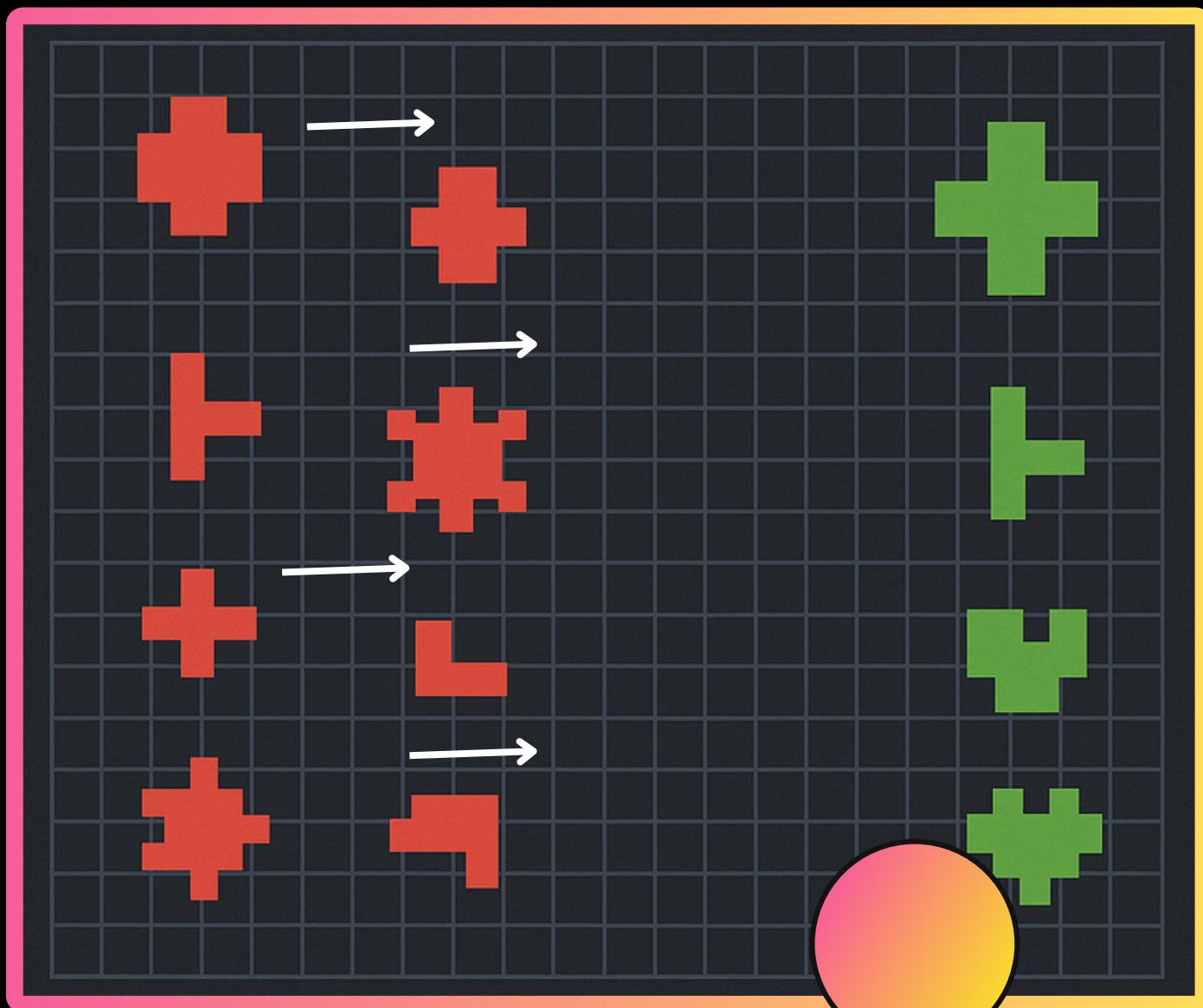




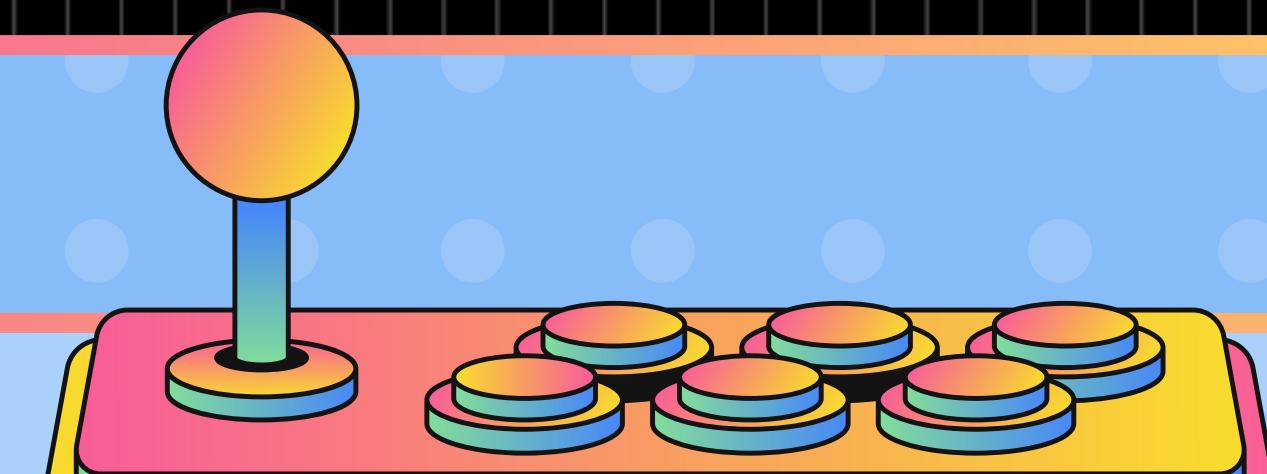
DESCRIPCIÓN DEL PROYECTO



- Desarrollado con Python y Pygame.
- Basado en los principios del Juego de la Vida de Conway y los autómatas celulares.
- Simula una batalla entre células buenas y células patógenas.
- A través de reglas similares a las de los autómatas celulares, las células combaten y desaparecen según su entorno.

