Estudio sobre el uso de IAs basadas en comportamiento en *Pac-Man*

Raúl Ramos Macias

Resumen– En este trabajo se ha implementado un *remake* del videojuego *Pac-Man* original y se ha diseñado un fantasma nuevo que utiliza en su estrategia particularidades de un tipo de Inteligencia Artificial llamada 'Inteligencia Artificial basada en comportamiento', caracterizada por ser adaptable y utilizar elementos como el entorno, la memoria, el aprendizaje y la coordinación con otras IA. Una vez finalizada la implementación, se han realizado pruebas con 21 usuarios reales, analizando datos percibidos y datos objetivos numéricos, y se ha concluido que este tipo de IA mejora la experiencia del jugador y que existen relaciones entre la complejidad de la IA, la dificultad que esta aporta al juego y el disfrute resultante que se experimenta.

Palabras clave– Inteligencia Artificial, IA, Basada en comportamiento, Videojuegos, *Pac-Man*

Abstract– In this project we implemented a remake of the original *Pac-Man* videogame, including a new ghost that uses in its strategy characteristics of a type of Artificial Intelligence named 'Behavior based Artificial Intelligence'. This type of IA has a high situational adaptability, and uses technical resources like surroundings, memory analysis, machine learning and coordination with other AIs. Once the implementation was finished, it was tested with 21 real users, analyzing perceptual and gameplay data, and we concluded that this type of AI improves the player experience, and that there is a connection between AI complexity, the difficulty it contributes to the game and the resulting enjoyment.

Keywords– Artificial Intelligence, AI, Behavior based, Videogames, *Pac-Man*

1 Introducción

La **inteligencia Artificial** (o IA) es aquella disciplina científica que se ocupa de crear programas informáticos capaces de ejecutar operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.

En el ámbito de los **videojuegos**, el término IA, como sustantivo, se utiliza para referirse a entidades concretas del juego que tienen un comportamiento inteligente y que va dirigido a una función específica - tanto NPCs (personajes no controlados por el jugador) tangibles como adversarios intangibles en, por ejemplo, un ajedrez virtual.
A su vez, se puede utilizar el término en singular para referirse de forma mas genérica al conjunto de Inteligencias Artificiales que forman parte de él - así pues, un juego con 'una buena IA' es aquél donde todas las entidades que utilizan técnicas de Inteligencia Artificial cumplen con su función satisfactoriamente.

Las entidades con Inteligencia Artificial pueden diferenciarse en dos tipos según el planteamiento de su estrategia: **IA de conocimiento** e **IA de comportamiento**. Sus diferencias se pueden apreciar en la Tabla 1.

Las IA de conocimiento en el mundo de los videojuegos son **un recurso en auge y continuo desarrollo**, propiciada por la necesidad de crear Inteligencias Artificiales que simulen la improvisación natural, estrategia y acciones reflejas de un jugador humano, que sean capaces de procesar su comportamiento rápidamente a fin de retrasar lo mínimo posible una partida en linea o que sean capaces de reaccionar de forma satisfactoria a multitud de escenarios no previstos.

El **propósito del proyecto** es evaluar la influencia real de la aplicación de IAs de comportamiento en los juegos.

E-mail de contacte: amphiros@gmail.com
Menció realitzada: Enginyeria del Software
Treball tutoritzat per: Maria Vanrell Martorell (Departament de Ciències de la Computació)

Tipo de IA	
Conocimiento	**Comportamiento**
Utiliza solo datos imprescindibles para su estrategia (*input* de usuario y recursos propios)	Utiliza más datos para su estrategia (entorno, aprendizaje, memoria, otras IA...)
Más óptima - Orientada a la especialización: Conoce muchas técnicas concretas con anterioridad y escoge la mejor para una situación dada.	Mas adaptable - Orientada al dinamismo. Analiza la situación actual rápidamente e improvisa utilizando una base de técnicas reducida.
Propia de entornos cerrados y controlados.	Propia de entornos abiertos y caóticos.
Generalmente más lenta.	Generalmente más rápida.
Más predecible.	Más impredecible.
Ejemplo: Adversario en ajedrez virtual (donde prima la efectividad).	Ejemplo: IA en juego en linea de acción (donde prima la toma rápida de decisiones).

Tabla 1: Diferencias entre tipos de IA

Para ello, se aplicará en un juego clásico como *Pac-Man*.

Así pues, el objetivo del proyecto es triple:
1. **Desarrollar una nueva implementación del juego original** de *Pac-Man*.
2. **Aplicar características de las IA basadas en comportamiento** en un juego clásico como este.
3. **Evaluar las diferencias respecto al original** basándose en datos numéricos objetivos y datos percibidos de los usuarios.

2 *Pac-Man*, UN CLÁSICO INNOVADOR

Pac-Man, tradicionalmente conocido como 'Comecocos' en España, es un videojuego Arcade [1] creado por el diseñador de videojuegos Toru Iwatani para la empresa nipona Namco, y fue publicado el 21 de mayo de 1980. Consiguiendo el récord Guinness al videojuego arcade con más máquinas recreativas vendidas, se ha convertido en un símbolo de los videojuegos.

2.1. Objetivos, entidades y reglas

El **objetivo del juego** consiste en mover el epónimo *Pac-Man* por un laberinto, comiéndose unos puntos amarillos (o *dots*) al pasar por encima de ellos y completando el nivel cuando no quede ninguno.
Sin embargo, en el laberinto se encuentran también cuatro **fantasmas**, que salen de un habitáculo cerrado - *House* - y tratan de darle caza. Si los toca, el jugador perderá una vida y tendrá que comenzar el nivel actual de nuevo, con los fantasmas encerrados de nuevo y sin restaurar los *dots* consumidos. Si el jugador pierde todas las vidas, la partida acaba.

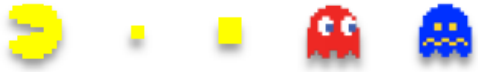


Figura 1: *Sprite* (imagen) de *Pac-Man*, un *dot*, un *Power Pellet*, un fantasma y un fantasma en modo *Fright*

Como añadido, existen unos *dots* especialmente grandes llamados ***Power Pellet***, que al ser recogidos provoca que los fantasmas sean vulnerables a *Pac-Man* durante unos segundos, tras los cuales volverán a la normalidad. Este estado de vulnerabilidad se conoce como *Fright*, y se representa con un cambio de color de los fantasmas a azul oscuro y un cambio en la expresión facial. Si *Pac-Man* los toca durante este estado, serán comidos, volverán a *House* y volverán a la normalidad.

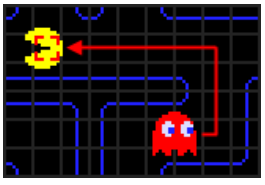
Tanto comer *dots*, como *Power Pellets*, como fantasmas en *Fright* concede puntos, que se acumulan para un ranking. Cada nivel aporta mas puntos pero incrementa la dificultad del juego acelerando a los fantasmas, dando menos tiempo de *Fright* o reduciendo la velocidad de *Pac-Man*.

2.2. La personalidad de los fantasmas

Una de las características más destacables de *Pac-Man* es que se trató de dar una **personalidad característica** a cada uno de los cuatro fantasmas. Esto se refleja en la estrategia particular de cada uno al tratar de cazar a *Pac-Man*; algunos son más agresivos, otros más esquivos, otros más cobardes...

La siguiente lista presenta los fantasmas originales, con el siguiente formato:
Nombre de la estrategia en inglés - *Nombre del fantasma* (nombre de la estrategia en japonés, 'significado')

-  **Shadow - *Blinky*** (oikake, 'perseguidor')
  Comportamiento: Agresivo
  Su objetivo es directamente el tile[2] de *Pac-Man*.

