

# PACMAN.

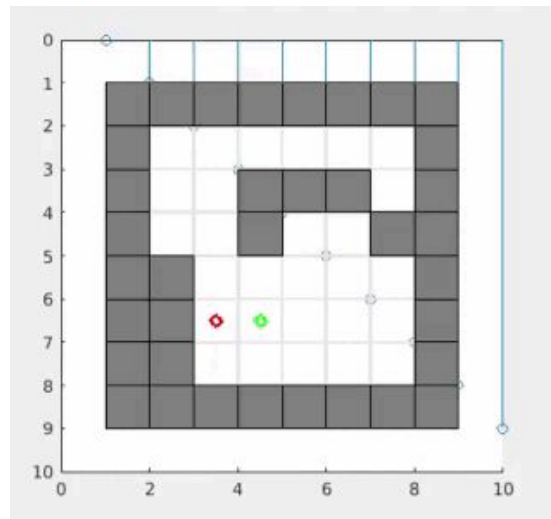
APRENDIZAJE AUTOMÁTICO: APRENDIZAJE  
REFORZADO

Roxana Aanei  
Juan Carlos Manzanares  
Marina Gil  
Cristian Sánchez

# INTRODUCCIÓN

La idea principal del Pacman consiste en diseñar un entorno completamente aleatorio con paredes y obstáculos.


Nuestro agente tiene que avanzar hacia las distintas recompensas para conseguir la mayor puntuación siguiendo el modelo de Aprendizaje por Refuerzo.



# APRENDIZAJE POR REFUERZO

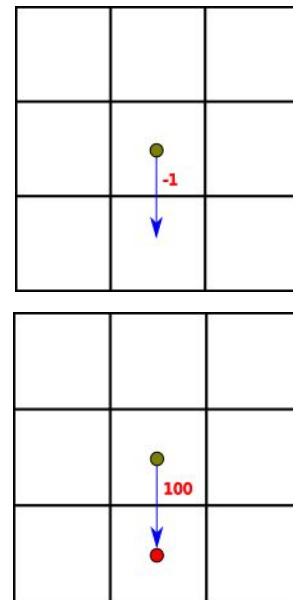
El aprendizaje por refuerzo consiste en dividir el mapa en celdillas (estados) donde cada uno de ellos posee una recompensa distinta.

La idea principal es que el agente consiga la máxima recompensa posible siguiendo el camino más óptimo.

-1000000	-1000000	-1000000	-1000000	-1000000
-1000000		-1	-1	-1000000
-1000000	-1	-1	-1	-1000000
-1000000	-1	-1	100	-1000000
-1000000	-1000000	-1000000	-1000000	-1000000

Cada paso tendrá una penalización de -1.

Conseguir llegar a la comida nos aporta una recompensa 100.

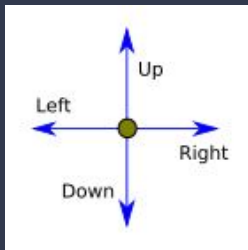


Una vez identificadas todas las transiciones y penalizaciones del mapa, crearemos un modelo utilizando el Proceso de Decisión de Markov para poder entrenar al agente.

# PROCESO DE DECISIÓN DE MARKOV

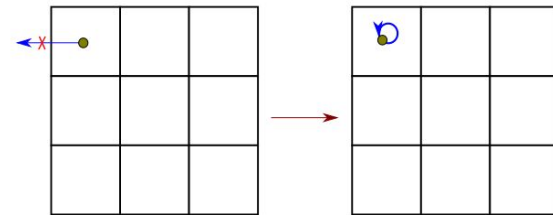
Para crear el modelo utilizando el el Proceso de Markov necesitamos:

- Conjunto de acciones.
- Recompensas y penalizaciones en cada estado.
- Todas las transiciones posibles.



