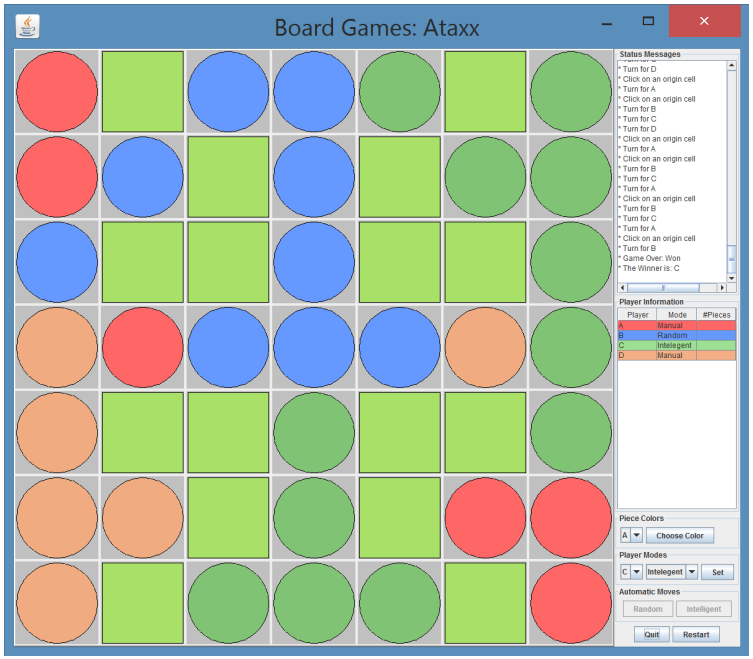
Tecnología de la Programación

Curso 2015-2016

Práctica 5

Interfaz gráfico de usuario para los juegos de tablero



Antes de comenzar...

¡Importante!

- En el campus virtual está disponible una nueva versión de `basecode`.
- Antes de empezar la práctica 5, descarga la nueva versión y copia tus fuentes del paquete `practica4` en ella.

- Debes adaptar tu código de la siguiente forma:
  - La signatura del método `GameRules.evaluate` ha cambiado. Añade el argumento que falta, aunque no lo utilices en tu código.
  - El interfaz `GameFactory` ahora implementa `java.io.Serializable`. Te aparecerán algunos *warnings* (los puedes ignorar por ahora).
  - Se ha modificado el nombre del paquete `connectn` para que esté en minúsculas para seguir –en lo posible– unas **normas de codificación**. Por ejemplo, puedes consultar las de Google en:
    `https://google.github.io/styleguide/javaguide.html`
    Debes adaptar tu código a este cambio.
  - Otros cambios útiles que no afectan a tu código. Puedes verlos en:
    `basecode/src/es/ucm/fdi/tp/basecode/CHANGES.txt`

Antes de comenzar...

- Crea un paquete `practica5` y copia en él el fichero `Main.java` del paquete `practica4`.
- Todo el código de esta práctica debe estar definido dentro de `practica5`.
- Recuerda que no está permitido modificar ningún fichero del paquete `basecode`.
- Además, te recomendamos que no modifiques nada de `practica4`, excepto si vas a corregir algún error de programación.
- No está permitido utilizar herramientas de generación automática de interfaces gráficos de usuario (como *NetBeans*).

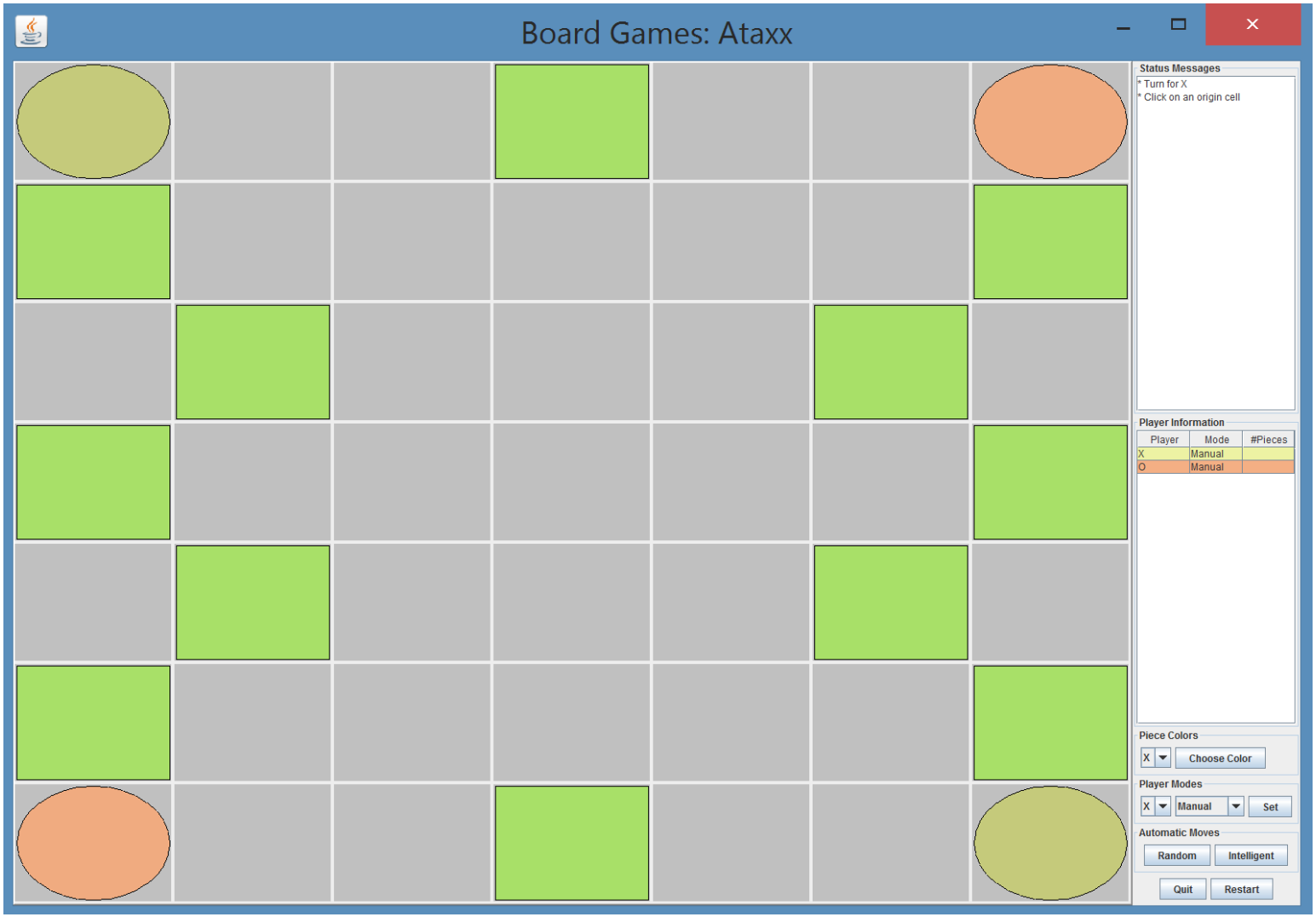
Descripción general de la práctica

- **Fecha de entrega: 22 de abril de 2016 a las 16:00h.**
- **Objetivos:**
  - **GUI en Java utilizando Swing.**
  - **Patrón de diseño Modelo-Vista-Controlador.**
  - **Diseño Orientado a Objetos.**
- En esta práctica debes desarrollar e integrar varias **vistas gráficas** para los juegos *Connect-N*, *Tic-Tac-Toe*, *Advanced Tic-Tac-Toe* y *Ataxx* (nos referiremos a ellas como vistas de `ventana`).
- **Debes utilizar el patrón de diseño MVC**. Además, debes utilizar un diseño que permita desarrollar **una sola vista que después puedas adaptar a cada juego concreto** utilizando **herencia** o **composición**.
- En las clases de la asignatura veremos más reglas generales de diseño que puedes utilizar.

Descripción general de la práctica

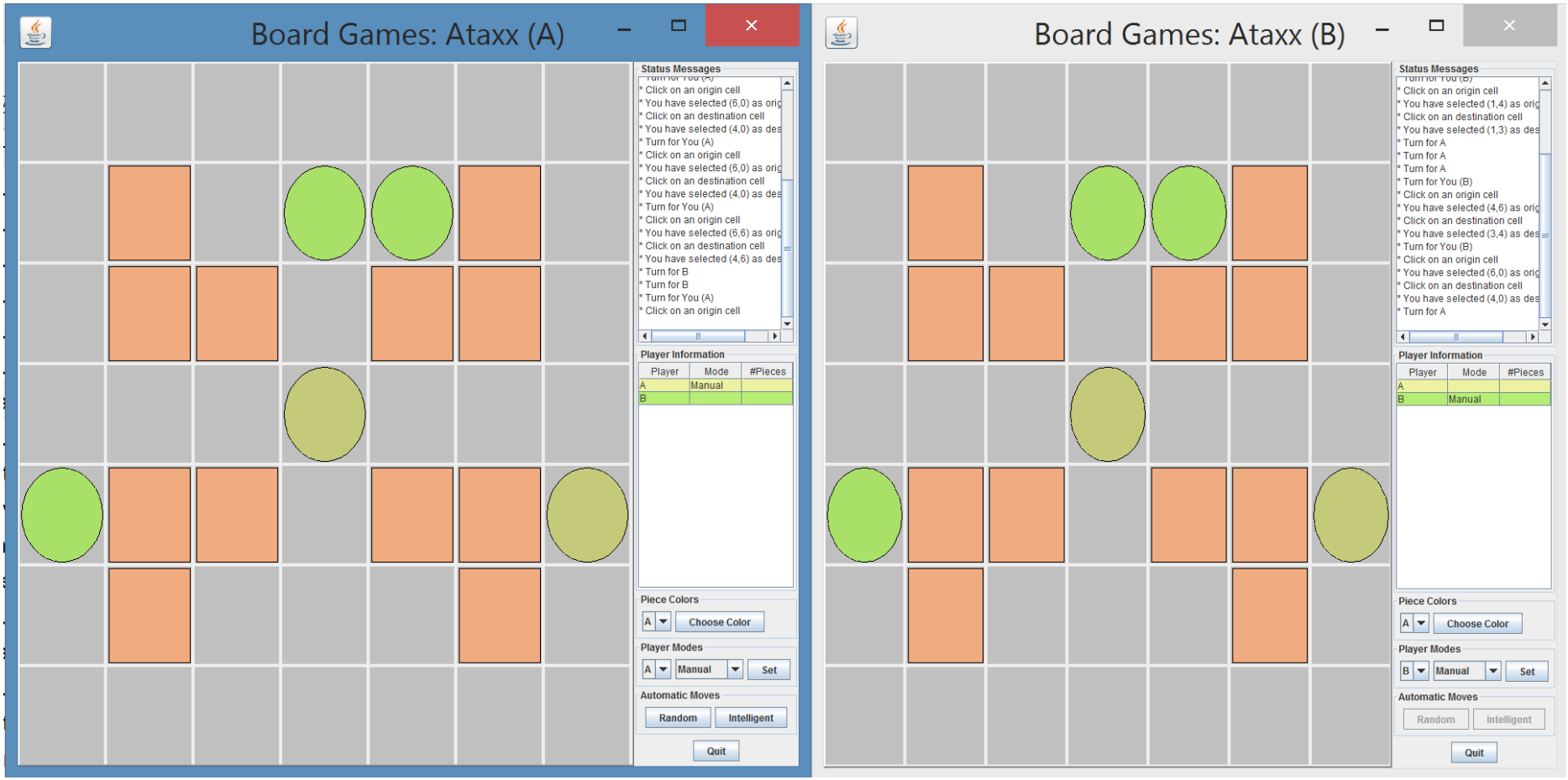
- La vista `ventana` debería permitir **dos modos de uso**:
  1. `ventana única`, en la que todos los jugadores utilizan **la misma ventana**.
  2. `múltiples ventanas`, en la que **cada jugador tiene su propia ventana** y cada ventana solo la puede utilizar un jugador.
- Por defecto, se debe utilizar el modo `ventana única`.
- El modo `múltiples ventanas` debe utilizarse cuando el usuario proporciona la opción `-m` (o `--multiviews`) en la línea de órdenes.
- El interfaz gráfico de la vista `ventana` debe tener la misma información que el que se muestra en las siguientes imágenes y debe proporcionar la funcionalidad que se detalla a continuación.