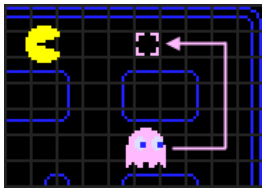


-  **Speedy - *Pinky*** (machibuse, 'emboscador')
  Comportamiento: Semi-táctico, situacional
  Trata de predecir posiciones futuras de *Pac-Man* para emboscarle. Su objetivo es la posición de *Pac-Man* +

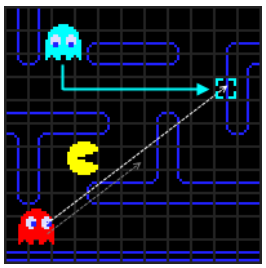
[1] Arcade es el término genérico de las máquinas recreativas de videojuegos disponibles en lugares públicos de diversión, centros comerciales, restaurantes, bares, o salones recreativos especializados.

[2] Los juegos 2D suelen definir su mundo usando una matriz, asignando valores para indicar, por ejemplo, dónde hay paredes. Cada una de estas casillas se conocen, en el mundo de los videojuegos, como tile.

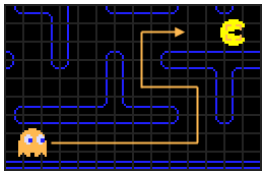
4 tiles en la dirección en la que mira.



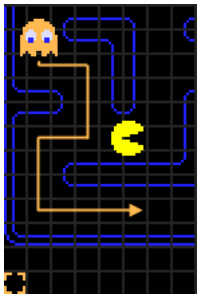
-  **Bashful - *Inky*** (kimagure, 'caprichoso')
Comportamiento: Semi-cooperativo
Su posición objetivo es relativa a *Blinky* y trata de utilizarla para rodearle entre los dos. Se calcula trazando un vector desde la casilla de Bliky hasta la posición de *Pac-Man* + 4 tiles en la dirección en la que mira (coincidiendo con la posición objetivo de *Pinky*), y posteriormente doblando su longitud. Es el único que utiliza datos de otro fantasma.



-  **Pokey - *Clyde*** (otoboke, 'ignorancia fingida')
Comportamiento: Máquina de estados, asustadizo.
El único que se comporta de dos modos distintos dependiendo de una condición dada. Estos se alternan en función de la distancia euclidiana entre él y *Pac-Man*. Si hay más de 8 tiles de distancia entre ambos, el objetivo es *Pac-Man* (como *Blinky*).



Si hay 8 o menos tiles de distancia entre ambos, el objetivo es la esquina inferior izquierda del escenario.



Se puede observar cómo los fantasmas aparentemente incorporan elementos de IAs de comportamiento como uso del entorno para cumplir un propósito - *Pinky* está diseñado para rodear a *Pac-Man* - o de coordinacion entre IAs - *Inky*.

Sin embargo, el comportamiento de la IA tiene un patrón **limitado e inalterable**, y los resultados positivos son, aunque facilitados por el diseño, casuales; *Pinky* puede efectivamente rodear a *Pac-Man*, pero es un resultado casual, ya que no ha realizado un análisis del entorno para tratar de asegurar que esto ocurra.
A su vez, *Inky* puede acorralar a *Pac-Man* entre él y *Blinky*, pero no ha habido una coordinación real y planificada entre las dos IA. Además, estas estrategias tan limitadas provocan situaciones no deseadas bien documentadas y conocidas, como problemas de ineficacia o incluso movimiento en bucle sin ningún progreso.

A pesar de sus limitaciones, *Pac-Man* en su momento fue un juego muy novedoso por incorporar este tipo de Inteligencia Artificial modular tan compleja y se podría considerar que su estrategia contiene **elementos primitivos de Inteligencia Artificial basada en comportamiento**, lo que lo convierte en un juego ideal para valorar la diferencia al incorporarle IAs más complejas de este tipo.

3 Propuestas de fantasmas

La forma más fácil de incorporar elementos avanzados de IAs de comportamiento era **incorporando fantasmas nuevos** que se basaran en características de este tipo de IAs. Las ideas más prometedoras figuran en la Tabla 2.

Tras hacer un análisis de ventajas, desventajas y costes previstos, se llegó a la conclusión de que el mejor fantasma para comenzar el proyecto era *Kiry*, ya que sus habilidades eran las más simples de desarrollar y una vez implementadas podían aplicarse al resto de fantasmas. Así pues, **se estableció a *Kiry* como el objetivo principal** y *Charles* como objetivo opcional.

Goldy requería un conocimiento de perfiles de jugador para su funcionamiento, algo que se podría obtener una vez recreado el juego original y con las adecuadas herramientas de recolección de datos, por lo que su desarrollo, si existiera, sería muy tardío. Por último, la inteligencia colectiva podía utilizar elementos de todos los fantasmas anteriores, así que se estableció como la menos prioritaria.

El proyecto en su estado actual de publicación presenta una **implementación completa de *Kiry***. Es posible que se traten de implementar el resto de fantasmas en líneas de continuación que queda fuera del tiempo dedicado a este proyecto.

4 Definición del proyecto

Con los fantasmas definidos, ya se podían pasar a aspectos formales de la definición, planificación y evaluación del

Nombre	Característica
 ***Kiry*** (Hunter)	**Situacional** - Analiza el entorno y trata de acorralar a *Pac-Man* utilizando el ambiente. Análogo a *Pinky*.
 ***Charles*** (Keeper)	**Situacional** - Dragón que protege un tesoro, máquina de estados. Analiza el entorno y busca un área con alta concentración de *dots*, que patrulla. Si *Pac-Man* se acerca, le trata de alejar persiguiéndole hasta estar lo suficientemente lejos, momento en el que retoma la patrulla. Análogo a *Clyde*.
Goldy (Wise)	**Aprendizaje** - Fantasma que toma decisiones en función de tendencias, analizadas a partir de datos tomados de partidas anteriores de el mismo jugador o el conjunto de todos los jugadores.
 ***Collective Intelligence***	**Cooperativo** - Conjunto de fantasmas que trabaja de forma cooperativa para atrapar a *Pac-Man*. Compuesto por un *master* y varios *servant*, el *master* realiza los análisis de situación teniendo en cuenta todo el grupo, toma las decisiones precisas y manda órdenes a los *servant*. Análogo a *Blinky-Inky*.

Tabla 2: Nuevos fantasmas planteados

proyecto. Cada uno de estos aspectos del diseño del software se tratará en una subsección diferenciada.

4.1. Objetivos del proyecto

Los objetivos se categorizaron en críticos (necesarios para el éxito del proyecto), prioritarios (no críticos más favorables) y secundarios (prioridad mínima);

1. Objetivos críticos
   a) Realizar un estudio del juego original
   b) Implementar el juego original con el mayor detalle posible
   c) Desarrollar el fantasma *Kiry*
   d) Realizar pruebas con como mínimo 20 jugadores reales
   e) Implementar un sistema de recolección de datos para las pruebas.
   f) Analizar los resultados objetivos numéricos y sacar conclusiones.
   g) Analizar los resultados percibidos de los jugadores y sacar conclusiones.
2. Objetivos prioritarios
   a) Desarrollar el fantasma *Charles*
3. Objetivos secundarios
   a) Recoger datos de profiling para *Goldy*
   b) Desarrollar el fantasma *Goldy*
   c) Desarrollar Master
   d) Desarrollar Servant

Al finalizar el proyecto, **todos los objetivos críticos se han cumplido satisfactoriamente**.

4.2. Planificación del proyecto

La planificación del proyecto - incluyendo requisitos, *stakeholders*, metodología del proyecto, herramientas y riesgos/respuesta - está documentada en el **Apéndice A**.
La decisión de esta organización se ha basado en dos criterios, la longitud requerida y la importancia de mantener una lectura dinámica de los contenidos. Así pues, hemos optado por mantener la descripción de los objetivos y las propuestas algorítmicas en el artículo y todos los aspectos metodológicos en el apéndice, de esta manera hemos conseguido un mejor encaje con la lectura de la introducción del proyecto (secciones anteriores).