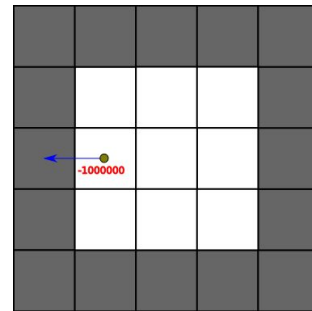
## ENTORNO SIN PAREDES:

Las acciones que transicionen hacia estados inexistentes, harán que el agente se quede en el mismo estado.



## ENTORNO CON PAREDES:

Cada transición hacia una pared tendrá una recompensa muy negativa.

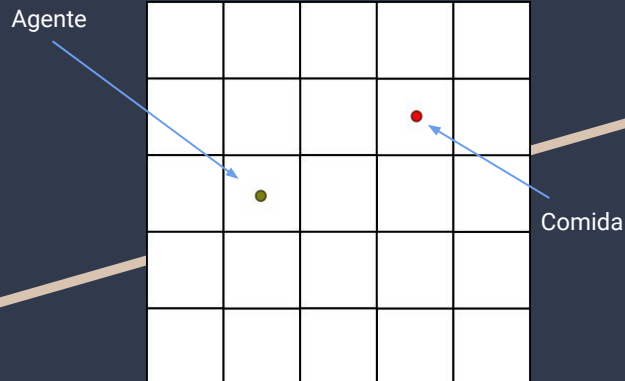


## ELABORACIÓN DEL ENTORNO

# GENERACIÓN DE COMIDAS

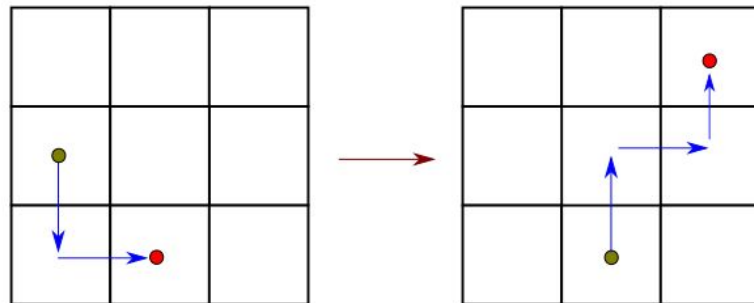
Generación de comidas aleatorias en función del estado inicial de nuestro agente.

`comida != estado_inicial`



Si generamos varias comidas, la posición en donde ha acabado el agente es una posición inválida para generarla.

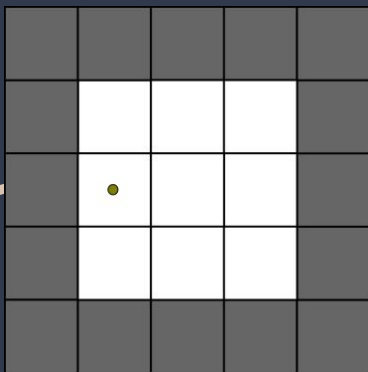
`comida != estado_actual`



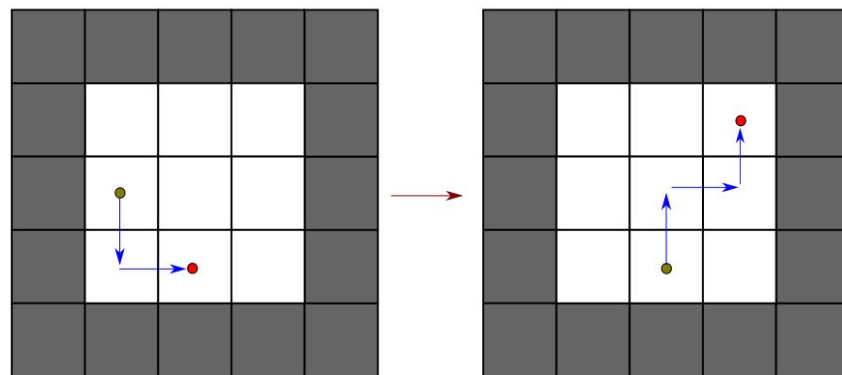
# GENERACIÓN DE PAREDES

Para formarlas, tendremos que tener en cuenta las  $c$  columnas que tenemos:

- Superior  $\rightarrow c$  primeros estados
- Inferior  $\rightarrow c$  últimos estados
- Lateral Derecha  $\rightarrow \text{estado\_siguiente} \% c$
- Lateral Izquierda  $\rightarrow \text{estado\_anterior} \% c$



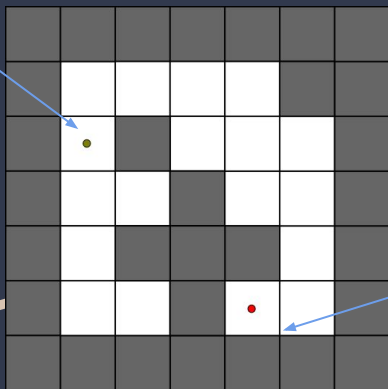
Agente moviéndose por un entorno con varias comidas. Si se choca contra las paredes, recibirá una recompensa muy negativa, además de que estas serán un estado terminal.



# GENERACIÓN DE OBSTÁCULOS

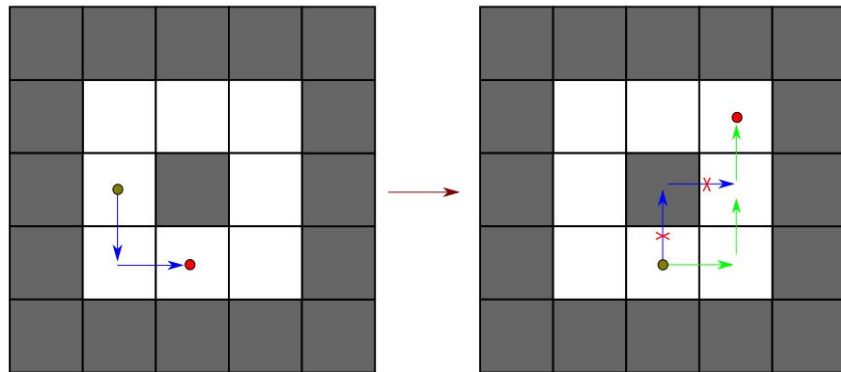
Para ello, habrá que tener en cuenta las paredes del entorno y generar obstáculos aleatorios.

Agente



Comida

Volviendo al caso anterior, ese obstáculo le impide seguir el mismo camino por lo que tendrá que llevar a cabo otro distinto, que en algunos casos será más largo.





# DETALLES ESPECIALES

## Explicación de agente “ciego”:

El agente conoce las posibles acciones a emplear para su estado actual, pero desconoce el estado del resto de posiciones del entorno.

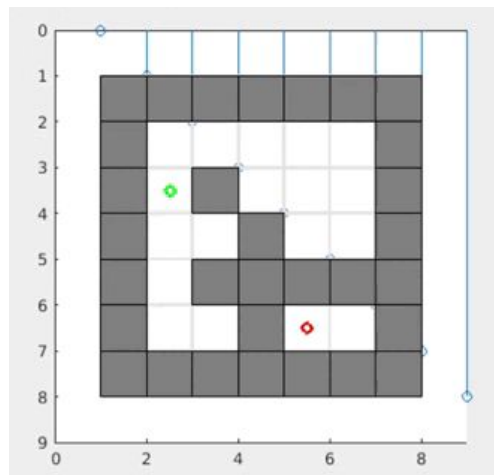
Es decir, si le entrenamos para llegar hasta un estado recompensa X, solo sabrá llegar hasta X.

Con cada nueva comida generada, deberemos crear un nuevo modelo utilizando el Proceso de Markov y entrenar al agente.

## Caso de estado final inalcanzable:

Como la generación de obstáculos es aleatoria, se puede dar la casualidad de que el agente no es capaz de alcanzar la recompensa.