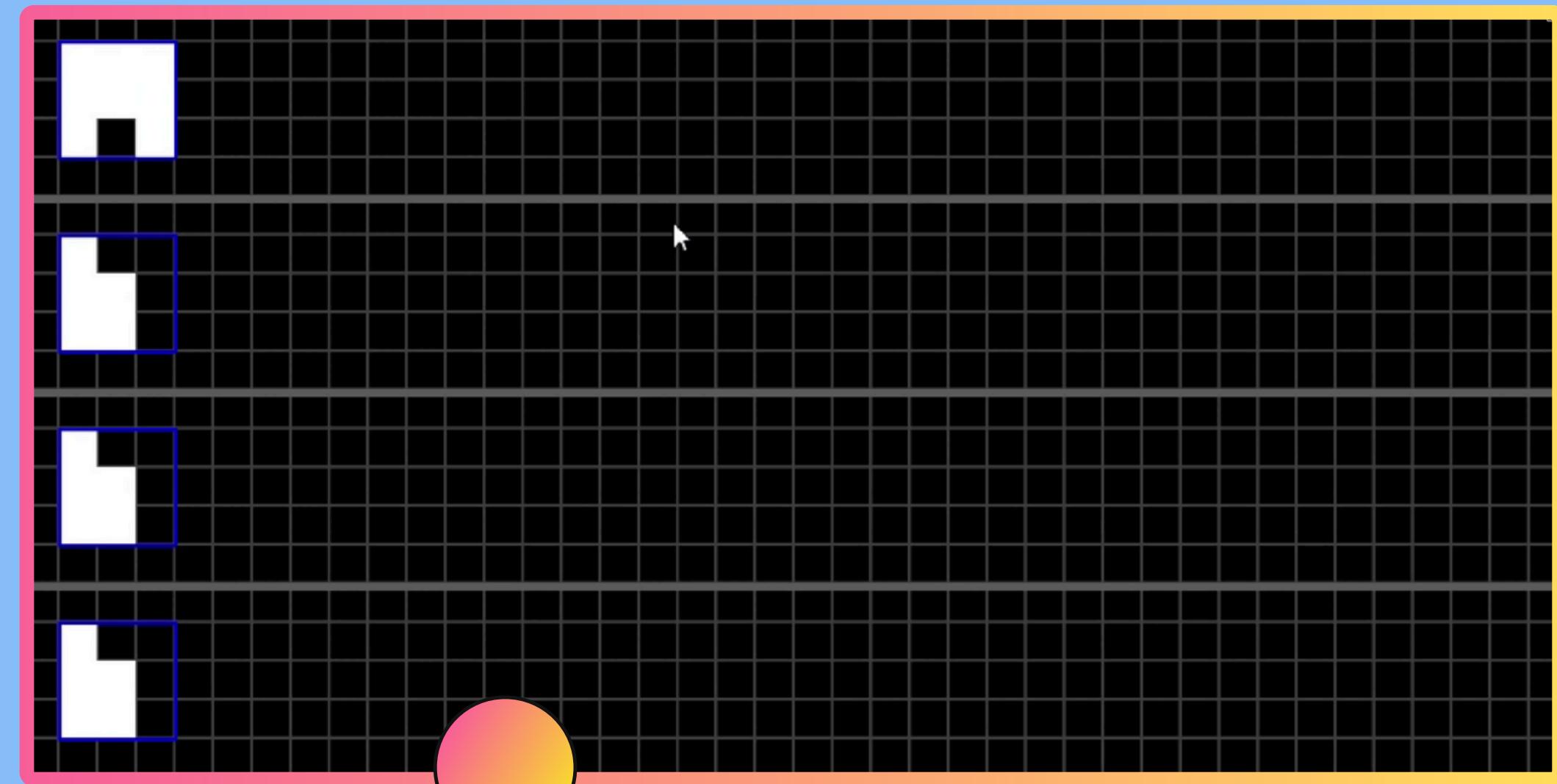
DISEÑO DEL JUEGO

El juego está conformado por un tablero cuadriculado con cuatro carriles horizontales.