4.3. Métricas

Para el análisis de la influencia del modelo de IAs se deben considerar una serie de métricas;

1. Subjetivas (percibidas)
   a) **Dificultad** - Sensación de aumento en la dificultad del juego respecto al original apreciada por el usuario al incorporar la nueva IA.
   b) **Complejidad** - Sensación de que la IA sigue estrategias más elaboradas, originales, ingeniosas o realistas, con independencia de que estas añadan un nivel de dificultad extra apreciable. (Ejemplo: en un juego de disparos un enemigo que toma cobertura puede ser menos eficiente venciendo al jugador que uno que no lo hace, pero éstos lo reconocen como una IA mas compleja).
   c) **Diversión** - Sensación de que añadir una IA en particular hace más divertido el juego. El motivo de esta métrica es intentar buscar patrones sobre la influencia de la complejidad y la dificultad en la diversión general que aporta el juego.
2. Objetivas (Numéricas)
   a) **Consumo de recursos** - En que aumento de costes de memoria (RAM) y procesamiento (FPS)[3] se traduce la incorporación de algoritmos mas complejos.

[3]Número de frames por segundo, es decir, el número de veces que se procesa un ciclo de juego en un sólo segundo. Conviene que se estable para evitar ralentizaciones súbitas.

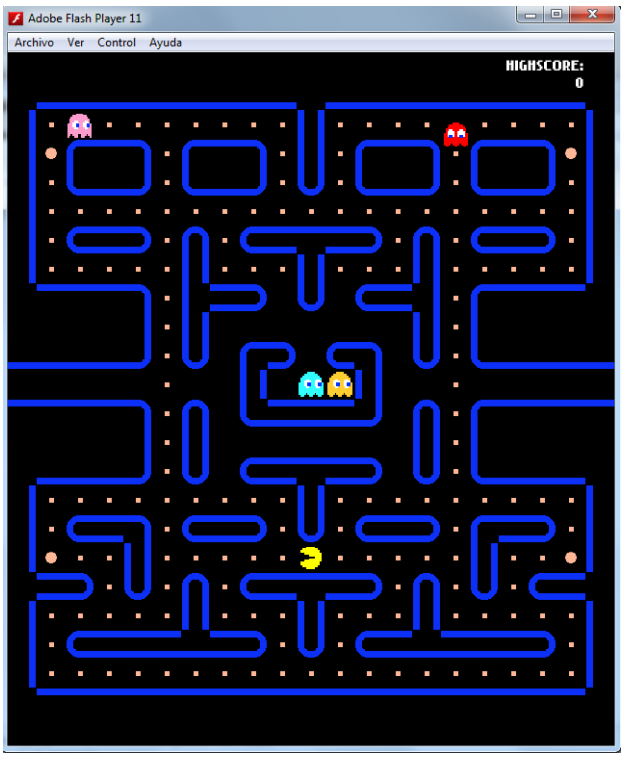


Figura 2: Imagen del juego final.

b) **Efectividad del fantasma** - Cuán efectivo es el funcionamiento del fantasma, teniendo en cuenta factores como la frecuencia de victorias en función del número de partidas donde participa o la velocidad con la que consigue derrotar al jugador.

Durante el análisis se intentarán relacionar las tres métricas subjetivas. Es decir, demostrar si las personas que creen que una IA es más compleja la suelen considerar más difícil, si las personas que creen que una IA es más difícil de vencer es más divertida y si creen que incorporar una IA más compleja, independientemente de si hace el juego más difícil o no, añade mas variedad y lo hace mas divertido.

5 Desarrollo del juego base

El desarrollo de la base del juego base transcurrió durante aproximadamente mes y medio, momento en el que se tenía una versión suficientemente funcional y fiel a la original como para empezar a implementar fantasmas nuevos.

5.1. Planificación del software

Algunos elementos del diseño del software han sido importantes, requiriendo una importante planificación que facilitara la adaptabilidad a la hora de crear fantasmas nuevos. Una breve descripción de dos de estos elementos - el modelo de capas que sigue el juego y el modelo fantasma/módulo - se puede encontrar en el **Apéndice B**, diferenciado del resto del documento por su importancia relativa al diseño de Software.

5.2. Resultado final

El juego ha sido **extremadamente fiel al original** y las únicas modificaciones que han ocurrido han sido técnicas y son inapreciables durante el juego. Aquellos cambios que afectan a los fantasmas se aplican tanto a IAs nuevas como a viejas, evitando desequilibrar el juego y desacreditar los resultados.

El juego ha sido **apropiadamente testeado** con checklist, y se han utilizado varias herramientas del framework como un controlador de recursos consumidos (Figura 3) o una capa de dibujo de debugging que permitía, por ejemplo, dibujar un circulo en el objetivo de desplazamiento de un fantasma para asegurar su correcto funcionamiento.

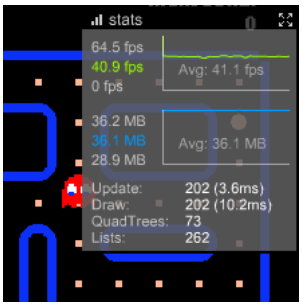


Figura 3: Herramienta de debugging utilizada en el testing, registrando frames por segundo y consumo de memoria.

6 *Kiry*: Fantasma situacional basado en comportamiento

Una vez finalizado el juego base, el siguiente paso consistía en **implementar el nuevo fantasma, *Kiry***.

6.1. Planificación inicial

La planificación y programación del nuevo fantasma ocupó el segundo tercio del tiempo total del proyecto, siendo la primera **especialmente larga y difícil**, tratando de buscar un comportamiento que aplicara eficazmente la cualidad situacional de las IA de comportamiento.

Se consideraron dos tipos de enfoque; uno que utilizara exclusivamente información del laberinto y otro que, además, utilizara otros datos del entorno como la posición de otros fantasmas (sin coordinarse con ellos) o la posición de los *dots*. Viendo que el segundo era una extensión del primero, se optó por implementar el primer modelo y dejar pendiente el segundo.

6.2. Mapa de presión

Descartando información de otras entidades del entorno, la principal arma de *Kiry* es el propio laberinto. La estrategia principal de *Kiry* sería la de analizar el entorno y tratar de **reconocer zonas con más peligro potencial para *Pac-Man***. Conociendo estas zonas, *Kiry* podría presionar a *Pac-Man* y obligarle a ir hasta ellas, donde

estaría mas expuesto tanto a ella como al resto de fantasmas.

Para satisfacer esta idea nació el **mapa de presión**. El mapa de presión consiste en hacerse con una copia del laberinto y asignar un valor de peligro a un *tile* transitable concreto. Este valor se calcula en función de varios principios tácticos:

1. **Los cruces** (tiles con mas de dos salidas) **son relativamente seguros**, pues tienes alternativas si un camino está bloqueado
2. **Los pasillos son zonas de gran peligro**, entendiendo pasillo como los espacios entre dos cruces. Si un fantasma entra por un extremo y otro por el contrario, *Pac-Man* no tiene escapatoria
3. Por consiguiente, **el peligro de un pasillo es proporcional a su tamaño**, pues más espacio que recorrer implica más tiempo en el que *Pac-Man* es vulnerable.
4. **Los caminos sin salida son zonas de peligro máximo**, sólo hay un punto de entrada y basta con un fantasma para bloquearlo.
5. **Cuanto más largo es el pasillo hasta el camino sin salida, más peligro existe**, pues la distancia a recorrer es el doble de la longitud del camino.
6. **Si el mapa es separable en zonas interconectadas por un sólo camino, las zonas de menor tamaño tienen presión añadida**, pues ofrecen menos espacio de maniobra.

El modelo que utiliza *Kiry* se basa en estos principios. Primero se crea un grafo utilizando los cruces como vértices y los caminos que los conectan como aristas. Se buscan componentes biconectados - que serían las zonas interconectadas por un sólo camino - y se les asigna una presión base dependiendo de su tamaño.
Una vez hecho esto, se recorren los pasillos y se añade presión extra en función de la distancia con el cruce mas cercano, y por último se penaliza aún más a los caminos sin salida.

Se implementó un **nuevo nivel de prueba** diseñado para contener todas las características de las que este modelo podría sacar provecho. El mapa de presión correspondiente se puede ver en la Figura 4, donde el rojo representa mayor presión o peligro y el verde representa zonas mas seguras para *Pac-Man*..
Se puede observar como **la zona central**, más grande, **tiene una presión base menor**, mientras que las tres islas tienen considerablemente más.

También se puede apreciar como **los pasillos hacen un "degradado"** donde la parte central, al estar más alejada de los cruces, tiene una presión bastante mayor.