Descripción general de la práctica



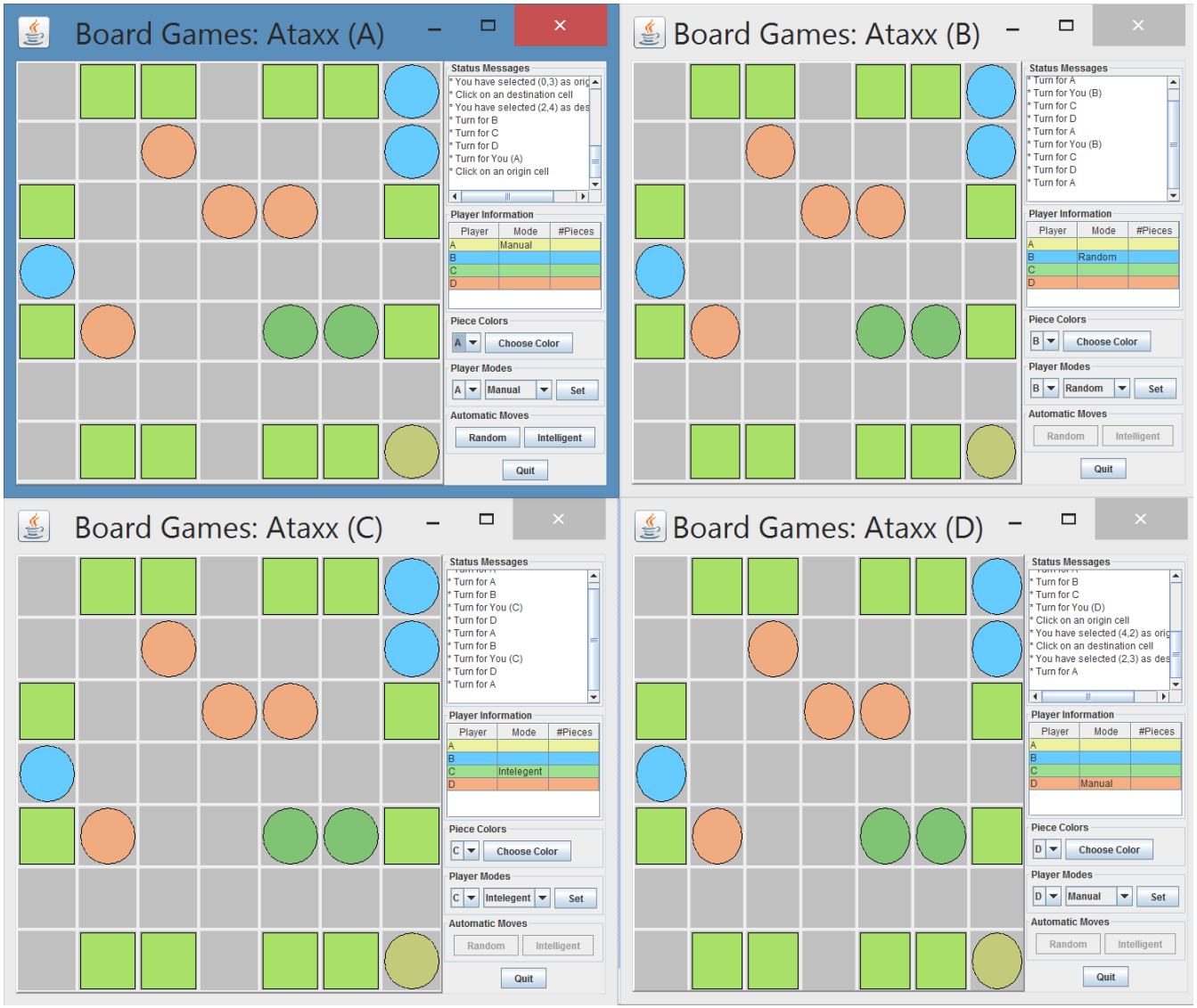

```
-g ataxx -d 7x7 -o 4
```

Descripción general de la práctica




```
-g ataxx -d 7x7 -o 4 -p A:a,B:r -m
```

Descripción general de la práctica

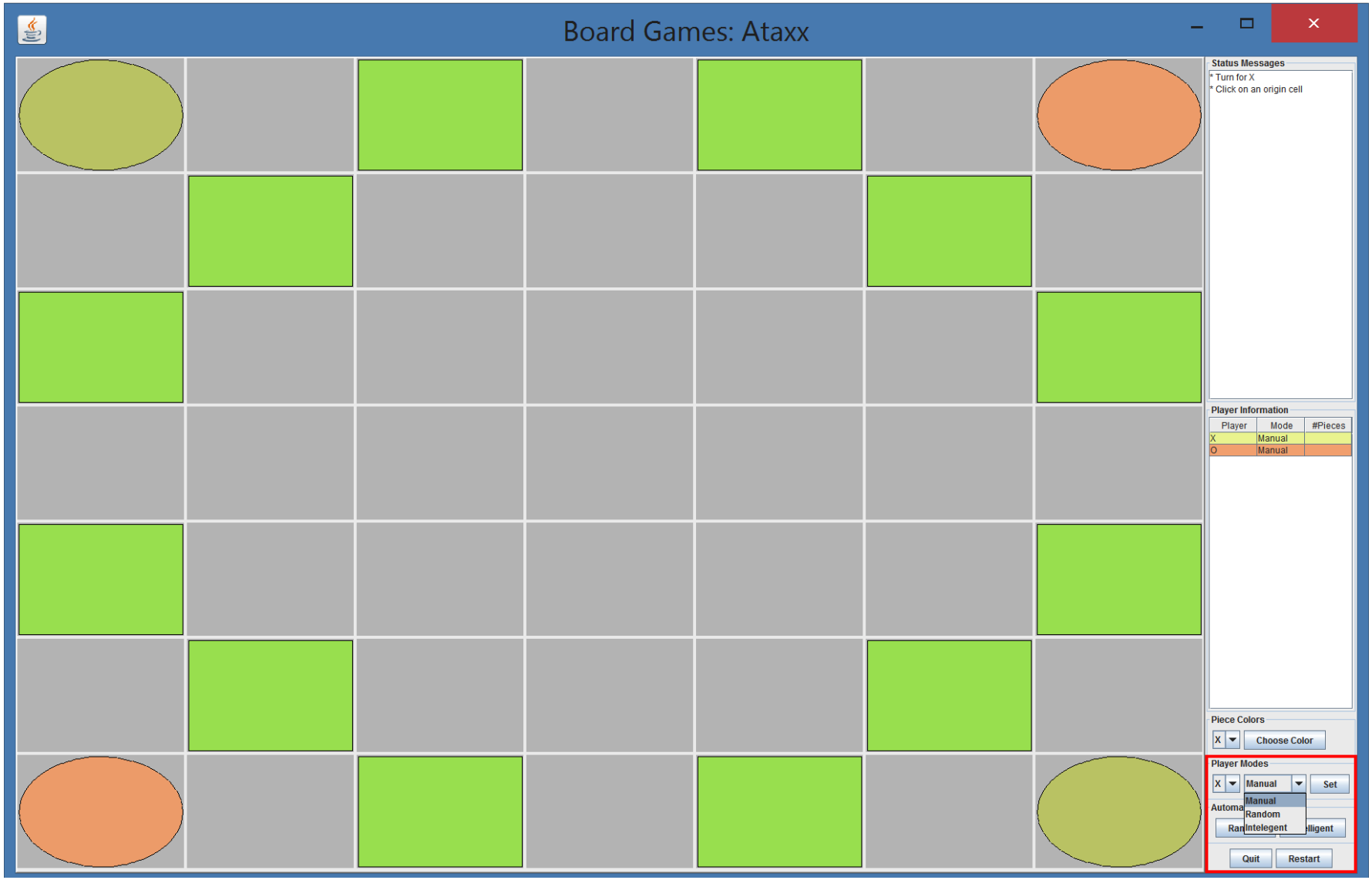



```
-g ataxx -d 7x7 -o 4 -p A:m,B:r,C:a,D:m -m
```

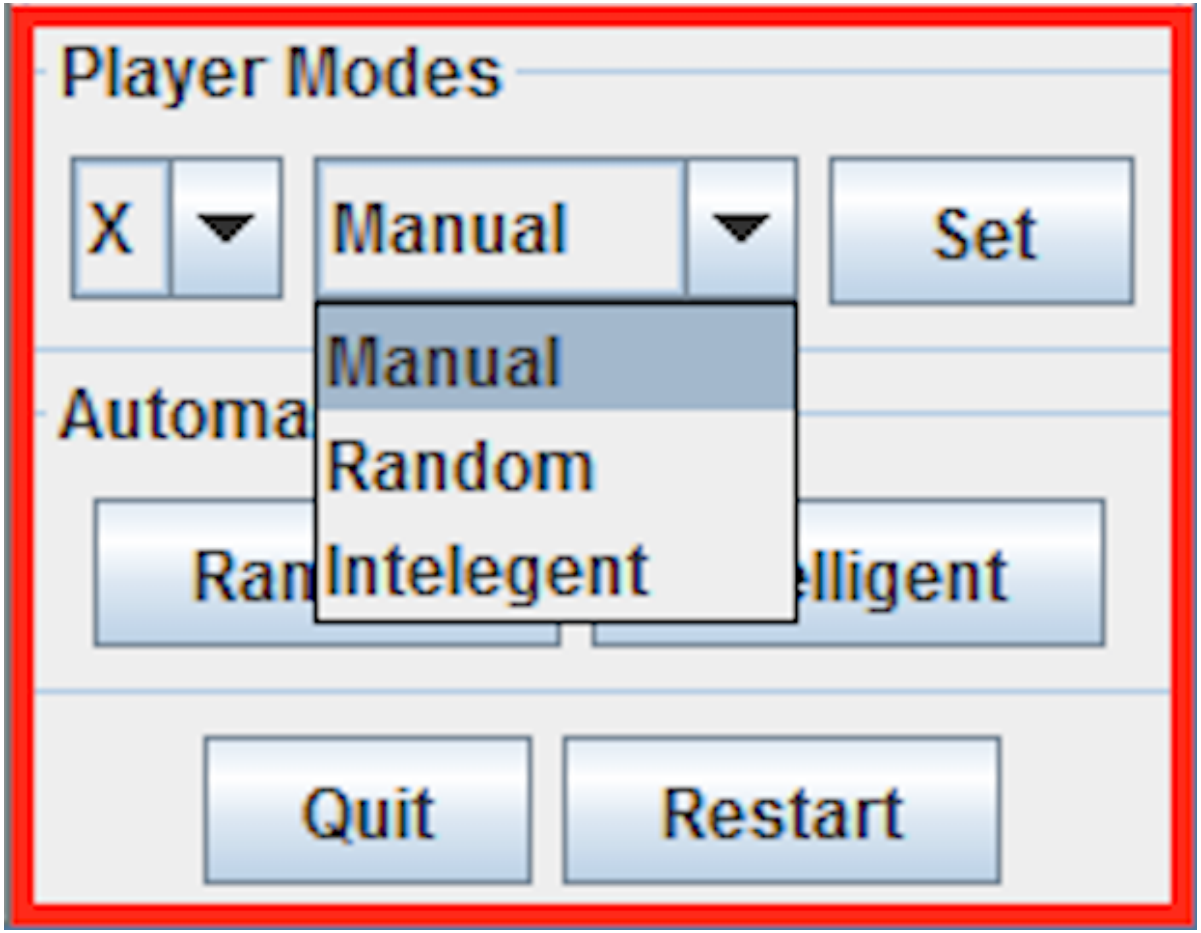
Descripción general de la práctica

- En esta vista, un jugador puede estar en cualquiera de los siguientes **modos de juego**: `MANUAL`, `RANDOM` o `INTELLIGENT`.
- Todos los jugadores comienzan en modo `MANUAL`, sea cual sea el modo que se haya especificado en la línea de órdenes.
- Los jugadores pueden cambiar su modo de juego durante la partida utilizando los *combo-boxes* del área "*Player Modes*" de la ventana.
- Los jugadores en modo `MANUAL` pueden hacer movimientos, cuando están en su turno, utilizando uno de los siguientes métodos:
  1. Pulsando con el ratón sobre el tablero, de una forma que dependerá del juego al que se esté jugando.
  2. Utilizando el botón *Random* (en el área "*Automatic Moves*") para hacer un movimiento aleatorio (mediante un jugador aleatorio).
  3. Utilizando el botón *Intelligent* (en el área "*Automatic Moves*") para realizar un movimiento inteligente (mediante un jugador inteligente).