El agente se moverá sin rumbo lógico evitando las paredes, porque es la mayor recompensa posible que puede obtener.



# REPRESENTACIÓN

A dark blue diagonal shape, resembling a thick line or a wedge, starts from the bottom-left corner and extends towards the top-right corner, covering the lower half of the image. The background is white.

# REPRESENTACIÓN DEL PACMAN

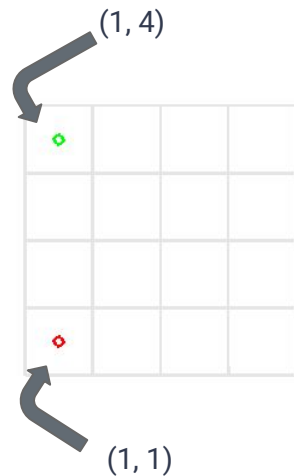
Para representar el Pacman y todos los elementos del entorno, nos creamos una matriz cuyos elementos son los estados del modelo.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

El problema principal es que [fila, columna] del estado no coincidía con (x, y) en coordenadas.


Por ejemplo:

- Estado 1:
  - Coordenadas (1, 4)
  - Fila, columna (1, 1)



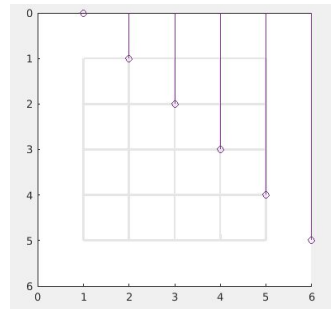
**Solución 1:** Voltear el mapa.

(1, 4)



13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

**Solución 2:** Decrementar el eje Y.

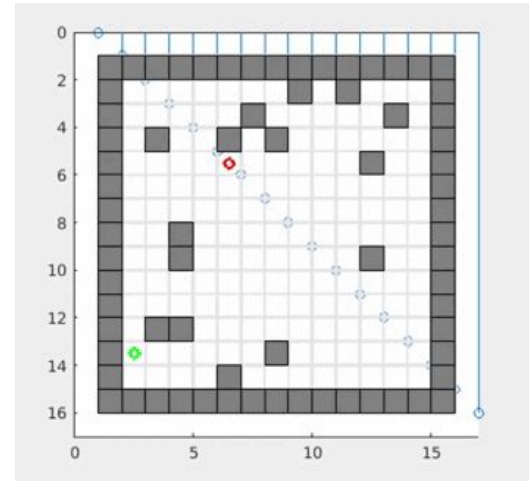
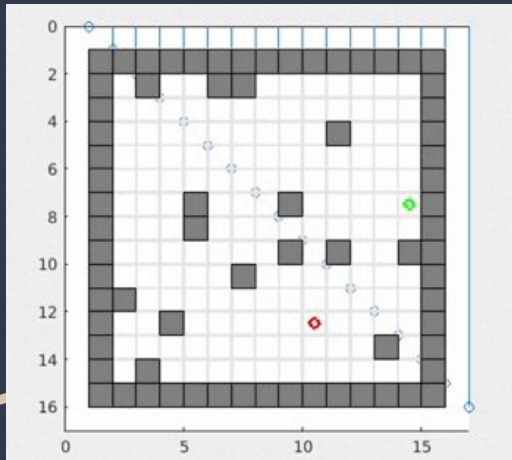


A 10x10 grid with a 6x6 white center and a gray border. A green diamond is at (3, 4.5), a red diamond is at (7.5, 7.5), and blue diamonds are at (1, 0), (2, 1), (3, 2), (4, 3), (5, 4), (6, 5), (7, 6), (8, 7), (9, 8), and (10, 9).

# MAPA GRANDE

Con un mapa de 15x15, tenemos un total de 225 estados.

En dicho mapa, según estén posicionados pacman y recompensa, puede dar lugar a bloqueo o a éxito.