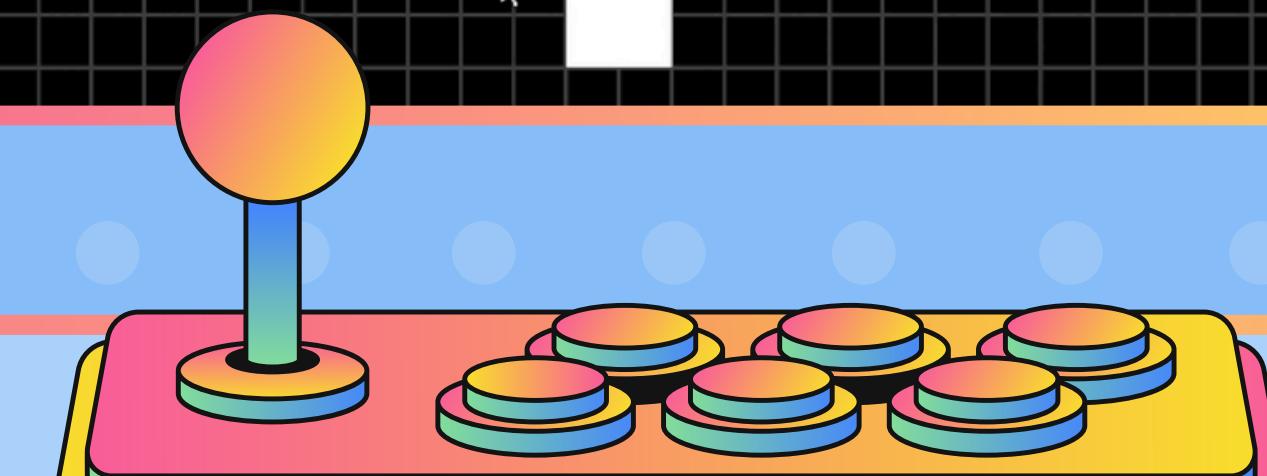
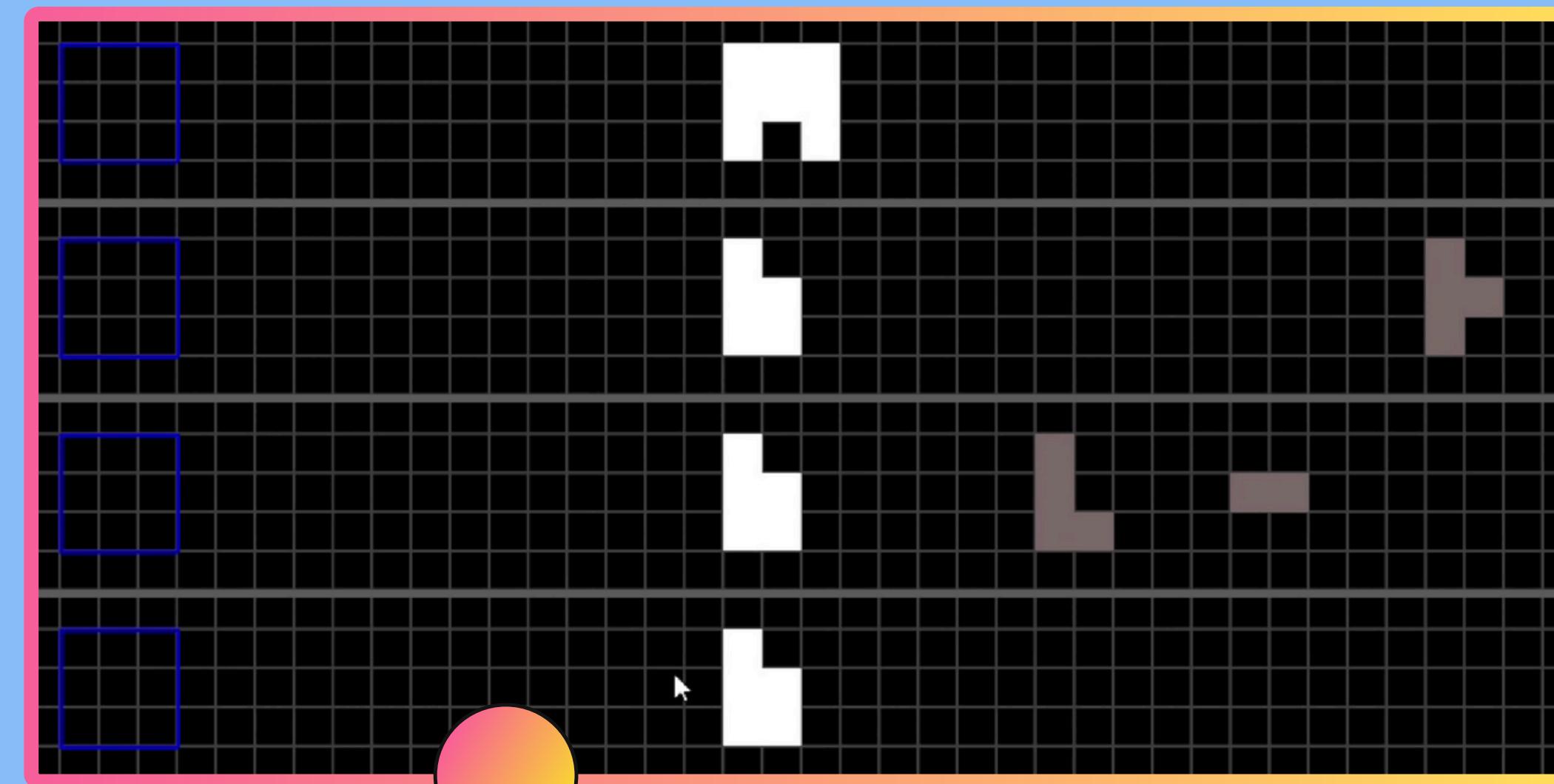
DISEÑO DEL JUEGO

Del lado izquierdo del tablero, el usuario dibujará las células “buenas” que enfrentarán a las patógenas.