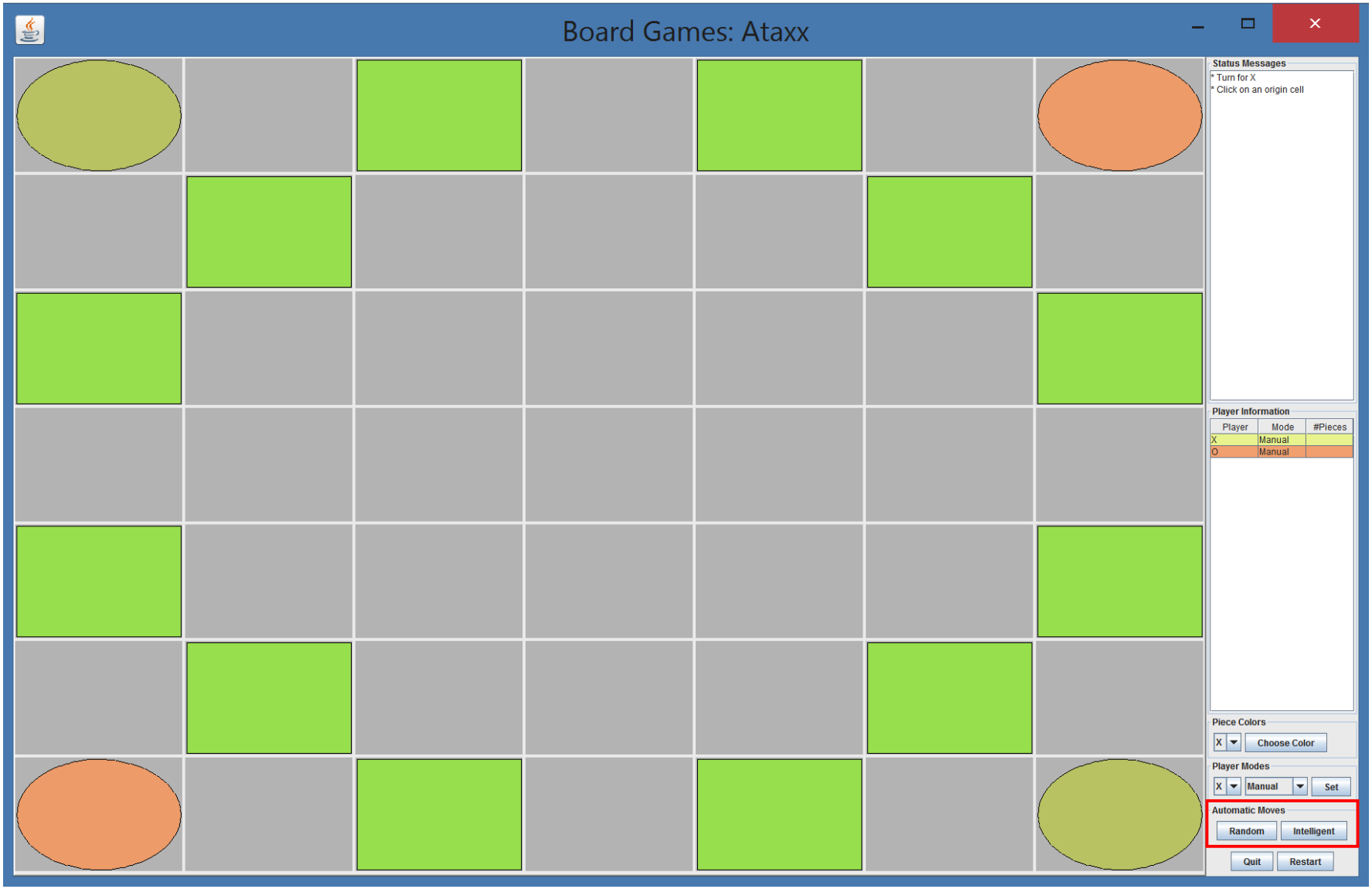
Descripción general de la práctica



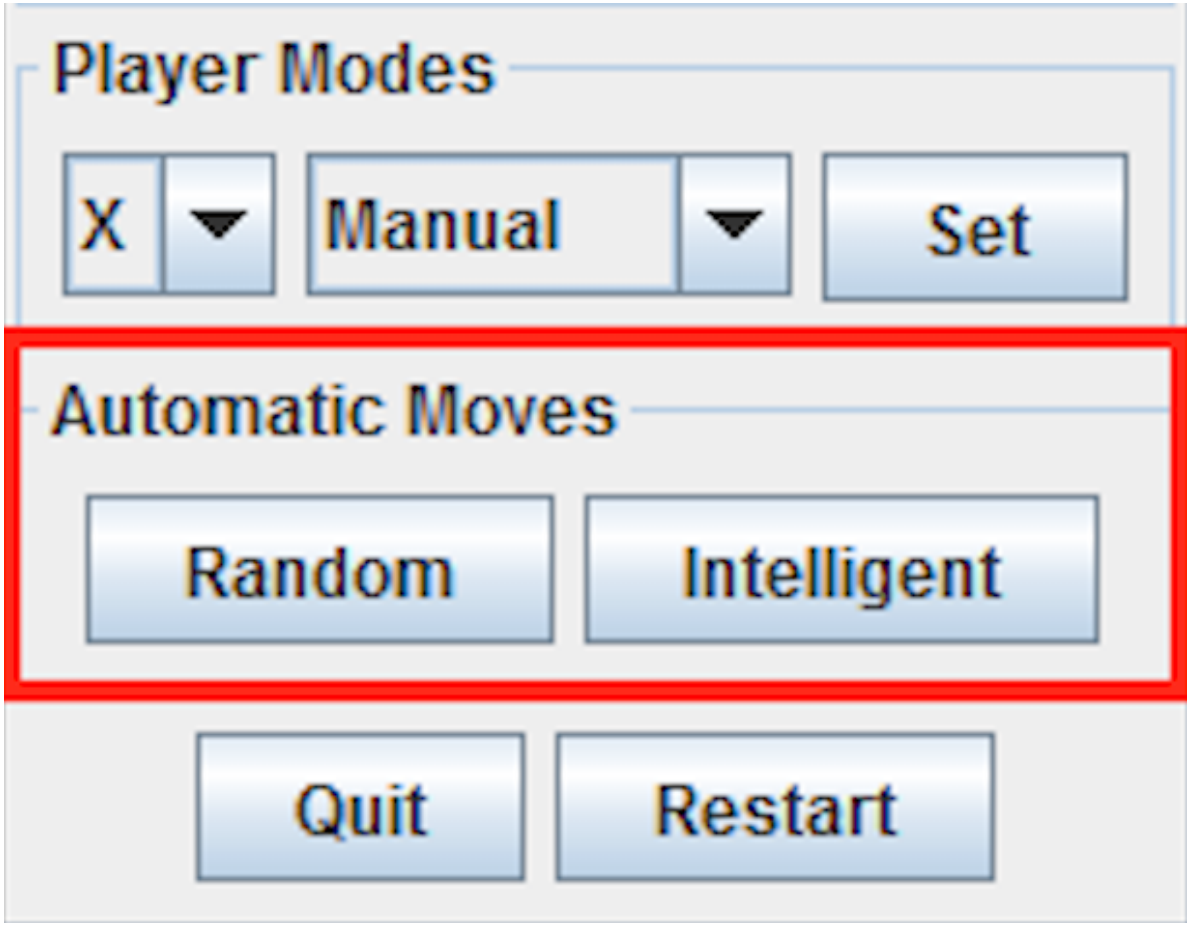
Descripción general de la práctica



Descripción general de la práctica



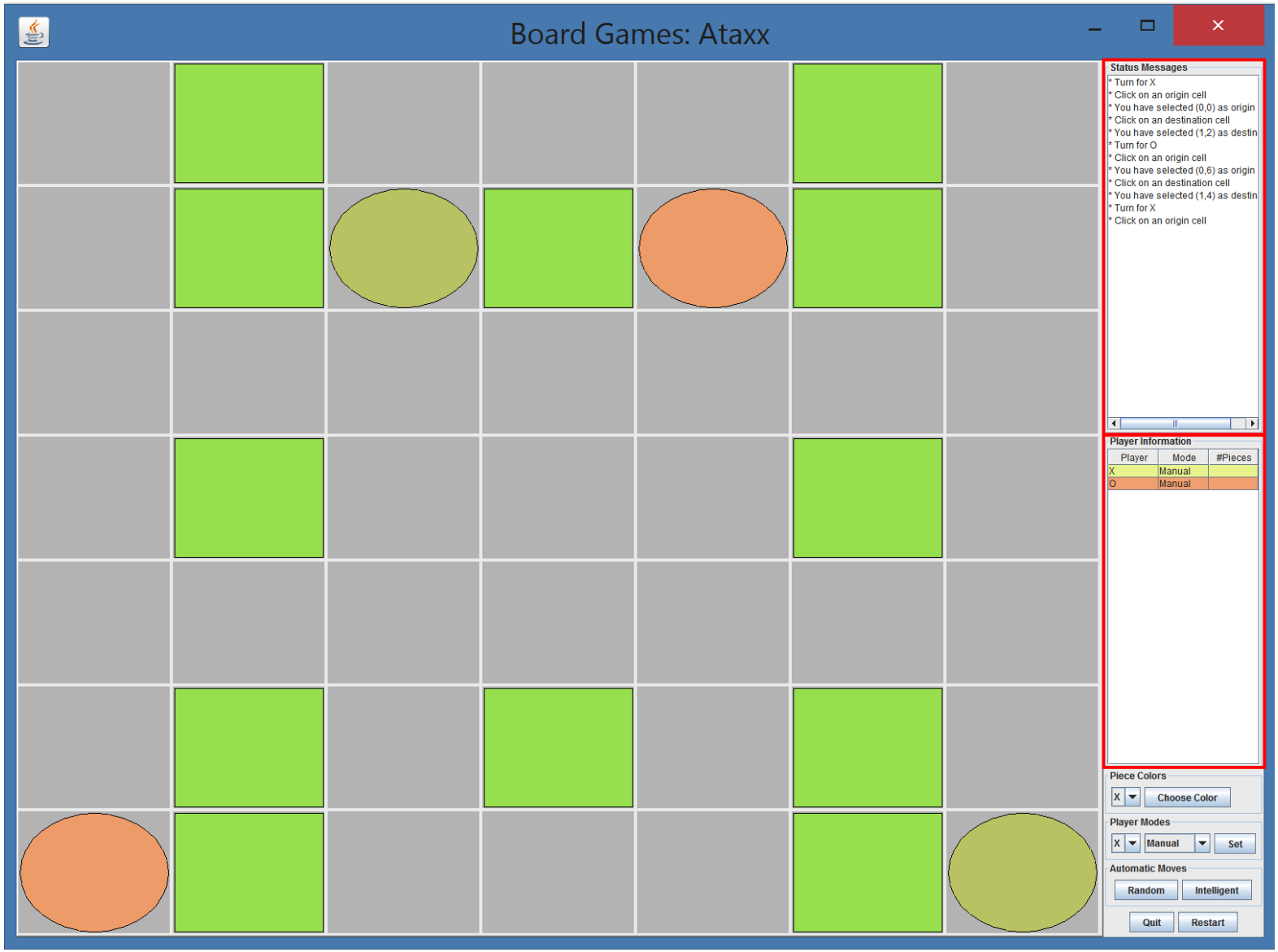
Descripción general de la práctica



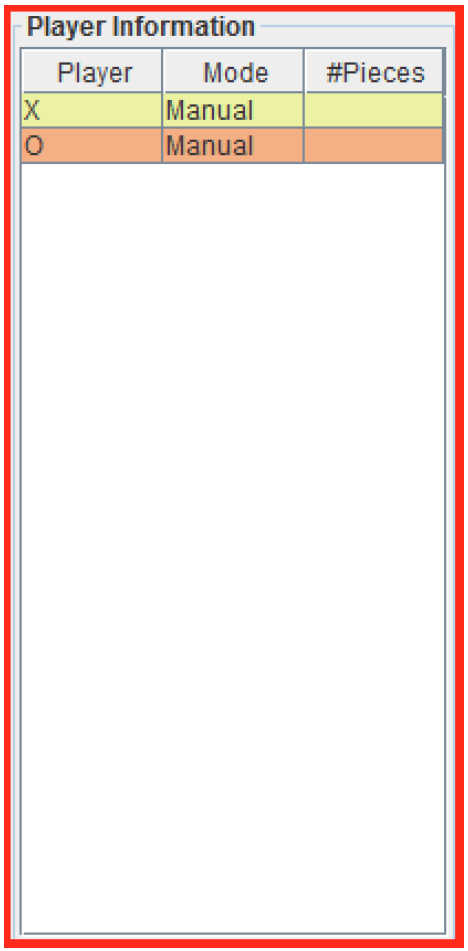
Descripción general de la práctica

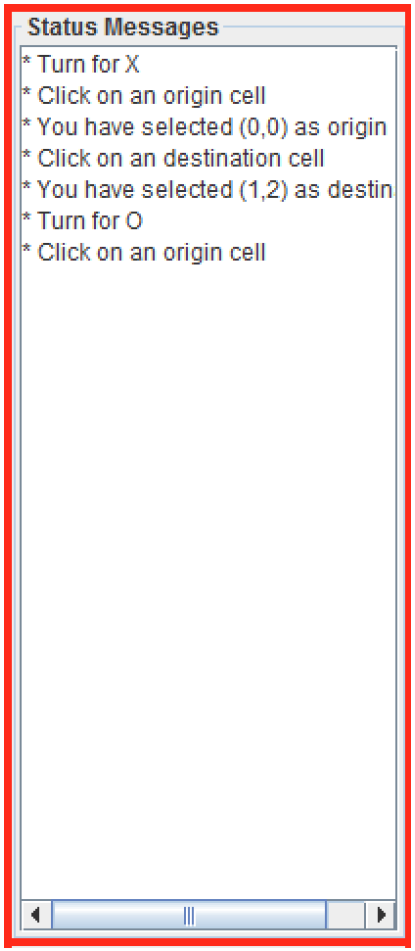
- Los jugadores en modo `MANUAL` no deben poder introducir un movimiento cuando no sea su turno.
- Los jugadores en modo `RANDOM` o `INTELLIGENT` deben jugar *automáticamente* cuando sea su turno y no deben poder introducir un movimiento manual si están en un modo no manual.
- Si todos los jugadores están en modo no manual, podemos sentarnos a ver cómo juegan automáticamente hasta que termina la partida (o hasta que modificamos el modo de juego de uno de los jugadores y llega su turno).
- Todos los movimientos deben realizarse en el modelo a través del controlador, mediante la correspondiente llamada a `makeMove(...)`.