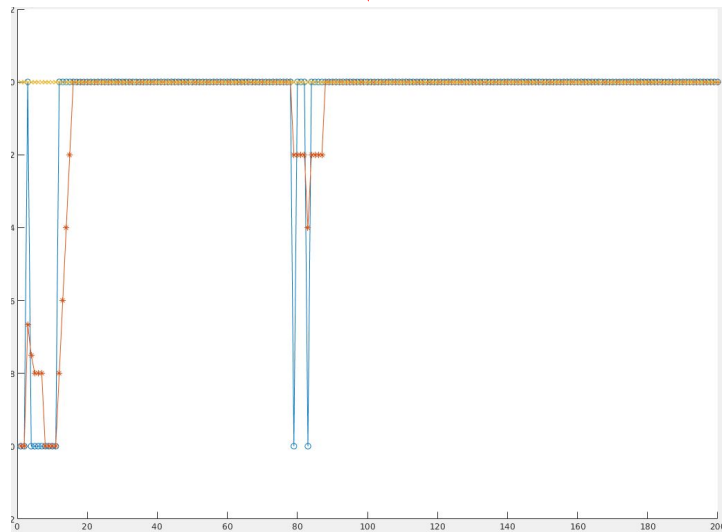
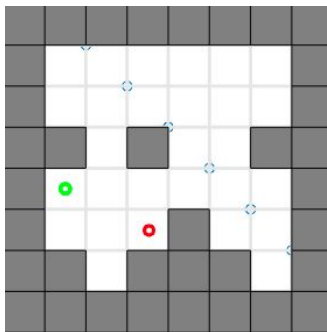


## ANÁLISIS DE LOS RESULTADOS

# GRÁFICAS DE ENTRENAMIENTO (I)

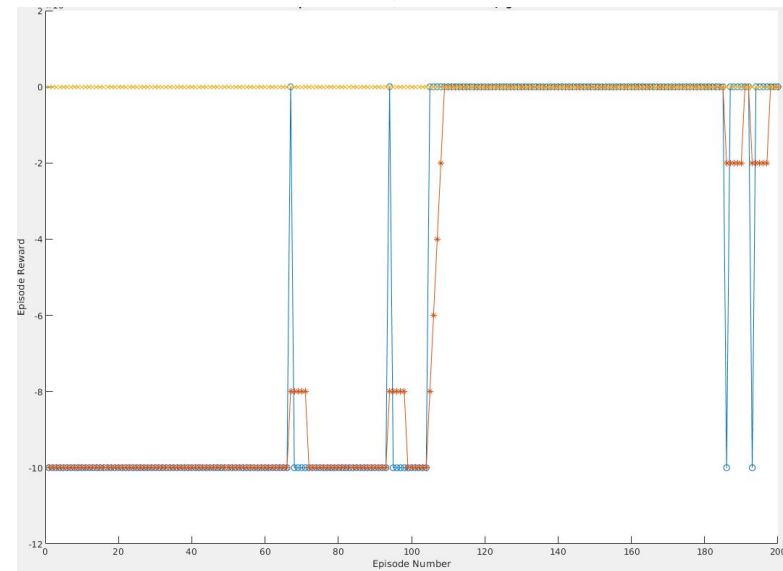
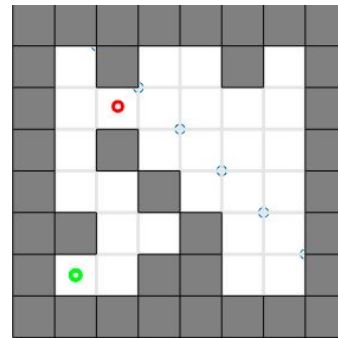
Alcanzar la comida es sencillo, el agente solo tiene que cruzar una línea recta.

En el entrenamiento se ve como alcanza la recompensa rápidamente sin demasiadas iteraciones.



