



## **Proyecto 1**



## **Sincronización**

**Bryan Mejia Ramos**

**Sistemas Operativos**

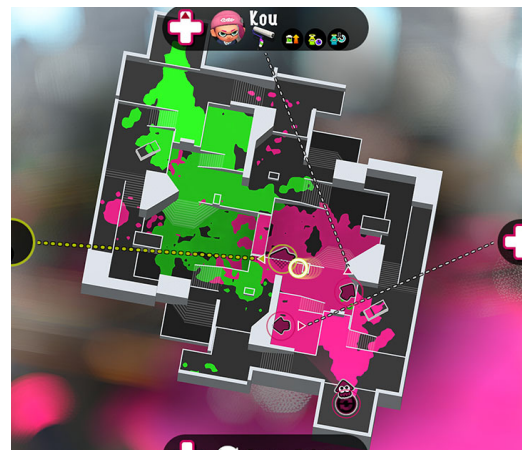
**Profesor: Gunnar Eyal Wolf Iszaevich**

En los videojuegos podemos encontrar muchas situaciones que deben sincronizarse debido a la gran cantidad de personas usando los mismo recursos, recursos compartidos, en lo personal me interesa mucho ese mundo y los algoritmos e ingenios que se inventan para resolver problemas, con el fin de brindar un entorno interactivo e inmersivo por eso quise enfocar el proyecto en estos casos de la vida real.

Hisashi Nogami creo en Nintendo una franquicia de juegos llamada Splatoon, un shooter dónde el objetivo es pintar la mayor cantidad de campo de batalla para poder ganar, mi implementación está inspirada en este juego pero simplificando y enfocado en ver las relaciones entre el recurso compartido, que es el terreno junto a los jugadores que serían los hilos.

### Situación a modelar:

Simulare el ingreso a una partida y el juego en si, se trata de un juego multijugador, los jugadores ingresan a la sala de espera para iniciar la partida, en cuanto entran se encuentran un campo donde su objetivo es pintar el terreno hasta terminar el tiempo de partida, al final se hace recuento de puntos y se determina el ganador.

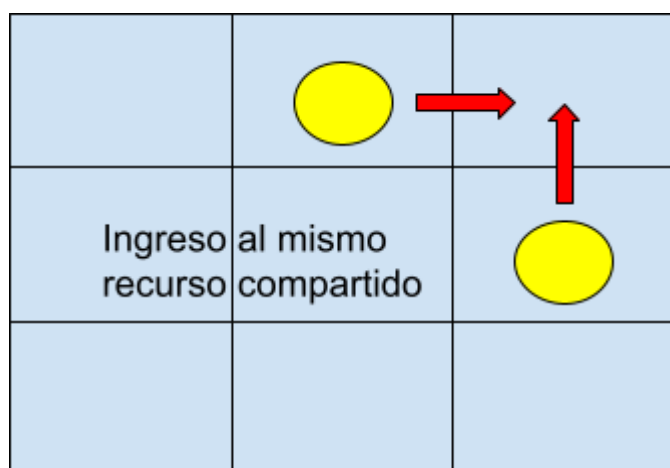


Reglas:

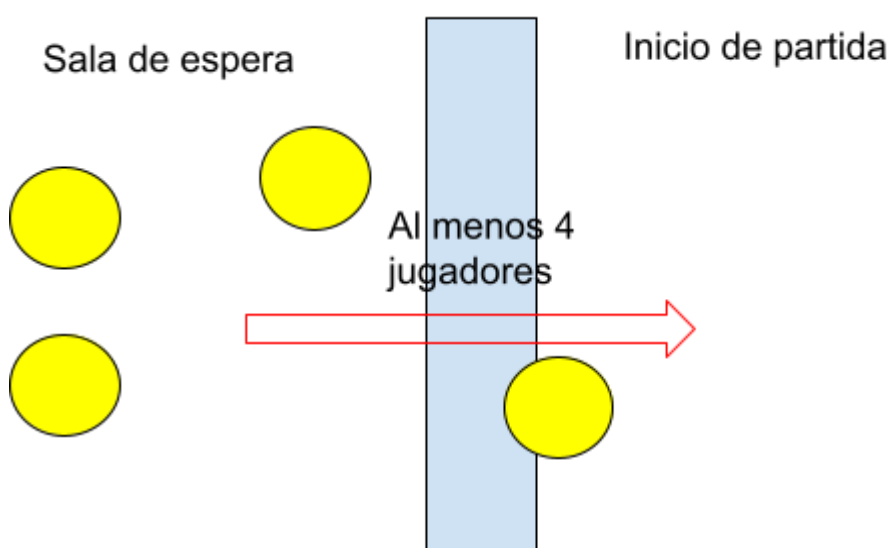
- Los jugadores primero entrar a una sala de espera hasta juntar el número requerido de jugadores, cuando esto sucede, los jugadores entran el campo de batalla
- Al ingresar al campo de batalla los jugadores contarán con una cantidad de movimientos limitados para pintar el terreno
- Los jugadores solo pueden avanzar hacia adelante, atrás , derecha e izquierda, seleccionando la dirección de forma aleatoria.
- Se debe realizar el recuento de puntos para determinar el ganador.
- Solo puede pintar un jugador a la vez determinada zona del mapa.