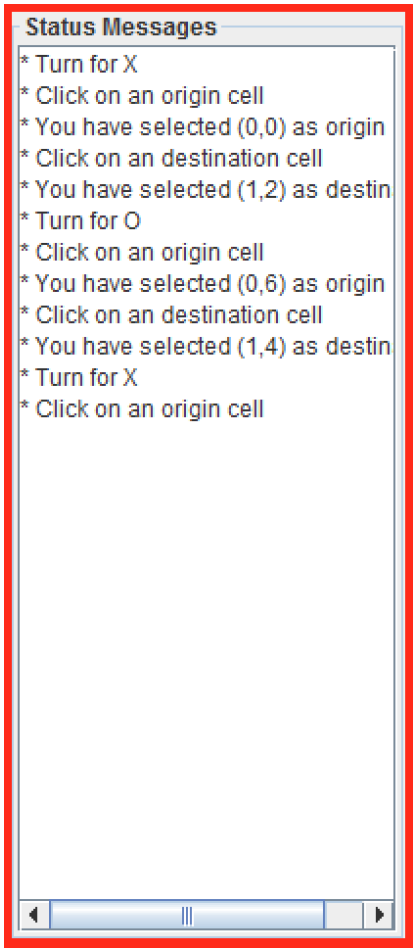
Descripción general de la práctica



Descripción general de la práctica







Descripción general de la práctica

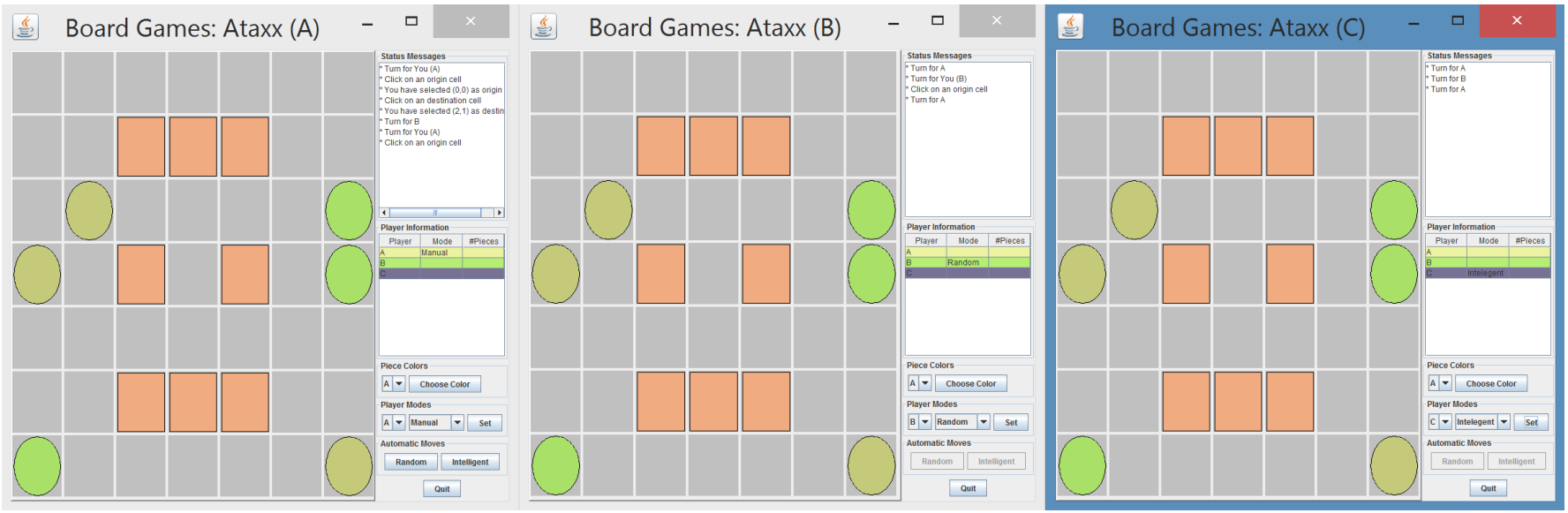
- Los objetos `Player` que se deben utilizar para hacer movimientos `RANDOM` o `INTELLIGENT` deben pasarse a la clase de la vista mediante el método `GameFactory.createSwingView(...)`, que a su vez los recibe de `Main.java`.

- Cuando uno de los jugadores es `null` (por ejemplo, cuando estos jugadores no están permitidos por el juego al que se vaya a jugar), la vista se debe adaptar a esta situación y no debería permitir realizar movimientos con ese modo de juego.

- En las siguientes transparencias se explican los componentes de la vista y se indican los requisitos que la implementación **debe satisfacer**.

Título de la ventana

- El **título de la ventana** debe mostrar la descripción del juego al que se está jugando, y si está activado el modo `múltiples ventanas`, debe incluir el nombre del jugador (el identificador de la ficha) a la que pertenece la ventana.

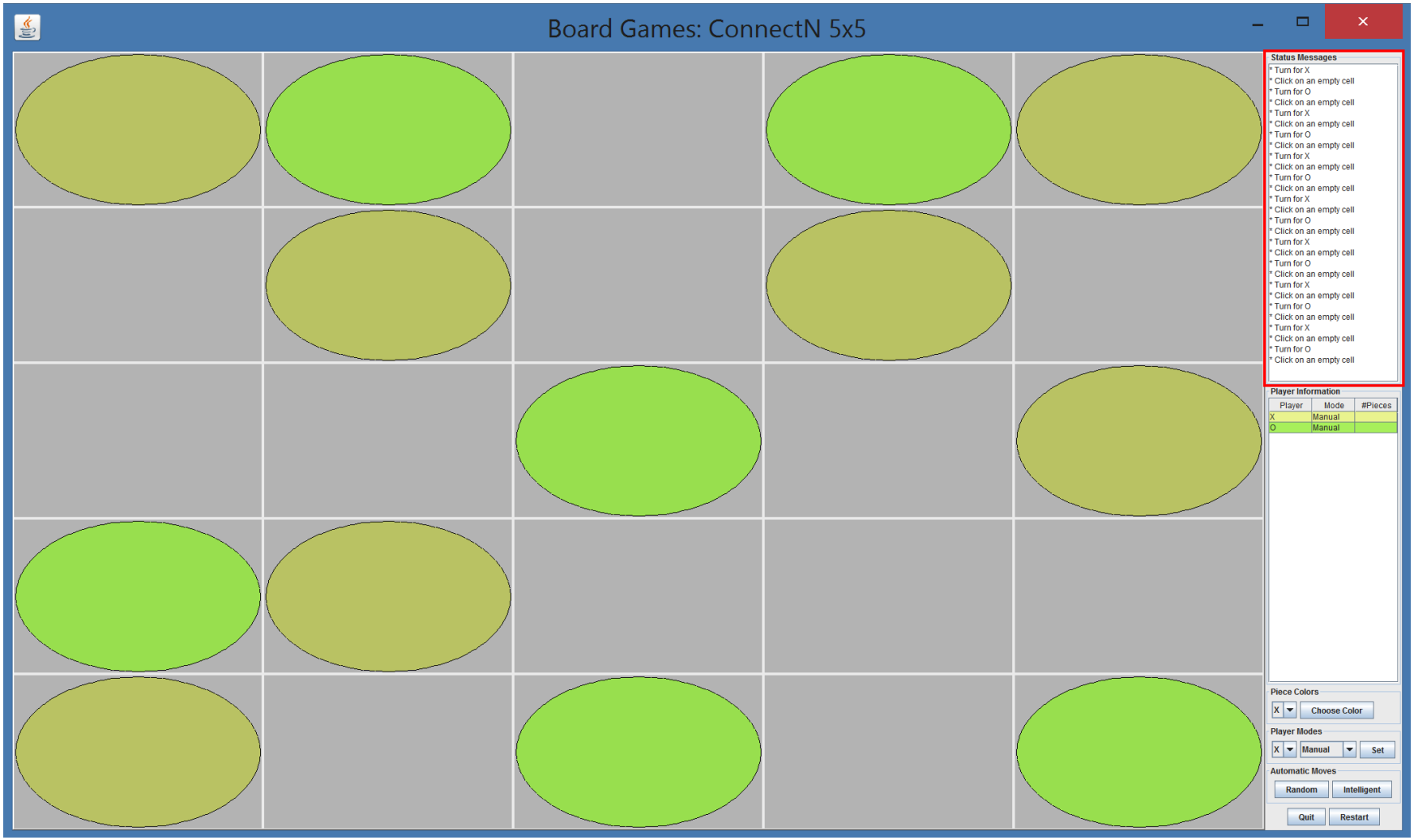



```
-g ataxx -d 7x7 -o 4 -m -p A:m,B:r,C:a
```