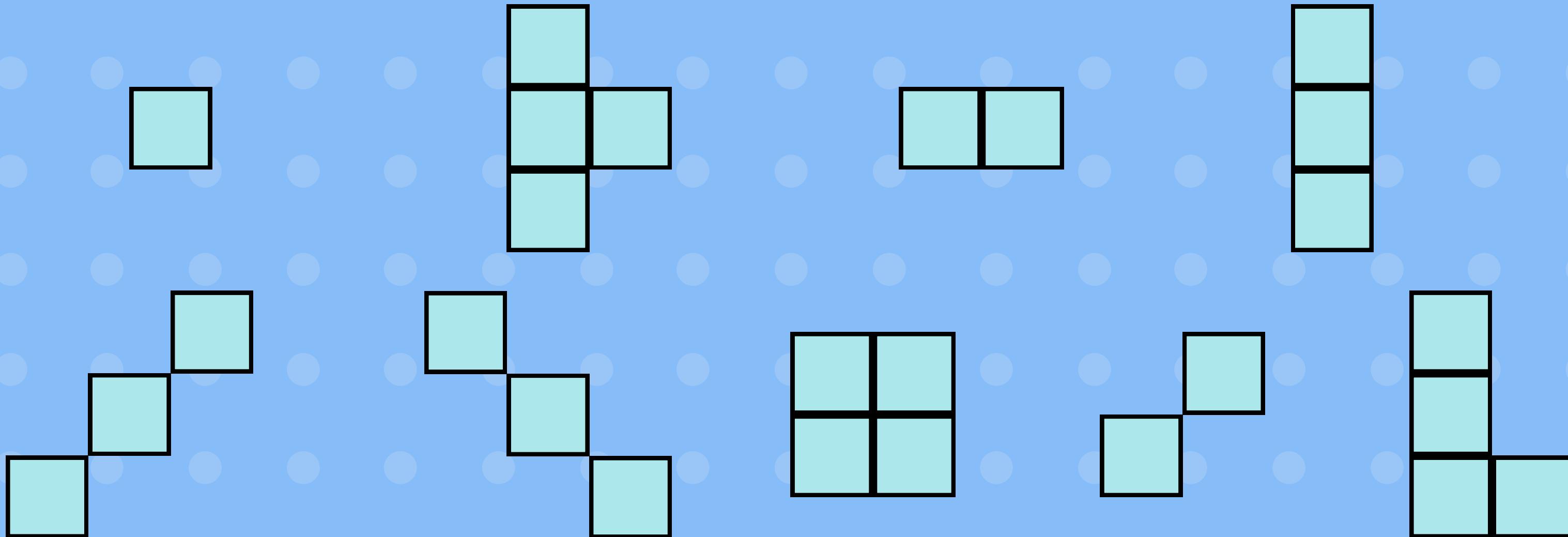
DISEÑO DEL JUEGO

Del lado derecho del tablero, irán apareciendo las células patógenas de manera aleatoria.



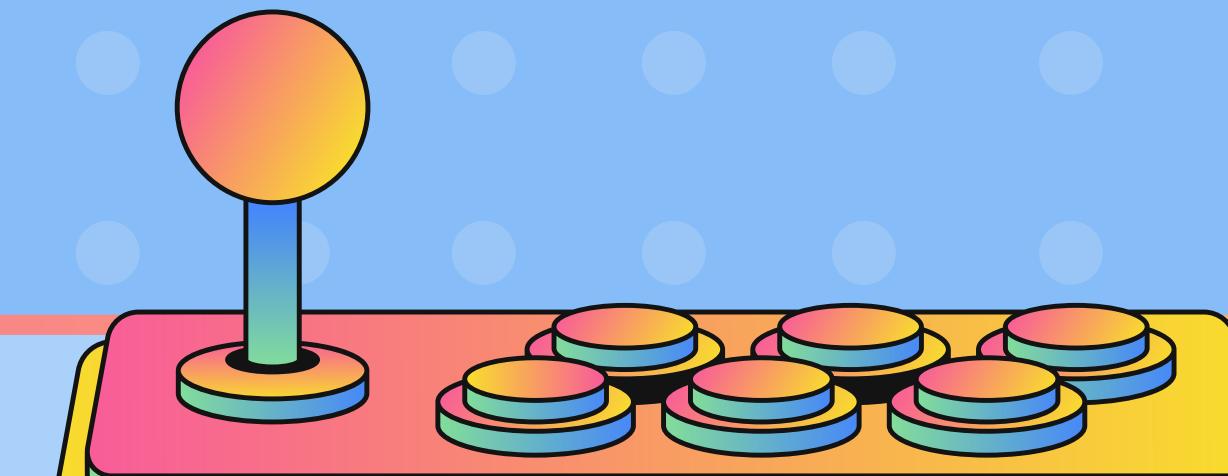
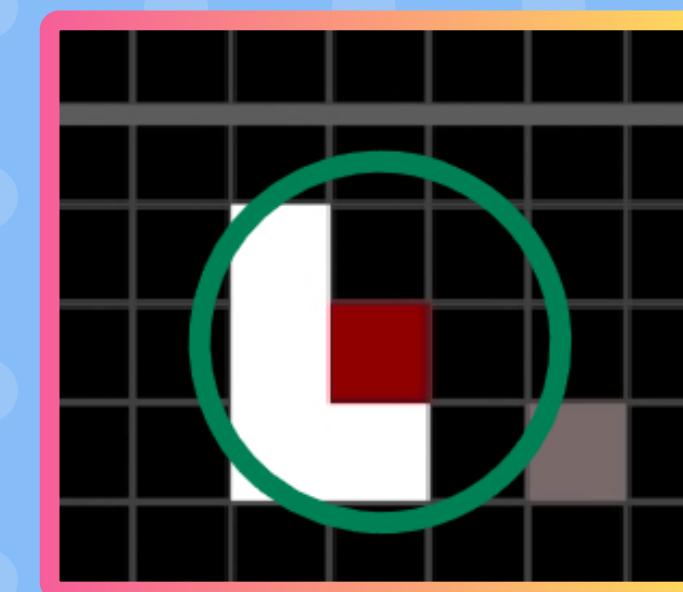
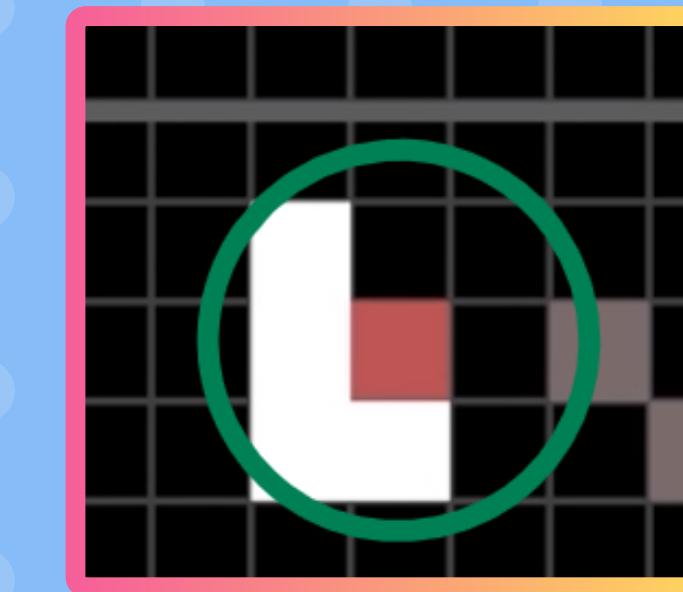
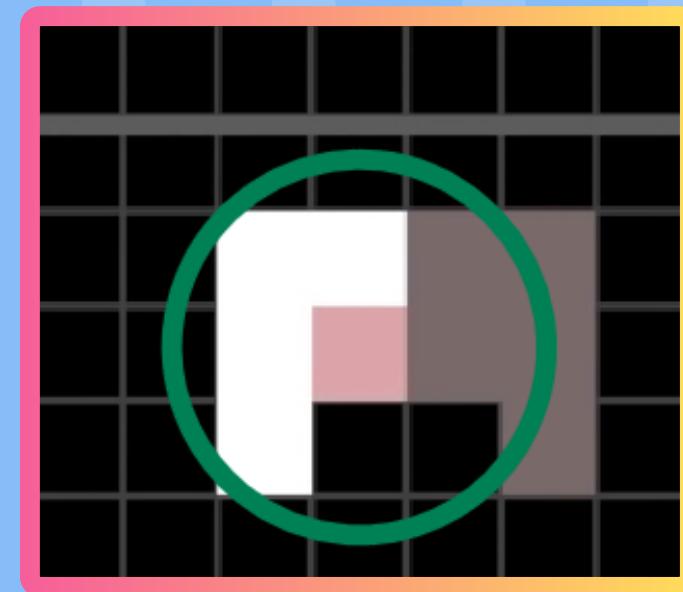
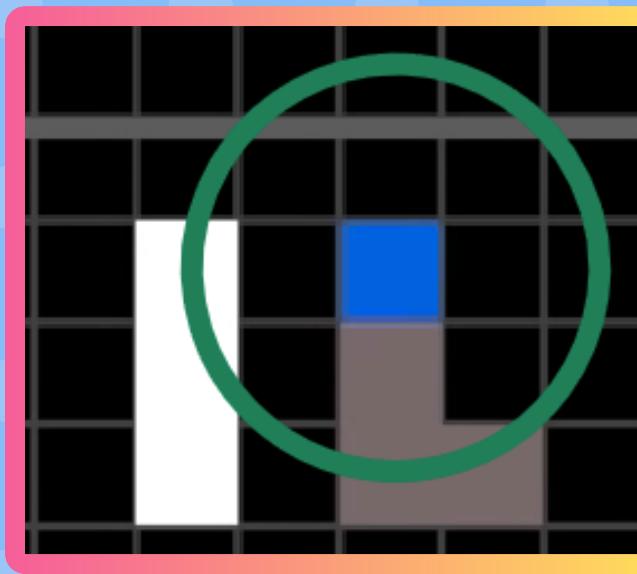
DISEÑO DEL JUEGO