Por último, se puede ver como **en los caminos sin salida la presión se dispara**, sobretodo si vienen precedidos de un pasillo largo como es el caso de la parte central superior.

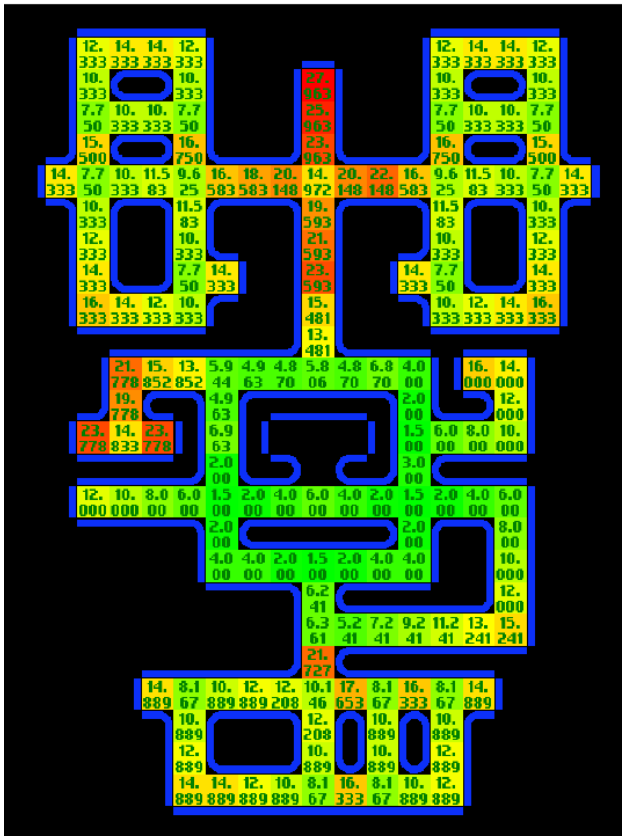


Figura 4: Mapa de presión en un mapa diseñado para explotar sus capacidades.

El mapa de presión del laberinto original se puede observar en la Figura 5.

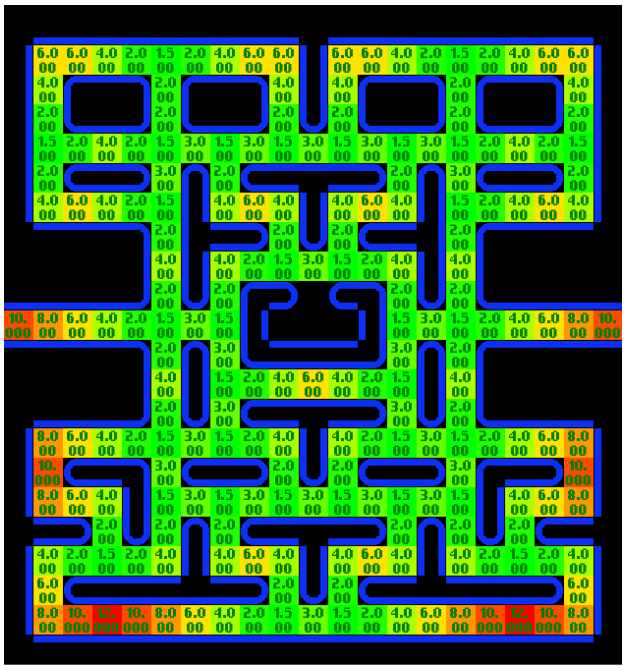


Figura 5: Mapa de presión en el mapa original

Kiry utiliza este mapa leyendo los valores de las casillas en un radio determinado desde la posición de *Pac-Man* y identificando la zona con más presión. Una vez identificada, trata de moverse dejando a *Pac-Man* entre ella y la zona,

forzándole a arrinconarse.

Este comportamiento convierte a *Kiry* en **un fantasma esquivo**, que prefiere presionar a ir directamente a por *Pac-Man* (como se puede ver en la Figura 6), que suele facilitar el trabajo a fantasmas más directos como *Blinky* y que tiene costumbre a desplazarse en paralelo a *Pac-Man* cuando se encuentra en los extremos del escenario (sobre todo en la zona con más presión del mapa, situada en la parte inferior) a fin de evitar que vuelva al centro, donde hay mas libertad de movimiento.

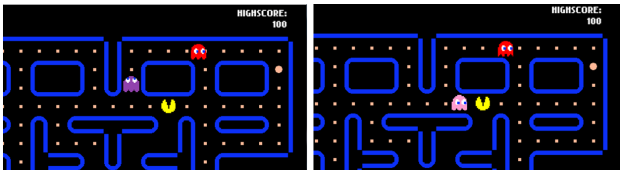


Figura 6: Diferencias de comportamiento entre *Pinky* y *Kiry* dada la misma situación.

6.3. Movimiento mejorado

Como extra a su comportamiento nuevo, se ha **mejorado la toma de decisión de desplazamiento**. Los fantasmas originales, llegados a un cruce, comprueban la distancia euclidiana desde el primer *tile* de cada uno de los pasillos disponibles hasta *Pac-Man*, y toman el que presenta un valor menor. Este cálculo, aunque rápido, **no toma en cuenta los muros del laberinto**, y a veces provoca que los fantasmas tomen caminos más largos para llegar a su objetivo.

Kiry, en cambio, implementa un A*[4] que calcula el camino más corto desde su cruce de origen hasta su destino. Este sistema, al tener en cuenta los muros, resulta más eficaz en la planificación del movimiento y provoca que *Kiry* mantenga **distancias de seguimiento más cortas** con *Pac-Man* que otros fantasmas. Además, su incorporación no ha supuesto penalizaciones apreciables en el rendimiento.

7 Análisis del comportamiento

Para comprobar la efectividad de *Kiry* en comparación con su homólogo, *Pinky*, se diseñó una **prueba con usuarios reales**.

7.1. Introducción al experimento

El experimento incluía un apartado perceptivo y otro empírico.

Los valores **perceptivos** serían registrados por formulario y ayudarían a hacerse una idea general de las impresiones del jugador relacionadas con la dificultad y complejidad del nuevo fantasma en comparación con el antiguo y el efecto de estos en la diversión general.
Por otro lado, los valores **objetivos numéricos** serían registrados automáticamente por el programa, y servirían para comprobar la influencia de la incorporación de una estrategia distinta en resultados medibles del juego, como tiempo o *dots* recogidos.

De entre los dos, los resultados **perceptivos tienen una relevancia mayor**, ya que la intención del estudio otorga más prioridad al efecto sobre el jugador final que en la efectividad numérica del modelo.
Como añadido, los valores objetivos numéricos son más inflexibles, registrando valores absolutos (derrotas-victorias) pero ignorando datos más complejos como la diferencia de dificultad entre los dos modelos que ha sufrido un jugador a pesar de haber logrado victoria con ambos. Por este motivo, **los valores objetivos numéricos cumplen una función complementaria**.

7.2. Preparación del experimento

El objetivo de número de sujetos inicialmente era de 10 personas mínimo, extendiéndose a 20 tras comprobar que la cantidad de sujetos no arrojaba resultados concluyentes y **alcanzando finalmente la cantidad de 21 sujetos**.

El **perfil** buscado en los jugadores era sujetos jóvenes, conocedores del juego original y preferentemente jugadores frecuentes. Tener experiencia directa con el juego original o derivados de la franquicia también estaba altamente valorado.

Las pruebas se realizaron de forma autónoma, siguiendo unas instrucciones en un documento de Google Forms y rellenando los campos paso a paso. La duración de la prueba se sitúa normalmente entre los 15 y los 45 minutos.

Durante la **primera fase del experimento** se instruye a los jugadores sobre la interfaz general y los controles, y después se les deja un tiempo ilimitado para que prueben el juego con los fantasmas originales hasta que se habitúen al juego.
Una vez están preparados, se les instruye para substituir a *Pinky* por *Kiry*, y al finalizar una única partida se les pregunta por diferencias apreciadas, de una forma ambigua y sin hacer referencia al nuevo fantasma o su comportamiento. El propósito de esta pregunta es averiguar si los usuarios han notado alguna diferencia apreciable sobre el nuevo fantasma de forma inconsciente, sin llamar la atención sobre el asunto de forma explícita.