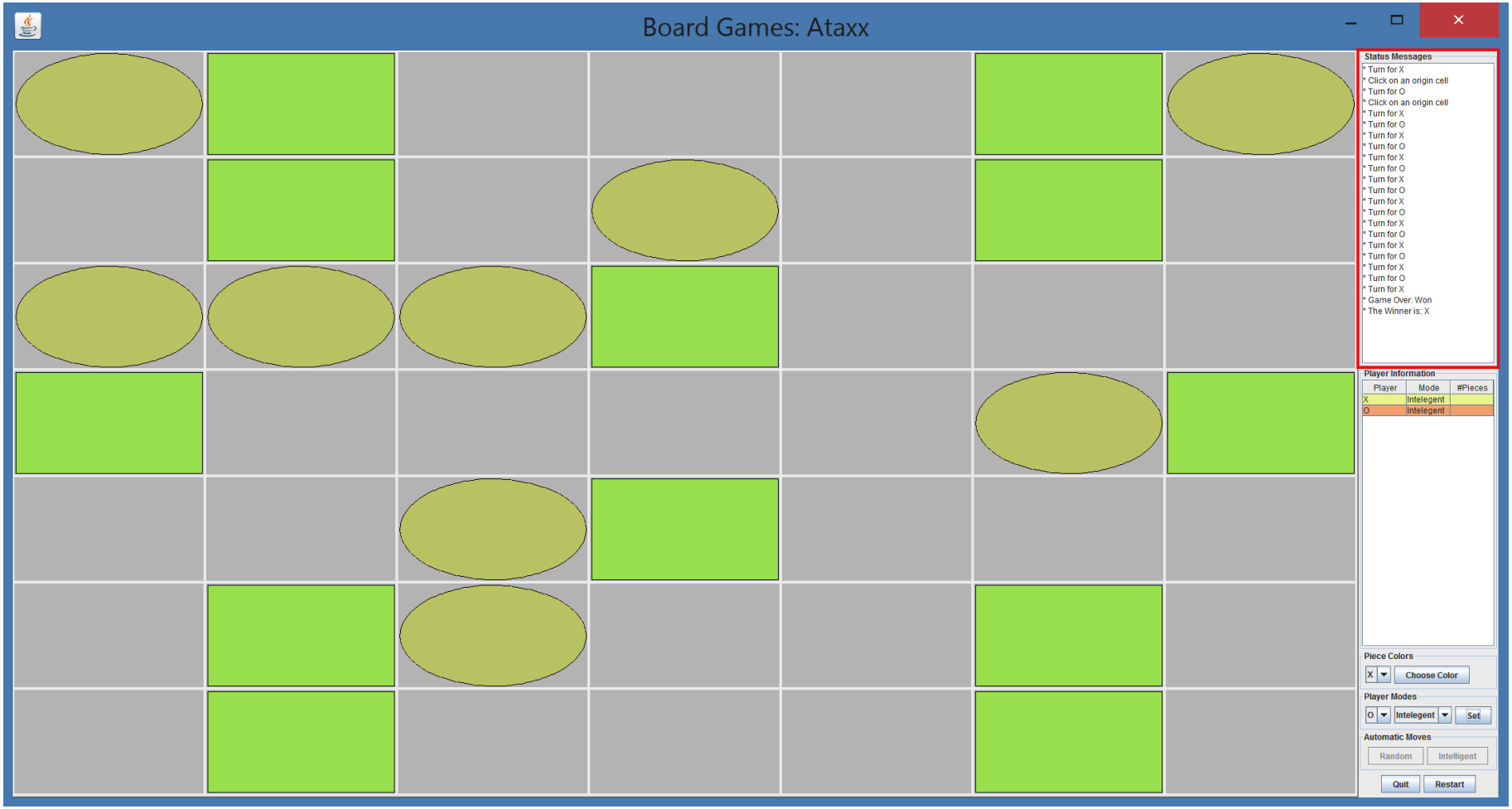
Status Messages

- Este área de la ventana debe mostrar la información del estado del juego, pedir a cada jugador que realice un movimiento, etc.
- Debe implementarse utilizando un componente `JTextArea` de solo lectura con un `JScrollPane`.
- Ejemplos de la información que **debe mostrarse:**
  - Cuando cambia el turno se debe mostrar el identificador de la ficha que debe jugar a continuación. Por ejemplo: "*Turn for X*".
  - Si se está en modo `múltiples ventanas` también debe incluir la palabra "*You*" en la ventana del jugador al que le corresponde el turno. Por ejemplo: "*Turn for X (You!)*" o "*Turn for You X!*".
  - Cuando sea el turno de jugadores en modo `MANUAL`, se debe mostrar información que les indique cómo realizar un movimiento. Por ejemplo:
    - ★ En el caso de *Connect-N* se puede mostrar un mensaje como el siguiente: "*Click on an empty cell*".
    - ★ En el caso de *Ataxx* se puede mostrar "*Click on an origin piece*" y cuando se ha seleccionado la ficha "*Click on the destination position*".