Células que conforman el juego.



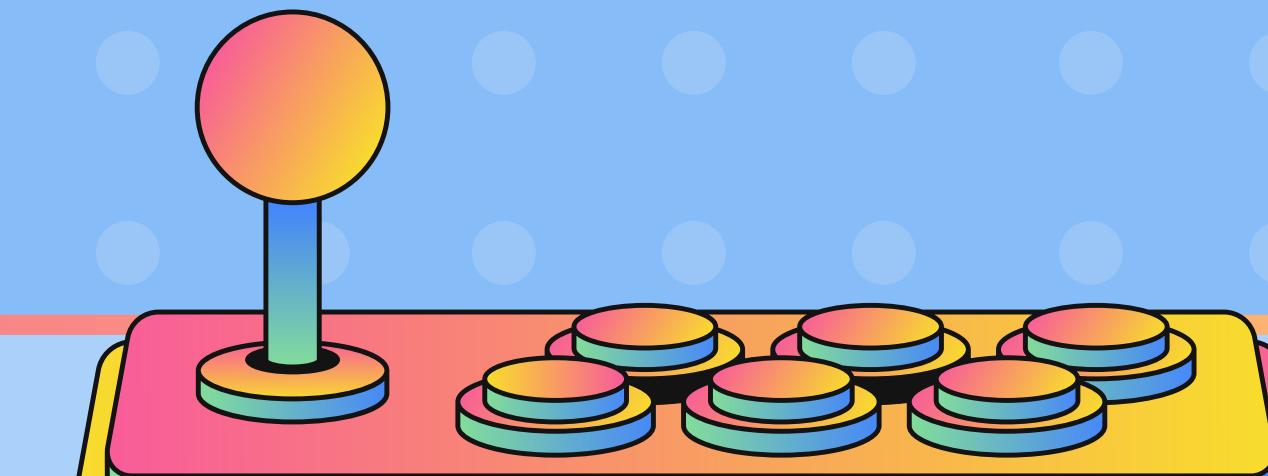
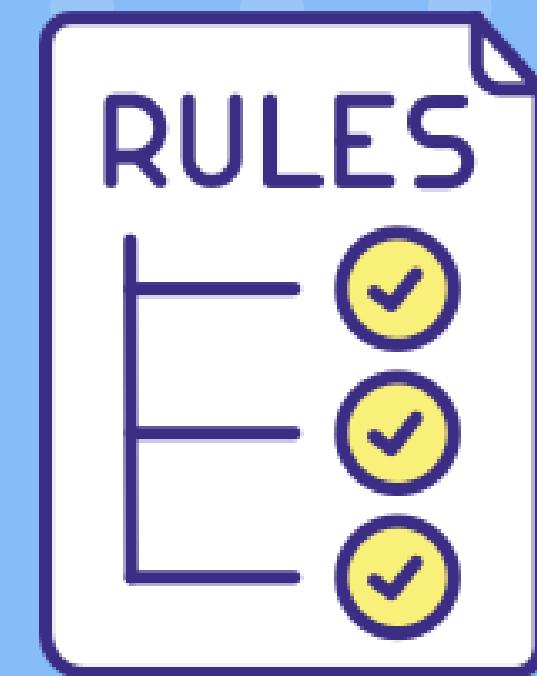
DISEÑO DEL JUEGO