# GRÁFICAS DE ENTRENAMIENTO (II)

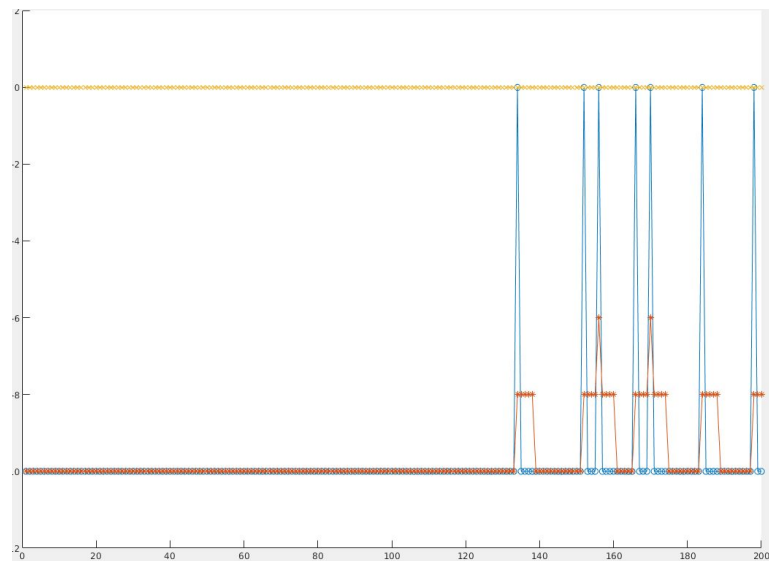
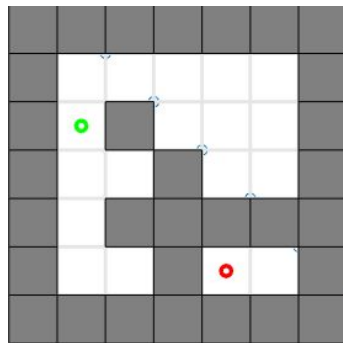
En este caso vemos cómo para alcanzar la comida el agente debe tomar un camino complejo y laberíntico, tardando más en encontrar la ruta.





# GRÁFICAS DE ENTRENAMIENTO (III)

En este caso vemos cómo el acceso a la comida está obstaculizado, siendo el agente incapaz de alcanzar la recompensa.



# CONCLUSIONES

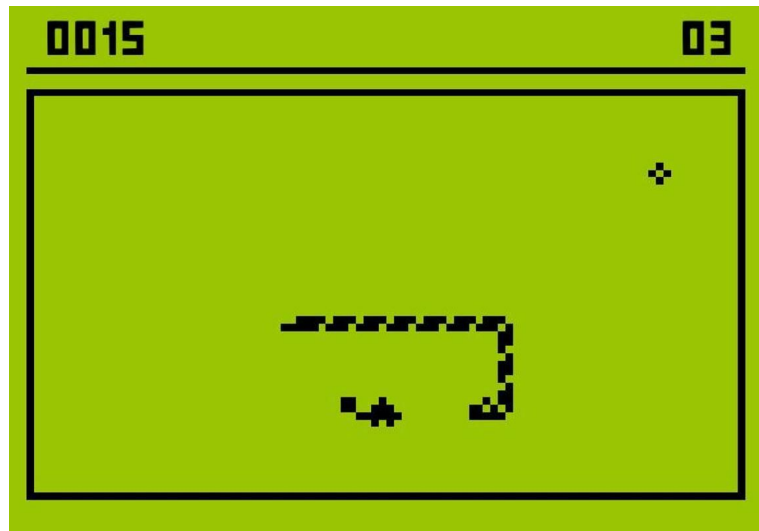
A través del aprendizaje por refuerzo, se genera un agente que toma decisiones dentro de un entorno regulado, y consigue resolver un problema de forma eficiente.

## ¿Snake?

Este proyecto nació con la idea de resolver el snake, por tanto sienta las bases para extender su funcionalidad.

## Posibles mejoras:

- + Inputs
- - Entrenamientos (pero más completos).



**FIN**