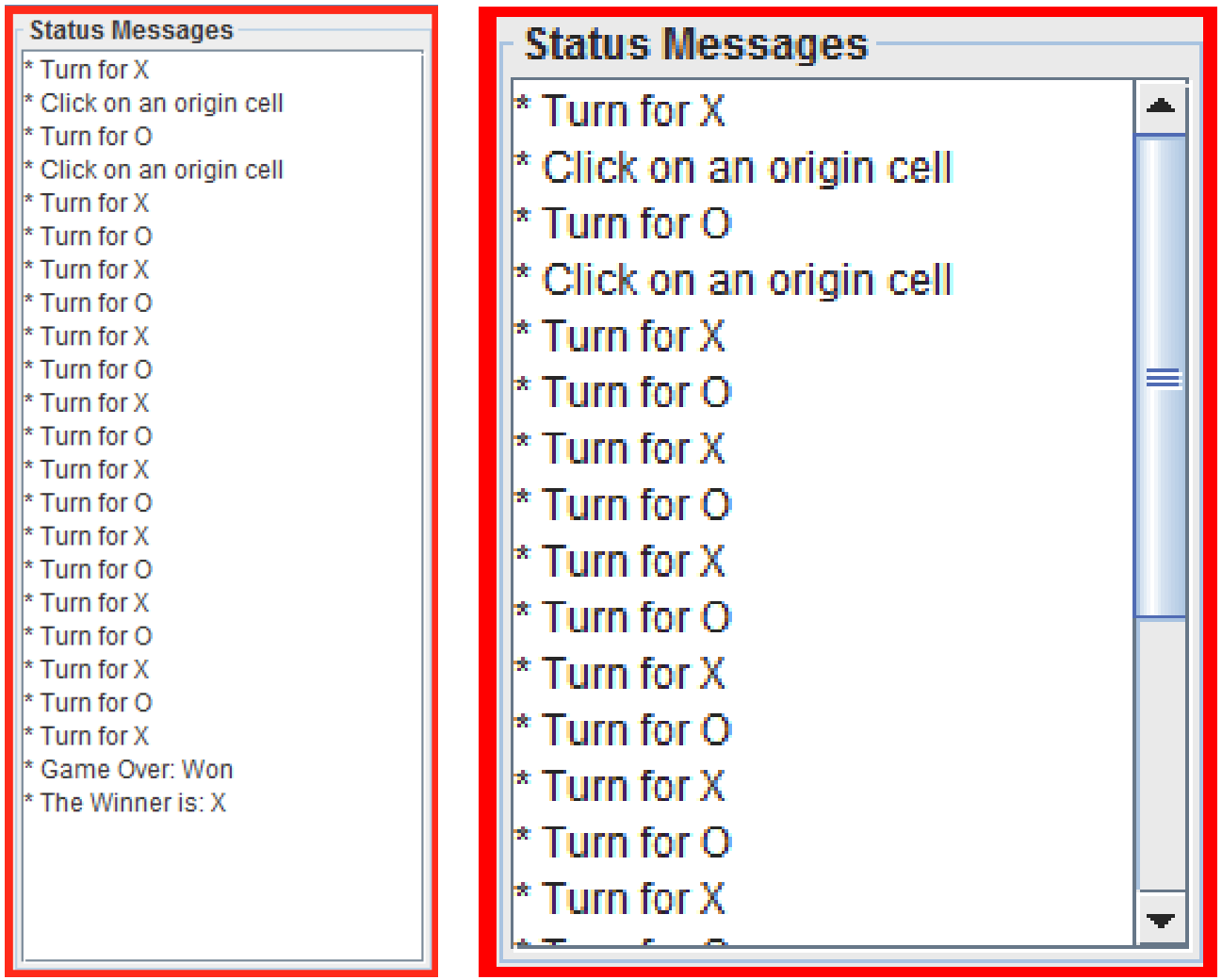
Status Messages



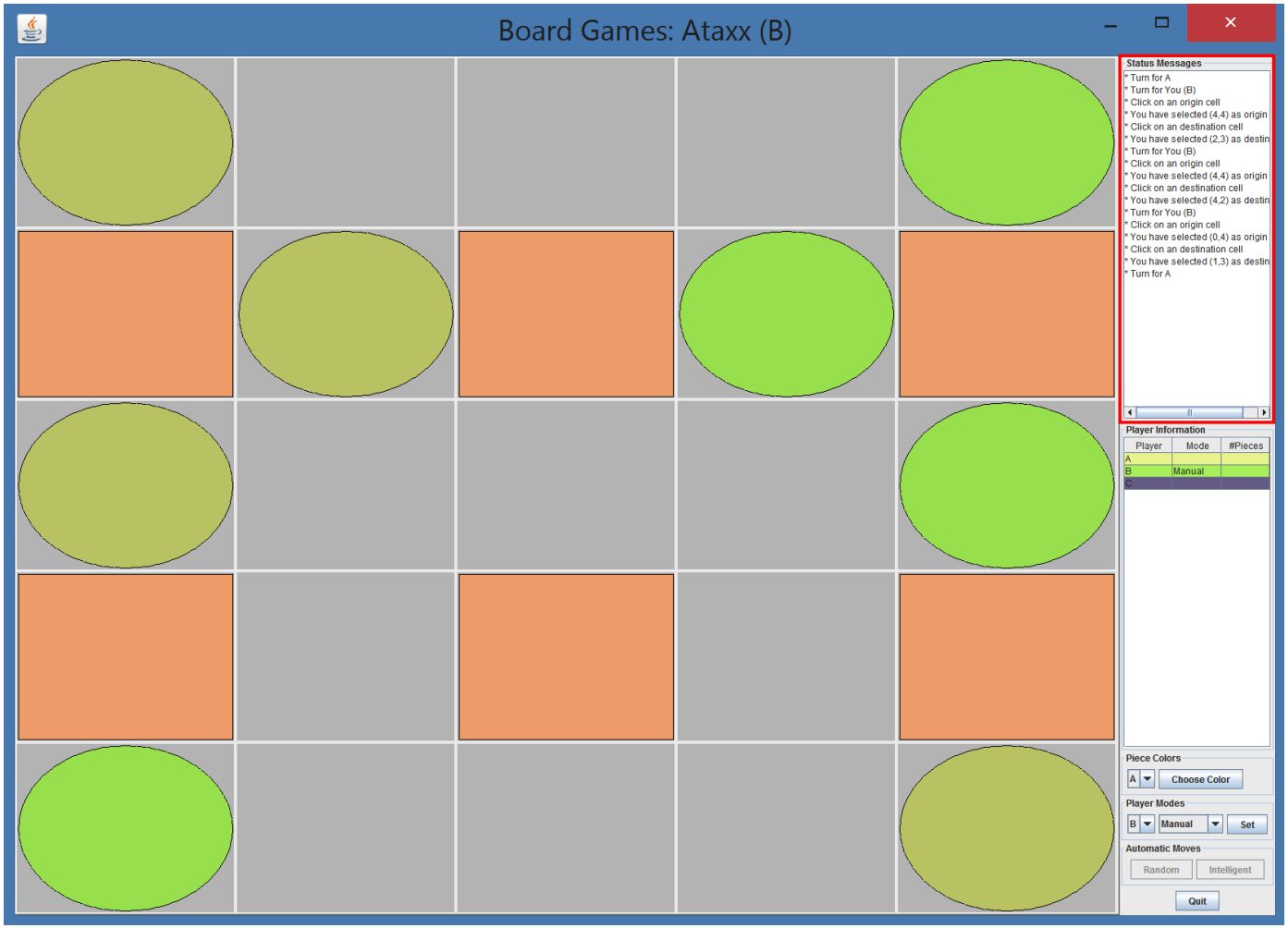
Status Messages



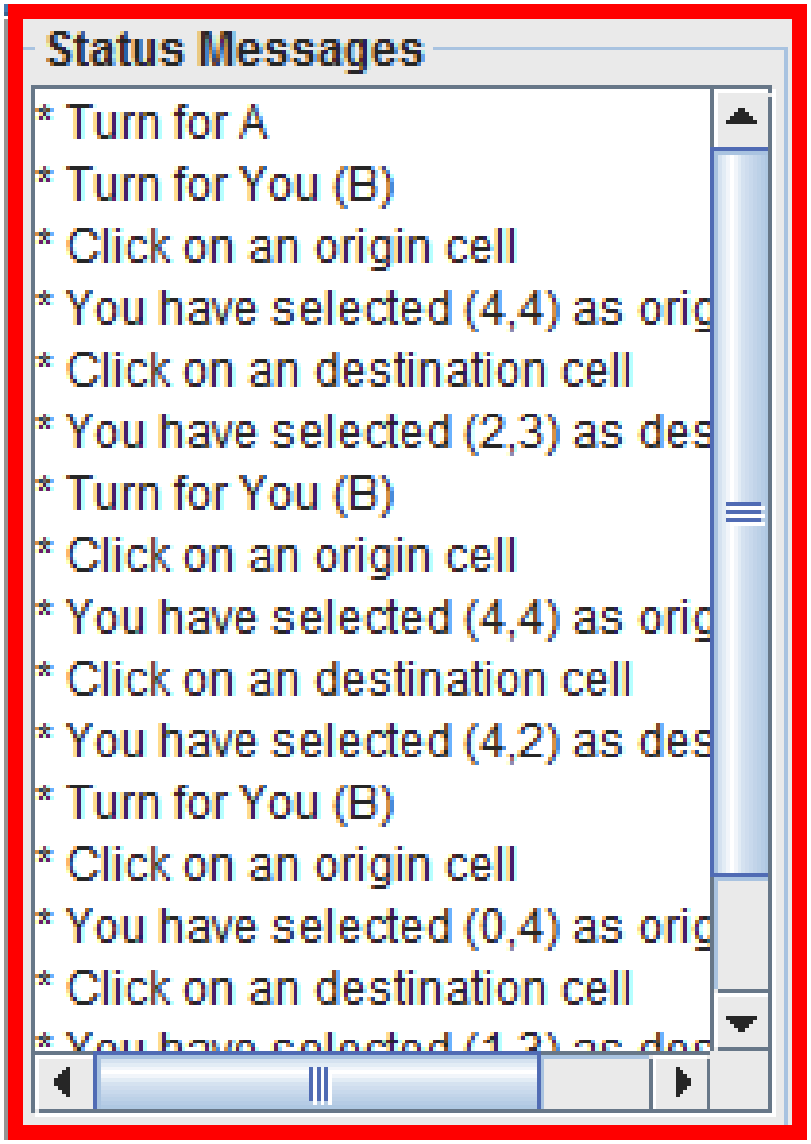
Status Messages



Status Messages

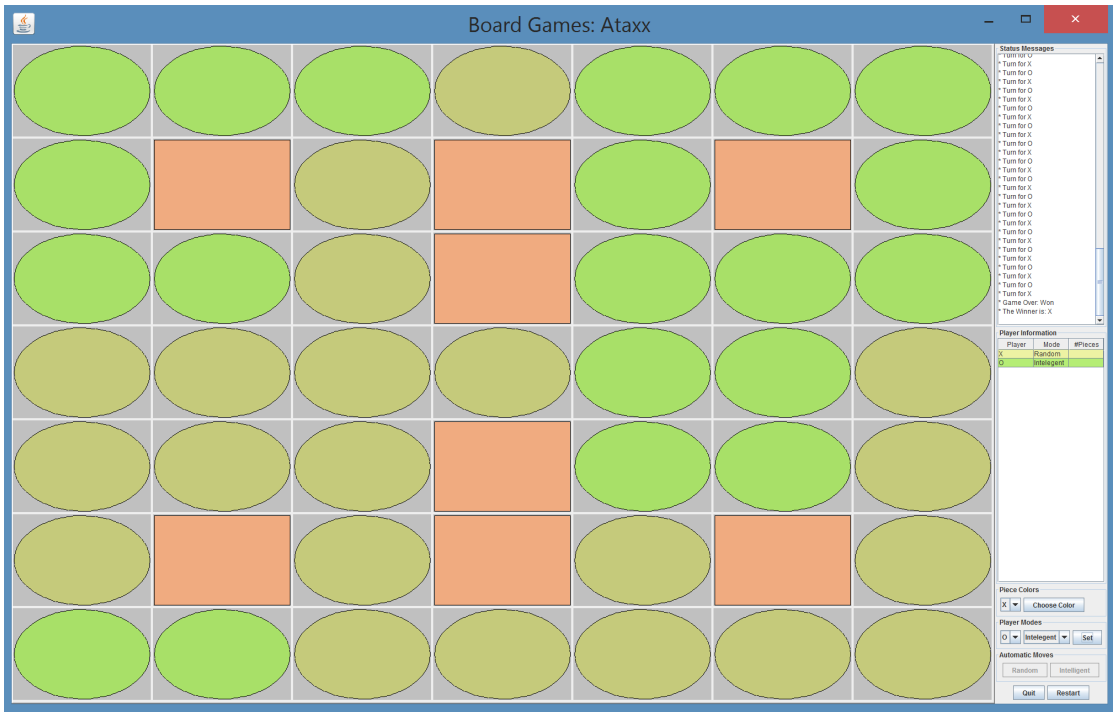


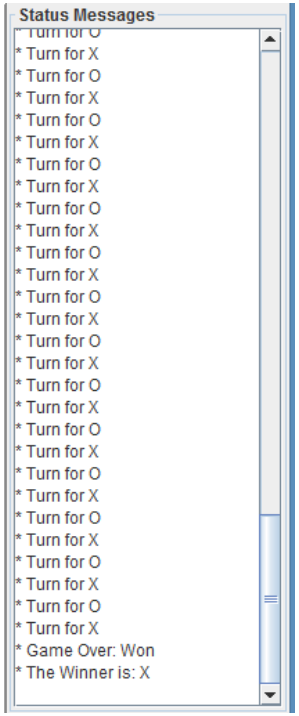
Status Messages



Status Messages

- Cuando termina el juego se debe mostrar:
  - ★ El mensaje "*Game Over!*".
  - ★ El estado de finalización del juego ("*Stopped*", "*Won*" o "*Draw*").
  - ★ El nombre del ganador (si hay ganador).



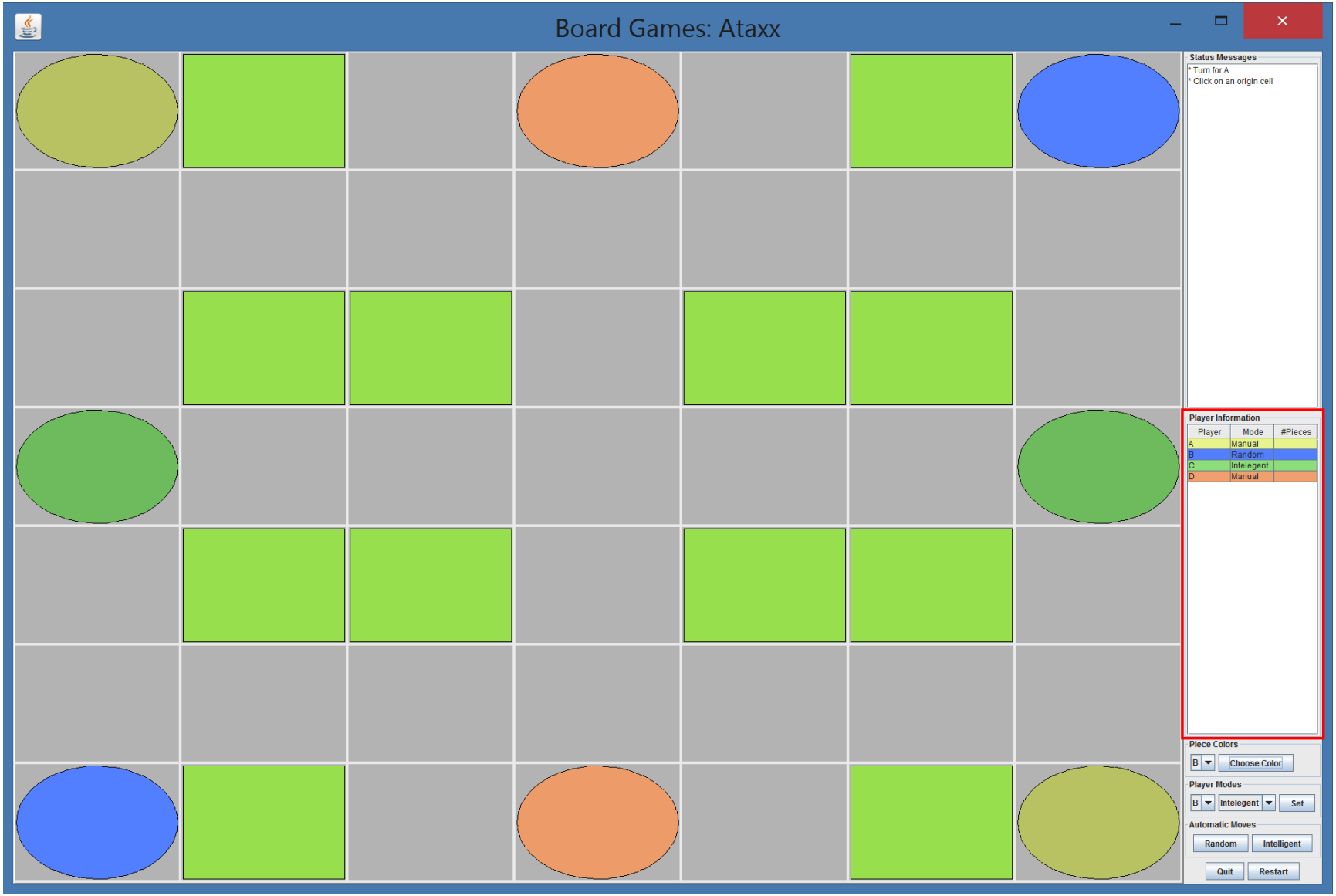


- Puedes mostrar cualquier otro mensaje que creas necesario: cuando se cambia el modo de juego, cuando se inicia o termina un movimiento, etc.

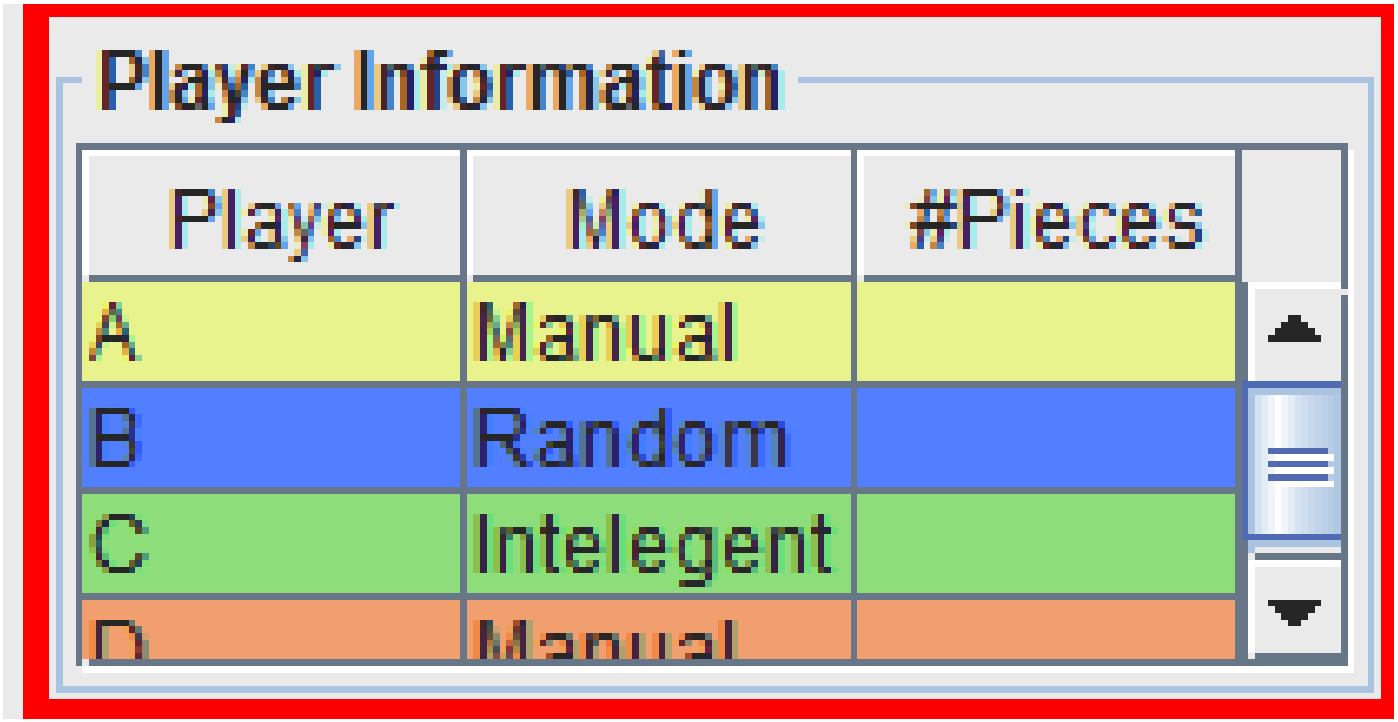
Player Information

- Este área de la ventana está formado por una tabla de solo lectura (implementada utilizando la clase `JTable` con un `JScrollPane`).
- Contiene la siguiente información de cada jugador:
  1. El **identificador de la ficha**.
  2. El **modo de juego**: en el modo `múltiples ventanas` solo se debe mostrar el modo de juego del jugador al que pertenece la ventana (no se sabe el modo de los demás jugadores).
  3. El **contador de fichas**: si `board.getPieceCount(piece)` devuelve un valor distinto de `null` (por ejemplo, en *Advanced Tic-Tac-Toe*).
- El color de fondo de cada línea de la tabla debe ser el color de la ficha correspondiente.

