene más accesibles. También es relevante el orden en el que se capturan los diamantes, ya que, en ocasiones, hay acciones que no permiten retroceder al estado anterior y lanzarse a conseguir un diamante puede suponer que el nivel se vuelva irresoluble. Por ejemplo, en el nivel mostrado en la figura 2.1, el diamante de arriba a la derecha solamente es accesible si el círculo salta desde encima del rectángulo, que además debe estar situado encima de la plataforma amarilla. Si el rectángulo decidiese coger primero los diamantes de abajo a la derecha, no habría forma de que volviera a subir a ayudar al círculo.

El ejemplo más común de colaboración surge cuando se debe alcanzar un diamante que está en una posición muy elevada del mapa, de modo que la única manera de alcanzarlo es haciendo que el rectángulo sirva de plataforma para el círculo. Sin embargo, hay otras situaciones en las que se requiere otro tipo de colaboración, como puede ser el nivel mostrado en la figura 2.3.

En este nivel, vemos cómo la única manera de resolverlo es si el rectángulo consigue acceder a la parte superior de la plataforma negra, ya que es el único que puede

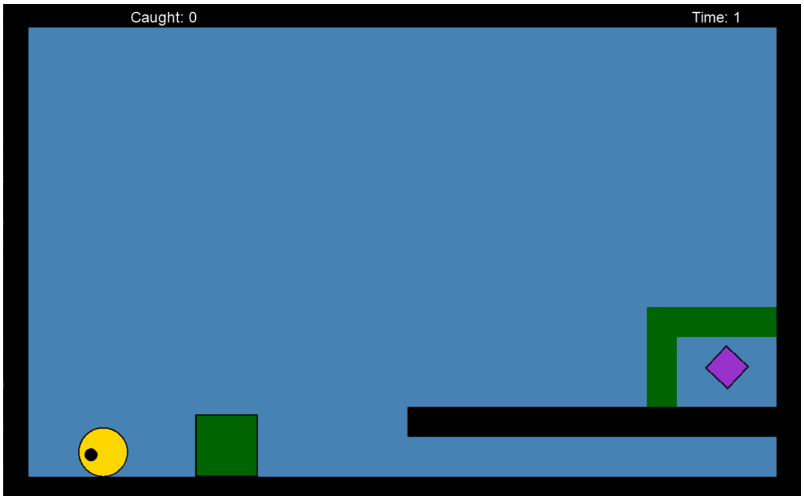


Figura 2.3: Ejemplo de un nivel que requiere otro tipo de colaboración.

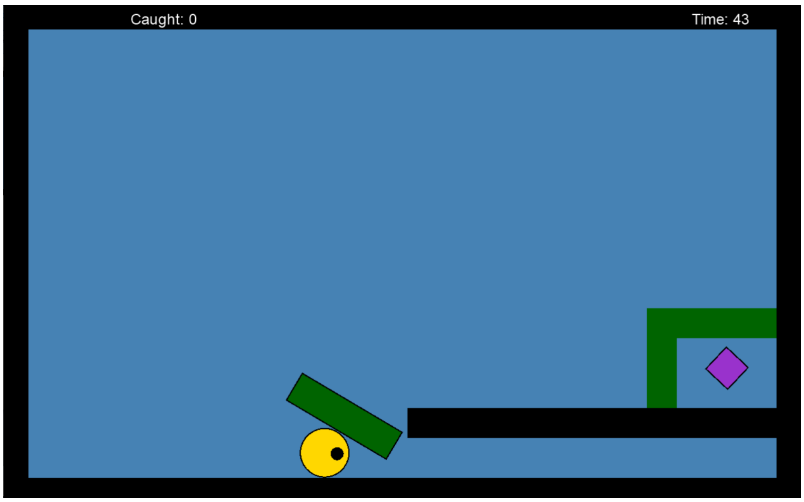


Figura 2.4: Colaboración en la que el círculo sirve como base.

atravesar la plataforma verde. Sin embargo, él solo no tiene las habilidades suficientes para hacerlo. Una solución consiste en que el rectángulo se vuelva lo más alto y estrecho posible y gire alrededor del círculo, de forma que este último salte cuando el rectángulo esté encima suya, impulsándole hacia arriba. Esta configuración se muestra en la figura 2.4, en la que, si el círculo saltara en ese momento, el rectángulo subiría a la plataforma negra. Hay más ejemplos con diferentes situaciones, lo que hace que el entorno sea mucho más complejo de lo que pueda parecer a primera vista.

2.2. Competiciones

Anualmente, se celebran competiciones del juego Geometry Friends asociadas a conferencias internacionales como IJCAI, que consisten en la implementación de agentes de IA que sean capaces de resolver algunos de los niveles. En ellas, hay tres categorías o modalidades de participación: CircleTrack, RectangleTrack y CooperationTrack. Aunque en un primer momento el juego estaba diseñado para dos jugadores, se extendió con niveles en los que solo compitiera uno de los dos personajes, o bien el círculo, o bien el rectángulo. Estos niveles permiten tratar con las complicaciones del entorno físico y la resolución de problemas de planificación, sin adentrarse inicialmente en los retos derivados de la cooperación. Habitualmente, la competición CircleTrack presenta retos más relacionados con la interacción con el entorno físico donde se requiere una gran precisión en la ejecución de las acciones. Por su parte, en el RectangleTrack, la planificación toma un mayor peso, ya que el rectángulo no tiene la capacidad de saltar y, cuando cae, generalmente no puede retroceder al estado anterior.

Dentro de cada modalidad, el criterio de evaluación del rendimiento de cada agente se determina mediante un sistema de puntos. Los agentes son evaluados en un total de 10 niveles, 5 públicos y 5 privados. Los niveles públicos están disponibles desde el comienzo de la competición para ayudar al desarrollo de los agentes, mientras que los privados se mantienen ocultos hasta que termina el plazo de participación, con el fin de evitar una sobreespecialización. Cada agente es ejecutado 10 veces en cada uno de los 10 niveles (para minimizar las diferencias entre ejecuciones debida a la aleatoriedad de algunas implementaciones). La puntuación final se obtiene sumando la puntuación de cada nivel, que a su vez resulta de realizar la media aritmética de las puntuaciones de las 10 ejecuciones en dicho nivel. La puntuación de cada ejecución se calcula con la fórmula

$$\text{SCORE} = V_{\text{completed}} \times \frac{maxTime - agentTime}{maxTime} + (V_{\text{collected}} \times N_{\text{collected}}),$$

donde $V_{\text{completed}}$ es la puntuación recibida por completar el nivel, es decir, conseguir todos los diamantes del nivel, $maxTime$ es el límite de tiempo del nivel, $agentTime$ es el tiempo invertido por el agente en completar el nivel, $V_{\text{collected}}$ es la puntuación por conseguir un diamante y $N_{\text{collected}}$ es el número de diamantes obtenidos. Los valores de $V_{\text{completed}}$, $V_{\text{collected}}$ y $maxTime$ son proporcionados por la organización en cada una de las ediciones.

Algunas características del juego hacen que pueda ser un buen entorno donde desarrollar agentes mediante técnicas de Inteligencia Artificial. Entre ellas, destaca la cooperación, que para jugadores humanos surge de forma natural como forma de trabajo en equipo y comunicación para lograr un objetivo común, pero que todavía presenta muchos retos en este tipo de problemas. En el contexto del juego, los agentes deben conseguir resolver problemas, como determinar qué diamantes es capaz de conseguir cada uno individualmente y cuáles requieren de cierta cooperación, cuál

es el orden apropiado para recoger los diamantes, o qué serie de acciones coordinadas deben tomar los agentes para alcanzarlos, entre otros. Además, todo ello debe realizarse en un entorno físico simulado, en tiempo real y lo más rápido posible.

Para obtener buenos resultados, los agentes deberán realizar las acciones de manera precisa en un entorno físico, generar una planificación que tenga en cuenta las restricciones del nivel y coordinarse para actuar de manera conjunta. Esta coordinación se debe realizar en varios niveles de abstracción (planificación y ejecución) y es necesaria en tanto en cuanto los agentes han sido diseñados para ser en cierta manera complementarios, lo que queda patente al analizar sus respectivos conjuntos de acciones.

2.3. Estado del arte

En esta sección, presentamos una recopilación, ordenada cronológicamente, de algunas de las soluciones propuestas por otros autores a las tres competiciones del juego Geometry Friends: CircleTrack, RectangleTrack y CooperationTrack. Explicaremos en más detalle las implementaciones más relevantes para el desarrollo de nuestros agentes. Al final de esta sección, en los apartados 2.3.14 y 2.3.15, ofrecemos un resumen que engloba todas las implementaciones anteriormente expuestas y extraemos conclusiones sobre las similitudes y diferencias de las diversas implementaciones y sus resultados obtenidos.

2.3.1. CIBot

Los desarrolladores de los agentes CIBot [21, 22], de la Universidad de Senjong, participaron en las tres modalidades de la competición durante los años 2013, 2014 y 2015. Para los agentes individuales, construyen un grafo en el que los nodos son los extremos de las plataformas y las aristas representan la conectividad entre estos nodos. Después, utilizan el algoritmo de Dijkstra [14], en el caso del círculo, o un árbol de búsqueda de Monte Carlo [9], en el caso del rectángulo, para obtener el camino a seguir. Para moverse de un nodo a otro, los agentes emplean un sistema de reglas sencillo en el que se aplica una estrategia voraz.

Para la competición de cooperación, utilizan una estrategia sencilla pero razonablemente efectiva, toda basada en reglas. En primer lugar, los agentes trabajan independientemente hasta que ninguno de los dos pueda coger más diamantes por sí mismo. En ese momento, ambos agentes entran en la fase de cooperación y el círculo toma el control. Se asume que la única cooperación necesaria es que el círculo se sitúe sobre el rectángulo y que el rectángulo sirva de trampolín al círculo para poder alcanzar los diamantes que están a una mayor altura. Para ello, el círculo elige el diamante más cercano y el rectángulo sigue al círculo, procurando situarse siempre

debajo del mismo. Cuando el círculo está sobre el rectángulo, sincronizan sus velocidades hasta llegar a la vertical del diamante y, en ese momento, el rectángulo cambia de forma (haciendo la base más estrecha y aumentando de altura) y el círculo salta con la esperanza de alcanzar el diamante.

Los agentes CIBot contaban con varias limitaciones, comenzando con que las simplificaciones hechas sobre la cooperación y el modelo del nivel en forma de grafo no se ajustaban demasiado bien al problema. Además, la precisión de las acciones no era la deseable y tenían algunos problemas de planificación. Sin embargo, obtuvieron buenos resultados en las competiciones en las que participaron, aunque sufrían de una sobreespecialización en los niveles públicos y su actuación en los niveles privados caía notablemente.

2.3.2. KUAS-IS Lab

Los desarrolladores del agente KUAS-IS Lab [25], de la Universidad Nacional de Ciencias Aplicadas de Kaohsiung, se presentaron únicamente a las competiciones individuales del año 2014. Las aproximaciones para el círculo y el rectángulo fueron similares. En ellas utilizaban técnicas de búsqueda en el espacio de estados como A* [18] y de aprendizaje por refuerzo como *Q-Learning* [45] para resolver los niveles. En la fase de planificación, encontraban el camino más corto que recorriera todos los diamantes con el algoritmo A* aplicado sobre un grafo similar al que generaban los agentes CIBot. La heurística tenía en cuenta la distancia del nodo al diamante más cercano. A la hora de seguir el camino generado, en lugar de hacerlo de manera voraz, ya que se veían atrapados en algunos obstáculos, otorgaban cierto peso a los valores de una tabla Q obtenidos durante un proceso de aprendizaje por refuerzo. Los resultados obtenidos no fueron buenos.

2.3.3. OPU-SCOM

Los desarrolladores del agente OPU-SCOM [3, 4], de la Universidad de la Prefectura de Osaka, participaron en las ediciones del RectangleTrack los años 2014 y 2015. El agente se divide en dos capas de IA.

En la primera capa, crean un grafo asociado al nivel prolongando las aristas de las plataformas y generando una rejilla. Cada arista de la rejilla se convierte en un nodo, y las celdas se convierten en las aristas que conectan los nodos. También se crean nodos para los diamantes y para la posición inicial del rectángulo (puntos de interés). A continuación, utilizan el algoritmo de Dijkstra y una modificación del algoritmo PSO (Particle Swarm Optimisation) [6] que decide el mejor orden para recolectar los diamantes.

En la segunda capa, se transforman los caminos entre puntos de interés en una serie de órdenes generales como MoverseA(x), AlcanzarVelocidad(x) o AlcanzarAltura(x), ente otras. Cada orden se divide en varias acciones atómicas y, una vez ejecutadas, se evalúa si el agente está en la posición esperada, generándose nuevas órdenes en caso contrario.

En 2015, toman una aproximación diferente. Utilizan su agente del año anterior para entrenar una red neuronal NEAT (Neuro Evolution of Augmenting Topologies) [41]. Esta red neuronal recibe como entrada el estado del juego y tiene como salida la orden general que el agente debe cumplir. Al igual que el año anterior, esta orden es traducida a una secuencia de acciones atómicas y ejecutada.

Los resultados que obtuvieron en 2014 son comparables a los del agente CIBot, con una actuación aceptable para los niveles públicos y algo más limitada en los privados. Sin embargo, en 2015 el rendimiento del agente fue notablemente inferior, ya que de los 10 niveles no consiguió completar ninguno y solo fue capaz de recolectar dos de los 29 diamantes entre todos los niveles.

2.3.4. RL-Agent

El RL-Agent [29] es el primer agente que utiliza aprendizaje por refuerzo exitosamente para algunas fases de la resolución del nivel. La estrategia que se sigue consiste en varios pasos. En primer lugar, se hace un preprocesamiento del nivel en el cual se crean las plataformas, que son regiones del tablero delimitadas por obstáculos en las que el agente se puede mover sin necesidad de saltar. Cada diamante es asociado a la plataforma que tiene inmediatamente debajo. Después, se divide la resolución de cada nivel en tres fases que se repiten sucesivamente: resolver una plataforma capturando todos los diamantes en ella, decidir la siguiente plataforma a la que ir y moverse a la siguiente plataforma.

La primera y la tercera fase se resuelven mediante aprendizaje por refuerzo con la técnica *Q-Learning*, siendo las acciones utilizadas para el entrenamiento en ambos casos rodar a la derecha, rodar a la izquierda y saltar. El agente sigue una política *ε-greedy* [31].

Respecto a decidir a qué plataforma debe moverse el agente una vez resuelta la actual, se utiliza el algoritmo de búsqueda primero en profundidad (DFS) sin control de repetidos con profundidad limitada [32]. Se utiliza este algoritmo sobre un grafo generado en la fase inicial donde cada plataforma es un nodo y cada arista representa la posibilidad de ir de una plataforma a otra. El algoritmo devuelve el camino que maximiza el número de diamantes que todavía se pueden recoger.

Los resultados aportados indican que la puntuación global de este agente es comparable con la del agente CIBot. Sin embargo, la gran diferencia es que este agente

supera la sobreespecialización en los niveles públicos y tiene resultados aceptables en los privados. A pesar de ello, esta aproximación también tiene limitaciones.

Este agente sirvió de base para el desarrollo del agente PL-RL Agent [17]. Se extendió la aproximación del círculo a las competiciones del rectángulo y de cooperación. Además, se mejoraron algunos aspectos relativos a la precisión de los saltos y a la simplificación del espacio de estados en las fases de aprendizaje por refuerzo. También se cambió el algoritmo DFS por Dijkstra para generar caminos más cortos y completar el nivel en menos tiempo. En cuanto a la colaboración, se añade al estado la información relativa al otro agente con la esperanza de que así aprendan a colaborar, pero no obtuvieron buenos resultados.

2.3.5. RRT-Agent

Hay varios agentes, basados todos ellos en el uso del algoritmo Rapidly-Exploring Random Tree [23], que se desarrollan a lo largo de los años y cuyas implementaciones extienden cada una de la anterior. Son las que se presentan a continuación.

RRT-Agent original

En 2015, aparece por primera vez RRT Agent, de un grupo de desarrolladores de la Universidad de Lisboa [40]. Se centran en ser capaces de resolver cualquier nivel y no en optimizar el proceso. El nombre del agente viene del empleo del algoritmo Rapidly-Exploring Random Tree, que, aun utilizando exploración aleatoria, tiene una alta probabilidad de encontrar la solución.

Para su implementación, realizan una planificación creando puntos interesantes en el mapa y expandiendo un grafo de exploración con varias posibles acciones. Esto devuelve una secuencia de puntos clave etiquetados con lo que hay que hacer en cada punto. A continuación, se intenta seguir este plan, para lo que se centran en el control de la velocidad que hay que llevar en ciertos puntos clave y que resuelven con un sistema de control PID (Proportional Integral Derivative) [16].

A pesar de que los autores tuvieron que entregar una versión incompleta de los agentes, el RRT-Agent ganó en la competición del círculo de 2015, superando a CIBot con bastante margen. En el rectángulo, sin embargo, los resultados fueron inferiores de los de CIBot. Cabe destacar que, aunque el agente del círculo es mucho menos especializado que CIBot, los resultados siguen sin acercarse a ser perfectos.

RRT2017

El agente RRT2017 [33] se desarrolló utilizando el agente RRT como referencia. Mejoró la búsqueda utilizando el modelo de predicción proporcionado por el juego, que no existía cuando se desarrolló el primer agente. Aunque era costoso desde el punto de vista computacional, era muy preciso. También se incluyeron algunas otras extensiones a la búsqueda RRT para reducir el tiempo de búsqueda, junto con una capacidad de replanificación.

Además, se desarrolla un agente cooperativo. Este agente utiliza un controlador diferente al de los agentes individuales, ya que realiza la planificación del movimiento en tiempo real, verificando la posición actual y la velocidad del agente y comparándola con la deseada, para después calcular la acción a ejecutar.

Los principales problemas están relacionados con el tiempo de búsqueda debido al uso del modelo directo, lo que es especialmente grave en los agentes cooperativos. Muestran sus resultados en la competición de 2017, en la que quedaron segundos en el CircleTrack, y primeros en las otras dos, con grandes mejoras sobre el RRT-Agent original.

Rule Base RRT

En 2019, algunos de los desarrolladores del RRT2017 implementaron una versión mejorada del agente en la que se basan en un sistema de reglas [34, 39]. Este está especialmente diseñado para mejorar la cooperación entre agentes, de forma que cada diamante se asigna a uno de los agentes, si lo pueden coger sin ayuda, o a ambos, si necesitan cooperar. En función del problema, se emplean unas reglas u otras.

Tras probar varias extensiones para el algoritmo, los desarrolladores acabaron usando *Behavioral Kinodynamic Balanced Growth Trees* [46] con un *bias* hacia los estados objetivo. Para la planificación de sus agentes individuales, eligen el diamante objetivo al que se van a dirigir, priorizando el que esté en la misma plataforma que ellos, y, si no hay, el diamante que esté en una posición más alta. Respecto a los predictores que utilizaba RRT2017, vuelven a usar el predictor proporcionado para el rectángulo, pero para el círculo desarrollan uno más rápido.

Para el modo cooperativo, la estrategia es muy similar, pero teniendo en cuenta los movimientos de ambos agentes a la hora de expandir el árbol de búsqueda, lo que aumenta en gran medida el tiempo de planificación.

Aún con todas sus aparentes mejoras, en la competición de 2019, el agente RRT2017 obtuvo ligeramente mejores resultados en la competición cooperativa, aunque ambos estuvieron muy por detrás del ganador, MARL-GF.

2.3.6. Subgoal A* Agent

El desarrollador de este agente [15], de la Universidad de Maastricht, participó en 2015 en la competición RectangleTrack. La aproximación seguida es similar a la del agente CIBot, donde en primer lugar se crea una abstracción del nivel en forma de grafo dirigido, cuyas aristas se etiquetan con la forma que debe tener el rectángulo para poder pasar por ellas. Los nodos son los puntos de interés del nivel, que pueden ser, por ejemplo, extremos de las plataformas, posiciones de caída del rectángulo o localizaciones justo debajo de los diamantes.

Tras computar el grafo, se lanza el algoritmo de búsqueda Subgoal A*, que es una modificación del algoritmo A* desarrollada por el propio autor.[2] Una vez generado el camino, se procede a la ejecución del mismo mediante un sistema de reglas.

Los resultados obtenidos son mejores que los de los ganadores de los años 2013 y 2014, y se proclamó vencedor de la edición del año 2015 de la competición RectangleTrack. En las competiciones posteriores, este agente fue usado de referencia para comparar el rendimiento de las nuevas implementaciones.

2.3.7. KIT Agent

Los desarrolladores del agente KIT Agent [28], del Instituto Tecnológico de Kioto, participaron en las ediciones de los años 2017 y 2022 de la competición para el círculo. Dividen la resolución del nivel en dos fases: encontrar el camino más corto que consiga recolectar todos los diamantes y seleccionar las acciones del agente para seguir dicho camino. En la primera fase, generan un grafo dirigido que representa el nivel y aplican el algoritmo de búsqueda Subgoal A* para encontrar el camino buscado. La aproximación que siguen para generar el grafo es novedosa respecto a las implementaciones previas, ya que las aristas son generadas analizando las posibles trayectorias de salto y caída del agente desde una plataforma.

En primer lugar, se parte de la simplificación de que la trayectoria del agente no tiene obstáculos y que esta, al encontrarnos en un entorno dinámico, viene dada por las ecuaciones del tiro parabólico. Es decir, la posición del móvil a tiempo $\mathtt{t}$ viene dada por las ecuaciones

$$\begin{cases} x(\mathtt{t}) = x_0 + v_x\mathtt{t}, \\ y(\mathtt{t}) = y_0 + v_y\mathtt{t} - \frac{g\mathtt{t}^2}{2}, \end{cases}$$

donde (x_0, y_0) es la posición inicial del móvil, v_x es la velocidad horizontal inicial, v_y

[2] En este algoritmo, la heurística se fija a 0, pero a los nodos se les añade información relativa al orden y número de diamantes ya recogidos. Por tanto, los nodos finales son aquellos en los que se capturan todos los diamantes. Sin embargo, se limita el tiempo dedicado a la búsqueda. Si tras alcanzar dicho límite no se ha conseguido un camino que consiga recolectar todos los diamantes, se resta 1 al número de diamantes que se quiere recolectar y se vuelve a ejecutar el algoritmo, y así hasta que se tiene éxito.

es la velocidad vertical inicial y g es la aceleración de la gravedad. En este contexto, el valor de g es una constante proporcionada por el juego. Para los saltos, v_y es otra constante proporcionada por el juego y, para los casos en los que el agente cae por un extremo de la plataforma, $v_y = 0$. Además, para una plataforma concreta, y_0, es decir, la altura de la plataforma, es un valor fijo y $x_i \leq x_0 \leq x_d$ siendo x_i y x_d la posición de los extremos izquierdo y derecho de la plataforma, respectivamente. Por otro lado, $|v_x| \leq v_{max}$ para cierta constante positiva v_{max} que indica la velocidad horizontal máxima del agente y que es un parámetro proporcionado por el juego. Por tanto, tenemos dos variables libres, x_0 y v_x, que se mueven en un cierto intervalo y que nos determinan la trayectoria del agente en los saltos desde una plataforma concreta.

La solución propuesta consiste en discretizar los intervalos anteriores y probar todas las combinaciones de valores x_0 y v_x, que generarán una buena aproximación de todos los posibles saltos y caídas desde una plataforma. Para cada una de las trayectorias, se analiza a qué plataforma se llega tras realizar dicho salto o caída y si se intercepta algún diamante. Si la plataforma de llegada es distinta de la que se parte, se genera una arista que se etiqueta con la información (x_0, v_x). Es decir, si el agente se encuentra en la posición x_0 con velocidad v_x y salta (o se deja caer, dependiendo del caso), entonces llegará a la plataforma destino. Por tanto, el problema de desplazarse de una plataforma a otra y de alcanzar los diamantes se reduce a que el agente aprenda a llegar a una determinada posición x_0 con una determinada velocidad v_x. Este aprendizaje se realiza mediante *Q-Learning* y la representación de estados propuesta utiliza las simetrías del juego para acelerar el proceso de aprendizaje.

Las principales limitaciones del agente [27] se producen por el hecho de no tener en cuenta que las trayectorias pueden encontrarse con obstáculos y porque la plataforma de llegada puede ser demasiado estrecha y, al llegar a ella con una velocidad horizontal demasiado elevada, puede que el agente no tenga suficiente tiempo para frenar y caiga por uno de los extremos. Con todo, los resultados obtenidos son muy buenos.

2.3.8. Supervised DL Agent

En 2017, un estudiante de Máster del Instituto Técnico de Lisboa [7] intentó usar varios enfoques diferentes mediante Deep Learning [2] para desarrollar varios agentes del círculo. En primer lugar, implementó un agente que emplea únicamente Aprendizaje Supervisado [13], intentando clasificar los diferentes estados del juego según qué acción se debía realizar en ellos. Sin embargo, tenía muy pocos datos para entrenar el modelo, ya que los tuvo que generar jugando él mismo. A continuación, creó varios agentes que utilizaban Aprendizaje por Refuerzo [35]. Entre ellos, hay uno basado en *Q-Learning* con redes neuronales (algoritmo DQN [26]), otro que optaba por descenso de gradiente y otro que utilizaba demostraciones (algoritmo

DQfD [19]), que también hacía uso de partidas jugadas por humanos.

El primero de los agentes es el que obtuvo un mejor rendimiento, seguido del agente basado en el algoritmo DQfD. Sin embargo, comparados con otros agentes, no destacan en cuanto a resultados.

2.3.9. Neural Reinforcement Agent

El desarrollador de este agente [1], de la Universidad de Lisboa, participó en la edición de 2017 en la competición CircleTrack. Esta propuesta se basa en Aprendizaje por Refuerzo, utilizando una red neuronal prealimentada para aproximar la función de valor. Se modela el entorno como un MDP (Markov Decision Process) y se utiliza *Q-Learning* para aproximar la función de valor, que mide la recompensa esperada a largo plazo de realizar una determinada acción en un determinado estado. De esta forma, la política óptima es elegir en cada estado la acción que maximice la función de valor.

Se define la función de recompensa de tal forma que se premie al agente al acercarse a los diamantes y se le penalice cada iteración para que el tiempo que invierta sea mínimo. Los estados finales son aquellos en los que se captura un nuevo diamante y se sigue una estrategia *ε-greedy*. El algoritmo de aprendizaje utilizado fue una modificación de *Neural Fitted Q Iteration* [30], que entrena la red por descenso de gradiente estocástico para minimizar el error de la predicción de la función de valor. El autor reconoce que esta aproximación requiere de un mayor tiempo de entrenamiento y un mayor volumen de datos para obtener buenos resultados.

2.3.10. MARL-GF Agent

El desarrollador de este agente [36], de la Universidad de Lisboa, participó en las tres competiciones (CircleTrack, RectangleTrack y CooperationTrack) del año 2019. Se inspiró en la solución propuesta por el agente KIT y la extendió al rectángulo y al modo cooperación. Para el rectángulo, la fase de entrenamiento consistió en que el agente aprendiera mediante *Q-Learning* a alcanzar una determinada posición con una determinada velocidad horizontal, de manera similar a cómo fue entrenado el agente KIT (círculo), pero teniendo en cuenta que la altura del rectángulo es variable. Para ello, redujo las posibles formas del rectángulo a base ancha, cuadrado o base estrecha, y el resto del problema se abordó de manera idéntica a como lo hace el agente KIT, aunque sin considerar que el agente pueda saltar.

En cuanto al modo cooperación, se sigue una estrategia en la que el círculo actúa como líder y se comunica con el rectángulo para conseguir los objetivos. Durante la fase de modelado del nivel, cada agente crea su propio grafo. Lo novedoso es que

el círculo incluye plataformas cooperativas, que son plataformas ficticias sobre las plataformas alcanzables por el rectángulo y que representan la posibilidad de que el círculo se sitúe encima del rectángulo. Estas plataformas tienen una altura dinámica y pueden ser vistas como 3 plataformas, situadas a cada una de las posibles alturas del rectángulo. Todas las plataformas, incluyendo las cooperativas, son utilizadas para analizar las posibles trayectorias de salto del círculo al estudiar la conectividad entre plataformas.

Durante la fase de ejecución, los agentes tienen un sistema de mensajes tal que, si el círculo va a efectuar un movimiento que requiera de una plataforma ficticia, demanda la ayuda del rectángulo para situarse en una posición específica. Utilizan el método de aprendizaje cooperativo *Team Q-Learning* [11], aunque también se probó a utilizar OAL (Optimal Adaptative Learning) [44]. Se les entrenó para que se desplazasen manteniendo la posición en la que el círculo se sitúa sobre el rectángulo hasta llegar a una posición determinada. Los estados objetivos son aquellos en los que se llega a una determinada posición con velocidad 0 y el círculo se mantiene sobre el rectángulo.

Aunque los resultados obtenidos para el rectángulo son solamente algo mejores que los de las ediciones anteriores, en el modo cooperación obtuvieron puntuaciones considerablemente mayores a las de los agentes ya existentes.

2.3.11. NKUST

Los agentes NKUST [20] también abordan la cooperación. La "posición de conducción" es el enfoque principal, ya que se considera que es el tipo de colaboración más común a través de los niveles presentados. Primero, se calcula el área donde se puede mover el rectángulo y, por lo tanto, qué diamantes puede atrapar. Con esto, el personaje del círculo puede obtener más información sobre las áreas a las que puede llegar, ya que el rectángulo se ve como una plataforma móvil.

Al buscar un camino, la prioridad es atrapar los diamantes en un orden que permita capturar la mayor cantidad de diamantes usando una búsqueda en profundidad. Luego, se intenta encontrar un camino con el algoritmo A*. Si no se encuentra una ruta completa, el agente sigue, de las que ya descubrió, la que tiene más diamantes.

En términos de división de tareas, el rectángulo tiene prioridad. Si este personaje puede atrapar todos los diamantes por sí mismo, entonces el círculo no se usa. Solo cuando hay uno o más diamantes que el rectángulo no puede atrapar, el círculo realiza la planificación de la ruta considerando el espacio de movimiento del rectángulo. Siempre que el círculo necesita del rectángulo para llegar a una posición, esta información se agrega a los "nodos colaborativos" de la búsqueda. Finalmente, cuando el círculo termina su planificación, el rectángulo realiza otra, con la información de los nodos que necesitan colaboración si los hubiere. Para realizar estas

tareas cooperativas, el agente que llegue antes esperará al otro.

2.3.12. AGAgent

Los desarrolladores de este agente [12], de la Universidad de Texas en Arlington, participaron en la competición CircleTrack en el año 2019. Dividen la resolución de los niveles en creación del modelo, planificación y ejecución.

En primer lugar, siguiendo una idea parecida a la de los agentes KIT y MARL-GF, convierten el nivel en un grafo dirigido que construyen incrementalmente, partiendo de la posición actual del agente y estudiando la conectividad entre segmentos (lo que otros autores llaman plataformas) y diamantes de acuerdo a las posibles trayectorias parabólicas, que ignoran la presencia de obstáculos. Sin embargo, sí que tienen en cuenta cómo de cerca pasan las trayectorias de las esquinas de los obstáculos y, en la medida de lo posible, favorecen trayectorias que no pasan próximas a estos, ya que tienen más posibilidades de ser precisas.

En la fase de planificación, se realiza una búsqueda en anchura sobre el grafo para encontrar un camino que recorra el mayor número de diamantes. Finalmente, en la fase de ejecución, utilizan unas ciertas políticas de control que genera la fase de planificación. Estas consisten en subobjetivos de alto nivel, que se combinan para resolver el problema. Cuando una acción falla, se repite el proceso de generación del grafo desde la plataforma en la que se encuentra el agente.

Los resultados que obtienen son bastante buenos, ya que afirman que consiguen completar todos los niveles de todas las competiciones anteriores al año 2019. Sin embargo, en algunas ocasiones, el proceso de creación del grafo, aunque útil a la hora de generar información precisa relativa al movimiento entre plataformas, es costoso en tiempo y tiene que ser limitado.

2.3.13. Otros agentes

Mencionamos dos agentes más que, aun habiendo sido desarrollados y podido encontrar detalles de su implementación, no participaron en ninguna competición. Los autores de ambos agentes modificaron ligeramente el juego para poder desarrollarlos.

En primer lugar, algunos desarrolladores de la Universidad Técnica de Estambul [47], crearon un agente para la competición del círculo en el que utilizan una red neuronal convolucional [24] para recibir como entrada la imagen de la pantalla, junto con la puntuación y el tiempo. Para procesar la pantalla, se convierte esta imagen a una escala de grises y se cambia el tamaño a un 30 % de la original. Para el estado se usa una superposición de 4 *frames*, y las acciones solo se toman cuando el círculo

Agente	CircleTrack	RectangleTrack	CooperationTrack
CIBot	2013-2015	2013-2015	2013-2015
KUAS-IS Lab	2014	2014	No
OPU-SCOM	No	2014-2015	No
RL-Agent	2015-2016	2016	2016
RRT-Agent	2015-2017	2015-2017	2016-2017
RRT2017	2017-2019	2017-2019, 2022	2017-2019, 2022
Rule Based RRT	No	No	2019
Subgoal A* Agent	No	2015	No
KIT Agent	2017, 2022	No	No
Supervised DL Agent	2017-2019	No	No
Neural Reinforcement Agent	2017	No	No
MARL-GF Agent	2019	2019	2019
NKUST	No	No	2019
AGAgent	No	2019	No

Tabla 2.1: Ediciones en las que participaron los agentes analizados.

está en una plataforma o en el suelo. El resultado es bastante malo, siendo apenas mejor que un agente aleatorio.

En segundo lugar, investigadores de la Universidad de Aveiro [38] utilizaron el aprendizaje profundo asíncrono para crear nuevos agentes. Los autores explicaron que inicialmente tuvieron dificultades durante la capacitación debido al alto coste del tiempo por pasos de capacitación y a la cantidad insuficiente de muestras para lograr políticas decentes. Para ayudar a acelerar el proceso de capacitación, utilizaron el enfoque de migas de pan y técnicas de modelado de entrada para disminuir la complejidad de la red.

2.3.14. Resumen

Tras esta recapitulación de algunas soluciones propuestas para el juego Geometry Friends, vamos a mostrarlas de manera más compacta resaltando las características más destacadas de cada agente. En primer lugar, la tabla 2.1 recoge las ediciones y las competiciones en las que participaron los agentes analizados. La tabla 2.2 resume las técnicas de IA utilizadas por cada agente. En ellas, se han excluido los agentes que no han participado en ninguna competición. Por último, en la tabla 2.3 se muestran los agentes ganadores en cada una de las 3 competiciones de cada año según los resultados publicados en la página web del concurso.

Agente	Planificación	Ejecución
CIBot	Dijkstra, MCTS	Sistema de reglas
KUAS-IS Lab	A*	*Q-Learning*
OPU-SCOM	Dijkstra, PSO, NEAT	Sistema de reglas
RL-Agent	DFS, Dijkstra	*Q-Learning*
RRT-Agent	RRT	Controlador PID
RRT2017	RRT	Controlador PID
RRT2017	RRT	Sistema de reglas
Subgoal A* Agent	Subgoal A*	Sistema de reglas
KIT Agent	Subgoal A*	*Q-Learning*
Supervised DL Agent	DL, RL	DL, RL
Neural Reinforcement Agent	RL	RL
MARL-GF Agent	Subgoal A*	*Q-Learning*, *Team Q-Learning*
NKUST	A*	*Q-Learning*
AGAgent	A* (BFS)	*Q-Learning*

Tabla 2.2: Técnicas de IA utilizadas por los agentes.

	CircleTrack	RectangleTrack	CooperationTrack
2013	CIBot	CIBot	CIBot
2014	CIBot	CIBot	CIBot
2015	RRT Agent	Subgoal A* Agent	-
2016	RRT Agent	RRT Agent	RRT Agent
2017	KIT	RRT2017	RRT2017
2019	KIT	MARL-GF Agent	MARL-GF Agent
GF-CoG 2022	KIT	RRT2017	-
GF-IJCAI-ECAI 2022	KIT	RRT2017	MARL-GF Agent

Tabla 2.3: Ganadores de las competiciones celebradas hasta el momento.

2.3.15. Conclusiones

Una vez expuestas todas las ideas de las implementaciones anteriores, nos gustaría identificar patrones comunes en las diferentes propuestas de agentes. Podemos distinguir dos ramas claramente diferenciadas en cuanto al funcionamiento de los agentes.

Por un lado, tenemos las implementaciones que están fuertemente asociadas al modelo que se crea del nivel. Estas soluciones, a grandes rasgos, siguen un planteamiento común. En primer lugar, parten de un procesamiento del nivel en el que se genera un grafo para representar la disposición de los agentes, diamantes y obstáculos en el mapa. Este preprocesamiento se hace al principio del nivel y no afecta al tiempo de juego.

Hay dos vertientes para la generación del grafo, pudiendo optarse o bien por analizar los puntos críticos en el mapa, o bien por estudiar la conectividad entre plataformas

	Generación del grafo	
Algoritmo de búsqueda	Puntos de interés	Trayectorias parabólicas
Subgoal A*	Subgoal A* Agent	KIT Agent MARL-GF Agent
A*	KUAS NKUST	
Dijkstra	CIBot OPU-SCOM RL-Agent	
BFS/ DFS		AGAgent
	RL-Agent	
Otro	CIBot OPU-SCOM RRT	

Tabla 2.4: Organización conceptual de los agentes que representan el nivel en forma de grafo

basada en trayectorias parabólicas de saltos y caídas. Los resultados muestran que la segunda de las opciones tiene un mejor rendimiento. Posteriormente, hay una fase de planificación, que consiste en lanzar algún algoritmo de búsqueda sobre el grafo generado para encontrar un camino óptimo o subóptimo que recorra todos los diamantes. De entre todos los algoritmos utilizados, uno que es implementado por agentes que obtienen buenos resultados es Subgoal A*. Sin embargo, debido a que los grafos generados no son excesivamente grandes, no parece que la elección del algoritmo de búsqueda sea determinante a la hora de resolver el problema, ya que algunos algoritmos de búsqueda exhaustiva han sido utilizados con buenos resultados. Por último, el plan de acción generado en la planificación se le proporciona a un controlador, que elige los movimientos básicos del agente para seguir dicho plan.

Dentro de este grupo de agentes, existen diferencias entre aquellos que realizan una replanificación cuando completan algún hito del plan (lo que les sirve para corregir errores), y otros que manteniendo el mismo plan adaptan las acciones a ejecutar. Esta clasificación se resume en la tabla 2.4.

La otra aproximación no utiliza un modelo del nivel, sino que se apoya en el uso de aprendizaje supervisado profundo para que el agente aprenda por sí solo a resolver el nivel. La principal dificultad con la que se topan estos agentes es la falta de niveles de entrenamiento. Para entrenar las redes neuronales en las que se basan estas implementaciones, el volumen necesario de niveles debería ser inmenso. Si existiera un gran repertorio de niveles que se supiera que son resolubles, se podría utilizar como base para el entrenamiento, y esta aproximación tendría más oportunidades de resultar exitosa. Sin embargo, aunque no es muy complicado generar niveles automáticamente, resulta imposible saber de antemano si un nivel va a ser resoluble al generarlo.

Así, en el contexto actual, los resultados de estos agentes son significativamente peores que los que toman la otra alternativa de implementación, y apenas superan los agentes que eligen sus acciones de forma aleatoria. Estos agentes son OPU-SCOM en su versión de 2015, Supervised DL Agent, Neural Reinforcement Agent y el agente de la Universidad Técnica de Estambul.

Tras haber descrito las aproximaciones del estado del arte, en el siguiente capítulo comenzamos a explicar nuestra propuesta de solución. En concreto, hablaremos sobre la arquitectura común de los personajes del círculo y el cuadrado.

Capítulo 3

Desarrollo común al círculo y el rectángulo

En el presente capítulo, se detallará la parte de la arquitectura de la solución que es común a ambos agentes y podremos entender, en líneas generales, cuál es la estrategia global que siguen los agentes para resolver los niveles individuales. Las particularidades del círculo se explicarán con más detalle en el capítulo 4, mientras que las del rectángulo se encuentran en el capítulo 5. Asimismo, se presentan las herramientas que se van a necesitar durante el desarrollo de los agentes.

3.1. Introducción

Nuestro objetivo principal del trabajo es el desarrollo de agentes individuales y cooperativos que puedan participar en las competiciones de Geometry Friends y obtener resultados equiparables a los mejores agentes creados hasta el momento. Para abordarlo, vamos a comenzar desarrollando los agentes individuales ya que, más allá de que todo lo que aprendamos en el desarrollo de los mismos es aplicable al caso de la cooperación, los retos para la IA que aparecen en estos modos de juego tienen interés *per se*. En esta sección, pretendemos dar una visión general de aquello que explicaremos con más detalle en la sección 3.3 y en los capítulos 4 y 5, la arquitectura de la solución. Describimos la forma que tienen nuestros agentes de abordar los niveles a continuación.

En primer lugar, el juego proporciona la información del nivel al agente. Fundamentalmente, esta información es la posición de los obstáculos y del agente. Con esta información el agente crea una representación propia del nivel, más simplificada que la que le proporciona el juego y que tiene que procesar. Este procesamiento consiste fundamentalmente en tres aspectos:

1. Identificar las plataformas, es decir, las partes superiores de los obstáculos que

el agente puede recorrer de manera sencilla, simplemente con las acciones de desplazamiento de moverse a izquierda o derecha.

2. Identificar y simular los movimientos que permiten cambiar de plataforma o alcanzar un diamante, como pueden ser saltos, caídas u otros movimientos especiales que iremos detallando según surjan.

3. Elegir, de todos los movimientos que permiten cambiar de plataforma o alcanzar un diamante, un subconjunto "pequeño" que garantice la máxima conexión y que esté formado por los "mejores" movimientos.

A este procesamiento lo llamaremos de aquí en adelante *Representación del nivel* y explicaremos la parte común al círculo y al rectángulo en la sección 3.3.1, las particularidades del círculo en la sección 4.2 y las del rectángulo en la sección 5.2.

Una vez hayamos identificado las plataformas y el subconjunto de movimientos, generamos un grafo donde los vértices son las plataformas y las aristas son los movimientos que las unen. Sobre el grafo ejecutamos un algoritmo de búsqueda que encuentra un plan de alto nivel para el agente, es decir, una lista de movimientos que cambian de plataforma. Por ejemplo, un plan para el círculo podría ser (Saltar de la plataforma 1 a la 4, Caer desde la plataforma 4 a la 6). El plan se genera de tal forma que si el agente, antes de realizar cada uno de los movimientos para cambiar de plataforma, se asegura de haber alcanzado todos los diamantes que se encuentran sobre su plataforma actual, entonces el agente alcanza todos los diamantes. Por tanto, dado el plan anterior, lo que deberá hacer el agente es (Alcanzar todos los diamantes de la plataforma 1, Saltar de la plataforma 1 a la 4, Alcanzar todos los diamantes de la plataforma 4, Caer desde la plataforma 4 a la 6, Alcanzar todos los diamantes de la plataforma 6). El *Trazado de un plan a seguir* es fundamentalmente común al círculo y a la rectángulo y se detalla en la sección 3.3.2, aunque algunas particularidades de la planificación del rectángulo se explican en la sección 5.3.

Por último, el agente debe realizar la fase de *Ejecución del plan*, donde lleva a cabo el plan de alto nivel mediante acciones de bajo nivel. Esto lo detallamos en las secciones 3.3.3, 4.3 y 5.4. La forma de ejecutar el plan de alto nivel es realizando cada uno de los movimientos que lo componen de manera ordenada. En la fase de generación de movimientos, a estos se les añade cierta información como la posición del movimiento dentro de la plataforma de partida y la velocidad que debe alcanzar el agente en esa posición. Allí también nos aseguramos de que, si el agente llega con esas condiciones al punto del movimiento y realiza las acciones oportunas asociadas al movimiento, entonces este se realiza de manera exitosa y se llegará correctamente a la plataforma destino del movimiento. Por tanto, realizar cada uno de los movimientos consiste en cumplir estas precondiciones del movimiento, es decir, llegar a la posición del movimiento con una determinada velocidad y realizar las acciones asociadas a dicho movimiento. Esto finaliza la visión general de cómo resuelven nuestros agentes los niveles y, como hemos ido comentando, los detalles concretos que dan forma a la arquitectura se pueden encontrar en las secciones oportunas.

3.2. Software y herramientas

La implementación de los agentes debe acogerse a una plantilla que se proporciona en la página del juego. Esta plantilla viene en la forma de un archivo comprimido, que contiene una solución a abrir mediante Visual Studio. En esta se puede encontrar un archivo correspondiente a cada uno de los agentes, "CircleAgent.cs" y "RectangleAgent.cs". Estos son los únicos archivos que se han de modificar de toda la plantilla, aunque también se pueden crear nuevos archivos para poder realizar la implementación de una forma más modular. Como ya se mencionó en la introducción, esta implementación debe estar desarrollada, al menos parcialmente, en el lenguaje de programación C#. Este lenguaje de programación es muy similar a Java, y todo se opera a través de clases. Algunas características reseñables de C# son que es un lenguaje de programación orientado a objetos, con tipado fuerte, polimorfismo y que cuenta con un recolector de basura.

El juego se ejecuta a través de una aplicación que enlaza dinámicamente el código de los agentes mediante una DLL. Una DLL, del inglés *Dynamic Link Library*, es una biblioteca de enlace dinámico, es decir, un archivo de código ejecutable que se carga bajo demanda de un programa. La principal ventaja de las DLLs que se explota en este caso es la flexibilidad frente a los cambios, ya que permite que se realicen implementaciones de múltiples agentes sin la necesidad de modificar el código fuente del juego, que por otro lado no se proporciona. Por tanto, las implementaciones de los agentes se compilan como DLLs y el código fuente del juego llama a las funciones de los agentes como si de una librería se tratase.

Cada uno de los agentes está representado en su clase, que hereda de una clase abstracta que hace las veces de interfaz de comunicación con el juego. Cada agente cuenta con unos métodos que reciben información del juego (aquellos que son utilizados para que el agente actualice la información dinámica periódicamente) y otros que sirven para que el agente envíe información al juego (como su próxima acción).

De entre los métodos que reciben información del juego, destacamos:

- El método *Setup*, que es llamado por el juego al inicio de cada nivel. Recibe como argumentos la información sobre la disposición de los obstáculos y los diamantes, así como la posición inicial de los agentes. Adicionalmente se proporciona información como el tiempo límite o las dimensiones del nivel.

- El método *SensorsUpdated*, que el juego llama continuamente para actualizar la información dinámica del nivel, es decir, los diamantes que quedan por coger y las posiciones y velocidades de los agentes.

- El método *EndGame* que se llama al finalizar el nivel y que informa al agente de su desempeño, indicándole el tiempo empleado y los diamantes capturados.

Adicionalmente, el juego llama periódicamente al método *Update*, que ha de contener toda la lógica de ejecución del agente. Aquí es donde se actualiza toda la información dinámica del mismo y que posteriormente será proporcionada al juego cuando este llame a los métodos apropiados. Por ejemplo, es aquí donde se debe actualizar la variable *currentAction*, que posteriormente será devuelta en *GetAction*.

Queremos detenernos momentáneamente en la representación de los objetos del juego, es decir, diamantes, obstáculos y agentes. Los tipos de todos sus atributos son *float*, ya que se nos ofrece una representación "continua" del estado. De todos estos objetos se conoce la posición (x, y) de su centro. Además, los obstáculos tienen siempre forma rectangular y se proporciona su altura y anchura. Los diamantes, por su parte, tienen forma de rombo y son de tamaño fijo. Por último, la representación de los agentes incluye su velocidad en ambos ejes (v_x, v_y). Para conocer la forma del rectángulo, también se proporciona su altura, que es información suficiente al ser su área constante.

En un principio, puede parecer que la información que proporciona el entorno es suficiente para tener una representación completa de cada instante, pero hay cierto conocimiento del que no disponemos, lo que se traduce en limitaciones que detallaremos a medida que se vayan introduciendo.

Por último, el juego Geometry Friends también incorpora funcionalidades muy interesantes y que resultan de gran ayuda para el desarrollo de los agentes. Entre ellas, destacamos un *batch simulator*, que permite la ejecución de una gran cantidad de niveles de manera automática y a una velocidad mayor que la real, un depurador visual, que posibilita el imprimir información por la interfaz del juego de manera dinámica, y un generador de niveles, que permite diseñar nuevos niveles desde cero, pudiendo elegir la localización y tamaño de los obstáculos y las posiciones de diamantes y agentes. A lo largo de la memoria, explicaremos en más detalle estas herramientas cuando las utilicemos, pero toda la información de las mismas está accesible en la página web de la organización del juego: `https://gaips.inesc-id.pt/geometryfriends/?page_id=801`.

3.3. Arquitectura de la solución

Aunque detallaremos cada implementación específica en un capítulo separado, algunas de las técnicas que aplicaremos son comunes a ambos agentes, especialmente en lo relativo a cómo se trata el entorno y se describe el estado.

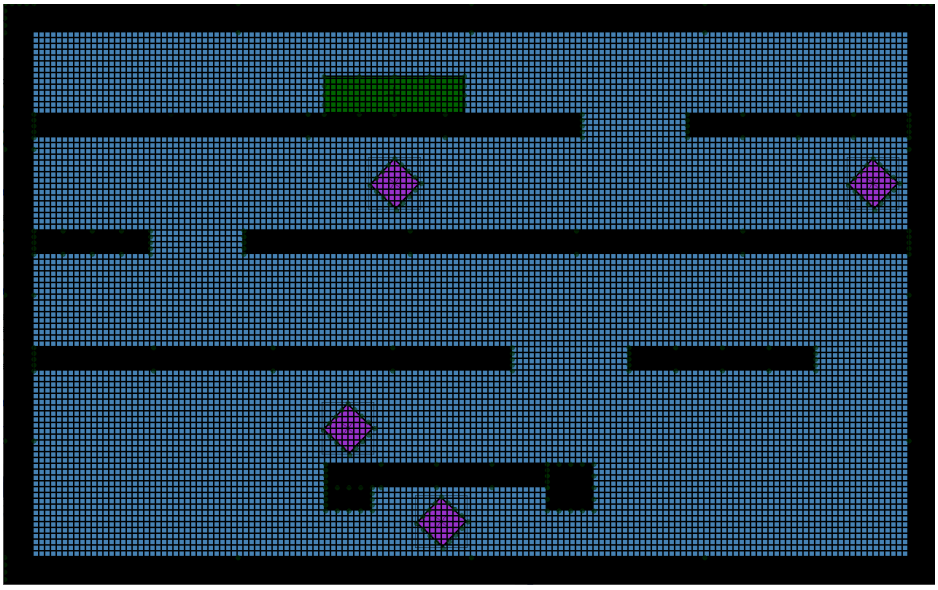


Figura 3.1: Mallado usado para la discretización. Cada cuadrado tiene un lado de 8 píxeles.

3.3.1. Representación del nivel

El primer paso que realizamos es traducir la información estática proporcionada a través de los agentes a una representación con la que sea cómoda tratar. Para ello, contamos con la clase *LevelMap*, de la que se creará una instancia al comenzar el nivel. En primer lugar, dado que el tamaño de los niveles es fijo (1272 píxeles de ancho por 776 píxeles de alto) y la representación de las posiciones toman valores continuos, realizamos una discretización que nos permita tratar con menos información sin perder funcionalidad. Se trata de encontrar un equilibrio entre precisión y rendimiento. Una discretización demasiado fina eleva excesivamente los costes computacionales, críticos en problemas que hay que resolver en tiempo real. Por el contrario, si nuestra discretización no consigue distinguir situaciones distintas, estaremos perdiendo exactitud. Con todo ello en mente, elegimos dividir todas las medidas entre 8 y quedarnos con la parte entera. Hemos podido comprobar que esta escala ha sido utilizada con éxito en otras implementaciones. Además, el tamaño mínimo de un obstáculo en el juego es un cuadrado de lado 16 píxeles, que con nuestra representación correspondería con un cuadrado de 2 unidades de lado, lo cual resulta razonable.

El objetivo es crear en primer lugar una representación de puntos, es decir, cuadrados de 8×8 píxeles, a través de una matriz, que nos permita decir qué hay en un punto concreto. Para lo que sigue, debemos tener en mente la figura 3.1. Las posibilidades para cada punto vienen dadas a través de un enumerado, que hemos llamado *PixelType*, que puede tomar los valores OBSTACLE, DIAMOND, PLATFORM o EMPTY. Describiremos qué representa cada uno de estos valores en los métodos siguientes. Así, cuando se crea una instancia de la clase *LevelMap* y se llama a la

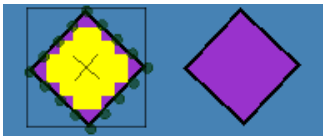


Figura 3.2: Diamante discretizado (izquierda) frente al diamante completo (derecha).

función *CreateLevelMap*, se llevan a cabo una serie de acciones:

1. Método *SetCollectibles*: este método asigna a los puntos que corresponden a los diamantes el valor DIAMOND. Sin embargo, hay que tener en cuenta que, debido al compromiso entre precisión y rendimiento en la discretización que hemos realizado, la representación resultante de los diamantes no es tan fiel como se podría desear. Para los diamantes, un ejemplo de esta discretización se encuentra en la figura 3.2.

   En esta imagen, la información discretizada es la que aparece en color amarillo.

   Como se ve, se ha elegido una representación estrictamente contenida en el diamante completo. Esto se debe a que, si en nuestra discretización tuviéramos algún punto que no perteneciera al diamante original, nuestros agentes, a la hora de planificar, podrían pensar que son capaces de coger algún diamante siguiendo una determinada trayectoria cuando en realidad no podrían.

2. Método *SetDefaultObstacles*: este método fija los bordes externos del nivel, es decir, los obstáculos que aparecen en todos los niveles en las partes superior, inferior, izquierda y derecha. Los puntos pertenecientes a estos objetos se fijan al valor OBSTACLE.

3. Método *SetObstacles*: análogo al anterior, pero para los obstáculos específicos de cada nivel. Cada agente fija al valor OBSTACLE los puntos correspondientes a objetos que no son de su color, es decir, aquellos que no puede atravesar. Por tanto, para un mismo nivel, es posible que los agentes tengan representaciones diferentes.

4. Métodos *IdentifyPlatforms*, *IdentifyDefaultPlatforms* y *PlatformUnion*: este conjunto de métodos tiene como objetivo identificar aquellos puntos del mapa que son obstáculos sobre los que el agente puede apoyarse, que es lo que llamamos plataformas. Es decir, para un obstáculo, se evalúa sobre qué puntos de su parte superior podría situarse el agente sin intersecar con otros obstáculos. A estos, se les asigna el valor PLATFORM. Dichos puntos son agrupados si son adyacentes horizontalmente, y se genera un objeto de tipo *Platform* que representa una línea horizontal sobre la cual nuestro agente podría estar situado sin moverse. Como la forma del rectángulo es variable, profundizaremos en sus particularidades en el capítulo correspondiente.

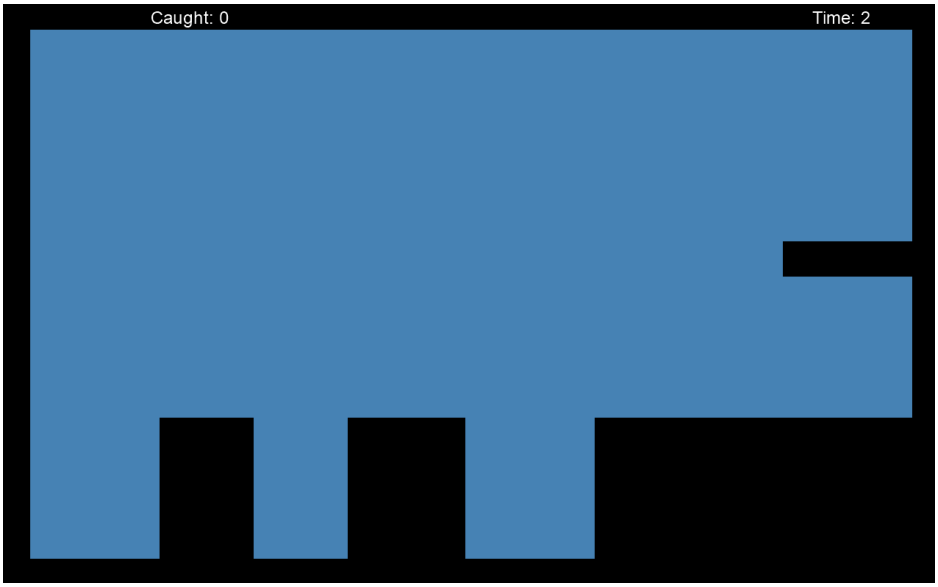


Figura 3.3: Obstáculos originales del nivel 7 de la competición del círculo de 2016.

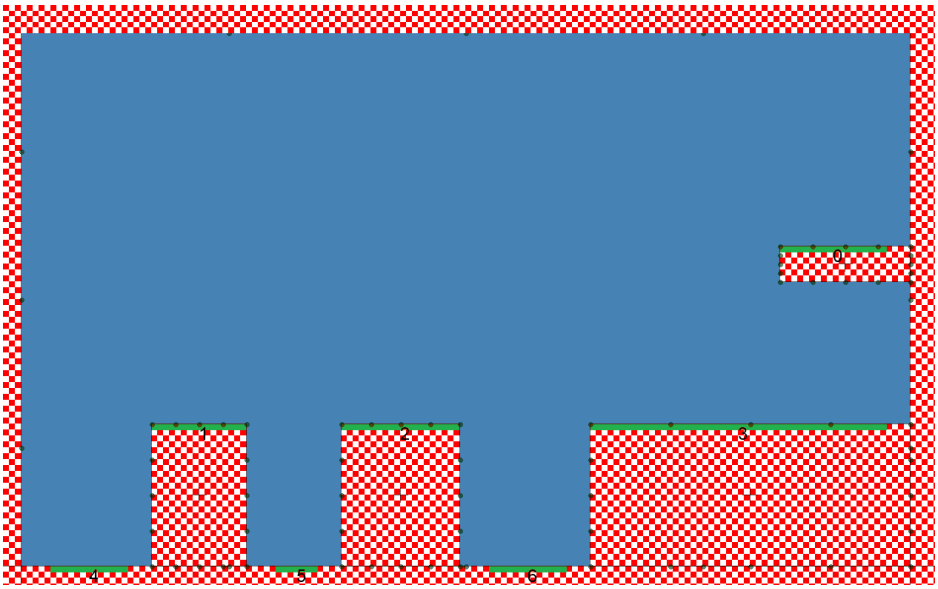


Figura 3.4: Discretización de los obstáculos del nivel 7 de la competición del círculo de 2016.

En la figura 3.3, se pueden observar los obstáculos de un nivel, mientras que la figura 3.4 es la representación interna discretizada. Hacemos notar que los obstáculos aparecen representados en colores rojo y blanco, donde cada cuadrado es nuestra unidad de discretización y las plataformas son las líneas horizontales en color verde.

5. Método *GenerateMoveInformation*: este método es el encargado de encontrar los movimientos, es decir, las trayectorias que permiten conectar plataformas diferentes y capturar diamantes. Para ello, simula cada posible trayectoria desde cada plataforma. Advertimos que no se deben confundir *movimientos* con

acciones. Las acciones son atómicas y se toman en cada llamada al método *Update*. Recordamos que para el círculo son ROLL_RIGHT, ROLL_LEFT, JUMP y NO_ACTION, mientras que para el rectángulo son MOVE_RIGHT, MOVE_LEFT, MORPH_UP, MORPH_DOWN y NO_ACTION. Sin embargo, los movimientos son de más alto nivel y representan conectividad entre plataformas o posibilidad de alcanzar algún diamante. Como veremos, para completar un movimiento, es necesario tomar multitud de acciones atómicas.

Los movimientos vienen determinados por la posición del agente dentro de la plataforma, su velocidad horizontal y el tipo de movimiento. Al generar cada movimiento, se calcula la plataforma sobre la que aterriza, así como los diamantes que el agente cogería por el camino. Lógicamente, las características propias del rectángulo y el círculo hacen que los tipos de movimientos que puedan realizar sean distintos. Por ejemplo, ambos pueden dejarse caer con cierta velocidad por los extremos de las plataformas, pero solo el círculo puede saltar y solo el rectángulo puede hacerse más estrecho para caer por el hueco entre dos plataformas. De entre todos los movimientos que simulamos, solamente guardamos una selección de ellos, que contiene aquellos más interesantes de cara a una posible planificación. Detallaremos más esta selección y daremos más información acerca de los movimientos en sí en los capítulos 4 y 5, correspondientes a cada uno de los tipos de agentes.

Una vez retorna la función *CreateLevelMap*, se ha obtenido una representación discretizada del nivel, una lista de todas las plataformas encontradas, y una lista de movimientos relevantes. Hemos decidido realizar esta modelización de los niveles por lo aprendido al revisar toda la bibliografía del estado del arte. En rasgos generales, la discretización, la representación de plataformas y el establecimiento de conexiones entre ellas ha dado buenos resultados en el pasado. En lo que nos detendremos y pondremos especial énfasis en los capítulos del círculo y el rectángulo es en la generación de movimientos y en su filtrado. Gran parte del desempeño de los agentes recae en que la simulación de los movimientos que hacemos se ajuste de la forma más precisa posible a la realidad y en que seamos capaces de establecer reglas generales para saber cuándo un movimiento es *mejor* (en algún sentido) que otro.

3.3.2. Trazado de un plan a seguir

El diseño del plan a seguir se realiza una vez generadas las plataformas y los movimientos que se pueden realizar desde ellas, es decir, cuando se ha completado la representación inicial. Generamos un grafo con los movimientos que habíamos almacenado, donde los vértices representan nuestras plataformas y las aristas son las conexiones entre ellas mediante los movimientos. En la creación del grafo, se le asocia a cada diamante una lista de plataformas desde las cuales existe un movimiento que alcanza ese diamante y aterriza en la misma plataforma.

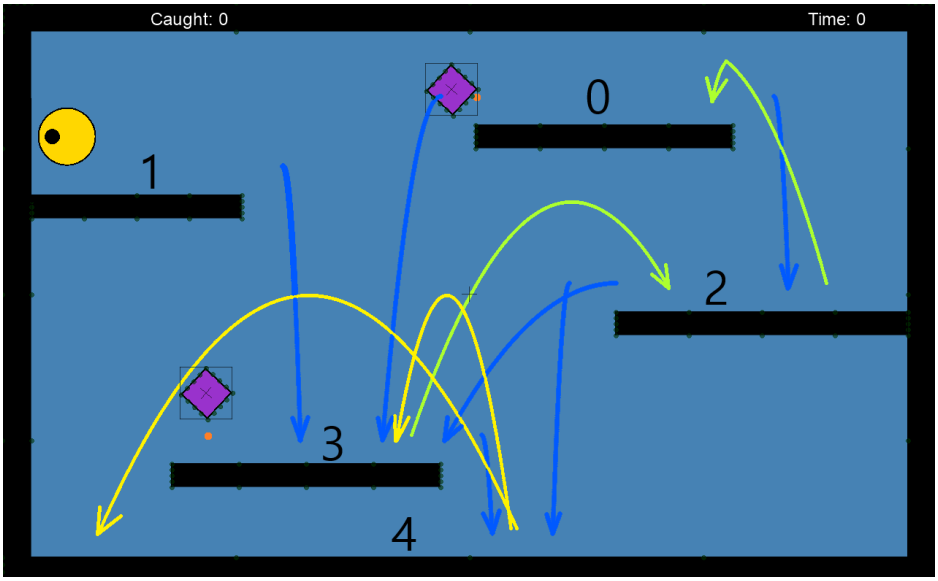


Figura 3.5: Representación de plataformas y movimientos del nivel 5 de la competición del círculo de 2014.

Para ilustrar lo que comentamos, recomendamos analizar la figura 3.5. En ella podemos observar el nivel 5 de la competición del círculo del año 2014, en el que ya están representados los movimientos que permiten cambiar de plataforma y alcanzar diamantes. Aunque habrá que esperar hasta la sección 4.2.1 para introducirlos formalmente, adelantamos que los movimientos del círculo son caer, saltar y mantenerse quieto para alcanzar un diamante. En la figura 3.5 los movimientos de salto están representados por las flechas verdes y amarillas, las caídas por las azules y los movimientos que mantienen al círculo en reposo, alcanzando un diamante, por los puntos naranjas. Nótese que ya se ha realizado el filtrado y no existen dos movimientos que partan de la misma plataforma y aterricen en la misma plataforma. Por ejemplo, a la derecha de la plataforma 3 hay una flecha azul que representa que el círculo puede caer desde esta a la plataforma 4 por el borde derecho. Sin embargo, no aparece la caída simétrica por el extremo izquierdo de la plataforma 3 porque sería redundante. Este modelo es el que se tiene al finalizar la fase de representación del nivel y con él podemos construir el grafo asociado de la figura 3.6.

Como se puede apreciar, las plataformas se han convertido en los vértices del grafo y los movimientos, es decir, la conectividad entre plataformas, en las aristas. A su vez, las aristas tienen la información de los diamantes que alcanzan. Por ejemplo, los puntos naranjas se han transformado en auto-aristas naranjas, que llegan a la misma plataforma de la que parten y alcanzan el diamante. Algo similar sucede con el salto amarillo que sale y aterriza en la plataforma 4 y alcanza un diamante. Por último, destacamos que el vértice que representa la plataforma en la que se encuentra el círculo está apuntado por una flecha y es desde donde comienza el plan. Es sobre este grafo simplificado sobre el que se lanza la búsqueda que da como resultado el plan de actuación. El plan consiste, fundamentalmente, en una lista de movimientos que permiten cambiar de plataforma y está generado de tal manera

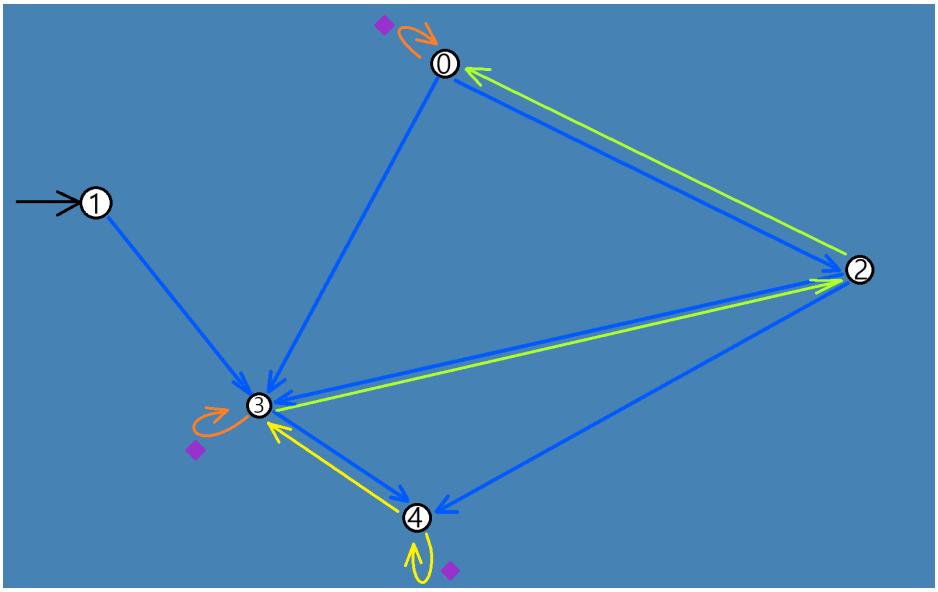


Figura 3.6: Grafo asociado al nivel 5 de la competición del círculo de 2014.

que si el agente, antes de realizar cada uno de los movimientos para cambiar de plataforma, se asegura de haber alcanzado todos los diamantes que se encuentran sobre su plataforma actual entonces el agente alcanza todos los diamantes.

La representación interna de nuestro grafo posee estructuras de tipo nodo. Los nodos, que se usan para el algoritmo de búsqueda, contienen la lista de movimientos (cambios de plataforma) que se han tomado desde la plataforma de partida, es decir, el plan hasta el momento, una lista de booleanos que representan si se ha cogido cada diamante concreto, el número de diamantes cogidos, y si el nodo es arriesgado (en unos párrafos veremos exactamente qué significa esto).

Tras construir el grafo, se llama al algoritmo de búsqueda, que aparece detallado en el algoritmo 1. Se trata de una búsqueda en anchura con control de repetidos y límite de tiempo, que recorre el grafo y encuentra la mejor solución posible. Hay que tener en cuenta que por "mejor solución", entendemos aquel plan que alcanza todos los diamantes efectuando un menor número de movimientos para cambiar de plataforma, aunque también se toman en consideración conceptos como la dificultad de los movimientos (nodos arriesgados). Asimismo, hacemos notar que la solución que menos cambia de plataforma no tiene por qué ser la óptima en tiempo. No obstante, como podremos comprobar, los resultados obtenidos hacen que esta simplificación sea suficientemente buena y no sea necesario realizar un enfoque más complejo, que implicaría también al módulo de ejecución.

Este algoritmo se llama al principio de cada nivel, pero también puede volver a ser llamado si un movimiento no se ha efectuado correctamente y el agente ha aterrizado en una plataforma diferente de la esperada. Esto es lo que denominamos replanificación, y lo detallaremos más en la siguiente sección.

