**Reporte Sprint #3**

Implementen todas las características que permiten a un jugador humano jugar un juego SOS simple o general contra un oponente humano y **refactoricen su código existente si es necesario**. Las características mínimas incluyen elegir el modo de juego (simple o general), elegir el tamaño del tablero, configurar un nuevo juego, hacer un movimiento (en un juego simple o general) y determinar si un juego simple o general ha terminado. El siguiente es un diseño de GUI de muestra.

Se requiere el uso de una jerarquía de clases para hacer frente a los requisitos comunes del juego simple y general. Si tu código para Sprint 2 no ha considerado la jerarquía de clases, es hora de refactorizar su código.

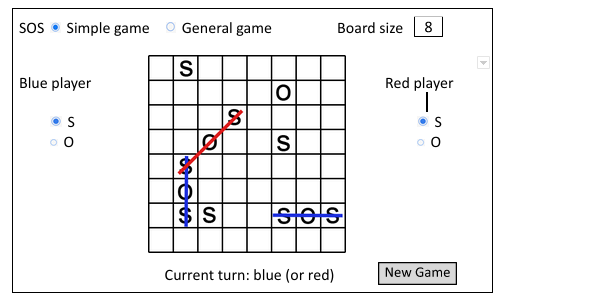


Figura 1. Diseño de GUI de muestra del programa en el Sprint 3

**Entregables: expande y mejora tu entrega para el sprint 2.**

1. **Demostración (6 puntos)**

Envíen un video de no más de cinco minutos, que demuestre claramente las siguientes características.

1. Un juego simple en el que el jugador azul es el ganador.
2. Un juego simple empatado con el mismo tamaño de tablero que es 1
3. Un juego general en el que el jugador rojo es el ganador y el tamaño del tablero es diferente de 1
4. Un juego general empatado con el mismo tamaño de tablero que es 1
5. Algunas pruebas unitarias automatizadas para el modo de juego simple
6. Algunas pruebas unitarias automatizadas para el modo de juego general

En el video, debes explicar lo que se está demostrando.

1. **Resumen del código fuente (2 puntos)**

|  |  |  |
| --- | --- | --- |
| Nombre del archivo de código fuente | ¿Código de producción o de prueba? | # lineas de código |
| Test\_board.py | prueba | 161 |
| Main.py | Produccion | 5 |
| Manager.py | Produccioon | 33 |
| Screens.py | Produccion | 281 |
| Board.py | produccion | 128 |
| Total | | 608 |

**Deben enviar todo el código fuente para obtener más puntos por esta tarea.**

1. **Código de producción vs Historias de usuario/Criterio de aceptación (4 puntos)**

Resuman cómo se implementa cada uno de los siguientes criterios de aceptación/historia de usuario en tu código de producción (nombre de clase y nombre de método, etc.)

|  |  |
| --- | --- |
| **ID de historia de usuario** | **Nombre de historia de usuario** |
| 1 | Crea un tablero vacio con un tamaño valido |
| 2 | Hace un movimiento simple |
| 3 | Completa un sos |
| 4 | Termina el juego en modo simple |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre y ID de la historia usuario** | **AC ID** | **Nombre clase(s)** | **Nombre Método(s)** | **Estatus (completo o no)** | **Notas (opcional)** |
| 1ccreacion de tablero simple |  | Board | create\_board | Complete |  |
| 2 inseratr ficha en modo simple |  | Board | insert\_board | Complete |  |
| 3completar sos en modo simple | … | Board | complete\_SOS | Complete |  |
| 4termina el juego en modo simple |  | Board | win\_or\_tie | completo |  |

1. **Pruebas vs Historias de usuario/Criterio de aceptación (4 puntos)**

Resuman cómo cada uno de los criterios de aceptación/historia de usuario es probado por su código de prueba (nombre de clase y nombre de método) o pruebas realizadas manualmente.

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Crea un tablero vacio con un tamaño valido |
| 2 | Hace un movimiento simple |
| 3 | Completa un sos |
| 4 | Termina el juego en modo simple |