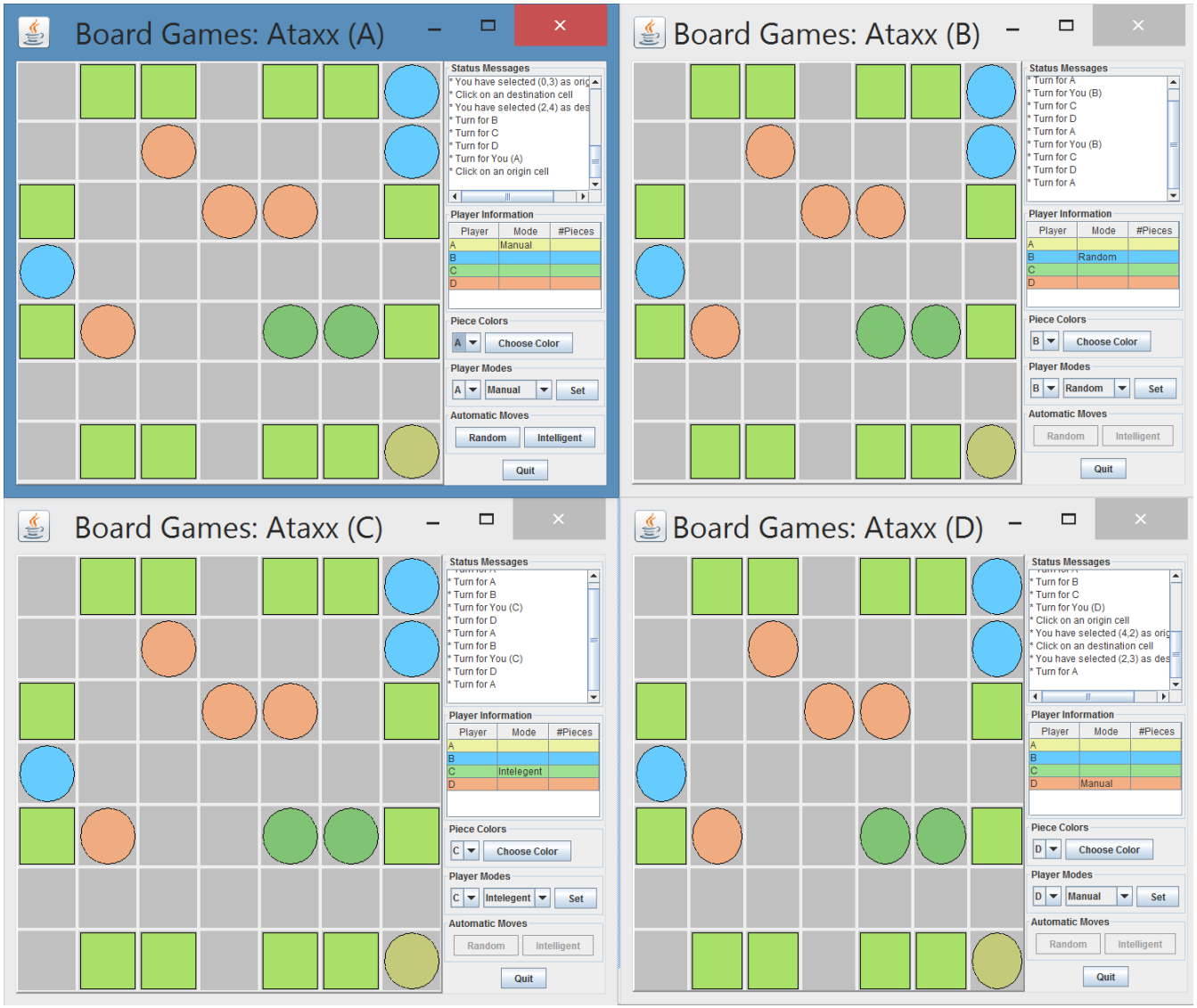
Player Information



Player Information



Player Information

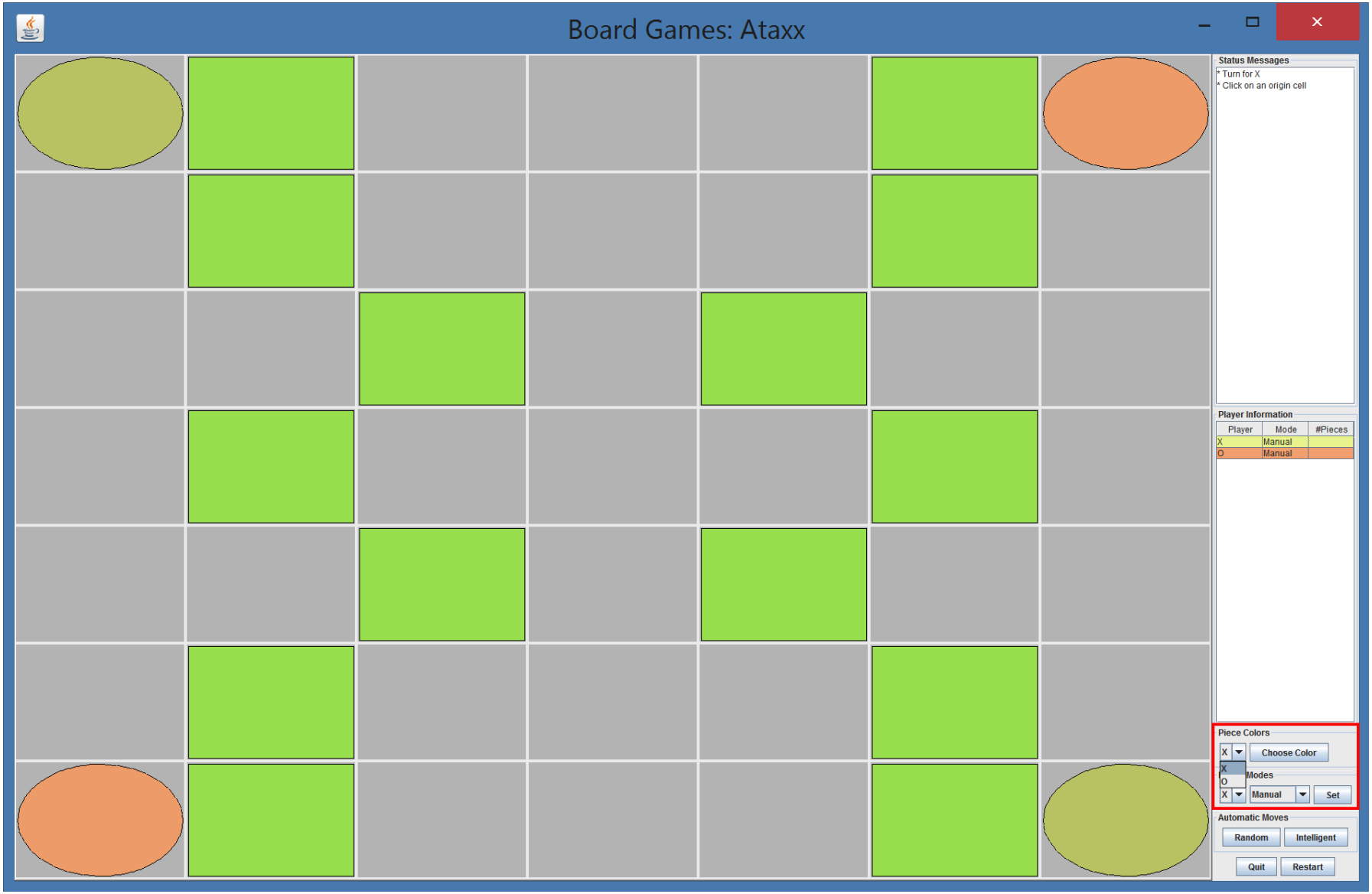



```
-g ataxx -o 4 -p A:m,B:r,C:a,D:m -m
```

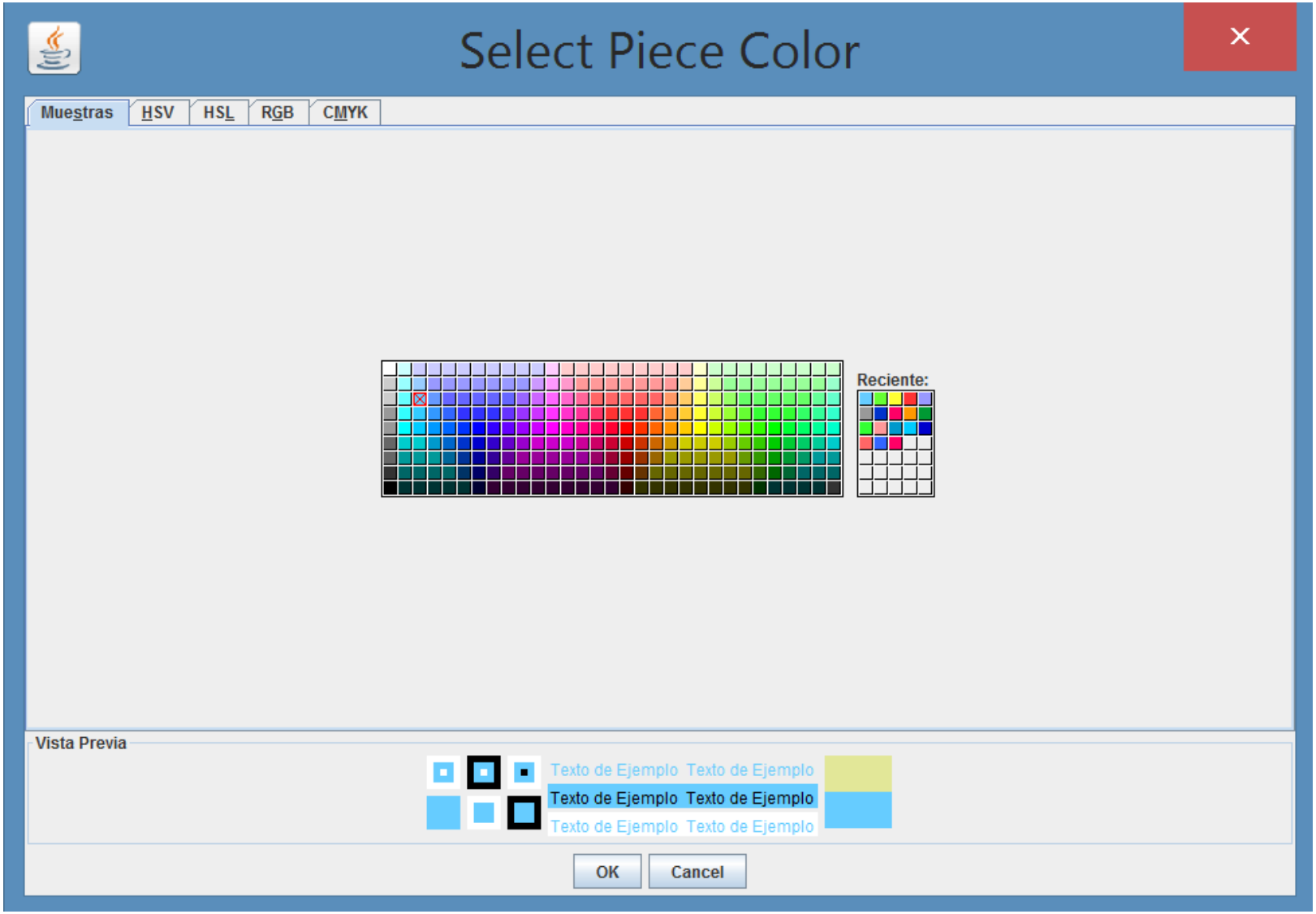
Piece Colors

- Este área de la ventana permite cambiar el color de todas las fichas.
- Estos colores se deben utilizar cuando se dibuja el tablero.
- Se debe permitir cambiar el color de las fichas de los jugadores.
- No es obligatorio permitir cambiar el color de los obstáculos de *Ataxx*.
- Cuando se cambia un color, se debe reflejar inmediatamente en el tablero y en el área "*Players Information*" de la ventana.
- Observa que en el caso de `ventana única` esto solo afecta a los colores de la ventana local. Por ejemplo, cada ventana usa colores diferentes para el mismo juego.
- Esta parte debería implementarse usando `JColorChooser`.