Algoritmo 1 Algoritmo de búsqueda para la generación de un plan

```
1: queue ← Cola con nodo raíz
2: hay_plan_completo ← false
3: mejor_sol ← Nodo vacío con plan vacío
4: nodos_sin_riesgo ← 1
5: vistos ← Conjunto vacío
6: while queue es no vacía && tiempo de búsqueda <T do
7:     node ← queue.pop
8:     if node no es arriesgado then
9:         nodos_sin_riesgo − −
10:    end if
11:    move ← Último movimiento del plan parcial en nodo
12:    p ← plataforma de aterrizaje de move
13:    Procesar move y p
14:    if node ya se ha visitado then
15:        Continuar
16:    else
17:        Añadir node a vistos
18:        if node es mejor que mejor_sol then
19:            mejor_sol ← node
20:        end if
21:        if Plan de node coge todos los diamantes then
22:            hay_plan_completo ← true
23:            if Plan de node no es arriesgado || nodos_sin_riesgo == 0 then
24:                Devolver plan de node
25:            end if
26:        else
27:            foreach Movimiento m que salga desde p y cambie de plataforma do
28:                Crear nuevo_nodo expandiendo node con m
29:                Actualizar nodos_sin_riesgo
30:                Introducir nuevo_nodo en queue
31:            end foreach
32:            if nodos_sin_riesgo == 0 && hay_plan_completo then
33:                Devolver plan de mejor_sol
34:            end if
35:        end if
36:    end if
37: end while
38: Devolver plan de mejor_sol
```

Durante el algoritmo, surgen una serie de preguntas y cuestiones que respondemos a continuación:

- *¿Qué significa que un nodo sea arriesgado?*

Un nodo puede llamarse arriesgado debido a alguno de estos dos factores:

1. Alguno de los movimientos que contiene su plan es arriesgado. Como veremos, existen algunos movimientos muy complejos que preferiremos evitar en la medida de lo posible, aún si eso conlleva seguir un plan más largo (el primer ejemplo se verá en la sección 5.2.2). Esta clase de movimientos los llamaremos arriesgados, y los identificaremos mediante un bit $risky$.

2. Alguno de los movimientos que contiene su plan coincide con el último movimiento efectuado que ha fallado. Esto puede suceder si se ha llamado al algoritmo como motivo de una replanificación. En este caso, el algoritmo recibe el movimiento que ha fallado. Si alguno de los movimientos del plan del nodo coincide con este movimiento, potencialmente puede volver a fallar, por lo que intentaremos buscar otro camino (aunque sea más largo) con la esperanza de que tenga mejores resultados.

- *¿Qué significa que un nodo sea mejor que otro?*

Un nodo se considera mejor que otro si bien coge más diamantes (independientemente de cuáles sean) o bien coge los mismos, pero este nodo no es arriesgado y el otro sí. Esta manera de comparar es útil para asegurar que priorizamos la solución que coja el mayor número de diamantes posible, pero en caso de haber varias, preferimos quedarnos con una solución que no sea arriesgada.

- *¿Cómo se procesan una plataforma y un movimiento?*

Con procesar un movimiento, nos referimos a marcar que el nodo coge todos los diamantes que se alcanzan mientras se ejecuta el movimiento. Esto es, por ejemplo, que si procesamos un movimiento del círculo de tipo salto que mueve el círculo de una plataforma a otra y, durante la trayectoria del salto se alcanza un diamante, se añade el diamante como capturado al nodo (ver tipos de movimiento del círculo en la sección 4.2.1). Con procesar una plataforma, queremos decir que marcamos que el nodo coge todos los diamantes que se pueden alcanzar desde esa plataforma sin necesidad de abandonarla. Para esto es para lo que, durante la creación del grafo, hemos asociado a cada diamante desde qué plataformas existen movimientos que cogen el diamante sin abandonarla. Esta manera de procesar movimientos y plataformas garantiza la propiedad esencial de nuestro plan que es la siguiente: si antes de tomar el siguiente movimiento del plan (que cambia de plataforma), el agente se asegura de haber alcanzado todos los diamantes disponibles asociados a su plataforma actual, entonces el agente alcanzará todos los diamantes.

- *¿Cómo se expande un nodo?*

Si tenemos un nodo n, para cada movimiento m que parte desde la plataforma de llegada del último movimiento del plan de n, podemos expandir n a través de m. Para ello, consideramos un nuevo nodo cuyo plan sea el de n yuxtapuesto a m. Así,

podremos coger diamantes durante el movimiento y alcanzar una nueva plataforma desde la que conseguir más diamantes. Expandir un nodo con un movimiento se refiere a recorrer una de las aristas del grafo (la asociada al movimiento) para llegar a un nuevo vértice.

Además del nuevo plan, debemos calcular también si el nuevo nodo es arriesgado, para lo que evaluaremos si n ya lo era o si m es un movimiento arriesgado, de acuerdo a alguno de los dos criterios que ya hemos comentado.

- *¿Para qué sirven las variables nodos_sin_riesgo y hay_plan_completo?*

La variable *nodos_sin_riesgo* lleva la cuenta de los nodos que quedan en la cola que no son arriesgados. Hacemos notar que, si encontramos una solución completa (es decir, que capture todos los diamantes), no siempre vamos a querer terminar la búsqueda. Esto puede deberse a que la solución que hayamos encontrado es arriesgada, y aún quede la posibilidad de encontrar otra solución que no lo sea. Sin embargo, si al encontrar la solución arriesgada sabemos que todos los nodos que quedan en la cola son arriesgados, sabemos que la solución que tenemos es la mejor posible, por lo que podemos terminar la búsqueda con seguridad (esto se ve en las líneas 23-25 del algoritmo 1). Lo mismo sucede si al expandir un nodo que no era arriesgado solo nos quedamos con nodos arriesgados. Si ya hemos encontrado una solución completa (que será arriesgada, ya que en caso contrario se habría devuelto al encontrarla), sabemos que será la solución que buscamos (líneas 32-34 del algoritmo 1).

La variable *hay_plan_completo* guarda si la mejor solución encontrada hasta el momento captura todos los diamantes. Además de la utilidad mencionada en el párrafo anterior, esta variable cobrará más importancia durante el preprocesamiento de los niveles cuando hablemos del rectángulo que hemos desarrollado, concretamente, en la sección 5.3.

- *¿Cómo se lleva a cabo el control de repetidos?*

Para realizar el control de repetidos, utilizamos el conjunto *vistos* (clase `HashSet` de C#), en el que introducimos una tupla que contiene la plataforma en la que se encontraría el agente después de realizar los movimientos del plan (es decir, la plataforma de llegada del último movimiento) y un entero que identifica el conjunto de diamantes obtenido. Este entero se calcula mediante una función de valor, que suma 2^i si el nodo captura el diamante i. Esto permite comparar dos nodos de manera sencilla, y limita enormemente la cantidad de elementos que puede haber en el conjunto, ya que, entre otras cosas, siempre que se esté en una plataforma, se habrán cogido los diamantes que se puedan alcanzar desde ella.

Sin embargo, hay que tener en cuenta una pequeña consideración adicional. Queremos ser capaces de distinguir nodos arriesgados de nodos no arriesgados, y, si utilizáramos la función de valor tal y como la hemos descrito en el párrafo anterior, no seríamos capaces de hacerlo. Para ello, basta hacer que si el nodo es arriesgado se

devuelva el opuesto del valor que tendría el nodo si no lo fuera. Es decir, el valor de un nodo arriesgado es no positivo (puede ser 0), y el de un nodo no arriesgado es no negativo. Por supuesto, esta función de valor no es la que se utiliza para determinar si un nodo es mejor que otro, sino que se hace según lo explicado en uno de los párrafos anteriores.

- *¿Cuánto tiempo empleamos para la búsqueda?*

Si bien el algoritmo que hemos presentado es completo, en el sentido de que encuentra una solución si la hay, y contamos con un control de repetidos poco costoso para mejorar el tiempo de cálculo, es conveniente tener un límite de tiempo en la búsqueda. Es cierto que antes de comenzar el nivel podemos utilizar todo el tiempo que queramos para planificar, pero las replanificaciones se realizan en tiempo de ejecución y no sería sensato adentrarnos en una búsqueda que puede ser potencialmente larga si el nivel es muy complejo y tiene muchas plataformas. Por ello, al comenzar la búsqueda iniciamos un cronómetro, y al tratar cada nodo de la cola comprobamos si ya hemos superado el tiempo límite de búsqueda. Hemos establecido el tiempo límite de la búsqueda en las replanificaciones a 400 ms. En la práctica este es un tiempo más que suficiente para encontrar un plan completo. No obstante, si el tiempo expirara antes de conseguirlo, se devolvería un plan parcial. Tras ejecutar dicho plan parcial se realizaría otra replanificación y, si hay suerte, la composición de esos dos planes parciales, aunque no sea óptima, puede llevar al agente a completar el nivel.

Vamos a mostrar el resultado de aplicar el algoritmo de búsqueda anterior al grafo de la figura 3.6. El plan que obtenemos es el dado por la figura 3.7, es decir, $1 \rightarrow 3 \rightarrow 2 \rightarrow 0$ y su interpretación es la siguiente:

1. Coger todos los diamantes alcanzables desde la plataforma 1 (no hay).

2. Moverse desde la plataforma 1 a la plataforma 3 con el movimiento azul (la caída).

3. Coger todos los diamantes alcanzables desde la plataforma 1 (se alcanza el diamante mediante el movimiento naranja).

4. Moverse desde la plataforma 3 a la plataforma 2 con el movimiento verde (el salto).

5. Coger todos los diamantes alcanzables desde la plataforma 2 (no hay).

6. Moverse desde la plataforma 2 a la plataforma 0 con el movimiento verde (el salto).

7. Coger todos los diamantes alcanzables desde la plataforma 0 (se alcanza el diamante mediante el movimiento naranja).

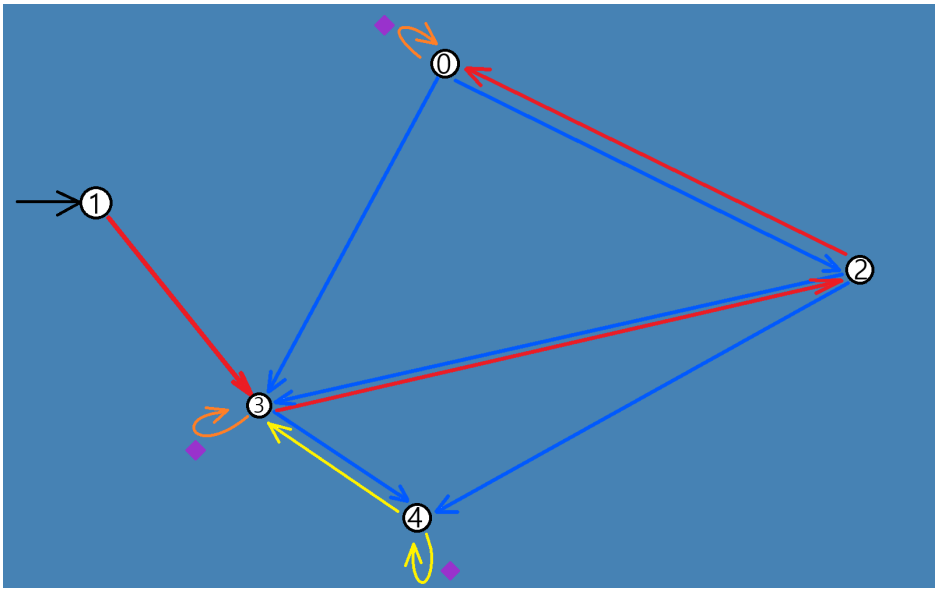


Figura 3.7: Plan asociado al nivel 5 de la competición del círculo de 2014.

Destacamos que las plataformas únicamente se abandonan cuando ya se han conseguido todos los diamantes alcanzables desde esa plataforma con movimientos que aterrizan en esa misma plataforma. Asimismo, los diamantes alcanzables desde las plataformas 3 y 4 son en realidad el mismo, con lo que, en la línea 13 del algoritmo 1, al procesar la plataforma de aterrizaje de la caída desde la plataforma 1 a la 3, se habrá marcado el diamante como capturado. Por tanto, no será necesario en el plan visitar la plataforma 4 porque ese diamante ya se habrá alcanzado desde la plataforma 3.

Esto concluye la explicación del trazado de un plan a seguir. En la siguiente sección, detallamos cómo se sigue el plan que se ha retornado como resultado de la llamada al algoritmo 1.

3.3.3. Ejecución del plan

La principal diferencia que presenta la *Ejecución del plan* frente a los pasos de *Representación del nivel* y *Trazado de un plan a seguir* es el tiempo. Mientras que los dos primeros se realizan antes de que se inicie el nivel, y por tanto no tienen límite de tiempo, la ejecución del plan se debe hacer en tiempo real, por lo que es un factor a tener en cuenta a la hora de realizar algoritmos complejos en esta parte.

Aunque la ejecución concreta del plan es muy diferente para los dos agentes, ya que depende en gran medida del tipo de movimiento que se vaya a ejecutar, sí que posee ciertas similitudes. El algoritmo general se muestra en el algoritmo 2. En ambos casos se parte del plan que ha sido generado en la fase anterior y que está compuesto por una lista de movimientos que permiten pasar de una plataforma a

Algoritmo 2 Algoritmo de actuación general

```
1: while El juego no ha terminado do
2:     if Plataforma actual != Plataforma origen del primer paso del plan then
3:         Replanificar
4:     end if
5:     if Quedan diamantes por coger en la plataforma actual then
6:         m ← Movimiento que alcanza el diamante más cercano
7:         Ejecutar mejor acción que lleva a completar m
8:     else
9:         if El plan es no vacío then
10:            m ← Primer movimiento del plan
11:            Ejecutar mejor acción que lleva a completar m
12:        else
13:            Realizar una acción aleatoria
14:        end if
15:    end if
16: end while
```

otra. Dentro de cada plataforma y antes de pasar a la siguiente, el agente deberá coger todos los diamantes alcanzables desde dicha plataforma, ya que esta era una asunción que realizaba el algoritmo de planificación.

Al igual que en la sección anterior, merece la pena detallar algunos de estos puntos:

- *Replanificación*

Ya adelantamos en la sección anterior en qué consistía la replanificación. Dadas las características del juego, en muchas ocasiones la resolución de los niveles depende en gran medida de un grado de precisión muy alto en la toma de acciones, que resulta en la práctica inalcanzable, debido a las sucesivas discretizaciones, simplificaciones y aproximaciones efectuadas a lo largo de todo el proceso. Este no es un problema exclusivo de los agentes ya que, como se verá en el capítulo de resultados, el juego presenta retos de coordinación que los humanos no podemos resolver siempre de manera óptima. Es por tanto inevitable que se puedan llegar a tomar acciones que provoquen que el agente acabe en un estado indeseado o al menos imprevisto.

En virtud a lo anterior, es oportuno contar con un mecanismo de recuperación en el caso de que algo no salga de la manera esperada. Por lo tanto, antes de realizar una acción se comprueba si ha surgido cualquier tipo de eventualidad, verificando si la plataforma actual es la plataforma origen del primer paso del plan. Si este no es el caso, lanzamos nuevamente el algoritmo de búsqueda sobre el grafo para construir un nuevo plan partiendo de la plataforma actual. En la práctica, estas replanificaciones solamente se ejecutan cuando el agente no ha llegado a la plataforma destino tras tratar de realizar el movimiento que tenía previsto.

Como ya comentamos, a la función de búsqueda del grafo se le proporciona el último movimiento que falló (en caso de que exista) y, en la medida de lo posible, el algoritmo intenta generar un nuevo plan evitando realizar ese movimiento. Esto es porque, si un movimiento falla, es posible que se deba a que es potencialmente complicado de realizar y es conveniente explorar otros caminos, aunque tengan más pasos. Por supuesto, si este es el único movimiento posible, el nuevo plan lo incluirá.

- *Movimientos y acciones a tomar*

Ya mencionamos que los movimientos se referían a objetivos de alto nivel como un cambio de plataforma o la posibilidad de alcanzar algún diamante. Por ejemplo, dejarse caer por un lado de una plataforma es lo que llamaremos un movimiento de tipo FALL, y tanto el círculo como el rectángulo pueden efectuarlo. Todos los movimientos que planifiquemos tendrán un tipo asociado, que identificará cómo debe actuar el agente para completarlo.

Por otro lado, las acciones son atómicas, y se refieren a qué debe hacer el agente en cada instante. El conjunto de acciones posibles ya viene determinado por el juego, y no podemos modificarlo. Para completar un movimiento, necesitaremos realizar una serie de acciones. Cómo elegir la mejor acción en cada momento para lograr ese objetivo es algo que depende de cada agente y tipo de movimiento, por lo que lo especificaremos en los capítulos correspondientes.

- *Acciones aleatorias*

Si el agente no es capaz de encontrar ningún plan y no puede coger más diamantes desde la plataforma en la que se encuentra, quiere decir que o bien el conjunto de movimientos que ha generado es insuficiente para completar el nivel o bien ha fallado un movimiento y se ha quedado en un estado desde el que no puede coger todos los diamantes. En estos casos, los agentes de otros años optaban por no hacer nada, es decir, tomaban siempre la acción NO_ACTION. Sin embargo, nosotros hemos optado por realizar acciones aleatorias, ya que, si bien es probable que no sirva de nada (especialmente en el segundo de los casos anteriores), alguna vez ha conseguido rescatar a un agente de la posición en la que se encontraba, o ha realizado un movimiento que el agente no había sido capaz de encontrar, llevándole a completar el nivel. No es común, pero es una pequeña mejora de rendimiento que consideramos mejor que resignarse y no realizar ninguna acción.

Esto concluye la descripción de la arquitectura de nuestra solución para los agentes individuales que es común al círculo y al rectángulo. En el siguiente capítulo comenzaremos a detallar las particularidades del círculo en cuanto a la representación del nivel y a la ejecución del plan.

Capítulo 4

Desarrollo del círculo

En este capítulo, explicamos todos los detalles y particularidades que hemos tenido en cuenta durante el desarrollo del círculo. En la sección 4.1 daremos una breve introducción sobre generalidades del círculo y citaremos algunas constantes físicas asociadas al círculo. En la sección 4.2 explicamos cómo representamos internamente los niveles. Nos centraremos en la generación de movimientos y el filtrado de los mismos. Por último, en la sección 4.3, comentaremos cómo ejecuta el plan de acción el círculo. La explicación la guiaremos partiendo de un algoritmo en pseudocódigo que desgranaremos en las distintas subsecciones. Adelantamos que hemos desarrollado dos versiones del círculo que se diferencian en la parte de ejecución. Sin embargo, salvo donde queramos realizar esta distinción de manera explícita, hablaremos de "el agente" para referirnos al círculo en cualquiera de sus dos versiones, pues casi todo el desarrollo es idéntico para ambos agentes.

4.1. Introducción

El círculo es uno de los dos personajes del juego Geometry Friends. Es una pelota amarilla de dimensiones constantes cuyo objetivo es alcanzar todos los diamantes de los niveles. Hay niveles en los que el círculo y el rectángulo deben colaborar para alcanzar los diamantes y otros en los que los agentes están solos. En este capítulo nos vamos a centrar en aquellos niveles en los que únicamente participa el círculo.

Las acciones que puede tomar el círculo son distintas a las del rectángulo y estas son ROLL_RIGHT, ROLL_LEFT, JUMP y NO_ACTION. Las dos primeras le permiten rodar hacia la derecha y la izquierda. Internamente, se le aplica una fuerza mientras se elige una de esas acciones, que se traduce en que el círculo sufre una aceleración constante que hace que ruede hacia uno de los lados. La acción JUMP puede ser tomada mientras el círculo reposa sobre una plataforma y provoca que

este salte. Para conseguirlo, se le aplica una fuerza instantánea que hace que el círculo tenga cierta velocidad vertical. La acción NO_ACTION, se muestra como alternativa cuando no se desea tomar ninguna de las acciones anteriores.

Como ya comentamos en capítulos anteriores, el círculo rebota contra los obstáculos cuyo color no es amarillo. Es decir, atraviesa los obstáculos amarillos y colisiona contra los verdes y los negros.

Antes de comenzar a desarrollar el círculo, tuvimos que calcular experimentalmente los siguientes datos, que sabíamos que nos serían imprescindibles para poder realizar una modelización adecuada del problema:

- El radio del círculo.
- La aceleración que experimenta al aplicarle una acción de giro.
- La velocidad inicial de salto.
- La aceleración de la gravedad.
- La velocidad máxima horizontal que puede alcanzar.

Además, comprobamos que la fuerza de rozamiento tenía un efecto despreciable sobre la velocidad, por lo que la descartamos de todos nuestros cálculos.

Dado que la representación del estado del entorno del juego quedó detallada en el capítulo anterior, el primer aspecto en el que se diferencian los dos agentes es en la generación de movimientos. En las siguientes secciones también trataremos la estrategia del círculo para el filtrado de movimientos y la ejecución del plan.

4.2. Representación del nivel

Ahora que tenemos una idea general de las acciones que puede realizar el círculo, en esta sección nos detendremos a explicar las particularidades del círculo en cuanto a la representación interna de los niveles. Principalmente nos centraremos en la generación de los movimientos del círculo, que son distintos a los del rectángulo, y en el filtrado de los mismos. Como veremos, es necesario hacer una selección de aquellos que sean más interesantes para no tener duplicidades y reducir el coste computacional de la búsqueda.

4.2.1. Generación de movimientos

Como anticipamos en la sección 3.3.1, los movimientos que puede realizar el círculo son distintos a los que puede realizar el rectángulo. Es conveniente recordar que movimientos y acciones son conceptos distintos en nuestra implementación. Las acciones que puede tomar el círculo son ROLL_RIGHT, ROLL_LEFT, JUMP y NO_ACTION y estas son atómicas. Sin embargo, los movimientos son trayectorias completas que permiten coger algún diamante o cambiar de plataforma. Un movimiento está determinado por su velocidad horizontal, la posición en la plataforma de partida y el tipo de movimiento. En las siguientes secciones detallaremos cada uno de los posibles movimientos del círculo: NOMOVE, JUMPMOVE y FALL. A grandes rasgos, el movimiento NOMOVE permite al círculo coger un diamante que se encuentra a la altura de la plataforma, el movimiento JUMPMOVE (no confundir con la acción JUMP) representa los saltos del círculo partiendo desde una determinada posición y con cierta velocidad horizontal y el tipo de movimiento FALL hace referencia a las caídas del círculo por los extremos de las plataformas. Estos tres tipos de movimientos son todos los que permiten al círculo alcanzar un diamante o cambiar de plataforma.

Antes de explicar en más detalle cada tipo de movimiento, es conveniente comentar algún detalle sobre la simulación de los mismos. Como veremos, se basarán en evaluar en cada punto de la trayectoria las colisiones que tiene una discretización de nuestro círculo con diamantes u obstáculos.

Hacemos notar que contamos con dos discretizaciones del círculo. La que aparece en la figura 4.1 es una discretización que está estrictamente contenida en el círculo real y es la que utilizamos para comprobar si el círculo captura algún diamante. Esto permite, junto a la discretización estrictamente interior de los diamantes (figura 3.2), que cuando se produce una intersección entre las discretizaciones tengamos garantizado que el diamante se va a alcanzar. La otra discretización, la de la figura 4.2, es la que utilizamos para las colisiones con los obstáculos. Se puede apreciar que es una discretización que contiene estrictamente al círculo real. Con ello nos aseguramos de que, si no hay intersección entre la discretización del círculo y la del obstáculo, en la realidad el círculo no colisionará con el obstáculo. A partir de ahora, cuando digamos que "comprobamos si la discretización del círculo interseca con algún diamante" siempre nos referiremos a la discretización de la figura 4.1 y cuando digamos que "comprobamos si la discretización del círculo colisiona con algún obstáculo" siempre nos referiremos a la discretización de la figura 4.2.

Movimientos NOMOVE

Los movimientos NOMOVE representan la situación en la que el círculo está en reposo sobre una plataforma. Estos movimientos sirven para que el círculo alcance los diamantes que se encuentran a la altura de la plataforma sin la necesidad de

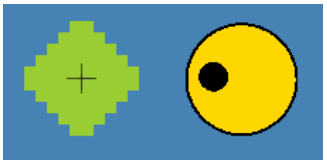


Figura 4.1: Discretización interior del círculo (izquierda) para las colisiones con diamantes frente al círculo completo (derecha).

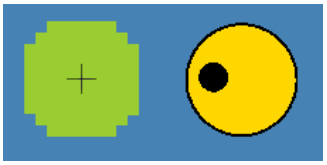


Figura 4.2: Discretización exterior del círculo (izquierda) para las colisiones con obstáculos frente al círculo completo (derecha).

saltar o de caer. Por tanto, en la función *GenerateMoveInformation*, evaluamos en cada punto de cada plataforma si la discretización del círculo (ver figura 4.1) situada en dicho punto interseca con la discretización de alguno de los diamantes (ver figura 3.2). De estos movimientos, solo guardamos aquellos donde esta intersección es no vacía.

Movimientos JUMPMOVE

Los movimientos JUMPMOVE están asociados a la acción de salto del círculo. Al seleccionar dicha acción, al círculo se le aplica una fuerza instantánea vertical, que le otorga una velocidad vertical v_y. A partir de entonces, asumimos que el círculo se comporta como una masa puntual y que el rozamiento es despreciable, por lo que la posición de su centro a tiempo $\mathtt{t}$ viene dada por las ecuaciones del movimiento (tiro parabólico), ya que nos encontramos en un entorno físico con gravedad. Estas ecuaciones son las correspondientes a un movimiento uniforme en el eje de abscisas y un movimiento uniformemente acelerado en el de ordenadas:

$$\begin{cases} x(\mathtt{t}) = x_0 + v_x\mathtt{t} \\ y(\mathtt{t}) = y_0 + v_y\mathtt{t} - \frac{g\mathtt{t}^2}{2} \end{cases}$$

donde (x_0, y_0) es la posición inicial del centro del círculo, v_x es la velocidad horizontal inicial y g es la aceleración de la gravedad. Tanto el valor de v_y como el de g son constantes. Además, para una plataforma concreta, y_0, es decir, la altura de la plataforma más el radio del círculo, es un valor fijo y $x_i \leq x_0 \leq x_d$ siendo x_i y x_d la posición de los extremos izquierdo y derecho de la plataforma, respectivamente. Por otro lado, $|v_x| \leq v_{max}$ para cierta constante positiva v_{max} que indica la velocidad horizontal máxima del agente. Por tanto, tenemos dos variables libres, x_0 y v_x, que se mueven en un cierto intervalo que nos determinan la trayectoria del agente en los saltos desde una plataforma concreta.

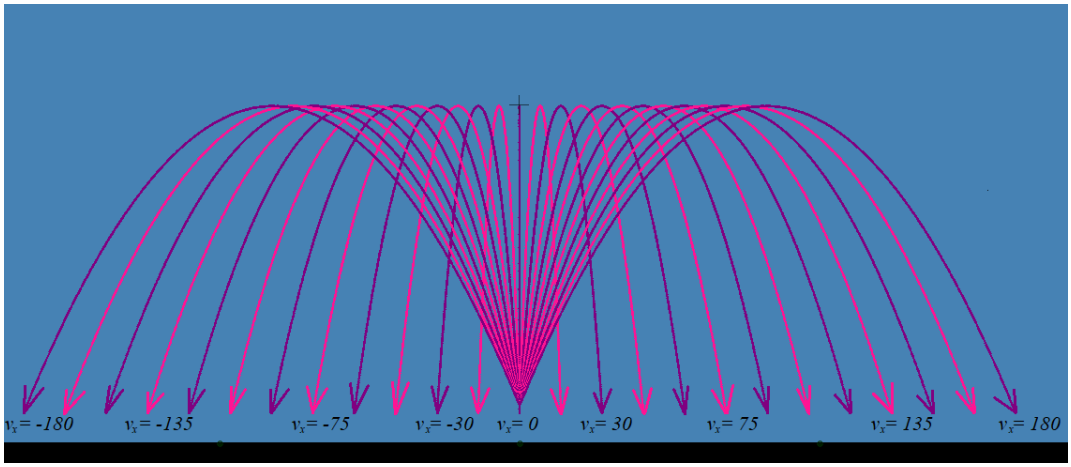


Figura 4.3: Parábolas consideradas desde un punto con velocidades horizontales variables $v_x = 15i$, con $i = 0, \pm1, \pm2, ..., \pm12$.

Con esta información, podemos simular los saltos del círculo que parten de un determinado punto x_0 de nuestra discretización con una velocidad horizontal inicial v_x. Como el rango de velocidades horizontales también es continuo, tomamos una discretización del intervalo $[-v_{max}, v_{max}]$ con paso fijo. En la figura 4.3 podemos ver todas las trayectorias que se consideran desde uno de los puntos de la plataforma inferior.

Una vez tenemos clara la idea general de los saltos del círculo, vamos a explicar cómo tratamos las colisiones con los obstáculos, así como otros aspectos relevantes de las trayectorias. El proceso de simulación de las trayectorias consiste, una vez más, en realizar una discretización, esta vez en el tiempo. Se consideran los tiempos $\mathtt{t}_n = n\Delta\mathtt{t}$ y se sitúa la discretización del círculo (su centro) en la posición $(x(\mathtt{t}_n), y(\mathtt{t}_n))$, donde el incremento de tiempo $\Delta\mathtt{t}$ lo hemos tomado como 5 ms. Esta elección en el incremento de tiempo permite que $||(x(\mathtt{t}_n),\, y(\mathtt{t}_n)) - (x(\mathtt{t}_{n+1}),\, y(\mathtt{t}_{n+1}))|| < 8$ píxeles, es decir, que dos puntos consecutivos de la trayectoria se encuentran a una distancia menor que un cuadrado de nuestra discretización, con lo que garantizamos que la trayectoria sea de algún modo continua.

Cuando la discretización del círculo se encuentra en esa posición, comprobamos si colisiona con las discretizaciones de algún diamante o algún obstáculo. En el primero de los casos, se añade el diamante al conjunto de diamantes que se capturan en este movimiento, mientras que en el segundo se evalúa el punto de contacto entre el círculo y el obstáculo.

Distinguimos 5 tipos diferentes de colisiones con obstáculos (aunque dos de ellos son análogos), según el punto de contacto, como se ve en la figura 4.4:

- Si la colisión se produce con la parte inferior (rojo en la figura), significa que el círculo ha aterrizado en una plataforma, por lo que finaliza el movimiento y se identifica la plataforma de llegada.

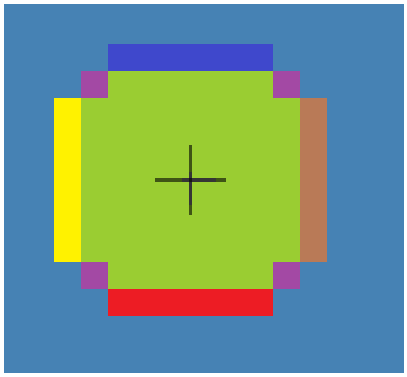


Figura 4.4: Tipos de colisiones con obstáculos de la discretización del círculo.

- Si ha colisionado con uno de los laterales (amarillo o marrón en la figura), entonces se produce un rebote contra una pared. Como las colisiones son inelásticas, hemos estimado que las nuevas velocidades del círculo en ambos ejes se reducen por un factor de 1/3 y la velocidad horizontal cambia de signo. Una vez calculadas las nuevas velocidades, se continúa simulando el movimiento hasta que se aterriza en una plataforma.

- Cuando la colisión se produce con la parte superior del círculo (azul oscuro en la figura), el planteamiento es similar al del caso anterior. Esta situación representa que el círculo colisiona con un techo, y hemos estimado que las velocidades se reducen en un factor de 1/3. Además, esta vez la velocidad vertical cambia de signo.

- Por último, quedan las colisiones que se producen en las esquinas del círculo, representadas con el color morado en la figura. Estas colisiones son algo más conflictivas, y es difícil predecir su evolución, ya que reproducen una situación en la que el círculo impacta con la esquina de alguno de los obstáculos. Como contamos con un número más que suficiente de parábolas, decidimos descartar los movimientos con este tipo de colisiones.

Movimientos FALL

Los movimientos FALL modelizan las caídas del círculo por los extremos de una plataforma. El procedimiento por el cual se simulan las caídas es exactamente igual que el explicado anteriormente para los saltos, con la salvedad de que la velocidad vertical inicial es 0. Asimismo, no se simulan las parábolas desde todos los puntos de la plataforma, sino que solamente consideramos los extremos de las mismas, como se puede apreciar en la figura 4.5.

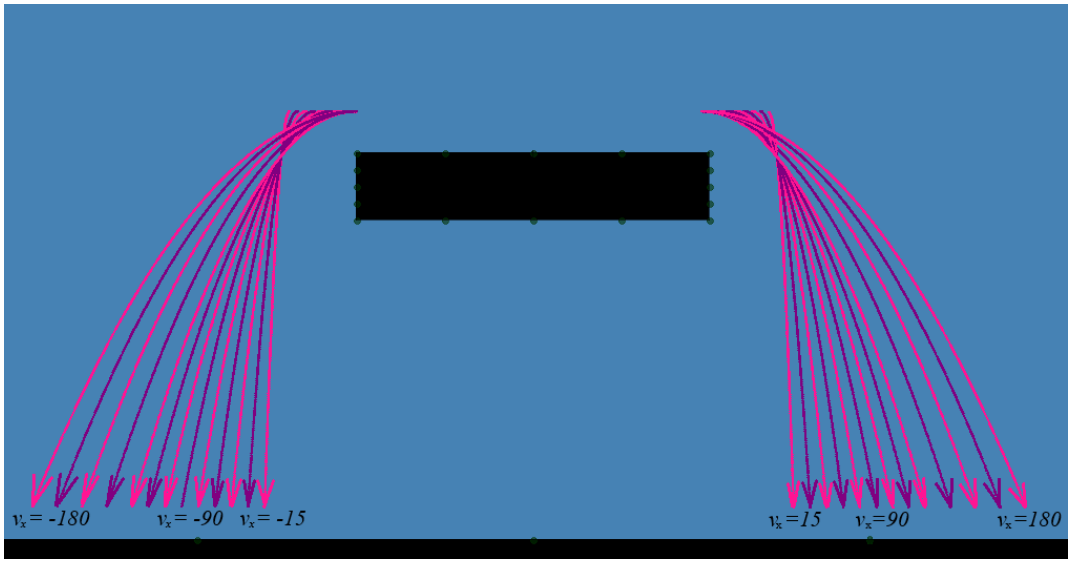


Figura 4.5: Caídas simuladas desde los extremos de una plataforma.

Comentarios adicionales sobre la generación de movimientos

Llegados a este punto, nos encontramos con la problemática de que el número de parábolas a analizar es considerablemente alto. Nótese que si el número de plataformas es N_p, la anchura media de una plataforma es L puntos y discretizamos el número de velocidades en N_v, el número de caídas y saltos a tratar es del orden de $N_p \times (L+1) \times N_v$. Para paliar este inconveniente, decidimos paralelizar los cálculos para obtener los movimientos más rápidamente.

También ha sido útil no tratar algunos movimientos que son imposibles de conseguir por las limitaciones de las plataformas o que podrían ser complicados. En relación a lo primero, debemos eliminar las parábolas que requieran de una velocidad horizontal inicial que es imposible de alcanzar en ese punto de la plataforma. Para lo último, nos es útil calcular durante la simulación el mínimo de las distancias de la trayectoria a los obstáculos. En general, queremos que esta distancia sea lo mayor posible, ya que así evitaremos que el círculo choque con obstáculos por potenciales problemas de aproximación o por márgenes de error al tomar estas parábolas. Aun así, es posible que no haya ningún camino alternativo, en cuyo caso nos quedamos con ellos.

Recordamos que las acciones del círculo en una determinada plataforma son: rodar hacia la izquierda, rodar hacia la derecha, o no realizar ningún movimiento. Las dos primeras aplican una fuerza constante al círculo, que provoca que este se mueva según las ecuaciones del movimiento uniformemente acelerado. De esta forma, se puede calcular el espacio necesario para, partiendo en reposo desde el origen, alcanzar una velocidad v_x dada. Para ello, despejamos la distancia d de la Ecuación 4.1,

correspondiente a un movimiento uniformemente acelerado:[1]

$$v_f^2 = v_0^2 + 2ad \tag{4.1}$$

Donde v_f es la velocidad final, v_0 es la velocidad inicial, a es la aceleración (con signo), y d es el desplazamiento, es decir, la diferencia entre la posición final y la posición inicial. Puesto que la velocidad inicial en este caso es nula, deducimos que, para llegar a un determinado punto con velocidad v_x, se necesita como mínimo un espacio de $\left|\frac{v_x^2}{2a}\right|$ para acelerar, en el sentido apropiado según el signo de la velocidad. Para evaluar si es factible realizar determinados saltos o caídas, simplemente hay que comprobar que hay suficiente espacio en la plataforma para alcanzar la velocidad deseada.

4.2.2. Filtrado de movimientos

Pasamos ahora a explicar el filtrado de los movimientos. Como hemos argumentado anteriormente, se han generado muchos movimientos y, para realizar las búsquedas posteriores, es conveniente quedarnos con los más apropiados. Por tanto, cuando se genera un nuevo movimiento se evalúa si es mejor, si es peor, o si no se puede comparar con los actuales. Los únicos movimientos que nos interesan son aquellos que cogen algún diamante o los que sirven para cambiar de plataforma. Cualquier movimiento que no cumpla ninguna de estas dos condiciones es automáticamente descartado. Si el movimiento pasa este primer filtro, se compara con todos los movimientos actuales. Para que dos movimientos sean comparables, deben partir de la misma plataforma y llegar a la misma plataforma.

En primer lugar, priorizamos que un diamante se alcance con un NOMOVE frente a un JUMPMOVE. Esto se debe a que si el diamante está a la altura de una plataforma, es más rápido llegar a esa posición rodando que saltando (rodando se puede aumentar la velocidad horizontal y en los saltos la velocidad horizontal es constante). Dentro de los NOMOVE que alcanzan el mismo diamante, nos vamos a quedar con aquel que tenga como coordenada x la misma que la del diamante. Nótese que hay movimientos de tipo NOMOVE que alcanzan el diamante sin estar justo en su vertical. Como para alcanzar un diamante sobre una plataforma todos estos movimientos nos sirven y querríamos quedarnos únicamente con uno, es lógico elegir este.

Por otro lado, están los movimientos que cambian de una plataforma a otra, que pueden ser o saltos o caídas. Si el conjunto de diamantes alcanzados en un movimiento contiene estrictamente al conjunto de diamantes que se recogen con otro, es claro que preferimos el primero. Si ambos conjuntos contienen algún diamante que el otro movimiento no coge, los movimientos son incomparables y querremos quedarnos con ambos. Por último, si los diamantes alcanzados son exactamente los mismos, tenemos que tener en consideración otros factores.

[1] La deducción de esta fórmula se puede encontrar en la sección 2.3 de [42].

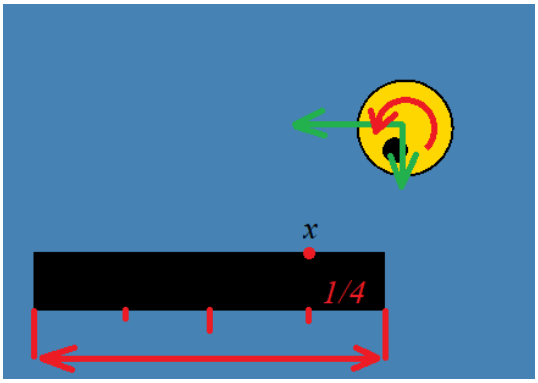


Figura 4.6: Punto idóneo de aterrizaje.

Un problema típico, y que se hace evidente tras analizar algunas partidas en el juego, es caerse por los lados al aterrizar cerca de los extremos de una plataforma. Por tanto, descartamos los movimientos que aterrizan demasiado cerca de los extremos de las plataformas frente a los que no lo hacen. Por último, tomamos en consideración otros cuatro elementos cuya importancia relativa no es clara. Estos son:

- Distancia del punto de aterrizaje al extremo de la plataforma por donde se puede caer el círculo si no hay suficiente espacio para frenar. En general, vamos a querer minimizar la distancia del punto de aterrizaje al punto x de la figura 4.6 (d_1).

- Velocidad horizontal del salto (v_x). Relacionado con el factor anterior: cuanto menor sea la velocidad horizontal, menos espacio se necesitará para frenar.

- Distancia de la trayectoria a los obstáculos (d_2). Como adelantamos, queremos maximizar esta distancia para evitar choques indeseados con obstáculos.

- Distancia del punto de salto al centro de la plataforma de origen (d_3). Aunque idealmente nuestro agente tomará siempre los movimientos que ha planeado, en ocasiones no llegará con la velocidad adecuada para realizar exitosamente el salto. Detallaremos el comportamiento del círculo en estas situaciones más adelante, pero adelantamos que habrá situaciones en las que el agente no saltará. Para evitar que un salto abortado precipite una caída por un extremo de la plataforma de origen, queremos tener suficiente espacio para frenar si esto sucede, justificando así por qué queremos minimizar la distancia anterior.

Tomando lo anterior en consideración, elegiremos el movimiento con menor valor de la combinación lineal

$$Value = d_1/3 + v_x/10 - d_2 + d_3/10.$$

Los pesos se han elegido teniendo en cuenta que las distancias y velocidades tienen escalas distintas.

Algoritmo 3 Algoritmo de actuación del círculo

```
1: while El juego no ha terminado do
2:     if Plataforma actual != Plataforma origen del primer paso del plan then
3:         Replanificar
4:     end if
5:     if Quedan diamantes por coger en la plataforma actual then
6:         Obtener el movimiento m que alcanza el diamante más cercano
7:         if Pos. círculo ≈ Pos. m && Velocidad círculo ≈ Velocidad m. then
8:             Realizar acción asociada al tipo de movimiento m
9:         else
10:            if (Pos. círculo, Velocidad círculo) es igual de válido que m then
11:                Realizar acción asociada al tipo de movimiento m
12:            else
13:                Realizar acción que más acerque al círculo a (pos. m, vel. m)
14:            end if
15:        end if
16:    else
17:        if El plan es no vacío then
18:            Obtener el primer movimiento m del plan
19:            if Pos. círculo ≈ Pos. m && Velocidad círculo ≈ Velocidad m. then
20:                Realizar acción asociada al tipo de movimiento m
21:                Desapilar el primer movimiento del plan
22:            else
23:                if (Pos. círculo, Velocidad círculo) es igual de válido que m then
24:                    Realizar acción asociada al tipo de movimiento m
25:                else
26:                    Realizar acción que más acerque al círculo a (pos. m, vel. m)
27:                end if
28:            end if
29:        else
30:            Realizar una acción aleatoria
31:        end if
32:    end if
33: end while
```

4.3. Ejecución del plan

En esta sección, partimos de un plan de actuación que ha sido generado tras realizar una búsqueda en un grafo donde los vértices son las plataformas y las conexiones, las aristas. Recordamos que el plan se compone de una lista de movimientos entre plataformas (saltos o caídas) donde la plataforma origen del primer salto es la plataforma sobre la que se encuentra el círculo. En rasgos generales, el algoritmo que sigue el círculo es el que se describe en el algoritmo 3.

Analizando el algoritmo, quedan todavía varias incógnitas por resolver. ¿Cómo se realiza la replanificación? ¿Qué significa que la posición del círculo y su velocidad sean aproximadamente las del movimiento? ¿Qué significa que dos movimientos sean igual de válidos? ¿Cómo se obtiene la acción que más acerca al círculo a la posición y velocidad deseadas? A continuación, procedemos a aclarar estas y otras cuestiones que surgen del algoritmo presentado.

4.3.1. Replanificación

En primer lugar, hablaremos sobre la replanificación. Aunque la necesidad de contar con este mecanismo de recuperación se motivó en la sección 3.3.3, queremos detallar en qué situaciones puede ser útil.

En el caso del círculo, la replanificación se aplica especialmente a los saltos, que es el tipo de movimiento más complejo, pero también se puede usar en el caso de las caídas. Como ya dijimos, antes de realizar una acción se comprueba si la plataforma actual es la plataforma origen del primer paso del plan. De no ser así, lanzaríamos nuevamente el algoritmo de búsqueda sobre el grafo para construir un nuevo plan partiendo de la plataforma actual. En la práctica, estas replanificaciones solamente se ejecutan cuando el agente no ha llegado a la plataforma destino tras realizar un salto o en las caídas.

Recordamos que a la función de búsqueda del grafo se le proporciona el último movimiento que falló (en caso de que exista) y, en la medida de lo posible, el algoritmo intenta generar un nuevo plan evitando realizar ese movimiento, aunque, lógicamente, si este es el único movimiento que permite completar el nivel, el nuevo plan lo incluirá. En el caso particular del círculo, dado que no almacenamos demasiadas parábolas, es infrecuente que haya más de dos caminos alternativos para completar el nivel, y más aún que nuestro agente falle en la ejecución de ambos. Por ello, a la hora de replanificar, nos basta considerar el último movimiento mal ejecutado, en lugar de almacenar en una lista el histórico de movimientos que no se han podido completar. Así, en caso de que el agente falle continuamente, en la práctica alternará entre dos caminos diferentes, aunque pueda haber más.

4.3.2. Determinación del siguiente movimiento a realizar

Una vez se ha replanificado, pueden darse dos situaciones. O bien queda un diamante por coger desde la plataforma actual, o bien es necesario cambiar de plataforma. Existen situaciones en las que un mismo diamante puede alcanzarse con un movimiento que no cambia de plataforma y otro que sí cambia de plataforma. En un principio, según la asociación que hacemos en el grafo por la que se determina para cada diamante si es posible alcanzarlo desde alguna plataforma sin necesidad de

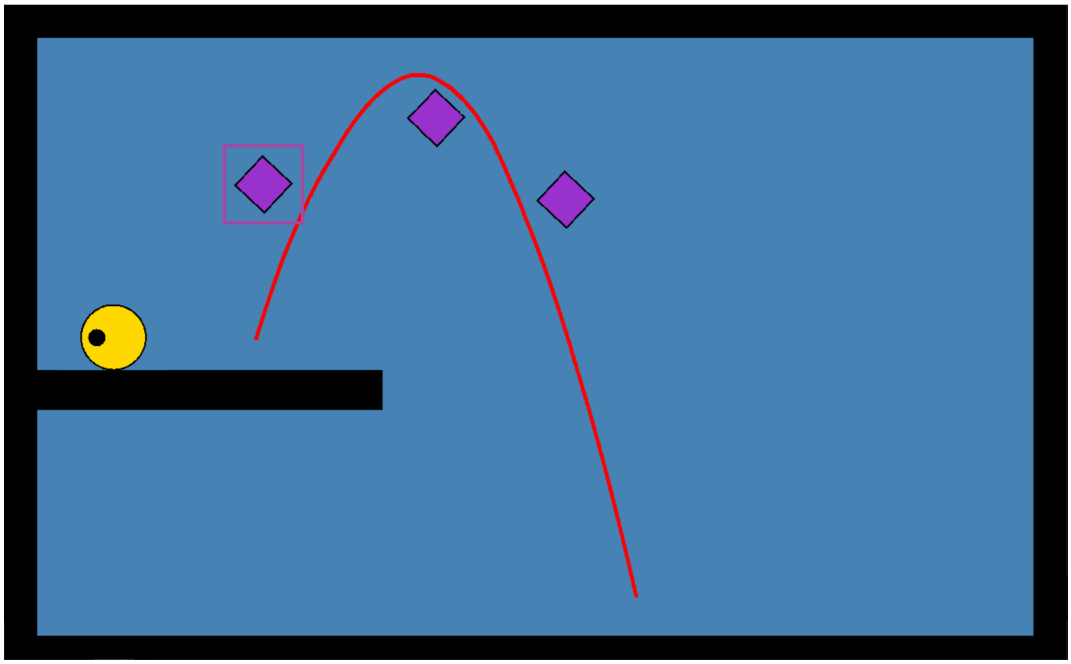


Figura 4.7: Nivel en el que es más eficiente no coger el diamante recuadrado sin cambiar de plataforma.

cambiar a otra, favorecemos los primeros. Sin embargo, existen situaciones en las que es conveniente modificar esta política.

Consideremos por ejemplo el nivel de la figura 4.7. En este nivel, el diamante recuadrado puede ser alcanzado desde la plataforma superior, aterrizando en la misma. No obstante, una vez el círculo está en esa plataforma, su siguiente movimiento en el plan es la parábola roja, que, como vemos, alcanza el mismo diamante (además de otros dos). Por tanto, es más eficiente simplemente realizar el movimiento del plan, que primero saltar para coger el diamante recuadrado y luego efectuar el plan.

Por tanto, nuestro agente elige el movimiento que captura el diamante más cercano, manteniéndose en la plataforma actual, siempre y cuando ese diamante no se capture en el siguiente movimiento del plan. Si esto sucediera, ya hemos visto que es más eficiente no perder tiempo en coger ese diamante. Cuando no se encuentra ningún movimiento que capture un diamante desde la plataforma actual, se consulta el siguiente movimiento del plan.

Es importante remarcar que esta selección para obtener el siguiente movimiento que se debe realizar no se lleva a cabo en el algoritmo de búsqueda. Allí, se asume que al llegar a una determinada plataforma se capturan todos los diamantes posibles sin abandonarla. De esta manera se reduce el tiempo de búsqueda.

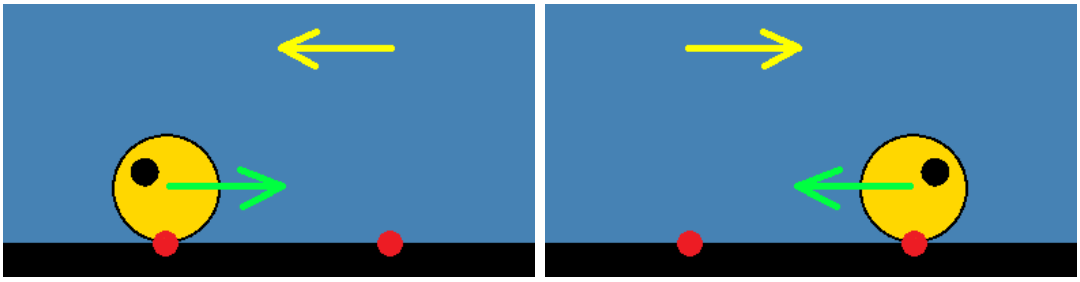


Figura 4.8: Reducción del problema de la imagen de la izquierda donde $x < x_{\text{objetivo}}$ a su problema simétrico.

4.3.3. Elección de la acción a realizar

Supongamos ahora que ya tenemos un movimiento m seleccionado que nos ayuda a coger algún diamante o nos permite cambiar de plataforma. El movimiento está determinado por el tipo de movimiento, la posición de origen del movimiento en la plataforma en la que nos encontramos y la velocidad horizontal que el círculo debe llevar en ese punto. Si la simulación que realizamos en la representación del nivel fue correcta y el agente cumple las precondiciones de posición y velocidad, al realizar la acción asociada al tipo de movimiento, deberá alcanzar los diamantes del movimiento y deberá aterrizar en la plataforma destino. Por tanto, nuestro objetivo ahora es conseguir que el círculo llegue a una determinada posición de la plataforma con una velocidad concreta.

Este problema consiste en, dados los datos (x, v_x, x_{objetivo}, v_{objetivo}) que representan la posición y velocidad actual y la posición y velocidad objetivos, respectivamente, encontrar la acción $A \in$ {ROLL_RIGHT, ROLL_LEFT} más adecuada. Nótese que podemos realizar una primera reducción, que es la siguiente: podemos asumir que el círculo se encuentra siempre a la derecha del punto objetivo, es decir, $x \geq x_{\text{objetivo}}$. En caso contrario, el problema se reduce a encontrar la acción asociada al problema simétrico de entrada I =($\bar{x}$, $\bar{v}_x$, $\bar{x}_{objetivo}$, $\bar{v}_{objetivo}$)=($2x_{\text{objetivo}} - x$, $-v_x$, x_{objetivo}, $-v_{\text{objetivo}}$). Si la acción asociada a la entrada I es ROLL_RIGHT, la acción apropiada para el problema original es ROLL_LEFT, y viceversa. En la figura 4.8, mostramos visualmente la reducción efectuada. La flecha amarilla representa la velocidad objetivo, la flecha verde, la velocidad actual y los puntos rojos son las posiciones actual y objetivo. Conservaremos esta convención en sucesivas figuras, aunque se omitirá por claridad el punto que representa la posición del círculo.

Una segunda simplificación que nos reducirá la dimensión de la entrada es considerar únicamente la diferencia $d = x - x_{\text{objetivo}} \geq 0$. Está reducción es equivalente a fijar el origen de coordenadas en la posición objetivo.

Para elegir la siguiente acción a realizar, y, como ya adelantamos en la introducción del capítulo, presentamos dos soluciones alternativas: un sistema de reglas basado en las ecuaciones del movimiento y una aproximación basada en aprendizaje por

refuerzo.

Elección de la acción mediante un sistema de reglas

El sistema de reglas consiste en una serie de heurísticas que permiten al agente saber la acción que debe tomar en función de las distancias que necesita para frenar o acelerar según distintos casos. Este sistema basado en las ecuaciones de la física dará lugar al agente que bautizamos como UCM Physics. Las acciones de rodar hacia la derecha e izquierda inducen un movimiento uniformemente acelerado en el eje horizontal del círculo, hasta que este alcanza una cierta velocidad límite. Asimismo, asumimos nuevamente que el rozamiento es despreciable, que el círculo es una masa puntual y que no hay movimiento en el eje vertical.

Por un lado, asociado al movimiento del círculo tenemos lo que llamaremos el punto de frenado, x_{frenado}. Dada la posición del círculo y su velocidad queremos saber en qué posición alcanzará velocidad nula si tomamos la acción que frena al círculo, es decir, aquella cuya aceleración tiene el signo contrario a la velocidad.

Si la posición objetivo se toma como punto de referencia, se obtiene fácilmente mediante la Ecuación (4.1) que la distancia de frenado es $-\frac{v_x^2}{2a}$, pues la velocidad inicial es v_x y la velocidad final es nula. Así, si el círculo está actualmente en la posición x_0, el punto de frenado será $x_{\text{frenado}} = x_0 - \frac{v_x^2}{2a}$, donde a (la constante de aceleración) es negativa cuando el movimiento es hacia la derecha y positiva cuando el movimiento es hacia la izquierda, es decir, siempre opuesta a la velocidad actual. En las figuras, cuando sea necesario, indicaremos el punto de frenado con un punto verde, ya que este depende esencialmente de la velocidad horizontal, representada a su vez con una flecha verde.

De manera análoga, podemos definir un punto de aceleración $x_{\text{aceleracion}}$. Queremos saber la posición desde donde, partiendo en reposo, debemos empezar a acelerar para llegar al punto objetivo con la velocidad objetivo.

Con cálculos análogos, donde ahora a tiene el mismo signo que la velocidad objetivo, se obtiene que $x_{\text{aceleracion}} = -\frac{v_{\text{objetivo}}^2}{2a}$. Nótese que si $v_{\text{objetivo}} > 0$, entonces $a > 0$ y el punto de aceleración se encuentra lógicamente a la izquierda del origen. Por consistencia, en las figuras el punto de frenado aparecerá representado con un punto amarillo.

Se puede observar que el punto de frenado y el punto de aceleración solamente dependen de x_0, v_x y v_{objetivo}, ya que $|a|$ es constante y su signo viene determinado por los signos de v_x o v_{objetivo} según el caso.

Distinguimos ahora cuatro casos en función del signo de la velocidad actual y de la velocidad objetivo.

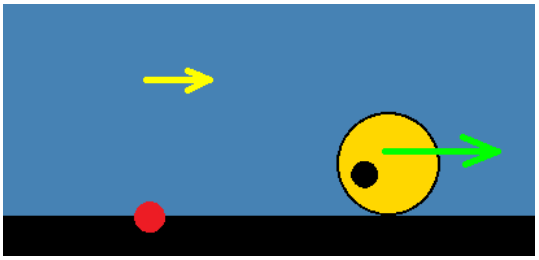


Figura 4.9: Caso $v_x \geq 0$ y $v_{\text{objetivo}} \geq 0$.

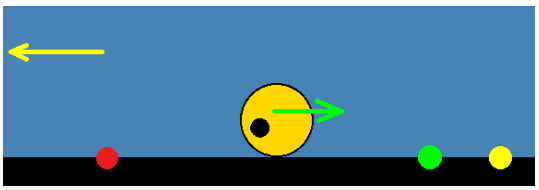


Figura 4.10: Caso $v_x \geq 0$, $v_{\text{objetivo}} < 0$ y $x_{\text{frenado}} < x_{\text{aceleracion}}$.

(a) $v_x \geq 0$ y $v_{\text{objetivo}} \geq 0$. Representa la situación en la que el círculo está a la derecha del punto objetivo y está moviéndose hacia la derecha. La velocidad objetivo es también hacia la derecha (ver figura 4.9). En este caso, el círculo debe dar media vuelta porque está yendo en la dirección contraria al punto objetivo y debe tomar siempre la acción ROLL_LEFT.

(b) $v_x \geq 0$ y $v_{\text{objetivo}} < 0$. Esta situación es parecida a la anterior, pero en este caso la velocidad objetivo es negativa. Esto hace que la acción a tomar no sea tan evidente. ¿Debemos seguir rodando hacia la derecha para tener más espacio para acelerar? ¿Tenemos que cambiar ya de sentido porque nos estamos alejando demasiado? Estas preguntas se responden observando los puntos de aceleración y de frenado. Hacemos notar que, por los signos de las velocidades, tanto el punto de frenado como el de aceleración son positivos. Hay entonces tres casos posibles:

- Si el punto de frenado está a la izquierda del punto de aceleración (figura 4.10) significa que, si frenáramos ahora, el círculo se detendría por completo antes de tener espacio suficiente para acelerar. Por tanto, la acción correcta no es frenar, sino acelerar más, es decir, ROLL_RIGHT.

- Si el punto de frenado está a la derecha del punto de aceleración (figura 4.11) significa que si seguimos acelerando, este punto cada vez se alejará más del punto de aceleración y tendremos que desandar el camino después. Por tanto, la acción correcta en este caso no es acelerar sino empezar a frenar, es decir, ROLL_LEFT.

- Si ambos puntos coinciden (figura 4.12), debemos empezar a frenar, ya que alcanzaremos la velocidad nula en la posición adecuada para empezar a acelerar. Por tanto, elegimos la acción ROLL_LEFT.

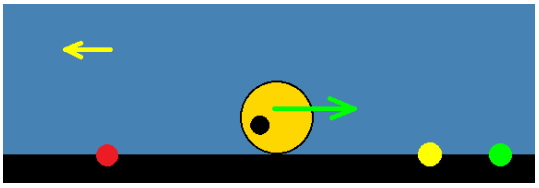


Figura 4.11: Caso $v_x \geq 0$, $v_{\text{objetivo}} < 0$ y $x_{\text{frenado}} > x_{\text{aceleracion}}$.

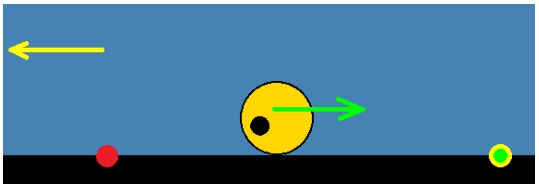


Figura 4.12: Caso $v_x \geq 0$, $v_{\text{objetivo}} < 0$ y $x_{\text{frenado}} = x_{\text{aceleracion}}$.

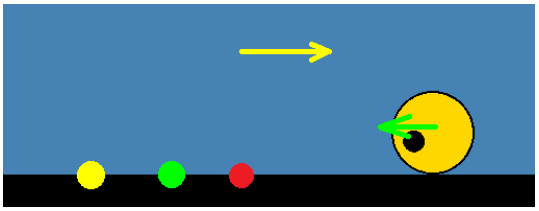


Figura 4.13: Caso $v_x < 0$, $v_{\text{objetivo}} \geq 0$ y $x_{\text{frenado}} > x_{\text{aceleracion}}$.

(c) $v_x < 0$ y $v_{\text{objetivo}} \geq 0$. Representa la situación en la que el círculo está a la derecha del punto objetivo, moviéndose hacia la izquierda y la velocidad objetivo es hacia la derecha. Es una situación bastante parecida a la anterior en la que, para saber cómo actuar, hay que atender a los puntos de frenado y aceleración. En este caso, el punto de aceleración es negativo porque $a > 0$, pero el signo del punto de frenado no es claro. Distinguimos entonces varios casos:

- Si el punto de frenado está a la derecha del punto de aceleración (figura 4.13) y el agente frenara en ese instante, llegaría al punto de frenado en un cierto tiempo con velocidad 0. Pero el punto de frenado está más cerca del origen que el punto de aceleración y no habría espacio suficiente para alcanzar la velocidad deseada. Por tanto, la acción correcta es seguir acelerando, es decir, ROLL_LEFT.

- Si el punto de frenado está a la izquierda del punto de aceleración (figura 4.14) no tiene sentido seguir acelerando porque el círculo aleja cada vez más su punto de frenado del punto de aceleración. La acción adecuada en este caso es frenar, es decir, ROLL_RIGHT.

(d) $v_x < 0$ y $v_{\text{objetivo}} < 0$. En esta situación, el círculo está a la derecha del origen, dirigiéndose hacia él y la velocidad objetivo es negativa (figura 4.15). En este caso no es evidente cuál es la acción apropiada.

Figura 4.14: Caso $v_x < 0$, $v_{\text{objetivo}} \geq 0$ y $x_{\text{frenado}} \leq x_{\text{aceleracion}}$.

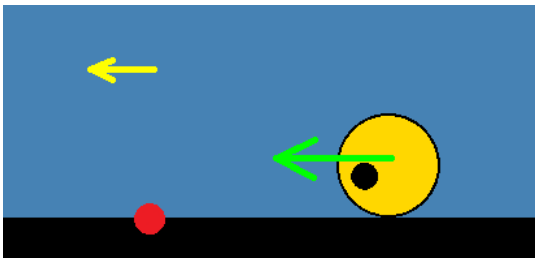


Figura 4.15: Caso $v_x < 0$ y $v_{\text{objetivo}} < 0$.

Para resolver esta situación, calculamos las velocidades máxima y mínima que podemos alcanzar. Es decir, queremos conocer cuál será la velocidad del círculo en el origen si toma siempre la acción de rodar hacia la izquierda y cuál será la velocidad del círculo en el origen si toma siempre la acción de rodar hacia la derecha. Para calcularlas, despejamos una vez más de la Ecuación (4.1), teniendo en cuenta que la aceleración puede ser positiva o negativa, resultando en:

$$v_{\text{max}}^2 = v_x^2 + 2|a|x_0 \qquad\qquad v_{\text{min}}^2 = v_x^2 - 2|a|x_0$$

donde no hemos despejado la raíz cuadrada debido a que la segunda de ellas podría no existir. Aunque matemáticamente es incorrecto considerar $v_{\text{min}}^2 < 0$, en este caso es conveniente debido a que esto representa que la pelota tiene espacio suficiente para frenar por completo antes de llegar al origen.

Tenemos ahora varios casos posibles:

- $v_{\text{max}}^2 < v_{\text{objetivo}}^2$: En este caso, por mucho que aceleremos, no vamos a alcanzar la velocidad suficiente para llegar a la velocidad objetivo, por lo que debemos frenar y coger velocidad desde más lejos. Elegimos por tanto la acción ROLL_RIGHT.

- $v_{\text{min}}^2 > v_{\text{objetivo}}^2$: En este caso, aunque intentemos frenar nos vamos a pasar de la velocidad objetivo. A pesar de ello la mejor acción es frenar, es decir, ROLL_RIGHT ya que acelerar solamente empeora la situación.

- $v_{\text{max}}^2 \geq v_{\text{objetivo}}^2$ y $v_{\text{min}}^2 \leq v_{\text{objetivo}}^2$: Este es el caso que nos interesa, ya que significa que vamos a poder alcanzar la velocidad objetivo. Aquí podríamos realizar varias acciones, pero, teniendo en cuenta que queremos completar el nivel en el menor tiempo posible, lo mejor es ir siempre con la mayor velocidad posible, hasta que $v_{\text{min}}^2 = v_{\text{objetivo}}^2$, en cuyo momento

comenzaremos a frenar. Es decir, si $v^2_{\text{min}} < v^2_{\text{objetivo}}$, realizamos la acción ROLL_LEFT, y si $v^2_{\text{min}} = v^2_{\text{objetivo}}$, realizamos la acción ROLL_RIGHT. Nuestro agente es un poco más conservador, de forma que empieza a frenar si $v^2_{\text{min}} \leq v^2_{\text{objetivo}} + \varepsilon$, donde $\varepsilon > 0$ es una constante de error. Esta mejora evita que acelere demasiado y no alcance la velocidad objetivo.

Esto concluye la explicación del sistema de reglas. Se ha podido comprobar que, para resolver el problema, ha sido necesario incluir conocimiento experto al sistema. De hecho, todas las reglas que hemos diseñado parten de un estudio de las ecuaciones del movimiento que rigen el comportamiento del agente en este problema, y es necesario contar con conocimientos básicos de mecánica clásica para deducirlas.

Elección de la acción mediante aprendizaje por refuerzo

La otra alternativa que nos planteamos es que sea el propio agente el que, mediante un sistema de recompensas, aprenda a resolver el problema sin la necesidad de que se le proporcione un conocimiento *a priori*.

Para esta implementación, hemos optado por la resolución mediante *Q-Learning*, en base a los buenos resultados que habían obtenido agentes anteriores como MARL-GF Agent, del que hablamos en la sección 2.3.10. Como resultado, obtuvimos el agente UCM QLearning, que compararemos en la sección 7.2 con el agente UCM Physics y con los mejores círculos del estado del arte. Para el desarrollo de este agente, caracterizamos los estados por los siguientes atributos:

- La distancia horizontal al punto objetivo (atributo *distance*), medida en cuadrados de 8×8 píxeles del juego. Ya comentamos en la sección 4.3.3 que bastaba adaptar el problema para que este atributo fuera siempre positivo, reduciendo el número de estados posibles. Además, solamente consideramos las distancias menores o iguales que 25 cuadrados de 8×8 píxeles. Si el rectángulo se encontraba más alejado, lo acercábamos, haciendo que ejecutara la acción ROLL_RIGHT si se encontraba a la izquierda del punto objetivo y ROLL_LEFT si se encontraba a la derecha. Esto reducía enormemente el número de estados a considerar, ya que cada nivel tiene una anchura de más de 150 cuadrados de 8×8 píxeles, por lo que resultaría inviable entrenar con todos los estados posibles. Calculamos que, con 25 cuadrados, el agente debería tener espacio más que suficiente para acelerar y frenar sin problema, independientemente de la velocidad objetivo.

- La velocidad actual del círculo (atributo *current_velocity*), redondeada al múltiplo de 20 más cercano. Dado que la máxima velocidad posible es alrededor de 200, esto supone un total de 21 velocidades posibles (10 positivas, 10 negativas y la velocidad nula) como valor del atributo.

- La velocidad objetivo del círculo (atributo *target_ velocity*), que también está redondeada al múltiplo de 20 más cercano.

El entrenamiento se llevó a cabo en un nivel plano, sin obstáculos, en el que el objetivo del agente era llegar al centro del nivel con la velocidad que estuviéramos entrenando. Una vez se lograba, el agente saltaba, se movía a un lugar aleatorio del nivel, lo que no se tenía en cuenta para la recompensa, y volvía a empezar. Esto se hacía para que los estados que explorara el agente fueran diferentes cada vez y se pudiera rellenar la tabla adecuadamente. Es importante resaltar que como velocidad objetivo asignábamos siempre una cantidad positiva, ya que la simetría que realizamos para la distancia implica que el estado que se considera puede tener una velocidad objetivo tanto positiva como negativa. Esto reduce el número de entrenamientos diferentes necesarios.

Así, generamos una tabla para cada una de las posibles velocidades objetivo. En cada una de ellas, para cada estado, se le asignaría un valor a cada una de las acciones del conjunto $A = \{$ ROLL_LEFT, ROLL_RIGHT, NO_ACTION $\}$, que es el que se iría puliendo a medida que avanzara el entrenamiento. Asignamos una recompensa de 500 si se logra llegar al punto medio con la velocidad que se desea, y una recompensa de -1 por cada iteración en la que no se logre. Para que el valor positivo se propague adecuadamente, utilizamos un valor de la tasa de descuento de $\gamma = 0,95$. Por otro lado, ya que el entrenamiento de cada velocidad se realizó durante unas tres horas a una velocidad de 10 veces la del juego, se fijó un valor del ratio de aprendizaje de $\alpha = 0,1$. Por último, el valor de ε, que determina la probabilidad de realizar una acción aleatoria y por tanto la posibilidad de explorar nuevas acciones frente a explotar las que hasta ahora dan el mejor resultado, se fue decrementando durante las ejecuciones pasando de $0,3$ a 0, mediante un decrecimiento exponencial.

De esta forma, en el estado s se toma la acción $a \in A$ que maximiza la función $Q(s,a)$ con probabilidad $1-\varepsilon$ y una acción aleatoria con probabilidad ε. Una vez elegida la acción a, que desde el estado s lleva al estado s', se actualiza el valor de la tabla Q mediante la fórmula,

$$Q(s,a) = (1-\alpha) \cdot Q(s,a) + \alpha \cdot (R(s,a) + \gamma \cdot \max_{a \in A} R(s',a)). \tag{4.2}$$

Tras realizar el entrenamiento, comprobamos que la mayoría de velocidades se alcanzaban con relativo éxito, aunque empleando más tiempo y espacio del estrictamente necesario a pesar de la recompensa negativa por cada iteración en la que no se lograra el objetivo. Sin embargo, para las más altas (160, 180 y 200), el agente encontraba muchas dificultades, quedándose normalmente estancado en un punto sin lograr avanzar. En consecuencia, decidimos que, de cara a la ejecución del agente, solamente consideraríamos trayectorias que se hubieran simulado con velocidades inferiores a 160. Lógicamente, esto imposibilita la resolución de algunos niveles, como veremos en la sección 7.2. También cabe destacar que la discretización de velocidades que se hacen siguiendo las dos alternativas para determinar la siguiente acción no es la misma. Mientras que en el caso del Q-Learning, donde cada nueva velocidad que se incluye requiere esfuerzo independiente para ser entrenada, el sistema de reglas basa-

do en las ecuaciones físicas toma como parámetro de entrada la velocidad objetivo y no requiere ninguna modificación. Esto nos ha permitido realizar una discretización más fina de velocidades para el agente UCM Physics (de paso 15) frente al agente UCM QLearning (de paso 20), lo que también afecta en los resultados de los agentes.