### Eventos a controlar hablando en términos de concurrencia:

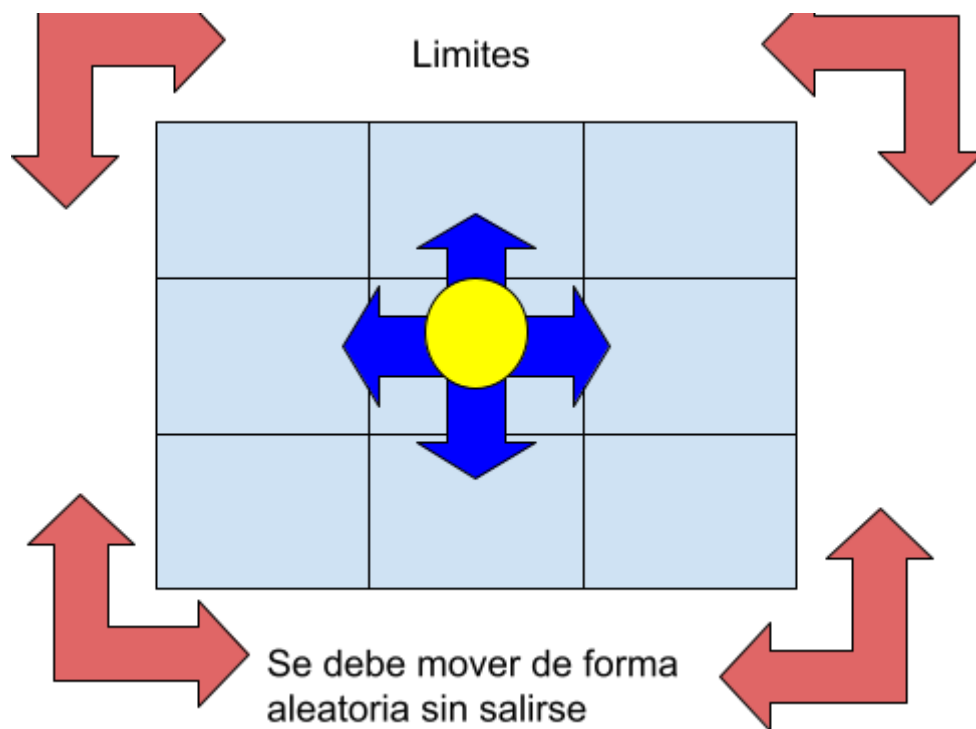
Durante el juego pueden ocurrir diferentes condiciones que queremos evitar.  
Condición de carrera.



Ingreso a la misma zona del mapa, y posible sobre escritura indeseada.  
Ingreso no simultáneo al juego.

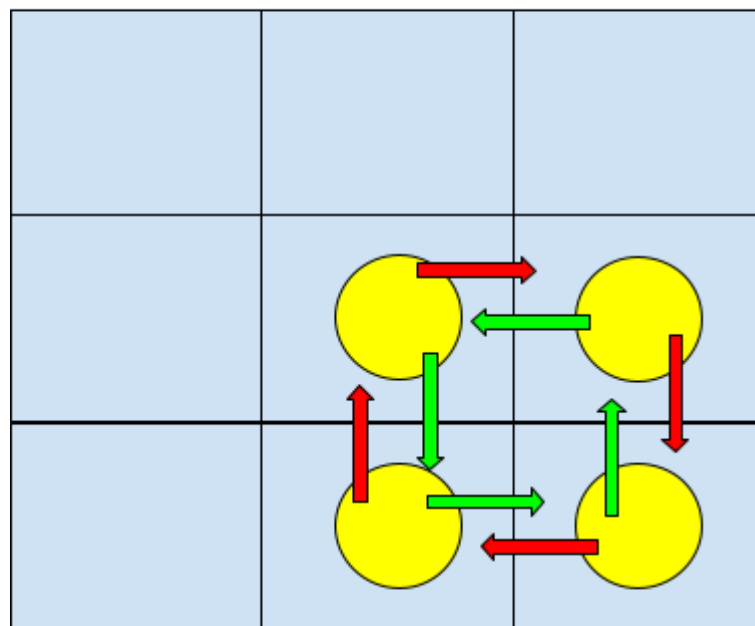



Rebasar los límites del mapa.



Bloqueo mutuo.