Tras esto se inicia la **segunda fase**, donde se permite a los jugadores jugar libremente con la combinación original y con la combinación que incorpora a *Kiry*, intercalando partidas de una y otra y pidiendo un mínimo de cuatro partidas con cada una. Tras ello, se rellenan datos sobre dificultad, complejidad y diversión apreciadas y se pide enviar los datos objetivos numéricos registrados para su valoración.

Los datos objetivos numéricos se guardaban al final de cada partida en formato **.XML** - ejemplo en la Figura 7 - a fin de diferenciarlos y facilitar su lectura en Matlab.

[4] Algoritmo de búsqueda en grafos que encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo.

```
<?xml version="1.0"?>
- <game>
    <map>Nivel 1</map>
    <player>MLF</player>
    <date>2015-06-03 20:54:11</date>
    <version>windows</version>
  - <ghosts>
      <ghost ID="Blinky"/>
      <ghost ID="Pinky"/>
      <ghost ID="Inky"/>
      <ghost ID="Clyde"/>
    </ghosts>
  - <end>
      <data reason="killed">Clyde</data>
      <score>1210</score>
      <dots>56</dots>
      <powerpellet>1</powerpellet>
      <ghostEated>2</ghostEated>
      <time>23751</time>
    </end>
</game>
```

Figura 7: Ejemplo de XML con datos objetivos numéricos.

7.3. Resultados de los datos subjetivos

El experimento se ha realizado sobre 21 sujetos de edades entre los 19 y los 31 años, estando la media en los 22. De ellos, el 57.1 % son hombres, el 42.9 % restante siendo mujeres.

Como se puede ver en la Figura 8, la búsqueda de un perfil concreto de jugador ha sido satisfactoria y la mayoría de gente, además de ser **jugadores frecuentes**, tenían **cierto nivel de experiencia con el juego original**.

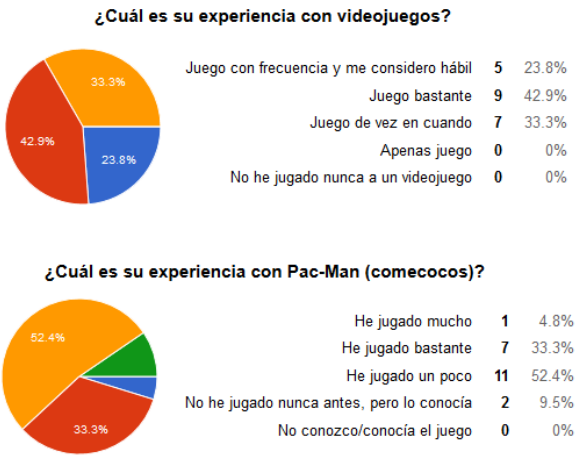


Figura 8: Experiencia de los sujetos.

La primera impresión es polarizada. 11 personas (52.4 %) no han notado ninguna diferencia, mientras que las 10 personas restantes (47.6 %) si han anotado alguna diferencia apreciable, que escribieron en un campo del formulario. Obviando algunas observaciones no relacionadas como el cambio de color o cuándo los fantasmas abandonan la *House*, destacan **dos tipos de observaciones principales**. En primer lugar:

"Da la sensación que Kiry tiene un compartimiento menos simple que el de Pinky. Parece estar más coordinado con los movimientos de los otros fantasmas. Más sincronizado para poder cerrarte el camino."

"[...] Por otro lado parecía que Kiry iba por otros caminos para arrinconarme en lugar de seguirme ciegamente."

"Me emboscan antes!!!"

."Es un poco más complicado huir. Normalmente los fantasmas te atacan por detrás y este te pilla por delante, y es más inesperado."

"Kiry no ataca primero, siempre espera a que otros fantasmas de acerquen. Va un poco por su cuenta."

Todas estas observaciones describen con mayor o menor precisión el tipo de comportamiento de *Kiry*, o bien consecuencias de él. Esto significa que hay un número importante de personas (en torno al 28 % de los encuestados) **capaz de apreciar características complejas de el comportamiento de una IA con una sóla muestra**.
La segunda observación interesante es la siguiente:

"No estoy seguro, pero puede ser que Kiry vaya más rápido?"

"Parecía que iba ligeramente más rápido que el rosa."

Estas observaciones, a primera vista, pueden parecer incorrectas y sin relación. Sin embargo, tras una segunda consideración, es probable que esta sensación de mayor velocidad sea debido a el uso de un sistema de movimiento basado en A*, mas eficaz que el de los fantasmas originales. Aunque por su imprecisión no es posible conectar estos dos hechos con seguridad, se podría considerar que estos sujetos **también son conscientes en cierta medida de un cambio en la IA**, en cuyo hipotético caso los sujetos capaces de apreciar diferencias con sólo una muestra ascienden al 38 %.

Los resultados sobre la valoración de los tres parámetros percibidos se pueden observar en la Figura 9. Se puede observar que **los resultados en general son satisfactorios**; Más de la mitad de la gente ha experimentado un **incremento de dificultad apreciada** al incluir a *Kiry*, la extensa mayoría ha apreciado un **mayor grado de complejidad** en el fantasma tras varias pruebas y, aunque de forma más discutida, parece que estos cambios han aportado una **mejora en la diversión que otorga el juego**. Estos datos parecen confirmar que la incorporación de IAs basadas en conocimiento tiene **un efecto apreciable sobre la experiencia de muchos de los jugadores**, cumpliendo el objetivo principal del proyecto.

En la Figura 10, 11 y 12 se puede observar el método utilizado para comprobar la relación entre dos variables. Cada eje representa una variable perceptual. Se han colocado en ellos los valores recogidos en la encuesta, 1 representando los más negativos y 5 los más positivos, y en cada casilla se ha asignado el número de usuarios que han puntuado de esa forma esas dos variables. Si existiera una dependencia concreta entre dos variables dadas, la mayoría de valores deberían presentarse en la diagonal, ya que alterar uno

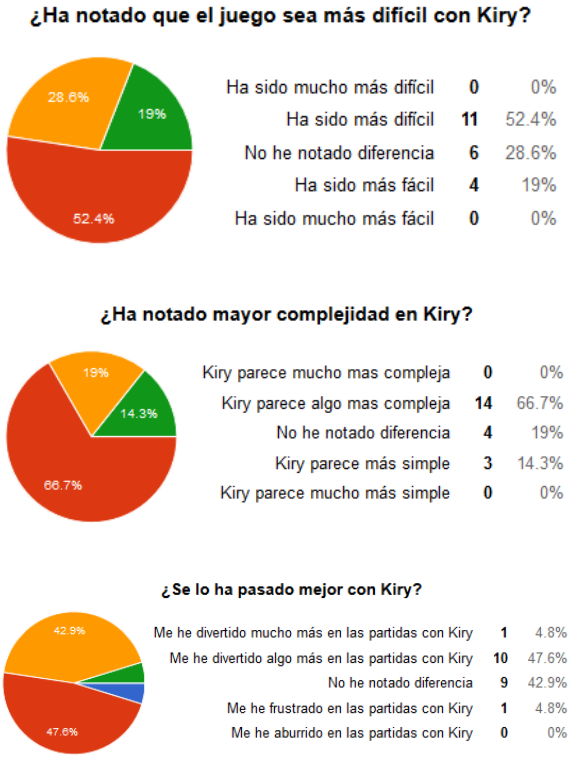


Figura 9: Valores percibidos.

implicaría alterar el otro.

La relación más notable es la de complejidad-dificultad (Figura 10). Como se puede apreciar en el mapa topológico, hay muchísima tendencia hacia la diagonal. Esto implica que **un aumento de complejidad implica con mucha seguridad un un aumento de dificultad**.

En segundo lugar podemos ver como la diversión y la dificultad (Figura 11), aunque de forma mucho menos decisiva, también parece indicar que **hay cierta relación entre la dificultad y la diversión**. La mayor dispersión de valores se puede explicar por el desagrado de algunos jugadores a los juegos que suponen un mayor desafío, aunque parece que la tendencia es a, como mínimo, valorar la posibilidad de poder seleccionar una dificultad mayor.