De las preguntas que nos planteamos al inicio de la sección 4.3, ya hemos contestado aquellas relativas a la replanificación y a la elección del siguiente movimiento y la siguiente acción a tomar. Para esto último, hemos presentado dos sistemas alternativos, uno basado en reglas y otro en aprendizaje por refuerzo. En lo que resta de capítulo, nos centraremos en explicar a qué nos referimos con que dos movimientos sean igual de válidos, lo que nos ayudará a entender cuándo se toman los saltos. Asimismo, trataremos dos mejoras que perfeccionan la actuación de los agentes. A partir de aquí, las dos versiones del círculo que hemos desarrollado se comportan de la misma forma, por lo que volveremos a referirnos a "el agente" para referirnos a ambos indistintamente.

4.3.4. Política de saltos

Una vez el agente se encuentra cerca de la posición en la que comienza el siguiente movimiento (a una distancia menor o igual de dos píxeles discretizados), se compara su velocidad con la deseada. A continuación, detallamos las decisiones tomadas en función del resultado de esta comparación, aunque supondremos que el movimiento es de tipo JUMPMOVE, es decir, es un salto. Esto se debe a que para los movimientos de tipo FALL y NOMOVE la decisión tomada no supone ninguna diferencia, como comentaremos después.

(a) Si la discretización de la velocidad actual es la velocidad objetivo, entonces el agente salta.

(b) Si la discretización de la velocidad no es la velocidad objetivo, entonces se realiza una simulación del salto en caso de que el agente saltara con la velocidad actual desde la posición actual. Si esta simulación coge el mismo número de diamantes y aterriza en la misma plataforma que el movimiento original, entonces el agente salta igualmente, ya que se supone que ambos movimientos son igual de válidos. Como el criterio para filtrar las parábolas conseguía que nos quedáramos con la mejor parábola y ahora estamos planteándonos saltar con otra que no es tan buena, se aplica un filtro adicional antes de decidir si saltar o no. Este es que el movimiento deje suficiente espacio para frenar desde el punto de aterrizaje al borde de la plataforma destino, pero este filtro únicamente se aplica si el movimiento original también lo superaba, para no exigirle más de lo que le exigíamos al que considerábamos como mejor movimiento.

(c) En caso de que no se dé ninguna de las situaciones anteriores, el agente frenará y volverá a intentar llegar a la posición objetivo con la velocidad adecuada.

Esto es lo que llamamos un salto abortado.

El motivo por el que esta política no afecta a los movimientos NOMOVE y FALL se debe a situaciones diferentes. En primer lugar, los movimientos NOMOVE siempre tienen una velocidad objetivo de 0, por lo que, si el agente está cerca de la posición objetivo, siempre va a querer frenar. En cuanto a los movimientos FALL, lo que sucede es que, dado que estos movimientos parten de las esquinas de las plataformas, no hay espacio para frenar aún en el caso en el que se quisiera abortar el movimiento, por lo que la política anterior no tiene ninguna consecuencia en el comportamiento del círculo.

4.3.5. Comentarios adicionales sobre la ejecución del plan

Por último, destacamos otros aspectos de menor importancia que hemos tenido en cuenta durante el desarrollo del agente, y que han permitido obtener mejores resultados en determinadas situaciones.

Elección de la acción a tomar estando en el aire

Durante un vuelo, el agente puede seguir tomando acciones. Aunque estas acciones no influyen en la trayectoria del agente, sí que afectan al momento angular de la pelota, lo que a su vez tiene consecuencias en su comportamiento al rebotar con el suelo de la plataforma en la que aterriza. Si la velocidad horizontal es positiva, un sentido de giro horario hará que el primer contacto con el suelo aumente la velocidad horizontal, mientras que un sentido de giro antihorario ayudará a frenar. El caso de una velocidad horizontal negativa es simétrico.

Por tanto, dado que una vez aterricemos de un salto queremos mantenernos en dicha plataforma sin caer por los extremos, hemos implementado un sistema de reglas para elegir las acciones durante el vuelo. Inicialmente planteamos un sistema sencillo tal que, si la velocidad horizontal era positiva, elegíamos rodar a la izquierda para conseguir un sentido de giro antihorario y que al contactar con la plataforma de aterrizaje se redujera la velocidad horizontal. Por otro lado, si la velocidad horizontal era negativa, elegíamos rodar a la derecha para conseguir un sentido de giro horario con un propósito análogo. De esta forma evitábamos caer por un extremo de la plataforma por llevar demasiada velocidad y no tener espacio suficiente para frenar.

Sin embargo, este sistema tan sencillo tenía una desventaja. Cuando el agente intentaba alcanzar una plataforma muy alta, nada más tocar la plataforma destino caía por el borde por el que había subido, haciendo imposible completar el salto. Un ejemplo de esto se encuentra en la figura 4.16.

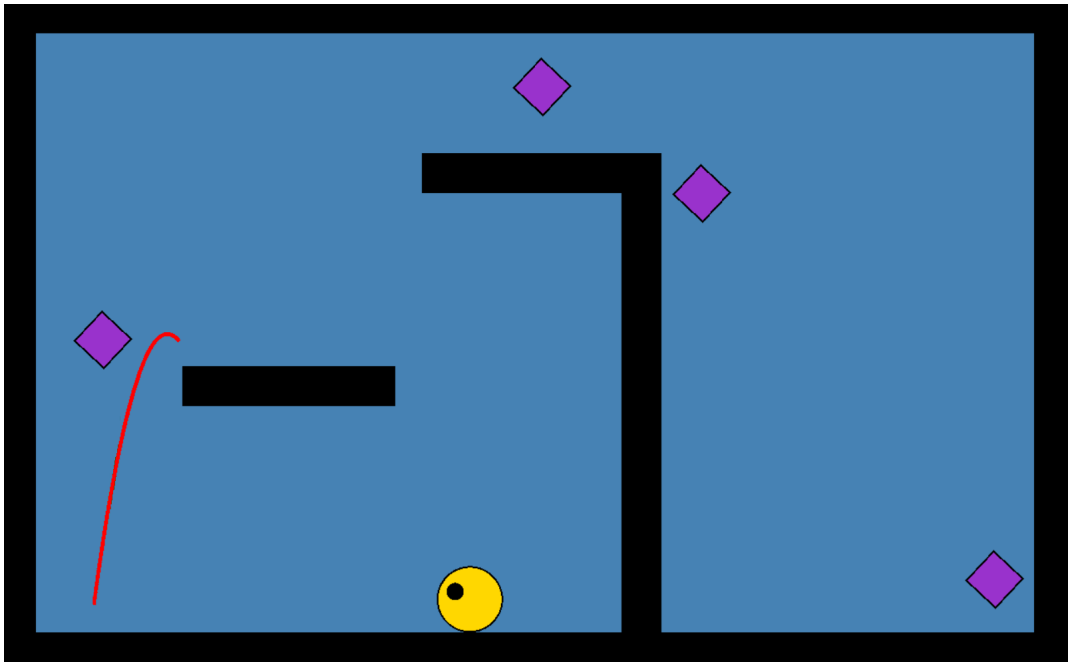


Figura 4.16: Nivel en el que el primer movimiento del plan es un salto a una plataforma muy alta.

Como se puede observar, dado que la plataforma destino está muy por encima de la de salida, el punto de aterrizaje está muy cerca del borde, propiciando una caída si se llega con giro muy acentuado en sentido antihorario. Esto es inevitable, ya que el punto de aterrizaje está muy cerca del punto más alto de la trayectoria, lo que provoca que el primer rebote expulse al círculo de la plataforma.

Ante estas situaciones, decidimos cambiar ligeramente el sistema de reglas anterior, haciendo que, si la plataforma de destino estaba a una altura cercana a la altura máxima que se alcanza en el vuelo, el sentido de giro favorezca el poder subir a la plataforma. Así, en esta situación, si la velocidad es positiva, el agente efectuará la acción ROLL_RIGHT, mientras que si es negativa, realizará un giro ROLL_LEFT.

Sistema para evitar caídas indeseadas

Durante las pruebas que realizamos con nuestro agente, a pesar de que en el filtrado de movimientos priorizábamos los movimientos que dejaran espacio detrás del punto de aterrizaje para frenar (ver figura 4.6) y tomábamos acciones en el aire para evitar caídas por los bordes, era frecuente que el agente, si llegaba al borde de una plataforma, acabara cayendo. Esto se debe a que, si el centro de la pelota está medio píxel discretizado fuera de la plataforma, el agente no identifica que sigue en la plataforma, a pesar de que rodando podría volver sin problema. En consecuencia, el agente no sabía qué acción realizar en este caso y acababa cayendo, normalmente de manera muy lenta, por el peso de la gravedad.

Para solucionar este problema, implementamos una comprobación, que no hemos incluido en el algoritmo 3 por claridad, y que evalúa si el círculo está en el último píxel de una plataforma o si está en el primer píxel fuera de la plataforma. En ese caso, ordenamos al agente que ruede hacia el centro de la plataforma. Esta verificación se lleva a cabo antes incluso de la replanificación, ya que en estas situaciones queremos elegir esta acción antes de realizar ningún cálculo que pueda demorar la ejecución de la acción y empeorar la situación a un estado desde el que el agente ya no pueda recuperarse. Se puede ver cómo actúa el sistema para evitar caídas en la animación del siguiente enlace.

Con el sistema para evitar caídas concluimos la última sección del capítulo sobre las particularidades del círculo. Recordemos que esta sección se ha centrado en desgranar el algoritmo 3, que describe la ejecución del agente círculo. En el siguiente capítulo abordaremos las particularidades del rectángulo. La estructura será similar a la de este capítulo, comenzando por la representación propia del nivel que hace el rectángulo y finalizando con un algoritmo de ejecución para ese personaje.

Capítulo 5

Desarrollo del rectángulo

En este capítulo, y de manera análoga al capítulo 4, explicamos el funcionamiento del rectángulo que hemos desarrollado. Para ello, completaremos lo explicado en el capítulo 3 sobre el desarrollo común de los agentes. En la sección 5.1 ofrecemos una breve introducción al rectángulo, en la cual se recuerdan sus características básicas y se recogen algunas de las constantes físicas que han sido útiles para el desarrollo. En la sección 5.2, nos centraremos en la representación de los niveles, explicando las plataformas del rectángulo, la generación de movimientos y su posterior filtrado. En la sección 5.3 ampliamos la exposición realizada en la sección 3.3.2 sobre el trazado del plan a seguir. Por último, en la sección 5.4, se comenta en detalle el algoritmo de ejecución del rectángulo.

5.1. Introducción

El rectángulo es el segundo personaje del juego Geometry Friends. Se trata de un rectángulo verde que puede variar su forma y desplazarse a izquierda y derecha. Aunque existe una modalidad en la que el círculo y el rectángulo deben colaborar para alcanzar los diamantes, eso lo dejaremos para el capítulo 6 y antes nos vamos a centrar en aquellos niveles en los que únicamente participa el rectángulo.

Las acciones características del rectángulo son diferentes a las del círculo y eso se aprecia en el conjunto de acciones que puede tomar. Estas son MOVE_RIGHT, MOVE_LEFT, MORPH_UP, MORPH_DOWN y NO_ACTION. Las dos primeras permiten al rectángulo deslizarse hacia la derecha y la izquierda. Esto se consigue aplicándole una fuerza mientras se elige dicha acción, que se traduce en que el rectángulo sufre una aceleración constante que hace que se desplace hacia uno de los lados. Las acciones MORPH_UP y MORPH_DOWN sirven para redimensionar el rectángulo. Su área debe mantenerse constante, pero puede ser más vertical si su

altura crece al tomar la acción MORPH_UP o más horizontal si su base se ensancha al tomar la acción MORPH_DOWN. La acción NO_ACTION, se muestra como alternativa cuando no se desea tomar ninguna de las acciones anteriores.

Como ya comentamos en capítulos anteriores, el rectángulo colisiona contra los obstáculos cuyo color no es verde. Es decir, atraviesa los obstáculos verdes y choca contra los amarillos y los negros.

Antes de comenzar a desarrollar el rectángulo, fue de gran utilidad calcular experimentalmente las siguientes constantes, que sabíamos que nos serían imprescindibles para poder realizar una modelización adecuada del problema:

- El área del rectángulo.
- Las alturas máxima y mínima del rectángulo.
- La aceleración que experimenta al aplicarle una acción de desplazamiento.
- La aceleración de la gravedad.
- La velocidad máxima horizontal que puede alcanzar el rectángulo.

Además, comprobamos que la fuerza de rozamiento tenía un efecto despreciable sobre la velocidad, por lo que la descartamos de todos nuestros cálculos.

Dado que la representación del estado del entorno del juego quedó detallada en el capítulo 3, común al círculo y al rectángulo, la primera particularidad del rectángulo la encontramos en la definición de las plataformas. En las siguientes secciones también trataremos la generación de movimientos, su filtrado, la elaboración del plan y su ejecución.

5.2. Representación del nivel

En primer lugar, detallamos las consideraciones tomadas durante el preprocesamiento de un nivel. En las siguientes subsecciones trataremos temas que ya habíamos anticipado en la sección 3.3.1 como son las plataformas del rectángulo y los movimientos propios del rectángulo y su posterior filtrado.

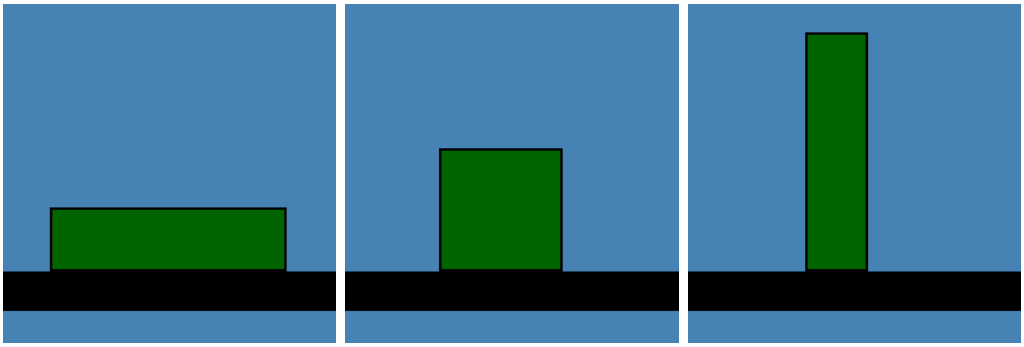


Figura 5.1: Diferentes formas del rectángulo.

5.2.1. Plataformas del rectángulo

De acuerdo a lo que comentamos en la sección 3.3.1, donde explicamos cómo se realizaba la discretización y se identificaba cada punto discretizado como OBSTACLE, PLATFORM, DIAMOND o EMPTY, quedó pendiente explicar las particularidades de las plataformas del rectángulo. Recordamos que una plataforma es un conjunto de puntos contiguos horizontalmente donde el agente puede apoyarse sin intersecar con otros obstáculos. Para el círculo era fácil identificar las plataformas porque simplemente había que evaluar si la discretización exterior del círculo (ver Figura 4.2) situada sobre cada uno de los putos de un obstáculo intersecaba con otro obstáculo. Como comentamos, esto era sencillo porque la forma del círculo es siempre la misma. Sin embargo, el rectángulo tiene la particularidad de que puede cambiar de forma tomando las acciones MORPH_UP y MORPH_DOWN. Con ello puede variar su forma de manera continua, manteniendo constante su área, como se puede ver en la Figura 5.1.

Puede partir desde una en la que está completamente horizontal, hasta otra en la que está completamente vertical, pasando por todas las formas intermedias, como por ejemplo la forma de cuadrado. Por lo tanto no es tan sencillo definir en este caso qué es una plataforma para el rectángulo, porque es posible que el rectángulo pueda descansar sobre un obstáculo en posición horizontal, pero no en posición vertical, como se muestra en la Figura 5.2.

Es necesaria una extensión de la definición de las plataformas para el caso del rectángulo. Por simplicidad, de aquí en adelante y salvo que se indique lo contrario, únicamente vamos a tratar con las tres formas del rectángulo que aparecen en la Figura 5.1, es decir, horizontal, cuadrado y vertical, y descartaremos todas las demás formas intermedias. Como antes, una plataforma será un conjunto de puntos contiguos horizontalmente sobre los cuales se puede situar el rectángulo sin que interseque con ningún otro obstáculo. Sin embargo, en esta ocasión, se debe cumplir que el subconjunto de las tres formas destacadas de la Figura 5.1 con las que el rectángulo puede descansar sea el mismo en todos los puntos.

Esto se entiende mejor con una imagen como la de la Figura 5.3. En ella podemos

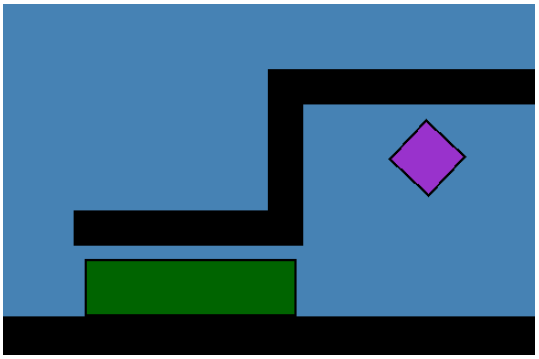


Figura 5.2: El rectángulo puede descansar en posición totalmente horizontal pero no en posición totalmente vertical.

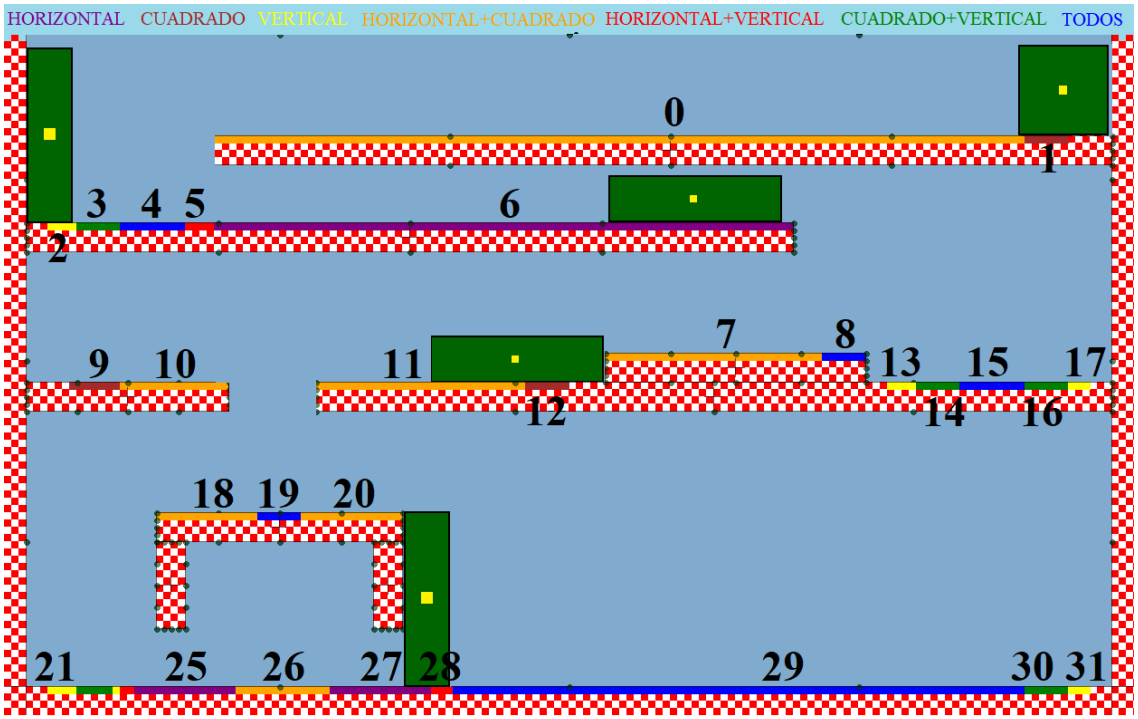


Figura 5.3: Identificación de las plataformas del rectángulo en el nivel 5 de la competición del año 2013.

ver identificadas con distintos colores las plataformas según las formas posibles del rectángulo en cada una de ellas. Además, aparecen las formas destacadas del rectángulo en lugares estratégicos para tomar una referencia de los tamaños y guiar la explicación. Hay siete posibles tipos de plataforma.

- Con color marrón, representamos las plataformas en las que la única forma posible es el cuadrado, por ejemplo, las plataformas 1, 9 y 12. En la plataforma 1, nos podemos plantear por qué no se puede extender la plataforma más hacia la derecha, continuando hasta la pared o por qué no podemos situar en la plataforma 12 la forma horizontal. La respuesta es que el centro del rectángulo es lo que se utiliza para situarlo. Para cada punto del obstáculo, se

sitúa cada una de las formas de tal manera que el rectángulo descanse sobre el suelo y el centro del rectángulo se proyecte sobre dicho punto. Tal y como se puede apreciar, el cuadrado de la esquina superior derecha está pegado a la pared y el centro del mismo se proyecta en el extremo derecho de la plataforma 1. Si extendiéramos la plataforma 1 un punto más hacia la derecha, entonces deberíamos poder mover el cuadrado un punto más a la derecha, lo cual es imposible porque atravesaría la pared. De igual manera, no se puede situar en la plataforma 12 la forma horizontal porque tomamos como referencia el centro. Como se puede apreciar el rectángulo descansa sobre las plataformas 11, 12 y sobre un obstáculo de cuadros rojos y blancos. Sin embargo, para nosotros el rectángulo está situado sobre la plataforma 11, ya que es donde se proyecta su centro. Por último, es claro que no se puede situar el centro del rectángulo en forma horizontal sobre la plataforma 12 sin que este choque con el obstáculo de la derecha.

- De color morado, están pintadas las plataformas en las que la única forma de rectángulo posible es la horizontal, como son las plataformas 6, 25 y 27. Es claro que, en la plataforma 6, la única forma que se puede situar sin que choque con el obstáculo superior es la horizontal, pero para las plataformas 25 y 27 no es tan evidente. Pero el motivo es el mismo que en el caso anterior: la referencia es el centro de los rectángulos y no sus extremos. En este caso, el centro del rectángulo vertical se proyecta sobre la plataforma 28.

- Las plataformas en las que la única forma posible es la vertical aparecen marcadas de amarillo (2, 13, 17, 21, ...). Como se puede apreciar, en la plataforma 2 por ejemplo, como la anchura del rectángulo en forma vertical es mínima, el centro del mismo puede estar más próximo a las paredes y, por tanto, se generan plataformas amarillas donde únicamente se puede situar el rectángulo en forma vertical.

- De color anaranjado aparecen coloreadas las plataformas que admiten tanto la forma horizontal como el cuadrado, pero no la vertical. Por ejemplo, las plataformas 0, 7, 10, 11, 18, etc.

- Aunque pueda parecer extraño en un primer momento, también hay plataformas en las que se puede estar en la forma horizontal y vertical, pero no en forma de cuadrado. Estas están coloreadas de rojo y ejemplos de ellas son la plataforma 5 o la 28.

- Las plataformas verdes son aquellas que admiten la forma cuadrada y vertical, pero no la horizontal, como las plataformas 3, 14 o 15.

- Por último, aquellas plataformas en las que se puede situar el rectángulo con cualquier tipo de forma las señalamos con color azul oscuro. Ejemplos de estas últimas son las plataformas 4, 5, 8, 15, 19 y 29.

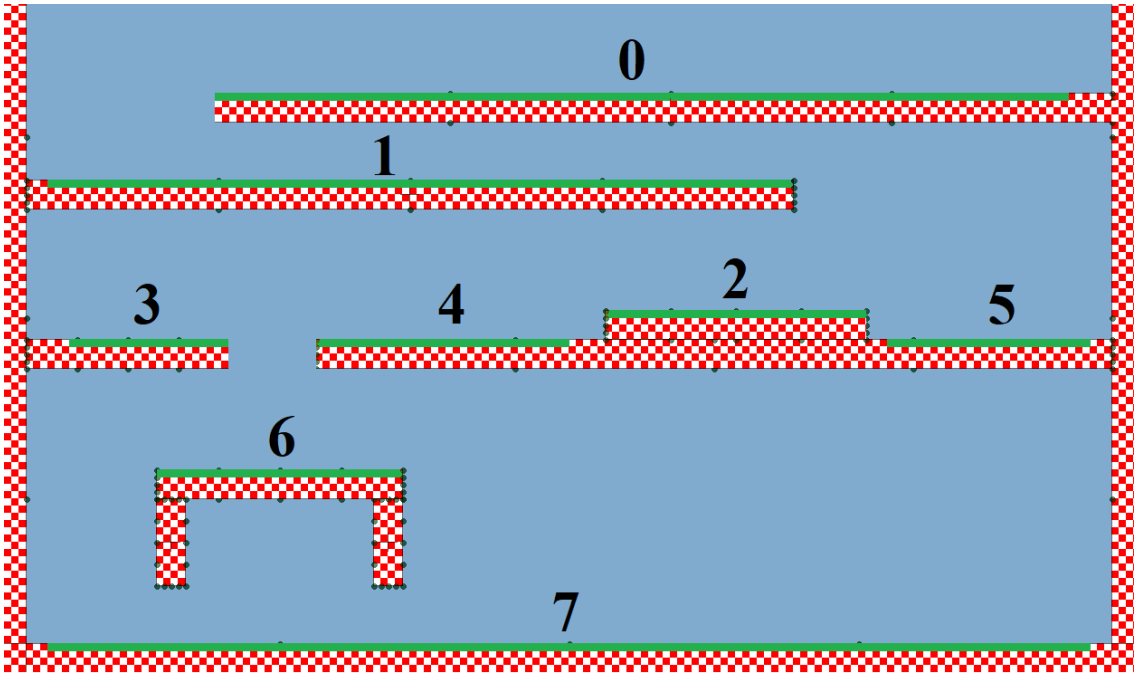


Figura 5.4: Plataformas simplificadas del rectángulo en el nivel 5 de la competición del año 2013.

Aunque este sistema pueda parecer complejo, el centro del rectángulo es la información más accesible que tenemos del mismo. Además, saber qué formas son admisibles en cada plataforma nos será muy útil más adelante, cuando tengamos que generar movimientos o decidir qué acción tomar. Sin embargo, hay muchas de las plataformas que hemos generado que son adyacentes y que permiten que el rectángulo se mueva de una a otra con las acciones MOVE_LEFT o MOVE_RIGHT, si admite una forma común en ambas plataformas.

En consecuencia, y dado que durante la planificación contar con un número más reducido de plataformas es más eficiente, introducimos el concepto de plataforma simplificada. Una plataforma simplificada no es más que la unión de plataformas adyacentes. Las plataformas simplificadas del nivel de la Figura 5.3 aparecen en la Figura 5.4.

5.2.2. Generación de movimientos

Como ya comentamos en la sección 3.3.1, los movimientos que pueden realizar el círculo y el rectángulo son diferentes porque tienen características únicas. Mientras que el círculo puede saltar mediante la acción JUMP, el rectángulo no posee esa aptitud. Sin embargo, el rectángulo puede cambiar su forma manteniendo constante su área.

A continuación, vamos a definir los tipos de movimiento del rectángulo, que serán

los que permitan pasar de una plataforma a otra o alcanzar algún diamante, y que no deben ser confundidos con las acciones. Las acciones son atómicas y se toman en cada llamada al método *Update*, mientras que los movimientos representan estrategias de más alto nivel, que se pueden completar realizando una secuencia de acciones, y que juntos forman el plan completo. Los tipos de movimiento del rectángulo son: NOMOVE, ADJACENT, FALL, DROP, TILT, MONOSIDEDROP, BIGHOLEADJ, BIGHOLEDROP y HIGHTILT. En los siguientes apartados se detalla cada uno de ellos. Esta sección es la más importante del capítulo, y el resto de ellas se siguen de manera natural una vez se identifican los diferentes tipos de movimientos.

Movimientos NOMOVE

Comenzamos con el tipo de movimiento más sencillo, el NOMOVE. Al igual que en el caso del círculo (ver sección 4.2.1), los movimientos NOMOVE representan la situación en la que el rectángulo puede alcanzar un diamante que está a la altura de una plataforma. Se procede de la misma manera: se evalúa en cada punto de cada plataforma si el rectángulo, en cada una de sus tres formas, interseca con la discretización de alguno de los diamantes. Al contrario que con el círculo, donde había que tener precaución sobre si era más conveniente tomar una discretización interior (Figura 4.1) o exterior (Figura 4.2) para calcular las intersecciones, en este caso, la forma rectangular hace que la discretización del rectángulo coincida con el propio rectángulo y por tanto no nos referiremos a ella a no ser que sea estrictamente necesario.

Sin embargo, la discretización del diamante de la Figura 3.2 sí nos ocasionaba problemas de precisión. Hay algunos niveles, como el de la Figura 5.5 de la competición del año 2016, en el cual no se generaba un movimiento de tipo NOMOVE porque el rectángulo no puede alcanzar la discretización del diamante, pero sí que puede alcanzar el vértice inferior del diamante.

Esta es una de las limitaciones que surgen de discretizar el entorno y que provocan una pérdida de precisión que, en este caso, nos resulta problemática. A pesar de ello, es posible solucionarlo, a costa, claro, de aumentar el coste computacional. Como la representación del nivel se realiza antes de que se inicie la cuenta del tiempo y no supone un empeoramiento apreciable en el rendimiento, decidimos realizar una comprobación adicional. Cuando se evalúan las intersecciones del rectángulo en forma vertical, comprobamos si la parte superior del rectángulo interseca con el diamante (no con su discretización). Esto tampoco es complicado ya que esa condición es equivalente a que la distancia L^1, es decir, la distancia Manhattan, entre los píxeles del borde superior del rectángulo y el centro del diamante sea menor que la mitad de la altura del diamante. Nótese que el diamante no es otra cosa que una bola sobre la métrica L^1. Esta medida consigue solventar el problema.

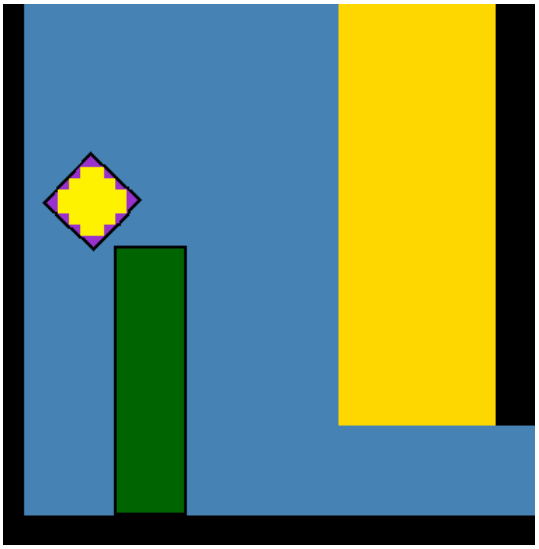


Figura 5.5: Fragmento del nivel 10 de la competición del año 2016 que muestra un diamante (morado) que puede ser alcanzado por el rectángulo, pero su discretización (amarilla) no puede ser alcanzada.

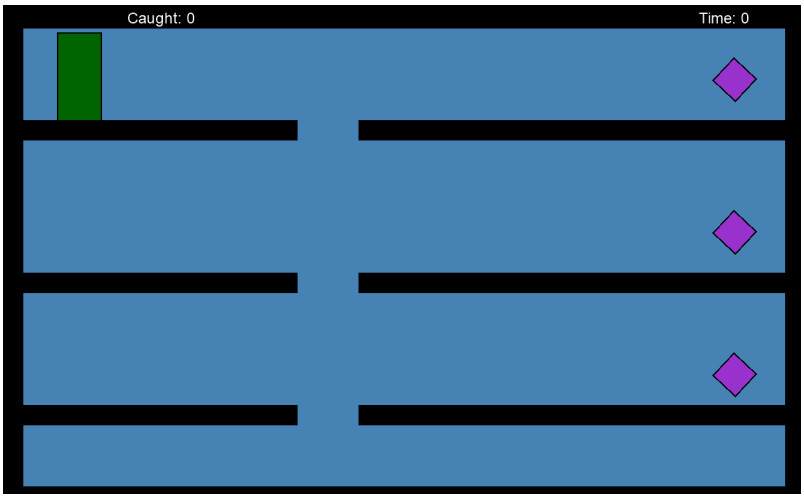


Figura 5.6: Nivel 10 de la competición del año 2013.

Movimientos DROP

Este tipo de movimiento se crea por la existencia de un patrón común en los niveles de los años anteriores. El patrón se puede ver en la Figura 5.6 y consiste en una estructura en la que hay dos plataformas a la misma altura con un hueco de tamaño pequeño entre ellas. El tamaño del hueco es suficientemente pequeño para que el rectángulo pueda pasar de una plataforma a otra cuando está en forma horizontal y suficientemente grande para que el rectángulo pueda dejarse caer cuando está en forma vertical. Por tanto, el movimiento añade la posibilidad de que el rectángulo se deje caer por el hueco entre las plataformas.

Figura 5.7: Secuencia del movimiento DROP.

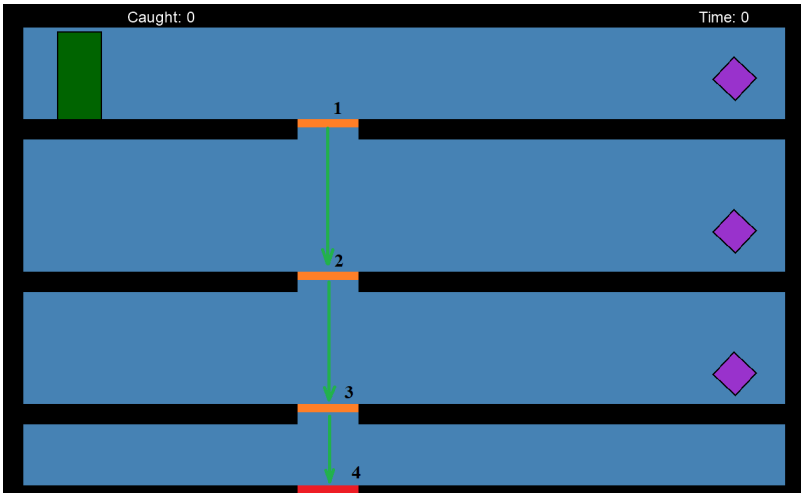


Figura 5.8: Plataformas ficticias y movimientos DROP del nivel 10 de la competición del año 2013.

Aunque en esta fase únicamente estamos considerando la existencia de una conexión entre las plataformas superiores y una plataforma inferior, describimos el movimiento que tenemos en mente que realice el rectángulo. Queremos que se sitúe en forma horizontal con su centro alineado con el centro del hueco y con velocidad cero y entonces pase a forma vertical, dejándose caer por el hueco (ver Figura 5.7). Recomendamos acceder a una animación mediante el siguiente enlace.

Para modelizar este tipo de movimiento nos ha sido de gran utilidad considerar lo que hemos llamado *plataformas ficticias*, que son aquellas que hacen las veces de puente imaginario entre las dos plataformas a la misma altura en una estructura de tipo DROP. Son las plataformas de color naranja de la Figura 5.8. De esta forma, la plataforma destino de un movimiento DROP es la primera plataforma, real o ficticia, en la que se proyecta el punto medio del hueco del DROP. Por tanto, en la Figura 5.8 hay tres movimientos de tipo DROP: de la plataforma ficticia 1 a la plataforma ficticia 2, de la plataforma ficticia 2 a la plataforma ficticia 3 y de la plataforma ficticia 3 a la plataforma real 4.

Finalmente, durante el movimiento de caída, hasta que se llega a una plataforma (real o ficticia), se evalúa si el rectángulo interseca alguno de los diamantes. Los alcanzados se añaden al conjunto de diamantes capturados durante el movimiento.

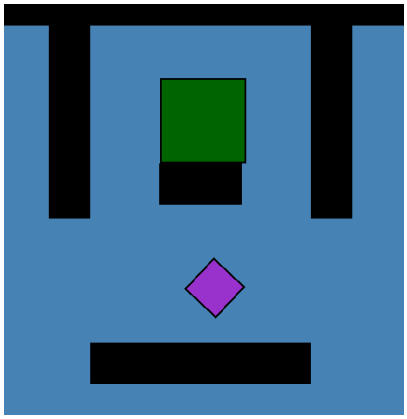


Figura 5.9: Fragmento del nivel 4 de la competición GF-IJCAI-ECAI 2022, en el que aparecen dos estructuras MONOSIDEDROP.

Movimientos MONOSIDEDROP

Los movimientos MONOSIDEDROP son muy similares a los movimientos DROP. En lugar de representar una situación en la que hay un hueco por el que puede dejarse caer el rectángulo entre dos plataformas, los MONOSIDEDROP modelizan la posibilidad de que el rectángulo caiga por el hueco entre una plataforma y una pared. Para ellos no es necesaria la creación de plataformas ficticias, pero por lo demás se comportan de la misma forma que los DROP en la fase de generación de movimientos. Sin embargo, en la fase de ejecución se tratarán de forma distinta ya que la heurística que funcionaba para los DROP ahora no es aplicable. Un ejemplo de ellos es el que se encuentra en la Figura 5.9. Anticipamos que el comportamiento que buscamos es que el rectángulo se ponga en forma vertical, se dirija hacia la pared con una velocidad alta y caiga pegado a la pared. En el siguiente enlace se puede ver una animación que clarifica lo descrito.

Movimientos ADJACENT

Estos movimientos representan la conectividad entre plataformas que están a la misma altura y que son adyacentes. Modelizan la posibilidad de pasar de una a otra, ya sean ambas plataformas reales o una de ellas ficticia. Para las plataformas simplificadas, los únicos movimientos de tipo ADJACENT que se tendrán en cuenta serán aquellos en los que una de las plataformas es ficticia. Esto se debe a que no hay plataformas simplificadas adyacentes (véase la Figura 5.4) ya que justamente estas plataformas habrán sido unificadas por ser adyacentes. Prescindir de estos movimientos en la fase de planificación, donde se utilizan las plataformas simplificadas, reduce el coste computacional. No obstante, a la hora de decidir la acción a ejecutar, conocer los movimientos ADJACENT entre dos plataformas reales no simplificadas sí será necesario.

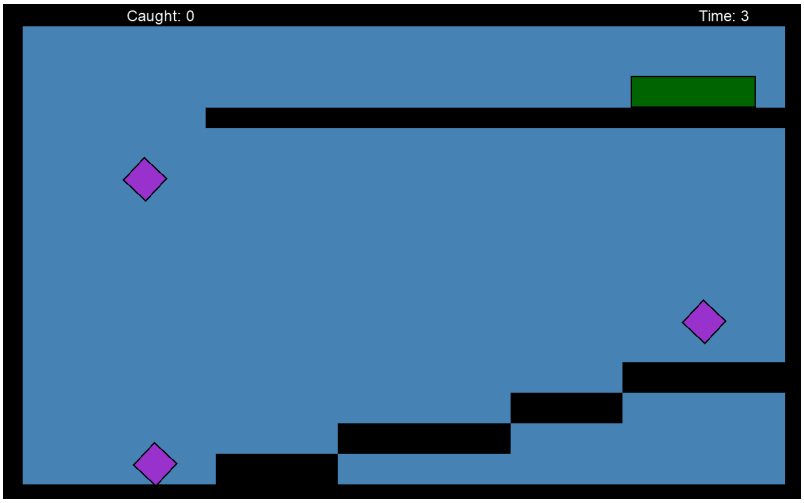


Figura 5.10: Nivel 7 de la competición del año 2014.

Figura 5.11: Secuencia del movimiento TILT.

Movimientos TILT

Los movimientos de tipo TILT son aquellos que modelizan la habilidad del rectángulo para "subir escalones" no muy grandes. Es ilustrativo mirar la animación del siguiente enlace. Estos movimientos se han añadido por la aparición de patrones como en el nivel de la Figura 5.10. Los movimientos de tipo TILT permiten al rectángulo moverse desde una plataforma origen a una plataforma destino que está a mayor altura. En esta fase de representación del nivel, únicamente nos tenemos que preocupar de añadir los movimientos que permiten moverse de una plataforma a otra, tan solo atendiendo a su existencia y sin preocuparnos en exceso sobre cómo se deben efectuar. Sin embargo, al igual que hicimos para los movimientos de tipo DROP, es conveniente tener una idea mental clara de cómo hay que proceder. El conocimiento heurístico humano nos dice que la mejor estrategia para afrontar la subida de un escalón es situarse en la forma vertical, deslizarse con suficiente velocidad hacia el escalón y dejar que el vértice del escalón haga de punto de apoyo que provoque que el rectángulo se voltee, como se puede ver en la Figura 5.11.

La pregunta que surge es, ¿cuál es la diferencia de altura máxima entre la plataforma de origen y la de destino para que sea posible realizar este movimiento? En este caso, la respuesta es significativamente más compleja de lo que pudiera parecer. Parece intuitivo que la diferencia de altura entre las plataformas debe ser menor que la mitad de la altura del rectángulo. Sin embargo, esto no siempre es así.

Hicimos un pequeño experimento, en el cual diseñamos un nivel con el editor que

viene incorporado en el juego. El nivel únicamente consistía en subir un escalón e, iterativamente, íbamos aumentando la diferencia de altura entre la plataforma inferior y la superior. Con el modo de jugador humano activado, comprobábamos si el rectángulo tenía realmente la capacidad de subir el escalón y, en tal caso, añadíamos una unidad a la diferencia de altura entre las plataformas. Llegó el momento en el que la diferencia de altura era exactamente la mitad del rectángulo y nuestra sorpresa fue que, en algunas ocasiones los jugadores humanos éramos capaces de completar el movimiento y otras veces no. Cabe destacar que siempre realizábamos las mismas acciones, que eran únicamente crecer a la altura máxima y mantener pulsada la tecla de deslizarse hacia la dirección del escalón.

Intrigados por este suceso, investigamos qué podría estar pasando y, gracias al sistema de depuración visual, donde comenzamos a mostrar todos de los parámetros del nivel, pudimos comprobar que la altura del rectángulo no era exactamente la misma en todas las ocasiones. A pesar de activar la acción MORPH_UP hasta que el rectángulo no podía crecer más, la altura máxima que alcanzaba era distinta de una ejecución para otra y también dentro de la misma ejecución. Creemos que se debe a una consecuencia de discretización dentro del propio juego y totalmente ajena a nosotros. La diferencia en las alturas del rectángulo no era muy grande, imperceptible para el ojo humano a simple vista y del orden de las unidades del píxel del juego, pero suficiente para que no nos permitiera completar el movimiento en todas las ocasiones.

Conscientes de ese problema, y siendo conservadores, concluimos que la diferencia de altura máxima entre las plataformas de un escalón debía ser la mitad de la altura del rectángulo menos una unidad. Quisimos comprobar este resultado en uno de los niveles que nos parecía que tenía un escalón de altura ligeramente menor a dicha altura máxima, pero, sin embargo, nuestra solución no conseguía generar el movimiento porque lo consideraba imposible, ya que, en realidad, la diferencia de altura era mayor que la mitad de la altura máxima. Como se trataba de un nivel de la competición oficial, el de la Figura 5.12, lo probamos nosotros mismos y pudimos comprobar que el nivel podía ser resuelto siempre. Esto nos descolocó todavía más. ¿Cómo podía ser posible que un escalón con una diferencia de altura mayor que la máxima pudiera subirse?

Ya habíamos comprobado que para una altura mayor que la máxima el rectángulo era incapaz de subir el escalón. En este caso, la novedad está en que, como se puede ver en la Figura 5.12, la plataforma a la que queremos subir tiene un hueco que, aparentemente, permite que el rectángulo se voltee al chocar con el escalón. Ante una teoría mejor, esto es lo que creemos que sucede, luego es el motor físico del juego el que nos está jugando una mala pasada.

Nos planteamos añadir una comprobación adicional que evaluara si el frontal del escalón era sólido o hueco, pero, cuando la probábamos en niveles auxiliares que nosotros mismos creábamos, no conseguíamos capturar aquello que en algunas ocasiones permitía que el escalón se pudiera subir y en otras no. Había veces que nuestro

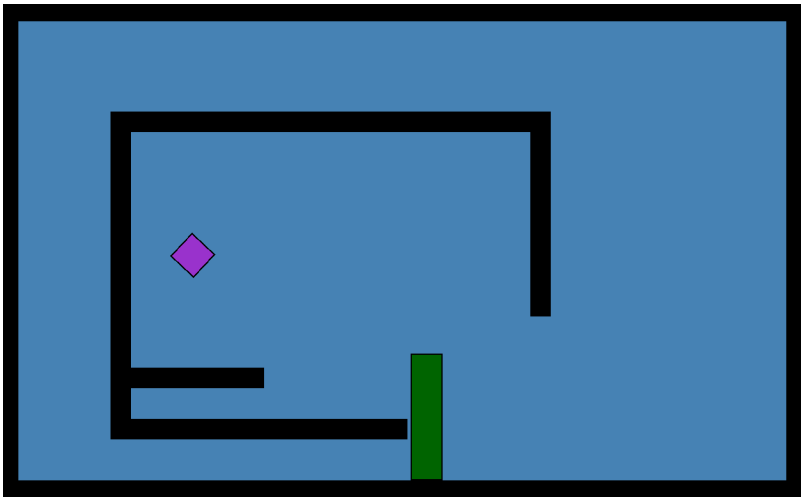


Figura 5.12: Nivel 6 de la competición del rectángulo del año 2020.

sistema afirmaba que se podían subir escalones que nosotros no éramos capaces de subir manualmente y al contrario. Por tanto, y de nuevo siendo conservadores, redujimos una unidad más la altura máxima de la diferencia entre plataformas, con lo que se consiguen identificar correctamente todos los movimientos TILT que se podían resolver de manera sencilla y garantizada.

Para resolver el de la Figura 5.12 y otros similares, decidimos separar estos movimientos y denominarlos como una categoría aparte, los movimientos HIGHTILT.

Movimientos HIGHTILT

Los movimientos HIGHTILT son muy parecidos a simple vista a los movimientos TILT. Sin embargo, el planteamiento para resolverlos es diferente y más complicado. Como hemos comentado en la sección anterior, estos movimientos se generan en las mismas situaciones que los TILT, pero cuando la diferencia de altura con la plataforma de destino es muy alta.

El procedimiento que intentaría seguir un humano consiste en coger la máxima carrerilla posible e ir hacia la esquina sobre la que voltearse estando en posición vertical, tal y como se muestra en la animación del siguiente enlace. Por tanto, optamos por seguir la misma estrategia. En la generación de movimientos, calculamos la máxima distancia que podía recorrer el rectángulo en forma vertical, (dejando un pequeño margen, como en la animación) y calcularíamos la velocidad que es capaz de alcanzar con esa distancia. Así, en ejecución, basta con decirle que llegue al punto de la esquina con la velocidad que hemos calculado, porque sabremos que posee la distancia que necesita para acelerar.

Sin embargo, estos movimientos acarrean varias complicaciones:

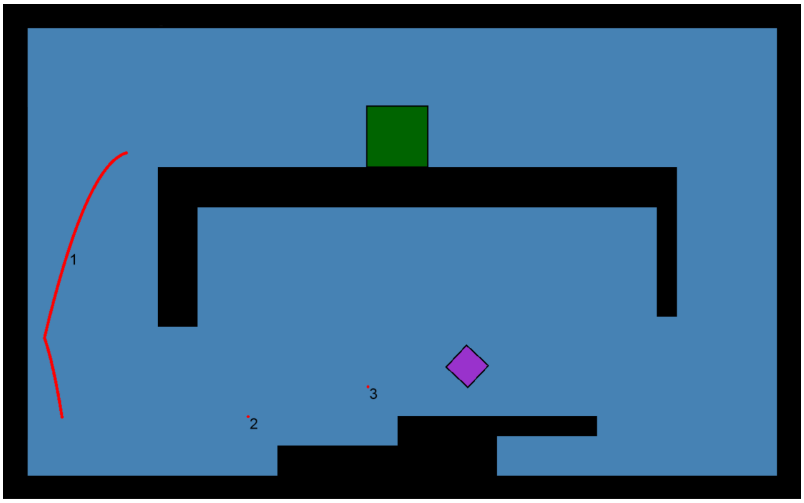


Figura 5.13: Nivel en el que el plan que se toma, formado por un movimiento FALL (dibujado como una línea continua) y dos movimientos TILT (marcados mediante un punto cada uno), no es el más corto posible porque evita el movimiento HIGHTILT.

- En primer lugar, esta estrategia no siempre tiene éxito. Si el rectángulo no es lo suficientemente alto (aunque el juego no le deje crecer más debido a su discretización interna), no será capaz de subir a la plataforma superior.

- Además, hay que tener en cuenta que se pueden estar generando movimientos imposibles, debido a la aparente aleatoriedad con la que el juego parece decidir si un movimiento HIGHTILT tendrá éxito o no. Por este motivo, recordando lo explicado en la sección 3.3.2, a esta clase de movimientos les activamos el bit `risky = true`, lo que quiere decir que se intentará evitar realizar esta clase de movimientos, siempre y cuando se encuentre otro plan que coja al menos el mismo número de diamantes. Tomando por ejemplo el nivel de la Figura 5.13, el rectángulo evita el camino de la derecha, pues, aunque es más corto (requiere solo dos movimientos), involucra un movimiento de tipo HIGHTILT, mientras que el camino de la izquierda, pese a tener tres movimientos, solo contiene un FALL y dos TILTs, que son más fáciles de ejecutar. Lógicamente, si para completar el nivel es necesario efectuar un movimiento HIGHTILT, el plan lo incluirá.

Normalmente, nuestro agente logra realizar siempre los movimientos HIGHTILT, aunque a veces le lleva muchos intentos, lo que lo hace más lento que un humano en este aspecto. Aun así, haber superado niveles que los involucran sin perjudicar el resto es una hazaña que ningún agente anterior había logrado, como ya veremos en el capítulo 7.

Movimientos FALL

Los movimientos de tipo FALL son los que modelizan las caídas desde los extremos de las plataformas (simplificadas), al igual que sucedía con el círculo (ver sección 4.2.1). Evidentemente solo consideramos las plataformas simplificadas porque las no simplificadas, en general, podrán tener plataformas adyacentes a los lados y no permitirán que el rectángulo se precipite por uno de sus extremos. La simulación de los movimientos es bastante parecida, pero tiene dos diferencias fundamentales que complican el caso del rectángulo.

La primera de ellas es que las acciones que el rectángulo toma en el aire afectan a su trayectoria. Si recordamos, cuando el círculo tomaba las acciones de ROLL_LEFT o ROLL_RIGHT durante su vuelo, el agente mantenía una trayectoria parabólica determinada únicamente por la velocidad horizontal inicial y la posición inicial. Se trataba de un movimiento uniforme en el eje de abscisas y un movimiento uniformemente acelerado en el eje de ordenadas. Las acciones únicamente afectaban al momento angular del círculo y provocaban que los choques con los obstáculos pudieran ser impredecibles.

El caso del rectángulo es diferente. Cuando el rectángulo toma las acciones MOVE_LEFT y MOVE_RIGHT durante el vuelo, es decir, en el aire y sin ningún contacto con ninguna plataforma, varía su velocidad horizontal, tal y como si se encontrara sobre una plataforma. Las acciones MOVE_LEFT y MOVE_RIGHT inducen una fuerza horizontal instantánea sobre el rectángulo, que provoca que este sufra una aceleración y cambie su posición y velocidad. Por tanto, mientras se tome continuamente la misma acción, el rectángulo se moverá en el eje de abscisas mediante un movimiento uniformemente acelerado, independientemente de si se encuentra en el aire o sobre una plataforma. Cuando esté sobre una plataforma, el movimiento en el eje de ordenadas será nulo, y cuando no esté apoyado en ninguna plataforma, la fuerza de la gravedad hará que en el eje de ordenadas el movimiento sea también uniformemente acelerado.

Por tanto, durante las caídas deberíamos considerar todas las combinaciones de acciones que el rectángulo puede tomar. Como esta secuencia de acciones crece exponencialmente con la longitud de la trayectoria, haremos la simplificación de que se da uno de los tres siguientes casos: el rectángulo toma siempre la acción NO_ACTION, el rectángulo toma siempre la acción MOVE_RIGHT o el rectángulo toma siempre la acción MOVE_LEFT. Entonces el movimiento del rectángulo, asumiendo que es una masa puntual y despreciando el rozamiento, viene determinado por las ecuaciones

$$\begin{cases} x(\mathtt{t}) = x_0 + v_x\mathtt{t} + \frac{a\mathtt{t}^2}{2} \\ y(\mathtt{t}) = y_0 + v_y\mathtt{t} - \frac{g\mathtt{t}^2}{2} \end{cases}$$

donde (x_0, y_0) es la posición inicial, v_x es la velocidad horizontal inicial, v_y es la velocidad vertical inicial y es 0 por tratarse de una caída y g es la aceleración de la gravedad. El parámetro a es la aceleración del movimiento en el eje horizontal

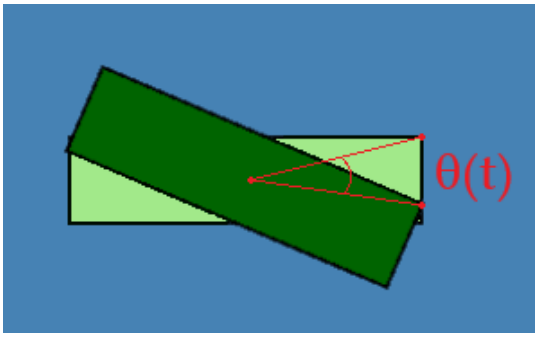


Figura 5.14: Orientación del rectángulo respecto a su posición inicial en función del ángulo $\theta(t)$.

y será 0 cuando la acción tomada sea siempre NO_ACTION (lo que recupera el caso de movimiento uniforme en el eje de abscisas del círculo), o $\pm|a|$, si la acción tomada es siempre MOVE_LEFT o siempre MOVE_RIGHT, siendo $|a|$ el módulo de la aceleración asociada a dichas acciones.

Como vemos, si bien las caídas del rectángulo son más complejas que las del círculo, en la práctica únicamente vamos a simular el triple de trayectorias, una para cada acción que se toma durante el vuelo. Además, las ecuaciones son igual de sencillas o complejas que en el otro caso. Sin embargo, la segunda diferencia que antes anticipábamos involucra un problema de modelización bastante más delicado.

En una caída, cuando el centro de masas del rectángulo abandona la plataforma, el rectángulo empieza a girar sobre sí mismo. Se recomienda acceder al enlace para visualizar una animación de lo que sucede. Se puede comprobar que el rectángulo rota a la vez que se desplaza. Esto no es distinto a lo que sucedía con el círculo, pero en ese caso el espacio físico que ocupaba el círculo en el nivel estaba únicamente determinado por la posición de su centro. En el caso del rectángulo, además de la posición de su centro, que está perfectamente determinada por las ecuaciones anteriores, hay que conocer la orientación del rectángulo o la posición de sus vértices.

A simple vista, no parece evidente cuál es la evolución de la orientación del rectángulo a lo largo del tiempo, es decir, cuál es la posición angular de sus vértices respecto al centro del rectángulo (ver Figura 5.14). Tras realizar diferentes pruebas, llegamos a dos conclusiones: la velocidad angular del rectángulo es constante y cuanto mayor es la velocidad inicial horizontal, menor es la velocidad angular. Por tanto, conjeturamos que la relación entre la velocidad angular y la velocidad horizontal inicial es inversamente proporcional, es decir,

$$\theta'(\mathtt{t}) = \frac{C}{v_x^{\alpha}}$$

para todo tiempo, con C y α constantes positivas por determinar. Para realizar una regresión que nos permitiera aproximar los valores de C y α realizamos una serie de experimentos para un conjunto de velocidades iniciales horizontales fijas. Comparamos visualmente la trayectoria real de los cuatro vértices del rectángulo

Velocidad inicial v_x	Mejor valor de β encontrado
100	1.25
200	0.5
300	0.3125
400	0.28

Tabla 5.1: Relación de los valores de v_x y el mejor valor de β asociado.

con la simulación de la trayectoria de los vértices en función de un parámetro β y elegimos el valor de β que haga que las trayectorias reales y simuladas coincidan (o estén lo más próximas posibles). Esta simulación consiste en situar para cada instante de tiempo el centro del rectángulo en la posición dada por las ecuaciones del movimiento y los vértices en la circunferencia de diámetro la diagonal del rectángulo y de tal forma que el ángulo entre la posición inicial del vértice y la actual sea exactamente $\theta(\mathtt{t}) = \beta\mathtt{t}$, donde β es el parámetro que variamos. Si el ángulo que forma inicialmente uno de los vértices con la base es $\theta_0 = \operatorname{arc\,tg}\left(\frac{h}{w}\right)$ donde h es la altura y w es la anchura, no es complicado deducir por trigonometría básica que la posición de dicho vértice a lo largo del tiempo es

$$\begin{cases} x(\mathtt{t}) = x_0 + v_x\mathtt{t} + \frac{a\mathtt{t}^2}{2} \pm r\cos(\theta_0 - \theta(\mathtt{t})) \\ y(\mathtt{t}) = y_0 + v_y\mathtt{t} - \frac{g\mathtt{t}^2}{2} \pm r\sin(\theta_0 - \theta(\mathtt{t})) \end{cases}$$

donde r es la mitad de la diagonal, es decir, $r = \frac{1}{2}\sqrt{h^2 + w^2}$ y el signo del seno y el coseno depende del vértice en cuestión. En la tabla 5.1 se encuentran los mejores valores de β para cada velocidad, es decir, aquellos para los que las trayectorias reales y simuladas de los vértices son muy parecidas. A modo de ejemplo, en el siguiente enlace mostramos la comparación entre la trayectoria simulada de los vértices y la trayectoria real del rectángulo para una velocidad inicial $v_x = 200$ y el mejor valor de β encontrado. Se puede observar que la aproximación no es perfecta, pero sí suficientemente buena.

Atendiendo a la tabla 5.1, podemos realizar la ya anunciada regresión, obteniendo que la función potencia que mejor aproxima los datos es $\theta'(\mathtt{t}) = \beta(v_x) = \frac{204{,}35}{v_x^{1{,}12}}$. La gráfica de la función junto a los puntos de regresión se encuentran en la Figura 5.15.

Tomando esa función, podemos simular la trayectoria del centro del rectángulo y de sus vértices. A partir de aquí, todo es igual que para el círculo: se discretiza el tiempo y en cada instante se comprueba si algún obstáculo interseca con el rectángulo. Si es así, se comprueba si el impacto se produce con uno de los lados o con un vértice. Si se produce con el vértice que tenga una menor altura en ese momento, entonces el movimiento termina porque se asume que se aterriza en una plataforma. Si se produce con cualquiera de los otros tres vértices se actualizan las velocidades y se sigue simulando. Por ejemplo, cuando la colisión se produce con el vértice derecho, entonces la velocidad horizontal se cambia de signo y se divide entre dos y la velocidad vertical se divide entre tres.

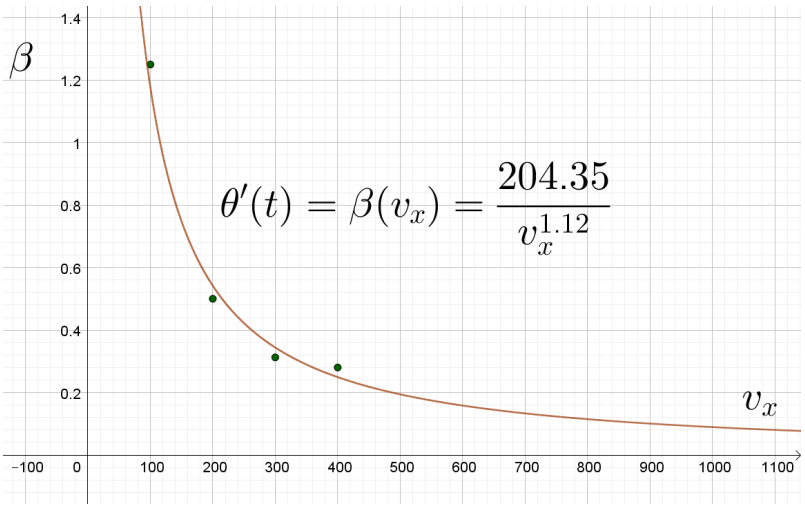


Figura 5.15: Representación de la función $\beta(v_x)$ junto a los puntos con los cuales se realiza la regresión.

Es importante destacar que, a partir del primer rebote, la velocidad angular pasa a ser siempre 0, pues resulta muy complicado actualizarla correctamente. Esta simplificación no ofrece malos resultados, ya que es muy infrecuente que se necesite de un movimiento que rebote al menos dos veces. Todas estas decisiones sobre cómo actualizar las velocidades para seguir simulando tras una colisión han sido determinadas empíricamente. Si se impacta con uno de los lados del rectángulo, entonces se descarta el movimiento porque eso significa que ha colisionado contra una esquina de un obstáculo y la trayectoria resultante es muy complicada de predecir.

Por último, al igual que hiciéramos con el círculo, debemos prescindir de las caídas que requieran de una velocidad horizontal inicial que es imposible de alcanzar. En la sección 4.2.1 calculamos cuál era el espacio necesario para, partiendo en reposo, lograr una velocidad v_x dada. Por tanto, para evaluar si es factible realizar las caídas, simplemente hay que comprobar que hay suficiente espacio en la plataforma simplificada para alcanzar la velocidad deseada.

Movimientos BIGHOLEADJ

Los movimientos BIGHOLEADJ son, en cierta forma, una mezcla entre los movimientos ADJACENT, FALL y TILT. Supongamos que nos encontramos con el patrón de la Figura 5.16, que llamaremos BIGHOLE (agujero grande). Se trata de dos plataformas a la misma altura, pero el espacio que hay entre ellas es demasiado grande como para que se genere una plataforma ficticia y sus movimientos DROP y ADJACENT asociados. Esto es intencionado, debido a que el rectángulo en su forma horizontal no es capaz de pasar de uno de los lados al otro, aunque coja mucha velocidad.

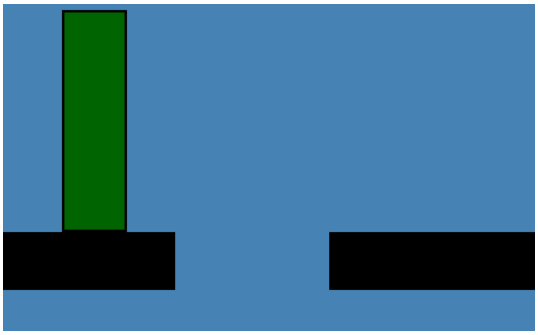


Figura 5.16: Patrón de la estructura BIGHOLE. Nótese que la longitud del hueco es más de la mitad de la anchura del rectángulo en forma horizontal y no se puede repetir la secuencia de la Figura 5.7.

Además, la simulación de las caídas hace que el punto de contacto entre el rectángulo y el obstáculo sea en uno de los lados del primero y, en una situación general, querremos descartarlo. Sin embargo, en esta situación, es la única forma de conectar las plataformas. Por eso, decidimos añadir este tipo de movimientos, que en ejecución serán parecidos a un FALL con la forma vertical y un TILT, que haga que el rectángulo se voltee y aterrice en la plataforma de destino en forma horizontal. Se puede acceder a una animación de los movimientos BIGHOLEADJ mediante el siguiente enlace.

Movimientos BIGHOLEDROP

Los movimientos BIGHOLEDROP son el análogo de los movimientos DROP para el patrón de estructura BIGHOLE, de forma similar a como los movimientos BIGHOLEADJ son el análogo de los movimientos ADJACENT. Sin embargo, estos movimientos son mucho más complicados de solventar. Dado que el hueco es demasiado grande, no sirve la misma estrategia que para los movimientos DROP, que consistía en colocar al rectángulo horizontal encima del hueco, apoyado en las dos plataformas de los lados (Figura 5.7). En esta ocasión, no hay una manera consistente de colocar al rectángulo en esta posición, lo que obliga a encontrar otra estrategia.

Al enfrentarnos nosotros mismos a la estructura BIGHOLE, no encontrábamos ninguna manera intuitiva de conseguir caer por el hueco. Por eso, decidimos que la mejor manera de solucionar estos movimientos era que el rectángulo aprendiera a ejecutarlos. Así, diseñamos el nivel de la Figura 5.17.

Durante las pruebas, nos dimos cuenta de que, por nuestra discretización, los huecos por los que resultaba difícil caer eran los que tenían una anchura de 14 o 15 cuadrados de 8×8 píxeles del juego (la anchura máxima del rectángulo son unos 24-25 cuadrados). Si tenían una anchura menor, podíamos construir una plataforma ficticia y efectuar un DROP. Si tenían una anchura mayor, el rectángulo detectaba

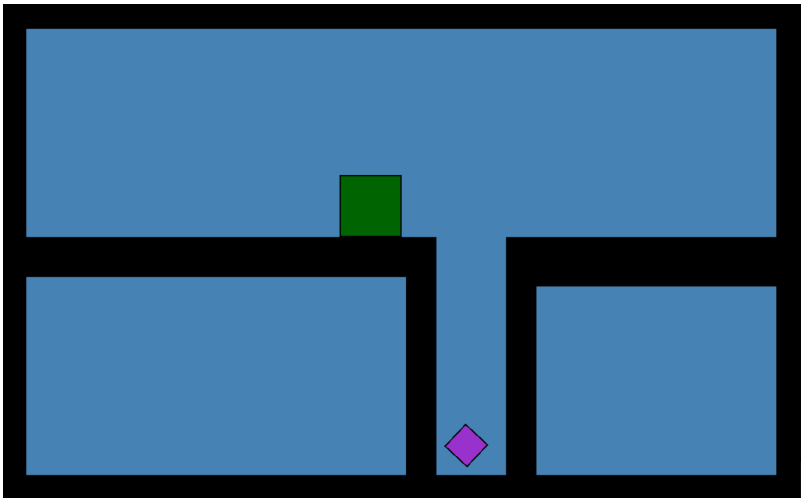


Figura 5.17: Nivel utilizado para el aprendizaje de los movimientos BIGHOLE-DROP.

que podía hacer un movimiento de tipo FALL sin mayores complicaciones.

Así, entrenamos por separado ambas anchuras (modificando la anchura del hueco del nivel de la Figura 5.17), ya que preveíamos (acertadamente) que las estrategias serían diferentes. Para el entrenamiento, optamos por la técnica de *Q-Learning*, al igual que habíamos hecho para el círculo (sección 4.3.3).

En la representación del estado, consideramos la distancia horizontal y vertical del centro del rectángulo a la esquina de la plataforma actual en la que está el hueco (llamamos a estos atributos *distance_x* y *distance_y*, respectivamente), la velocidad horizontal del rectángulo (*current_velocity*), la altura del rectángulo (*height*) y la anchura del hueco (*hole_width*). En la tabla 5.2, se muestran detalles de cómo se calculan y miden cada uno de estos atributos, así como un valor que lo ejemplifica para la configuración de la Figura 5.18, en la que la anchura del hueco es 14 y las distancias se miden con respecto a la esquina de color morado.

Con todo, la función que queremos maximizar es que *distance_y* sea menor o igual que 0, lo que significará que el centro del rectángulo está por debajo de la altura de la plataforma, luego este tiene al menos dos de sus vértices por debajo de la altura de la plataforma y, por tanto, caerá por el hueco. A los estados que cumplan esta propiedad se les asignará una recompensan de 500 y a los que no la cumplan, una recompensa de -1, para que el agente aprenda a caer en el menor tiempo posible. La forma en la que se eligen las acciones, cómo se actualizan los valores de la tabla Q y los valores concretos de los parámetros α, γ y ε son los mismos que los la sección 4.3.3, donde ya introdujimos la técnica del *Q-Learning*.

Realizamos una serie de simplificaciones para acelerar el entrenamiento:

- En primer lugar, solamente consideramos las distancias menores o iguales que

Atributo	Cómo se calcula	Positivo si	Negativo si	Valor en el ejemplo
Distance_x	Medida en cuadrados de 8×8 píxeles desde la esquina	El centro del rectángulo se encuentra a la derecha de la esquina	El centro del rectángulo se encuentra a la izquierda de la esquina	-5
Distance_y	Medida en cuadrados de 8×8 píxeles desde la esquina	El centro del rectángulo se encuentra encima de la esquina	El centro del rectángulo se encuentra debajo de la esquina	6
Current_velocity	Se redondea la velocidad real al múltiplo de 10 más cercano	El centro del rectángulo se mueve hacia la derecha	El centro del rectángulo se mueve hacia la izquierda	20
Height	Se divide la altura real entre 16 y se trunca el resultado	Siempre	Nunca	6
Hole_width	Medido en cuadrados de 8×8 píxeles desde la esquina	Siempre	Nunca	14

Tabla 5.2: Detalles relativos al cálculo del estado durante el entrenamiento de los movimientos BIGHOLEDROP, así como el valor resultante para la configuración de la Figura 5.18.

Figura 5.18: Configuración de ejemplo para un BIGHOLEDROP de anchura 14. Se muestran los cuadrados de 8×8 píxeles que hay entre el centro del rectángulo y la esquina de la plataforma actual en la que está el hueco, que aparece de color morado.

5 cuadrados de 8×8 píxeles. Si el rectángulo se encontraba más alejado, lo acercábamos, haciendo que llegara a una distancia de 5 cuadrados con una

velocidad baja y en forma de cuadrado. Esto ayudaba a que comenzara siempre en la misma posición, y es la estrategia que luego seguiríamos para ejecutar un movimiento BIGHOLEDROP[1].

- Por otro lado, si el rectángulo estuviera en la parte derecha del hueco (y no a la izquierda como en la Figura 5.18), podríamos realizar una simetría del estado de manera similar a como hacíamos en la sección 4.3.3. Esto quiere decir que, el estado que asociaríamos, y por tanto el que consultaríamos en la tabla Q, tendría cambiadas de signo la distancia horizontal (*distance_x*) y la velocidad horizontal (*current_velocity*), y cambiaríamos la acción MOVE_LEFT por MOVE_RIGHT y viceversa. De esta manera, reducimos a la mitad el número de estados a entrenar.

- Además, consideramos solamente los estados en los que el centro del cuadrado está encima de la esquina, es decir, $distance\_y \geq 0$, ya que si el centro está por debajo de la esquina, significa que hay al menos 2 vértices del rectángulo que ya han atravesado el hueco, de modo que el rectángulo caerá por su propio peso si no ejecuta ninguna acción.