La interacción entre las células buenas y patógenas produce cierto desgaste en las células, el cual se representa con un cambio en el color



REGLAS DEL JUEGO

- 1). El jugador dibuja células “buenas” con el mouse
- 2). Las células “malas” aparecen automáticamente
- 3). Las células combaten al encontrarse
- 4). Las células tienen una barra de vida
- 5). El jugador da inicio al juego con un botón



IMPLEMENTACIÓN

El desarrollo del juego tiene dos clases principales: La clase Cell y la clase Operación algunos de sus métodos principales son:

```
def cells_fight(good_cell, bad_cell) -> any:

    good_cell_warriors = 0
    good_cell_max_x = max(dx for dx,dy in good_cell.pattern)
    for i in range(len(good_cell.pattern)-1, -1, -1):
        if (good_cell.pattern[i][0] < good_cell_max_x):
            break
        good_cell_warriors += 1

    bad_cell_warriors = 0
    for i in range(len(bad_cell.pattern)):
        if bad_cell.pattern[i][0] > 0:
            break
        bad_cell_warriors += 1

    to_erase = []

    if good_cell_warriors > bad_cell_warriors:

        for w in range(bad_cell_warriors):
            point = bad_cell.pattern.pop(0)
            bad_cell.damage_bitmap.pop(0)
```

IMPLEMENTACIÓN

El desarrollo del juego tiene dos clases principales: La clase Cell y la clase Operación las cuales de los métodos principales son:

```
# ENEMIGOS
    for i in range(4):
        for enemy in enemies[i]:
            cell_i = enemy.detect_cells(good_cells[i], enemies[i])
            if enemy.next_move == Cell.MOVE:
                enemy.erase(world)
                enemy.move_left()
                if enemy.x <= 0:
                    running = False
                    game_over = True
                    break

            elif enemy.next_move == Cell.FIGHT:
                good_cell = good_cells[i][cell_i]
                enemy.erase(world)
                to_erase = Cell.cells_fight(good_cell, enemy)
                if enemy.pattern == []:
                    enemies[i].remove(enemy)
                if good_cell.pattern == []:
                    good_cells[i].pop(cell_i)
                for x,y in to_erase:
```

RELACIÓN CON LA TEORÍA

El concepto de “Estado”:

- *La clase Cell representa cada celda del tablero*
- *Cada celda tiene un tipo: buena, mala o vacía*
- *Tiene atributos como vida, color y posición*

Vecindad y entorno:

- *Las células interactúan con otras que están cerca (colisión o proximidad).*
- *Esto es similar a la “vecindad de Moore” en los autómatas celulares (vecinos cercanos).*



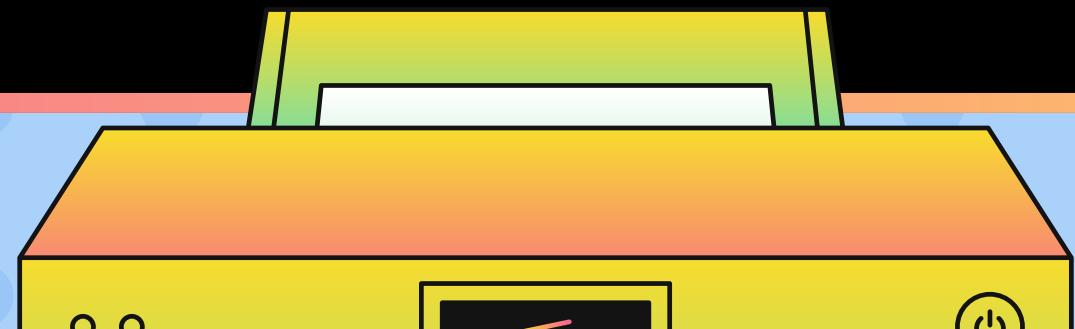
RELACIÓN CON LA TEORÍA

Reglas de evolución:

- *Si dos células se cruzan → combaten*
- *Si una muere ($\text{vida} \leqslant 0$) → desaparece*
- *Las células malas avanzan solas → movimiento automático*

Simulación basada en tiempo (ticks):

- *Cada "frame" del juego es como una generación del autómata.*
- *En cada frame se actualizan las células.*