Por último tenemos la relación entre diversión y complejidad (Figura 12). En este caso, parece que **podríamos descartar una relación entre el incremento de complejidad y la diversión**.

7.4. Resultados de los datos objetivos

Tras varios análisis y pruebas bajo diversas condiciones, los datos objetivos numéricos recogidos **no parecen indicar diferencia entre los dos modelos**.

El dato mas destacable es el tiempo medio antes de la derrota del jugador; en *Kiry* ronda los 36.2 segundos, con una desviación de 22.21. El tiempo medio de *Pinky*, por otra parte, es de 38 segundos, con una desviación de 47.17.

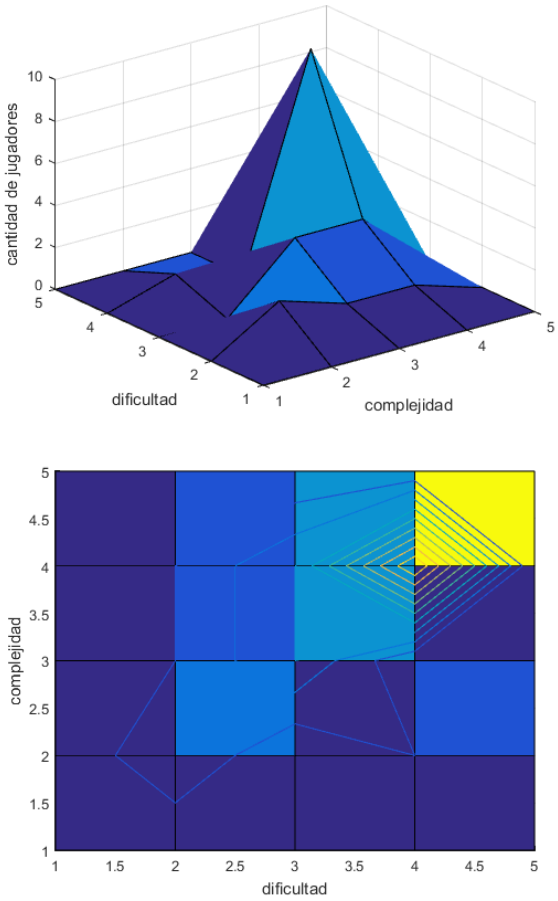


Figura 10: Relación entre dificultad y complejidad

Los *dots* medios obtenidos antes de la derrota son de 72 para *Kiry* y 67 para *Pinky* (desviación 38.76 y 37.91).
Los *Power Pellet* medios recogidos antes de la derrota son 1.31 para *Kiry* y 1.23 para *Pinky* (desviación 1.29 y 1.25).
Los fantasmas comidos medios son de 1.05 para *Kiry* y 1.08 para *Pinky* (desviación 1.4 y 1.48).
El score medio con *Kiry* es de 1065, con *Pinky* siendo 1005 (desviación 814 y 760).

Los valores similares con apenas desviación estándar entre ellos, la ligera ventaja de *Pinky* en algunos campos y sin embargo el tiempo más reducido de *Kiry* y su mayor precisión en esta medida parecen apuntar a una idea general;

Kiry, al ser liberada en segundo lugar, y estar basada en ayudar a los otros fantasmas, **es más ineficiente al principio del juego**, donde sólo hay un fantasma más a parte de ella. Esta menor presión, contraria al enfoque más directo de *Pinky*, permite al jugador conseguir mayor puntuación.
Sin embargo, **en cuanto se empiezan a liberar los otros dos fantasmas** y *Kiry* puede sacarle provecho a su estrategia, **su efectividad incrementa**, cazando muchos jugadores en un espacio de tiempo reducido, de ahí la poca desviación estándar.

Esta idea parece ser apoyada por el hecho de que aquellos jugadores que expresaron una mayor dificultad apreciada

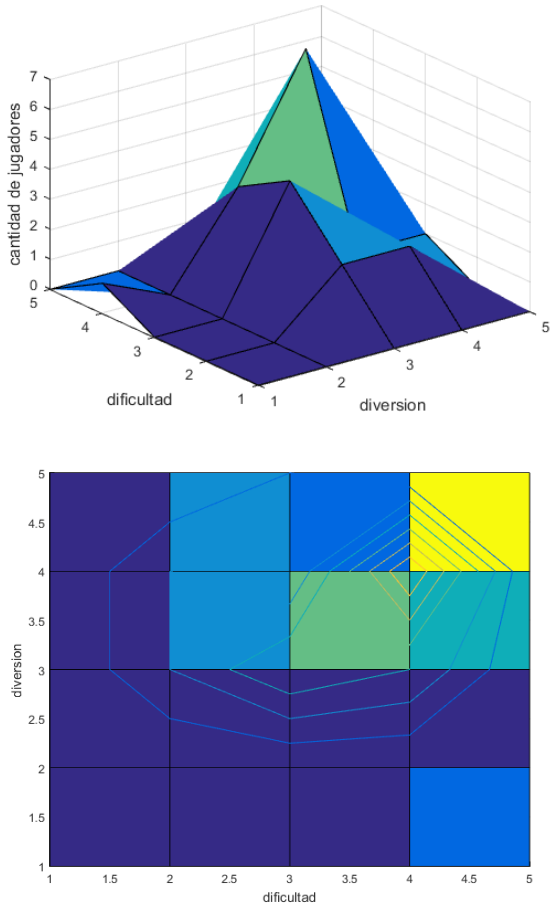


Figura 11: Relación entre dificultad y diversión

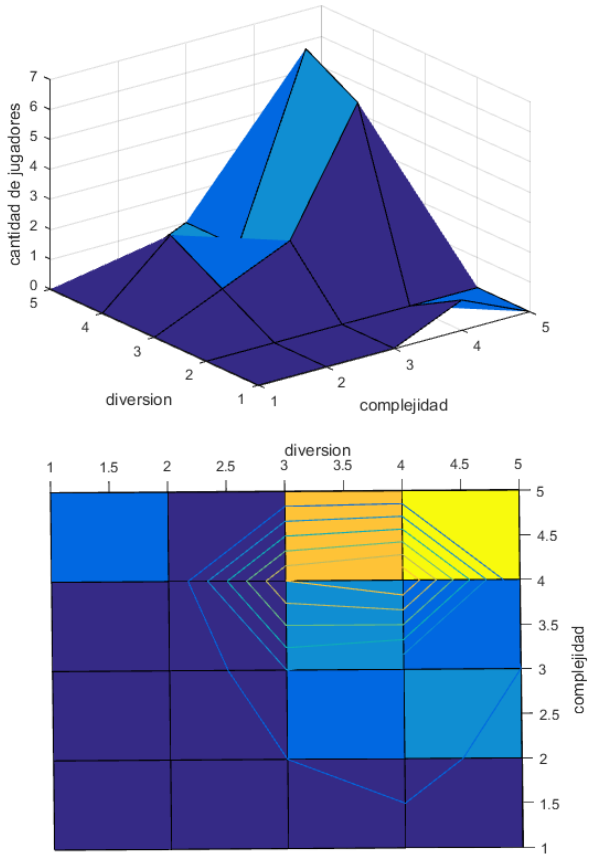


Figura 12: Relación entre diversión y complejidad

suelen ser aquellos que más tiempo aguantaron en el juego, enfrentándose a *Kiry* en condiciones más favorables para ella.

Aunque los resultados arrojados nos ayudan a comprender la diferencia de condiciones de efectividad relativa de las dos IA, **no son concluyentes de una forma más absoluta**.

8 Conclusiones

Con este proyecto se ha aportado una nueva implementación del Pac-Man original, especialmente orientado al diseño de nuevos fantasmas, que permite experimentar con técnicas para la programación de Inteligencias Artificiales en videojuegos.

Adicionalmente, se han utilizado técnicas de las IAs basadas en comportamiento para definir un nuevo fantasma más complejo, que combina el uso de información espacial en su estrategia y optimización de desplazamiento con A*.