- Finalmente, asignamos un tiempo límite de 15 segundos al nivel con el hueco de 15 cuadrados y 20 segundos al nivel con el hueco de 14 cuadrados (que es un movimiento más complicado de ejecutar). Esto provocaba que el nivel acabara rápidamente si el rectángulo se quedaba estancado en el hueco o no progresaba, acelerando así el tiempo entre ejecuciones. Además, le incentivaba a aprender a caer lo más rápido posible por el hueco. Solamente actualizábamos la tabla Q si el nivel terminaba, ya fuera porque el rectángulo caía y cogía el diamante o porque se le acababa el tiempo límite.

Tras entrenar con estas características, realizamos una prueba para ver el porcentaje de éxito resultante. La implementación resultante de este aprendizaje tenía limitaciones, y no siempre consigue caer por el hueco, sino que muchas veces sigue quedándose estancado, como en la Figura 5.19.

Parte de la razón por la que sufrimos estas limitaciones es debido a que el juego no proporciona en ningún momento la orientación que tiene el rectángulo (inclinación respecto al eje horizontal), sino simplemente su posición, velocidad y altura. Dado que no podemos incluir esta orientación en el estado, las configuraciones de la Figura 5.20, suponiendo que el rectángulo lleva una misma velocidad (nula) en ambas, son

[1]La única diferencia sería que, aunque el rectángulo acabaría con velocidad baja cuando llegue a una distancia de 5 cuadrados, antes de llegar a esa posición iría lo más rápido posible para emplear menos tiempo.

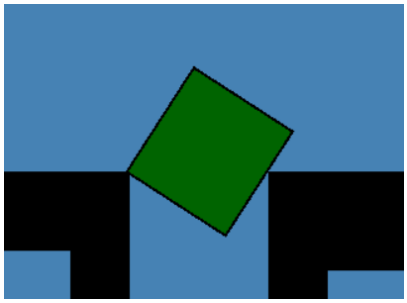


Figura 5.19: Configuración en la que el rectángulo se ha quedado estancado intentando realizar un BIGHOLEDROP, siendo incapaz de cambiar de estado.

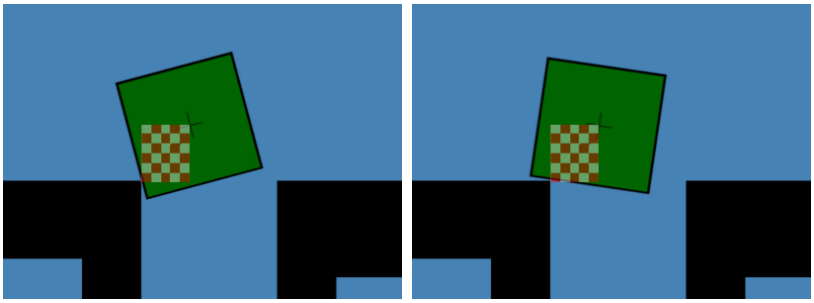


Figura 5.20: Configuraciones diferentes a las que se les asocia el mismo estado. Suponiendo una velocidad horizontal nula, la configuración izquierda consigue caer por el hueco, mientras que la derecha se queda estancada.

indistinguibles. Es decir, se les asocia el mismo estado, que es:

$$\begin{cases} \text{Distance\_x} & \rightarrow & 5 \\ \text{Distance\_y} & \rightarrow & 6 \\ \text{Current\_velocity} & \rightarrow & 0 \\ \text{Height} & \rightarrow & 6 \\ \text{Hole\_width} & \rightarrow & 15 \end{cases}$$

Sin embargo, suponiendo una velocidad horizontal nula, la configuración izquierda consigue caer por el hueco, mientras que la derecha se queda estancada. En consecuencia, no se puede determinar si este estado es deseable o no, ni cuál es la mejor acción a realizar.

Aun así, nuestro agente logra caer por el hueco un 26 % de las veces si la anchura del hueco es 14 cuadrados y un 62 % de las veces si es de 15 cuadrados (lógicamente es mucho más sencillo quedar atascado cuando el hueco es más estrecho). Sin embargo, dado que no podemos garantizar que este sea un movimiento que se realice siempre con precisión, activamos su bit `risky = true`. Recordando lo explicado en la sección 3.3.2 y en la sección 5.2.2, tener este bit activo quiere decir que se intentará evitar realizar esta clase de movimientos, siempre y cuando se encuentre otro plan que coja al menos el mismo número de diamantes. Esto puede verse, por ejemplo, en la Figura 5.21, en la que el plan que se elige es más largo que el que realiza un

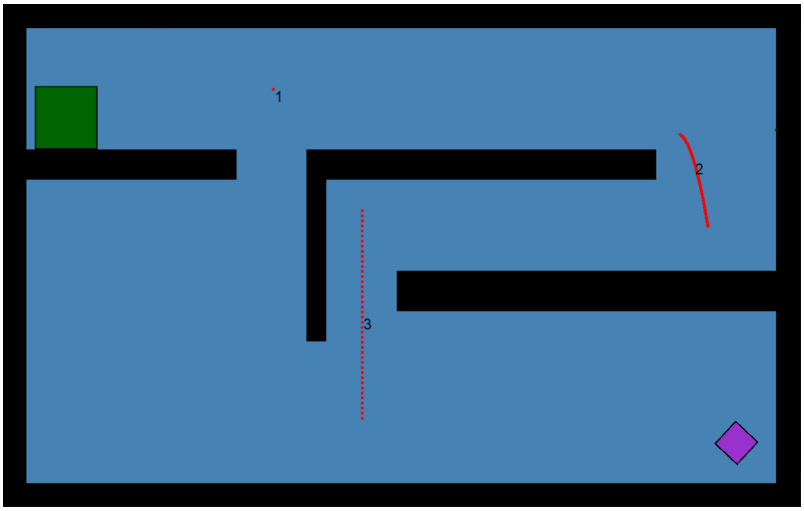


Figura 5.21: Nivel en el que el plan que se toma, formado por un movimiento BIGHOLEADJ (representado con un punto), un movimiento FALL (marcado con una línea continua) y un movimiento MONOSIDEDROP (dibujado como una línea discontinua), no es el más corto posible porque evita el tener que realizar un movimiento BIGHOLEDROP.

solo movimiento de tipo BIGHOLEDROP para caer directamente a la plataforma inferior. Como ya vimos en los movimientos HIGHTILT, priorizamos la consistencia sobre la velocidad.

5.2.3. Filtrado de movimientos

De manera similar a lo que sucedía con el círculo, la fase de generación de movimientos produce una gran cantidad de movimientos, siendo algunos de ellos equivalentes. Para reducir los costes computacionales de la posterior búsqueda, es necesario restringirnos a un subconjunto de movimientos que garantice que se puedan recoger todos los diamantes alcanzables y que haya conectividad entre las plataformas. Mientras que en el caso del círculo la redundancia de movimientos se producía, en mayor medida, porque dos movimientos del mismo tipo con parámetros distintos eran equivalentes, en el caso del rectángulo la duplicidad también afecta notablemente a movimientos de tipos diferentes. En esta sección, explicaremos los criterios que seguimos para guardar aquellos más convenientes.

Al igual que hacíamos con el círculo, inicialmente descartamos aquellos nuevos movimientos que no nos sirvan para cambiar de plataforma o alcanzar algún diamante. Estos serán los NOMOVE que no recojan diamantes, ya que los demás tipos de movimientos llevan asociado un cambio de plataforma. Si el nuevo movimiento supera este primer filtro, se compara con todos los movimientos ya añadidos para evaluar si puede sustituir a uno (o varios) de los anteriores.

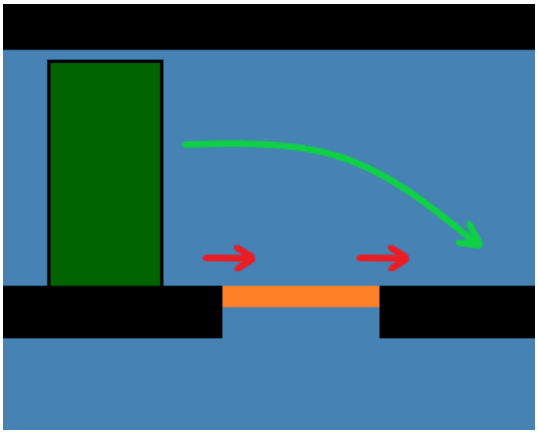


Figura 5.22: Estructura de tipo DROP donde hay que prescindir del movimiento FALL espurio (verde) frente a los movimientos ADJACENT (rojos).

El primer caso que trataremos es aquel en el que los movimientos comparados no tienen plataformas de partida o aterrizaje iguales. Aunque pudiera parecer que es bueno conservar ambos movimientos, lo cierto es que esto no siempre es así.

Supongamos que nos encontramos una estructura de tipo DROP, como la de la Figura 5.22. Ya hemos visto que se generarán movimientos de tipo ADJACENT que conecten la plataforma ficticia con las plataformas reales a su izquierda y su derecha (los movimientos rojos). Lo cierto es que también se pueden generar movimientos de tipo FALL espurios entre las dos plataformas reales y que son indeseados (movimiento verde). En la fase de ejecución, si tenemos movernos entre las plataformas reales, preferiremos hacerlo simplemente deslizando el rectángulo en forma horizontal antes que realizando una caída con las a veces impredecibles colisiones que puedan surgir.

Por tanto, cuando se compara un movimiento de tipo ADJACENT y otro de tipo FALL, hay que evaluar si existe otro movimiento de tipo ADJACENT que conforme una situación como la de la Figura 5.22, en cuyo caso se eliminará el movimiento de tipo FALL, conservando los dos de tipo ADJACENT. Nótese que el movimiento ADJACENT de la izquierda tiene como plataforma destino la ficticia (mientras que el movimiento de tipo FALL tiene como destino la de la derecha) y el movimiento de tipo ADJACENT de la derecha tiene como origen la plataforma ficticia (mientras que el movimiento de tipo FALL tiene como origen la de la izquierda).

Algo similar sucede con movimientos de tipo FALL y DROP, como se aprecia en la Figura 5.23. Por conveniencia, la plataforma origen del movimiento DROP es la ficticia y se pueden generar movimientos de tipo FALL indeseados que ofrezcan la misma conectividad que un movimiento DROP. Sin embargo, los movimientos de tipo DROP son mucho más estables, más seguros de ejecutar y su resultado no varía al perturbar levemente las condiciones iniciales. En consecuencia, y siguiendo el mismo criterio que en el caso anterior, prescindiríamos del movimiento FALL espurio (verde), conservando los movimientos de tipo ADJACENT (rojo) y DROP (amarillo).

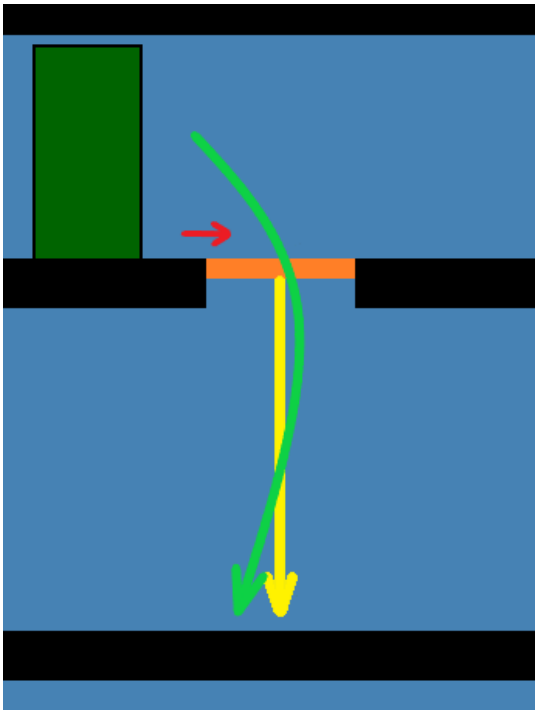


Figura 5.23: Estructura de tipo DROP donde hay que prescindir del movimiento FALL espurio (verde) frente a los movimientos ADJACENT (rojo) y DROP (amarillo).

Una vez tratados estos dos casos algo más especiales, en los restantes ya sí que asumimos que las plataformas de partida y de llegada son las mismas en los dos movimientos a comparar porque, si no, los movimientos son incomparables y se deberán almacenar por el momento ambos. Siguiendo con la filosofía que llevábamos hasta ahora, vamos a seguir eliminando otros movimientos de tipo FALL para los que existe otro tipo de movimiento que los modelice mejor. Por ejemplo, si en una estructura de tipo MONOSIDEDROP se generan movimientos de tipo FALL y de tipo MONOSIDEDROP prescindiremos de los primeros porque los segundos se han creado específicamente para modelizar esa situación. Lo mismo sucede con los movimientos BIGHOLEDROP frente a los de tipo FALL.

Si ambos movimientos son de tipo NOMOVE, entonces hay que evaluar sus conjuntos de diamantes capturados. Si uno está contenido en otro, guardamos el movimiento que alcance más diamantes, y si no, son incomparables. Cuando alcanzan exactamente los mismos diamantes, atendemos a la forma asociada al movimiento. Preferiremos el movimiento con una forma más horizontal, porque en esa posición la estabilidad del rectángulo es máxima. Si tienen la misma forma, nos quedamos con el que tenga una posición más próxima a la localización real del diamante, al igual que hacíamos en el caso del círculo.

Nos queda por tratar el caso en el que ninguno de los dos movimientos es de tipo NOMOVE y no entra dentro de los casos ya expuestos. Como antes, comparamos los conjuntos de diamantes y seguimos el mismo razonamiento. Cuando los diamantes

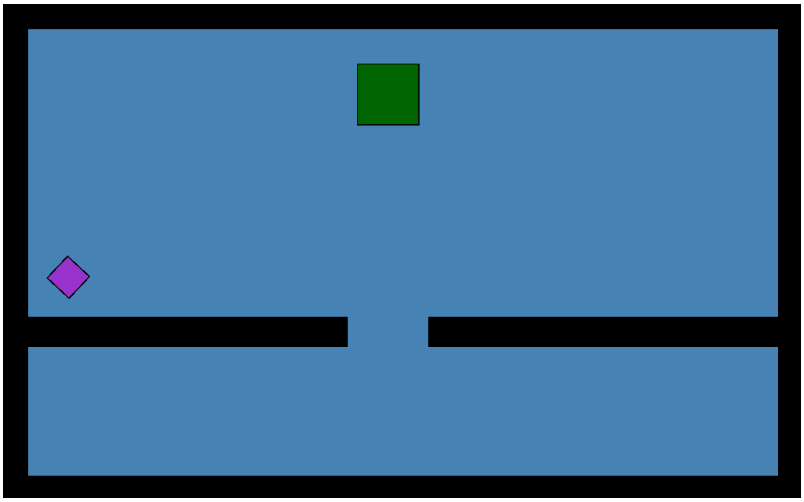


Figura 5.24: Nivel 7 de la competición GF-IJCAI-ECAI 2022.

alcanzados son exactamente los mismos, elegimos la forma más horizontal, y si tienen la misma forma, valoramos los movimientos mediante una función de valor que nos indica la bondad de un movimiento. Esta función es la misma que en el caso del círculo (ver la última parte de la sección 4.2.2), que recordamos que tomaba en consideración parámetros como la velocidad del movimiento, la posición del punto de aterrizaje en la plataforma de llegada o el punto de partida del movimiento. En el caso del rectángulo, pasamos más por alto esta función de valor, ya que no es tan determinante como en el círculo. Esto se debe a que, para el rectángulo, es significativamente más complicado caer de forma involuntaria por los extremos de la plataforma o chocar con un obstáculo de manera imprevista. Recordemos que muchos de esos problemas surgían a raíz de los saltos del círculo y el rectángulo no tiene ninguna acción de salto.

Esto completa nuestro filtrado de movimientos, que, si bien no es tan determinante para reducir el coste computacional como en el caso del círculo, sí que nos permite seleccionar de manera precisa los movimientos que se van a tener en cuenta posteriormente en la fase de actuación. Con ello, finalizamos también la sección relativa a la representación del nivel del rectángulo, que se ha centrado en explicar las particularidades de las plataformas del rectángulo y la generación de movimientos. En lo que resta de capítulo, abordaremos la realización de un plan a seguir y la ejecución de dicho plan.

5.3. Trazado de un plan a seguir

El trazado de un plan, en el caso del rectángulo, no consiste en una única llamada al algoritmo de búsqueda. Para ilustrarlo, consideremos el nivel de la Figura 5.24, de una de las competiciones de 2022.

En este nivel, el rectángulo comienza en el aire. No solo eso, sino que, además, si no efectúa ninguna acción pasados un par de segundos, no será capaz de completar el nivel, ya que caerá a la plataforma de abajo. Hay dos maneras de completar el nivel, y ambas involucran realizar una acción en el aire. La manera óptima es realizar la acción de MOVE_LEFT, y es en la que nos vamos a centrar, ya que la otra es realizar la acción de MORPH_DOWN, pero, si el hueco fuera más ancho, realizar esta acción no serviría de nada. Por tanto, ya que queremos generalizar para la mayor cantidad de niveles posibles, debemos encontrar una forma de que nuestro agente quiera realizar la acción MOVE_LEFT.

Al ejecutar el algoritmo de búsqueda por primera vez, como la plataforma inicial que se detecta es la de abajo del todo (se toma la primera plataforma real que se encuentra debajo del rectángulo), el plan que se devuelve es lógicamente vacío. Además, como ese plan vacío no captura todos los diamantes[2], la variable *plan_es_completo* que introdujimos en el algoritmo 1 se mantiene a `false`.

Para solucionar los niveles que requieren ejecutar acciones nada más empieza el tiempo, decidimos que, si tras esta primera llamada al algoritmo de búsqueda, el plan que se devuelve no es completo, probaríamos a simular en qué plataformas caería el rectángulo al realizar siempre la acción MOVE_LEFT o MOVE_RIGHT. En caso de aterrizar en plataformas diferentes, lanzaríamos nuevamente el algoritmo de búsqueda desde cada una de ellas. Si uno de los planes resultantes es completo, nos quedamos con ese, y, si ambos lo son, elegimos el más corto, tomando el plan que realiza la acción MOVE_LEFT si son igual de largos. De esta manera, conseguimos solucionar estas situaciones, ya que le indicamos al rectángulo que haga la acción que corresponda, y este la mantendrá hasta que detecte que está sobre una plataforma.

5.4. Ejecución del plan

Una vez hemos representado internamente el nivel y hemos construido un plan, compuesto de una serie de movimientos que nos permiten cambiar de plataforma, tan solo queda que el rectángulo ejecute dicho plan. El algoritmo que guía la actuación del rectángulo es el algoritmo 4. Este es un pseudocódigo muy simplificado del algoritmo real pero que nos ofrece una idea clara de cómo actúa el rectángulo. En las siguientes secciones desgranaremos todos los detalles de interés del algoritmo y otros que no se incluyen en él.

[2]Nótese que esto no se deduce de que el plan sea vacío, ya que, si el diamante se pudiera coger desde la plataforma inferior, el plan sería vacío pero completo.

Algoritmo 4 Algoritmo de actuación del rectángulo

```
1: while El juego no ha terminado do
2:     if Platf. simpl. actual != Platf. (simpl.) origen del 1er paso del plan then
3:         Replanificar
4:     end if
5:     if Quedan diamantes por coger en la plataforma simplificada actual then
6:         m ← movimiento que alcanza el diamante más cercano
7:     else
8:         if El plan es no vacío then
9:             m ← primer movimiento del plan
10:        else
11:            Realizar una acción aleatoria
12:        end if
13:    end if
14:    /*m es el siguiente movimiento a realizar*/
15:    S ← Mejor forma de rectángulo para completar el movimiento m
16:    if Forma del rectángulo != S then
17:        Realizar la acción que más acerca al rectángulo a la forma S
18:    else
19:        if Pos. rectángulo ≉ Pos. m || Vel. rectángulo ≉ Vel. m then
20:            Realizar la acción que más acerca al rectángulo a (Pos. m, Vel. m)
21:        else
22:            Realizar la acción asociada al tipo de movimiento
23:        end if
24:    end if
25: end while
```

5.4.1. Replanificación

La replanificación en el rectángulo no aporta nada nuevo con respecto a lo explicado en las secciones 3.3.3 y 4.3, salvo las llamadas al algoritmo de búsqueda en el caso de que sea necesario realizar alguna acción nada más comenzar el nivel, como en el de la Figura 5.24, que ya detallamos en la sección anterior. Por tanto, no creemos que merezca la pena volver a detallar el comportamiento de la replanificación.

5.4.2. Consideraciones sobre la forma del rectángulo

Una de las características distintivas del rectángulo es su habilidad para cambiar de forma. Muchas veces, qué forma tomemos va a determinar qué movimientos vamos a poder realizar.

Para empezar, vamos a justificar el motivo por el que debemos tener precaución sobre qué forma debe tener el rectángulo antes de realizar ninguna acción que des-

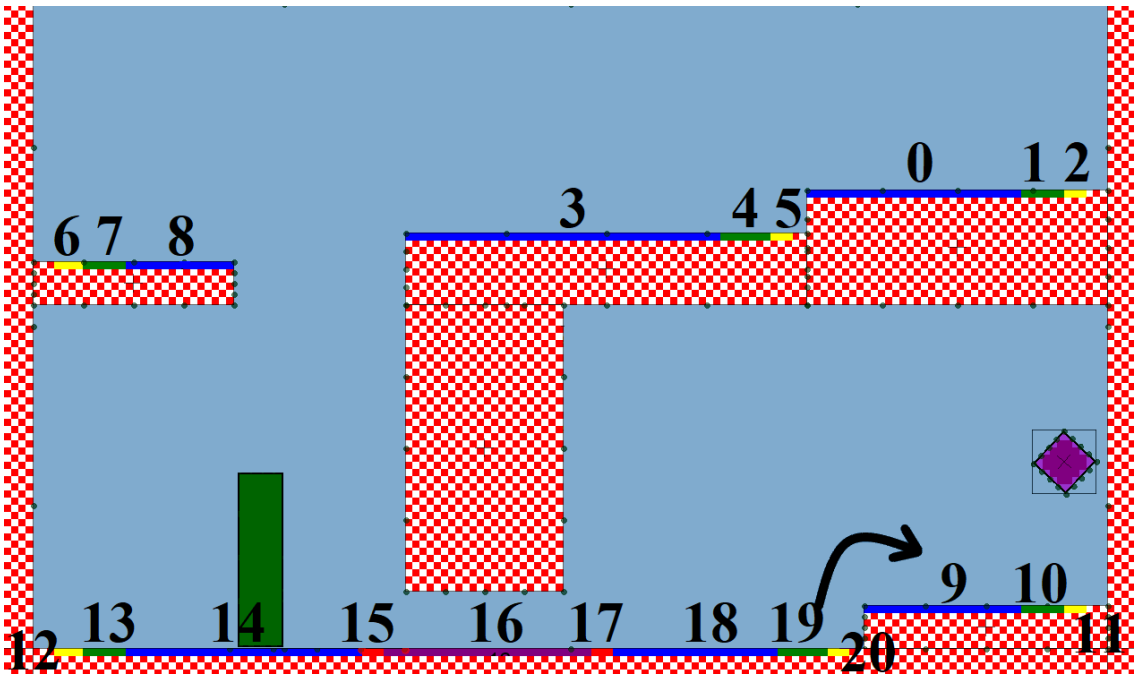


Figura 5.25: Representación del nivel 8 del año 2016.

place al rectángulo dentro de una plataforma simplificada. Como hemos comentado antes, la planificación se realiza con las plataformas simplificadas, ya que son muchas menos y reducen el coste computacional de la búsqueda. Pero dentro de una misma plataforma simplificada, el rectángulo puede encontrarse en una plataforma (no simplificada) distinta a la asociada a la posición del siguiente movimiento, como en el ejemplo de la Figura 5.25.

En la figura, se aprecia que el rectángulo y la posición del próximo movimiento (el TILT que permite subir a la plataforma simplificada donde se encuentra el último diamante) están en distintas plataformas, pero en la misma plataforma simplificada. Recordamos que la diferencia entre las plataformas (no simplificadas) y las plataformas simplificadas es que las primeras tienen un conjunto de formas admisibles para el rectángulo, mientras que las segundas ignoran las formas. Además, las plataformas simplificadas se crean por yuxtaposición de plataformas (no simplificadas) adyacentes horizontalmente, como vimos en la sección 5.2.1. Por lo tanto, cuando queremos movernos dentro de una plataforma simplificada, no es tan sencillo como simplemente tomar las acciones MOVE_RIGHT o MOVE_LEFT. También hay que comprobar que, cuando el rectángulo pasa de una plataforma a otra, lo hace con una forma admisible en ambas plataformas.

Teniendo esto en mente, nos encontramos dos posibilidades. Si el rectángulo se encuentra en la misma plataforma en la que se inicia el siguiente movimiento, entonces la forma que debe adoptar el rectángulo es la asociada al movimiento. Por ejemplo, para los movimientos de tipo DROP, la forma horizontal; para los de tipo TILT, la vertical; y para los de tipo FALL, aquella con la que se hubiera simulado el movimiento.

La segunda posibilidad es que el rectángulo no se encuentre en la misma plataforma que el origen del movimiento, como comentábamos antes. En ese caso, hay que estudiar cuál es la forma más adecuada, para lo cual tenemos una política sencilla. Lo primero que hacemos es comprobar si, dada la forma del siguiente movimiento, podemos mantener dicha forma desde la plataforma actual del rectángulo a la del movimiento. Retomando el ejemplo de la Figura 5.25, el rectángulo se encuentra en la plataforma 14 y la forma del siguiente movimiento es vertical (porque es un TILT). Sin embargo, no puede mantener la forma vertical desde su plataforma hasta la 20, ya que la plataforma 16 solamente admite la forma horizontal. No obstante, desde la plataforma 17 sí que podría mantener la forma vertical hasta la 20. Está política nos permite evitar cambiar repetidamente de forma cuando es innecesario. Asimismo, permite adoptar la forma del movimiento con cierta antelación, asegurando que, cuando el rectángulo llegue al punto del movimiento, lo hará con la forma correcta.

Si no pudiera mantener la misma forma durante todo el recorrido, únicamente evaluamos la siguiente plataforma. Si la siguiente plataforma admite la forma actual del rectángulo, la mantenemos, siguiendo la filosofía de cambiar de forma solamente cuando sea estrictamente necesario. Si esto último no es posible, elegimos la primera forma de la secuencia "cuadrado, horizontal, vertical" que sea admisible tanto en la plataforma actual como en la siguiente.

Una vez se ha seleccionado lo que en el algoritmo 4 hemos llamado la "mejor forma para completar el movimiento", hay que tomar la acción que hace que el rectángulo adopte dicha forma objetivo. Evidentemente, si el rectángulo ya tiene la forma objetivo pasaremos directamente a la siguiente fase del algoritmo. En caso contrario, la acción adoptada será MORPH_UP cuando la altura de la forma objetivo sea mayor que la altura actual del rectángulo, y MORPH_DOWN en caso contrario.

Con esta política, conseguimos llegar a la posición del movimiento con la forma del movimiento, es decir, totalmente vertical, totalmente horizontal o cuadrado. El problema que nos surge ahora es que en algunas ocasiones ninguna de estas tres formas es válida. Este problema se da principalmente con los movimientos TILT en algunos niveles de competiciones de años anteriores. Tomemos por ejemplo el nivel 5 del año 2013, que se muestra en la Figura 5.26. Al realizar un movimiento de tipo TILT, partiendo desde una forma totalmente vertical, el rectángulo queda atascado, como se aprecia en la Figura 5.27. Nuestra discretización en tres formas básicas, si bien nos ha simplificado mucho el trabajo, nos perjudica en este caso por no tener suficiente granularidad.

Para realizar el movimiento de tipo TILT, el rectángulo debe adoptar una forma cuya altura sea algo menor que la máxima. Sin embargo, dicha altura no puede ser demasiado baja ya que, en ese caso, el rectángulo no podrá voltearse por tener su centro de masas demasiado bajo y ser más estable. Por tanto, tenemos que calcular la máxima altura de tal forma que, durante el movimiento TILT, el rectángulo no colisione con su parte superior con ningún obstáculo. Para ello, tenemos que volver a utilizar conocimientos de geometría básica.

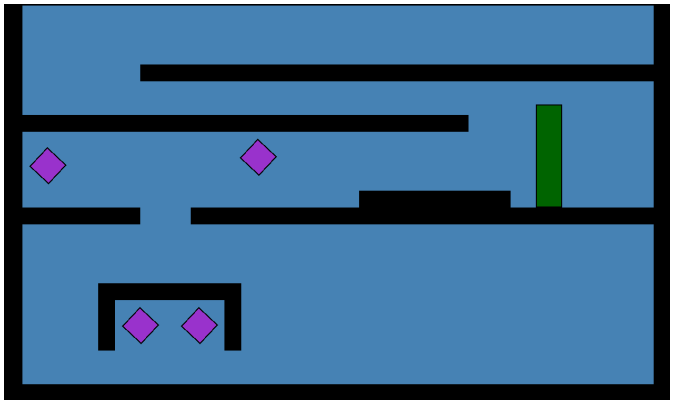


Figura 5.26: Nivel 5 de la competición del rectángulo del año 2013.

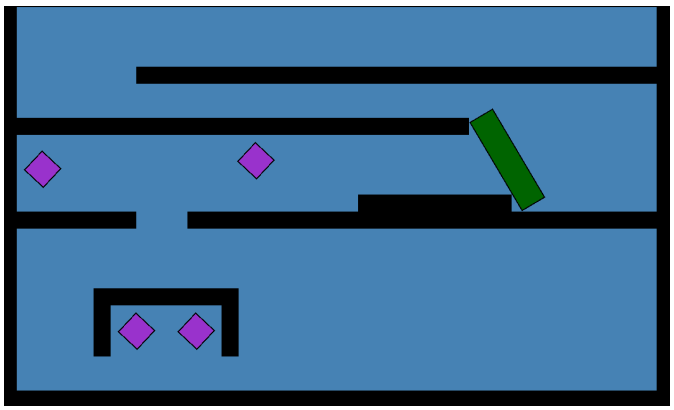


Figura 5.27: Rectángulo atascado al realizar el movimiento TILT.

Lo que vamos a realizar es un bucle en el que vamos decrementando la altura h del rectángulo desde su máximo hasta una unidad más de la altura del cuadrado. Para cada valor de h, simulamos si el rectángulo, realizando el TILT con altura h, colisiona con algún obstáculo. Para entender la simulación, es conveniente observar la Figura 5.28 (cuando el escalón está a la derecha, el procedimiento es simétrico).

En la figura, podemos observar la trayectoria del rectángulo a lo largo del tiempo. El rectángulo tiene altura h, que es el parámetro que estamos decrementando y que queremos optimizar. Como su área A es constante, la base está determinada y tiene longitud fija $w = A/h$. Si la altura del escalón es a, entonces el radio de la circunferencia descrita por el vértice superior izquierdo del rectángulo es $r_1 = h - a$. Análogamente, el radio de la circunferencia descrita por el vértice superior derecho es $r_2 = \sqrt{r_1^2 + w^2}$. Con esto, si fijamos el origen de coordenadas en el vértice del escalón, la posición inicial de los vértices superiores izquierdo y derecho son respectivamente $Q(0) = (r_1 cos(\theta_1), r_1 sin(\theta_1))$ y $P(0) = (r_2 cos(\theta_2), r_2 sin(\theta_2))$ donde $\theta_1 = \pi/2$ y $\theta_2 = arctan(h/w)$. Es inmediato verificar que la trayectoria que recorren los vértices se puede describir por las curvas

$$\begin{cases} P(\theta) &= (r_1 cos(\theta_1 + \theta), r_1 sin(\theta_1 + \theta)), \quad \theta \in (0, \pi/2) \\ Q(\theta) &= (r_2 cos(\theta_2 + \theta), r_2 sin(\theta_2 + \theta)), \quad \theta \in (0, \pi/2). \end{cases}$$

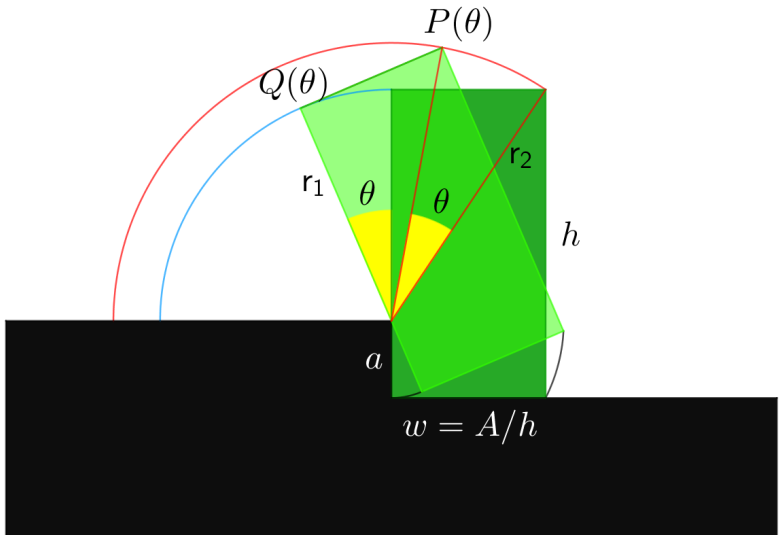


Figura 5.28: Análisis de las colisiones de los movimientos TILT variando la altura.

Tras estos cálculos, únicamente hay que comprobar, para una discretización de θ, si los píxeles en los que se encuentran $P(\theta)$ y $Q(\theta)$ tienen tipo distinto de OBSTACLE y PLATFORM, es decir, si la trayectoria colisiona o no con algún objeto.

Con este proceso, conseguimos nuestro objetivo de calcular la altura máxima que puede tener el rectángulo para realizar un movimiento de tipo TILT sin quedarse atascado. Si esta altura no es la altura máxima que puede tener el rectángulo, no coincidirá con la altura de ninguna de las tres formas básicas de la Figura 5.1. Por tanto, para integrar este caso a la situación general, formalmente se debería haber hablado durante toda esta sección de "mejor altura" o "altura objetivo", para referirnos a la altura de la mejor forma o la forma objetivo.

5.4.3. Acción que más acerca al rectángulo a la posición del movimiento

En el apartado anterior, explicamos cómo se tratan las formas, cuál es la política que hay que llevar para pasar de una a otra, y cómo solucionamos la problemática de los movimientos de tipo TILT, evitando así que el rectángulo quede atascado. Nuestro objetivo ahora es conseguir que el rectángulo consiga desplazarse hacia una posición dentro de la misma plataforma simplificada y que llegue con una velocidad determinada, olvidando los cambios de forma, ya que eso lo podemos solventar aplicando lo explicado en el apartado anterior.

La estrategia es muy similar al caso del círculo. Si recordamos, en la sección 4.3.3 expusimos dos alternativas para escoger la acción que más acercaba al rectángulo a una posición y velocidad dada. Una era un sistema de reglas que tenía en cuenta la velocidad actual y la posición actual y, mediante las ecuaciones del movimiento,

calculábamos puntos de interés que llamábamos punto de frenado y punto de aceleración. Estudiando el signo de la velocidad y discutiendo los casos en función de qué puntos estaban a la izquierda o a la derecha llegábamos a un sistema de reglas que nos permitía tomar la acción (en el caso del círculo ROLL_RIGHT o ROLL_LEFT) que más acercaba al círculo a su posición y velocidad objetivo.

La otra alternativa era que el propio círculo aprendiera la acción que debía tomar mediante un algoritmo de aprendizaje por refuerzo, concretamente, mediante Q-learning. Para el rectángulo, hemos optado por utilizar únicamente el sistema de reglas. Esto se debe a que las reglas son prácticamente las mismas a las del círculo. Únicamente hay que tener la precaución de cambiar las acciones asociadas al desplazamiento del círculo (ROLL_RIGHT y ROLL_LEFT) por las análogas para el desplazamiento del rectángulo (MOVE_RIGHT y MOVE_LEFT). Asimismo, las constantes de aceleración no son iguales para el círculo que para el rectángulo, pero simplemente hay que calcular estas aceleraciones del rectángulo con otro experimento. En esencia, todo se reduce a traducir lo que hicimos para el círculo en la sección 4.3.3 y creemos que no merece la pena volver a detallarlo todo de nuevo. Simplemente apuntamos que este sistema de reglas trataba de ir todo lo rápido que pudiera en cada momento y frenar lo más tarde posible, con el objetivo de ahorrar tiempo.

Elegir el sistema de reglas evita tener que entrenar al agente una vez más, ya que no nos vale la tabla Q aprendida por el círculo, pues esta se ajustaba a la aceleración propia del círculo, que ya hemos dicho que es diferente de la del rectángulo. Pero es que, además de esta reducción de tiempo y esfuerzo, adelantamos (se verá en el capítulo 7) que el círculo que utiliza aprendizaje por refuerzo obtiene peores resultados que el que utiliza un sistema de reglas, por lo que creemos que la mejor decisión es no implementar las dos versiones del rectángulo.

Para el rectángulo, en ocasiones no es tan determinante llegar al punto objetivo con una velocidad concreta, como lo era para el círculo. Depende del tipo de movimiento:

* En los movimientos de los tipos DROP, NOMOVE, FALL, ADJACENT, HIGHTILT y BIGHOLEDROP, sí es importante llegar con una velocidad concreta. Para estos movimientos, el mismo sistema de reglas que utilizábamos para el círculo ofrece buenos resultados.

    - Para los movimientos de tipo DROP, la velocidad objetivo es siempre 0, ya que, para comenzar el movimiento, el rectángulo debe estar en forma horizontal, en reposo, alineado con el centro del hueco (ver tercera imagen de la secuencia de la Figura 5.7).

    - Para los movimientos de tipo NOMOVE, la velocidad objetivo también es siempre 0, ya que, cuando alcanzamos un diamante con un NOMOVE, lo queremos hacer con velocidad baja, porque no podemos saber de antemano si el siguiente objetivo estará a la derecha o la izquierda.

- Los movimientos de tipo FALL han sido simulados partiendo con una determinada velocidad horizontal inicial y, si no cumplimos esa precondición, es muy probable que el resultado de la ejecución diste mucho de lo simulado.

- Para los movimientos de tipo ADJACENT, que recordemos conectan plataformas reales y ficticias, hemos comprobado empíricamente que las velocidades altas pueden llegar a ser problemáticas por *bugs* en el juego. Además, como estos movimientos de tipo ADJACENT pertenecen a estructuras de tipo DROP, suelen preceder en los planes a movimientos de tipo DROP, que tienen velocidad objetivo 0. En consecuencia, es razonable que la velocidad objetivo de los movimientos de tipo ADJACENT sea también pequeña. Hemos considerado, tras la realización de varias pruebas con distintas velocidades, que 50, la décima parte de la velocidad máxima alcanzable, es un valor adecuado de velocidad objetivo de los movimientos de tipo ADJACENT.

- Para los movimientos HIGHTILT, recordamos que se necesitaba coger la máxima velocidad posible, que ya se había calculado en la etapa de planificación. Ya que estos son movimientos que hemos catalogado como arriesgados, queremos maximizar las posibilidades de realizarlos con éxito, para lo que debemos llegar con la velocidad objetivo que se les asocia.

- Los movimientos BIGHOLEDROP, debido a la implementación que hemos elegido, son parecidos a los movimientos ADJACENT. Dado que hemos optado por utilizar aprendizaje por refuerzo para resolver estos movimientos, y este aprendizaje siempre comenzaba con el rectángulo a una distancia de 5 cuadrados con velocidad muy baja y en forma de cuadrado, ya que esto reducía significativamente la cantidad de estados que se debían entrenar, queremos replicar esas condiciones. Por eso, nos interesa llegar al punto que se encuentra a una distancia de 5 cuadrados con, digamos, velocidad 20, con la que el rectángulo tiene tiempo de sobra para calibrar. Con una velocidad que fuera mucho mayor, el rectángulo no tendría espacio para frenar, y muy probablemente se quedaría estancado en el hueco.

∗ Por otro lado, en los movimientos de tipo TILT, BIGHOLEADJ y MONOSIDEDROP, no es crítico llegar con una velocidad determinada. Por tanto, no es necesario un sistema tan sofisticado para asegurarse de que se llega con una velocidad concreta al punto de inicio del movimiento. Simplemente habrá que realizar la acción que más acerque al rectángulo a la posición objetivo, es decir, si el punto objetivo está a la derecha de la posición actual del rectángulo, entonces se tomará la acción MOVE_RIGHT y, en caso contrario, la acción MOVE_LEFT. No va a ser relevante la velocidad que lleve el rectángulo cuando se encuentre en la posición objetivo, pero sí que es de interés que

la velocidad no sea excesivamente grande. Para controlar que la velocidad no se dispare, cuando el módulo de la velocidad es mayor que 250[3], tomamos la acción NOACTION para mantener la velocidad.

- Para los movimientos de tipo TILT, la velocidad con la que se debe llegar al escalón no es clara, siempre que sea suficientemente grande. Recordamos que los movimientos de tipo TILT pretendían voltear al rectángulo como se aprecia en la Figura 5.11 o en la animación del siguiente enlace.

- Para los movimientos BIGHOLEADJ, pretendemos que suceda lo que muestra la animación del siguiente enlace. Como se puede intuir de la animación, perturbaciones de la velocidad no afectarían al resultado final del movimiento.

- Finalmente, para los movimientos MONOSIDEDROP, la velocidad tampoco es determinante. Queremos que el rectángulo se comporte como en la animación del siguiente enlace. Al igual que para los movimientos BIGHOLEADJ, el movimiento se efectuará de la misma forma, aunque se llegue con velocidades diferentes.

Esto concluye la sección de cómo se elige la acción que más acerca al rectángulo a la posición del movimiento. En la siguiente sección, asumiremos que el rectángulo ya tiene la forma deseada y la posición y velocidad necesarias para iniciar el movimiento, y describiremos las acciones que toma mientras lo ejecuta.

5.4.4. Acción asociada al tipo de movimiento

En esta sección, explicamos las acciones que toma el rectángulo desde el comienzo del movimiento. Para ello, debe estar en la posición objetivo con la forma objetivo y la velocidad objetivo (si esta es relevante). Podemos clasificar los movimientos en 3 categorías: movimientos que no realizan ninguna acción, movimientos que realizan siempre la misma acción y movimientos que toman varias acciones diferentes.

Movimientos que no realizan ninguna acción

En esta categoría, se encuentran los movimientos NOMOVE y los movimientos ADJACENT, que terminan nada más comienzan. Esto se debe a que, en el caso de los NOMOVE, el rectángulo debería coger los diamantes asociados al movimiento si se encuentra en la posición correcta con la forma adecuada, mientras que, en los ADJACENT, el rectángulo cambia de plataforma en la siguiente actualización del

[3]Como referencia, la velocidad máxima que alcanzaba el círculo era del orden de 200, por lo que 250 no es, en ningún caso, una velocidad pequeña.

estado. En ninguno de los dos casos hay tiempo para realizar ninguna acción.

Movimientos que toman siempre la misma acción

En esta categoría, se encuentran varios de los movimientos del rectángulo, a los que les basta con tomar siempre la misma acción mientras se ejecutan para completarse. Los describimos a continuación:

- Movimientos FALL

Los movimientos FALL tienen asociada una acción que realizar durante el aire, tal y como se explicó en la sección 5.2.2. Ya anticipamos que, aunque los movimientos FALL deberían incluirse en la categoría de movimientos que toman varias acciones diferentes, consideramos que la simplificación de siempre tomar la misma acción es capaz de generar los movimientos adecuados en todos los niveles, lo que han corroborado todas las pruebas que hemos realizado. En consecuencia, durante un movimiento FALL siempre se toma una de las acciones NO_ACTION, MOVE_LEFT o MOVE_RIGHT.

- Movimientos TILT

Para los movimientos TILT, una vez se llega a la esquina sobre la que el rectángulo debe voltearse (segunda imagen de la figura 5.11), es necesario que el rectángulo continúe ejecutando la acción de desplazamiento lateral, de modo que siga haciendo fuerza con la esperanza de voltearse. Para ello, tenemos una variable que denominamos $has\_finished\_tilt$ a la que se le da el valor `false` cuando el rectángulo se encuentra a menos de 3 cuadrados de la posición objetivo, de tal manera que, mientras esa variable tenga el valor `false`, mantenga su acción actual. Tras varias pruebas, consideramos que liberar el valor de la variable una vez el rectángulo se aleja 4 cuadrados del punto de giro daba resultados satisfactorios. Se puede ver una animación de la ejecución de los movimientos de tipo TILT accediendo al siguiente enlace.

- Movimientos HIGHTILT

En los movimientos HIGHTILT, una vez se acerca a la esquina sobre la que se debe voltear, el rectángulo necesita mantener la acción que esté ejecutando (ya sea MOVE_LEFT, como en el ejemplo de la figura 5.12, o MOVE_RIGHT si la plataforma a la que se pretende subir está a la derecha). Esto ya sucedía en los movimientos TILT, pero en los movimientos HIGHTILT es más importante, lo que nos hizo dejar más margen antes de liberar la variable $has\_finished\_tilt$ del rectángulo, teniendo que encontrarse a una distancia de 10 cuadrados en lugar de 4, antes de considerar que ha terminado el movimiento. Esto se debe a que, como se puede apreciar en la animación del siguiente enlace, el rectángulo puede alejarse mucho al impactar

contra la esquina. Como ya dijimos en la sección 5.2.2, en muchas ocasiones, el rectángulo no logra subir en el primer intento, y suele quedarse en la posición de la figura 5.12, pero sin moverse, pues la variable $has\_finished\_tilt$ sigue estando a `false`. En este caso, hemos de esperar a que se llame al sistema de recuperación, que explicaremos más adelante, de forma que lo aleje de la esquina y se vuelva a empezar el proceso de coger carrerilla.

- Movimientos BIGHOLEADJ

Los movimientos BIGHOLEADJ son similares a los anteriores. Si no ejecutáramos ninguna acción, el rectángulo impactaría contra el otro lado del hueco, con lo que perdería mucho momento lineal y acabaría cayendo, o, aún peor, se quedaría estancado. Esto puede solucionarse si se mantiene la acción que ha tomado para acelerar, de forma que acabe volteándose sobre la esquina de la plataforma de llegada, como se puede observar en la animación del siguiente enlace. En el ejemplo de la figura 5.16, esto corresponde con la acción MOVE_RIGHT, mientras que si el hueco estuviera a la izquierda del rectángulo, sería la acción MOVE_LEFT. Una vez el centro del rectángulo se encuentra en la plataforma de llegada, el movimiento termina (pues el rectángulo se estabilizará por su propio peso y velocidad horizontal).

Movimientos que toman varias acciones diferentes

En esta categoría, se encuentran los movimientos que toman varias acciones diferentes mientras se ejecutan. Estos son los movimientos más complejos que es capaz de realizar el rectángulo, que son aquellos movimientos que tratan de caer por un hueco.

- Movimientos DROP

Los movimientos DROP requieren más sincronización y precisión que los movimientos mencionados hasta ahora. Una vez nos encontramos en la posición de la tercera imagen de la secuencia de la figura 5.7, el objetivo es ponerse en posición vertical (cuarta imagen de la figura) para caer por el hueco.

La primera dificultad que presenta esto es que, como explicaremos más adelante, antes de que el rectángulo elija la acción MORPH_UP para crecer, realizamos una comprobación para verificar que este dispone de espacio suficiente para hacerlo. En el caso de los movimientos DROP, muchas veces esta comprobación cancelaba la acción de crecer e impedía que el rectángulo se pusiese suficientemente vertical como para atravesar el hueco. En consecuencia, tuvimos que crear una variable $has\_finished\_drop$ que imposibilitara la interrupción del crecimiento del rectángulo. Esta variable se desactiva una vez el rectángulo ha alcanzado su altura máxima.

Sin embargo, como se puede observar en la figura 5.8, no basta con caer por el hueco

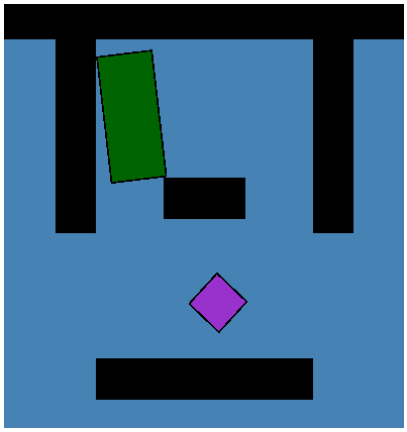


Figura 5.29: Configuración en la que el rectángulo no ha conseguido ejecutar el movimiento MONOSIDEDROP.

y esperar a aterrizar en la plataforma de llegada, ya que esta puede ser ficticia y la atravesaríamos si mantuviéramos la forma vertical. Por ese motivo, decidimos que, en cuanto el rectángulo atravesara el hueco, se intentara hacer horizontal lo antes posible (mediante la acción MORPH_DOWN). De esta manera, el rectángulo aterrizará siempre sobre la plataforma de llegada de manera horizontal. Esto permite que dicha plataforma de llegada pueda ser formalmente ficticia, aunque los puntos reales de apoyo serán las plataformas reales adyacentes, como en el caso de la figura 5.8. Para visualizar el comportamiento de volverse horizontal lo antes posible, junto a la secuencia conjunta de acciones que componen un movimiento de tipo DROP, recomendamos ver la animación del siguiente enlace.

- Movimientos MONOSIDEDROP

Los movimientos MONOSIDEDROP son una combinación de varios de los movimientos anteriores. Aunque el centro del rectángulo llegue al punto medio del hueco de la figura 5.9 (supongamos que se trata del hueco de la izquierda), puede suceder que el vértice inferior derecho todavía siga en la plataforma, especialmente en niveles como el de la figura, donde el hueco es estrecho y no hay mucho espacio para crecer. Por tanto, si no realizáramos ninguna acción, el rectángulo no caería y permanecería apoyado contra la pared, como se muestra en la figura 5.29.

Para solucionar este problema, nos aprovechamos de que al otro lado del hueco hay una pared, por lo que podemos mantener la acción MOVE_LEFT (si el hueco estuviera a la derecha de la plataforma de partida, se mantendría la acción MOVE_RIGHT) para asegurarnos de que los cuatro vértices del rectángulo abandonan la plataforma y el rectángulo cae con éxito por el hueco. Podemos ver un ejemplo del comportamiento anterior en la animación del siguiente enlace.

Podría parecer entonces que basta con mantener la misma acción y estos movimientos se realizarían siempre correctamente. Sin embargo, ¿qué sucede si la plataforma de destino del movimiento es ficticia? Con esta estrategia, es muy posible que el rec-

tángulo atravesara esta plataforma y cayera más de lo deseado. Al igual que sucedía con los movimientos DROP, es conveniente realizar la acción MORPH_DOWN para adoptar la forma horizontal lo antes posible, siempre que haya espacio suficiente para realizar dicha acción.

En resumen, si el centro del rectángulo está por encima del hueco, realizamos la acción que nos hará caer, ya sea MOVE_LEFT o MOVE_RIGHT. Si, por el contrario, el centro del rectángulo está por debajo del hueco, realizamos la acción MORPH_DOWN, como si se tratara de un DROP.

- Movimientos BIGHOLEDROP

Los movimientos BIGHOLEDROP son los movimientos más complicados del rectángulo. Tras la discusión que realizamos en la sección 5.2.2, debería ser evidente por qué pertenecen a esta categoría. El comportamiento de los movimientos BIGHOLEDROP varía de ejecución en ejecución como resultado del entrenamiento. Encontramos que, tras probar varias veces cada nivel, el rectángulo era capaz de seguir estrategias diferentes. Por ejemplo, en este enlace se puede ver una forma de realizar el movimiento, mientras que en este segundo enlace y en este tercer enlace se ven otras maneras diferentes. En consecuencia, la estrategia a seguir es la aprendida por el rectángulo en el entrenamiento y no depende de unas reglas que hayamos implementado nosotros, luego es más difícil de interpretar.

Esto concluye la explicación de las acciones que toma el rectángulo en función del tipo de movimiento. En la siguiente sección, explicamos algunas consideraciones adicionales que se tienen en cuenta durante la ejecución del agente.

5.4.5. Comentarios adicionales sobre la ejecución del plan

En esta última sección, destacamos otros aspectos que hemos considerado durante el desarrollo del agente, y que han permitido obtener mejores resultados en determinadas situaciones.

Sistema de recuperación

Comenzamos con uno de las características más importantes de nuestro agente. El comportamiento del rectángulo es bastante impredecible, y las acciones que puede tomar están muchas veces limitadas. Por ejemplo, salvo que el rectángulo tenga algún lado aproximadamente paralelo al suelo, las acciones MORPH_UP y MORPH_DOWN no tienen ningún efecto. Esto quiere decir que, si por ejemplo el rectángulo efectúa un movimiento de tipo FALL, gira durante el aire y acaba inclinado y apoyado sobre una pared, no va a poder cambiar de forma hasta que no se

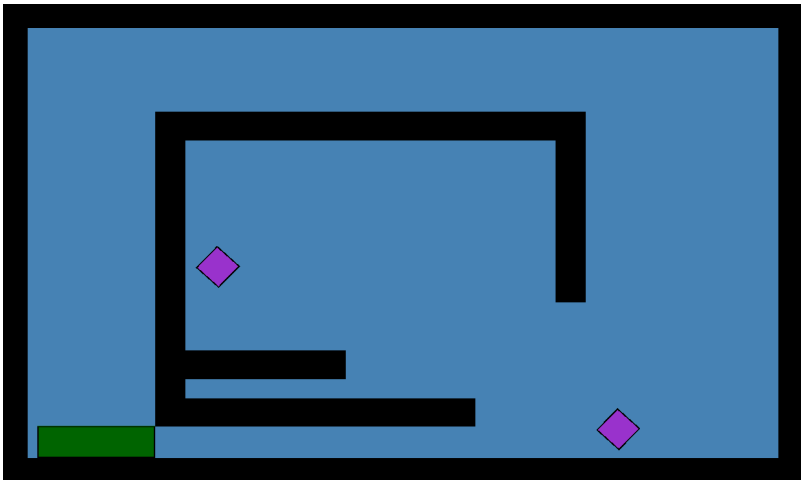


Figura 5.30: Configuración en la que el rectángulo piensa que está horizontal e intenta realizar la acción de ir a la derecha. Al no caber, su acción no tiene ningún efecto.

ponga otra vez "recto". Por ejemplo, en la situación de la figura 5.29, el rectángulo no sería capaz de realizar las acciones MORPH_UP y MORPH_DOWN, pero en la de la figura 5.9 sí.

Sin embargo, como ya mencionamos en la sección 5.2.2, el juego no proporciona en ningún momento la orientación o inclinación del rectángulo. En consecuencia, nuestro rectángulo no tiene manera de saber que se encuentra en una situación en la que no puede crecer o decrecer, y que debe moverse para solucionarlo.

Dado que nuestro algoritmo de ejecución intenta ponerse en la mejor forma posible antes de acercarse al punto del movimiento, esta característica del juego es muy problemática, y, sin el sistema de recuperación, nos dejaría estancados en la mayoría de niveles. Para solucionarlo, implementamos un sistema mediante el cual el agente comprueba en cada actualización del estado si alguno de los parámetros que tiene (posición horizontal, posición vertical, velocidad horizontal, velocidad vertical y altura) ha cambiado con respecto a la actualización anterior. Si el estado no varía durante 30 actualizaciones (equivalente a aproximadamente 3 segundos), es porque el rectángulo está intentando realizar una acción que no tiene ningún efecto. Esto puede ser por intentar variar su forma en la situación anterior o por intentar ir a la izquierda o a la derecha cuando se está dando contra una pared. Esto último puede pasar, por ejemplo, en situaciones como la de la figura 5.30.

En la situación de la figura, el rectángulo piensa que ya ha adoptado una forma completamente horizontal, debido a la discretización que realizamos. Sin embargo, todavía no es lo suficientemente bajo como para caber por el hueco, por lo que se queda estancado intentando moverse a la derecha.

Nuestro sistema de recuperación, al detectar estas situaciones, comienza a tomar acciones aleatorias, actualizándolas cada 200 ms. Debido a lo que hemos explicado en la discusión anterior, no sabemos cuál de estas acciones es la que debemos realizar

para continuar el nivel, por lo que no realizamos ninguna clase de filtro (salvo el de quitar la acción NO_ACTION, que lógicamente no va a ayudar al rectángulo en ninguna ocasión). Este sistema de recuperación termina cuando se detecta que el estado ha cambiado, aunque, si la acción aleatoria todavía no se había ejecutado durante 200 ms, se deja que termine. Esta decisión se debe a que, debido a las pruebas que hemos realizado, si interrumpíamos el sistema de recuperación directamente, muchas veces no le había dado tiempo a solucionar por completo la situación del rectángulo, por lo que quedaba estancado otra vez. Un tiempo de 200 ms es un valor que ha demostrado ser suficiente, y sin llegar a perjudicar al rectángulo en ninguna ocasión.

Sistema para evitar caídas

Durante las pruebas que realizamos, a pesar de que en el filtrado de movimientos priorizábamos los movimientos que dejaran espacio detrás del punto de aterrizaje para frenar (ver figura 4.6), ocasionalmente el agente podía acabar cayendo, si llegaba al borde de una plataforma con demasiada velocidad. Este problema ocurría con bastante más frecuencia en el círculo, y en el caso del rectángulo, aunque no era tan grave, igualmente decidimos probar a implementar un sistema parecido al que en el círculo nos solucionó el problema.

Para ello, implementamos una comprobación, que no hemos incluido en el algoritmo 4 por claridad, que evalúa si el rectángulo está en el primer cuadrado fuera de la plataforma. En ese caso, ordenamos al agente que se mueva hacia el centro de la plataforma. Dado que el rectángulo se mueve con velocidades mayores a las del círculo, este sistema nunca le perjudica, ya que solo entra en acción si sale de una plataforma, y solo tendrá algún efecto si el tiempo que permanece en el primer cuadrado es suficiente para contrarrestar la velocidad que llevara. Este sistema no es tan importante en el comportamiento del agente como lo era en el círculo, pero es una pequeña mejora que creemos relevante mencionar.

Sistema de recuperación de la orientación

Cuando se realiza un movimiento de tipo TILT o una caída en la que se gira en el aire, el rectángulo puede acabar volteado, de forma que lo que antes era su base se convierte en uno de sus laterales. Lógicamente, esto altera las dimensiones del rectángulo, intercambiando el valor de la altura y la anchura del mismo. Sin embargo, el juego no hace este intercambio por defecto. Es decir, si el rectángulo se voltea con una altura h, el juego va a seguir devolviendo como altura el valor h hasta que se ejecute una acción de MORPH_UP o MORPH_DOWN, momento en el que se recalcula la altura y se comienza a devolver de forma correcta el valor.

Aunque no sabemos a qué se debe este fenómeno (creemos que se trata de un *bug*), lo

cierto es que es fundamental que lo tratemos de solucionar. Existen muchas ocasiones en las que conocer la forma del rectángulo es fundamental, ya sea para calcular la plataforma en la que está o para determinar si tiene la forma adecuada para realizar un movimiento determinado. En consecuencia, tuvimos que desarrollar lo que denominamos como "Sistema de recuperación de la orientación". La idea subyacente es simple, si detectamos que la altura que devuelve el juego no es consistente con los datos que obtenemos, ejecutamos MORPH_UP y MORPH_DOWN alternativamente hasta que se solucione.

La forma de realizar esta comprobación es identificando la plataforma que está inmediatamente debajo del centro del rectángulo, ya que la información sobre el centro siempre es correcta. Si la distancia entre dicha plataforma y el centro del rectángulo es la mitad de la altura del rectángulo que nos proporciona el juego (obviando un pequeño margen de error razonable), entonces la orientación es la correcta. Si, por el contrario, la distancia entre la plataforma inmediatamente inferior al centro del rectángulo y este último es la mitad de la anchura proporcionada por el juego, debe ejecutarse el sistema de recuperación de la orientación.

Este sistema podría parecer menor y con poco impacto, pero aumenta enormemente el rendimiento en niveles que involucran cualquier movimiento que no sea de tipo DROP o NOMOVE, es decir, la inmensa mayoría de los niveles. Sin este sistema, tendríamos que esperar a que el sistema de recuperación fuera el que restaurara la orientación, pudiendo tener consecuencias indeseadas por el camino.

Sistema de comprobación de cambio de forma

Por último, queremos mencionar cómo solucionamos dos de los *bugs* que afectan al rectángulo. Ambos se deben a que el juego no comprueba que el rectángulo tiene el espacio suficiente para poder cambiar su forma cuando intenta hacerlo.

En primer lugar, consideremos el estado de la figura 5.31. Esta situación se da cuando en dos actualizaciones sucesivas del rectángulo, en una el lado superior se encuentra por debajo del techo, y en otra se encuentra por encima. El juego no comprueba entre actualizaciones que el rectángulo va a superar la altura máxima que cabría en el espacio actual.

Cuando el rectángulo entra en esta situación, empieza a actuar de manera incontrolable, como se puede apreciar en la animación del siguiente enlace. Usualmente, esto lleva a comportamientos no deseados que pueden provocar perder el nivel.

Para solucionarlo, realizamos dos comprobaciones. La primera se lleva a cabo antes de realizar la acción MORPH_UP, donde se comprueba si el rectángulo todavía tiene al menos un cuadrado de margen con el techo. Tenemos que ajustarnos lo más posible al techo, porque hay muchos movimientos que necesitan que la forma sea lo

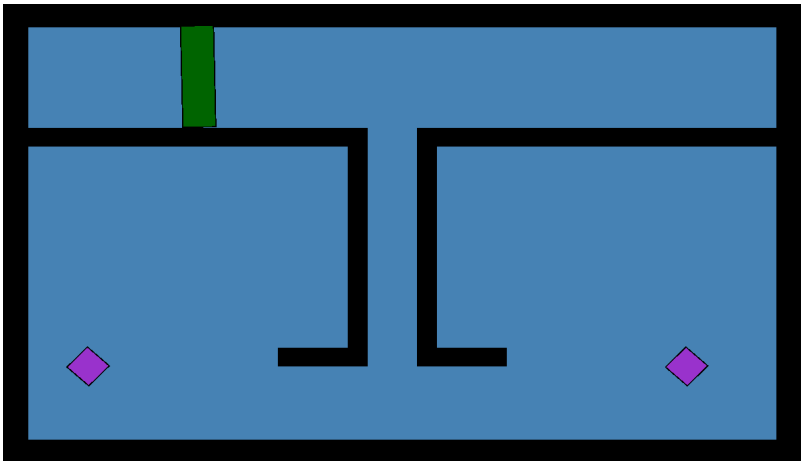


Figura 5.31: Configuración en la que el rectángulo ha excedido la altura máxima que permitirían los obstáculos de alrededor.

más vertical posible (véase, por ejemplo, la figura 5.9).

Sin embargo, el dejar tan poco margen, la frecuencia de actualización del estado y lo rápido que se efectúan los cambios de forma, provoca que en ocasiones el rectángulo acabe superando esta altura máxima, comenzando a actuar de forma incontrolable. En estas situaciones, realizamos otra comprobación, que efectúa inmediatamente la acción MORPH_DOWN en caso de que el rectángulo exceda la altura permitida. Normalmente, esta estrategia da buenos resultados, y la frecuencia con la que se toman acciones permite al rectángulo salir de este estado caótico lo suficientemente rápido y sin que sea gravemente perjudicado.

El otro *bug* al que nos referíamos es análogo al anterior, pero con la acción de MORPH_DOWN, es decir, cuando el rectángulo está intentando hacerse más ancho. Este *bug* es mucho más severo que el anterior, ya que suele dejar al rectángulo en situaciones irrecuperables, como la de la figura 5.32, o la de la animación del siguiente enlace.

Esto sucede, análogamente a la situación anterior, cuando el rectángulo trata de hacerse más ancho de lo que permite el espacio que tiene. Este problema es mucho más grave que el otro, por lo que no nos sirve de nada tener un sistema que trate de sacarlo de él. La solución que hemos implementado consiste en realizar la misma comprobación que hacíamos antes de tomar la acción MORPH_UP, pero con la acción MORPH_DOWN. En esta ocasión, sin embargo, somos más exigentes con el espacio que debe tener el rectángulo para poder hacerse más ancho, debiendo contar al menos con un cuadrado de espacio a la izquierda y al menos un cuadrado de espacio a la derecha. Esta comprobación ha resultado efectiva en todas las pruebas que hemos realizado, y de suma importancia en la implementación de los movimientos de tipo DROP y MONOSIDEDROP.

Antes de terminar y aunque no tenga mucha relación con el sistema de comprobación

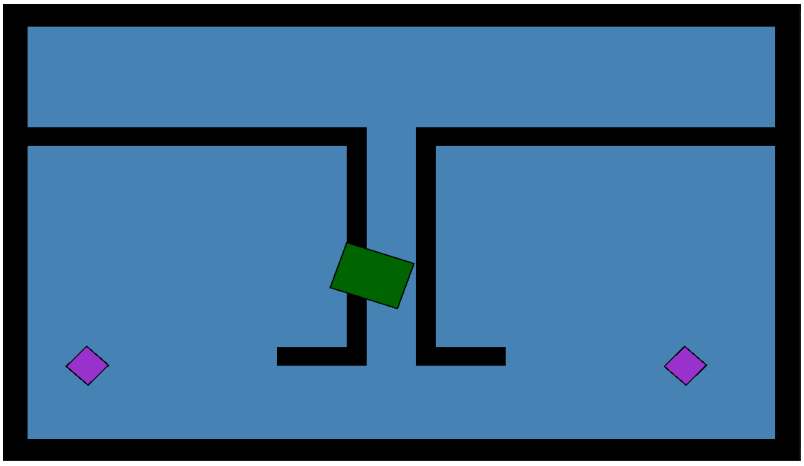


Figura 5.32: Configuración en la que el rectángulo se ha quedado en un estado irrecuperable tras intentar hacerse demasiado ancho en un espacio en el que no se podía.

de cambios de formas, creemos que es el momento apropiado de presentar un tercer *bug* del juego. En algunas ocasiones, cuando el rectángulo cae desde una plataforma elevada y aterriza bruscamente sobre otra plataforma, golpeándola con uno de sus vértices, se bloquea la capacidad del rectángulo para cambiar de forma. Hemos comprobado que este *bug* se produce tanto para los jugadores humanos como para los agentes implementados y, como se explicará en la sección 7.3 del capítulo de resultados, puede hacer que algunos niveles que nuestro rectángulo es capaz de resolver, no puedan ser completados porque el juego no aplica la acción de crecer o decrecer que está eligiendo nuestro agente.

Con esto, concluimos la explicación de la implementación que hemos realizado para el rectángulo. En el siguiente capítulo, detallaremos las modificaciones y extensiones que hemos efectuado sobre el círculo y el rectángulo para adaptarlos a los niveles cooperativos.

Capítulo 6

Implementación de técnicas cooperativas

En los tres capítulos anteriores, hemos explicado el funcionamiento de los agentes del círculo y el rectángulo cuando participaban en los niveles de manera individual. En el presente capítulo, abordaremos los niveles cooperativos, es decir, aquellos en los que ambos agentes deben coordinarse para alcanzar todos los diamantes posibles. En la sección 6.1, comenzaremos con una introducción a la cooperación y los retos adicionales que presenta sobre un entorno ya de por sí complejo. De manera similar a como hemos hecho en los capítulos anteriores, nuestra estrategia para superar los niveles está estructurada en una representación adecuada del entorno, que se detalla en la sección 6.2; la elaboración de un plan a seguir, explicada en la sección 6.3; y, finalmente, la ejecución de dicho plan, que se encuentra en la sección 6.4. Incluimos también la sección 6.5, en la que nos detendremos a hablar de la explicabilidad de nuestro modelo.

6.1. Introducción y retos derivados de la cooperación

La idea original del juego Geometry Friends y de la competición asociada a él era el desarrollo de agentes cooperativos. Estos agentes formarían parte de un mundo de plataformas gobernado por las leyes de la física, en un entorno dinámico con fricción, gravedad y colisiones realistas, y donde compartirían un objetivo común: alcanzar el máximo número de diamantes en el menor tiempo posible. Así, los agentes fueron diseñados para que poseyeran características complementarias. Recordemos que, aunque ambos agentes pueden desplazarse a izquierda y derecha, únicamente el círculo puede saltar y solamente el rectángulo puede modificar su forma. Esta diversidad de acciones, que ya hemos explotado por separado, permite alcanzar un nivel superior de juego al combinarlas. Pero si la concepción de Geometry Friends

buscaba, principalmente, establecer un entorno para el desarrollo de agentes cooperativos, ¿por qué han tenido tanto interés las competiciones individuales del círculo y el rectángulo?

Lo cierto es que, para abordar la cooperación, en primer lugar, los agentes individuales deben ser capaces de resolver muchos retos por separado, lo que no es en absoluto sencillo en el entorno dinámico en el que transcurre la acción. Por ejemplo, las parábolas de salto y caída del círculo son muy sensibles a la velocidad y la posición inicial, la forma del rectángulo es determinante para moverse por el nivel, el orden en el que se planea alcanzar los diamantes debe ser el correcto, los agentes deben identificar los obstáculos que pueden utilizar como plataformas y los que no, también deben estar preparados ante posibles eventualidades y circunstancias imprevistas como quedar atascados o errar un movimiento que les conduzca a una plataforma indeseada, y tienen que ser capaces de analizar si pueden moverse de una plataforma a otra y cómo pueden conseguirlo. Además, deben hacer todo lo anterior de la manera más rápida posible. Todos estos retos, entre muchos otros, también están presentes en los niveles cooperativos, pero se pueden abordar de manera más sencilla restringiéndonos únicamente a los niveles individuales. Y es por ello por lo que hemos dedicado tanto tiempo a desarrollar y explicar los agentes individuales, pues, más allá de que también sea necesario resolver estos retos para abordar la cooperación, su mera resolución tiene un gran valor e interés por sí mismo.

Sin embargo, por muy complejos que puedan haber resultado, no son comparables a la dificultad añadida que ofrece la cooperación. La cooperación presenta nuevos desafíos en cada una de las tres fases en las que dividimos la resolución de un nivel: representación del nivel, planificación y ejecución.