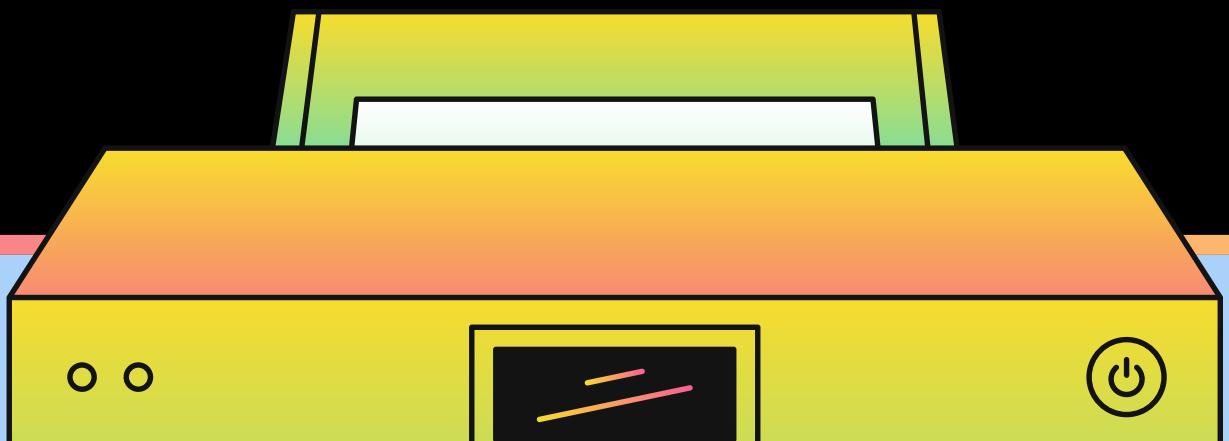


RESULTADOS

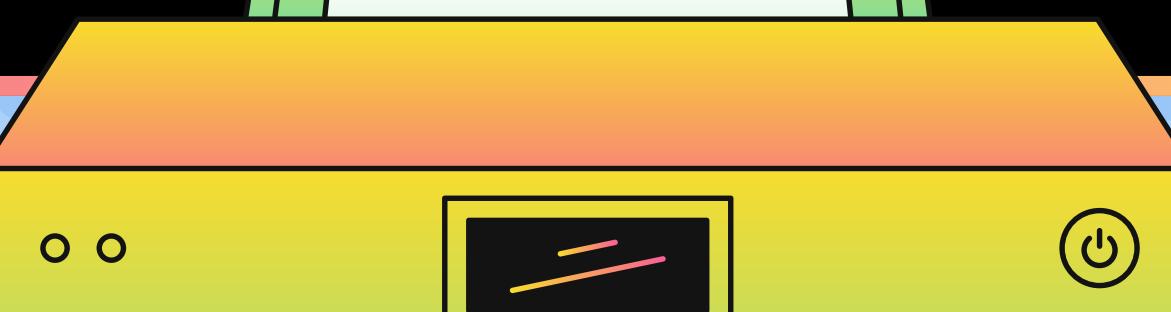
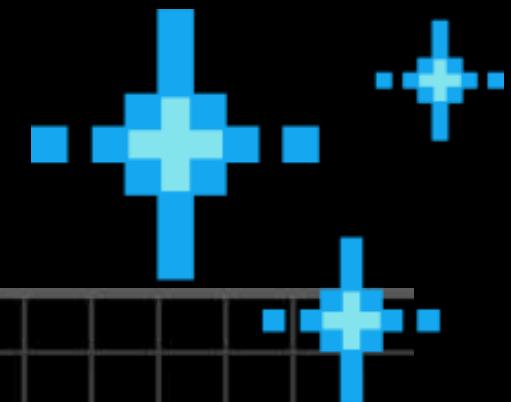
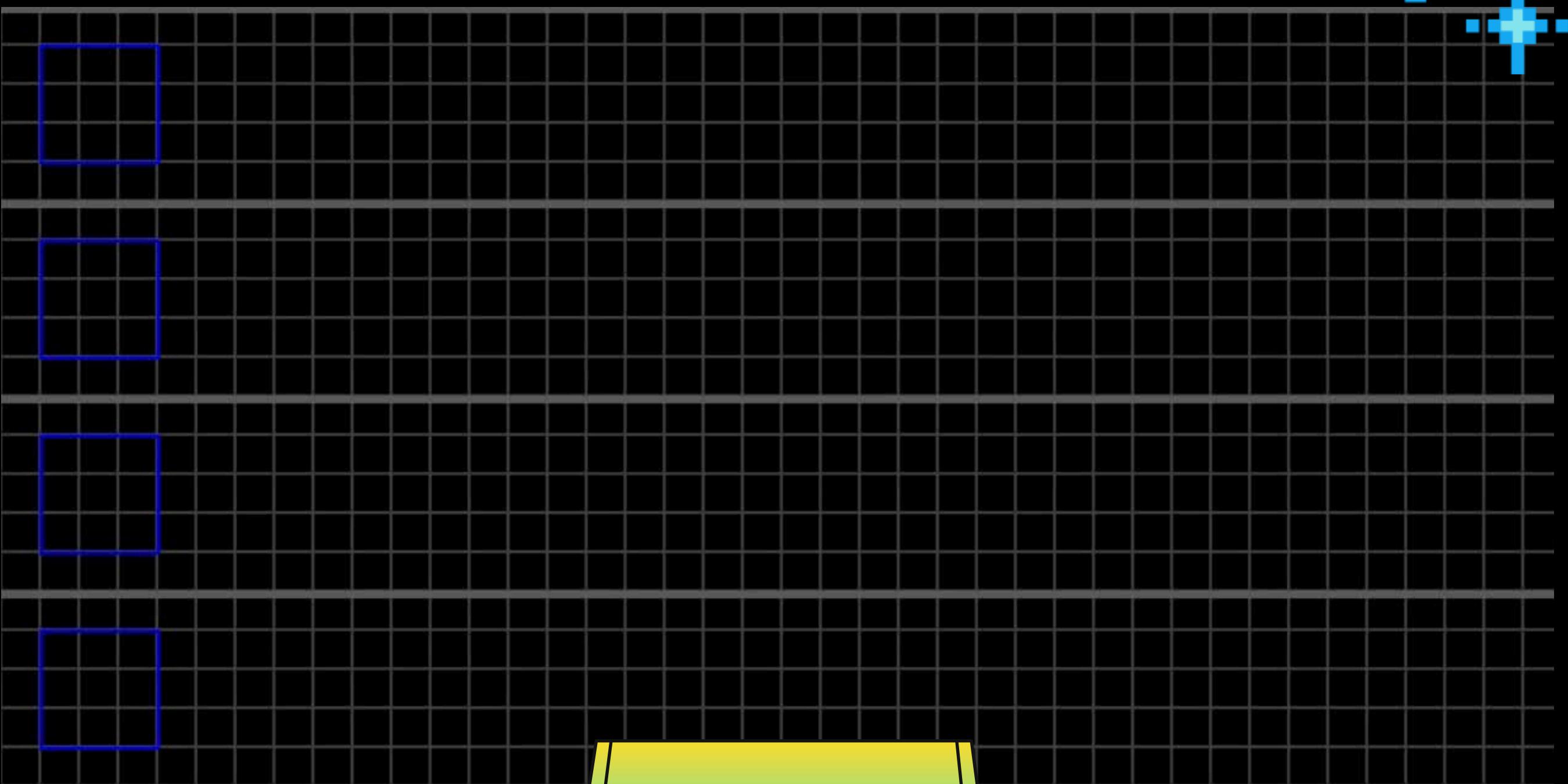
**FUNCIONAMIENTO EXITOSO
DEL SISTEMA DE JUEGO.**

**LÓGICA DE JUEGO BASADA EN
AUTÓMATAS CELULARES.**

INTERACCIÓN DEL USUARIO.