Se necesitan al menos 4 hilos



Recurso asignado 

Recurso solicitado 

**Descripción de los mecanismos de sincronización empleados:**

Para resolver el problema planteado se eligió construir dos matrices, una para pintar y tener el registro de las zonas que cada jugador había pintado, se pinta utilizando el id del hilo, la otra matriz está compuesta de semáforos con el fin de representar cada espacio del mapa, haciendo así que solo un hilo pueda entrar a dicha zona.

Para escribir los datos de la matriz que lleva el conteo de zonas pintadas se hace uso de un semáforo como mutex.

También se usa un mutex para la impresión de información en consola, ya que durante el desarrollo presentó problemas porque la información se solapaba.

Para el ingreso de los jugadores y la sala de espera se hizo uso de un torniquete, obviamente usando semáforos, esto con el fin de tener un orden de entrada y que todos los hilos tuvieran las mismas oportunidades de ganar.

Las direcciones que toma cada hilo, y la posición inicial que toma, son realizados de forma aleatoria.

### **Lógica de operación :**

El programa consta de variables generales, variables compartidas, la función printmapa, la función contar puntos y la función player

#### **Variables generales:**

numSpace : Se usa para asignar el tamaño de juego

numPlay Numero de jugadores

barreraParaJugar : Barrera para inicio y final de función player

mutexDatos : mutex para modificar de forma segura la matriz donde se registran las zonas pintadas

mutexPrint = mutex para imprimir sin interrupciones datos en consola.

- **Identificación del estado compartido (variables o estructuras globales)**

El programa cuenta con variables globales las cuales son:

mapaJuegoProteccion : es una lista de semáforos que nos ayuda con la condición de carrera.

mapaJuego : es la lista de listas, que inicia llena de 0 y que representa nuestro escenario para el juego

jugadores y jugadores2: variables auxiliares para el uso de las barreras

- **Descripción algorítmica del avance de cada hilo/proceso**

Los hilos en el programa podría decirse que se comportan de forma constante ya que la cantidad de cálculos computacionales requeridos es muy baja.

- **Descripción de la interacción entre ellos (sea mediante los mecanismos de sincronización o de alguna otra manera)**

Los hilos que representan a los jugadores, se comunican con una barrera, con el fin de una ejecución más justa y ordenada.

Se protegen las zonas críticas del código con mutex y semáforos.

El contador de puntos se llama al final de la función cuando todos los hilos acabaron sus movimientos.

- **Descripción del entorno de desarrollo, suficiente para reproducir una ejecución exitosa**

El programa fue desarrollado en el lenguaje de programación python 3 utilizando el navegador para programar.

Puede ejecutarse con éxito desde Google colab o OnlineGDB python 3 o desde terminal.

Las únicas bibliotecas utilizadas fueron de las que podemos considerar estándar de python, siendo específicos threading y random, nada mas !

- El programa fue ejecutado en un navegador en Google Colab, en el sistema operativo de windows.

- **Ejemplos o *pantallazos* de una ejecución exitosa**


Ejecución del programa:

Se crean los hilos y se mantienen esperando

```

Esperando jugadores...
Esperando jugadores...
Esperando jugadores...
Esperando jugadores...
Listo, pueden jugar!
1: Posicion inicial = 10
2: Posicion inicial = 5
3: Posicion inicial = 3
4: Posicion inicial = 12

```


 Zona de espera, uso de la barrera.  
 Los jugadores están completos y entran al juego

Se imprimen el mapa de forma matricial:

```

[4, 0, 0, 0]
[4, 2, 0, 0]
[4, 0, 1, 0]
[4, 0, 0, 0]

```

0 - Zona sin pintar.  
 1 - Zona pintada por hilo 1  
 2 - Zona pintada por hilo 2  
 3 - Zona pintada por hilo 3  
 ...

```

[4, 0, 0, 2]
[4, 2, 2, 2]
[4, 0, 1, 2]
[4, 0, 0, 2]

```

En lo que se imprime el mapa, los hilos avisan cuando terminan de jugar todos sus movimientos:

**2: Termino sus movimientos**

```

[1, 0, 0, 3]
[1, 2, 2, 3]
[1, 1, 1, 3]
[1, 1, 0, 3]

```

**1: Termino sus movimientos**

Al final después del recuento de puntos se anuncia el ganador.

```

[1, 0, 0, 3]
[1, 2, 2, 3]
[1, 1, 1, 3]
[1, 1, 0, 3]

```

**3: Termino sus movimientos**

**El ganador es Hilo: 1 con 7 puntos !**

Se informa el número de puntos que obtuvo y el hilo ganador.

Puede haber empates y el programa lo toma como ganadores a ambos jugadores

Ejemplo de múltiples ganadores:

```
[1, 1, 3, 3]  
[1, 2, 2, 3]  
[1, 2, 2, 3]  
[1, 2, 0, 3]
```

3: Termino sus movimientos

El ganador es Hilo: 1 con 5 puntos !  
El ganador es Hilo: 2 con 5 puntos !  
El ganador es Hilo: 3 con 5 puntos !