En la fase de representación del nivel debemos ser capaces de reflejar todas las nuevas posibilidades que ofrece la cooperación. Debemos lograr capturar las nuevas formas en las que el círculo y el rectángulo pueden moverse entre las plataformas y alcanzar diamantes, e incluso las nuevas plataformas "cooperativas" que se pueden generar. Como muestra de las nuevas situaciones que surgen de la cooperación, ponemos como ejemplos que el rectángulo sirva de plataforma para el círculo (considerando cada una de sus formas) (ver Figura 6.1), que el círculo sirva de punto de apoyo para el rectángulo para que pueda voltearse y alcanzar plataformas más elevadas (ver Figura 6.2), que el círculo impulse verticalmente al rectángulo con su salto (ver Figura 6.3), o que el rectángulo transporte al círculo de una plataforma a otra mientras el círculo va montado sobre él (ver Figura 6.4).

Los dos últimos ejemplos anteriores hacen entrever que la variedad de situaciones que se pueden llegar a generar modificando los colores de los obstáculos es endiablada. Como los obstáculos pueden ser negros, amarillos o verdes, y los agentes atraviesan aquellos que son de su mismo color, pueden darse situaciones en las que únicamente uno de los personajes pueda alcanzar un diamante, como se aprecia en las Figuras 6.5 y 6.6. Las posibles configuraciones de los niveles, como vemos, son infinitas y nuestro ambicioso objetivo es desarrollar unos agentes capaces de enfrentarse a cualquiera

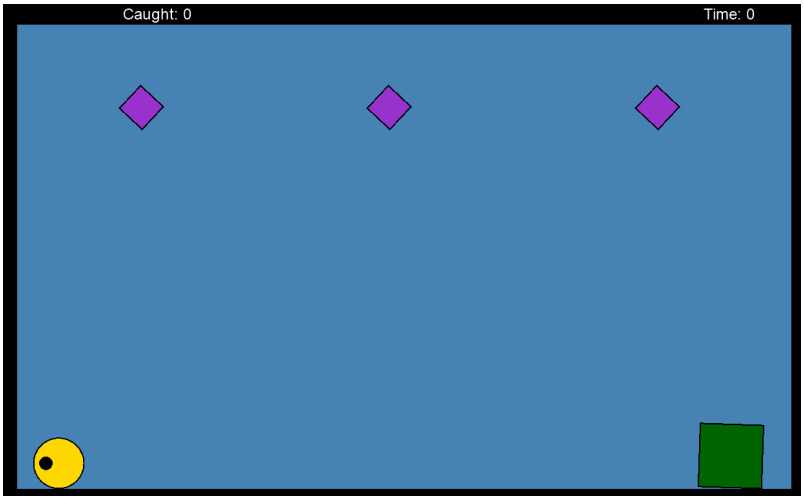


Figura 6.1: Nivel 1 de la competición cooperativa del año 2013, en el que los diamantes solamente son alcanzables si el círculo salta sobre el rectángulo, el rectángulo crece hasta estar completamente vertical y el círculo vuelve a saltar.

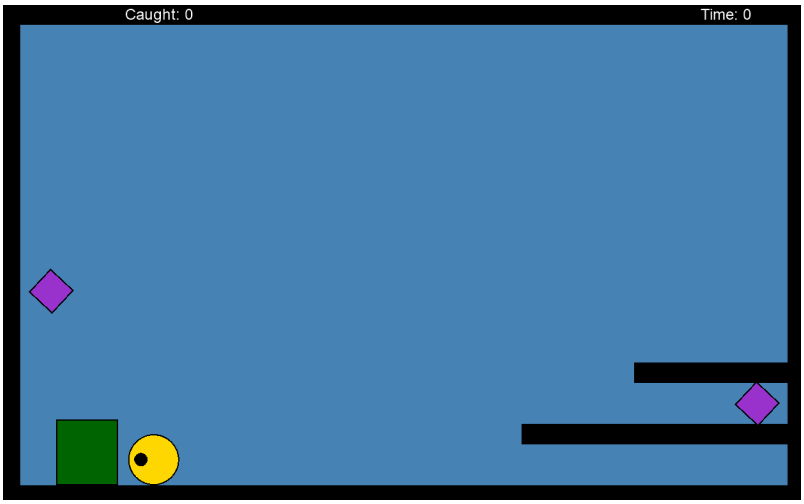


Figura 6.2: Nivel 3 de la competición cooperativa del año 2017, en el que el rectángulo es el único que puede alcanzar el diamante de la derecha porque el hueco es demasiado estrecho para el círculo. Sin embargo, el rectángulo no puede subir por sí mismo a la plataforma porque está demasiado alta, y necesita ayuda del círculo para lograrlo.

de ellas, o, al menos, a la mayoría.

En cuanto a la elaboración del plan, lo primero que debemos reseñar es que hay que construir un plan doble, ya que cada agente necesitará un plan individualizado. En segundo lugar, se debe tener en cuenta cuándo se pueden realizar los movimientos para cambiar de plataforma o alcanzar un diamante. Por ejemplo, en el nivel de la Figura 6.2, el rectángulo no podrá tener en su plan un movimiento en el que se apoye en el círculo y se voltee para subir a la plataforma en la que se encuentra el diamante de la derecha si el círculo no está en la plataforma inferior. De igual

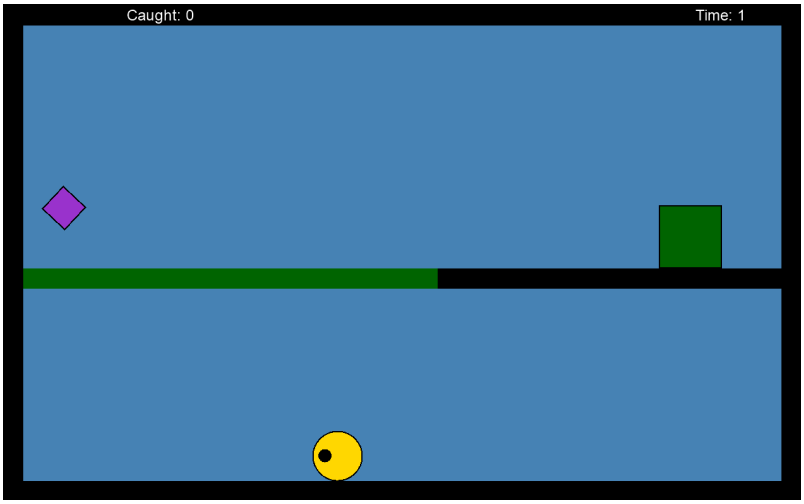


Figura 6.3: Nivel 8 de la competición cooperativa del año 2020, en el que el rectángulo es el único que puede alcanzar el diamante, ya que el círculo rebota contra los obstáculos verdes y negros. Sin embargo, el rectángulo no puede llegar por sí mismo a la posición del diamante y, para lograrlo, el círculo debe saltar en el momento adecuado para impulsar al rectángulo mientras este último cae.

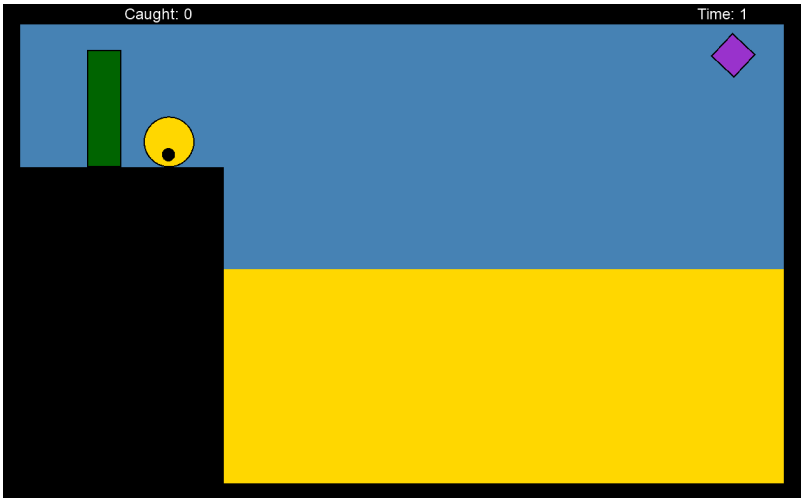


Figura 6.4: Nivel 9 de la competición cooperativa del año 2013, en el que el rectángulo debe transportar al círculo encima suya hasta el diamante de la derecha, que solamente puede ser alcanzado por el círculo. Nótese que el círculo no puede rodar por la plataforma amarilla porque la atraviesa.

forma, en el nivel de la Figura 6.4, el círculo no podrá tener planificada una caída para situarse sobre el rectángulo hasta que el rectángulo este sobre la plataforma amarilla.

Todo lo anterior impone ciertas restricciones espaciales que se deben cumplir a la hora de expandir los nodos en la búsqueda. Asimismo, el plan debe ser idealmente óptimo en tiempo. Para ello se debe exprimir al máximo la división de tareas, en la que cada agente puede coger un diamante distinto, pero priorizando siempre que los

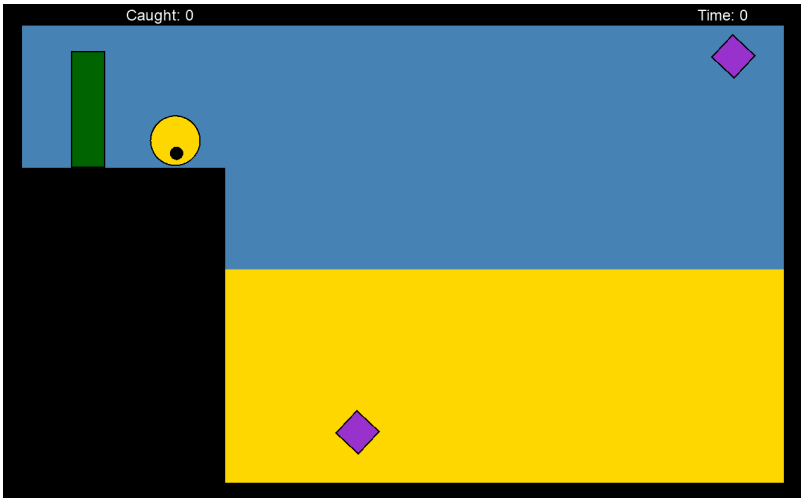


Figura 6.5: Nivel 6 de la competición cooperativa del año 2016, en el que el diamante dentro del obstáculo amarillo solamente puede ser alcanzado por el círculo.

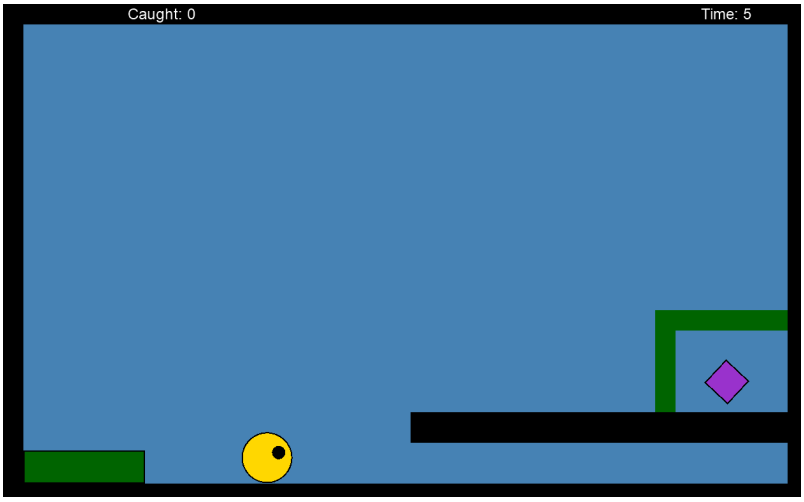


Figura 6.6: Nivel 4 de la competición cooperativa del año 2020, en el que el diamante solamente puede ser alcanzado por el rectángulo. Para ello necesita la colaboración del círculo porque no es capaz de subir de forma independiente a la plataforma, que está demasiado alta.

planes alcancen todos los diamantes, luego el orden en el que se cogen y qué agente los alcanza es determinante para completar los niveles (ver Figura 6.7). En este aspecto, debemos tener precauciones a la hora de dar soluciones subóptimas, ya que la cooperación es un cuello de botella de los planes, porque puede tener bloqueado a un agente en una plataforma hasta que el otro finaliza parte de su plan y realicen una acción de cooperación.

Por último, la ejecución del plan no es precisamente la parte más sencilla de todo el proceso. Los agentes deben mostrar una gran coordinación para tomar acciones combinadas. Que los agentes no se dejen espacio para realizar las acciones, que el rectángulo no pueda mantener el equilibrio cuando está en forma vertical y el círculo

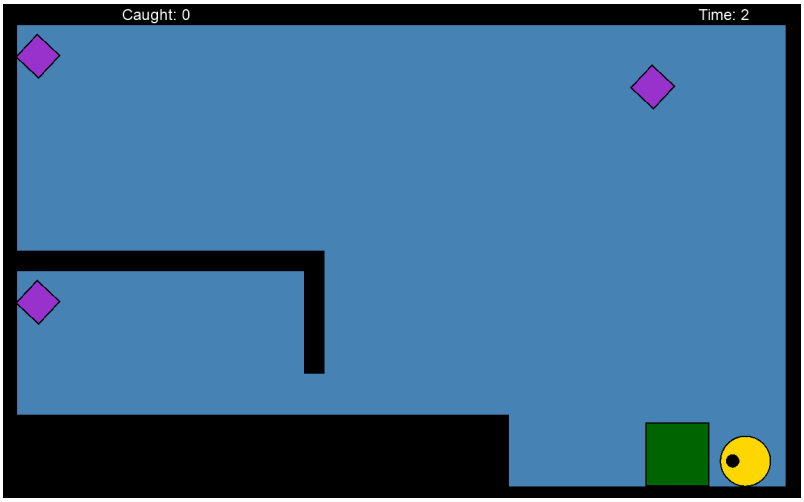


Figura 6.7: Nivel 1 de la competición cooperativa del año 2016, en el que el plan óptimo de alto nivel es: en primer lugar, que el círculo se suba encima del rectángulo para alcanzar el diamante que se encuentra en la zona superior derecha, y que después se dividan las tareas, de forma que el círculo coja el diamante de la esquina superior izquierda y el rectángulo el de la izquierda. Para esto último el rectángulo necesita la ayuda del círculo para subir a la plataforma. Nótese que ninguno de los dos agentes puede coger el otro diamante.

se sitúa sobre él o que los agentes quieran ir en direcciones contrarias y choquen, son una pequeña muestra de los problemas a los que se tienen que enfrentar el círculo y el rectángulo.

Confiamos en que todo lo anterior haya prevenido al lector de que el problema al que nos enfrentamos es muy complejo. La cooperación multiplica la dificultad, ya de por sí alta, del entorno físico de Geometry Friends y lo hace un banco de pruebas muy interesante para la implementación de técnicas de IA. En las siguientes secciones daremos una explicación detallada de nuestra propuesta para resolver los retos anteriormente expuestos. Como veremos más adelante, somos capaces de completar casi la totalidad de los niveles cooperativos de los años anteriores, hito al que no se habían acercado ninguna de las implementaciones propuestas anteriormente.

6.2. Representación del nivel

Para resolver los niveles cooperativos, partiremos de lo realizado para los agentes individuales, empezando con la identificación de plataformas y tratando después la generación de movimientos y su filtrado. Teniendo eso en mente, necesitamos pensar qué representación adicional vamos a necesitar para resolver las situaciones colaborativas que hemos descrito en la sección anterior.

6.2.1. Plataformas cooperativas

La colaboración más básica, y la primera en la que nos vamos a centrar, es aquella en la que el círculo utiliza de plataforma al rectángulo, con el fin de alcanzar diamantes o plataformas que se encuentren muy altos. Para ello, recordamos que tanto el círculo como el rectángulo habían identificado sus correspondientes plataformas (que en general son diferentes), y las usaban para detectar qué movimientos permitían moverse de una a otra. Esta información puede ser compartida de forma bidireccional entre ambos agentes mediante un sistema de mensajes que ofrece la interfaz del juego. En definitiva, si queremos utilizar al rectángulo de plataforma y detectar qué movimientos son posibles para el círculo cuando se encuentra sobre el rectángulo, la solución más evidente es considerar plataformas ficticias en los lugares donde se podría situar el rectángulo y ver qué elementos se pueden alcanzar desde ahí.

De esta forma, al igual que en el desarrollo del rectángulo considerábamos plataformas ficticias en lugares donde el agente podía apoyarse bajo ciertas condiciones (en aquel caso, era que la forma fuera horizontal y hubiese un hueco no demasiado grande entre dos plataformas), decidimos generar plataformas ficticias para el círculo y reutilizar la estrategia que ya seguimos anteriormente.

Sin embargo, durante la generación de estas plataformas, nos encontramos con varios elementos a tener en cuenta. En primer lugar, las plataformas del rectángulo debían estar ya generadas, pues es a partir de ellas de donde se crearán las ficticias del círculo. Es decir, solamente tiene sentido considerar aquellos lugares en los que sabemos que el rectángulo puede apoyarse. Además, el círculo se situará a diferentes alturas según la forma que adopte el rectángulo. Por tanto, decidimos crear tantas plataformas como formas básicas (totalmente horizontal, totalmente vertical y cuadrado) admisibles tuviera el rectángulo en esa plataforma. Finalmente, antes de generar la plataforma ficticia, comprobamos que el círculo realmente puede apoyarse en esa posición (supuesto que el rectángulo está debajo) y no choca con ningún obstáculo. Todo esto puede vislumbrarse en la figura 6.8, un nivel dividido en cuatro regiones, donde las plataformas ficticias del círculo en cada una de ellas aparecen de color rosa.

Como puede observarse en el nivel de la imagen, el rectángulo tiene una única plataforma simplificada (que solo mira adyacencia independientemente de las formas admisibles del rectángulo), pero tiene múltiples plataformas no simplificadas (que tienen en cuenta las formas admisibles del rectángulo). En la zona de la plataforma marcada de color violeta, el rectángulo solo cabe en forma horizontal (ya que colisiona con el obstáculo amarillo), por lo que solamente se genera esa plataforma ficticia. Por otro lado, en la parte recuadrada de color rojo, el rectángulo puede estar en cualquiera de las tres formas, pero el círculo colisiona con el obstáculo superior si se posicionara encima del rectángulo estando este en posición vertical, por lo que esa plataforma ficticia no se genera. Lo mismo sucede en la parte de color marrón, en

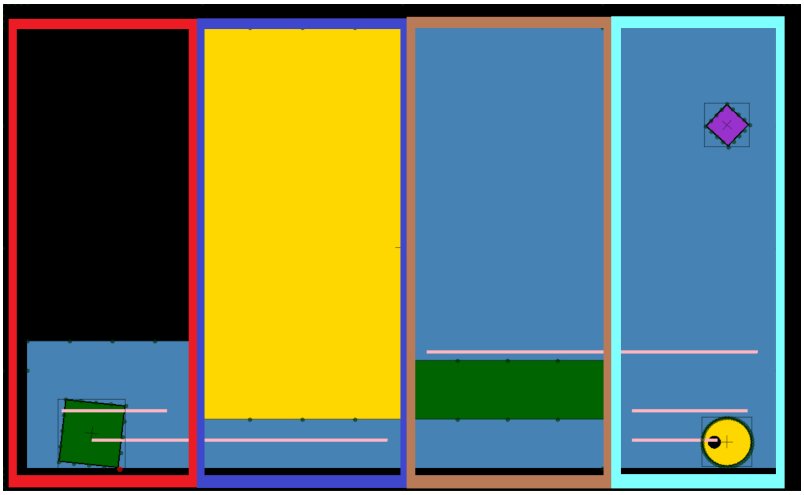


Figura 6.8: Nivel en el que la única plataforma simplificada del rectángulo se ha dividido en cuatro regiones en las que se han generado diferentes plataformas ficticias del círculo.

la que, de estar sobre el rectángulo, el círculo colisionaría con algún obstáculo que no puede atravesar, salvo que este estuviera en forma vertical (pues el rectángulo sí puede atravesar el obstáculo verde). Por último, en la zona de color cian, se generan las tres plataformas ficticias sin problema.

Nótese que las plataformas del último caso tienen diferente anchura en función de la forma del rectángulo a partir de la cual se han generado. Esto se debe a que las plataformas del rectángulo indican dónde puede posicionarse el centro del mismo. De esta forma, aunque el círculo podría estar más a la derecha de lo que indica la plataforma ficticia asociada a la forma horizontal del rectángulo (si se apoyara sobre la parte derecha del rectángulo), la generación que realizamos no es tan extensa. Hubo un momento en el que consideramos extender manualmente las plataformas ficticias generadas para que representaran realmente dónde podía colocarse el círculo sin necesidad de estar sobre el centro del rectángulo, pero, de cara a la ejecución, suponía muchas más complicaciones innecesarias. La aproximación que elegimos, que es la que hemos detallado, aunque no es perfecta, proporciona muy buenos resultados, como ya veremos en la sección 7.4.

Si no realizáramos ninguna modificación adicional, las plataformas ficticias que generaríamos tendrían un gran problema, y es que el círculo no sabría cómo pasar *adecuadamente* de una altura a otra. Con esto, queremos referirnos a que, si el círculo se encuentra encima del centro del rectángulo, que está, por ejemplo, con forma de cuadrado, es obvio que puede acceder a las plataformas ficticias que tiene encima y debajo haciendo que el rectángulo ejecute la acción MORPH_UP o MORPH_DOWN, respectivamente. Es decir, puede cambiar entre determinadas plataformas ficticias sin tener que realizar ningún movimiento.

Este fenómeno es similar al que teníamos en el rectángulo con las plataformas no sim-

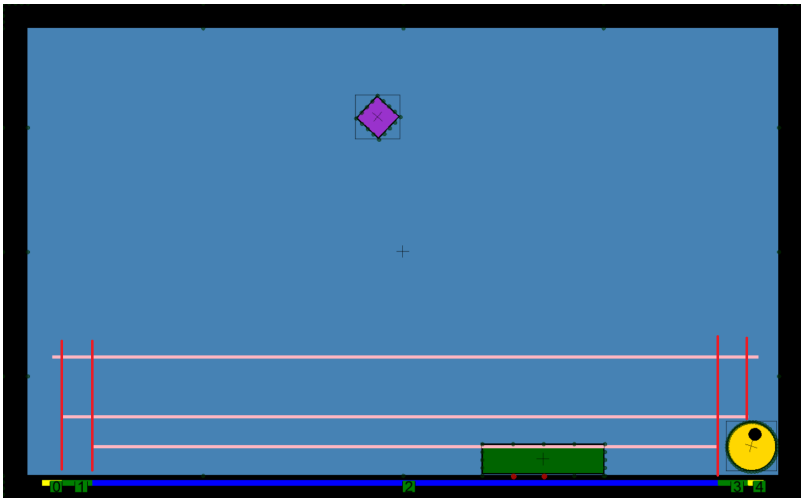


Figura 6.9: Nivel en el que cada plataforma no simplificada del rectángulo, separadas por líneas rojas verticales, genera sus propias plataformas ficticias del círculo, no uniéndose a pesar de que son adyacentes.

plificadas: podíamos pasar de una a otra simplemente adoptando la forma adecuada, independientemente de la velocidad que tuviéramos y de manera bidireccional. En consecuencia, decidimos que el círculo también iba a tener asociadas plataformas simplificadas. Así, si la plataforma es real, es decir, el círculo se encuentra encima de un obstáculo, la plataforma simplificada será la misma que la no simplificada; pero, si la plataforma es ficticia, la simplificada agruparía todas las ficticias a las que se pudiera llegar sin más que cambiar la forma del rectángulo.

La agrupación de estas plataformas ficticias en plataformas simplificadas no es tan inmediata como pueda parecer. La primera idea que tuvimos fue agrupar aquellas plataformas ficticias que hubieran sido generadas a partir de la misma plataforma no simplificada del rectángulo, pues son las que usamos para esta tarea. Sin embargo, esta primera aproximación dejaba mucho que mejorar, siendo lo más problemático el hecho de que no agrupara plataformas adyacentes, es decir, plataformas ficticias cuyas plataformas no simplificadas del rectángulo, a partir de las cuales se habían generado, estuvieran asociadas a la misma plataforma simplificada. Por ejemplo, en la figura 6.9, las plataformas generadas a partir de las no simplificadas 0 y 1 del rectángulo deberían pertenecer a la misma plataforma simplificada, ya que el rectángulo tan solo debe cambiar a forma vertical para poder moverse libremente entre ellas.

Tras este primer e infructuoso intento, probamos a realizar lo mismo, pero agrupando plataformas que se hubieran generado a partir de la misma plataforma simplificada del rectángulo. Si bien esto solucionaba el problema anterior, tenía otras consecuencias aún peores. Si consideramos el nivel de la figura 6.10, podemos ver que las plataformas ficticias de la izquierda y de la derecha se agrupan, cuando claramente no hay manera de ir de una a otra (de hecho, el círculo ni siquiera es capaz de llegar a la parte derecha del nivel). Es decir, si bien antes agrupábamos menos plataformas

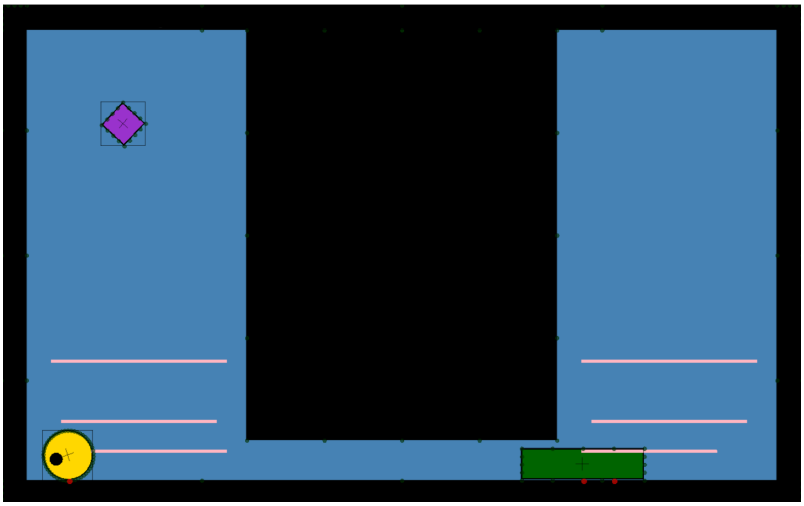


Figura 6.10: Nivel en el que se generan dos grupos de plataformas ficticias del círculo, pero generadas a partir de plataformas no simplificadas del rectángulo que pertenecen a la misma simplificada, por lo que se unirían incorrectamente.

ficticias de las que queríamos, con esta segunda idea agrupábamos de más.

La tercera opción que barajamos fue exigir que las plataformas que se agruparan tuvieran que estar conectadas horizontalmente. Esto quiere decir que solamente agruparíamos plataformas cuya proyección en el eje X tuviera intersección no vacía (o estuvieran conectadas a través de una cadena de plataformas ficticias). Esto no se hizo como alternativa al método anterior, sino que se incorporó como una restricción adicional. Si bien esta aproximación mejoraba en gran medida las plataformas resultantes, todavía agrupaba más plataformas de las apropiadas. Tras probar en muchos niveles, encontramos el nivel de la figura 6.11, en el que las dos plataformas ficticias de la izquierda se agrupaban incorrectamente, ya que el círculo no es capaz de moverse entre ellas incluso si el rectángulo cambia de forma, por no poder atravesar el obstáculo verde. Este problema es muy específico, pues se unen incorrectamente plataformas ficticias que se han generado a partir de la misma plataforma no simplificada del rectángulo, y ninguna de las tres aproximaciones que habíamos seguido lo había tenido en cuenta.

Por último, dimos con una solución que resultó adecuada en todos los niveles de competiciones de años anteriores que probamos. Partiendo de la idea anterior, añadimos otra nueva comprobación que debía cumplirse para poder agrupar las plataformas. Dada una agrupación A (es decir, una futura plataforma simplificada), para añadir una nueva plataforma p a la agrupación, debía cumplirse que, entre la altura más baja de A (que correspondería a la altura de la plataforma más baja que contenga A) y la altura de p, no podía haber ninguna plataforma p_r del círculo que fuera real y cuyos límites horizontales contuvieran los de p. Esto, escrito en términos

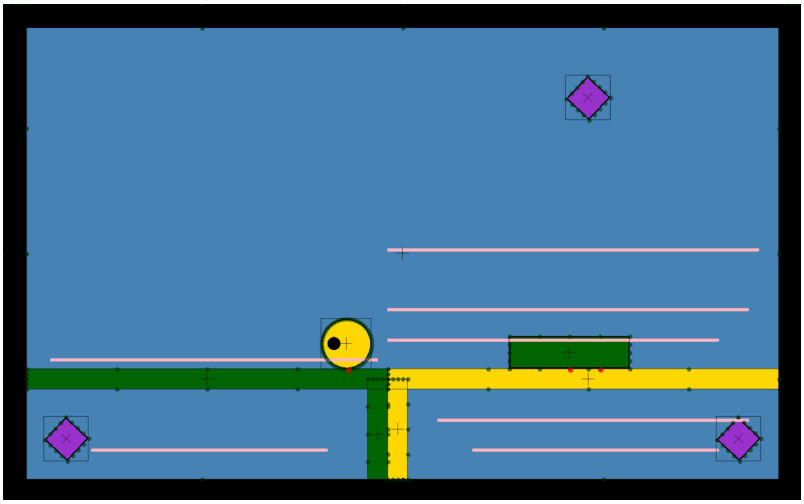


Figura 6.11: Nivel 8 de la competición de 2017, en el que las plataformas ficticias de la parte izquierda no deberían agruparse a pesar de haberse creado a partir de la misma plataforma, ya que el círculo no puede pasar de una a otra.

matemáticos, quiere decir que p se puede añadir a A si, y solo si,

$$\nexists\, p_r \text{ tal que } \begin{cases} p_r.\text{is\_real}, \\ p_r.\text{righ\_edge} \geq p.\text{right\_edge}, \\ p_r.\text{left\_edge} \leq p.\text{left\_edge}, \\ \text{Sign}(p_r.\text{height} - p.\text{height}) = \text{Sign}(A.\text{min\_height} - p_r.\text{height}), \end{cases}$$

donde $\text{Sign}(x)$ es la función signo (0 si x es 0 y $|x|/x$ en caso contrario), por lo que, pensándolo con cuidado, la última condición expresa que la altura de la plataforma p_r está entre la de p y la más baja del conjunto A.

Aunque aún se pueden crear niveles diseñados específicamente para que este sistema de agrupamiento falle, consideramos que eran situaciones demasiado concretas, que nunca se iban a dar en las competiciones reales, donde sí que agrupamos siempre de manera correcta.

Como último comentario, hemos tratado de manera especial las plataformas ficticias del rectángulo. A partir de ellas, únicamente hemos creado la plataforma ficticia del círculo asociada a la forma horizontal, ya que la forma de cuadrado es demasiado inestable en cuanto el hueco es mínimamente ancho. Sabemos que esto puede limitar la ejecución en niveles específicos, pero, al no encontrar ninguno en competiciones de años anteriores, y teniendo en cuenta las limitaciones temporales para realizar la implementación de los agentes, decidimos omitirlo.

Además, estas plataformas ficticias del círculo, generadas a partir de plataformas ficticias del rectángulo, no se agrupan con el resto, ya que no han sido generadas a partir de la misma plataforma simplificada del rectángulo. Aunque esto parece contradecir la idea original de las plataformas simplificadas del círculo, recordamos que en el rectángulo se hacía lo mismo, y teníamos movimientos específicos para

cambiar entre una plataforma ficticia y una real. Como ya veremos, hemos optado por continuar con esta filosofía y generar movimientos para el círculo para estas situaciones.

Con esto, concluimos esta sección, de forma que en la siguiente hablaremos de los movimientos adicionales que hemos creado en nuestro desarrollo de técnicas cooperativas.

6.2.2. Generación de movimientos

Si bien en la sección anterior nos hemos centrado en la cooperación consistente en que el círculo utilice al rectángulo de plataforma, lo cierto es que, para subir a una plataforma ficticia o para bajar de ella, únicamente se necesitan saltos y caídas, que ya generábamos. Aunque hemos tenido que realizar ciertas modificaciones que describiremos a continuación, adelantamos que la parte principal de esta sección se centra en los movimientos CIRCLETILT del rectángulo, generados para resolver situaciones como la de los niveles de las figura 6.6 y 6.7.

Modificaciones a los movimientos del círculo

En primer lugar, vamos a explicar los ajustes que hemos tenido que introducir a los movimientos del círculo. Dado que hemos creado plataformas ficticias para el círculo, sus movimientos ahora pueden detectar que aterrizan en una plataforma ficticia. El problema con esto es que, tal y como diseñamos la simulación de movimientos, el movimiento terminaría al llegar a la plataforma ficticia, cuando realmente, salvo que el rectángulo se encuentre justo en el lugar adecuado y con la forma adecuada, el círculo continuaría cayendo hasta aterrizar en una plataforma real.

De esta manera, alteramos la simulación de movimientos, de forma que no terminara hasta que no se aterrizara en una plataforma real. Como resultado de la simulación, se devolvería una lista de movimientos, cada uno de ellos simulado hasta el instante en el que se aterrizaba en una plataforma, ya fuera ficticia o real. Así, cada posición y velocidad inicial ya no determinan unívocamente el movimiento, sino que generan una familia de ellos. Esto no es problemático en absoluto para nuestros agentes, ya que simplemente se considerarán como movimientos diferentes, que aterrizan en plataformas diferentes. Toda la simulación de movimientos también se realiza desde las plataformas ficticias, para capturar la posibilidad de que el círculo salte o caiga cuando esté sobre el rectángulo para alcanzar un diamante o moverse a una nueva plataforma. En estos casos, hay que tener la precaución de limitar la velocidad máxima horizontal que el círculo puede alcanzar, ya que el espacio disponible para acelerar cuando se encuentra sobre el rectángulo es escaso. Por lo demás, todo el proceso es el mismo.

Antes hemos comentado que había un caso en el que dos plataformas ficticias del círculo no se agrupaban, incluso cuando eran adyacentes, que es cuando una de ellas ha sido creada a partir de una plataforma ficticia del rectángulo. En el rectángulo, tratábamos ese caso generando movimientos de tipo ADJACENT que indicaran que el rectángulo era capaz de moverse de una a otra si se ponía en forma horizontal. Por tanto, decidimos crear movimientos del mismo tipo para el círculo, de forma que, de cara a un futuro plan, ambos agentes combinen movimientos de tipo ADJACENT para que el rectángulo pase de una plataforma real a ficticia (o viceversa), en forma horizontal, cuando el círculo está situado sobre él (ver el siguiente enlace). Sin embargo, por limitaciones de tiempo y no porque añadiera una gran dificultad, no tuvimos ocasión de realizar la implementación correspondiente, por lo que el comportamiento de los agentes en estos casos es muy limitado.

Modificaciones a los movimientos del rectángulo

Los movimientos del rectángulo no requieren de ninguna modificación para adecuarlos a la cooperación, al menos no al generarlos. Cuando hablemos de la ejecución, sí habrá que comentar diferentes aspectos que hemos tenido que añadir o adaptar a este nuevo modo de juego.

En consecuencia, dedicaremos el resto de la sección a hablar de los movimientos CIRCLETILT, el nuevo tipo de movimiento que hemos generado. Recordamos que estos movimientos pretenden identificar situaciones como la de la figura 6.6, en las que el rectángulo debe alcanzar una plataforma que está demasiado alta, necesitando que el círculo haga de apoyo intermedio para conseguir subir. Ilustramos este comportamiento pretendido mediante la animación que se puede encontrar en el siguiente enlace, es decir, que el círculo sirva de punto de apoyo al rectángulo y que, poco después de que colisionen, el círculo de un último impulso al rectángulo para permitirle subir (sin el impulso el rectángulo también lograría subir pero el salto acelera el proceso).

Identificar el patrón a resolver es muy sencillo, ya que se trata del mismo que cuando generábamos los movimientos TILT y HIGHTILT. Es decir, buscamos plataformas del rectángulo que estén contiguas y cuya diferencia de alturas sea, mayor que la que permitiríamos con un TILT, pero menor que una altura máxima cuyo valor exacto concretaremos más adelante. La razón de que queramos que esta altura sea mayor que la que se identifica con un TILT es evitar tener que usar la cooperación cuando no sea estrictamente necesaria. Ya hemos reiterado que la cooperación entre agentes añade mucha dificultad, tanto en el ámbito de planificación como en el de ejecución. Por tanto, ya que los movimientos TILT los resolvemos siempre con eficacia, vemos innecesaria la participación del círculo en esos casos. Además, esto libera al círculo de tener que ayudar al rectángulo, de forma que puede centrarse en su propio plan y ahorrar tiempo.

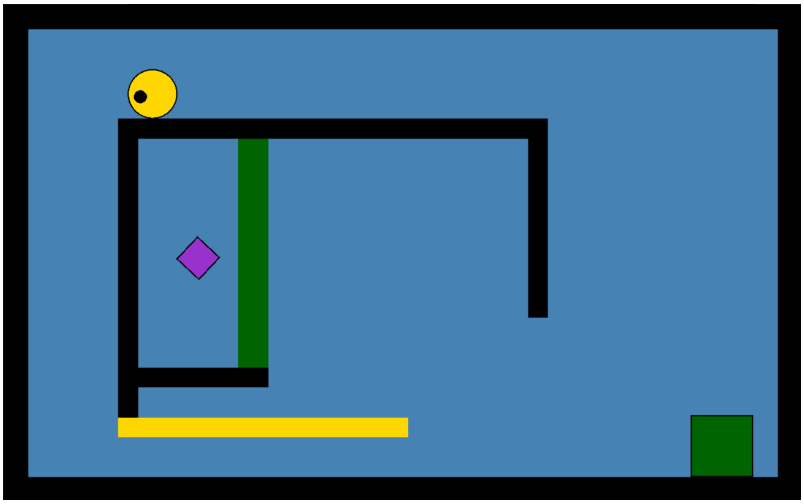


Figura 6.12: Nivel de la figura 5.12 con algunas modificaciones, de forma que se convierte en un nivel cooperativo y la plataforma de abajo ahora la puede atravesar el círculo, pero sigue siendo el rectángulo el que tiene que subir a la de arriba para coger el diamante.

Sin embargo, no podemos decir lo mismo de los movimientos HIGHTILT. Si recordamos la discusión de la sección 5.2.2, no sabíamos con certeza en qué situaciones se va a poder resolver un movimiento HIGHTILT y en cuáles no. Por ejemplo, la configuración específica del nivel de la figura 6.6 impide poder resolver el nivel sin la ayuda del círculo, aunque, en otros niveles en los que la diferencia de altura entre las plataformas era la misma, el movimiento HIGHTILT sí podía llevarse a cabo. En consecuencia, si somos capaces de sustituir estos movimientos por otros que seamos capaces de ejecutar correctamente siempre que los generemos, optaremos por realizarlos siempre de esta segunda manera.

Cabría pensar que en situaciones como las del nivel de la figura 6.12, muy parecido al de la figura 5.12, pero con algunas modificaciones para convertirlo en cooperativo, es preferible realizar un movimiento HIGHTILT en lugar de un CIRCLETILT cooperativo. Para el rectángulo las plataformas son iguales que las del nivel de la figura 5.12 y este podía ser resuelto individualmente con un HIGHTILT. Aunque pudiera parecer problemático querer resolverlo con un movimiento de tipo CIRCLETILT, ya que el círculo atraviesa el obstáculo amarillo cuando intenta hacer de apoyo para el rectángulo y hay un riesgo de que el rectángulo desplace al círculo como punto de apoyo y este no consiga subir, lo cierto es que nuestros agentes resuelven este nivel con solvencia, como demuestra la animación del siguiente enlace. Por tanto, reafirmamos que la preferencia de CIRCLETILT por HIGHTILT es adecuada.

Consecuentemente, solo falta hablar de la altura máxima para la que se siguen detectando estos movimientos. Tras diferentes pruebas, concluimos que la altura máxima a la que se puede subir realizando un movimiento CIRCLETILT es de 18 cuadrados de 8×8 píxeles del juego, bastante superior a los 13 con los que limitábamos los movimientos HIGHTILT. Como referencia, la altura máxima que alcanza el rectán-

gulo está alrededor de 24 cuadrados de 8×8 píxeles. Aunque puede conseguirse que el rectángulo alcance plataformas mucho más altas, con diferencias de altura incluso mayores que 30 cuadrados, el movimiento que logra esta hazaña es muchísimo más complejo que el que teníamos intención de implementar con los CIRCLETILT. Como muestra, en este enlace se puede apreciar cómo esto es posible, aunque advertimos de que es muy complicado lograrlo. En consecuencia, decidimos considerar como altura máxima aquella que se podía sortear realizando el movimiento básico, es decir, 18 cuadrados.

Esto concluye la sección de generación y adaptación de movimientos cooperativos. Como puede observarse, se trata de una sección mucho más breve que las análogas del círculo y del rectángulo. Esto se debe a que gran parte del trabajo ya estaba hecho al tratar los casos individuales del círculo y del rectángulo. A pesar de que, como veremos, estos movimientos son más que suficientes para obtener excelentes resultados en las pruebas, hemos dejado una situación cooperativa sin tratar que es la relativa a la figura 6.3, en la que el círculo debía impulsar al rectángulo mientras este cae. Este es un reto que todavía no hemos abordado por su complejidad, tanto a nivel de identificar un patrón general como en ejecución y sería una línea muy interesante para continuar este trabajo.

6.2.3. Filtrado de movimientos

Dados los pocos movimientos que hemos añadido durante esta parte del desarrollo, el filtrado de movimientos es prácticamente idéntico al que se realiza con los agentes individuales. Para el rectángulo, y debido a la discusión de la sección anterior, cuando se detecta un movimiento CIRCLETILT que podría ser resuelto mediante un HIGHTILT, ambos movimientos se dejan, en caso de que se trate de una plataforma a la que el círculo no pueda llegar (el HIGHTILT puede ser necesario) o de que el HIGHTILT generado sea falsamente detectado (por lo que el CIRCLETILT puede ser necesario). Tomar uno u otro dependerá del nivel, aunque priorizando el CIRCLETILT en la fase de planificación por no ser arriesgado. Hacemos notar que estos son los únicos movimientos comparables a los CIRCLETILT, ya que hemos tenido la precaución de generarlos solamente cuando un TILT es insuficiente.

Donde sí se necesita un filtrado de movimientos más elaborado es con el círculo. En concreto, cuando se realiza la agrupación de plataformas ficticias en plataformas simplificadas. Al igual que en el rectángulo descartábamos los movimientos ADJACENT que indicaban la conectividad entre dos plataformas no simplificadas reales, en este caso, cuando agrupamos plataformas, encontraremos muchos saltos o caídas que van a parar a plataformas no simplificadas diferentes, pero que se van a agrupar en la misma plataforma simplificada. De igual manera, habrá muchos saltos que partan de diferentes alturas de una misma plataforma simplificada.

Solamente nos basta tener, para cada plataforma simplificada del círculo, una ma-

nera de que el círculo suba encima del rectángulo. Y de manera análoga, cuando el círculo salta desde una plataforma ficticia, solamente nos basta tener un movimiento de entre todos los movimientos que aterricen en una misma plataforma real. Tras agrupar plataformas no simplificadas, realizamos este filtrado en dos etapas:

En primer lugar, se descartan aquellos movimientos que no son de tipo ADJACENT y conectan plataformas ficticias, salvo que sea un movimiento que coge diamantes sin cambiar de plataforma. Esto representa o bien la situación en la que el círculo quiere cambiar de altura del rectángulo o bien la situación en la que el círculo quiere saltar estando encima del rectángulo a otra plataforma alejada, pero aterrizando también encima del rectángulo. Estas últimas, si bien son posibles en determinados casos, preferimos que se dividan en más pasos para facilitar su ejecución.

En segundo lugar, comparamos todos los movimientos que han pasado el primer filtro dos a dos, realizando las mismas comparaciones que cuando se comparaban en el agente círculo individual. A esta comparación, hemos añadido los casos en los que se parte o se aterriza en plataformas ficticias que pertenecen a la misma plataforma simplificada. A continuación, consideraremos dos movimientos que parten y aterrizan en las mismas plataformas simplificadas y alcanzan los mismos diamantes.

Si la plataforma de salida es ficticia, queremos aquel movimiento que deje una distancia prudencial con los obstáculos con los que puede colisionar durante la trayectoria. Si ambos cumplen este requisito, nos queremos quedar con el movimiento que parta de una altura más baja, siempre y cuando deje una distancia adecuada con el extremo de la plataforma de aterrizaje más cercano al punto de salto.

Si el criterio anterior no decide, o la plataforma de salida es real, se comprueba si la plataforma de llegada es ficticia. Nos queremos quedar con aquella que aterrice con la forma más estable posible, es decir, priorizaremos caer sobre el rectángulo cuando este esté en forma horizontal, después en forma cuadrado, y, si no hay más remedio, en forma vertical.

Si ambos movimientos son iguales según el criterio anterior o ese criterio no procede, se vuelve a comprobar si la plataforma de salida es ficticia, y, en el caso de que los dos movimientos dejen una distancia prudencial con los obstáculos durante el vuelo, nos quedaremos con el que parta de una velocidad horizontal más pequeña, mientras deje una distancia adecuada al extremo por el que el círculo podría caer si aterriza con demasiada velocidad.

Si nada de lo anterior se puede aplicar, se compara de la misma forma que el agente individual. El sistema que tenemos puede parecer rebuscado y poco intuitivo, pero es el resultado de muchas pruebas y niveles que los agentes no eran capaces de resolver por elegir movimientos poco adecuados. Esta combinación de movimientos prioriza, en la medida de lo posible, partir y aterrizar sobre el rectángulo cuando este está lo más horizontal posible, pues es así como es más estable.

Esto concluye la sección de representación del nivel. Hemos hablado de las nuevas plataformas ficticias y simplificadas del círculo, de los movimientos nuevos que generamos y de cómo los filtramos. En la siguiente sección, explicaremos cómo se determina el plan cooperativo a seguir.

6.3. Trazado de un plan a seguir

En esta sección vamos a exponer y ejemplificar cómo se realiza el trazado de un plan cooperativo. Hay que tener en mente que debemos hacer un plan para cada uno de los agentes y que, en fase de ejecución, su interpretación será análoga a la de los planes individuales del rectángulo y el círculo. Esto es que, en primer lugar, cada personaje alcanza todos los diamantes disponibles mediante movimientos que terminan en su plataforma actual y, cuando no quedan más, utiliza el primer movimiento del plan para cambiar de plataforma y vuelve a repetir el proceso. En el caso de la cooperación, debemos tener en cuenta que hay movimientos de uno de los personajes que requieren la ayuda del otro, para lo cual debe estar en una plataforma apropiada. Asimismo, se deben poder incluir movimientos de espera dentro de una misma plataforma mientras el otro agente puede estar alcanzando un diamante en otra zona del nivel. En lugar de presentar un nuevo algoritmo como el de la sección 3.3.2, vamos a explicar las pocas modificaciones que se deben realizar sobre el algoritmo 1, ya que la estrategia para encontrar un plan cooperativo sigue siendo una búsqueda en anchura con control de repetidos.

El primer cambio destacable es que ahora los nodos contendrán dos planes parciales, el del círculo y el del rectángulo. No obstante, seguirán manteniendo la información de los diamantes recogidos por ambos agentes hasta el momento.

En segundo lugar, debemos explicar cómo se procesan ahora los movimientos y las plataformas, es decir, cómo se asignan ahora los diamantes que se van capturando. En el algoritmo 1, las líneas 11-13 estaban dedicadas a este efecto. En esta ocasión hacemos lo mismo, pero para los dos planes, es decir, tras extraer un nodo de la cola, consideramos el último movimiento de los planes del círculo y el rectángulo, que llamamos respectivamente $move_c$ y $move_r$. Procesamos estos movimientos, lo que significa que añadimos a los diamantes capturados del nodo los diamantes recogidos por estos movimientos, y consideramos las plataformas de aterrizaje de $move_c$ y $move_r$, que denotamos por p_c y p_r. Seguidamente, procesamos p_c y p_r, es decir, para el círculo añadimos a los diamantes capturados del nodo los diamantes alcanzables desde p_c con un movimiento que finaliza en p_c y lo análogo para el rectángulo. Únicamente resta por explicar cómo expandimos ahora los nodos.

Para ello, consideramos l_c y l_r las listas de movimientos que parten desde las plataformas p_c y p_r, respectivamente. A estas listas vamos a añadir un movimiento auxiliar para mantenerse en la misma plataforma, que nos servirá para modelizar tanto la

situación en la que un agente debe esperar al otro, como la ayuda en una hipotética acción cooperativa. Con estas consideraciones, los nodos expandidos se forman considerando todas las posibles combinaciones *compatibles* en las que el círculo toma un movimiento m_c de l_c y el rectángulo un movimiento m_r de l_r. De esta manera, el nodo será el mismo que antes salvo que los planes tendrán un paso más, concretamente, m_c el plan del círculo y m_r el plan del rectángulo. De aquí se sigue que los planes del círculo y el rectángulo son siempre de igual longitud. Solamente falta comentar cuándo un movimiento del círculo y uno del rectángulo son *compatibles* y cuándo no lo son.

Primeramente, no queremos que se expandan nodos en los que ambos agentes permanecen en su misma plataforma, luego la elección de movimientos auxiliares de espera por parte del círculo y del rectángulo simultáneamente no es compatible. De igual manera, en las listas l_c y l_r puede haber movimientos que no sean de espera ni cooperativos y que finalicen en la misma plataforma de la que parten, los cuales también queremos descartar porque los movimientos auxiliares ya capturan ese comportamiento. Las demás situaciones en las que una combinación de movimientos puede no ser compatible vienen dadas por las restricciones impuestas por la colaboración.

Cuando el movimiento del rectángulo es un CIRCLETILT, para poder expandir el nodo, se necesita que el círculo se encuentre en la misma plataforma y así le pueda ayudar a realizar ese movimiento cooperativo. Habrá que descartar entonces las combinaciones de movimientos en las que el rectángulo elija un CIRCLETILT y el círculo no elija el movimiento auxiliar cooperativo o no se encuentre en la misma plataforma del rectángulo. Algo similar sucede con los saltos o caídas del círculo desde o hacia plataformas ficticias, es decir, movimientos del círculo iniciados sobre el rectángulo o que pretenden aterrizar sobre este. Estos movimientos solamente serán compatibles si el rectángulo se encuentra en la plataforma adecuada y el movimiento que elige es el movimiento auxiliar de cooperación. Por último, y aunque ya advertimos que no implementamos su ejecución, el círculo puede ejecutar un movimiento de tipo ADJACENT solo si el rectángulo también realiza un movimiento de tipo ADJACENT y en la misma dirección.

Aunque pueda parecer que las combinaciones de movimientos pueden explotar, haciendo la búsqueda impracticable, lo cierto es que nuestro algoritmo ha dado resultados satisfactorios en todos los niveles en los que lo hemos probado. El hecho de que haya muchas restricciones de incompatibilidad entre movimientos y que los planes no suelan tener longitud mayor que 4, garantiza la terminación de la búsqueda. Para terminar de aclarar cómo se expanden los nodos, que quizás es el detalle más complicado del algoritmo, y para comprobar las restricciones impuestas por la compatibilidad entre movimientos, recomendamos atender al ejemplo del nivel de la figura 6.13.

Desde la situación inicial, la lista de movimientos l_c del círculo tiene longitud 4 y consiste en saltar a la plataforma real inmediatamente superior, a la plataforma

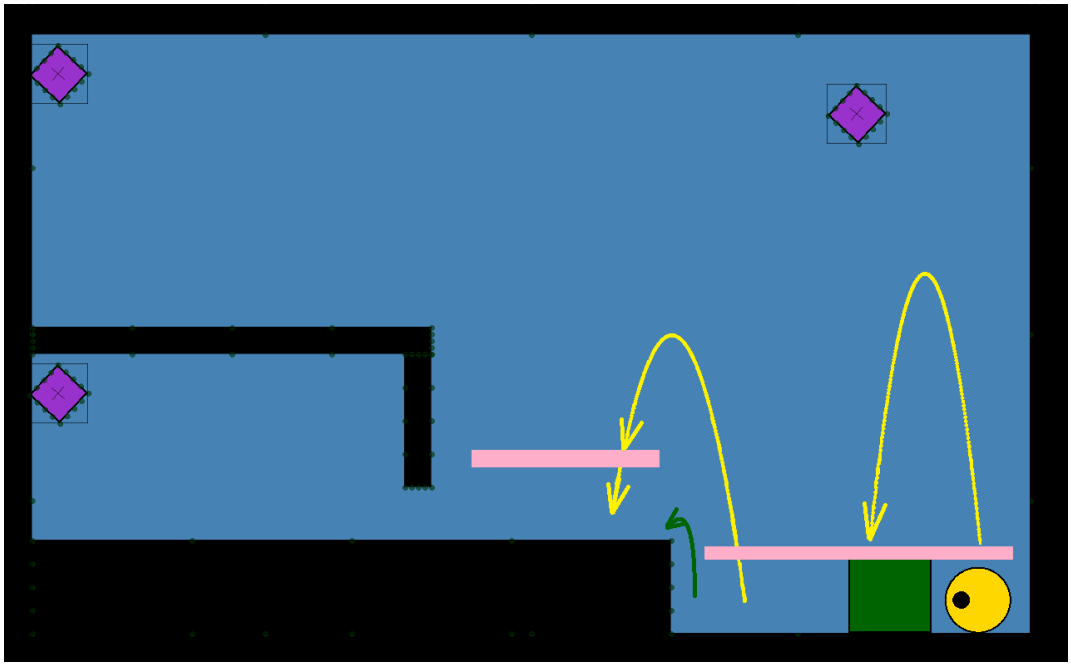


Figura 6.13: Nivel 1 de la competición del año 2016, en el que se indican los posibles movimientos de los agentes desde la situación inicial.

ficticia rosa de la izquierda, subir sobre el rectángulo y el movimiento auxiliar de espera/cooperativo. Por su parte, el rectángulo solamente puede subir a la plataforma inmediatamente superior con un CIRCLETILT o generar un movimiento auxiliar de espera/cooperativo. De las ocho combinaciones de movimientos, solamente son válidas tres: que el círculo salte a la plataforma superior y el rectángulo espere en su plataforma; que el círculo salte sobre el rectángulo y este último elija el movimiento auxiliar; y que el rectángulo realice un CIRCLETILT, para lo que el círculo deberá elegir el movimiento auxiliar.

Supongamos, por ejemplo, que se expande el primero de los nodos generados, en el cual los agentes se encontrarían en una situación como la imagen superior izquierda de la figura 6.14. Desde esa situación, la lista de movimientos del círculo es saltar a la plataforma superior, saltar sobre el rectángulo, saltar a la plataforma inferior, saltar a la plataforma ficticia rosa de la izquierda y el movimiento auxiliar. Por su parte, los movimientos del rectángulo vuelven a ser los mismos, pero esta vez nunca se podrá elegir el CIRCLETILT porque el círculo no se encuentra en la misma plataforma. Como el rectángulo solo puede elegir el movimiento auxiliar, el círculo ya no puede elegirlo y se vuelven a generar tres nodos (todos los saltos menos el que acaba en la plataforma ficticia de la izquierda porque el rectángulo no está en la plataforma adecuada), frente a los teóricos $5 \times 2 = 10$.

Si se expandiera el segundo de los nodos, es decir, el círculo salta sobre el rectángulo, nos encontraríamos en una situación como en la de la imagen superior derecha de la figura 6.14. Al procesar la plataforma del círculo, se identificaría que puede alcanzar el diamante superior y después se procedería a expandir los nodos. El rectángu-

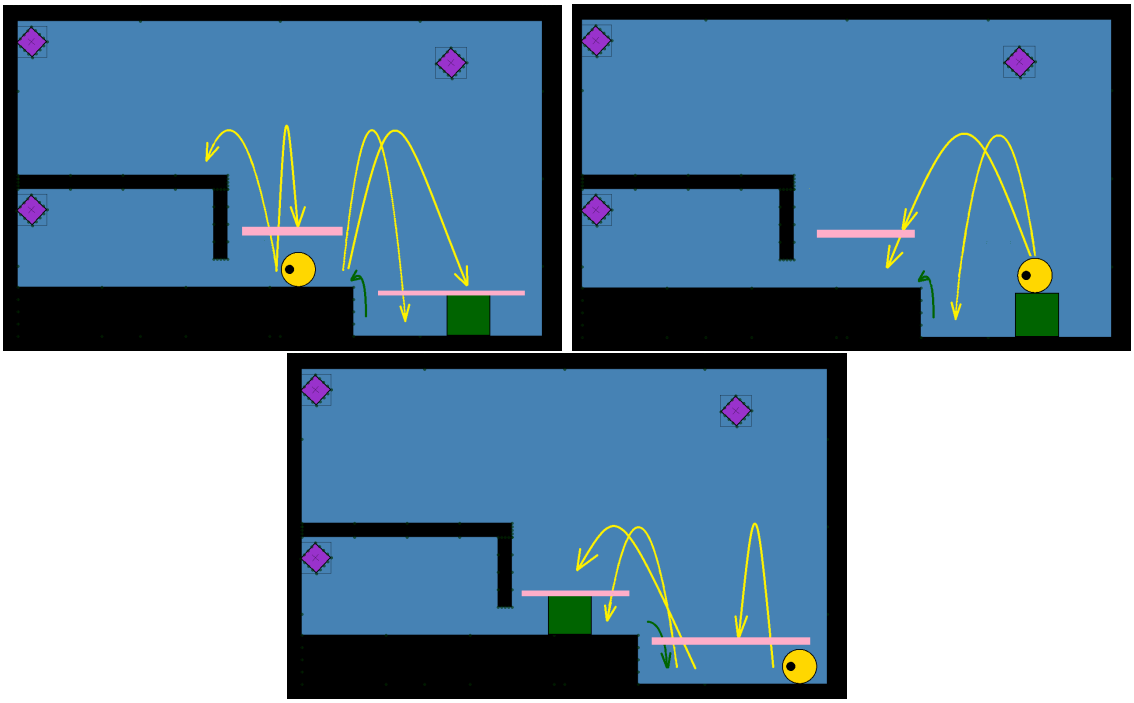


Figura 6.14: Nodos generados a partir del inicial donde las líneas rosas representan plataformas ficticias del círculo.

lo, aunque vuelve a tener dos movimientos, solamente puede elegir el movimiento auxiliar cooperativo porque el círculo está sobre él, y el círculo puede elegir entre caer a la plataforma inferior o saltar a la de altura intermedia, pero no puede elegir su movimiento auxiliar ni puede caer sobre la plataforma ficticia de la izquierda. Volvemos a reducir las ocho posibilidades iniciales a solamente dos.

Por último, si se expandiera el último nodo, es decir, el círculo ayuda al rectángulo a realizar un CIRCLETILT, los agentes se encontrarían como en la situación de la imagen inferior de la figura 6.14. Entonces, al procesar la plataforma del rectángulo se identificaría que alcanza el diamante de la izquierda y después se procedería a expandir los nodos. Los posibles movimientos del rectángulo son una caída y un movimiento auxiliar y los del círculo son un salto a la plataforma de altura intermedia, un salto sobre el rectángulo, un salto a la plataforma ficticia de la derecha y un movimiento auxiliar de espera. De las ocho posibilidades únicamente son posibles cuatro: o bien que el rectángulo elija el movimiento auxiliar y el círculo aterrice sobre él o en la plataforma de altura intermedia, o bien que el rectángulo caiga y el círculo espere o salte a la plataforma de altura intermedia.

En resumen, aunque el factor de ramificación teórico sea aproximadamente 8 en un nivel como el de la figura 6.13 (que es un nivel bastante estándar en cuanto a número de plataformas y conectividad), las restricciones impuestas a la hora de expandir nodos hacen que este factor de ramificación se reduzca a 3 en la práctica. En este nivel, en el que el plan es de los más largos porque tiene 5 pasos, en la práctica se expanden únicamente unos $3^5 = 243$ nodos frente a los aproximadamente

$8^5 = 32.768$ teóricos.

6.4. Ejecución del plan

Para concluir la arquitectura de nuestros agentes cooperativos, vamos a explicar cómo deciden qué acciones tienen que tomar en cada momento para alcanzar todos los diamantes, basándonos en los sistemas ya implementados para los agentes individuales. Partimos del plan cooperativo, que se compone de dos planes, uno para cada agente. Como en el caso individual, lo primero que hace cada agente al llegar a una nueva plataforma es alcanzar todos los diamantes que se pueden coger con movimientos que aterrizan en esa misma plataforma. Salvo el caso en el que el agente es el círculo y la plataforma sobre la que está es ficticia, es decir, está sobre el rectángulo, todo lo demás es igual que cuando los agentes actuaban de forma individual. Resumidamente, cada agente identifica el movimiento más cercano que parte de esa plataforma y alcanza un diamante y, mediante los sistemas de reglas basados en las ecuaciones del movimiento, se dirige al punto con la velocidad apropiada como se explicó en las secciones 4.3 y 5.4.

Por otro lado, cuando no tienen ningún diamante más que alcanzar en la plataforma en la que se encuentran, los agentes deben ejecutar el primer movimiento de su plan. Si este no es cooperativo, es decir, para el rectángulo no es un CIRCLETILT o un movimiento auxiliar y para el círculo no es un movimiento que aterrice o despegue sobre una plataforma ficticia o un movimiento auxiliar, todo trascurre como lo hacía en los casos individuales. En las siguientes secciones abordaremos los casos cooperativos y que constituyen las diferencias fundamentales con la fase de ejecución de los agentes individuales.

6.4.1. Ejecución de los movimientos de tipo CIRCLETILT

En primer lugar, nos planteamos la ejecución de los movimientos de tipo CIRCLETILT. Para ello, asumimos que el círculo y el rectángulo se encuentran en la misma plataforma, que el primer paso del plan del rectángulo es un CIRCLETILT y que el primer paso del plan del círculo es un movimiento auxiliar que se interpreta como que el círculo tienen la obligación de ayudar al rectángulo. Si este no fuera el caso, el sistema de replanificación entraría en juego porque el plan no es correcto. El procedimiento para ejecutar estos movimientos es sencillo, teniendo en mente que buscamos imitar un comportamiento como el descrito en la animación del siguiente enlace.

En primer lugar, se le indica al círculo que se dirija, mediante el sistema de reglas físico, al borde de la plataforma en el que se debe posicionar para hacer de punto de apoyo del rectángulo, al lado de la plataforma destino del rectángulo y con velocidad

0. Mientras tanto, se le notifica al rectángulo que se dirija al extremo contrario de la plataforma (para que ambos agentes no se estorben y que tenga suficiente espacio para acelerar) y que espere a que el círculo le dé la señal de que está listo para realizar el movimiento. Cuando el círculo se sitúa en la posición apropiada, envía una señal al rectángulo, que, al recibirla, se dirige con velocidad máxima hacia el punto donde se encuentra el círculo y mantiene la acción hasta que aterriza en la plataforma destino. Adicionalmente, cuando el rectángulo está sobrevolando al círculo, este último salta para darle un pequeño impulso y acelerar el proceso. El único inconveniente que surge en estos movimientos es cuando el rectángulo se encuentra entre el extremo por donde se debe realizar el CIRCLETILT y la posición del círculo, ya que, en ese caso, ambos agentes deben intercambiar sus posiciones. Trataremos estos intercambios de posiciones en un contexto más general en la sección 6.4.4.

6.4.2. Ejecución de movimientos del círculo que aterrizan en el rectángulo

El segundo reto cooperativo se da cuando el círculo debe aterrizar sobre una plataforma ficticia, es decir, debe aterrizar encima del rectángulo. Para ello, antes de iniciar el salto o la caída, el círculo debe esperar a recibir una señal informándole de que el rectángulo está listo para servirle de plataforma de aterrizaje, es decir, está en el lugar donde aterrizará el círculo. Lo que debe hacer el rectángulo es ir, mediante el sistema de reglas físico, a esa posición de aterrizaje, que ya se calculó durante la generación de movimientos, y situarse en la forma adecuada para recibir al círculo (se priorizaba la horizontal siempre que era posible por ser más estable). Cuando haya llegado a dicha posición con velocidad 0, activará una señal que le indicará al círculo que ya está listo para servirle de plataforma. Al recibir el círculo la señal, realiza el salto o caída de igual forma que lo haría en el caso individual. A este sistema sencillo, que se puede ver en funcionamiento en la animación del siguiente enlace, le añadimos varias optimizaciones que mejoran enormemente su rendimiento.

Recálculo de la trayectoria del círculo

Durante los saltos y caídas del círculo, permitíamos que la velocidad horizontal de partida fuera ligeramente distinta de la velocidad objetivo, siempre y cuando el movimiento acabara en la misma plataforma. Como estas pequeñas variaciones en la velocidad horizontal inicial hacían que el círculo aterrizara en un borde del rectángulo, desestabilizándolo, o incluso que aterrizara fuera del rectángulo, decidimos que el rectángulo recalculara la nueva posición de caída del círculo. Este cálculo utiliza el mismo sistema que en la generación de movimientos de saltos y caídas, con la ventaja de que la velocidad horizontal mientras el círculo está en el aire es exacta y nunca varía (salvo colisiones con obstáculos). Durante el vuelo del círculo, el rectángulo intercala llamadas al sistema físico de simulación de parábolas con acciones para

dirigirse a la nueva posición de aterrizaje del círculo, que es cada vez más precisa.

Sistema de consolidación del aterrizaje

Debido a que el juego Geometry Friends se desarrolla en un entorno físico realista, cuando el círculo aterriza sobre el rectángulo, lo suele hacer con bastante velocidad vertical y se suelen producir pequeños rebotes. Si el rectángulo, inmediatamente después del aterrizaje del círculo, pasara a realizar su siguiente tarea, el círculo caería del rectángulo en la mayor parte las ocasiones. Por tanto, durante el segundo inmediatamente siguiente al primer impacto del círculo con el rectángulo, el rectángulo tiene como deber evitar que el círculo se caiga. Para lograrlo, se le indica que se dirija a la posición horizontal en la que tiene su centro el círculo, con lo que conseguimos alinear los centros de ambos personajes y permitimos que se consolide el aterrizaje. Se puede apreciar cómo el rectángulo recalcula la posición en la que el círculo va a aterrizar, cómo adapta su colocación y cómo espera que el círculo termine de aterrizar en la animación del siguiente enlace.

Ajuste de la posición de espera del rectángulo y trayectorias alternativas del círculo

Son muchas las ocasiones, como en el nivel de la figura 6.15, en las que la trayectoria del círculo para situarse sobre el rectángulo, colisiona con el propio rectángulo. Es decir, en el nivel de la figura, para alcanzar cualquiera de los diamantes de la parte superior del nivel, el círculo debe primero saltar y aterrizar sobre el rectángulo, pero el movimiento que tiene asignado para hacerlo es de tipo JUMPMOVE y tiene velocidad horizontal 0. Si el rectángulo se situara en la posición de aterrizaje del círculo, el círculo colisionaría con el rectángulo al saltar. Esto se debe a que, en la generación de movimientos del círculo, asumíamos que el rectángulo no se encontraba en ninguna parte del nivel y el círculo, si bien tenía la capacidad de aterrizar sobre él (en las plataformas ficticias), no tenía la limitación de tener que esquivarle en sus saltos y caídas.

Para que todo funcione correctamente cuando el rectángulo interseca la trayectoria del círculo, el rectángulo evalúa si, situándose ligeramente a la izquierda o ligeramente a la derecha del punto de aterrizaje, sigue intersecando con la trayectoria del círculo. Evidentemente, no siempre va a poder situarse ligeramente a la izquierda o a la derecha porque, por ejemplo, en el nivel de la figura 6.15, no puede moverse a la izquierda ya que hay una pared. Sin embargo, son muy escasas las ocasiones en las que no puede situarse en ninguno de los dos lados. Una vez el rectángulo estuviera en su nueva posición, el círculo saltaría y, gracias a que se recalculan las trayectorias, el rectángulo acaba situándose en el punto de aterrizaje en el momento en el que se le necesita, como se aprecia en la animación del siguiente enlace.

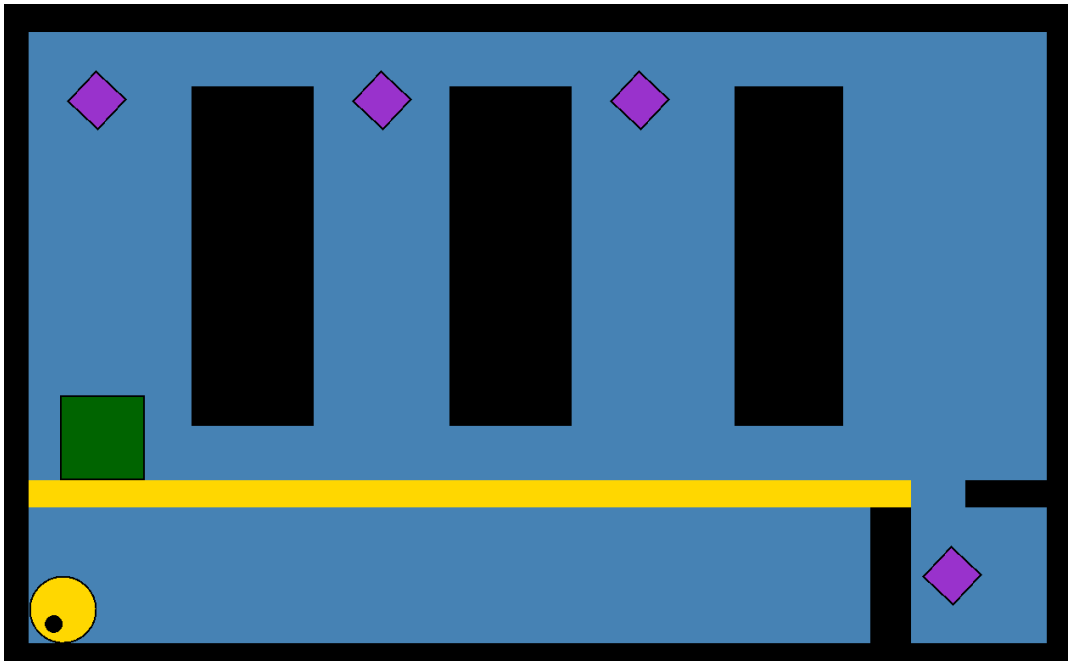


Figura 6.15: Nivel 8 de la competición del año 2013, en el que el rectángulo está en la trayectoria de salto del círculo para situarse sobre él.