Las pruebas experimentales realizadas aportan estas conclusiones generales:

- La incorporación de IAs basados en comportamiento en videojuegos influye positivamente en la dificultad, complejidad apreciada y diversión general del jugador.
- Entre el 28 % y el 38 % de los jugadores es capaz de apreciar una mayor complejidad en una IA de un solo vistazo.
- Hay una relación clara entre dificultad y complejidad.
- Hay cierta relación directa entre dificultad y diversión.
- No se puede demostrar una relación directa entre complejidad y diversión.
- Los resultados objetivos numéricos arrojados no son concluyentes.

Es posible **continuar el desarrollo del estudio al margen de este proyecto**, a fin de reafirmar las conclusiones obtenidas y probar la influencia de otras IAs con aspectos distintos de las Inteligencias Artificiales basadas en comportamiento.

Como añadido personal, me ha resultado muy ilustrativo **todo lo aprendido sobre experimentación y análisis de datos**, un enfoque científico al que no estaba habituado.
También ha sido muy valioso el tiempo consumido **el comportamiento de la nueva IA**, ya que, siendo un aspecto basado más en la planificación y diseño, ha sido un buen contraste frente al enfoque de implementación de la mayoría de las asignaturas de la carrera.
Por último, el diseño de la aplicación ha reafirmado más aún **la importancia de la planificación** en la ingeniería de software.

AGRADECIMIENTO

A todos los **jugadores que se presentaron para el experimento**, por su desinteresada paciencia.
A mi tutora, **Maria Vanrell**, por remar junto a mi semana tras semana a pesar de la tormenta.
A mi **familia y amigos**, por estar siempre ahí siempre.
A **Toru Iwatani** y todos los desarrolladores de juegos que ponen un poco de su alma en su trabajo.
Y a ***Pac-Man***, al que deseo un feliz 35 aniversario.

REFERENCIAS

[1] Birch, Chad. Understanding pac-man ghost behavior, dic 2010.

[2] Birk, Andreas. Behavior-based robotics, its scope and its prospects. 24th Annual Conference of the IEEE Industrial Electronics Society. Vrije Universiteit Brussel, Artificial Intelligence Laboratory, 1998.

[3] Daucet, Lars. Flash is dead, long live openfl!, mar 2014.

[4] Maes, Pattie. Behavior-based artificial intelligence. University of Nevada, Reno, 2010.

[5] Pittman, Jamey. The pac-man dossier, v. 1.0.26. jun 2011.

[6] Wikipedia. Pac-Man game introduction.

[7] Real Academia Española. Diccionario online.

[8] J. Russell, Stuart and Norvig, Peter. Artificial Intelligence: A Modern Approach. Pearson Education, 2003

APÉNDICE A: PLANIFICACIÓN DEL PROYECTO

A.1. Requisitos

La versión compacta - a fin de caber en el documento - de los requisitos de software base para este proyecto son:

1. Requisitos funcionales
   - *a*) El juego debe tener capacidad para introducir valor de nombre de jugador y con qué cuatro fantasmas se quiere jugar.
   - *b*) El juego debe registrar puntuación dinámicamente.
   - *c*) El juego debe informar de la puntuación una vez finalizada la partida
   - *d*) Deben implementarse los cuatro fantasmas originales de forma estrictamente igual.
   - *e*) Debe implementarse el mapa original.
   - *f*) Debe implementarse un modo normal, *Fright* y dispersión para todos los fantasmas.
   - *g*) El juego debe guardar datos de una partida para su análisis.
   - *h*) *Pac-Man* se controlará con el teclado.
   - *i*) *Pac-Man* debe incorporar el mismo sistema de memoria de direccionamiento que el original
2. Requisitos no funcionales
   - *a*) Accesibilidad - *Pac-Man* debe ser controlable tanto con las letras AWSD como con las de dirección
   - *b*) Portabilidad - El juego se desarrollará para multiplataforma; Flash y PC.
   - *c*) Escalabilidad - La base de los fantasmas debe ser modular para facilitar la creación de nuevos modelos.
   - *d*) Escalabilidad - Se debe implementar una lectura de escenarios externos y no uso de escenarios integrados, para facilitar experimentación en diversos entornos.
3. Restricciones
   - *a*) Los valores numéricos (puntuación, tiempos de modo...) deben ser estrictamente los mismo que en el juego original.
   - *b*) Las nuevas IA sólo pueden incorporar cambios en el modo de caza.
   - *c*) Las nuevas IA sólo pueden alterar comportamiento, no parámetros como velocidad.
   - *d*) Deben haber 4 fantasmas en una partida.
   - *e*) Los fantasmas saldrán del *House* en orden y con las condiciones de salida del juego original.
   - *f*) Los Fantasmas deben tomar decisiones únicamente en los cruces

A.2. *Stakeholders*

Personas o entidades que están o podrían estar interesados en el producto.

El desarrollador del estudio, el principal *stakeholder*, buscando conseguir experiencia y conocimiento.

Los jugadores, que buscan mejores experiencias de juego que el modelo basado en comportamiento les podría otorgar.

Los desarrolladores de juegos, que pueden utilizar los resultados que arroje el estudio para conocer datos como la relación entre la efectividad de estos modelos y su efectividad en distintos campos objetivos numéricos y percibidos, a fin de utilizarlos en futuros productos.

A.3. Método de gestión de proyecto

El objetivo del proyecto no es un producto final completamente definido, si no que se trata de un estudio que ha **ido evolucionando** según las necesidades que se presentaban en cada momento y en base a los resultados obtenidos.
La orientación del proyecto pues no se ha basado en una planificación total inicial si no que se ha seguido un sistema más flexible, de control contínuo y cíclico similar a **Scrum**.

El método se ha caracterizado por **reuniones semanales** con la tutora - documentadas en la página web de TFE -, donde se trata **el progreso pendiente** de la semana y el **progreso realizado**, se discuten **dudas, conclusiones y cambios** y se plantea el **progreso a realizar para la siguiente semana**.
Este progreso pendiente se decide dividiendo los objetivos pendientes en subobjetivos y organizándolos según prioridad, convirtiéndose lo más destacables finalmente en milestones.

A.4. Herramientas

La herramienta de desarrollo utilizada es HaxeFlixel, un framework para juegos 2D de código abierto y gratuito.

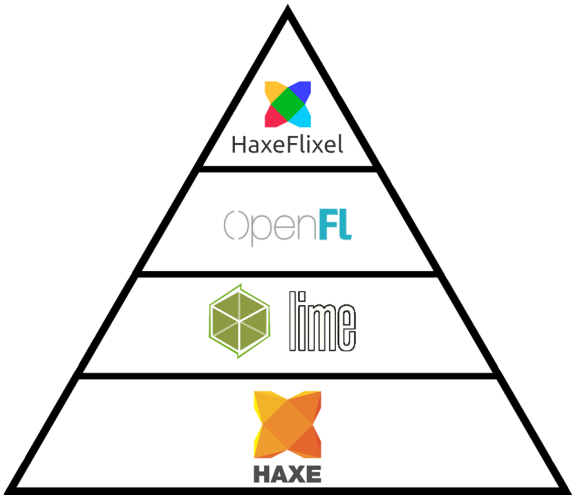


Figura 13: Pirámide representando las tecnologías utilizadas por HaxeFlixel, la base siendo las de menor nivel.

Las tecnologías que HaxeFlixel incorpora son las siguientes, ordenadas de menor a mayor nivel:

Haxe: Lenguaje de programación base. Orientado a objetos, similar a ActionScript 3.0 y exportable a varias plataformas - Flash, Windows, Linux, Android, HTML5... Para este proyecto se han utilizado las dos primeras.

Lime: Api de gráficos de bajo nivel, encargada de windowing, carga de ficheros e interacciones con OpenGL.

OpenFL: La API de gráficos de alto nivel. Basado en Flash.

HaxeFlixel: El framework. Incluye la estructura base de juego, las físicas y clases y funciones comunes de juegos 2D ya implementadas.

Se ha escogido esta tecnología porque se quería **agilizar el proceso de desarrollar el software** a fin de tener tiempo suficiente para hacer las pruebas con jugadores y analizar los datos.
Siendo un framework orientado a juegos, el trabajo se simplificaría bastante. Por otra parte, el estudiante tiene bastante experiencia con ActionScript 3.0, reduciendo el tiempo de aprendizaje de la nueva tecnología.