RESULTADOS



CONCLUSIONES

1. Aplicación efectiva de la teoría computacional

El proyecto permitió poner en práctica conceptos fundamentales como los autómatas celulares, estructuras de datos dinámicas y lógica basada en reglas, reforzando la conexión entre teoría y desarrollo práctico.

2. Desarrollo de pensamiento algorítmico y resolución de problemas

Durante la creación del juego se resolvieron desafíos relacionados con movimiento, combate, renderizado visual y sincronización, fortaleciendo la habilidad para pensar de manera lógica y estructurada.

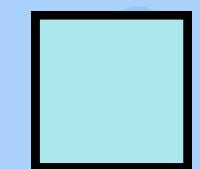
3. Simulación visual de procesos complejos

Se logró representar de forma interactiva fenómenos como el conflicto entre entidades celulares, haciendo más comprensibles y visuales conceptos abstractos como comportamiento emergente y evolución de sistemas.

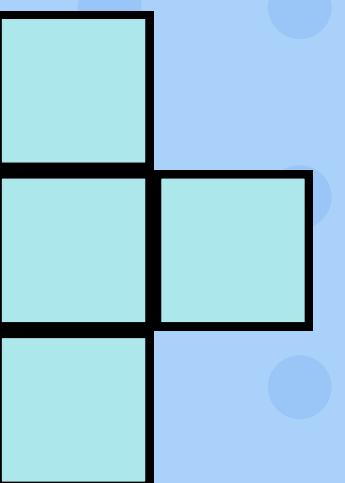
GRACIAS!!!



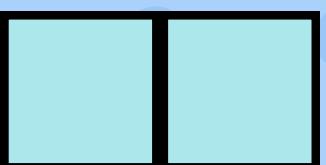
1



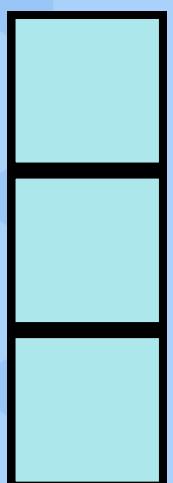
2



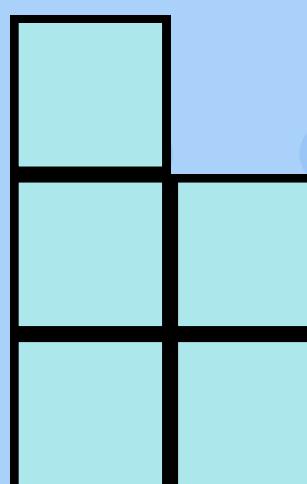
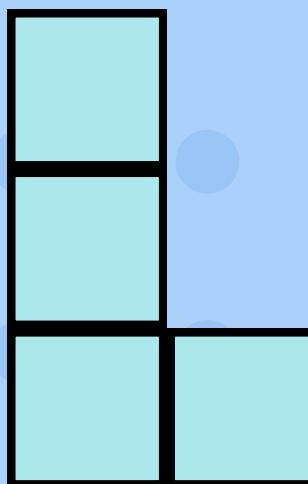
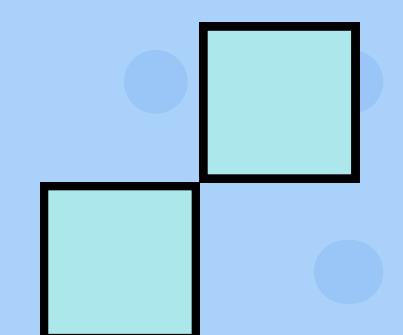
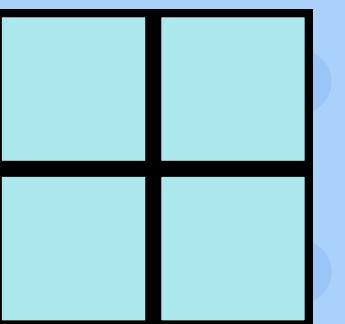
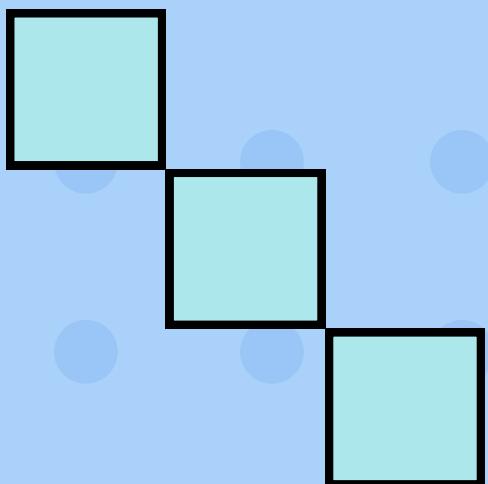
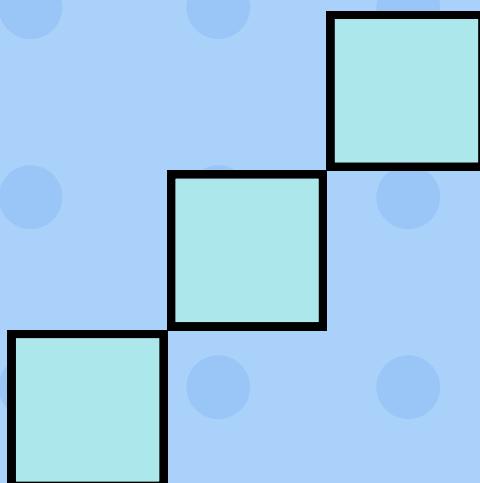
3



4



5



6

7

8

9

0