Cuando no conseguimos solucionar el problema situando al rectángulo ligeramente a la derecha o a la izquierda del punto de aterrizaje, tenemos que cambiar de estrategia. Lo que hacemos es volver a generar todos los movimientos desde la plataforma en la que se encuentra el círculo y filtramos aquellos que aterrizan en el rectángulo, pero considerando al rectángulo como un obstáculo fijo más. De entre todos ellos, elegimos el mejor movimiento que no interseca con el rectángulo pero aterriza sobre él, de acuerdo al filtrado habitual de parábolas. De encontrar un movimiento adecuado, se reemplaza el salto del plan por el nuevo movimiento.

Este proceso es bastante costoso en tiempo porque supone recalcular muchas trayectorias, demorando algunos segundos la resolución del nivel. Sin embargo, este problema no es muy significativo, ya que las trayectorias se recalculan justo cuando se va a comenzar a ejecutar el movimiento, mientras los agentes aún están posicionándose. En consecuencia, y dado que son muy pocos los niveles en los que el problema no se soluciona haciendo que el rectángulo se aparte ligeramente, consideramos esta solución suficientemente buena para nuestros propósitos.

6.4.3. Ejecución de movimientos del círculo que parten del rectángulo

En este apartado vamos a suponer que el círculo se encuentra sobre el rectángulo, ya ha aterrizado y también ha consolidado su posición. Se plantean dos posibilidades no excluyentes: el siguiente movimiento del círculo es para capturar un diamante o

el siguiente movimiento le sirve al círculo para cambiar de plataforma. En cualquiera de los dos casos, el círculo debe situarse en una determinada posición dentro de la plataforma ficticia en la que se encuentra y alcanzar una determinada velocidad. Para ello, es el rectángulo el que debe transportar al círculo a su posición objetivo. Por tanto, para realizar ese reto, simplemente se le indica al rectángulo que su posición objetivo es la posición desde la que parte el siguiente movimiento del círculo y su velocidad objetivo es 0. Por su parte, el círculo, para evitar caer del rectángulo mientras es transportado, tiene como objetivo mantenerse en el centro del rectángulo. Eso es sencillo lograrlo, ya que basta indicarle que se dirija a esa posición y que lo haga con velocidad nula (velocidad relativa a la velocidad del rectángulo). Este sistema tan sencillo es sorprendentemente muy efectivo y permite al rectángulo ir tan rápido como pueda para llegar lo antes posible y sin preocuparse de si el círculo cae o no, porque es el propio círculo el que se encarga de mantenerse sobre el rectángulo.

En lugar de dirigirse al punto objetivo, el rectángulo se sitúa ligeramente a la izquierda o a la derecha para que el círculo aproveche toda la anchura del rectángulo como plataforma y tenga espacio para acelerar. Cuando ha estabilizado su posición y ha adoptado la forma apropiada, lanza una señal al círculo para que este deje de preocuparse por caer del rectángulo y proceda a realizar el movimiento que tenía planificado. El círculo, actuando como lo haría en el caso individual, salta o cae según proceda. La animación del siguiente enlace muestra el comportamiento descrito. En ese momento se plantean 2 alternativas: o bien la plataforma de aterrizaje del círculo es real o bien es ficticia, es decir, salta del rectángulo para volver a aterrizar en él. Este segundo caso ya lo hemos tratado en la sección 6.4.2 y el rectángulo simplemente recalcula la posición de aterrizaje del círculo y se dirige a ella.

El primero de los casos es que la plataforma donde el círculo tiene previsto aterrizar sea real. En este caso también hay dos alternativas, que esta plataforma se encuentre más arriba o más abajo de la posición del rectángulo. Si se encuentra más arriba, y en previsión de un salto impreciso por las complicaciones de la física del juego, el rectángulo está prevenido y se dirige a la posición donde el círculo caería si no consigue completar el salto. Esta posición se recalcula continuamente con llamadas al sistema de simulación de trayectorias parabólicas. Un ejemplo de un salto no exitoso en el que el rectángulo previene una caída y ahorra tiempo se puede ver en la animación del siguiente enlace. Si la altura de la plataforma de aterrizaje del círculo es inferior a la posición del rectángulo, entonces el rectángulo podría estorbar al círculo e impedirle bajar si no le indicamos lo contrario. Por tanto, se calcula la posición horizontal en la que el círculo se encontrará cuando esté a la altura del rectángulo (nuevamente con la simulación de las trayectorias parabólicas), y se le indica al rectángulo que se situé ligeramente a la izquierda o a la derecha de tal manera que no colisione con el círculo. El comportamiento de nuestros agentes en esta situación es el que se muestra en la animación del siguiente enlace.

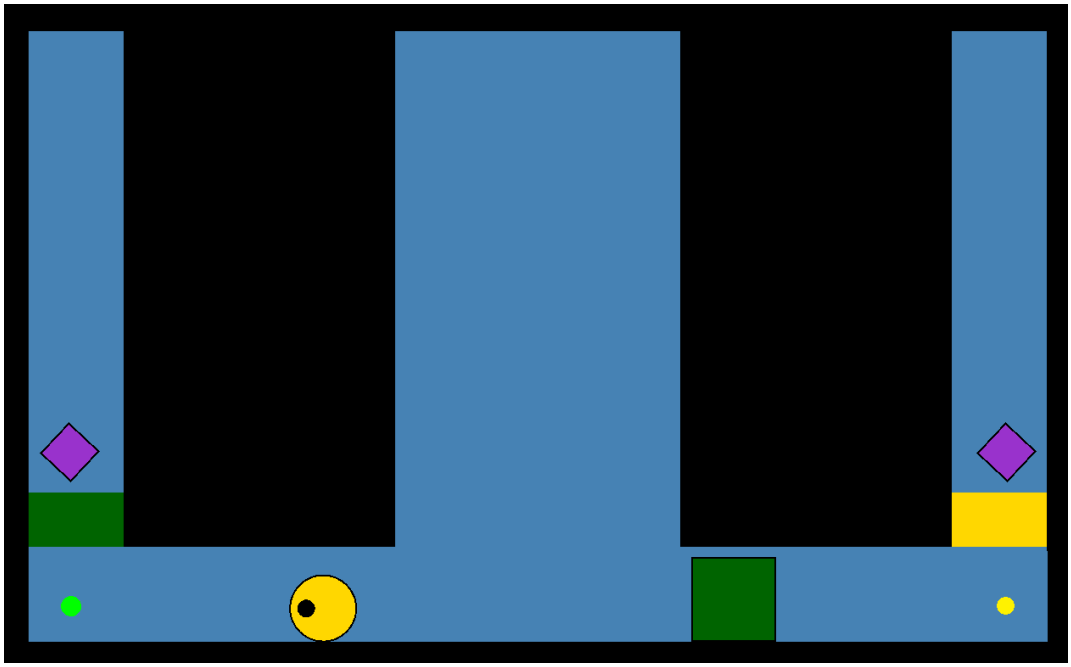


Figura 6.16: Nivel 9 de la competición cooperativa del año 2015, en el que los agentes no están ordenados igual que sus puntos objetivos. El punto objetivo del círculo (amarillo) está a la derecha del punto objetivo del rectángulo (verde), pero el círculo no está a la derecha del rectángulo.

6.4.4. Intercambios de posición

Una situación aparentemente sencilla y que nos ha supuesto un quebradero de cabeza durante el desarrollo de los agentes cooperativos viene dada por los intercambios de posición. Cuando los agentes se encuentran en la misma plataforma pero no se encuentran ordenados respecto a los puntos objetivos donde desean ir (ver figura 6.16), deben intercambiar sus posiciones para estar en el orden correcto.

Aunque en el nivel de la figura parece evidente que el círculo debe saltar al rectángulo en el centro del nivel, esta situación no es fácilmente generalizable. En algunas ocasiones, no hay espacio suficiente para que el círculo salte por encima del rectángulo, y no es sencillo identificar en qué parte de la plataforma es posible hacer el intercambio. Para realizar los intercambios optamos por un sistema alternativo. Los agentes se dirigirán el uno hacia el otro, con el rectángulo en forma vertical. Con ello pretendemos que el círculo sirva de pivote para el rectángulo y le permita voltearse e intercambiar posiciones con el círculo. Podemos ver una demostración del sistema de intercambio de posiciones en la animación del siguiente enlace.

Sin embargo, este método tiene sus limitaciones cuando los agentes cuentan con poco espacio para realizar el intercambio porque hay obstáculos cerca. En ocasiones, los agentes se quedan atascados y no podrían continuar si no fuera por el sistema de recuperación que presentamos a continuación.

6.4.5. Sistema de recuperación

De igual manera que hiciéramos con el rectángulo, hemos implementado un sistema que identifica cuándo los personajes están atascados y toma acciones aleatorias para salir de ese estado. Concretamente, se evalúa si los agentes están suficientemente próximos y permanecen en la misma posición durante un periodo de tiempo mayor que 3 segundos. De darse ese caso, los agentes comienzan a tomar acciones aleatorias que mantienen durante 300 ms. Para el círculo, la probabilidad de realizar un salto (una medida algo más drástica e imprevisible que rodar a izquierda o derecha) es de solamente un 10 %, mientras que las otras dos acciones tienen ambas una probabilidad del 45 %. Una pequeña optimización que mejora la actuación de los agentes en algunos niveles y no les perjudica en ninguno es incrementar el tiempo que se mantiene la acción aleatoria en 100 ms cada vez que los agentes vuelven a quedar atascados. De esta manera, si los agentes quedan recurrentemente atascados, las acciones aleatorias se tomarán durante más tiempo y hemos comprobado empíricamente que es más probable que salgan del estado de bloqueo, especialmente en situaciones en las que, aunque consigan separarse, es fácil que vuelvan a quedar bloqueados.

Esto finaliza las explicaciones relativas a la ejecución de movimientos por parte de los agentes y la coordinación. En la siguiente sección, expondremos un sistema de explicabilidad que permite a los usuarios ajenos al proyecto entender las decisiones de los agentes.

6.5. Visualización y explicabilidad de la cooperación

Una vez ya hemos desarrollado en profundidad el funcionamiento completo de nuestros agentes cooperativos, queremos detenernos para abordar el problema de la explicabilidad. Como desarrolladores, nosotros muchas veces sabemos qué es lo que está haciendo cada agente en cada momento o cuáles son sus subojetivos a corto plazo, pero, para una persona ajena al proyecto, hemos identificado que existen problemas para comprender las decisiones que toman los agentes. Por ese motivo, y con ayuda del depurador visual que viene incorporado con el juego, hemos desarrollado un sistema de explicabilidad que permita entender la motivación del agente para realizar las acciones o movimientos a distintos niveles.

Este problema de comprensión no se da únicamente con los agentes cooperativos, sino que también afecta, aunque en menor medida, a los agentes individuales. Por ello, y por las limitaciones temporales, nos hemos restringido a desarrollar el sistema de explicabilidad exclusivamente para los agentes cooperativos. Sin embargo, los agentes individuales también muestran cierta información por pantalla, aunque esta

está menos ordenada y puede ser quizás menos comprensible para un usuario ajeno al proyecto, ya que es la que utilizábamos como desarrolladores para depurar el código. Aun así, consideramos que es suficiente para interpretar las acciones tomadas por los agentes individuales, ya que estas son muy intuitivas la mayor parte de las veces.

Para desarrollar el sistema de explicabilidad cooperativo, lo primero que debemos hacer es presentar de manera *oficial* al sistema de depuración visual, y apostillamos lo de *oficial*, ya que, en el fondo, ha estado presente en muchas de las figuras que han aparecido en la memoria hasta el momento. Este sistema nos ha posibilitado la elaboración de gran cantidad de las figuras de imágenes del juego durante su ejecución, y sobre las que hemos podido añadir parábolas, texto, píxeles u otras trayectorias e información. Para poder mostrarlo y ocultarlo, el usuario de la aplicación debe pulsar la tecla `F1` durante la ejecución de los niveles.

En la sección 3.2, vimos las funciones que el juego Geometry Friends ofrece a los agentes en forma de interfaz para interaccionar con él, como, por ejemplo, los métodos *Update* o *GetAction*. En ese momento, obviamos deliberadamente el método *GetDebugInformation*, que comunica al juego la información que debe imprimir en el depurador visual por pantalla. Lo que devuelve periódicamente este método es una lista de objetos de tipo *DebugInformation* que pueden ser tanto un nuevo lienzo sobre el que pintar como un círculo, un rectángulo, una línea o incluso texto. Existen funciones para crear cada uno de estos objetos, establecer su tamaño y posición dentro del lienzo, su color y, en el caso del texto, el mensaje.

Con estas herramientas, hemos creado la interfaz visual de explicabilidad, pero debemos advertir que el depurador visual que ofrece el juego presenta un problema que limita en gran medida la cantidad de información que se muestra en pantalla. A medida que se añaden más objetos que pintar sobre el lienzo en la lista devuelta por *GetDebugInformation*, el rendimiento de la aplicación se resiente notablemente, los agentes se mueven de forma entrecortada, y su precisión decae. Por tanto, es preciso encontrar cierto equilibrio entre la cantidad de información que se muestra por pantalla y el rendimiento de la aplicación.

El aspecto de la interfaz de explicabilidad es el que se muestra en la figura 6.17. Aunque pueda parecer recargado, advertimos que se puede eliminar cierta información que no se desee mostrar de manera muy sencilla, con tan solo comentar unas líneas de código que están claramente identificadas. A lo largo de la explicación, iremos ocultando o mostrando ciertas partes de la interfaz para focalizarnos en la parte que exponemos.

En primer lugar, podemos observar que la interfaz visual se divide fundamentalmente en tres partes. En la parte izquierda, se muestra la información relativa al círculo (sobre un fondo de color amarillo claro), en la derecha, la información del rectángulo (sobre un fondo de color verde claro), y, dentro del nivel, la información de plataformas, puntos de interés, trayectorias y movimientos del plan, entre otros. Hacemos notar que el sistema proporcionado por el juego para mostrar información

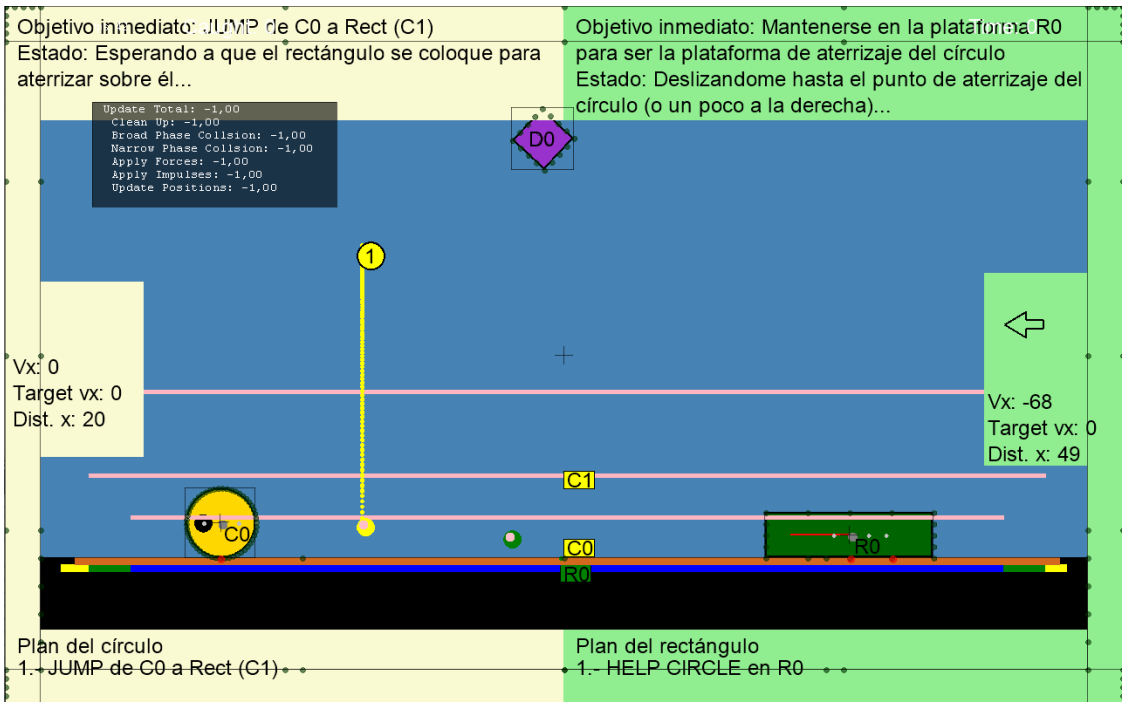


Figura 6.17: Aspecto de la interfaz de explicabilidad.

por el depurador visual tiene ciertas limitaciones, entre las que encontramos que aparezca un cuadro gris en la zona superior izquierda que no se puede ocultar, que se muestren líneas negras que delimitan los obstáculos y que rodean a los agentes y los diamantes, y que el tamaño de la letra del texto mostrado no se pueda modificar. Todo esto hace que, para mostrar la información necesaria para comprender los niveles, hayamos tenido que crear paneles que invaden parte del área del nivel y pueden llegar a ocultar parte de la acción. Lo cierto es que no tenemos ninguna capacidad para mejorar este aspecto visual de la interfaz, que puede resultar incómodo, pero lo importante de este sistema no es tanto que sea más o menos agradable a la vista, sino que sirva para ayudar a interpretar las decisiones que toman los agentes.

Comenzamos explicando la información relativa a los agentes. En los paneles laterales (en la izquierda para el círculo y la derecha para el rectángulo), encontramos la información de más bajo nivel. En la zona superior de dichos paneles laterales, hay una flecha que indica la acción que está tomando el agente. Por ejemplo, para el rectángulo, una flecha a la izquierda se interpreta como que el agente está tomando en ese momento la acción MOVE_LEFT y las flechas hacia arriba, la derecha o hacia abajo están asociadas a las acciones MORPH_UP, MOVE_RIGHT y MORPH_DOWN, respectivamente. Cuando la acción seleccionada es NO_ACTION, ese espacio aparecerá sin ninguna flecha, como sucede con el círculo en la figura 6.17. El círculo solamente cuenta con las flechas hacia arriba y hacia los lados, que se interpretan como que el círculo está seleccionando la acción JUMP o ROLL_LEFT/ROLL_RIGHT, según corresponda. En esos mismos paneles, un poco más abajo, aparecen tres datos etiquetados como Vx, Target vx y Dist. x. Representan, respectivamente, la velocidad horizontal actual (hacia la derecha positiva y hacia la izquierda negativa), la

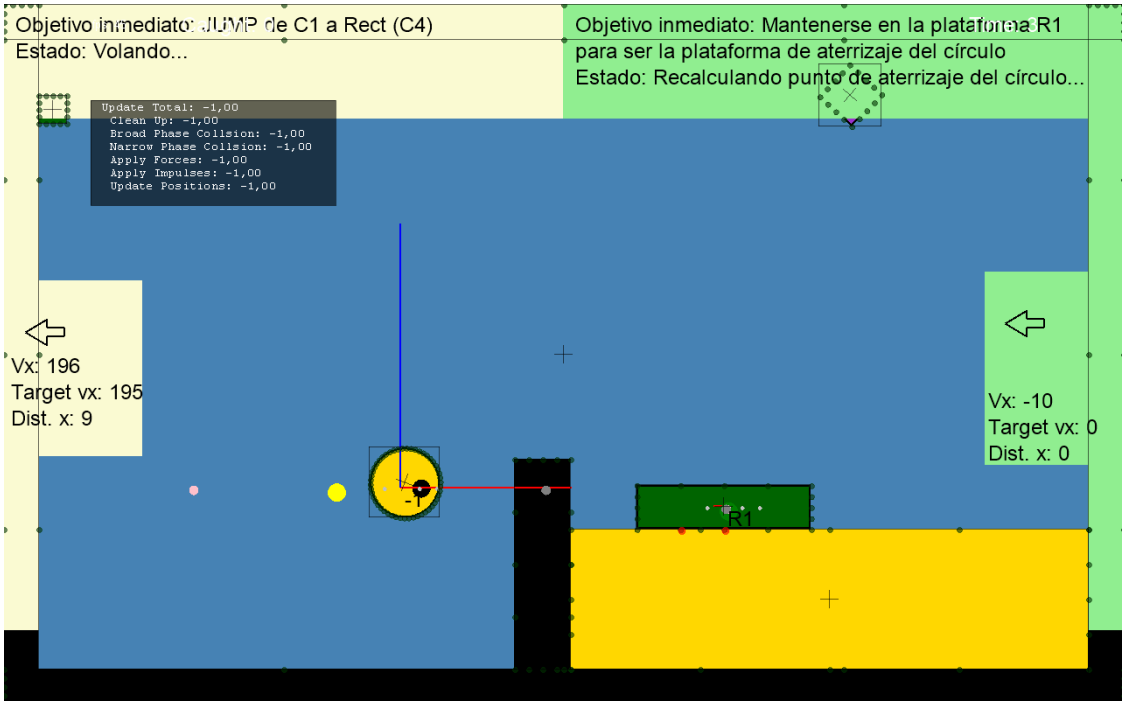


Figura 6.18: Ejemplo de velocidades actual y objetivo en la interfaz de explicabilidad.

velocidad objetivo que deben tener en el punto objetivo y la distancia a dicho punto objetivo.

En el ejemplo de la figura 6.18, podemos ver cómo, una vez ha saltado el círculo para aterrizar sobre el rectángulo, su velocidad horizontal actual es prácticamente la misma que la velocidad objetivo en el punto de salto. En este caso, Dist. x representa la distancia de la posición actual del círculo al punto desde donde ha saltado, que es la proyección del punto amarillo a la izquierda del círculo sobre la plataforma inferior. Asimismo, el rectángulo está en el punto objetivo (y por eso su Dist. x vale 0), aunque todavía debe ajustar su velocidad, pues esta es -10 y debe tener velocidad 0. Aunque en la figura 6.18 se aprecian con mayor dificultad, en la figura 6.17 sí que se ven claramente unos puntos verdes y amarillos que son los puntos objetivos de los agentes y de los cuales se encuentran a distancia 49 y 20. Además, representamos gráficamente las velocidades verticales y horizontales de ambos agentes con líneas azules y rojas sobre sus centros. También representamos, como pequeños puntos rosas y grises, los puntos de aceleración y frenado de ambos agentes, definidos en el sistema de reglas para elegir la siguiente acción de la sección 4.3.3.

Pasamos ahora a explicar los paneles inferiores, para lo cual nos ayudaremos de las figuras 6.19 y 6.20. En dichos paneles, aparecen los planes del círculo y del rectángulo, es decir, los intercambios de plataforma que deben hacer. En el caso del rectángulo, debe realizar dos caídas, alcanzar los diamantes de esa plataforma, realizar un MONOSIDEDROP pegado a la pared de la izquierda, en el que alcanza un diamante, y una última caída para capturar los dos últimos diamantes. Por su

parte, el círculo, debe esperar a que el rectángulo realice la primera caída[1], para después realizar dos caídas, alcanzar los diamantes de esa plataforma y caer una vez más por la derecha para alcanzar otro diamante.

En el plan, también se indica las plataformas origen y destino de los movimientos. Las plataformas del círculo comienzan por C y las del rectángulo por R y la forma de numerarlas, tanto para el círculo como para el rectángulo, es la siguiente: en primer lugar, se numeran las plataformas reales, comenzando arriba a la izquierda, desde el cero, y continuando de izquierda a derecha y de arriba hacia abajo, y después se realiza el mismo procedimiento con las ficticias. Hacemos notar que la numeración es aquella asociada a las plataformas simplificadas, tanto en el caso del círculo como del rectángulo. En la figura 6.20, aparece tanto la leyenda de las plataformas como su representación, que, como se puede apreciar, ofrece más información a costa de sobrecargar la interfaz. En ella, podemos ver las plataformas tanto del círculo como del rectángulo, sus etiquetas, y hasta dónde se extienden. Las plataformas del círculo tienen etiquetas rectangulares de color amarillo que comienzan por C, y las del rectángulo tienen etiquetas rectangulares de color verde que comienzan por R. Asimismo, las plataformas reales del círculo son de color chocolate, y las ficticias, rosa claro. Las del rectángulo están representadas un píxel por debajo de las del círculo, y la leyenda de colores es la misma que se explicó en la sección 5.2.1, que dependía de las formas admisibles del rectángulo en dicha plataforma. Para completar la información sobre las plataformas, nuestra interfaz de explicabilidad también es capaz de indicar las plataformas en las que se encuentran los agentes. La etiqueta de las plataformas se sitúa en el centro de los mismos, como se puede ver en la figura 6.17.

Volviendo a la figura 6.19, donde la vista está menos cargada, podemos observar cómo se dibujan sobre el nivel los movimientos que permiten cambiar de plataforma. Las trayectorias de los movimientos del círculo están coloreadas de amarillos, y sobre ellas aparece una etiqueta circular amarilla del paso del plan que les corresponde. Para el rectángulo, todo es análogo, y las trayectorias y etiquetas son verdes.

Por último, presentamos los paneles superiores, que son los que más información pueden aportar a un usuario esporádico de la aplicación. Tanto para el círculo como para el rectángulo, se muestran dos mensajes: su objetivo inmediato y su estado.

El objetivo inmediato puede ser, o bien realizar el primer movimiento del plan (cuando no hay más diamantes en la plataforma actual del agente), o bien alcanzar uno o varios diamantes desde la plataforma actual, sin necesidad de cambiar de plataforma. En el primero de los casos, se indica el tipo de acción que permite cambiar de plataforma o ayudar al otro agente, las plataformas destino y origen, y los diamantes que se capturan en la trayectoria del movimiento, si los hay. En el segundo caso, se omiten las plataformas de origen y destino, por ser ambas la plataforma actual del agente, pero se añade la información sobre si el diamante se alcanza de

[1]El plan del círculo ha generado un movimiento de espera porque recordemos que los planes del círculo y el rectángulo deben tener el mismo número de pasos.

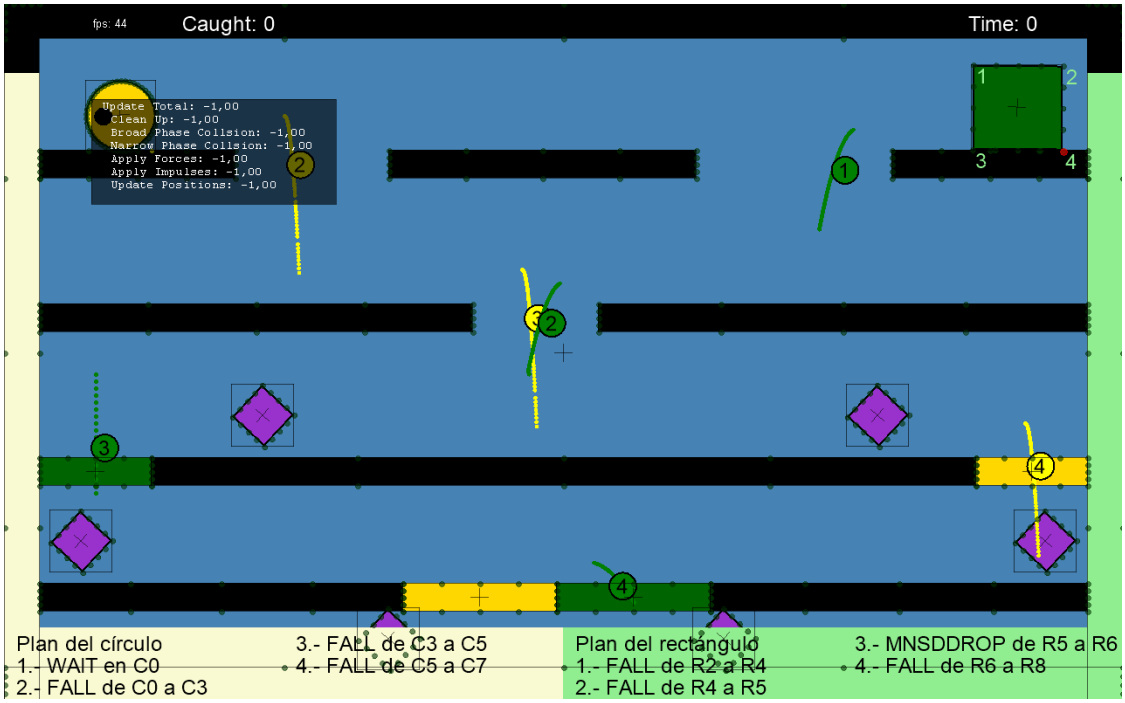


Figura 6.19: Ejemplo del panel inferior con los planes del círculo y el rectángulo.

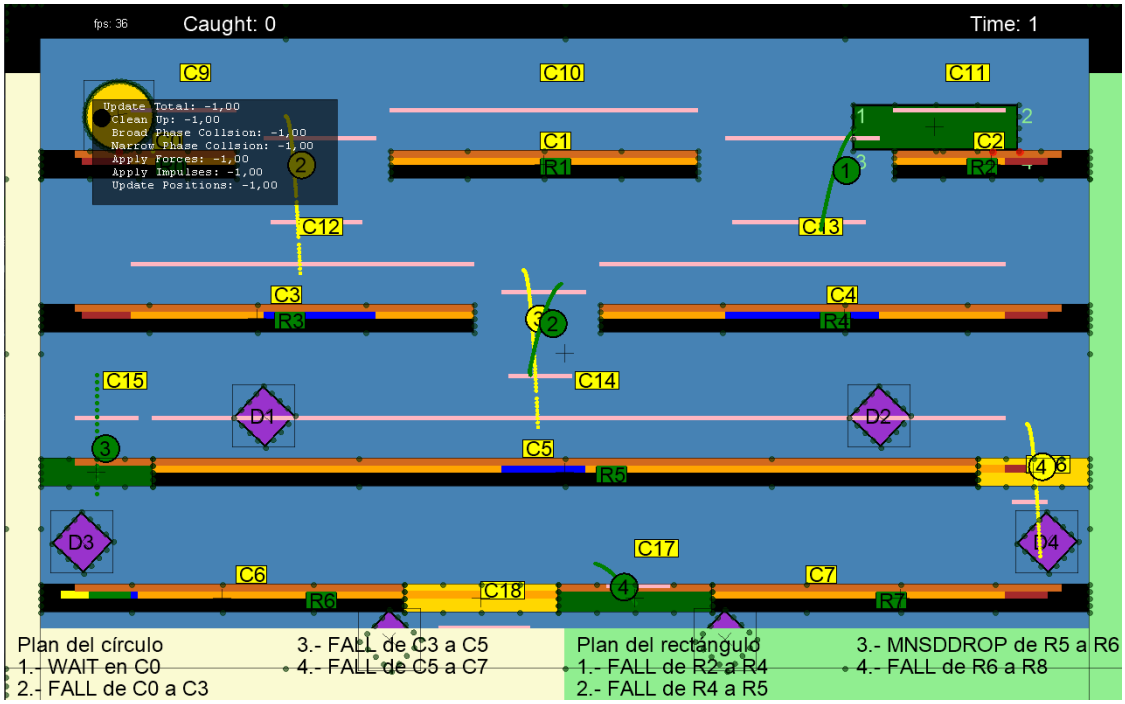


Figura 6.20: Ejemplo del panel inferior con los planes y mostrando la leyenda de las plataformas.

manera individual o con colaboración. Por tanto, como objetivo inmediato, podremos encontrar mensajes como “Coger diamante D2, individualmente mediante un NOMOVE”, “MONOSIDEDROP de R5 a R6 que alcanza D5” o “JUMP de C0 a Rect (C2)”, que indica que la plataforma de aterrizaje es el rectángulo (la platafor-

ma del círculo ficticia etiquetada como C2). El rectángulo también puede tener otros objetivos inmediatos colaborativos como "Mantenerse en la plataforma R1 sin estorbar al círculo", "Mantenerse en la plataforma R0 para ser la plataforma de despegue del círculo" o el análogo "Mantenerse en la plataforma R0 para ser la plataforma de aterrizaje del círculo". La numeración de los diamantes es la dada por la representación del juego y se puede mostrar la etiqueta de cada diamante en la interfaz visual como se aprecia en las figuras 6.17 y 6.20. Los objetivos inmediatos son mensajes en lenguaje natural y de muy alto nivel que permiten al usuario identificar las motivaciones de los agentes e interpretar su comportamiento a nivel de planificación.

Justo debajo de la información sobre el objetivo inmediato de los agentes, se encuentra la información sobre el estado de los mismos. Nuevamente, en lenguaje natural se puede leer una explicación sobre el estado en el que se encuentra el agente y se puede interpretar como lo que está "pensando" el agente en este momento. Son fundamentalmente útiles para los procesos de coordinación entre el círculo y el rectángulo, especialmente de cara a la realización de las acciones cooperativas. Algunos mensajes del círculo son "Rodando hacia el punto objetivo con la velocidad apropiada", "Esperando a que el rectángulo se coloque para aterrizar sobre él", "Manteniendo el equilibrio encima del rectángulo", "Volando", "Intercambiando posición con el rectángulo", "Evitando caer por un borde", "Rodando hacia el escalón para hacer un CIRCLETILT", "Preparado para hacer un CIRCLETILT" o "Sistema de recuperación". Sin embargo, advertimos que, durante sus saltos, cuando atraviesa plataformas ficticias, puede llegar a creer que está sobre ellas, iniciando un proceso de replanificación o mostrando estados que no son muy representativos.

Por su parte, el estado del rectángulo puede mostrar mensajes como "Transportando al círculo", "Esperando al aterrizaje del círculo", "Deslizándome hasta el punto de aterrizaje del círculo (o un poco a la izquierda)", "Dejando que el círculo termine de aterrizar", "Intercambiando posición con el círculo", "Volando", "Sistema de recuperación", "Deslizándome hasta la posición objetivo", "Recalculando punto de aterrizaje del círculo", "Esquivando al círculo", "Esperando a que el círculo esté listo para hacer el CIRCLETILT" o "Deslizándome hasta la posición del CIRCLETILT". Todos estos estados del círculo y el rectángulo describen con bastante expresividad las situaciones en las que se encuentran los agentes, qué hacen, y por qué lo hacen, lo que es una gran ayuda para la explicabilidad del sistema.

Para concluir, animamos al lector a que acceda al siguiente enlace para ver el comportamiento del sistema de explicabilidad en funcionamiento mediante un vídeo. Le advertimos que es conveniente que pause el vídeo periódicamente o lo reproduzca a una velocidad más lenta, para que así le dé tiempo a interpretar los mensajes, ya que, si bien hemos intentado que los mensajes sean claros y concisos, los agentes siguen teniendo la misión de completar los niveles en el menor tiempo posible, por lo que los estados, objetivos inmediatos y planes cambian rápidamente.

Capítulo 7

Resultados

En este capítulo presentamos los resultados de las pruebas realizadas a nuestros agentes en comparación a los agentes que participaron en las competiciones de años anteriores. Asimismo, compararemos los resultados de nuestros agentes con el jugador humano medio y estimaremos el margen de mejora de nuestros agentes respecto a un ideal agente óptimo. Las evaluaciones se realizarán celebrando una serie de competiciones para los tres tipos de niveles (CircleTrack, RectangleTrack y CooperationTrack) y que imiten al funcionamiento de aquellas en las que nos vamos a presentar.

7.1. Diseño experimental

Para evaluar el rendimiento de nuestros agentes círculos, rectángulo y cooperativos, vamos a comparar su ejecución con la de los agentes que mejores resultados han obtenido en las competiciones disputadas en años anteriores. Asimismo, compararemos también los resultados de nuestros agentes frente a jugadores humanos y a un agente "óptimo". Para realizar estas pruebas, hemos seleccionado los niveles de tres competiciones de años anteriores por cada modalidad de juego, para los cuales vamos a celebrar competiciones locales que imiten a las oficiales de Geometry Friends. Nos hubiera gustado poder presentar también los resultados de la competición de este año, en la que participaremos. Sin embargo, esta competición se acaba de abrir recientemente y no hemos tenido suficiente margen temporal para preparar la competición local. Además, la competición oficial se celebrará en agosto y no tendremos acceso a los cinco niveles privados ni a los resultados de otros agentes hasta entonces. No obstante, en el capítulo 8 haremos algún comentario sobre los resultados de nuestros agentes en los niveles públicos, que adelantamos que son muy alentadores.

Para las comparaciones con otros agentes, la organización de Geometry Friends

proporciona el código de propuestas de años anteriores y, en cada modalidad de juego, hemos escogido, entre estos, los dos agentes que mejores resultados obtienen. Concretamente, compararemos nuestros agentes círculo con los agentes MARL-GF Agent y AGAgent, nuestro agente rectángulo con NKUST y Sub Goal A Star y nuestra pareja de cooperativos con MARL-GF Agent y NKUST.

Por otro lado, conviene explicar qué entendemos por agente óptimo. Las puntuaciones asociadas a este agente son en realidad las mejores puntuaciones que ha conseguido un jugador experto en cada nivel, tras haber realizado múltiples intentos. Si bien las puntuaciones no son formalmente las óptimas, son una cota inferior suficientemente ajustada como para que resulten relevantes y nos referiremos a ellas como *óptimas* por simplicidad. Usaremos estas puntuaciones óptimas como escala en los gráficos de barras de las siguientes secciones del capítulo, representando para cada nivel el porcentaje de puntuación que ha obtenido cada agente sobre la puntuación óptima. Esta métrica nos permite comparar la actuación de los agentes en distintos niveles que pueden requerir un tiempo diferente para ser completados o que tienen distinto número de diamantes.

En cuanto al sistema de puntuación, recordamos que el concurso oficial de Geometry Friends premia tanto los diamantes recogidos como los niveles completados (así como el tiempo empleado en hacerlo), y es el mismo sistema el que vamos a seguir. Cada agente es ejecutado 10 veces en cada uno de los 10 niveles de la competición. La puntuación final se obtiene sumando la puntuación de cada nivel, que a su vez resulta de realizar la media aritmética de las puntuaciones de las 10 ejecuciones en dicho nivel. La puntuación de cada ejecución se calcula con la fórmula

$$\text{SCORE} = V_{\text{completed}} \times \frac{maxTime - agentTime}{maxTime} + (V_{\text{collected}} \times N_{\text{collected}})$$

donde $V_{\text{completed}}$ es la puntuación recibida por completar el nivel, $maxTime$ es el límite de tiempo del nivel, $agentTime$ es el tiempo invertido por el agente, $V_{\text{collected}}$ es la puntuación por conseguir un diamante y $N_{\text{collected}}$ es el número de diamantes obtenidos. Los valores de $V_{\text{completed}}$ y $V_{\text{collected}}$ varían de un año a otro y $maxTime$ en función del nivel.

Para facilitar que los participantes en la competición oficial realicen este tipo de comparaciones, el juego presenta una funcionalidad de *batch simulator*. Según se muestra en la figura 7.1, el cuadro de dialogo permite seleccionar los agentes que se van a ejecutar y los niveles donde van a participar. De igual manera, se permiten fijar parámetros de la simulación como el número de ejecuciones en cada nivel, la velocidad de ejecución, si se muestra visualmente a los agentes mientras intentan superar el nivel o la ejecución se produce en segundo plano y otras opciones. También ofrece la posibilidad de guardar los resultados en un fichero .csv y fijar los parámetros $V_{\text{completed}}$ y $V_{\text{collected}}$, propios de cada competición. De entre estas opciones, únicamente destacamos que optamos por realizar las ejecuciones en tiempo real, ya que al aumentar la velocidad de ejecución la mayoría de los agentes empeoraban sus resultados y estos quedarían contaminados.

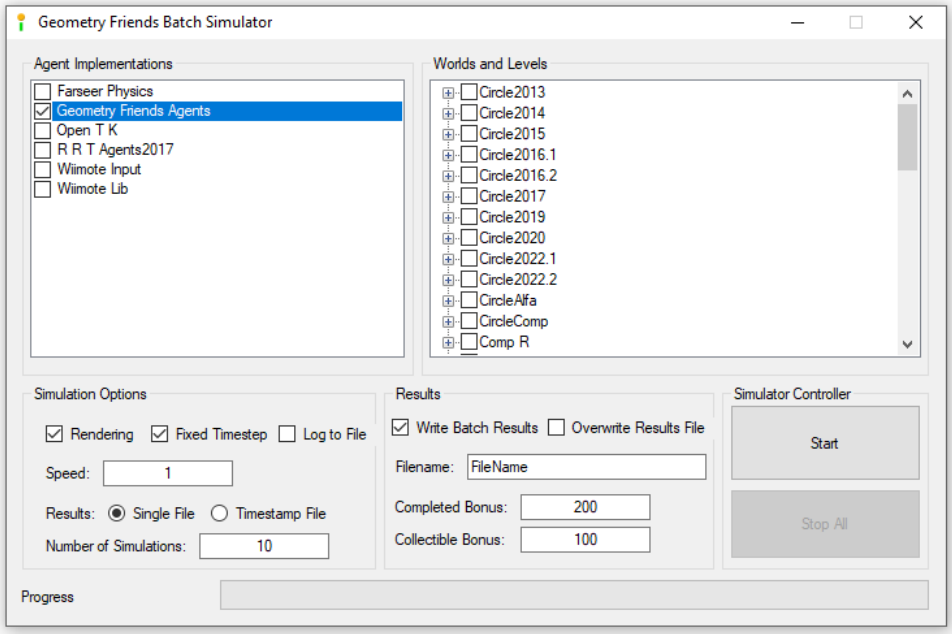


Figura 7.1: Ventana de diálogo del *batch simulator*.

Por otra parte, es conveniente explicar cómo vamos a evaluar a los jugadores humanos, cómo ha sido el proceso para obtener los datos y a qué llamamos "humano medio". Aunque toda la información se encuentra en el Apéndice B, resaltamos que las sesiones solo se organizaron para los modos de juego individuales del círculo y del rectángulo, por el reto logístico que suponía el modo cooperativo. Las sesiones de evaluación para cada agente individual duraron 40 minutos por participante aproximadamente. En los primeros 20 minutos, los participantes jugaron en niveles de prueba a modo de entrenamiento, para ajustarse a las físicas del juego, a los controles y a las acciones de los personajes. En los 20 restantes, tuvo lugar la evaluación, en la que los participantes jugaron en los niveles de las competiciones.

La principal diferencia en cuanto a las ejecuciones de los jugadores humanos es que estos tuvieron únicamente un intento para completar cada nivel. Tomamos esta decisión en el diseño de las pruebas en contraposición a que se hiciera la media de 10 ejecuciones, como en el caso de los agentes, por tres motivos. Por un lado, los jugadores humanos podrían aprender de una ejecución a otra, mientras que en las competiciones oficiales del juego Geometry Friends, una vez se realiza la entrega, el agente no puede modificar su comportamiento de una ejecución a otra ni puede aprender durante el proceso de evaluación. Por otro lado, el motivo por el que se realizan 10 ejecuciones para los agentes es para amortiguar los efectos de la aleatoriedad de algunas implementaciones y creemos que el azar se verá amortiguado en el caso de los humanos al considerar múltiples participantes. El último motivo es logístico: la evaluación dura 20 minutos y si se deben repetir todos los niveles 10 veces sería imposible encontrar voluntarios para la evaluación.

Con todo, el "jugador humano medio" no es más que la media de las puntuaciones de todos los participantes humanos que hemos evaluado. El número de participantes ha sido 16, fundamentalmente compañeros de clase, con edades comprendidas entre los 20 y los 25 años, con formación universitaria y habituados al uso de ordenadores. Hemos optado, en lugar de realizar un estudio amplio, hacer uno pequeño, en el que nosotros estuviéramos presencialmente guiando a los participantes, pues queríamos asegurarnos de que estos seguían las normas correctamente y que tenían la capacidad de preguntarnos cualquier duda sobre el desarrollo de las pruebas, el juego o los controles.

Como ya se advirtió, el juego presenta algunos *bugs* que afectan especialmente al rectángulo y aparecen cuando este crece o decrece y no tiene suficiente espacio para hacerlo o cuando cae de una plataforma elevada violentamente y no puede volver a cambiar de forma. Al organizar las competiciones locales, se nos planteó la duda de cómo tratar las ejecuciones en las cuales sucede alguno de estos *bugs*. Decidimos actuar de igual manera que los organizadores de las competiciones oficiales: los *bugs* forman parte del juego y, cuando uno sucede, la ejecución no queda invalidada o impugnada, sino que la puntuación obtenida es aquella que se obtuvo cuando sucedió el *bug*. De esta manera, las puntuaciones del rectángulo serán más bajas de lo que los agentes sean capaces de hacer en la realidad, pero todos los agentes (y humanos) jugarán con las mismas reglas y deberán evitar, en la medida de lo posible, tomar acciones que les conduzcan a estos estados incontrolables.

Esta decisión hace que la puntuación óptima en el caso del rectángulo, y en menor medida en los niveles cooperativos, esté considerablemente más separada de los resultados de los agentes y de los jugadores humanos que en la competición del círculo. Asimismo, la puntuación óptima del rectángulo utiliza en ocasiones los *bugs* del juego a su favor para disminuir el tiempo empleado en algunos niveles. Se pueden encontrar vídeos que demuestran que se pueden utilizar *bugs* para obtener la mejor puntuación en el siguiente enlace y en este segundo enlace. Por tanto, al analizar las puntuaciones óptimas, habrá que tener en mente estos detalles.

Para cada uno de nuestros agentes, los círculos, el rectángulo y la pareja colaborativa, encontraremos vídeos de sus actuaciones en todos los niveles de las competiciones. Con estas grabaciones pretendemos dar una idea del comportamiento cualitativo de nuestros agentes, además de las comparaciones cuantitativas que haremos posteriormente. Hacemos notar que son vídeos de unos 20 minutos de duración luego recomendamos, si se desea, mirar únicamente algunos pocos niveles o aumentar la velocidad de reproducción de los mismos para tener una idea general de la actuación de los agentes.

Una vez explicado el marco sobre el que se van a regir nuestras competiciones, en las siguientes secciones ofreceremos los resultados de las mismas y que nos servirán para comparar los círculos, los rectángulos y las parejas de agentes cooperativos.

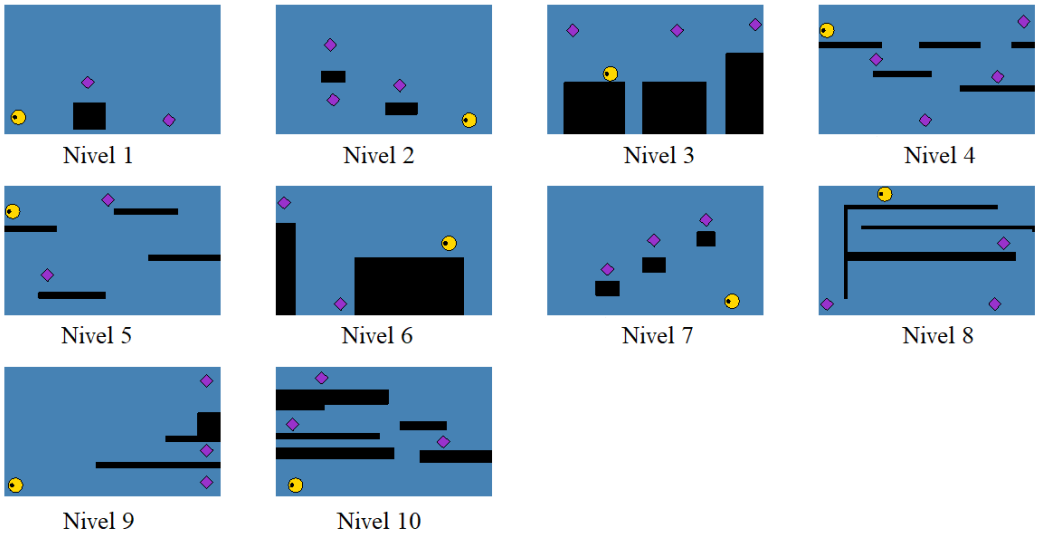


Figura 7.2: Niveles de la competición del año 2014.

7.2. Resultados de los agentes círculo

En esta sección presentamos los resultados en las competiciones de los años 2014, 2017 y 2022 de nuestros agentes círculos, el UCM Physics y el UCM QLearning, que toman sus nombres del tipo de actuador que tienen implementado según se vio en la sección 4.3.3. Se puede encontrar un vídeo de la actuación del agente UCM Physics en este enlace y del UCM QLearning en este enlace.

7.2.1. Competición Círculo 2014

Los primeros niveles que vamos a analizar (ver figura 7.2) son los de la competición celebrada en 2014. Ese año, las puntuaciones por completar un nivel y capturar un diamante fueron $V_{\text{completed}} = 1000$ y $V_{\text{collected}} = 100$, y el ganador fue el agente CIBot con una puntuación final de 4337 puntos. En la tabla 7.1 mostramos la puntuación final obtenida por los distintos agentes evaluados, ordenados de mayor a menor, y en la sección A.1.1 del Apéndice A, los resultados de cada agente desglosados por niveles (tablas A.1 a A.6). Con el fin de agrupar toda la información de las tablas anteriores, en la figura 7.3 aparece una gráfica donde se puede ver de manera más ilustrativa el porcentaje de puntuación sobre la puntuación óptima obtenido por cada agente en cada nivel.

En primer lugar, observamos que nuestro agente UCM Physics es el que mejores resultados obtiene. Consigue superar todos los niveles en todas las ocasiones salvo el nivel 7, en el que solamente logra completar 6 de las 10 ejecuciones, quedándose en las cuatro que falla a falta de un diamante. Si analizamos el nivel, vemos que las

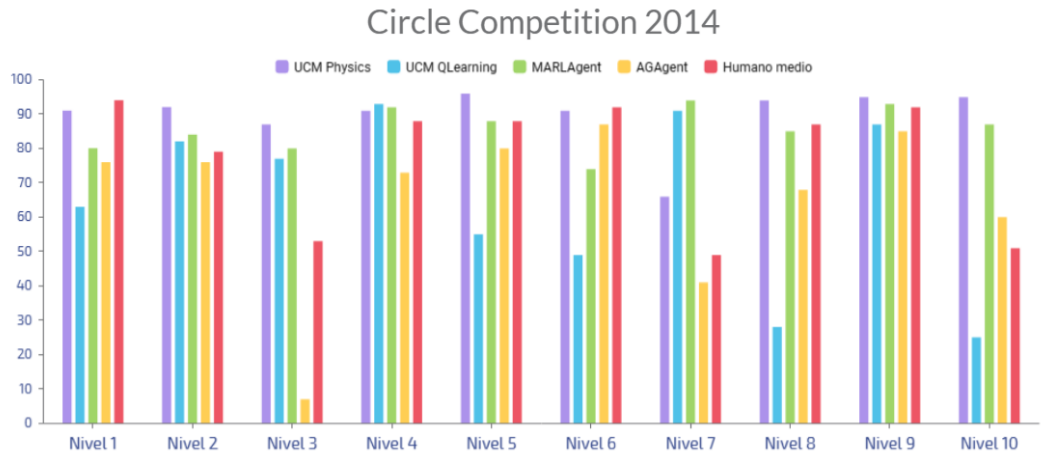


Figura 7.3: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2014.

Agente	Puntuación total obtenida
Óptimo	10451
UCM Physics	9364
MARL-GF Agent	8978
Humano medio	7987
UCM QLearning	7001
AGAgent	6743
Ganador 2014 (CIBot)	4337

Tabla 7.1: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2014.

plataformas son particularmente estrechas, causa del bajo rendimiento de nuestro agente.

Además, el tiempo límite es a veces insuficiente para que nuestro agente complete el nivel, pero con tiempo ilimitado lo acabaría consiguiendo. En el mismo nivel, podemos observar que los jugadores humanos y el agente AGAgent también encuentran dificultades, porque, como hemos comentado, el nivel requiere de una gran precisión en las acciones, al ser las plataformas tan estrechas, y una gran efectividad, por la falta de tiempo.

En el resto de niveles, nuestro agente UCM Physics se desenvuelve con solvencia, obteniendo las mejores puntuaciones en 6 de ellos y manteniéndose a una distancia cercana al óptimo (siempre con más del 85 % de la puntuación óptima). Si tenemos en cuenta los resultados de los participantes humanos tomados individualmente, solamente uno de ellos obtuvo una puntuación mayor a la del agente UCM Physics, concretamente 9483 puntos, no llegando el resto de los participantes a los 9000 puntos.

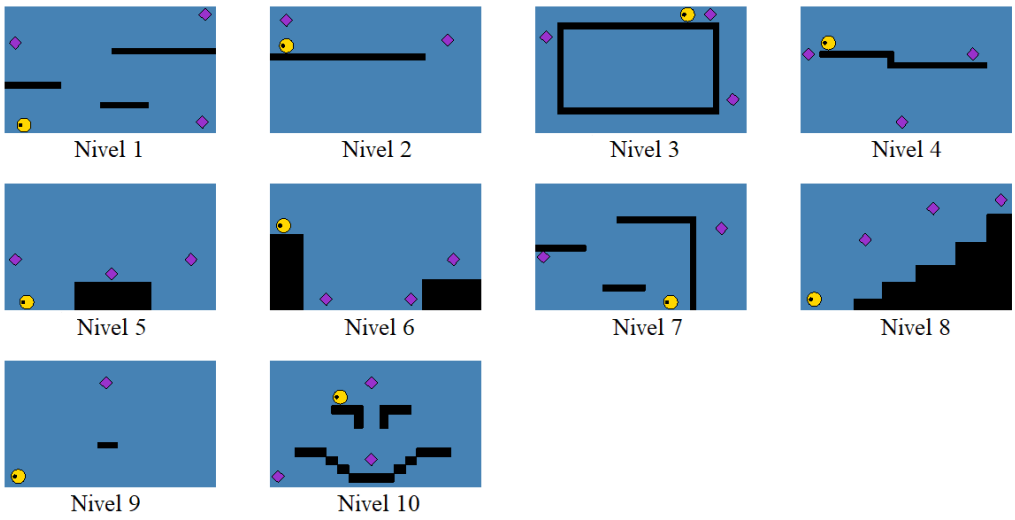


Figura 7.4: Niveles de la competición del año 2017.

Por su parte, el agente UCM QLearning tiene un peor comportamiento, pero aun así es capaz de aventajar a los agentes AGAgent y CIBot. Los niveles en los que más sufre el agente UCM QLearning son el 5, el 6, el 8 y el 10.

Las mayores limitaciones de este agente son que el entrenamiento con las velocidades más altas resultó infructuoso, que el agente ha aprendido a utilizar más espacio del que realmente necesita para alcanzar la velocidad objetivo y que la discretización de velocidades, menos fina que la del agente UCM Physics, le perjudica. A pesar de sus limitaciones, queremos destacar que consigue obtener la mejor puntuación en el nivel 4 y resuelve con solvencia el nivel 7, quebradero de cabeza de humanos y de muchos agentes.

7.2.2. Competición Círculo 2017

Pasamos ahora a la competición de 2017, compuesta por los niveles de la figura 7.4. En aquella edición, el agente ganador fue KITAgent con una puntuación de 4203. Cabe destacar que en esa ocasión las puntuaciones por completar un nivel y alcanzar un diamante cambiaron y pasaron a ser $V_{\text{completed}} = 200$ y $V_{\text{collected}} = 100$, por lo que no tiene sentido comparar las puntuaciones de la competición del año 2014 con las del año 2017. En la sección A.1.2, se muestran los resultados obtenidos por los agentes (tablas A.7 hasta A.12). Además, añadimos la figura 7.5, que representa el porcentaje de la puntuación óptima alcanzado por cada uno de los agentes y que resume los resultados de la competición.

Como se puede apreciar, esta competición presenta unas puntuaciones finales mucho más igualadas, por una parte debido al sistema de puntuación y por otra a que los niveles son más sencillos que en la competición anterior. Esto último se distingue

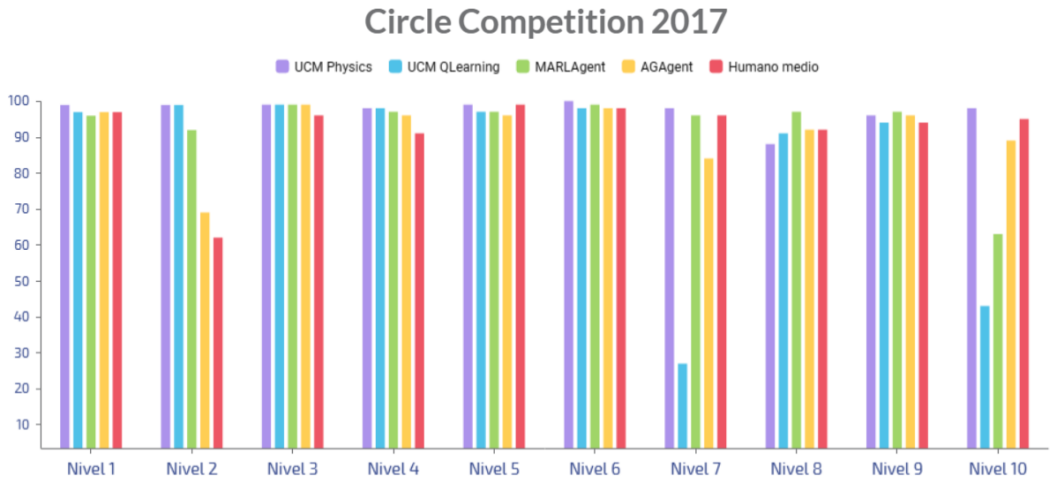


Figura 7.5: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2017.

Agente	Puntuación total obtenida
Óptimo	4425
UCM Physics	4309
Ganador 2017 (KITAgent)	4203
MARL-GF Agent	4119
Humano medio	4095
AGAgent	4075
UCM QLearning	3765

Tabla 7.2: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2017.

claramente comparando las figuras 7.3 y 7.5 y observando que los resultados de la competición de 2017 son mucho más cercanos al óptimo que los del año 2014, para todos los agentes. Los niveles que resultan más determinantes son el segundo, el séptimo y el décimo. Nuevamente, nuestro agente UCM Physics es el que mejores resultados obtiene, siendo el mejor en 7 de los 10 niveles y en todos los que resultan determinantes, que son presumiblemente los más difíciles. Consigue resolver todos los niveles en todas las ejecuciones y además emplea en general menos tiempo que los demás agentes. Además, únicamente 4 de los participantes humanos superaron individualmente al agente UCM Physics, siendo la mejor puntuación obtenida 4350, ligeramente superior a la puntuación de nuestro agente. La distancia al agente óptimo sigue siendo muy reducida ya que obtenemos un $97,4\,\%$ de la puntuación final del agente óptimo.

La bajada de rendimiento del agente UCM QLearning, que es el que peores resultados obtiene, se centra únicamente en los niveles 7 y 10. En el nivel 7, el salto que le permitiría llegar a la plataforma superior es con velocidad máxima y no se genera (nótese que hay un rebote en el techo y pierde velocidad) y en el último nivel sucede

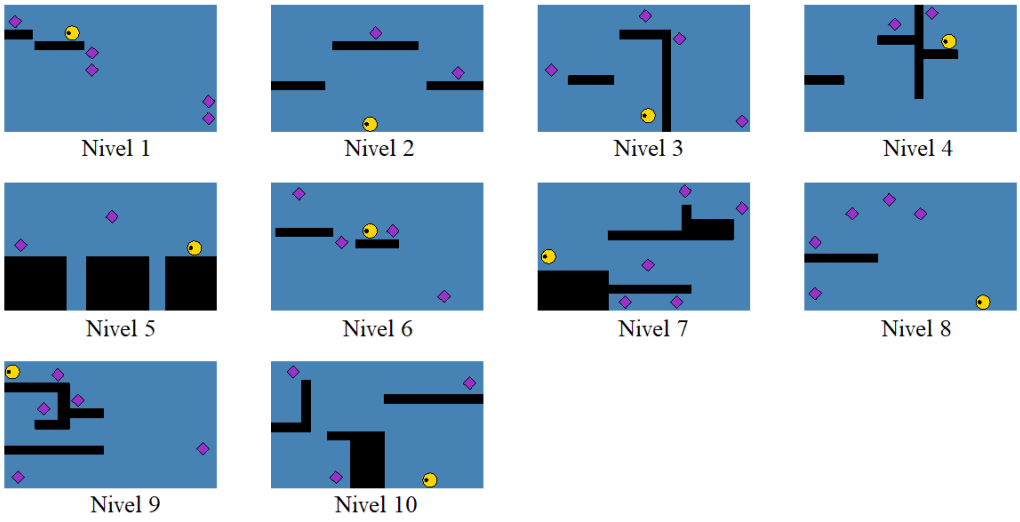


Figura 7.6: Niveles de la competición GF-CoG 2022.

algo similar. Cuando el agente está en la plataforma del diamante central inferior, debe alcanzar la velocidad máxima y saltar para salir de ella. Si no fuera por estos dos niveles, sus resultados serían cercanos a los de los mejores agentes. En cuanto al nivel 2, en el que el rendimiento de los humanos decae considerablemente y provoca que sea superado por los agentes KITAgent y MARL-GF Agent, se da la circunstancia de que alcanzar el diamante de la derecha no es tan sencillo como pudiera parecer. La estrategia que siguieron algunos humanos fue caer, pero sus cálculos no fueron acertados porque, aunque se parta con velocidad horizontal máxima, es imposible alcanzar el diamante cayendo. La forma correcta de hacerlo es saltar, rebotando en el techo. Esto último también supuso problemas de cálculo para los participantes humanos ya que muchos de ellos, aun identificando que lo apropiado era saltar, erraron a la hora de calcular el rebote en el techo y cayeron sin alcanzar el diamante, llegando a un estado donde era imposible completar el nivel.

7.2.3. Competición Círculo 2022

Por último, estudiaremos el comportamiento de los agentes en los niveles de la competición GF-CoG 2022, que se presentan en la figura 7.6. El agente ganador en esa competición fue KITAgent, con 16628 puntos. Nuevamente advertimos que los valores de $V_{\text{completed}}$ y $V_{\text{collected}}$ varían respecto a años anteriores, siendo en este caso $V_{\text{completed}} = 1000$ y $V_{\text{collected}} = 300$. Los resultados obtenidos por los agentes, desglosados por niveles, se pueden encontrar en la sección A.1.3 (tablas A.13 a A.18). Además, en la figura 7.7 y en la tabla 7.3 se puede ver de manera más resumida los resultados de la competición, que analizamos a continuación.

En esta competición, volvemos a apreciar diferencias significativas en el rendimien-

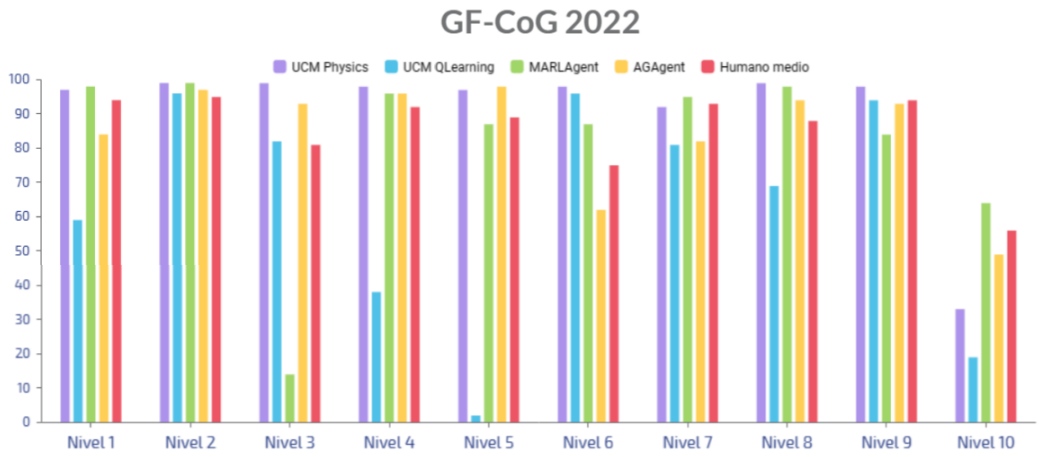


Figura 7.7: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición GF-CoG 2022.

Agente	Puntuación total obtenida
Óptimo	19942
UCM Physics	18266
Humano medio	17122
AGAgent	16835
Ganador GF-CoG 2022 (KITAgent)	16628
MARL-GF Agent	16390
UCM QLearning	13266

Tabla 7.3: Comparación de la puntuación obtenida por diferentes agentes en la competición GF-CoG 2022.

to de los agentes. Nuevamente, el agente UCM Physics se corona como vencedor, aventajando en casi 1500 puntos al siguiente agente. Nuestro agente es capaz de completar con éxito los 9 primeros niveles y es solamente en el último donde tiene dificultades. El plan en el último nivel es alcanzar en primer lugar el diamante de la plataforma inferior izquierda y, si bien se puede salir de esa plataforma y de hecho lo logra en algunas de las 10 ejecuciones, es muy difícil conseguirlo. Si el plan alcanzara primero los diamantes de la parte superior del nivel y finalizara en el de la zona inferior izquierda, el agente UCM Physics conseguiría completar el nivel en todas las ocasiones. Sin embargo, estos planes alternativos tienen el mismo número de cambios de plataforma que el plan que se toma, luego para el algoritmo de búsqueda son igual de buenos y es solamente el orden en el que se expanden los nodos lo que hace que se tome uno u otro. Una alternativa para solventar este problema sería utilizar otro algoritmo de búsqueda como A*, donde la heurística utilizada tuviera en consideración la dificultad de cada movimiento, el tiempo empleado u otras variables. Con todo, nuestro agente domina la competición, superando holgadamente a todos los agentes y al jugador humano medio. Tan solo tres participantes humanos lo superan, siendo 18417 la puntuación del mejor de ellos. Esta diferencia de 151 puntos con los resultados del agente UCM Physics, teniendo en cuenta el sistema de

puntuación, es insignificante.

En cambio, las limitaciones de nuestro agente UCM QLearning quedan expuestas en esta competición y sus resultados se resienten notablemente. El impedimento para aprender a llegar una determinada posición con velocidad máxima y otros problemas derivados del aprendizaje arrojan unos resultados que destacan que este agente no ha conseguido estar a la altura de las expectativas.

7.2.4. Comentarios sobre los resultados obtenidos

En vista de los resultados anteriormente expuestos, podemos afirmar que nuestro agente UCM Physics es el mejor agente del círculo desarrollado hasta el momento. En las competiciones que hemos evaluado, consigue superar holgadamente a los agentes AGAgent, MARL-GF Agent y los respectivos ganadores de las competiciones. Además, lo logra obteniendo los mejores resultados en la inmensa mayoría de los niveles, empleando menos tiempo y completando la práctica totalidad los niveles. Asimismo, su rendimiento respecto a los jugadores humanos también es muy satisfactorio. Hemos conseguido que el agente UCM Physics supere al humano medio (que obtiene una media del $84,94\,\%$ de la puntuación de un agente óptimo) en todas las competiciones, hito que no habían logrado los agentes AGAgent y MARL-GF Agent hasta el momento. Estos tenían resultados muy similares al humano medio, sobre todo el agente MARL-GF Agent. Por si fuera poco, el agente UCM Physics tiene resultados comparables con los mejores humanos evaluados y obtiene de media el $92,86\,\%$ de la puntuación de un agente óptimo cuando los agentes AGAgent y MARL-GF Agent obtienen un $80,34\,\%$ y un $87,06\,\%$, respectivamente.

Sin embargo, no podemos estar completamente satisfechos con el agente UCM QLearning, ya que se ha demostrado que todavía está un poco por debajo del rendimiento de los mejores agentes del estado del arte, teniendo una puntuación media del $72,86\,\%$ del óptimo. No obstante, en general podemos calificar su comportamiento como bueno, ya que tiene únicamente un punto débil que está claramente identificado.

7.3. Resultados del agente rectángulo

Las competiciones que celebramos para evaluar nuestro agente rectángulo UCM Physics toman como niveles los de las competiciones de los años 2014, 2016 y 2022. Como en el caso del círculo, podemos encontrar un vídeo con la actuación del rectángulo UCM Physics en las tres competiciones en el siguiente enlace. En las marcas temporales 4:00, 7:35, 8:35 y 8:55 se pueden visualizar algunos de los *bugs* del juego que no permiten al rectángulo cambiar de forma o que lo mantienen atrapado entre dos plataformas.

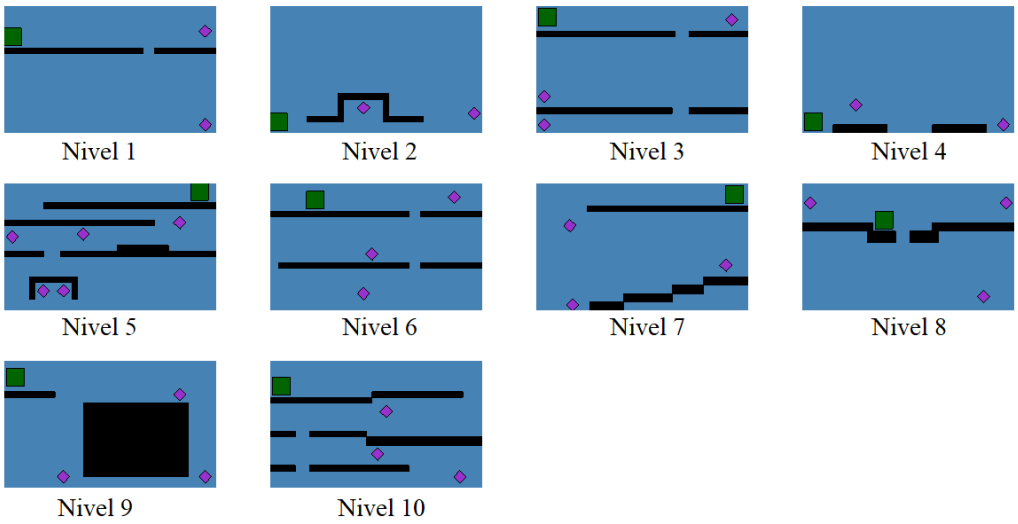


Figura 7.8: Niveles de la competición del año 2014.