4.1 Pruebas automatizadas que corresponden directamente a los criterios de aceptación de las historias de usuario anteriores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre y ID de la historia usuario** | **AC**  **ID** | **Nombre Clase (s) del código de prueba** | **Nombre método(s) del código Prueba** | **Descripción de los casos de prueba (entrada & salida esperada)** |
| 1creacion de tablero simple | 1.1 | TestBoard | Test\_create\_board | Se ingresa un tamaño valido del tablero y se cra un tablero vacio  Entrada=board(board\_size)  Salida=None |
| Inserta ficha modo simple | <iinserta pieza valida en una posicion valida> | TestBoard | Test\_insert\_piece\_valid() | Se inserta una pieza valida en una casilla valida  Entrada=insert.piece(0.0,”O”)  Salida=”O” |
|  | …<inserta pieza valida en una posicion invalida> | TestBoard | Test\_insert\_piece\_in\_vaid\_coordinales | se prueba si se inserta una pieza valida en una posicion invalida entrada=insert\_piece(n +1,n+1‘S’ esperada=None |
|  | <inserta pieza invalida en una `posicion calida> | TestBoard | teste\_insert\_piece\_i nvalid\_piece\_type test\_insert\_piece\_in valid\_piece\_value | Se prueba si se inserta una pieza invalida en una posicion valida entrada=board.insert\_pi ece(4) esperada=’La pieza debe ser de tipo strign entrada=board.insert\_pi ece(‘W’) esperada=’Pieza no valida’ |
|  | <inserta piez valido en una posicion ocupada> | TestBoard | teste\_insert\_piece\_i valid\_position\_busy | Se prueba si se inserta una pieza valida en una posicion ocuapada insert\_piece(0,0,’S’) entrada=insert\_piece(0, 0,’S) esperada=’Casilla Ocupada’ |
| Complete un sos | <se detecta una SOS en una fila> | TestBoard | Test\_complete\_row\_SOS | Se prueba si detecta que hay un SOS en una fila  Entrada=fila con SOS  Salida=true |
|  | <se detecta una SOS en una columna> | TestBoard | Test\_complete\_colum\_SOS | Se prueba si detecta que hay un SOS en una columna  Entrada=columnacon SOS  Salida =true |
|  | <se detecta una SOS en una diagonal de derecha a izquierda> | TestBoard | Test\_complete\_diagonal\_rigth\_to\_left | Se prueba si detecta que hay un SOS en una diagonal de derecha a izquierda  Entrada=SOS en una diagonal  Salida=true |
|  | <se detecta un SOS en una diagonal de izquierda a derecha> | TestBoard | Test\_complete\_diagonal\_left\_to\_rigth | Se prueba si detecta que hay un SOS en una diagonal de izquierda a derecha  Entrada=SOS en una diagonal  Salida=true |
| Termmina un juego simple | <se ve si el tablero aun esta vacio> | TestBoard | Test\_empty\_board | Se prueba enel juego si detecta un tablero vacio osea sin jugada alguna  Entrada= tablero vacio  Salida=empty voard |
|  | <el tablero esta lleno y no tiene un SOS> | TestBoard | Test\_board\_complete\_SOS\_incomplete | Se prueba cuando un tablero esta lleno y no hay un SOS  Entrada=Tablero lleno de s  Salida=Tie |
|  | <el tablero esta lleno y tiene un SOS> | TestBoard | Test\_board\_complete\_SOS\_complete | Se prueba cuando un tablero esta lleno y hay un SOS  Entrada=Tablero con un SOS  Salida=win |
|  | <el tablero esta incomplete y no tiene un SOS> | TestBoard | Test\_board\_incomplrete\_SOS\_incomplete | Se prueba cuando un tablero esta incompleto y no hay un SOS  Entrada=Tablero con una ficha  Salida=continue |
|  | <e tablero esta incomplete y tiene un SOS> | TestBoard | Test\_board\_incomplete\_SOS\_complete | Se prueba cuando un tablero no esta incompleto y hay un SOS  Entrada=Tablero con 1 sos  Salida=win |

4.2 Pruebas manuales que corresponden directamente a los criterios de aceptación de las historias de usuario anteriores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre y ID de la historia usuario** | **AC**  **ID** | **Entrada de caso de prueba** | **Salida esperada** | **Notas** |
| 1creacion de tablero simple | 1.1 | Entrada=board(board\_size) | esperada=None |  |
| Inserta ficha modo simple | <iinserta pieza valida en una posicion valida> | entrada=insert.piec e(0,0,,’O’) | esperada=’O’ |  |
|  | …<inserta pieza valida en una posicion invalida> | entrada=insert\_piec e(n+1,n+1‘S’ | esperada=None |  |
|  | <inserta pieza invalida en una `posicion calida> | entrada=board.inse rt\_piece(‘W’) | esperada=’Pieza no valida’ |  |
|  | <inserta piez valido en una posicion ocupada> | entrada=insert\_piec e(0,0,’S) | esperada=’Casilla Ocupada’ |  |
| Complete un sos | <se detecta una SOS en una fila> | Entrada =tablero con un SOS en una fila | Esperada=true |  |
|  | <se detecta una SOS en una columna> | Entrada =tablero con un SOS en una columna | Esperada=true |  |
|  | <se detecta una SOS en una diagonal de derecha a izquierda> | Entrada =tablero con un SOS en una diagonal de dereccha a izquierda | Esperada=true |  |
|  | <se detecta un SOS en una diagonal de izquierda a derecha> | Entrada =tablero con un SOS en una diagonal de izquierda a derecha | Esperada=true |  |
| Termmina un juego simple | <se ve si el tablero aun esta vacio> | Entrada=tablero vacio | Esperada=’empty board’ |  |
|  | <el tablero esta lleno y no tiene un SOS> | Entrada=tablero lleno de S | Esperada=tie |  |
|  | <el tablero esta lleno y tiene un SOS> | Entrada=tablero  Con una fila que tiene SOS y lo demas es s | Esperada=win |  |
|  | <el tablero esta incomplete y no tiene un SOS> | Entrada=tablero con solo una S y lo demas vacio | Esperada=continue |  |
|  | <e tablero esta incomplete y tiene un SOS> | Entrada=tablero con un SOS y lo demas vacio | Esperada=win |  |

4.3 Otras pruebas automatizadas o manuales que no corresponden a los criterios de aceptación de las historias de usuario anteriores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Entrada prueba** | **Resultado esperado** | **Nombre de clase del código de prueba** | **Nombre del método del código de prueba** |
|  |  |  |  |  |
|  |  |  |  |  |

5. Describe cómo la jerarquía de clases en tu diseño trata con los requisitos comunes y diferentes del juego simple y el juego general. (4 puntos)