Como añadido, HaxeFlixel es **un framework relativamente nuevo** y aún en desarrollo, lo que ha provocado algunos problemas. Sin embargo, la impresión final es en general favorable.

El IDE (aplicación de edición de código) utilizado durante el desarrollo ha sido **Flash Develop**.

En el apartado gráfico, se ha utilizado especialmente **Adobe Photoshop CS5**, y los sprites se han creado desde 0 a base de capturas del juego original.

Para el mapeado de niveles se ha utilizado **OGMO**, una herramienta que permite importar elementos gráficos y crear mapas basados en cuadrícula con ellos, a fin de exportarlos en CSV y utilizarlos en HaxeFlixel.

El control de versiones se ha realizado con **Bitbucket**, aprovechando su capacidad para crear repositorios privados.

La documentación se ha realizado con **ShareLatex** (una página web que permite crear documentos de Latex en línea y en grupo) para la entrega del informe final y con **Microsoft Word** para el resto.

Por último, se ha utilizado **Google Forms** para dar instrucciones a los jugadores y recoger los datos percibidos de los usuarios y **Google Drive** como plataforma de descarga de la versión de prueba. Los datos objetivos numéricos de las pruebas se han guardado en XML y se han analizado en **Matlab**.

A.5. Riesgos y respuesta

Esta lista incluye algunos de los riesgos posibles para el proyecto y su plan de contención/respuesta:

1. Externos impredecibles
    a) **Catástrofe natural** - Poco probable, daño catastrófico - Se debe asegurar la máxima frecuencia de actualizaciones en el repositorio para evitar perder progreso y poder avanzar con otro dispositivo.
2. Legales
    a) **Problemas legales** - Poco probable, daño moderado - Se debe utilizar software libre o apropiadamente registrado, y se evitarán acciones que puedan infringir el copyright de Namco sobre el juego original.
3. Internos no técnicos
    a) **No alcanzar los objetivos propuestos a tiempo** - Moderadamente probable, daño alto - Para evitar no alcanzar los objetivos propuestos se debe mantener un cuidadoso control, seguimiento y priorización de los objetivos.
    b) **No definir suficientemente bien los objetivos** - Moderadamente probable, daño alto - Se debe tener en todo momento claros los objetivos que se han asignado en la reunión semanal.

c) **Tener mala metodología de evaluación de datos** - Moderadamente probable, daño moderado - Se debe asegurar que el procesamiento de los resultados y conclusiones que se extraigan de ellos sean correctos.

d) **No tener suficientes testers** - Moderadamente probable, daño moderado - Se debe tratar de asegurar el mayor numero de participantes en las pruebas marcándose un mínimo, asegurándose de que los datos recogidos no son inverosímiles y teniendo cautela en las conclusiones sacadas en función del tamaño de la muestra.

e) **Aprendizaje** - Altamente probable, daño moderado - Se debe considerar un margen de aprendizaje en la planificación general.

4. Técnicos

a) **Problemas con las tecnologías** - Altamente probable, daño moderado - Siendo una tecnología en desarrollo, es inevitable. Se deben asegurar margenes de resolución de errores en la planificación semanal.

b) **Perdidas de datos** - Poco probable, daño alto - Mantener al día los repositorios para mitigar el daño.

c) **Problemas de rendimiento** - Moderadamente probable, daño pequeño - Se debe controlar los recursos consumidos por el software a lo largo del desarrollo del proyecto para que un funcionamiento incorrecto no afecte negativamente el juego.

APÉNDICE B: PLANIFICACIÓN DEL SOFTWARE

B.1. Modelo de capas

El modelo organizativo de elementos del juego ha sido un aspecto muy importante del proyecto, y el período de diseño duró toda una semana antes de ser lo suficientemente adecuado como para empezar la implementación.

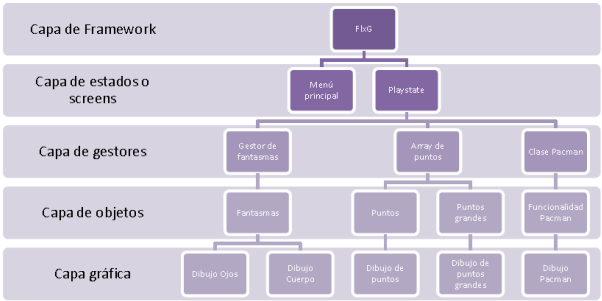


Figura 14: Capas del juego base

El modelo se basa en una serie de **capas abstractas de organización** que cumplen propósitos específicos y organizadas de forma jerárquica. Una representación gráfica se puede observar en la Figura 14.
La capa de **Framework** es, como su nombre indica, la capa de contenido de HaxeFlixel, en la cual se basa el resto del contenido
La capa de **estados** es la capa donde se organizan los distintos estados del juego. Un estado o screeen es una parte del juego que tiene una función especifica y viene acompañado de una interfaz gráfica en concreto. Por ejemplo, el menú principal, el de pausa, la pantalla de highscores y el propio juego serían estados distintos, este último llamado Playstate (estado de juego).
La capa de **gestores** contiene agrupaciones de objetos con entidad y las funciones comunes y de coordinación entre ellos que estos necesitan. Por ejemplo, hay un gestor de fantasmas que tiene referencias a ellos y tiene funciones para liberarlos de la *House*, para que entren en modo *Fright*, para salir de él...O el gestor de puntos, que informa de cuantos quedan por recoger o han sido recogidos.
La capa de **objetos** son los objetos con entidad, como los fantasmas, los puntos o el propio *Pac-Man*. Contiene las funciones y parámetros propios del objeto.
La capa **gráfica** contiene la representación visible del objeto.

El sistema es **dirigido y jerárquico**, por lo que entidades de una misma capa no pueden interaccionar entre ellas.
Por ejemplo, el encargado de comprobar colisiones entre los fantasmas y *Pac-Man* no son los dos implicados, si no que se encarga el primer elemento de capas superiores que tienen en común. En este caso, sería el Playstate, que comprobaría la colisión y en caso de ocurrir avisaría al gestor de fantasmas y a la clase *Pac-Man* para que actuaran adecuadamente.
Esta organización evita muchos problemas derivados de referencias cruzadas y listeners ineficientes, y permite ampliar el juego con facilidad y orden.

B.2. Modelo Fantasma-Modulo

El diseño de los fantasmas tenía que tener bien diferenciada la parte modificable de la no modificable, según lo establecido en las restricciones. Para ello, el modelo diseñado consta de dos clases: la base fantasma y el módulo.

La **base fantasma** incluye los gráficos y las funciones genéricas e invariantes del fantasma. También incluye parámetros de velocidad, animaciones y se encarga del desplazamiento, entre otros. Contiene una referencia a un módulo.

El **modulo** es una clase que funciona como un çerebro". Existe un modulo base que sirve como interfaz e incluye tres funciones; dónde ir cuando se está persiguiendo a *Pac-Man*, dónde ir cuando se está en *Fright* y dónde ir cuando el fantasma está muerto y tiene que resucitar.
Cada fantasma - *Blinky*, *Pinky*... - tiene un modulo personalizado que hereda de esta clase e implementa estas tres funciones según su estrategia particular.

Todos los fantasmas son el mismo "base fantasma", pero al crearlos se les puede **asignar cualquiera de estos módulos personalizados**. Cuando la base fantasma se encuentre en una situación cuya respuesta depende de la estrategia del fantasma, le **pedirá instrucciones al módulo**, logrando que dos implementaciones iguales de un

Figura 15: Pantalla de selección de fantasmas.

fantasma se comporten de forma distinta.

Siguiendo este modo la creación de nuevas IAs es muy fácil, pues consiste únicamente en **crear un nuevo módulo personalizado**, editar las funciones heredadas y, al crear un fantasma, asignárselo.

El juego incluye una **pantalla de selección de fantasmas** donde se puede elegir libremente la personalidad de cada uno de los cuatro fantasmas entre todas las existentes (Figura 15), lugar donde se decide qué módulos se aplican a los cuatro fantasmas.