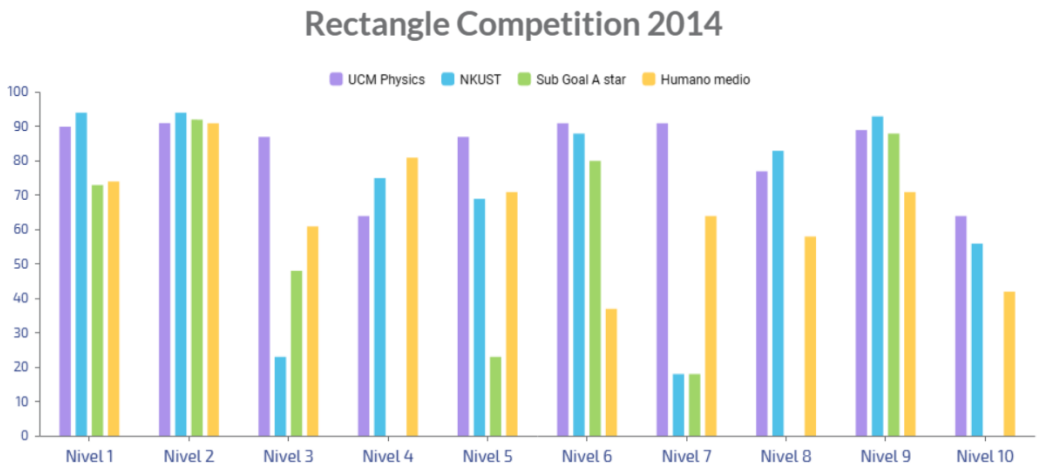


Figura 7.9: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2014.

7.3.1. Competición Rectángulo 2014

En primer lugar, presentamos la competición local celebrada con los niveles de la competición oficial del año 2014, que se pueden encontrar en la figura 7.8. Ese año, las puntuaciones por completar un nivel y capturar un diamante fueron $V_{\text{completed}} = 1000$ y $V_{\text{collected}} = 100$, y el ganador fue el agente CIBot, con una puntuación final de 6466 puntos. En la tabla 7.4, mostramos la puntuación final obtenida por los distintos agentes rectángulo evaluados, y, en la sección A.2.1 del apéndice, los resultados de cada agente desglosados por niveles. Se puede encontrar dicha información en las tablas desde la A.19 a la A.23. Para que nos sea más cómodo analizar toda esta información, en la figura 7.9 aparece una gráfica donde se puede ver de manera más ilustrativa los resultados de la competición.

Agente	Puntuación total obtenida
Óptimo	10462
UCM Physics	8733
NKUST	7157
Humano medio	6842
Ganador 2014 (CIBot)	6466
Sub Goal A Star	4365

Tabla 7.4: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2014.

Podemos observar que nuestro agente UCM Physics obtiene los mejores resultados en esta competición. Supera todos los niveles en las 10 ejecuciones, salvo el último, en el cual hay 3 ejecuciones en las que no captura un diamante. Esto se debe a que el último movimiento de tipo DROP debe ir acompañado inmediatamente de una acción de MORPH_DOWN para poder alcanzar el diamante que está en la zona inferior central del nivel. Sin embargo, hay ocasiones en las que no le da tiempo a alcanzar la forma horizontal y cae a la plataforma del suelo. Aun así, es el que mejores resultados obtiene en ese nivel si lo comparamos con otros agentes, y solo la mitad de humanos logró todos los diamantes en dicho nivel. Con todo, nuestro agente es el que mejores puntuaciones obtiene en 5 de los 10 niveles y aventaja al siguiente agente en más de 1500 puntos. Además, no solo supera al jugador humano medio, sino que supera a todos los jugadores humanos que participaron (los únicos jugadores humanos que superaron los 8000 puntos obtuvieron 8456 y 8011 puntos). En comparación con el agente óptimo, nuestro agente logra el $83,4\,\%$ de su puntuación final, menor que los resultados que habíamos obtenido para el agente círculo, pero aventajando en un $15\,\%$ al siguiente agente. Los jugadores humanos por su parte, quedan en multitud de ocasiones atascados en los niveles en los que hay que realizar un movimiento de los que hemos llamado de tipo DROP (en los que hay que dejarse caer por el hueco entre dos plataformas), como son los niveles 1, 3, 6, 8 y 10.

7.3.2. Competición Rectángulo 2016

La segunda competición local que organizamos fue aquella con los niveles de la figura 7.10, los de la competición oficial del rectángulo del año 2016. En aquella edición el agente ganador fue RRT Agent, con una puntuación de 892. Cabe destacar que en esa ocasión las puntuaciones por completar un nivel y alcanzar un diamante cambiaron y pasaron a ser $V_{\text{completed}} = 200$ y $V_{\text{collected}} = 100$. En la tabla 7.5, se muestran los resultados obtenidos por los agentes, ordenados de mayor a menor. Se pueden encontrar los resultados desglosados por niveles en la sección A.2.2, entre las tablas A.24 y A.28. Además, añadimos la figura 7.11, que representa la puntuación por nivel de cada uno de los agentes de manera más resumida e ilustrativa y que nos guiará para comentar los resultados obtenidos.

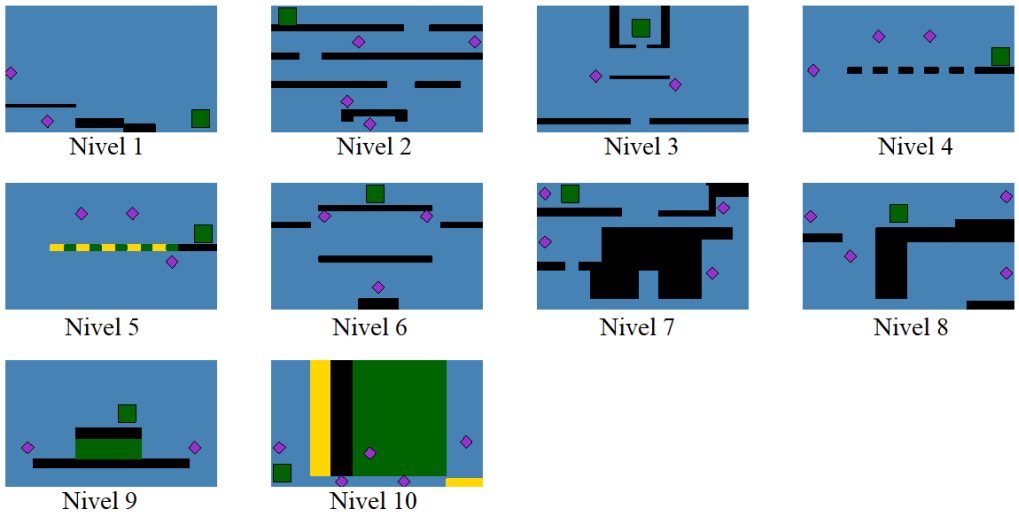


Figura 7.10: Niveles de la competición del año 2016.

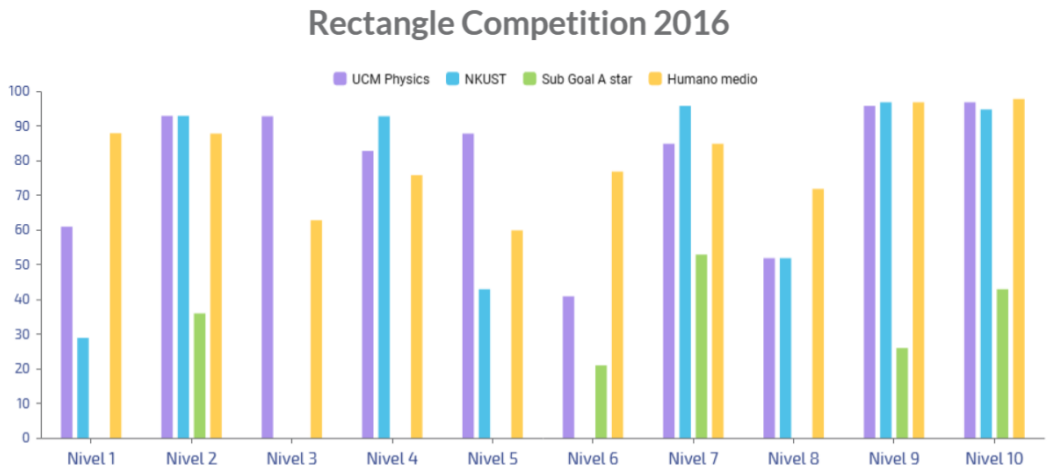


Figura 7.11: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2016.

Agente	Puntuación total obtenida
Óptimo	4933
Humano medio	3999
UCM Physics	3917
NKUST	3137
Sub Goal A Star	1000
Ganador 2016 (RRT Agent)	892

Tabla 7.5: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2016.

En esta ocasión, es el jugador humano medio el que domina, seguido por algo menos

Agente	Puntuación total obtenida
Óptimo	17758
Humano medio	15413
UCM Physics	15306
NKUST	7099
Ganador GF-IJCAI-ECAI 2022 (RRT Agent)	6402
Sub Goal A Star	5512

Tabla 7.6: Comparación de la puntuación obtenida por diferentes agentes en la competición GF-IJCAI-ECAI 2022.

de 100 puntos de nuestro agente UCM Physics. En esta competición, los niveles que nuestro rectángulo no logra completar en todas las ocasiones son el primero, el sexto, el séptimo y el octavo. En dos de ellos, el sexto y el octavo, nuestro agente se queda en todas las ejecuciones exactamente a un diamante de completar el nivel y en ambos casos se debe a uno de los *bugs* del juego. Efectivamente, al caer a la última plataforma, el rectángulo colisiona violentamente con uno de sus vértices y se le bloquea la posibilidad de crecer y decrecer. Como el último movimiento que debe realizar en ambos niveles es un TILT, y el rectángulo es incapaz de hacerse vertical, no consigue completar ninguno de los niveles cuando esto sucede. Hemos podido comprobar que, partiendo de la plataforma inferior, sin haber caído antes y por tanto sin entrar en el estado de bloqueo de las acciones de crecimiento, el rectángulo es capaz de completar los niveles sin ningún problema. En los otros dos niveles, el rectángulo queda en algunas ocasiones atascado en ciertas partes del nivel y no se puede recuperar. No obstante, este problema es muy acotado y también se reproduce en los jugadores humanos. En resumen, el rectángulo UCM Physics es capaz de resolver casi siempre todos los niveles, mientras que, de los demás agentes, NKUST es el único con resultados suficientemente buenos como para ser tomado en consideración y está bastante lejos de los resultados de nuestro agente. A pesar de los *bugs*, en esta competición nos seguimos manteniendo alrededor del 80 % de la puntuación óptima y conservamos el colchón de un 15 % sobre el agente NKUST.

7.3.3. Competición Rectángulo 2022

Finalizamos con una competición que toma sus niveles de la GF-IJCAI-ECAI 2022 - Rectangle Track. Dichos niveles se encuentran en la figura 7.12, y el agente ganador en esa competición fue RRT Agent 2017, con 6402 puntos. Reiteramos nuestra advertencia de que los valores de $V_{\text{completed}}$ y $V_{\text{collected}}$ variaron respecto a años anteriores, siendo en este caso $V_{\text{completed}} = 1000$ y $V_{\text{collected}} = 300$. Los resultados obtenidos por los agentes, desglosados por niveles, se pueden encontrar en el Apéndice A, en la sección A.2.3. Dicha información esta comprendida entre las tablas A.29 y A.33. Además, en la figura 7.13 y en la tabla 7.6 se puede ver de manera más resumida los resultados de la competición que analizamos a continuación.

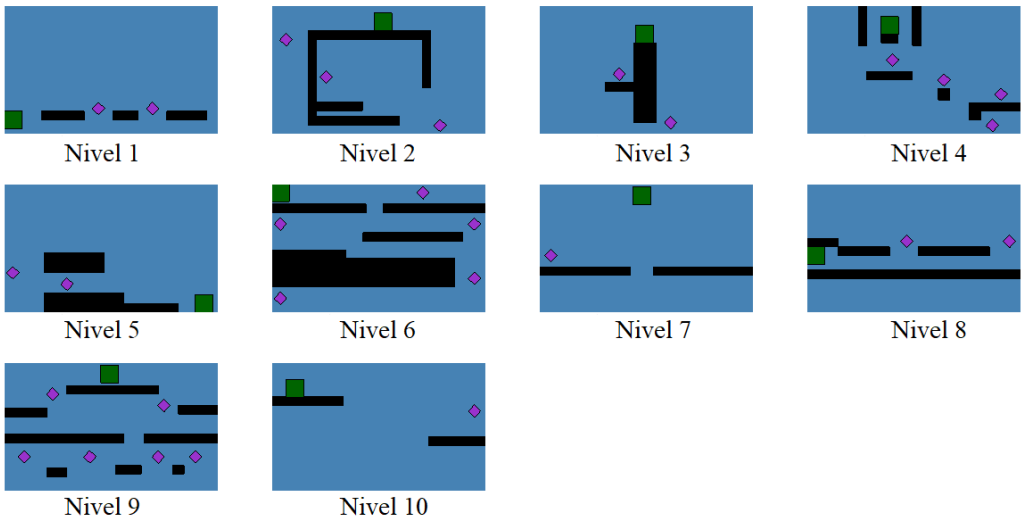


Figura 7.12: Niveles de la competición GF-IJCAI-ECAI 2022.

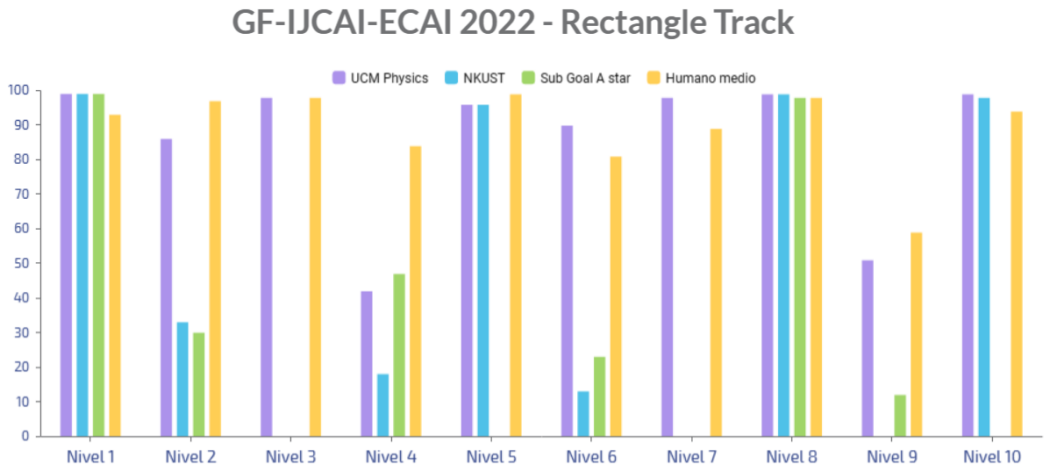


Figura 7.13: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición GF-IJCAI-ECAI 2022.

Como podemos observar, es el humano medio el que vence en esta competición, seguido a poco más de 100 puntos (menos del 1 % de la puntuación del agente óptimo) de nuestro agente UCM Physics. El siguiente agente es NKUST, al cual doblamos en puntuación, superándolo en todos los niveles a excepción del 5, en el que emplea 12 segundos de media para alcanzar los 2 diamantes cuando nuestro agente emplea 12,6. Los resultados en comparación con el resto de agentes son apabullantes, por lo que debemos evaluar a nuestro agente frente al jugador humano medio. Aunque conseguimos superarle en 6 niveles, eso no es suficiente para tener una puntuación global mayor. Nos penalizan el nivel 9 y, especialmente, el nivel 4. Concretamente, en el nivel 4, tras alcanzar los tres primeros diamantes, la simulación de movimientos identifica que una caída en forma de cuadrado es posible para aterrizar en la plataforma inferior. Sin embargo, esa caída nunca se completa ya que el cuadrado

queda atascado entre ambas plataformas. El movimiento adecuado debería ser un MONOSIDEDROP, pero no se detecta dicho patrón porque no hay una pared solida a la izquierda, sino un obstáculo que no continúa hacia abajo. El nivel 9 es un nivel complicado, porque el hueco que permite acceder a los cuatro diamantes de la parte inferior del nivel es de aquellos que no son suficientemente anchos para realizar un movimiento de tipo FALL ni suficientemente estrechos para realizar uno de tipo DROP. Para estos huecos de longitud 15 unidades generamos los movimientos arriesgados BIGHOLEDROP, cuya efectividad era de aproximadamente del $62,16\,\%$. Esto hace que no en todas las ocasiones consiga caer (los humanos tampoco lo logran) y quede atascado en el hueco hasta que expira el tiempo del nivel. Aun así, el comportamiento de nuestro agente rectángulo es muy satisfactorio y queremos destacar que resuelve con solvencia niveles conceptualmente difíciles como el 7 y el 10, y también otros que requieren movimientos con el bit *risky*, como puede ser el nivel 2, en el que los movimientos de tipo HIGHTILT se demuestran efectivos.

7.3.4. Comentarios sobre los resultados obtenidos

En virtud de los resultados anteriores, podemos asegurar que nuestro agente UCM Physics es el mejor agente del rectángulo desarrollado hasta el momento. En las competiciones que hemos evaluado, sus resultados son cuantitativamente y cualitativamente mejores que los de los agentes NKUST, Sub Goal A Star y los respectivos ganadores de las competiciones. Además, lo logra obteniendo los mejores resultados en la inmensa mayoría de los niveles, empleando menos tiempo y completando la práctica totalidad los niveles. En comparación a los jugadores humanos, hemos de decir que su rendimiento todavía es muy similar al de un humano medio. Los porcentajes de puntuación de nuestro agente (penalizado por algunos *bugs*) sobre el agente óptimo (que utiliza los *bugs* a su favor) es de $82,86\,\%$, mientras que para el humano medio (también penalizado por los *bugs*) es de un $77,75\,\%$.

Mientras que en el caso del círculo hemos comprobado que los mejores agentes de años anteriores conseguían resultados próximos al de un humano medio, en el caso del rectángulo la diferencia era todavía abismal. El porcentaje de puntuación óptima que obtiene el agente NKUST es de media un 57,59 % y el del agente Sub Goal A star es un escaso $31,01\,\%$. Nuestro agente ha dado un salto cualitativo de gran calibre, hasta lograr igualar el rendimiento de un jugador humano medio.

7.4. Resultados de los agentes cooperativos

En esta sección, evaluaremos el comportamiento de nuestros agentes cooperativos, los UCM Physics. Para ello, hemos organizado una competición local con los niveles de los años 2013, 2017 y 2022. Como advertimos en el diseño experimental, desaparece el concepto de humano medio por el reto organizativo y logístico para obtener una

Agentes	Puntuación total obtenida
Óptimos	10458
UCM Physics	6937
Ganadores 2013 (CIBot)	4308
NKUST	1209
MARL-GF Agent	1087

Tabla 7.7: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2013.

muestra significativa de participantes que pudieran jugar presencialmente en modo cooperativo a Geometry Friends. Nos planteamos que un único jugador humano controlara a los dos agentes, pero descartamos la idea porque el reto de coordinación para el jugador humano es muy difícil y la comparación con los agentes no sería justa, además de que se perdería la esencia del juego: la cooperación. Por último, en el siguiente enlace se encuentra un vídeo donde los agentes cooperativos UCM Physics realizan una ejecución en cada uno de los niveles de las competiciones.

7.4.1. Competición cooperativa 2013

La primera competición local que llevamos a cabo está basada en la competición inaugural, la del año 2013 y cuyos niveles se encuentran en la figura 7.14. Ese año, las puntuaciones por completar un nivel y capturar un diamante no eran las mismas en todos los niveles. Eso complicaba todo el proceso del cálculo de las puntuaciones y los organizadores de la competición oficial de Geometry Friends decretaron que, a partir del año 2014, los valores de $V_{\text{completed}}$ y $V_{\text{collected}}$ no variarían de un nivel a otro. Para seguir esa política, que permite utilizar el *batch simulator* en lugar de ejecutar las pruebas una por una, decidimos que las recompensas por completar un nivel y por alcanzar un diamante fueran en todos los niveles las mismas. En concreto, serían $V_{\text{completed}} = 1000$ y $V_{\text{collected}} = 100$, como en las competiciones de 2014 del círculo y el rectángulo. Los ganadores del año 2013 fueron los agentes cooperativos CIBot con una puntuación final (adaptada a nuestro sistema de puntuación) de 4308 puntos. En la tabla 7.7, mostramos la puntuación final obtenida por los distintos agentes evaluados y, en la sección A.3.1 (tablas A.34, A.35 y A.36), los resultados de cada agente desglosados por niveles. Con el fin de agrupar toda la información de las tablas anteriores, en la figura 7.15 aparece una gráfica donde se puede ver de manera más ilustrativa los resultados de la competición.

En primer lugar, nuestros agentes UCM Physics son los que obtienen los mejores resultados de la competición, aventajando en más de 2500 puntos a los segundos (aproximadamente un 25 % de la puntuación del agente óptimo). Además, consiguen las mejores puntuaciones en 8 de los 10 niveles y, quizás lo más importante, logran completar todos los niveles al menos una vez. Asimismo, nuestros agentes son capaces de completar en todas las ejecuciones los niveles 1, 3, 4 y 5, en los que la

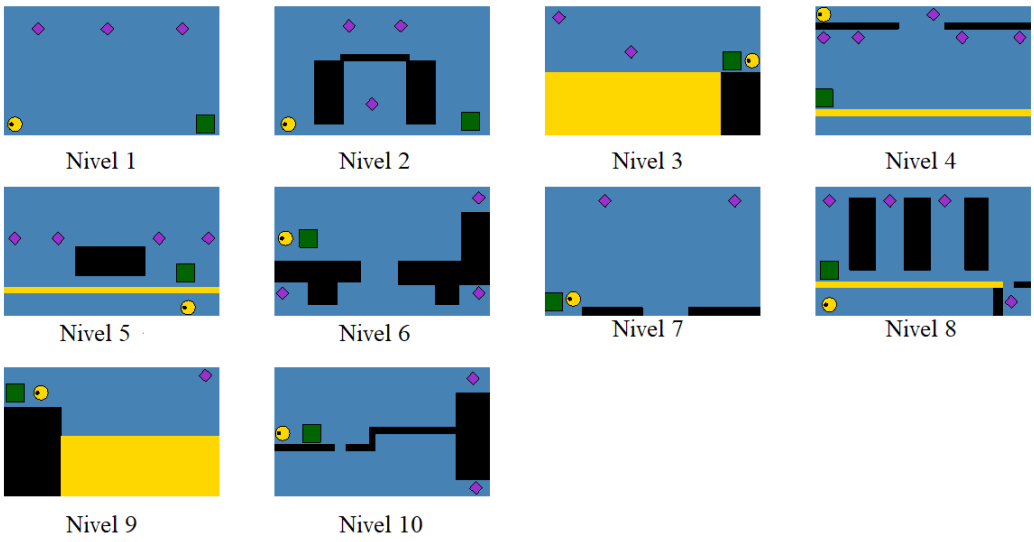


Figura 7.14: Niveles de la competición del año 2013.

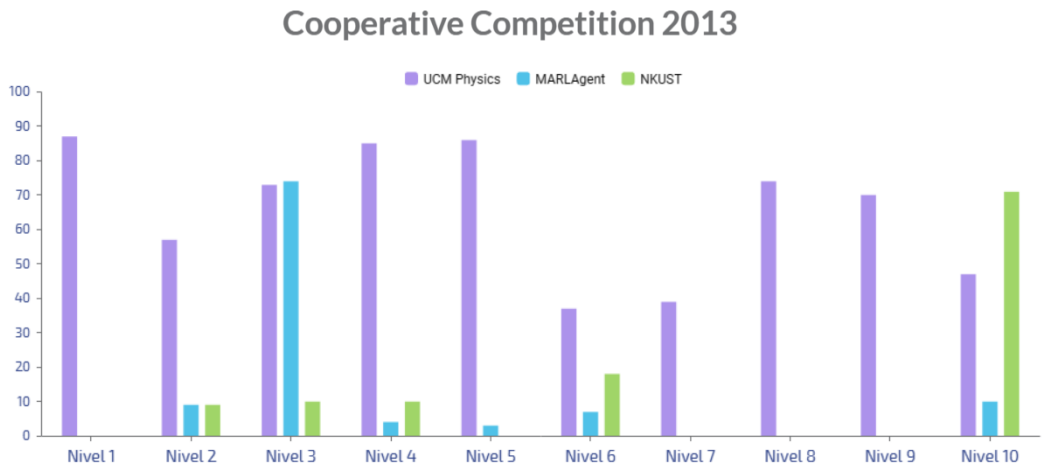


Figura 7.15: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2013.

cooperación es imprescindible. Aquellos en los que presentan más dificultades son, paradójicamente, en los que el modelo de cooperación que se necesita es la división de tareas, por ejemplo, los niveles 6 y 10. Esto se debe a que una parte fundamental de la cooperación consiste en la gestión de ambos agentes del espacio común y a los intercambios de posición cuando los agentes descansan en el orden equivocado en una misma plataforma. Nuestros agentes no son todavía totalmente efectivos en este tipo de retos, lo que influye en sus resultados. Sin embargo, sí que podemos afirmar que la cooperación en la que el círculo utiliza al rectángulo de plataforma tiene una gran efectividad, como demuestran los resultados de los niveles 1, 3, 4, 8 y 9. Con todo, logramos obtener un $66,33\,\%$ de la puntuación óptima en esta competición. En cuanto a las otras dos parejas de agentes, cabe destacar que la única en la que se identifican comportamientos cooperativos consistentes es en la de los agentes

Agentes	Puntuación total obtenida
Óptimos	4321
UCM Physics	3736
MARL-GF Agent	3209
NKUST	1268
Ganadores 2017 (RRT Agent)	1026

Tabla 7.8: Comparación de la puntuación obtenida por diferentes agentes en la competición de 2017.

MARL-GF Agent, que son capaces de resolver el nivel 3, para el cual es necesaria la cooperación. En los niveles 4 y 5 hay alguna ejecución en la que también consiguen cooperar, pero únicamente alcanzan un subconjunto muy reducido de los diamantes del nivel y, en otros niveles donde la cooperación también es necesaria, no logran conseguir ningún diamante. Los agentes NKUST, que por su forma de actuar, pareciese que no tienen implementada ningún tipo de cooperación, se limitan a alcanzar cada uno de ellos los diamantes que tienen accesibles sin contar con la ayuda del otro agente. Como primera aproximación puede ser aceptable, y, de hecho, obtienen los mejores resultados en el nivel 10, donde la estrategia óptima es la división de tareas, pero no parece que busquen implementar ningún tipo de técnica cooperativa.

Para concluir, puede sorprender la diferencia de puntuación entre los ganadores de la competición del año 2013, los agentes CIBot, frente a las dos mejores parejas del estado del arte. Lo cierto es que, si recordamos lo explicado en la sección 2.3.1 donde presentamos a los agentes CIBot, estos estaban sobreespecializados en los niveles públicos y sus resultados en los privados fueron igual de malos que los de los agentes MARL-GF Agent y NKUST. Como veremos en las siguientes competiciones, estas dos últimas parejas de agentes tienen algo más de capacidad de abstracción y no obtienen perores resultados en los niveles privados en comparación con los públicos.

7.4.2. Competición Cooperativa 2017

La segunda competición que celebramos tuvo como niveles los de la figura 7.16, es decir, los del año 2017. En aquella edición, el agente ganador fue RRT Agent, con una puntuación de 1026. Cabe destacar que, en esa ocasión, las puntuaciones por completar un nivel y alcanzar un diamante cambiaron y pasaron a ser $V_{\text{completed}} = 200$ y $V_{\text{collected}} = 100$. En la tabla 7.8, se muestran los resultados obtenidos por los agentes. Se pueden encontrar los resultados desglosados por niveles en la sección A.3.2 (tablas A.37 a A.39). Además, añadimos la figura 7.17, que representa la puntuación por nivel de cada uno de los agentes de manera más resumida e ilustrativa, y que nos guiará para comentar los resultados.

En esta ocasión, aunque los agentes UCM Physics siguen teniendo los mejores re-

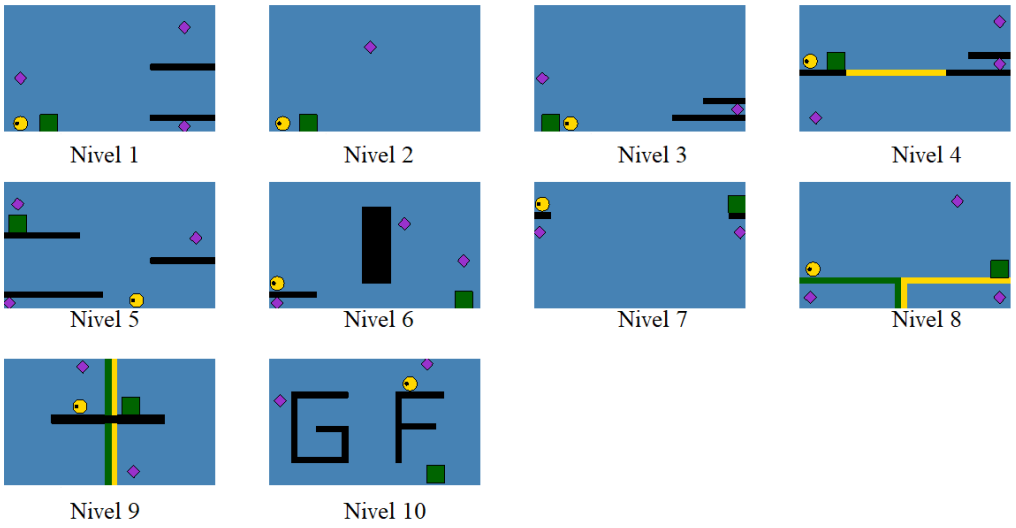


Figura 7.16: Niveles de la competición del año 2017.

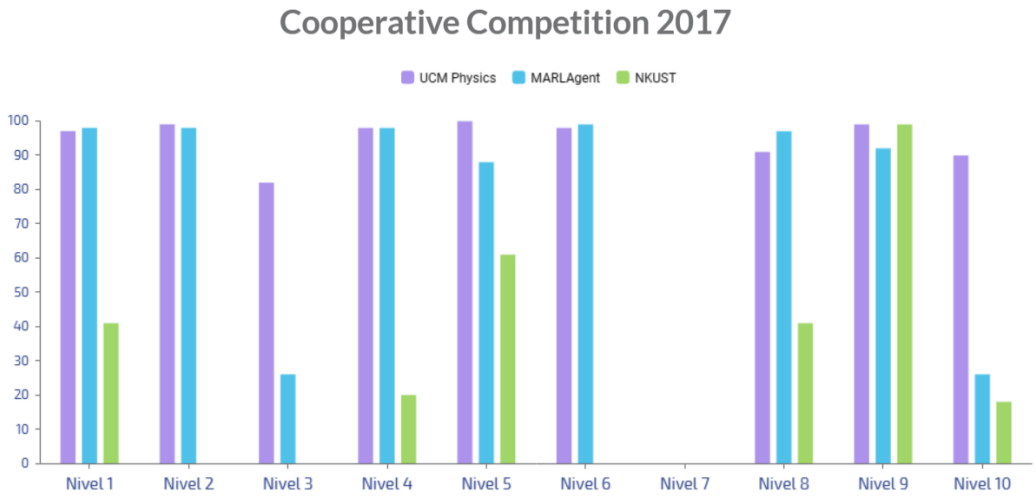


Figura 7.17: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición del año 2017.

sultados, sorprende la mejora de rendimiento de los agentes MARL-GF Agent. Esto nos hace pensar que dichos agentes están demasiado especializados a los niveles de esta competición, vistos los resultados de la anterior. Los resultados de los agentes UCM Physics y MARL-GF Agent son muy parejos en los niveles 1, 2, 4, 5, 6, 8 y 9, en los que ambos agentes completan el nivel en prácticamente la totalidad de las ejecuciones y están razonablemente cerca del agente óptimo. Es en los niveles 3 y 10 donde se encuentra la diferencia fundamental. En el nivel 3, se requiere un tipo de colaboración en la que el círculo ayuda al rectángulo para subir a la plataforma y alcanzar el diamante, y nuestros agentes son los únicos que logran ejecutar dicho comportamiento. Por su parte, el nivel 10 requiere que el círculo utilice el rectángulo como plataforma para alcanzar el diamante de la izquierda, lo que pudiera parecer

que es igual de sencillo o difícil que en otros niveles como el segundo, el sexto o el octavo. Sin embargo, el espacio con el que cuentan ambos agentes para realizar los movimientos cooperativos es reducido y estrecho, lo que dificulta la coordinación de acciones. Con todo, nuestros agentes logran solventar esta falta de espacio en 9 de las 10 ejecuciones. Los agentes NKUST parecen seguir confiando en la división de tareas para resolver los retos cooperativos, lo que hace que únicamente tengan resultados destacables en el nivel 9. En el nivel 7, ninguna de las parejas de agentes logra alcanzar ninguno de los diamantes en ninguna de las ejecuciones. El motivo, por lo menos para los agentes UCM Physics, es uno de los *bugs* del juego, ya que, al caer el rectángulo desde una plataforma a tanta altura, golpea bruscamente el suelo con una esquina y se le bloquea la capacidad de cambiar de forma. Esto impide que el comportamiento del rectángulo sea el deseado. Hemos podido comprobar que si en ese mismo nivel el rectángulo empieza desde la plataforma inferior, sin haber caído y por tanto sin sufrir los efectos negativos del *bug*, nuestros agentes logran alcanzar una media de 1,9 diamantes en 54,1 segundos, es decir, son capaces de completar el nivel casi siempre. En resumen, esta competición ratifica que nuestros agentes cooperan de manera efectiva cuando el círculo debe utilizar al rectángulo de plataforma, pero también al revés como se muestra en el nivel 3, donde el círculo era el que debía servir de apoyo al rectángulo. Asimismo, nuestros agentes usan de manera eficiente la división de tareas ya que los resultados de los niveles 5, 8 y 9 así lo demuestran. Por último, destacamos que aumenta el porcentaje de puntuación obtenida sobre el óptimo, que pasa a ser en esta competición un 86,46 %. A pesar de ello, el hecho de que los agentes MARL-GF Agent hayan aumentado también su rendimiento hasta un 74,27 %, con independencia de que estén más o menos especializados, hace entrever que los niveles de la competición del año 2017, al igual que pasó con el círculo, fueron algo más sencillos.

7.4.3. Competición Cooperativa 2022

Por último, celebramos la competición local asociada a los niveles de la competición oficial GF-IJCAI-ECAI 2022, que mostramos en la figura 7.18. El ganador de la competición fue el agente zvbe10, obteniendo un total de 6722 puntos. Por otro lado, los valores de $V_{\text{completed}}$ y $V_{\text{collected}}$ fueron 1000 y 300, respectivamente. Los resultados obtenidos por los agentes, desglosados por niveles, se pueden encontrar en la sección A.3.3 (tablas A.40 a A.42). Además, en la figura 7.19 y en la tabla 7.9 se puede ver de manera más resumida los resultados de la competición.

En esta competición, los resultados de nuestros agentes UCM Physics vuelven a resultar muy destacados frente al del resto de agentes, doblando en puntuación al segundo clasificado. Nuestro agente logra superar todos los niveles en todas las ejecuciones a excepción de los niveles 2, 3 y 5. En el nivel 2, hay una ejecución en la que el rectángulo cae a la plataforma inferior cuando el diamante de la esquina superior derecha todavía no ha sido alcanzado, por lo que el nivel se vuelve irresoluble. Este error lo achacamos a un fallo puntual del proceso de intercambio de posiciones en la

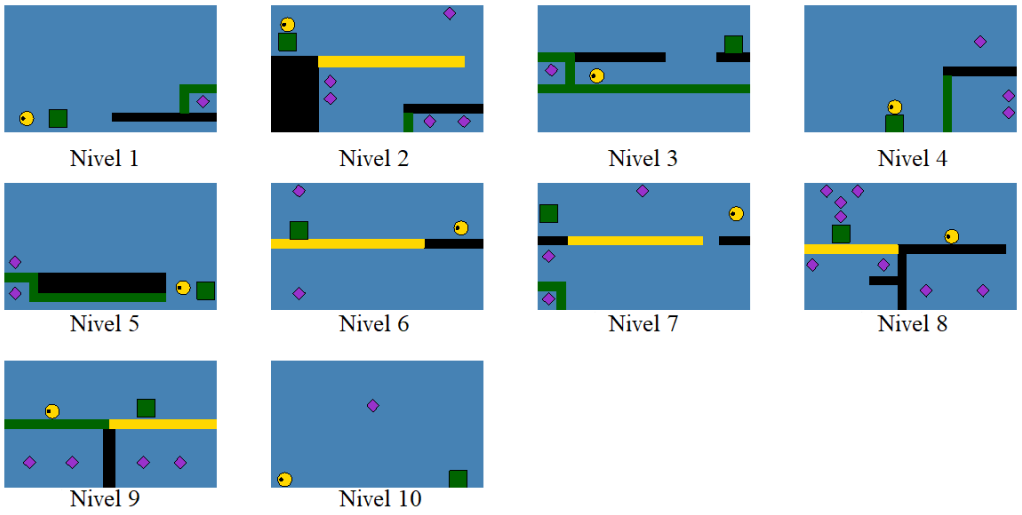


Figura 7.18: Niveles de la competición GF-IJCAI-ECAI 2022.

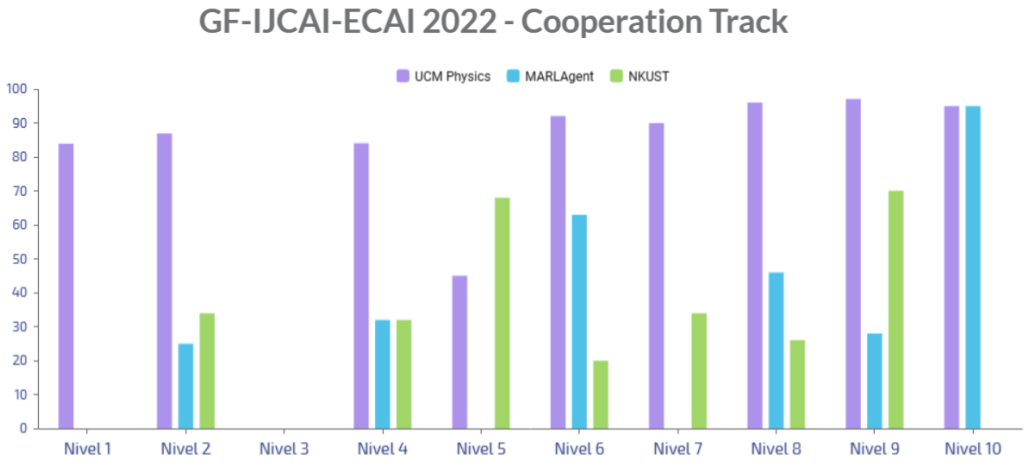


Figura 7.19: Porcentaje de la puntuación óptima obtenido por los agentes en los niveles de la competición GF-IJCAI-ECAI 2022.

Agentes	Puntuación total obtenida
Óptimos	18200
UCM Physics	14760
Ganadores GF-CoG 2022 (zvbe10 Agent)	6722
NKUST	5687
MARL-GF Agent	5448

Tabla 7.9: Comparación de la puntuación obtenida por diferentes agentes en la competición GF-IJCAI-ECAI 2022.

plataforma negra de la parte derecha del nivel, y no consideramos que sea demasiado grave. Este mismo problema de gestión de espacio se repite en el nivel 5, en el que

el círculo y el rectángulo no son siempre capaces de intercambiarse de posición en la plataforma donde comienzan (el rectángulo quiere coger el diamante de la plataforma inferior y el círculo quiere tener espacio suficiente para acelerar y realizar el salto). Sin embargo, en ambos niveles el comportamiento de nuestros agentes es más que aceptable, y los completan en la mayoría de las ejecuciones. El nivel en el que peores resultados obtenemos es el nivel 3, donde se requiere que el rectángulo sea impulsado por el círculo mientras cae para que así rebote y alcance la plataforma de la izquierda. Es un tipo de movimiento que, como advertimos, no hemos generado por la dificultad de identificar un patrón general y por la precisión que debía tener en ejecución. Este es el único reto cooperativo que no hemos abordado, pero el resto de niveles ratifican que está lograda la colaboración entre el círculo y el rectángulo en ambos sentidos. Por ejemplo, somos los únicos capaces de resolver el nivel 1, en el que el círculo debe ayudar al rectángulo a subir a la plataforma y también resolvemos con las mejores puntuaciones los niveles 2, 4, 6, 7 y 8, en los que el rectángulo debe actuar de plataforma del círculo e incluso transportarlo.

El agente MARL-GF Agent vuelve a tener unos resultados malos, como en la competición de 2013, lo que corrobora nuestra hipótesis de que estaba demasiado especializado para resolver los niveles de la competición de 2017 o que solamente sabe resolver los niveles muy sencillos. Con todo, posee comportamientos cooperativos efectivos que le permiten, por ejemplo, obtener los mejores resultados en el nivel 10. Por desgracia, no podemos decir lo mismo de los agentes NKUST, que, si bien obtienen mejores resultados que los agentes MARL-GF Agent, solamente parece entreverse cierta cooperación en el nivel 8. Aun así, sus agentes siguen limitándose a conseguir los diamantes que tienen accesibles individualmente, normalmente sin mucho éxito.

7.4.4. Comentarios sobre los resultados obtenidos

Atendiendo a los resultados anteriores, podemos concluir que nuestros agentes cooperativos son los mejores desarrollados hasta la fecha. Además, implementan distintos modelos de colaboración, que les permiten completar la gran mayoría de los niveles en casi todas las ejecuciones. Hemos logrado dar un paso significativo, dado que, hasta el momento, los agentes desarrollados solamente tenían dos modelos de colaboración: la división de tareas y que el rectángulo ayudara al círculo. Nosotros hemos conseguido mejorar la ayuda del rectángulo al círculo y hemos añadido la colaboración del círculo al rectángulo. Adicionalmente, hemos desarrollado unos patrones suficientemente genéricos como para que nuestros agentes no estén demasiado especializados, limitación de alguna de las implementaciones de años anteriores. Sin embargo, todavía queda margen de mejora en la colaboración y esta se centra en dos aspectos principales: una mayor ayuda del círculo al rectángulo y la gestión del espacio de ambos agentes.

Hemos podido apreciar como el círculo presta su ayuda al rectángulo para subir a

alguna plataforma, pero todavía queda por explorar otros modelos de colaboración, como por ejemplo que el círculo impulse al rectángulo mientras este cae, como en el nivel 3 de la competición GF-IJCAI-ECAI 2022. Incluso hemos llegado a plantearnos la posibilidad de que el círculo pueda servir de plataforma al rectángulo y transportarlo manteniendo el equilibrio. En cuanto a la gestión del espacio, nuestros agentes empeoran su rendimiento cuando los puntos de inicio de sus acciones están muy cercanos, ya que colisionan, se desequilibran y pierden mucho tiempo, llegando en ocasiones a caer de plataformas importantes y volviendo el nivel irresoluble. Asimismo, los intercambios de posición no son todo lo rápidos en tiempos ni todo lo efectivos que nos gustaría, pero, en general, estamos muy satisfechos. Es más, en cuanto a resultados cuantitativos, obtenemos de media un $77,96\,\%$ de la puntuación óptima, cuando los mejores agentes hasta el momento únicamente alcanzaban el $38,19\,\%$, mientras que, en cuanto a resultados cualitativos, hemos logrado dar un gran paso en términos de especialización y colaboración bidireccional que no se había logrado hasta el momento, sentando el nuevo objetivo a batir para los agentes que se desarrollen en el futuro.

En el siguiente capítulo, extraeremos las conclusiones del trabajo visto de manera global. Haremos especial énfasis en los retos para la IA que hemos conseguido resolver, e identificaremos aquellos sobre los que queda aún camino por recorrer, sin olvidarnos de repasar los objetivos que nos planteamos al comenzar el curso.

Capítulo 8

Conclusiones y Trabajo Futuro

En el presente capítulo, describimos las conclusiones del trabajo llevado a cabo durante el curso, dando un resumen de qué se ha desarrollado y qué objetivos se han logrado, y proponemos las líneas por las que este podría ampliarse.

8.1. Conclusiones

En el desarrollo del trabajo abordamos en primer lugar la implementación del agente círculo, que nos serviría de base para el resto del trabajo. Para ello, investigamos los principales agentes que se habían desarrollado hasta el momento, que, en el caso del círculo, eran más que suficientes para identificar qué enfoques eran los más utilizados y cuáles resultaban en mejores agentes. Con esto, cumplimos el objetivo $\mathcal{O}1$, ya que, si bien también estudiamos las propuestas para el agente rectángulo y los agentes cooperativos, debido a su escasez y bajos resultados, ninguna de las soluciones nos sirvió de base, comenzando el desarrollo desde cero en ambos casos.

Tras seleccionar uno de los agentes que funcionaba especialmente bien (MARL-GF Agent), nos basamos en la representación y discretización que se hacía allí para desarrollar nuestro agente círculo, teniendo que adquirir primero nociones intermedias del lenguaje C# para poder analizar la solución de MARL-GF, realizando así el objetivo $\mathcal{O}7$. Sin embargo, durante la implementación, uno de los aspectos clave para la correcta ejecución del agente, cómo conseguir que el círculo llegara a una determinada posición con una determinada velocidad, parecía especialmente complicado. La solución propuesta por el desarrollador del agente MARL-GF implicaba aprendizaje por refuerzo, pero nosotros tuvimos una idea alternativa, lo que nos llevó a implementar ambas técnicas, desarrollando así dos versiones del agente círculo. Más tarde, compararíamos los resultados de los dos agentes, completando así el objetivo $\mathcal{O}5$. Por tanto, acabamos con los agentes UCM Physics y UCM QLearning, prepa-

rados para medir sus fuerzas contra otros agentes anteriores como MARL-GF Agent en las competiciones del círculo.

A continuación, centramos nuestros esfuerzos en implementar un agente rectángulo. Si bien el desarrollo del círculo no había sido sencillo, el rectángulo añadía muchísima complejidad, e introducía muchas más situaciones que debían ser abordadas de maneras muy diferentes. Fue en esta implementación donde cumplimos con creces los objetivos $\mathcal{O}2$ y $\mathcal{O}3$, pues tuvimos que identificar muchas clases de patrones y buscar soluciones adecuadas para cada uno de ellos. Para resolver uno de los problemas que surgieron, tuvimos que utilizar la técnica de aprendizaje por refuerzo, pues no éramos capaces de encontrar una manera de resolverlo añadiendo conocimiento a los agentes. En este caso, nos quedamos con solo uno de los sistemas que habíamos implementado para conseguir llegar a una posición con una velocidad determinada, desarrollando así el agente UCM Physics en su versión rectángulo.

Una vez terminado el desarrollo de los agentes individuales, y de cara a cumplir el objetivo $\mathcal{O}4$, comenzamos la adaptacion de estos agentes a un sistema multiagente que fuera capaz de resolver niveles cooperativos, lo que suponía un reto sin precedentes para nosotros. La absoluta escasez de agentes anteriores que fueran capaces de resolver niveles medianamente complejos hizo este desarrollo increíblemente más complicado, pero, aun así, conseguimos implementar soluciones para los principales problemas de cooperación, lo que nos permitió resolver la mayoría de niveles de competiciones anteriores. Además, debido a que este sistema es más complejo que los agentes individuales y muchas veces toma decisiones difíciles de entender si no se conoce la arquitectura de la solución, decidimos aprovechar la interfaz del juego y desarrollar un sistema de explicabilidad XAI que mostrara qué acción estaban tomando los agentes y qué objetivos tenían a corto y largo plazo para resolver el nivel. Esta característica de nuestra solución facilita en gran medida la comprensión del funcionamiento de los agentes, y cumple el objetivo propuesto $\mathcal{O}6$.

Por último, hemos llevado a cabo simulaciones de algunas competiciones anteriores, que nos permitieran comparar el rendimiento de nuestros agentes con los mejores que había hasta el momento. Asimismo, también aprovechamos para medir el desempeño de una muestra de voluntarios que participaron en un pequeño experimento. De esta forma, pudimos estimar los resultados del jugador humano medio e incluirlos en las comparaciones. Estas competiciones estaban cuidadosamente elegidas para que sus niveles tuvieran retos diferentes, con la finalidad de identificar los aspectos en los que cada agente (o persona) destaca, y en cuáles necesita mejorar. Todos los resultados serían también comparados con el mejor resultado que un jugador experto podía conseguir tras jugar cada nivel repetidas veces, lo que nos serviría para ver cuánto se acerca cada agente al rendimiento óptimo.

Tras realizar el procedimiento de evaluación descrito en el capítulo 7, pudimos comprobar que nuestro círculo UCM Physics obtiene, de media, un 93 % de la puntuación óptima, mientras que nuestro agente UCM QLearning se queda en un 73 %. Como referencia, el mejor agente anterior consigue un 87 % y el humano medio un 85 %,

por lo que conseguimos mejorar el estado del arte para las competiciones del círculo y superar el comportamiento del humano medio.

Para nuestro agente rectángulo, la diferencia con los agentes anteriores es todavía más grande, demostrando que nuestro cuidadoso análisis de los problemas a los que se enfrenta este personaje se traduce en una mejora significativa de rendimiento. De media, obtenemos un resultado del 83 % sobre el agente óptimo, mientras que el humano medio logra un 78 % y el mejor agente anterior se queda en un 58 %. Por ello, consideramos que el salto cualitativo y cuantitativo es sumamente relevante, especialmente teniendo en cuenta cómo varios *bugs* que descubrimos durante el desarrollo afectaban significativamente a la puntuación obtenida por agentes y humanos. De hecho, las puntuaciones óptimas solamente son alcanzadas aprovechando mecánicas inintencionales del juego, y no pueden lograrse consistentemente, por lo que los resultados de nuestro agente son incluso mejores de lo que los números podrían sugerir.

Otra de nuestras mayores contribuciones viene dada en términos de nuestros agentes cooperativos, donde los mejores agentes anteriores obtenían una puntuación media del 38 % del óptimo, mientras que nuestros agentes son capaces de elevar esta cifra al 78 %. Lógicamente, partir de un rectángulo con mejor rendimiento ayuda a lograr estos resultados, pero no queremos menospreciar el esfuerzo que ha supuesto la implementación de técnicas cooperativas. Con esto, concluimos nuestra explicación de cómo hemos cumplido el objetivo $\mathcal{O}8$, a partir del cual hemos sido capaces de cuantificar la mejora que nuestros agentes suponen al estado del arte del juego, que no es en absoluto despreciable.

En resumen, hemos conseguido desarrollar los mejores agentes círculo, rectángulo y cooperativos para el juego Geometry Friends hasta la actualidad, siendo capaces de resolver la gran mayoría de niveles. Además, lo hacen con resultados que superan a los de un humano medio, y están considerablemente cercanos a los óptimos. Por tanto, y como veremos en breve en la sección 8.2.1, esperamos obtener muy buenos resultados en la competición que se celebrará este año en el mes de agosto y a la cual nos vamos a presentar.

Por último, nos gustaría añadir algunas conclusiones personales sobre el trabajo que hemos realizado durante todo el curso. El enfoque que hemos seguido ha consistido en añadir mucho conocimiento experto a nuestros agentes, lo cual nos ha permitido obtener resultados muy satisfactorios. Hemos podido comprobar que, en situaciones determinadas, técnicas de IA aparentemente sencillas como la búsqueda en el espacio de estados, los sistemas de reglas y la representación del conocimiento pueden dar la mejor solución ante problemas complejos. Sin embargo, esto ha sido a costa de invertir mucho tiempo y esfuerzo en la identificación de patrones generales y reglas lo más abstractas posibles para que fueran aplicables a un gran número de situaciones. En contraposición, algunas de las técnicas de IA que hemos aplicado y que están basadas en aprendizaje automático no han dado tan buenos resultados (véase el sistema de elección de la acción de nuestro agente UCM QLearning en compara-

ción con UCM Physics). No obstante, el aprendizaje por refuerzo nos ha permitido solventar algunas de las limitaciones de los sistemas ricos en conocimiento. El ejemplo más claro de esto es el de los movimientos de tipo BIGHOLEDROP, para los cuales no conseguimos encontrar ninguna heurística que pudiera ser utilizada por el agente para completarlos. La solución que ofrecimos se basa en aprendizaje por refuerzo y obtiene resultados razonablemente buenos, lo que refuerza la idea de que la combinación entre conocimiento y aprendizaje puede ser la más idónea.

8.2. Trabajo futuro

En esta sección, comentamos múltiples vías por las cuales el trabajo realizado puede ser ampliado o aprovechado. Sin embargo, comenzamos hablando sobre la próxima competición, prevista para agosto de 2023, y nuestra participación en ella.

8.2.1. Participación en la competición de 2023

Durante todo el documento, hemos mencionado repetidas veces la competición anual, organizada por los desarrolladores del juego y asociada a conferencias internacionales como IJCAI o CoG. En esta sección, detallamos los objetivos que nos hemos propuesto de cara a nuestra participación en la edición de este año, que se celebrará a mediados de agosto.

En primer lugar, hablaremos de los niveles públicos de la competición. Cada competición está formada por cinco niveles públicos, que se muestran al anunciar la competición y sirven para que, al subir los agentes al servidor de la competición, los desarrolladores puedan verificar su correcto funcionamiento; y cinco niveles privados, que solo aparecen cuando se cierra el plazo para realizar envíos. Dado que la competición ya ha sido anunciada, podemos comprobar el rendimiento de nuestros agentes en los niveles públicos de las tres competiciones. Podemos confirmar que, en las tres modalidades de juego (círculo, rectángulo y cooperativo), nuestros agentes son capaces de resolver todos los niveles sin problema (a excepción de uno cooperativo que requiere un movimiento que aún no hemos implementado), lo que reafirma nuestras expectativas de vencer en la competición. Es más, es de destacar que no tuvimos que realizar ningún cambio en la implementación una vez conocimos cuáles eran los niveles públicos, lo cual incide en la capacidad de abstracción de nuestra solución propuesta y nos hace ser optimistas de cara a los niveles privados.

Para participar, debemos redactar y enviar un informe técnico que describa a grandes rasgos las ideas subyacentes en nuestra implementación. Tenemos pensado escribir este informe cuando tengamos los agentes completamente terminados y listos para enviar. Pretendemos realizar algunas modificaciones para perfeccionar algunas carac-

terísticas menores, que hemos tenido que dejar de lado por limitaciones temporales, y escribir el informe después.

8.2.2. Limitaciones y mejoras de este trabajo

Nuestros agentes no son perfectos, y, a medida que aumenta la complejidad del entorno, también lo hacen las limitaciones que tienen. Comentaremos en primer lugar algunas de las concesiones y simplificaciones que hemos realizado con cada agente, y cómo se podrían mejorar sus resultados.

Empezamos con nuestro agente círculo, que es el que tiene mejores resultados y cuenta con pocas limitaciones que solo afectan en niveles específicos y diseñados al detalle. El aspecto más difícil de dominar para un jugador humano es la gestión de las colisiones con paredes y techos, cuyo resultado varía en función de la velocidad angular que tenga el círculo al impactar. Un estudio detallado de este suceso y de cómo aplicarlo podría resolver una gran limitación de nuestro agente. Por último, una planificación que no tenga en cuenta únicamente el número de pasos sino también la forma de ordenar el plan de la manera más conveniente posible ayudaría en algún que otro nivel.

El agente del que más orgullosos estamos es nuestro agente rectángulo, ya que es aquel en el que hemos tenido que sortear el mayor número de retos con técnicas muy diferentes, lo que se ha traducido en una diferencia abismal con el mejor agente anterior. La mayor limitación con la que cuenta nuestro agente viene dada por los *bugs* del juego, especialmente por uno de ellos que le impide cambiar de forma tras una caída violenta. Dejando esto a un lado, también es posible mejorar la manera de resolver las situaciones que ahora solucionamos mediante aprendizaje por refuerzo, lo que no siempre tiene éxito. Finalmente, al igual que en el caso del círculo, una simulación más precisa de los rebotes y las caídas podría mejorar los resultados en niveles concretos.

Por último, para nuestros agentes cooperativos, existen muchas mejoras que podrían incrementar significativamente su rendimiento. Además de las comentadas para los agentes individuales, una mejor gestión de cómo los agentes intercambian posiciones y de cómo se distribuyen el espacio del nivel ayudaría en una gran variedad de situaciones. También sería posible desarrollar la ejecución correspondiente a cuando el círculo debe sortear un hueco entre dos plataformas cercanas mientras está montado en el rectángulo, es decir, lo que hemos llamado movimientos de tipo ADJACENT para el círculo (ver sección 6.2.2). Asimismo, se podrían implementar los movimientos cooperativos más difíciles, en los que el círculo debe ayudar al rectángulo a mantenerse en el aire, impulsándolo en mitad de su vuelo (véase la figura 6.3). Finalmente, una estimación del tiempo que tarda cada agente en llegar a cada lugar del nivel y coger cada diamante optimizaría la distribución de tareas necesaria para la resolución de muchos niveles.

8.2.3. Aprovechamiento de este trabajo por futuros agentes

En este apartado, nos gustaría explicar por qué nuestros agentes podrían ayudar al desarrollo de agentes futuros. Como explicamos en la sección 2.3, podemos dividir las implementaciones de agentes en dos categorías, aquellas que están fuertemente asociadas al modelo que se crea del nivel, y aquellas que utilizan redes neuronales y aprendizaje supervisado profundo. Dado que, hasta el momento, las mejores soluciones empleaban la primera aproximación, es la que hemos decidido utilizar en este trabajo.

La principal limitación con la que cuentan los enfoques neuronales es la escasez de niveles disponibles. A diferencia de juegos como el Dota 2 o el ajedrez, donde las condiciones iniciales de cada partida son siempre las mismas (salvo por los personajes elegidos en el caso del Dota 2), en Geometry Friends estas varían enormemente según la distribución de los obstáculos y los diamantes en cada nivel. En otros juegos que también cuentan con esta dificultad, como el póquer, es muy fácil simular miles de partidas en las que se comienzan con manos diferentes, todo ello de forma automática. Sin embargo, aquí cada nivel debe ser creado con la certeza de que puede resolverse, lo que normalmente requiere interacción humana.

Consideramos que nuestros agentes son lo suficientemente buenos como para que puedan ser aprovechados por un generador automático de niveles. El objetivo sería crear un corpus de niveles que nuestros agentes han sido capaces de resolver al menos una vez, lo que indicaría que los niveles son resolubles. Añadiendo algunos niveles manualmente, para considerar aquellas situaciones que sobrepasan las capacidades de nuestros agentes, se podría disponer de una colección considerable de niveles que aproximaciones neuronales futuras podrían utilizar para entrenarse y aprender a enfrentarse a situaciones muy diferentes. Creemos que, como tantas veces se ha demostrado en numerosos juegos, es posible desarrollar agentes que superen a los mejores humanos mediante redes neuronales, y nuestra contribución con este trabajo podría resultar muy valiosa a este respecto.

Chapter 8

Conclusions and Future Work

In this chapter, we draw the conclusions of the work, giving a summary of what has been developed and what goals have been achieved, and we propose the lines by which it could be expanded.

8.1. Conclusions

In the development of our work we firstly addressed the implementation of the circle agent, which would serve as the basis for the rest of the work. To do this, we investigated the main agents that had been developed so far, which, in the case of the circle, were more than enough to identify which approaches were the most used and which resulted in better agents. With this, we fulfilled the objective $\mathcal{O}1$, since, although the proposals for the rectangle agent and cooperative agents were also studied, due to their scarcity and low results, none of the solutions served us as a basis, and we had to start the development from scratch in both cases.

After selecting one of the agents that worked especially well (MARL-GF Agent), we based ourselves on the representation and discretization that was done there to develop our circle agent, first having to acquire intermediate notions of the C# language to be able to analyze the MARL-GF solution, thus completing the $\mathcal{O}7$ objective. However, during the implementation, one of the key aspects for the correct execution of the agent, how to get the circle to reach a certain position with a certain speed, seemed especially complicated. The solution proposed by the developer of the MARL-GF agent involved reinforcement learning, but we had an alternative idea, which led us to implement both techniques, thus developing two versions of the circle agent. Later, we would compare the results of the two agents, thus fulfilling the objective $\mathcal{O}5$. Therefore, we end up with the UCM Physics and UCM QLearning agents, ready to measure their strength against previous agents, such as MARL-GF

Agent, in the circle competitions.

Next, we focused our efforts on implementing a rectangle agent. Although the development of the circle had not been easy, the rectangle added a lot of complexity, and introduced many more situations that had to be approached in very different ways. It was in this implementation where we more than met the objectives $\mathcal{O}2$ and $\mathcal{O}3$, since we had to identify many classes of patterns and look for adequate solutions for each of them. To solve one of the problems that arose, we had to introduce reinforcement learning, since we were not able to find another way to solve it. In this case, we kept only one of the systems that we had implemented to reach a position with a certain speed, therefore developing the UCM Physics agent in its rectangle version.

Once the development of the individual agents was finished, and in order to meet the objective $\mathcal{O}4$, we began adapting these agents to a multi-agent system capable of solving cooperative levels, which represented an unprecedented challenge for us. The absolute lack of previous agents that were capable of solving moderately complex levels made this development incredibly more difficult, but we still managed to implement solutions for the main challenges of cooperation, which let us solve the majority of the levels from previous competitions. Furthermore, since this system is more complex than individual agents and often makes decisions that are difficult to understand if the user does not understand completely its architecture, we decided to take advantage of the game interface and develop an XAI explainability system that would show what action the agents were taking and what goals they had in the short and long term to solve the level. This feature of our solution greatly facilitates understanding how agents work, and meets the proposed objective $\mathcal{O}6$.

Lastly, we have carried out simulations of some previous competitions, which allowed us to compare the performance of our agents with the best ones up to now. Likewise, we also took the opportunity, after obtaining a sample of volunteers willing to play Geometry Friends, to record the human results at those levels and include them in the comparisons. These competitions were carefully chosen so that their levels had different challenges, in order to identify the aspects in which each agent (or person) excels, and in which it needs to improve. Every result would also be compared to the best result that a skilled player could achieve after playing each level repeatedly, which would help us indentify how close each agent is to optimal performance.

After carrying out the evaluation procedure described in Chapter 7, we were able to verify that our circle UCM Physics obtains, on average, 93 % of the optimum score, while our agent UCM QLearning remains at 73 %. For reference, the best previous agent gets 87 % and the average human achieves 85 %, so we managed to improve the state of the art for circle competitions and surpass the average human behavior.

For our rectangle agent, the difference from previous agents is even bigger, showing that our careful analysis of the situations this character has to face translates into a significant performance boost. On average, we get a result of 83 % over the best

possible scores, while the average human gets 78 % and the previous best agent stays at 58 %. For this reason, we consider that the qualitative and quantitative leaps are quite relevant, especially considering how the *bugs* we discovered during development significantly impacted the scores of agents and humans alike. In fact, optimal scores are only achieved by taking advantage of unintended game mechanics, and cannot be achieved consistently, so our agent's results are better than what the numbers might suggest.

Another of our biggest contributions comes in terms of our cooperative agents, where the previous best agents obtained an average score of 38 % over the best possible results, while our agents are able to raise this number to 78 %. Logically, starting from a rectangle with better performance helps to achieve these numbers, but we do not want to underestimate the effort that the implementation of cooperative techniques has involved. With this, we conclude our explanation of how we have met the objective $\mathcal{O}8$, from which we have been able to quantify the improvement that our agents represent to the state-of-the-art of the game, which is by no means negligible.

To sum up, we have managed to develop the best circle, rectangle and cooperative agents for the game Geometry Friends to date, being able to solve the vast majority of levels. Furthermore, they do so with results that exceed those of the average human, and are considerably close to optimal. Thus, and as we will see in Section 8.2.1, we expect to have excellent results in this year's competition, which will take place in August.

Finally, we would like to add some personal conclusions about the work we have done throughout the course. The approach we have followed has been grounded in the inclusion of a lot of expert knowledge to our agents, which has allowed us to obtain very satisfactory results. We have been able to experience that, in certain situations, apparently simple AI techniques such as state space search, rule systems and knowledge representation can provide the best solution to complex problems. However, this has been at the cost of spending a lot of time and effort in identifying general patterns and rules that are as abstract as possible so that they are applicable to a large number of situations. In contrast, some of the AI techniques we have applied that are based on machine learning have not performed as well (see the action choice system of our UCM QLearning agent compared to UCM Physics). However, reinforcement learning has allowed us to overcome some of the limitations of knowledge-rich systems. The best example of this is the BIGHOLEDROP type movements, for which we were unable to find any heuristic that could be used by the agent to complete them. The solution we offered is based on reinforcement learning and obtains reasonably good results, which reinforces the idea that the combination of knowledge and learning may be the most suitable.

8.2. Future work

In this section, we discuss multiple ways in which the work done can be expanded or used. However, we start by talking about the next competition, scheduled for August 2023, and our participation in it.

8.2.1. Participation in the 2023 competition

Throughout the document, we have repeatedly mentioned the annual competition, organized by the game developers and associated with international conferences such as IJCAI or CoG. In this section, we detail the objectives that we have set for our participation in this year's edition, which will be held in mid-August.

First of all, we will talk about the public levels of the competition. Each competition is made up of five public levels, which are shown when the competition is announced and are used so that, when developers upload the agents to the competition server, they can verify they work as intended; and five private levels, which only appear when the competition deadline for submissions closes. Since the competition has already been announced, we can check the performance of our agents at the public levels of the three competitions. After evaluating them separately, we confirm that they are capable of solving all of them without trouble (except for a cooperative one that requires a move we have not implemented yet), which reaffirms our expectations of winning the competition. In addition, we would like to highlight the fact that we did not need to make any changes to our implementations once the public levels were released, which enhances the abstraction power of our solution and makes us be optimistic about the performance of our agentes at the private levels.

To participate, we must write and submit a technical report that outlines the ideas behind our implementation. We intend to write this report when we have the agents completely finished and ready to submit. We intend to make some modifications to refine some minor features, which we have had to put aside due to time constraints, and write the report later.

8.2.2. Limitations and improvements of this work

Our agents are not perfect, and as the complexity of the environment increases, so do the limitations they have. We will first discuss some of the compromises and simplifications we have made with each agent, and how their results could be improved.

We start with our circle agent, which is the one with the best results and only has a few limitations that only affect specific levels and designed in detail. The most

difficult aspect for a human player to master is the management of collisions with walls and ceilings, since their outcome varies depending on the angular velocity the circle has before colliding. A detailed study of this event and how to apply it could solve a major limitation of our agent. Finally, a planning that takes into acount not only the number of steps, but also the way to order the plan in the most convenient way possible, would help on some level or another.

The agent we are most proud of is our rectangle agent, since it is the one in which we have had to overcome the greatest number of challenges with very different techniques, which has resulted in an abysmal difference with the best previous agent. The biggest limitation that our agent has is due to the game *bugs*, especially one of them that prevents it from changing shape after a violent fall. Leaving this aside, it is also possible to improve the method to solve the situations we now resolve through reinforcement learning, which is not always successful. Lastly, as in the case of the circle, more accurate collision and fall simulations could improve the results at particular levels.

Finally, for our cooperative agents, there are many improvements that could significantly improve their performance. In addition to the ones discussed for individual agents, better management of how agents swap positions and how level space is distributed would help in a wide variety of situations. It would also be possible to develop the execution corresponding to when the circle must avoid a gap between two nearby platforms while it is mounted on the rectangle. Additionally, we suggest the implementation of the most difficult cooperative movements, in which the circle must help the rectangle to stay in the air, propelling it in the middle of its flight (see Figure 6.3). Lastly, an estimate of the time it takes each agent to reach each site and take each diamond would optimize the distribution of tasks necessary to solve many levels.

8.2.3. Exploitation of this work by future agents

In this section, we would like to explain why our agents could help the development of future agents. As we discussed in Section 2.3, we can divide agent implementations into two categories, those that are strongly associated with the model that is created from the level, and those that use neural networks and deep learning. Since, up to now, the best agents used the first approach, it is the one we have decided to use in this work.

The main limitation of neural approaches is the scarcity of levels available. Unlike games like Dota 2 or chess, where the initial conditions of each game are always the same (except for the chosen characters in the case of Dota 2), in Geometry Friends these vary enormously depending on the distribution of obstacles and diamonds in each level. In other games that also have this difficulty, such as poker, it is very easy to simulate thousands of games in which you start with different hands, all

automatically. However, here each level must be created with the certainty that it can be solved, which usually requires human interaction.

We think our agents are good enough to be exploited by an automatic level generator. The goal would be to create a corpus of levels that our agents have been able to solve at least once, indicating that the levels are solvable. By adding some levels manually, to consider those situations that exceed the capabilities of our agents, we could have a significant amount of levels that future neural approaches could use to train and learn how to deal with vastly different situations. We believe that, as numerous games have shown so many times, it is possible to develop agents that outperform the best humans using neural networks, and our contribution with this work could be very valuable in this regard.

Contribuciones Personales

A continuación, cada uno de los autores detallará sus contribuciones personales al proyecto. No obstante, advertimos que la gran parte del trabajo ha sido desarrollada de forma colaborativa y no hay prácticamente ninguna parte del proyecto que haya sido desarrollada exclusivamente por uno de nosotros.

Alberto Almagro Sánchez

El trabajo que hemos realizado entre Juan Carlos y yo ha estado completamente distribuido desde el principio. La primera parte del trabajo se centró en investigar el estado del arte para los problemas que íbamos a tratar, empezando por hacer un agente círculo. Durante esta fase, Juan Carlos y yo nos distribuimos el estudio de las aproximaciones anteriores cronológicamente, de modo que yo me encargaría de aquellas más recientes. Al mismo tiempo, creé el repositorio de Github y busqué plantillas para escribir la memoria.

La parte troncal del desarrollo del código se hizo de manera simultánea mediante la herramienta Live Share. Por tanto, pocos son los aspectos de los agentes que haya implementado solamente uno de nosotros. Entre aquellos que hice yo, se encuentran el diseño y optimización del algoritmo de búsqueda que empleamos para diseñar el plan que nuestros agentes intentan seguir, la implementación de los movimientos HIGHTILT del rectángulo y el sistema para evitar caídas de ambos agentes, que se detalla en las secciones 4.3.5 y 5.4.5. Juan Carlos, por otro lado, implementó funcionalidades como el sistema de explicabilidad y el depurador visual que hemos utilizado durante todo el desarrollo.

Donde sí que ha habido una mayor división de tareas es en la redacción de la memoria, aunque todo lo que escribía uno de los dos era minuciosamente revisado por el otro, con el fin de evitar erratas y mejorar la redacción lo máximo posible. Por eso, si bien listaré a continuación una serie de secciones que me he encargado de escri-

bir, muchas han sido significativamente modificadas por Juan Carlos, por lo que se debe entender como escribir el núcleo del contenido de la sección y no como haber realizado por completo esa parte del documento. De la misma forma, he contribuido en menor medida a las secciones que ha escrito él.

Más concretamente, he escrito el resumen y los capítulos 1 y 8, que corresponden a la introducción y a la conclusión, he incluyen la motivación y los objetivos del trabajo. Posteriormente, he traducido todos estos capítulos al inglés. Del resto de capítulos, me he encargado de escribir varias secciones, como aquellas relativas a las características que había implementado de manera individual, entre las cuales destacan las secciones 3.3.2 y 3.3.3, en las que se describe el algoritmo de búsqueda y cuál es el algoritmo general de actuación de los agentes, y la sección 5.2.2, relativa a los movimientos HIGHTILT, entre otras. También escribí las secciones relativas al aprendizaje por refuerzo, es decir, la 4.3.3 y la 5.2.2, y otras como las secciones 4.3.5 y 5.4.5 de comentarios adicionales de menor impacto.

Además de estas secciones, en el capítulo 3, redacté la sección 3.2, correspondiente al material proporcionado por los creadores del juego para realizar el desarrollo. En el capítulo 4, correspondiente al círculo, compuse las secciones 4.1, de introducción, y 4.3.4, que explica la política de saltos que siguen nuestros agentes. En el capítulo 5, relativo al rectángulo, escribí también la sección 5.3, de las particularidades correspondientes a cómo realiza el rectángulo su trazado de un plan a seguir, y las secciones 5.4.3 y 5.4.4, referidas a las acciones concretas que toma el rectángulo en cada momento para cumplir los movimientos del plan. Por último, del capítulo de técnicas cooperativas, redacté la sección 6.2, que incluye la identificación de plataformas ficticias del círculo y la generación y filtrado de movimientos que ambos agentes deben realizar de manera conjunta. La mayoría de las imágenes de estas secciones, y alguna concreta de otras, están editadas por mí, aunque esto fue un trabajo que mi compañero y yo hacíamos según lo requerían las secciones que teníamos que escribir o si considerábamos que mejoraba la facilidad de comprensión de una sección escrita por el otro.

Dejando a un lado la memoria, me he encargado de hacerme un jugador experto de Geometry Friends, jugando múltiples veces cada nivel y extrayendo los resultados óptimos que Juan Carlos utilizaría para comparar los resultados de los agentes. En particular, descubrí cómo el uso de *bugs* y mecánicas inintencionadas permiten obtener tiempos extremadamente bajos cuando se juega con el rectángulo de manera individual, y grabé vídeos que así lo demuestran. A continuación, subí algunos de estos vídeos, junto a la gran cantidad de animaciones que grabó mi compañero, a YouTube.

Además, debido a disponer de ligeramente más conocimientos de mecánica clásica, he modelado algunas de las situaciones físicas y he simplificado la obtención de algunas ecuaciones de movimiento que empleamos en nuestros sistemas de reglas. Por otro lado, he sido el responsable de la organización y reestructuración del repositorio, creación del archivo `README` y especificación de la licencia que utilizaremos cuando

liberemos el código. Finalmente, en lo relativo a tareas de comunicación, contacté y realicé las sesiones de evaluación con ocho personas, y he sido el que tenido más comunicación con los tutores mediante correo electrónico, si bien es importante resaltar que todos los mensajes eran acordados previamente entre los dos miembros del equipo.

Juan Carlos Llamas Núñez

Al comenzar el trabajo, nuestros primeros pasos consistieron en recopilar información sobre el juego y sobre el estado del arte del problema para crear una idea general sobre cuáles habían sido las aproximaciones que se habían llevado a cabo anteriormente. Aunque ambos participamos en esta recopilación de información, yo me centré más en los primeros años de la competición, es decir, los agentes que se habían implementado hasta el año 2019 aproximadamente. También fui quien realizó una primera aproximación al entorno, leyendo toda la información que estaba disponible en la página web de la organización de Geometry Friends sobre la instalación, la implementación y las herramientas con las que venía incorporado el juego como el depurador visual, el *batch simulator* y la herramienta para crear nuevos niveles. Gracias a ello, puede escribir una versión inicial del capítulo 2, que posteriormente fue modificada y revisada por ambos miembros de la pareja.

Durante el desarrollo de los agentes, el trabajo fue cooperativo casi en su totalidad. Alberto y yo nos conectábamos mediante la herramienta Live Share de Visual Studio mientras manteníamos una videoconferencia con Discord y desarrollábamos el código conjuntamente. Yo era el que ejercía de anfitrión, por lo que el código se almacenaba en mi equipo y después era el que realizaba los *commits* al repositorio de GitHub. Sin embargo, quiero recalcar que el trabajo era totalmente cooperativo y coordinado.

Gracias a que las prestaciones de mi equipo son algo mejores que las de mi compañero, fui el encargado de todas aquellas tareas que requerían un mejor rendimiento del ordenador. Entre ellas se encuentra la ejecución de las competiciones locales, en las que, una vez habíamos decidido cuáles iban a ser los niveles, yo era el que se dedicaba a ejecutar a nuestros agentes y procesar los resultados. Además, para que las competiciones fueran lo más justas posibles, no quisimos evaluar a algunos agentes en un equipo y a otros agentes en otro. Asimismo, identifiqué, de entre los muchos agentes proporcionados por la organización, aquellos cuyos resultados era interesante comparar con los nuestros en las ya mencionadas competiciones. Para cerrar el tema de las competiciones, también realicé seis de las sesiones con participantes humanos, redacté el capítulo 7 de resultados, elaboré los cuestionarios y las instrucciones para las sesiones no presenciales de evaluación y escribí los Apéndices A y B en los que se recogen las tablas con las puntuaciones de los agentes en las competiciones y se detalla el proceso seguido durante las sesiones con humanos, respectivamente.

El hecho de contar con un equipo informático que tuviera un mayor rendimiento y el haber tenido un mayor contacto con la herramienta del *batch simulator* también propició que yo ejecutara el entrenamiento con aprendizaje por refuerzo de los agentes círculo y rectángulo, una vez había sido diseñado y codificado conjuntamente. El entrenamiento para el agente círculo permitía decidir la acción que tomar para llegar a una posición con una determinada velocidad y el del rectángulo aprender a realizar los movimientos BIGHOLEDROP.

Durante el desarrollo del código, yo me sentía más cómodo con el depurador visual proporcionado por el juego, y eso condujo a que fuera el responsable de elaborar la interfaz visual de explicabilidad que presentamos en la sección 6.5 y en cuya redacción también intervine. En cuanto al resto del código, salvo el algoritmo de búsqueda, la implementación de los movimientos HIGHTILT del rectángulo y alguna otra funcionalidad adicional, ambos miembros de la pareja aportamos por igual, ya que desarrollábamos el código conjuntamente, en tiempo real y mientras estábamos conectados con una conferencia en la que comentábamos todos los detalles.

En relación a la memoria, evidentemente he revisado todo lo que ha escrito Alberto, aportando sugerencias y proponiendo alternativas sobre los conceptos que no habían quedado claros. Y en el sentido contrario, en las partes de la memoria en las que yo he redactado la primera versión, he aceptado de buen grado las recomendaciones que me ha hecho Alberto y las he corregido. En ese aspecto, la comunicación entre nosotros ha sido en todo momento muy fluida y bidireccional.

Si vamos a aspectos más concretos, yo he sido el encargado de realizar las animaciones a las cuales se pueden acceder a lo largo de la memoria por los distintos enlaces. El motivo es que las herramientas con las que las realizábamos inicialmente eran distintas y, buscando la uniformidad y atendiendo a que mi herramienta daba mejores resultados, decidimos que yo fuera quien las realizara. También he contribuido al trabajo elaborando muchas de las figuras, aunque eso no se puede decir que haya sido una tarea exclusivamente mía ya que, según íbamos redactando la memoria, cada uno añadíamos las figuras que considerábamos que iban a ayudar más al lector a comprender los conceptos explicados.

En cuanto al núcleo de la memoria, es decir, los capítulos 3, 4, 5 y 6, nos dividimos el trabajo y uno escribió la primera versión de algunas secciones y otro, la de las restantes, aunque posteriormente ambos revisábamos todas. En el capítulo 3, sobre las partes comunes del círculo y el rectángulo, yo me encargué de realizar la sección 3.1, que es un resumen sobre el algoritmo general que permite a los agentes completar los niveles individuales, y de la sección 3.3.1, en la que se explica la representación del nivel. En el capítulo 4, sobre nuestros agentes círculos, yo me dediqué a redactar las secciones 4.2 y 4.3, sobre la representación del nivel y la ejecución del plan del círculo, salvo las subsecciones en las que se explicaba el entrenamiento con aprendizaje por refuerzo y los comentarios adicionales sobre la ejecución del plan. En cuanto al capítulo 5 del rectángulo, elaboré los apartados introductorios y de representación del nivel (5.1 y 5.2), salvo la generación y filtrado de movimientos de tipo HIGHTILT

y BIGHOLEDROP. Por último, en el capítulo 6 sobre los agentes cooperativos, redacté las secciones 6.1 (sobre los retos de la cooperación), 6.3 (generación del plan cooperativo) y 6.4 (ejecución del plan cooperativo), además de la ya mencionada sección 6.5 del sistema de explicabilidad.

Por último, también nos repartimos los roles de comunicación. Mientras Alberto era el responsable de la comunicación no presencial con los tutores del TFG (vía correo electrónico), yo era la persona que mantenía contacto con los organizadores de la competición, para preguntarles cualquier duda que nos surgiera o demandarles algunos niveles o agentes de años anteriores.

Bibliografía

[1] Abreu, J. Reinforcement Learning Agent using Neural Networks for Geometry Friends. Página web de Geometry Friends, `https://gaips.inesc-id.pt/geometryfriends/?page_id=476`, 2017. Accedido por última vez: 9 de noviembre de 2022.

[2] Bengio, Y., Courville, A. y Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, páginas 1798–1828, 2013.

[3] Benoît, V., Alexandre, D. y Nakashima, T. Geometry Friends Competition (CIG 2014) OPU-SCOM. OPU-SCOM Rectangle Technical Report, página web de Geometry Friends, `https://gaips.inesc-id.pt/geometryfriends/?page_id=476`, 2014. Accedido por última vez: 28 de mayo de 2023.

[4] Benoît, V., Alexandre, D. y Nakashima, T. Geometry Friends Competition (CIG 2015) OPU-SCOM. OPU-SCOM Rectangle Technical Report, página web de Geometry Friends, `https://gaips.inesc-id.pt/geometryfriends/?page_id=476`, 2015. Accedido por última vez: 28 de mayo de 2023.

[5] Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., d. O. Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F. y Zhang, S. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680*, 2019.

[6] Bonyadi, M. R. y Michalewicz, Z. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. *Evolutionary Computation*, vol. 25(1), páginas 1–54, 2017. ISSN 1063-6560.

[7] Brandão, D. *A deep learning approach to the Geometry Friends game*. Proyecto Fin de Carrera, Insituto Superior Técnico, Universidade de Lisboa, 2017.

[8] Brown, N. y Sandholm, T. Superhuman AI for multiplayer poker. *Science*, vol. 365, páginas 885 – 890, 2019.

[9] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S. y Colton, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4(1), páginas 1–43, 2012.

[10] Campbell, M., Hoane, A. y hsiung Hsu, F. Deep Blue. *Artificial Intelligence*, vol. 134(1), páginas 57–83, 2002. ISSN 0004-3702.

[11] Cassano, L. y Sayed, A. H. Logical Team Q-learning: An approach towards factored policies in cooperative MARL. *arXiv preprint arXiv:2006.03553*, 2021.

[12] Cook, B. y Huber, M. Abstraction-guided Planning for Geometry Friends. BrianCook Technical Report, página web de Geometry Friends, `https://geometryfriends.gaips.inesc-id.pt/results?competition=GF-CoG+2019+-+Circle+Track`, 2019. Accedido por última vez: 28 de mayo de 2023.

[13] Cunningham, P., Cord, M. y Delany, S. J. *Supervised Learning*, páginas 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-75171-7.

[14] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik*, vol. 1(1), páginas 269–271, 1959.

[15] Fischer, D. *Development of search-based agents for the physics-based simulation game geometry friends*. Proyecto Fin de Carrera, Maastricht University, 2015.

[16] Franklin, G. F., Powell, J. D. y Emami-Naeini, A. *Feedback Control of Dynamic Systems (6th Edition)*. Pearson, 2010. ISBN 0-13-601969-2.

[17] de Gusmão, P. V. N. *Cooperation through Reinforcement Learning in a Collaborative Game*. Proyecto Fin de Carrera, Insituto Superior Técnico, Universidade de Lisboa, 2015.

[18] Hart, P., Nilsson, N. y Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, vol. 4(2), páginas 100–107, 1968.

[19] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z. y Gruslys, A. Deep Q-learning from Demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

[20] Hsiao, C. Solving two agent cooperation issue in geometry friends using a motion space-based approach. Página web de Geometry Friends, `https://geometryfriends.gaips.inesc-id.pt/results?competition=GF-CoG+2019+-+Cooperation+Track`, 2019. Accedido por última vez: 9 de noviembre de 2022.

[21] Kim, H.-T. y Kim, K.-J. Hybrid of Rule-based Systems Using Genetic Algorithm to Improve Platform Game Performance. *Procedia Computer Science*, vol. 24, páginas 114–120, 2013.

[22] Kim, H.-T., Yoon, D.-M. y Kim, K.-J. Solving Geometry Friends using Monte-Carlo Tree Search with directed graph representation. En *2014 IEEE Conference on Computational Intelligence and Games*, páginas 1–2. 2014.

[23] LaValle, S. M. Rapidly-exploring random trees: a new tool for path planning. *The annual research report*, 1998.

[24] Li, Z., Liu, F., Yang, W., Peng, S. y Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33(12), páginas 6999–7019, 2022.

[25] Lin, Y., Wei, L. y Chang, T. Using A* and Q-learning Algorithms to Implement the Task of Geometry Friends Single AI Track. KUAS-IS Lab Technical report, página web de Geometry Friends, `https://gaips.inesc-id.pt/geometryfriends/?page_id=476`, 2014. Accedido por última vez: 28 de mayo de 2023.

[26] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. y Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, vol. 518(7540), páginas 529–533, 2015. ISSN 00280836.

[27] Oonishi, H. y Iima, H. Circle Agent Displaying Generalization Ability for Geometry Friends. KIT Agent 2017 Technical Report, página web de Geometry Friends, `https://geometryfriends.gaips.inesc-id.pt/results?competition=GF-CoG+2022+-+Circle+Track`, 2017. Accedido por última vez: 28 de mayo de 2023.

[28] Oonishi, H. y Iima, H. Improving generalization ability in a puzzle game using reinforcement learning. En *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, páginas 232–239. 2017.

[29] Quitério, J., Prada, R. y Melo, F. S. A reinforcement learning approach for the circle agent of geometry friends. En *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, páginas 423–430. 2015.

[30] Riedmiller, M. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. En *Machine Learning: ECML 2005* (editado por J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge y L. Torgo), páginas 317–328. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-31692-3.

[31] Rodrigues Gomes, E. y Kowalczyk, R. Dynamic Analysis of Multiagent Q-Learning with ε-Greedy Exploration. En *Proceedings of the 26th Annual International Conference on Machine Learning*, página 369–376. Association for Computing Machinery, New York, NY, USA, 2009. ISBN 9781605585161.

[32] Russell, S. y Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edición, 2010.

[33] Salta, A., Prada, R. y Melo, F. A new approach for Geometry Friends' RRT Agents. RRT Agent 2017 Technical Report, página web de Geometry Friends, `https://geometryfriends.gaips.inesc-id.pt/results?competition=GF-CoG+2022+-+Rectangle+Track`, 2017. Accedido por última vez: 28 de mayo de 2023.

[34] Salta, A., Prada, R. y Melo, F. S. Solving motion and action planning for a cooperative agent problem using Geometry Friends. En *Proc. 20th Portuguese Conf. Artificial Intelligence*, páginas 86–97. 2019.

[35] Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, vol. 3(3), páginas 210–229, 1959.

[36] Sequeira, R. A. *Building a Multi-Agent Learning System for Geometry Friends*. Proyecto Fin de Carrera, Insituto Superior Técnico, Universidade de Lisboa, 2019.

[37] Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. y Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, vol. 529, páginas 484–489, 2016.

[38] Simões, D. A., Lau, N. y Reis, L. P. Guided Deep Reinforcement Learning in the Geometry Friends Environment. *2018 International Joint Conference on Neural Networks (IJCNN)*, páginas 1–7, 2018.

[39] Soares, J. y Rodrigues, P. Rule base cooperation in Geometry Friends. jrafaelsoares Technical Report, página web de Geometry Friends, `https://geometryfriends.gaips.inesc-id.pt/results?competition=GF-CoG+2019+-+Cooperation+Track`, 2019. Accedido por última vez: 28 de mayo de 2023.

[40] Soares, R., Leal, F., Prada, R. y Melo, F. Rapidly-Exploring Random Tree approach for Geometry Friends. En *DiGRA/FDG 16 - Proceedings of the First International Joint Conference of DiGRA and FDG*, vol. 13. Digital Games Research Association and Society for the Advancement of the Science of Digital Games, Dundee, Scotland, 2016. ISBN ISSN 2342-9666.

[41] Stanley, K. O. y Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, vol. 10(2), páginas 99–127, 2002.

[42] Tipler, P. y Mosca, G. *Física para la ciencia y la tecnología. I: Mecánica, oscilaciones y ondas, termodinámica*. Editorial Reverte, 2010. ISBN 9788429144291.

[43] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C. y Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, vol. 575(7782), páginas 350–354, 2019.

[44] Wang, T., Rui, Y. y Hu, S.-M. Optimal adaptive learning for image retrieval. En *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, páginas I–I. 2001.

[45] Watkins, C. J. C. H. y Dayan, P. Q-learning. *Machine Learning*, vol. 8, páginas 279–292, 1992.

[46] Zickler, S. y Veloso, M. Efficient Physics-based planning: Sampling search via non-deterministic tactics and skills. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 1, páginas 27–33, 2009.

[47] Özgen, A. C., Fasounaki, M. y Ekenel, H. K. Generalized circle agent for geometry friends using deep reinforcement learning. En *2018 26th Signal Processing and Communications Applications Conference (SIU)*, páginas 1–4. 2018.

Apéndice A

Resultados de las competiciones

En este apéndice recogemos las tablas de los resultados de las competiciones locales que realizamos con el objetivo de comparar a nuestros agentes con los mejores de los años anteriores. Este apéndice se estructura con una primera sección que contiene los resultados de las tres competiciones del círculo, una segunda sección con los resultados de las tres competiciones del rectángulo y una última con las tres competiciones cooperativas.

A.1. Resultados de los agentes círculo

En esta sección se presentan las tablas con los resultados de las tres competiciones del círculo que toman sus niveles de las competiciones oficiales de los años 2014, 2017 y 2022. Los agentes evaluados en este caso son UCM Physics, UCM QLearning, MARL-GF Agent, AGAgent, al humano medio y al agente óptimo. La información sobre los ganadores de las competiciones oficiales de esos años se encuentra en las páginas webs `https://gaips.inesc-id.pt/geometryfriends/` y `https://geometryfriends.gaips.inesc-id.pt`.

A.1.1. Competición Círculo 2014

Se puede encontrar la información de nuestro agente UCM Physics en la tabla A.1, de nuestro agente UCM QLearning en la tabla A.2, del agente MARL-GF Agent en la tabla A.3 y del agente AGAgent en la tabla A.4. Adicionalmente, las tablas A.5 y A.6 recogen los resultados obtenidos por un jugador humano medio y los resultados óptimos, respectivamente.

UCM Physics - Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	5,7	913
2	3	3	45	13,2	1007
3	3	3	60	18	1000
4	4	4	80	20,7	1141
5	2	2	70	17,7	948
6	2	2	40	10,5	937
7	3	2,6	60	32	726
8	3	3	40	24,8	681
9	3	3	80	26,4	971
10	3	3	100	25,9	1040
			Puntuación total		9364

Tabla A.1: Resultados del círculo del agente UCM Physics en la competición de 2014.

UCM QLearning -Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	11,5	625
2	3	3	45	17,8	904
3	3	2,9	60	24,2	887
4	4	4	80	18,7	1166
5	2	1,6	70	43,1	544
6	2	1,4	40	25,5	503
7	3	2,9	60	17,3	1002
8	3	2	40	40	200
9	3	3	80	32,5	894
10	3	2,2	100	94,4	276
			Puntuación total		7001

Tabla A.2: Resultados del círculo del agente UCM QLearning en la competición de 2014.

A.1.2. Competición Círculo 2017

Se pueden encontrar los resultados, desglosados por niveles, de nuestro agente UCM Physics en la tabla A.7, de nuestro agente UCM QLearning en la tabla A.8, del agente MARL-GF Agent en la tabla A.9, del agente AGAgent en la tabla A.10, del jugador humano medio en la tabla A.11 y del agente óptimo en la tabla A.12. Todos ellos hacen referencia a la competición del círculo con los niveles del año 2017.

MARL-GF Agent - Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	8	797
2	3	3	45	17	922
3	3	2,8	60	21,4	924
4	4	4	80	20	1150
5	2	2	70	23,1	870
6	2	1,8	40	16,8	760
7	3	3	60	16	1033
8	3	3	40	27,3	618
9	3	3	80	27,6	955
10	3	3	100	35,1	949
			Puntuación total		8978

Tabla A.3: Resultados del círculo del agente MARL-GF Agent en la competición de 2014.

AGAgent - Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	8,7	763
2	3	3	45	20,7	840
3	3	0,8	60	60	80
4	4	4	80	38,8	915
5	2	2	70	28,6	792
6	2	2	40	12,5	887
7	3	2,6	60	48,9	446
8	3	3	40	32,2	494
9	3	3	80	33,9	876
10	3	2,9	100	64	650
			Puntuación total		6743

Tabla A.4: Resultados del círculo del agente AGAgent en la competición de 2014.

A.1.3. Competición Círculo 2022

Para concluir con el círculo, los resultados obtenidos por los agentes en la competición del año 2022, desglosados por niveles, se pueden encontrar en la tabla A.13 para nuestro agente UCM Physics, en la tabla A.14 para nuestro agente UCM QLearning , en la tabla A.15 para el agente MARL-GF Agent, en la tabla A.16 para el agente AGAgent, en la tabla A.17 para el jugador humano y en la tabla A.18 para el agente óptimo.

Humano medio - Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2,1	20	5,3	941
2	3	2,9	45	19,3	865
3	3	2,6	60	39,3	608
4	4	4,0	80	23,9	1102
5	2	2,0	70	23,4	866
6	2	2,0	40	10,1	947
7	3	2,4	60	42,4	536
8	3	2,9	40	26,4	628
9	3	3,0	80	28,6	943
10	3	2,3	100	67,4	551
			Puntuación total		7987

Tabla A.5: Resultados del jugador humano medio en la competición de 2014 del círculo.

Óptimo - Circle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	4	1000
2	3	3	45	9	1100
3	3	3	60	9	1150
4	4	4	80	12	1250
5	2	2	70	15	986
6	2	2	40	7	1025
7	3	3	60	12	1100
8	3	3	40	23	725
9	3	3	80	22	1025
10	3	3	100	21	1090
			Puntuación total		10451

Tabla A.6: Resultados del círculo del agente óptimo en la competición de 2014.

A.2. Resultados de los agentes rectángulo

En esta sección se presentan las tablas con los resultados de las tres competiciones de los agentes rectángulo. Estas toman sus niveles de las competiciones oficiales de los años 2014, 2016 y 2022. Los agentes de los que ofrecemos información son UCM Physics, NKUST, Sub Goal A Star, el humano medio y el agente óptimo. La información sobre los ganadores de las competiciones oficiales de esos años se encuentra en las páginas webs `http://gaips.inesc-id.pt/` y `https://geometryfriends.gaips.inesc-id.pt`.

UCM Physics - Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	120	23,5	461
2	2	2	120	7,4	388
3	3	3	120	16,2	473
4	3	3	120	19,1	468
5	3	3	120	12,9	478
6	3	3	120	8	487
7	2	2	120	21,7	364
8	3	3	120	41,4	431
9	1	1	120	9,4	284
10	3	3	120	15,1	475
			Puntuación total		4309

Tabla A.7: Resultados del círculo del agente UCM Physics en la competición de 2017.

UCM QLearning - Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	120	28,3	453
2	2	2	120	7,8	387
3	3	3	120	16,4	473
4	3	3	120	19,5	468
5	3	3	120	17,1	472
6	3	3	120	13	478
7	2	1	120	120	100
8	3	3	120	32,7	446
9	1	1	120	13,8	277
10	3	2,1	120	119,6	211
			Puntuación total		3765

Tabla A.8: Resultados del círculo del agente UCM QLearning en la competición de 2017.

A.2.1. Competición Rectángulo 2014

En relación con la competición que toma sus niveles de la oficial del año 2014, se pueden encontrar los resultados de nuestro agente UCM Physics en la tabla A.19, del agente NKUST en la tabla A.20 y del agente Sub Goal A Star en la tabla A.21. Asimismo, las tablas A.22 y A.23 recogen los resultados obtenidos por un jugador humano medio y los resultados óptimos, respectivamente.

MARL-GF Agent - Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	120	30,9	448
2	2	1,9	120	20	357
3	3	3	120	16,8	472
4	3	3	120	21,4	464
5	3	3	120	17,5	471
6	3	3	120	9,4	484
7	2	2	120	27,7	354
8	3	3	120	15	475
9	1	1	120	7,8	287
10	3	2,4	120	79,7	307
			Puntuación total		4119

Tabla A.9: Resultados del círculo del agente MARL-GF Agent en la competición de 2017.

AGAgent - Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	120	27,3	454
2	2	1,6	120	54,2	270
3	3	3	120	18,1	470
4	3	3	120	25,5	457
5	3	3	120	21	465
6	3	3	120	13,3	478
7	2	1,9	120	46,5	312
8	3	3	120	30	450
9	1	1	120	9,6	284
10	3	3	120	39,1	435
			Puntuación total		4075

Tabla A.10: Resultados del círculo del agente AGAgent en la competición de 2017.

A.2.2. Competición Rectángulo 2016

Pasmos ahora a la competición asociada al año 2016, de la cual se pueden encontrar los resultados desglosados por niveles de nuestro agente UCM Physics en la tabla A.24, del agente NKUST en la tabla A.25, del agente Sub Goal A Star en la tabla A.26, del jugador humano medio en la tabla A.27 y del agente óptimo en la tabla A.28.

Humano medio - Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3,0	120	28,4	453
2	2	1,5	120	64,0	243
3	3	2,9	120	22,0	457
4	3	2,9	120	31,6	435
5	3	3,0	120	11,4	481
6	3	3,0	120	12,2	480
7	2	2,0	120	25,9	357
8	3	3,0	120	30,6	449
9	1	1,0	120	13,1	278
10	3	3,0	120	22,4	463
			Puntuación total		4095

Tabla A.11: Resultados del jugador humano medio en la competición de 2017 del círculo.

Óptimo-Circle 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	120	19	468
2	2	2	120	6	390
3	3	3	120	14	477
4	3	3	120	14	477
5	3	3	120	9	485
6	3	3	120	8	487
7	2	2	120	18	370
8	3	3	120	6	490
9	1	1	120	3	295
10	3	3	120	8	487
			Puntuación total		4425

Tabla A.12: Resultados del círculo óptimo en la competición de 2017.

A.2.3. Competición Rectángulo 2022

Para finalizar con las competiciones del rectángulo, los resultados obtenidos por los agentes en la competición del año 2022, desglosados por niveles, se pueden encontrar en la tabla A.29 para nuestro agente UCM Physics, en la tabla A.30 para el agente NKUST, en la tabla A.31 para el agente AGAgent, en la tabla A.32 para el jugador humano medio y en la tabla A.33 para el agente óptimo.

UCM Physics - GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	5	120	20	2333
2	2	2	120	8,4	1530
3	4	4	120	15,4	2071
4	2	2	120	16,6	1461
5	2	2	120	11,7	1502
6	4	4	120	16,5	2062
7	5	5	120	41,2	2157
8	5	5	120	19,7	2335
9	5	5	120	31	2241
10	3	1,6	120	108,7	574
			Puntuación total		18266

Tabla A.13: Resultados del círculo del agente UCM Physics en la competición GF-CoG 2022.

UCM QLearning - GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	4,1	120	95,2	1437
2	2	2	120	14	1483
3	4	3,4	120	36,5	1716
4	2	1,4	120	101,6	573
5	2	0,1	120	120	30
6	4	4	120	21,8	2018
7	5	4,7	120	61,2	1900
8	5	4,5	120	85,1	1641
9	5	5	120	43,5	2138
10	3	1,1	120	120	330
			Puntuación total		13266

Tabla A.14: Resultados del círculo del agente UCM QLearning en la competición GF-CoG 2022.

A.3. Resultados de los agentes cooperativos

En esta última sección del apéndice se muestran las tablas con las puntuaciones de las tres competiciones cooperativas que hemos celebrado. Estas toman sus niveles de las competiciones oficiales de los años 2013, 2017 y 2022. Las parejas de agentes de las que ofrecemos los resultados son UCM Physics, MARL-GF Agent y NKUST. La información sobre los ganadores de las competiciones oficiales de esos años se puede buscar en las páginas web oficiales de la competición: `http://gaips.inesc-id.pt/` y `https://geometryfriends.gaips.inesc-id.pt`.

MARL-GF Agent - GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	5	120	17,1	2357
2	2	2	120	8,5	1529
3	4	1	120	120	300
4	2	2	120	20,1	1433
5	2	1,8	120	22,7	1351
6	4	3,8	120	37,7	1825
7	5	5	120	33,2	2224
8	5	5	120	20,7	2327
9	5	5	120	69,9	1917
10	3	2,5	120	74,7	1127
			Puntuación total		16390

Tabla A.15: Resultados del círculo del agente MARL-GF Agent en la competición GF-CoG 2022.

AGAgent- GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	4,7	120	46,8	2020
2	2	2	120	11,9	1501
3	4	4	120	30,3	1948
4	2	2	120	19,4	1438
5	2	2	120	9,9	1518
6	4	3	120	72,2	1298
7	5	4,9	120	67,2	1910
8	5	5	120	33,8	2219
9	5	5	120	44,5	2129
10	3	2,3	120	100,3	854
			Puntuación total		16835

Tabla A.16: Resultados del círculo del agente AGAgent en la competición GF-CoG 2022.

A.3.1. Competición Cooperativa 2013

La primera competición de la cual ofrecemos resultados es la que toma sus niveles de los de la competición oficial del año 2013. Se puede encontrar la información de nuestros agentes UCM Physics en la tabla A.34, de los agentes MARL-GF Agent en la tabla A.35 y de los agentes NKUST en la tabla A.36.

Humano medio - GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	4,9	120	25,7	2267
2	2	2,0	120	15,1	1474
3	4	3,9	120	56,8	1690
4	2	1,9	120	25,2	1371
5	2	1,9	120	24,9	1374
6	4	3,4	120	55,8	1566
7	5	5,0	120	38,8	2177
8	5	5,0	120	49,2	2090
9	5	4,9	120	41,1	2139
10	3	2,4	120	90,8	974
			Puntuación total		17122

Tabla A.17: Resultados del jugador humano medio en la competición GF-CoG 2022 del círculo.

Óptimo - GF-CoG 2022 - Circle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	5	5	120	10	2417
2	2	2	120	6	1550
3	4	4	120	13	2092
4	2	2	120	13	1492
5	2	2	120	6	1550
6	4	4	120	12	2100
7	5	5	120	19	2342
8	5	5	120	16	2367
9	5	5	120	26	2283
10	3	3	120	18	1750
			Puntuación total		19942

Tabla A.18: Resultados del círculo del agente óptimo en la competición GF-CoG 2022.

A.3.2. Competición Cooperativa 2017

Para la competición cooperativa asociada al año 2017, se pueden encontrar los resultados desglosados por niveles de nuestros agentes UCM Physics en la tabla A.37, de los agentes MARL-GF Agent en la tabla A.38 y de los agentes NKUST en la tabla A.39.

UCM Physics - Rectangle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	40	12,7	882
2	2	2	25	7,2	911
3	3	3	80	25,4	983
4	2	2	20	11,2	639
5	5	5	90	35,4	1107
6	3	3	40	17,6	860
7	3	3	50	14,3	1014
8	3	3	60	27,3	845
9	3	3	35	12,2	951
10	3	2,7	35	25,5	541
			Puntuación total		8733

Tabla A.19: Resultados del rectángulo del agente UCM Physics en la competición de 2014.

NKUST - Rectangle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	40	11,2	920
2	2	2	25	6,5	941
3	3	2,1	80	76	260
4	2	1,9	20	8,9	746
5	5	5	90	55,8	880
6	3	3	40	18,7	832
7	3	2	50	50	200
8	3	3	60	23,4	910
9	3	3	35	10,7	994
10	3	2,4	35	26,8	474
			Puntuación total		7157

Tabla A.20: Resultados del rectángulo del agente NKUST en la competición de 2014.

A.3.3. Competición Cooperativa 2022

Para finalizar la presente sección y con ello el apéndice, presentamos los resultados obtenidos por los agentes en la competición cooperativa asociada al año 2022, desglosados por niveles. Se pueden encontrar en la tabla A.40 para nuestros agentes UCM Physics, en la tabla A.41 para los agentes MARL-GF Agent y en la tabla A.42 para los agentes NKUST.

Sub Goal A Star - Rectangle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	1,8	40	18,8	711
2	2	2	25	7	920
3	3	2	80	52,9	540
4	2	0	20	20	0
5	5	2,9	90	90	290
6	3	3	40	21,5	763
7	3	2	50	50	200
8	3	0	60	60	0
9	3	3	35	12,6	941
10	3	0	35	35	0
			Puntuación total		4365

Tabla A.21: Resultados del rectángulo del agente Sub Goal A Star en la competición de 2014.

Humano medio - Rectangle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	1,9	20	18,9	722
2	2	2,0	45	7,3	909
3	3	2,4	60	44,7	681
4	2	2,0	80	7,9	807
5	5	4,5	70	49,0	909
6	3	2,3	40	35,1	355
7	3	2,7	60	27,7	712
8	3	2,5	40	37,1	636
9	3	2,6	80	17,6	757
10	3	2,1	100	29,8	355
			Puntuación total		6842

Tabla A.22: Resultados del humano medio en la competición de 2014 del rectángulo.

Óptimo - Rectangle 2014 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	20	9	975
2	2	2	45	5	1000
3	3	3	60	14	1125
4	2	2	80	4	1000
5	5	5	70	20	1278
6	3	3	40	14	950
7	3	3	60	9	1120
8	3	3	40	12	1100
9	3	3	80	8	1071
10	3	3	100	16	843
			Puntuación total		10462

Tabla A.23: Resultados del agente rectángulo óptimo en la competición de 2014.

UCM Physics - Rectangle 2016 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	1,7	40	31,4	213
2	4	4	65	29	511
3	2	2	35	7,5	358
4	3	3	30	15	400
5	3	3	40	17,9	411
6	3	2	90	90	200
7	4	3,8	90	43,1	484
8	4	3	90	90	300
9	2	2	90	12,8	372
10	5	5	90	14,4	668
			Puntuación total		3917

Tabla A.24: Resultados del agente rectángulo UCM Physics en la competición de 2014.

NKUST - Rectangle 2016 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	1	40	40	100
2	4	4	65	27,8	514
3	2	0	35	35	0
4	3	3	30	8,1	446
5	3	2	40	40	200
6	3	0	90	90	0
7	4	4	90	24,6	545
8	4	3	90	90	300
9	2	2	90	9,8	378
10	5	5	90	20,6	654
			Puntuación total		3137

Tabla A.25: Resultados del rectángulo del agente NKUST en la competición de 2016.

Sub Goal A Star - Rectangle 2016 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	0	40	40	0
2	4	2	65	65	200
3	2	0	35	35	0
4	3	0	30	30	0
5	3	0	40	40	0
6	3	1	90	90	100
7	4	3	90	9	300
8	4	0	90	90	0
9	2	1	90	9	100
10	5	3	90	90	300
			Puntuación total		1000

Tabla A.26: Resultados del rectángulo del agente Sub Goal A Star en la competición de 2016.

Humano medio - Rectangle 2016 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2,0	40	18,4	308
2	4	3,9	65	33,0	484
3	2	1,6	35	20,0	243
4	3	2,7	30	16,0	365
5	3	2,4	40	31,5	278
6	3	2,8	90	47,5	373
7	4	3,8	90	42,3	485
8	4	3,4	90	55,8	412
9	2	2,0	90	10,9	376
10	5	5,0	90	11,0	676
			Puntuación total		3999

Tabla A.27: Resultados del humano medio en la competición de 2016 del rectángulo.

Óptimo - Rectangle 2016 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	40	10	350
2	4	4	65	16	551
3	2	2	35	3	383
4	3	3	30	3	480
5	3	3	40	7	465
6	3	3	90	7	484
7	4	4	90	15	567
8	4	4	90	12	573
9	2	2	90	5	389
10	5	5	90	4	691
			Puntuación total		4933

Tabla A.28: Resultados del agente rectángulo óptimo en la competición de 2016.

UCM Physics - GF-IJCAI-ECAI 2022 - Rectangle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	120	5,3	1556
2	3	3	120	41,5	1554
3	2	2	120	8,3	1530
4	4	3	120	120	900
5	2	2	120	12,6	1495
6	5	5	120	46,6	2111
7	1	1	60	2,2	1263
8	2	2	120	6,7	1545
9	6	5,2	120	53,6	2113
10	1	1	60	3,7	1239
			Puntuación total		15306

Tabla A.29: Resultados del agente rectángulo UCM Physics en la competición GF-IJCAI-ECAI 2022.

NKUST -GF-IJCAI-ECAI 2022 - Rectangle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	120	6,3	1547
2	3	2	120	120	600
3	2	0	120	120	0
4	4	1,2	120	117,8	378
5	2	2	120	12	1500
6	5	1	120	120	300
7	1	0	60	60	0
8	2	2	120	6,5	1546
9	6	0	120	48	0
10	1	1	60	4,3	1228
			Puntuación total		7099

Tabla A.30: Resultados del rectángulo del agente NKUST en la competición GF-IJCAI-ECAI 2022.

Sub Goal A Star -GF-IJCAI-ECAI 2022 - Rectangle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	120	5,9	1551
2	3	1,8	120	120	540
3	2	0	120	120	0
4	4	2,3	120	81,3	1012
5	2	0	120	120	0
6	5	1,8	120	120	540
7	1	0	60	60	0
8	2	2	120	7,3	1539
9	6	1,1	120	120	330
10	1	0	60	60	0
			Puntuación total		5512

Tabla A.31: Resultados del rectángulo del agente Sub Goal A Star en la competición GF-IJCAI-ECAI 2022.

Humano medio - GF-IJCAI-ECAI 2022 - Rectangle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	1,9	120	15,4	1450
2	3	3,0	120	18,8	1743
3	2	2,0	120	8,7	1527
4	4	3,8	120	39,9	1804
5	2	2,0	120	7,8	1535
6	5	4,6	120	56,9	1918
7	1	0,9	60	8,0	1145
8	2	2,0	120	7,9	1534
9	6	4,4	120	87,4	1579
10	1	1,0	60	7,4	1176
			Puntuación total		15413

Tabla A.32: Resultados del humano medio en la competición GF-IJCAI-ECAI 2022 del rectángulo.

Óptimo - GF-IJCAI-ECAI 2022 - Rectangle Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	2	2	120	4	1567
2	3	3	120	12	1800
3	2	2	120	5	1558
4	4	4	120	7	2142
5	2	2	120	5	1558
6	5	5	120	17	2358
7	1	1	60	1	1283
8	2	2	120	4	1567
9	6	6	120	15	2675
10	1	1	60	3	1250
			Puntuación total		17758

Tabla A.33: Resultados óptimos del rectángulo en la competición GF-IJCAI-ECAI 2022.

UCM Physics - Cooperative 2013 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	90	38,3	874
2	3	2,2	90	51,2	651
3	2	2	35	17,3	704
4	5	5	110	48,8	1056
5	4	4	100	32,1	1079
6	3	2	55	43,8	404
7	2	1,6	80	63,8	363
8	4	3,8	125	61,8	886
9	1	0,9	30	18,8	464
10	2	1,3	35	23,6	456
			Puntuación total		6937

Tabla A.34: Resultados de los agentes UCM Physics en la competición cooperativa del año 2013.

MARL-GF Agent - Cooperative 2013 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	0	90	90	0
2	3	1	90	90	100
3	2	2	35	16,9	717
4	5	0,5	110	110	50
5	4	0,4	100	100	40
6	3	0,8	55	55	80
7	2	0	80	80	0
8	4	0	125	125	0
9	1	0	30	30	0
10	2	1	35	35	100
			Puntuación total		1087

Tabla A.35: Resultados de los agentes MARL-GF Agent en la competición cooperativa del año 2013.

NKUST - Cooperative 2013 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	0	90	90	0
2	3	1	90	90	100
3	2	1	35	35	100
4	5	1,2	110	110	120
5	4	0	100	100	0
6	3	2	55	55	200
7	2	0	80	80	0
8	4	0	125	125	0
9	1	0	30	30	0
10	2	1,8	35	17,2	689
			Puntuación total		1209

Tabla A.36: Resultados de los agentes NKUST en la competición cooperativa del año 2013.

UCM Physics - Cooperative 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	300	29,6	480
2	1	1	300	14,1	291
3	2	1,9	300	104,2	320
4	3	3	300	31,4	479
5	3	3	300	8,5	494
6	3	3	240	27,2	477
7	2	0	240	240	0
8	3	2,9	240	51,8	447
9	2	2	240	7,4	394
10	2	1,9	300	54,1	354
			Puntuación total		3736

Tabla A.37: Resultados de los agentes UCM Physics en la competición cooperativa del año 2017.

MARL-GF Agent - Cooperative 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	3	300	29	481
2	1	1	300	14,2	290
3	2	1	300	300	100
4	3	3	300	31,8	479
5	3	2,8	300	67,6	435
6	3	3	240	20,3	483
7	2	0	240	240	0
8	3	3	240	29,4	476
9	2	1,9	240	29,4	365
10	2	1	300	300	100
			Puntuación total		3209

Tabla A.38: Resultados de los agentes MARL-GF Agent en la competición cooperativa del año 2017.

NKUST - Cooperative 2017 Competition					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	3	2	300	300	200
2	1	0	300	300	0
3	2	0	300	300	0
4	3	1	300	300	100
5	3	2,4	300	204,8	303
6	3	0	240	240	0
7	2	0	240	240	0
8	3	2	240	240	200
9	2	2	240	6	395
10	2	0,7	300	300	70
			Puntuación total		1268

Tabla A.39: Resultados de los agentes UCM Physics en la competición cooperativa del año 2017.

UCM Physics - GF-IJCAI-ECAI 2022 - Cooperation Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	1	1	120	31,3	1039
2	5	4,9	120	48,2	2068
3	1	0	120	120	0
4	3	3	120	38,8	1577
5	2	1,3	120	84	690
6	2	2	120	25,4	1389
7	3	3	120	37,9	1584
8	8	8	120	30,6	3145
9	4	4	120	13,6	2086
10	1	1	120	14,2	1182
			Puntuación total		14760

Tabla A.40: Resultados de los agentes UCM Physics en la competición cooperativa GF-IJCAI-ECAI 2022.

NKUST - GF-IJCAI-ECAI 2022 - Cooperation Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	1	0	120	120	0
2	5	2	120	120	600
3	1	0	120	120	0
4	3	2	120	120	600
5	2	0	120	120	0
6	2	1,4	120	55,6	957
7	3	0	120	120	0
8	8	5	120	120	1500
9	4	2	120	120	600
10	1	1	120	13	1191
			Puntuación total		5448

Tabla A.41: Resultados de los agentes MARL-GF Agent en la competición cooperativa GF-IJCAI-ECAI 2022.

NKUST - GF-IJCAI-ECAI 2022 - Cooperation Track					
Nivel	Diamantes disponibles	Promedio de diamantes alcanzados	Tiempo límite	Tiempo medio	Puntuación media
1	1	0	120	120	0
2	5	2,7	120	120	810
3	1	0	120	120	0
4	3	2	120	120	600
5	2	2	120	19,2	1040
6	2	1	120	120	300
7	3	2	120	120	600
8	8	2,8	120	120	840
9	4	3,5	120	66,3	1497
10	1	0	120	120	0
			Puntuación total		5687

Tabla A.42: Resultados de los agentes NKUST en la competición cooperativa GF-IJCAI-ECAI 2022.

Apéndice B

Pruebas con participantes humanos

En este apéndice, se pretende explicar el transcurso de las pruebas con jugadores humanos, que nos permitieron comparar el rendimiento de nuestros agentes individuales del círculo y el rectángulo con lo que definimos como humano medio. Nos abstuvimos de realizar las pruebas a humanos en modo cooperativo porque el problema logístico y organizativo que planteaba sobrepasaba nuestras capacidades. Comentaremos el proceso que hemos seguido para obtener la muestra de participantes y sus características, y explicaremos el guion sobre el desarrollo de las sesiones de evaluación.

B.1. Participantes

Los participantes que se prestaron a realizar estas pruebas fueron voluntarios que se encontraban dentro de nuestro círculo social universitario y personal. Conseguimos encontrar un total de 16 voluntarios para realizar las sesiones, de los que la gran mayoría fueron compañeros de nuestro mismo doble grado. Por tanto, el perfil de los participantes en estas pruebas es de personas de entre 20 y 25 años, a punto de finalizar sus estudios universitarios de grado, habituados al uso del ordenador y con un dominio de la tecnología. Elegimos mantener este estudio en un entorno controlado, motivo por el que solamente evaluamos a 16 personas, en el cual tuviéramos el control de las sesiones, pudiéramos detenernos a explicarles todos los detalles que necesitaran y pudiéramos asegurarnos que seguían todas las normas que les imponíamos.

B.2. Desarrollo de las sesiones

Hubo tres modalidades en las que se desarrollaron las sesiones de evaluación. La primera y más habitual fue la conexión remota entre el participante y uno de los autores, que a partir de ahora denotaremos por evaluadores. En dicha conexión, el participante y el evaluador se conectaron en alguna aplicación que permitiera realizar una videoconferencia y en la que el participante tuviera la capacidad de compartir su pantalla durante toda la sesión. La segunda modalidad fue presencial, en la que un participante y un evaluador realizaban las pruebas en el mismo lugar físico. En esta modalidad, el evaluador podía ver la pantalla en la que el participante jugaba durante toda la sesión. En ambos casos, el máximo de personas a las que evaluábamos simultáneamente fue uno, es decir, las sesiones de evaluación eran individuales para que los participantes contaran con toda nuestra atención y disponibilidad. Finalmente, el tercer modo de evaluación y más infrecuente (tan solo se hizo con dos personas de nuestra total confianza), fue que el participante realizara las sesiones solo, tras haberle facilitado un documento explicativo muy detallado, y que anotara sus resultados en un cuestionario de Google Forms. Las sesiones de evaluación tenían un tiempo previsto de duración de 40 minutos para cada uno de los personajes (el círculo y el rectángulo). Hubo algunos participantes que prefirieron realizar las dos sesiones seguidas, mientras que otros prefirieron realizarlas en días distintos, por motivos de disponibilidad. En lo que sigue, explicaremos el desarrollo de las pruebas en las que un evaluador estaba presente, ya fuera de forma presencial o remota, y para las pruebas que los participantes realizaron solos, se puede encontrar el documento explicativo que les proporcionamos para la correcta realización de las pruebas en el Apéndice C.

Para el desarrollo de las sesiones, el evaluador enviaba un correo electrónico al participante con este archivo .zip en el que se encuentra el ejecutable del juego. Dicho ejecutable solo es válido para el sistema operativo Windows, lo que también fue un motivo por el que el número de participantes tan solo alcanzara las 16 personas. Una vez descargado y descomprimido el archivo .zip, los participantes ejecutaron el juego y los evaluadores les guiamos por las pantallas de inicio y de menú principal.

Para cada agente, indicábamos a los participantes que durante los primeros 20 minutos debían completar unos niveles de entrenamiento (los de los mundos identificados con la palabra *Training*) para habituarse a los controles. Después, realizarían la evaluación en los siguientes 20 minutos. Les dimos un contexto del juego, en el que un círculo y un cuadrado debían conseguir el máximo número de diamantes en el mínimo tiempo posible y ellos eran los encargados de controlar a los personajes.

En el entrenamiento del círculo, les explicamos que los controles para el círculo eran las teclas `A`, para que ruede hacia la izquierda, la tecla `D`, para que lo haga a la derecha y la tecla `W`, para que el círculo salte cuando esté apoyado en una plataforma. Les advertimos que no utilizaran la tecla `S`, que permite al círculo aumentar su radio, porque es una acción relativamente reciente, que la mayoría de agentes a lo

largo de los años no tuvieron en cuenta, que nosotros mismos hemos descartado y que no es necesaria para completar ninguno de los niveles. Con esto, pretendíamos que los jugadores humanos no tuvieran una potencial ventaja competitiva respecto a los agentes, que no utilizan dicha acción. Durante el proceso de entrenamiento del círculo, animábamos a los jugadores a experimentar movimientos arriesgados, a estimar cuánto saltaba el círculo, a tomar acciones mientras el círculo estaba en el aire, a tener presentes las colisiones con los obstáculos, las paredes y el techo y a no frustrarse, ya que, a pesar de su aparente simplicidad, el entorno dinámico donde se desarrolla la acción hace que los movimientos deban tener una gran precisión y coordinación. También hacíamos que experimentaran con las plataformas amarillas y verdes. Durante el entrenamiento, pudieron repetir cada nivel en todas las ocasiones que consideraron oportunas, teniendo en mente que el tiempo recomendado era de 20 minutos.

Una vez finalizado el entrenamiento del círculo, pasamos a la evaluación en las competiciones Circle2014, Circle2017 y Circle2022.1. Antes de empezar, se informó a los participantes de que, al contrario que en los niveles de entrenamiento, durante la evaluación solamente tendrían un intento para superar cada nivel. Esta decisión se tomó en el diseño de las pruebas, en contraposición con el sistema para los agentes que hace la media entre 10 ejecuciones de cada nivel, por tres motivos. Por un lado, los jugadores humanos podrían aprender de una ejecución a otra, mientras que en las competiciones oficiales del juego Geometry Friends, una vez se realiza la entrega, el agente no puede modificar su comportamiento de una ejecución a otra ni puede aprender durante el proceso de evaluación. Por otro lado, el motivo por el que se realizan 10 ejecuciones para los agentes en las competiciones oficiales es para amortiguar los efectos de la aleatoriedad de algunas implementaciones y creemos que el azar se verá amortiguado en el caso de los humanos al considerar múltiples participantes. El último motivo es logístico: la evaluación, tal y como está planteada, dura 20 minutos y si hubiera que repetir todos los niveles 10 veces sería imposible encontrar voluntarios para las pruebas. Con esta decisión pretendemos que las comparaciones entre los agentes y los jugadores humanos sean lo más justas posibles y, siguiendo con esa filosofía, al igual que los agentes disponen de todo el tiempo que deseen para analizar los niveles antes de comenzar la ejecución y que se inicie el cronómetro, a los jugadores humanos les dimos las mismas facilidades. Durante la evaluación, antes de iniciar cada nivel les informábamos de cuál era el límite de tiempo del nivel y, al finalizar el nivel, apuntábamos los resultados.

En el entrenamiento del rectángulo, les explicamos que los controles para el rectángulo eran la tecla `J`, para que se deslice hacia la izquierda, la tecla `L`, para que lo haga a la derecha, y las teclas `I` y `K` para que el rectángulo crezca o decrezca, manteniendo su área constante. Durante el proceso de entrenamiento, animábamos a los jugadores a experimentar movimientos imaginativos, a tomar acciones mientras el rectángulo estaba en el aire, a tener presentes las colisiones con los obstáculos, las paredes y el techo y a comprender el significado de las plataformas amarillas y verdes. Asimismo, les explicamos que el juego presentaba *bugs* para el rectángulo. Les describimos y ejemplificamos las situaciones en las que más probable era que se

llegara a este tipo de estados y les advertimos que no les recomendábamos emplear los *bugs* para intentar completar los niveles de forma más rápida, porque, por nuestra experiencia, eran más las veces que los *bugs* perjudicaban que las que favorecían. Con todo, les dimos total libertad para utilizarlos si así lo deseaban y les informamos que para calcular las puntuaciones óptimas, el jugador experto había hecho uso de estos *bugs*. Para concluir con los *bugs*, también les advertimos que si durante la evaluación se producía alguno, que intentaran continuar como si se tratara de parte del juego, ya que este mismo criterio es el que habíamos establecido para los demás agentes y permitirles repetir el nivel, daría una ventaja competitiva a los jugadores humanos. Los jugadores humanos pudieron comprobar que había ocasiones en las que el rectángulo quedaba atrapado entre dos plataformas y no podía realizar ningún movimiento para salir de ese estado. Les informamos de que eso también les sucedía al resto de agentes y que era su responsabilidad, al igual que la de los agentes, no llegar a ese tipo de situaciones. Como en el caso del círculo, les animamos a no desesperar, ya que, a pesar de su aparente sencillez, el juego Geometry Friends es mucho más complicado de lo que parece, como coincidieron todos los participantes una vez finalizadas las sesiones. Durante el entrenamiento, pudieron repetir cada nivel en todas las ocasiones que consideraron oportunas, teniendo en mente que el tiempo recomendado era de 20 minutos.

Una vez finalizado el entrenamiento del rectángulo, pasamos a la evaluación en las competiciones Rectangle2014, Rectangle2016 y Rectangle2022.2. Al igual que en el caso del círculo, antes de empezar se informó a los participantes de que, durante la evaluación, solamente tendrían un intento para superar cada nivel y que disponían de todo el tiempo que necesitaran para analizar cada nivel antes de comenzar la ejecución. Por último, durante la evaluación, se informaba a los participantes de cuál era el límite de tiempo del nivel que iban a comenzar y, al finalizarlo, los evaluadores apuntaban los resultados.

Apéndice C

Instrucciones para las evaluaciones humanas no supervisadas

Introducción

Hola, somos Alberto Almagro y Juan Carlos Llamas, del Doble Grado en Ingeniería Informática y Matemáticas, y necesitamos de tu colaboración para ayudarnos con nuestro TFG. No te preocupes, lo único que vas a tener que hacer es... ¡jugar!

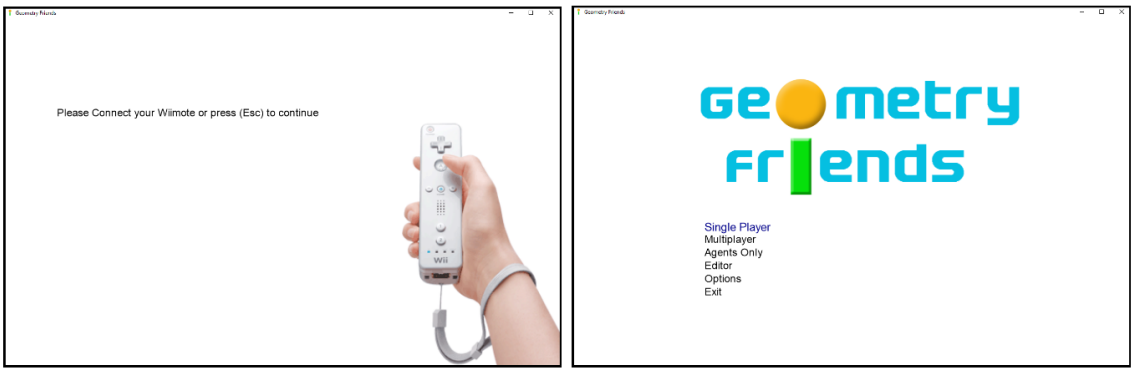
Como parte de nuestro TFG, estamos desarrollando unos agentes para el juego Geometry Friends. Este es un juego de plataformas en el que participan dos personajes (un rectángulo verde y un círculo amarillo), y su objetivo es alcanzar el máximo número de diamantes (rombos morados) en el menor tiempo posible. Existen tres modos de juego: modo rectángulo, modo círculo y modo cooperación. Te pedimos que juegues en los modos rectángulo y círculo, para así comparar tus resultados frente a los de nuestros agentes.

El tiempo aproximado que tendrás que dedicarle es de aproximadamente 40 minutos por agente. Puedes hacer cada agente en sesiones diferentes si quieres, pero una vez empieces la evaluación de uno de ellos, deberás terminarla.

Material necesario y pasos para la ejecución

Una vez hayas descargado y descomprimido este archivo .zip, debes entrar en la carpeta GeometryFriendsGame y después en la carpeta Release. Allí encontrarás, entre otros archivos, el ejecutable GeometryFriends.exe (necesitarás Windows para

ejecutarlo). Al iniciarlo, aparecerá una pantalla como la de la figura C.1a. Al pulsar en la tecla Esc, se abrirá el menú principal de la figura C.1b.



(a) Pantalla de inicio. (b) Menú principal.

Figura C.1: Primeras pantallas al iniciar el juego.

Puedes navegar por los menús con las flechas del teclado y utilizar el intro o la barra espaciadora para acceder a lo seleccionado. El elemento que esté seleccionado en la interfaz aparecerá coloreado de rojo, como se observa en la Figura C.2. Para jugar, debes entrar siempre en la opción *Multiplayer*. En ella, encontraras diferentes mundos (compuestos por 10 niveles) en los que tendrás que jugar. Por favor, **NO juegues en aquellos mundos que NO contengan la palabra *Training* hasta que te lo indiquemos**. Es decir, de momento solo puedes practicar en los mundos que contengan la palabra *Training*.

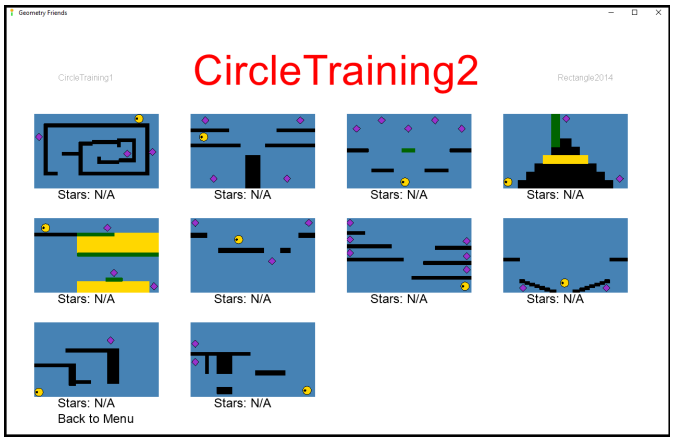


Figura C.2: Mundo CircleTraining2.

Orientaciones y procedimiento a seguir

Para cada agente, vas a necesitar completar unos niveles de entrenamiento (los de los mundos con *Training*) para hacerte con los controles, y después realizar una evaluación en la que solo tendrás un intento por nivel, debiendo apuntar tus resultados en una encuesta de Google Forms (una por personaje).

Te advertimos de que los controles deben ser muy precisos y pueden llegar a resultar algo frustrantes. Es normal. Aunque parezca inocente, el juego es realmente complicado y requiere de una gran precisión y coordinación. No hace falta que completes todos los niveles de entrenamiento, aunque te recomendamos que lo intentes. Aún así, algunos son muy difíciles y, si ves que no consigues superarlos, puedes saltártelos. Deberías emplear **al menos 20 minutos por personaje** entrenando para poder manejarte con soltura.

De cara a la prueba, es muy importante que sigas estas instrucciones al pie de la letra, ya que queremos garantizar una comparación justa entre los agentes y los jugadores humanos.

Cada nivel cuenta con un número de diamantes que hay que alcanzar y un tiempo límite para hacerlo. Tu objetivo es alcanzar todos los diamantes en el menor tiempo posible, siempre antes de llegar al tiempo límite. Puedes tomarte todo el tiempo que necesites para analizar y planificar antes de darle a Start, que es cuando empieza el tiempo.

Uno de los inconvenientes es que el juego no cuenta con un sistema para finalizar el nivel cuando el tiempo límite expira. Sí que podrás ver un contador con el tiempo empleado hasta el momento en la esquina superior derecha. Por tanto, antes de comenzar el nivel, deberás ver su tiempo límite asociado (en los niveles de entrenamiento no es necesario que hagas esto) y asegurarte de parar el juego cuando transcurra dicho tiempo. Todos los diamantes que recojas una vez que ha expirado el tiempo límite no debes contarlos como alcanzados. De todas formas, el tiempo ser más que suficiente.

Otro aspecto de gran importancia es que **debes apuntar los resultados de tu primera ejecución en cada nivel**. Debe ser la primera y no la segunda, la tercera o aquella en la que hayas realizado un mejor tiempo o hayas cogido un mayor número de diamantes. Esto es muy importante y te pedimos que lo respetes. Somos conscientes de que puede resultar frustrante tomar una acción de manera errónea o que sea incluso el juego el que pueda tener algún tipo de problema, pero para que la comparación con los agentes sea justa, insistimos, por favor, y bajo cualquier circunstancia, apunta los datos de la primera ejecución. Si cuando finalice el nivel quieres reiniciarlo porque crees que podrías haberlo hecho mejor, eres totalmente libre de hacerlo, pero de nuevo, apunta únicamente los datos de la primera ejecución. Además, en la encuesta de Google Forms, al final de cada mundo, tienes un cuadro de texto donde nos puedes contar tus impresiones sobre si ha surgido algún problema e indicándonos cualquier información relevante. No vamos a publicar los resultados de ninguna forma que te identifique, así que nadie se va a enterar de que has fallado el nivel X.

Hay ocasiones en las que el personaje con el que estés puede quedar atrapado en una estructura y que parezca que ya no se va a poder completar el nivel. ¡Te animamos a seguir intentándolo hasta llegar al tiempo límite! A veces puede sorprender cómo

se puede salir de ciertos lugares. Sin embargo, siempre tienes la opción de rendirte antes de tiempo, en cuyo caso puedes pausar el juego, apuntar los diamantes que has logrado y debes apuntar como tiempo empleado el tiempo límite del nivel. Para pasar al siguiente nivel deberás salir al menú principal, seleccionar la opción Multiplayer y navegar con las flechas hasta él.

En resumen, **el procedimiento que debes seguir en cada nivel es**:

1) Mirar en el Google Forms correspondiente cuál es el tiempo límite del nivel.

2) Analizar el nivel todo el tiempo que necesites hasta que tengas una estrategia.

3) Comenzar el nivel pulsando en Start (con enter o espacio).

4) Jugar hasta que el nivel finalice porque:

   a) El tiempo llega al límite del nivel. Entonces pausa el juego, apunta en el cuestionario de Google Forms los diamantes capturados hasta el momento en el que expira el tiempo límite y apunta también tu tiempo empleado, que es el tiempo límite del nivel.

   b) Te rindes. Entonces pausa el nivel, apunta en el cuestionario de Google Forms los diamantes capturados hasta el momento y tu tiempo empleado, que es el tiempo límite del nivel (no el tiempo en el que te has rendido).

   c) Has logrado capturar todos los diamantes un tiempo menor que el tiempo límite del nivel. Entonces el juego se detendrá automáticamente y te mostrará una pantalla con el tiempo que has necesitado. Apunta en el cuestionario de Google Forms el número de diamantes capturados (todos) y el tiempo empleado (asegúrate que es menor que el tiempo límite del nivel).

5) Puedes pasar al siguiente nivel seleccionando Next level o Last level si conseguiste superar el nivel o saliendo al menú principal y volviendo a entrar.

Instrucciones para el círculo

Introducción al personaje

En primer lugar, te vamos a pedir que juegues en los niveles del círculo. Antes de empezar la prueba, recomendamos leer esta introducción para maximizar tus resultados. Sería conveniente que todo lo que vayas leyendo lo pruebes en los niveles

que tenemos disponibles para practicar. Contamos con dos mundos (CircleTraining1 y CircleTraning2), en los que podrás jugar tantas vez como quieras hasta que te sientas cómodo con los comandos.

El círculo es una pelota amarilla que se controla con las teclas A, W y D. La tecla A, sirve para hacer que el círculo gire hacia la izquierda, la tecla D, para que lo haga a la derecha y la tecla W para que el círculo salte cuando esté apoyado en una plataforma. Te recomendamos que prestes atención al efecto de cada acción cuando el círculo está en el aire. ¡Siéntete libre para probar todo lo que se te ocurra en los niveles de entrenamiento!

Cuando un nivel se vuelve imposible de resolver, el juego no tiene un límite de tiempo tras el cual te expulse. Para pausar el juego, puedes pulsar la tecla Esc. Se desplegará un menú como el de la Figura C.3 en el que podrás reiniciar el nivel o volver al menú principal.

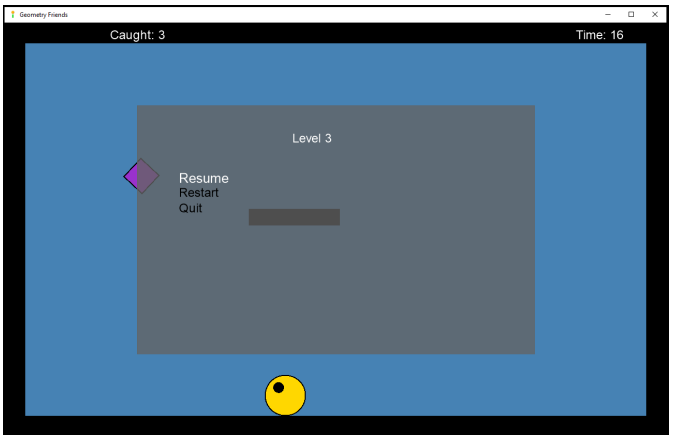


Figura C.3: Mundo CircleTraining2.

La tecla S realiza la acción de aumentar el radio del círculo. Es una novedad relativamente reciente y no es necesaria para completar ninguno de los niveles, así que no tienes por qué utilizarla durante la prueba si no quieres. Te lo comentamos solo para que no te sorprendas si por error la pulsas.

Instrucciones para la prueba

Deberás apuntar tus resultados en la siguiente encuesta de Google Forms.

Cuando estés listo, completa los mundos Circle2014, Circle2017 y Circle2022.1. ¡Ya puedes empezar! ¡Suerte y diviértete!

Instrucciones para el rectángulo

Introducción al personaje

Ahora, te vamos a pedir que juegues a los niveles del rectángulo. De nuevo, recomendamos leer está introducción para poder maximizar tus resultados. Puedes probar todo lo que vayas leyendo lo pruebes en los niveles que tenemos disponibles para practicar. Contamos con dos mundos (RectangleTraining1 y RectangleTraning2), en los que podrás jugar tantas vez como quieras hasta que te sientas cómodo con los comandos.

El rectángulo es un personaje verde que se controla con las teclas I, J, K y L. La tecla J, sirve para hacer que el rectángulo se desplace hacia la izquierda y la tecla L, para que lo haga a la derecha. La tecla I permite que el rectángulo crezca verticalmente y se haga más estrecho. La tecla K hace que el rectángulo decrezca verticalmente y que se ensanche. Te recomendamos que prestes atención al efecto de cada acción cuando el rectángulo está en el aire, ya que es diferente al círculo. ¡Siéntete libre para probar todo lo que se te ocurra en los niveles de entrenamiento!

Como último comentario, sentimos comunicarte que el juego tiene algunos *bugs* (que escapan a nuestro control). En primer lugar, la física de las colisiones no siempre actúa como debería. Puedes comprobarlo, por ejemplo, en el nivel 4 del mundo RectangleTraining1. Si, durante la caída para coger los dos diamantes de la parte inferior del nivel, pulsas la tecla K para que el rectángulo se ensanche, el rectángulo atraviesa las paredes y queda atrapado. Si reinicias el nivel y, en la plataforma de partida, activas la tecla I, para crecer verticalmente, comprobaras que el comportamiento del rectángulo es totalmente descontrolado. Es más probable que utilizalos estos *bugs* te perjudiquen, y te recomendamos tener precaución a la hora de tomar las acciones I y K por el comportamiento impredecible que surge cuando el rectángulo intenta crecer en un lugar en el que no tiene suficiente espacio para hacerlo.

El segundo *bug* es todavía más desesperante. En algunas ocasiones, cuando el rectángulo cae, se bloquea la posibilidad de que cambie de forma. Esto es, al pulsar las teclas I y K, el rectángulo no hace nada. Esto hace que algunos niveles puedan convertirse en irresolubles. Es un problema del juego que no podemos solucionar.

Pedimos disculpas por estos *bugs*, que, aunque no podamos subsanarlos porque nosotros no hemos diseñado el juego, somos conscientes de que resultan muy molestos.

Instrucciones para la prueba

Deberás apuntar tus resultados en la siguiente encuesta de Google Forms.

Recuerda que **debes apuntar los resultados de tu primera ejecución en cada nivel**. En el caso del rectángulo puede ser tentador repetir el nivel excusándose en que ha habido algún *bug*. Repítelo si quieres, pero por favor, apunta el resultado de tu primera ejecución. Tienes que ser consciente de que a los agentes también les afectan estos *bugs*, y si te permitiéramos escoger los resultados de tu segunda ejecución, la comparación no sería justa.

Cuando estés listo, completa los mundos Rectangle2014, Rectangle2016 y Rectangle2022.2. ¡Ya puedes empezar! ¡Suerte y diviértete!