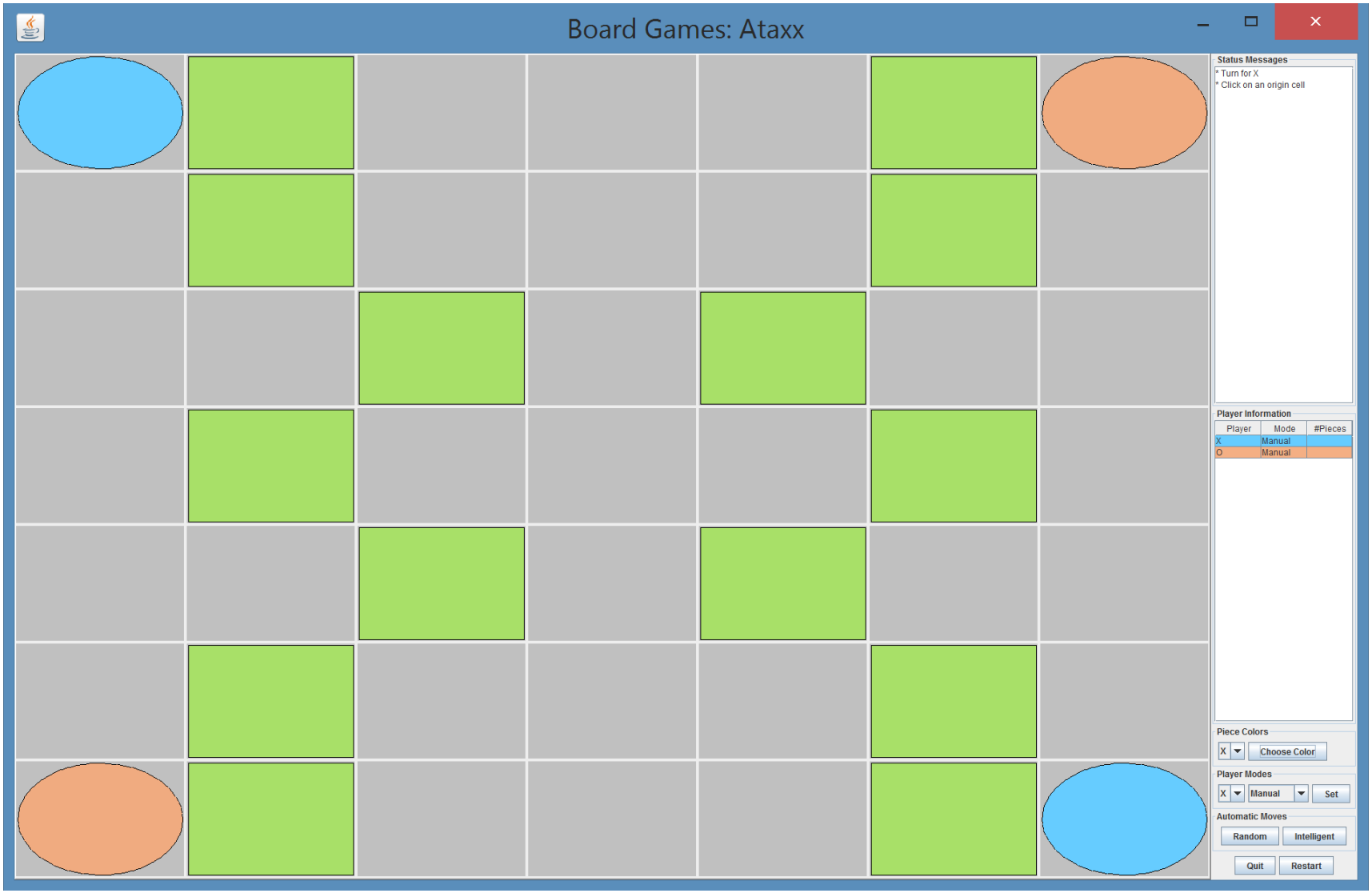
Piece Colors



Piece Colors



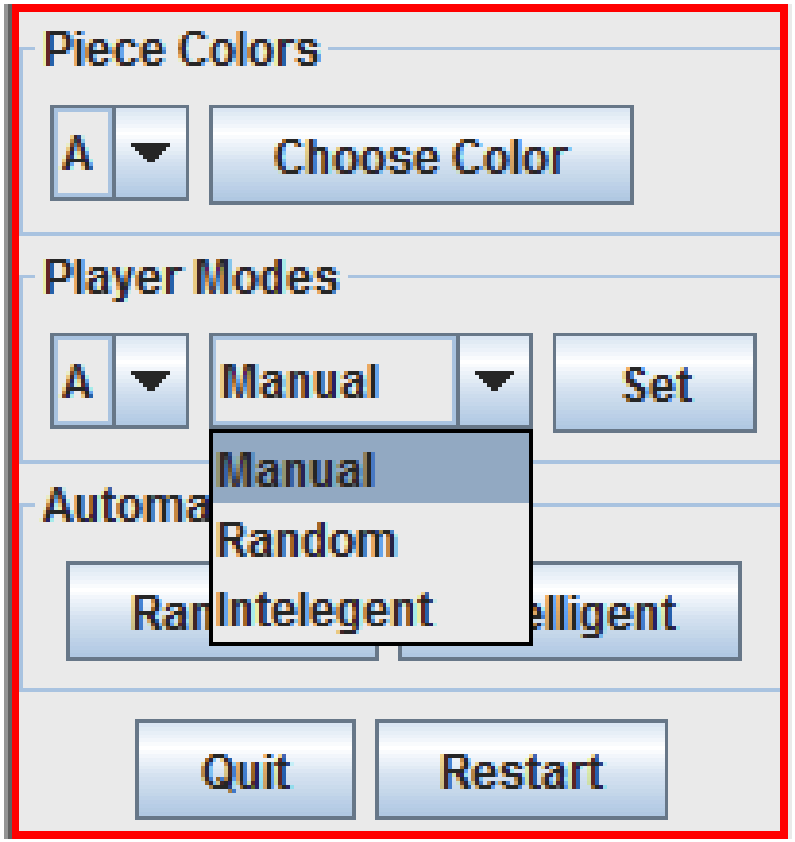
Piece Colors



Player Modes

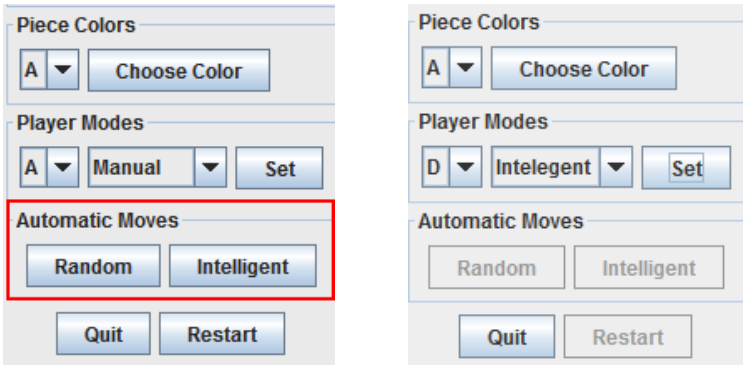
- Este área de la ventana permite cambiar el modo de juego de los distintos jugadores durante la partida.
- En el modo `múltiples ventanas` solo se puede cambiar el modo de juego del jugador al que pertenece la ventana.
- En este caso, el primer *combo-box* solo debe contener el identificador del jugador.
- Estos componentes siempre estarán activos para permitir el cambio del modo de juego de los jugadores en cualquier momento de la partida.
- El cambio en el modo de juego de un jugador se debe reflejar inmediatamente en la tabla de "*Player Information*" y el jugador debe jugar en ese modo la próxima vez que le toque el turno.
- **Si los jugadores `RANDOM` e `INTELLIGENT` no están disponibles, este área de la ventana no debe estar disponible en la vista.**
- **Si uno de los dos es `null`, no debe mostrarse la opción correspondiente en el *combo-box*.**

Player Modes



Automatic Moves

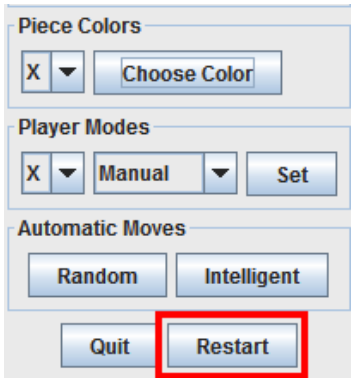
- Los jugadores en modo `MANUAL` pueden usar los botones de este área de la ventana para hacer un *único* movimiento aleatorio o inteligente.

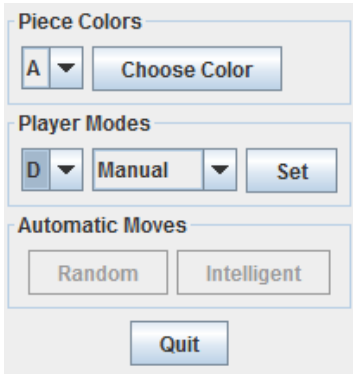


- **Si los jugadores de los modos `RANDOM` e `INTELLIGENT` no están disponibles, este área de la ventana no debe estar disponible en la vista.**
- **Si uno de los dos es `null`, no debe mostrarse el botón correspondiente en la vista.**
- **Debe estar deshabilitado durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.**

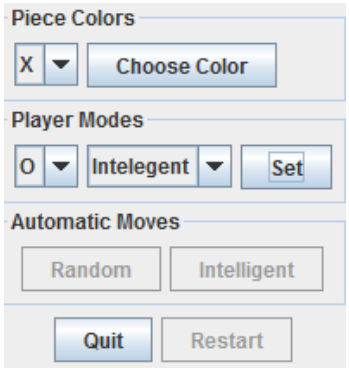
Restart Button

- Este botón debe aparecer solo en el modo `ventana única` y puede utilizarse para reiniciar la partida (llamando al método `restart()` del controlador).



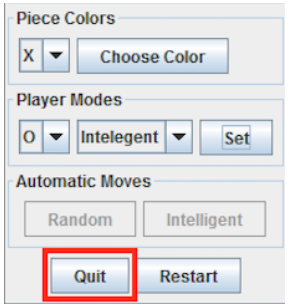


- **Debe estar deshabilitado durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.**

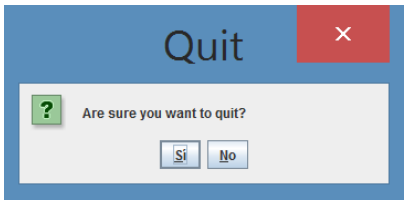


Quit Button

- Este botón permite salir del juego.

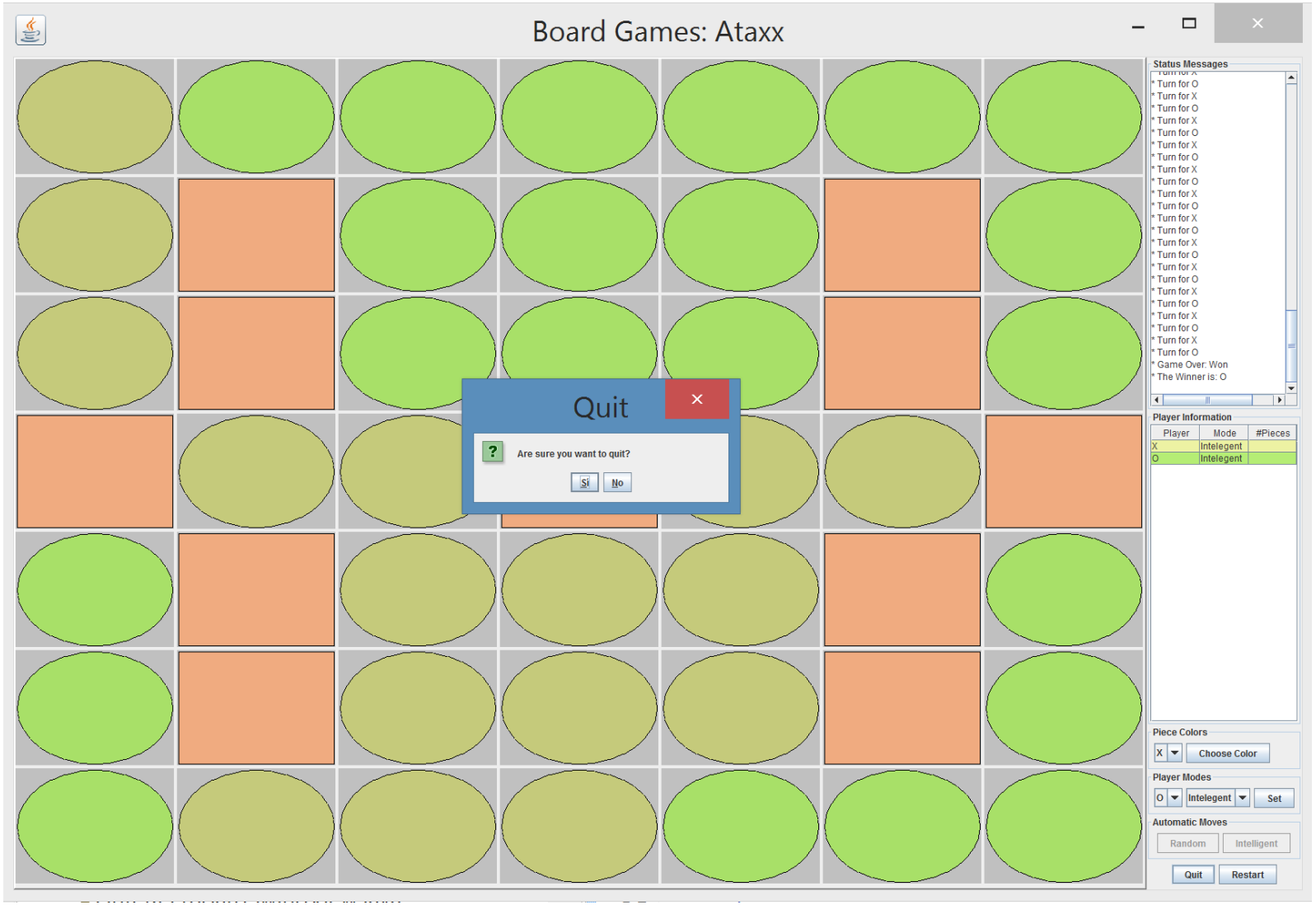


- Cuando se pulsa sobre él, debe aparecer un *dialog box* que solicite al usuario confirmación.



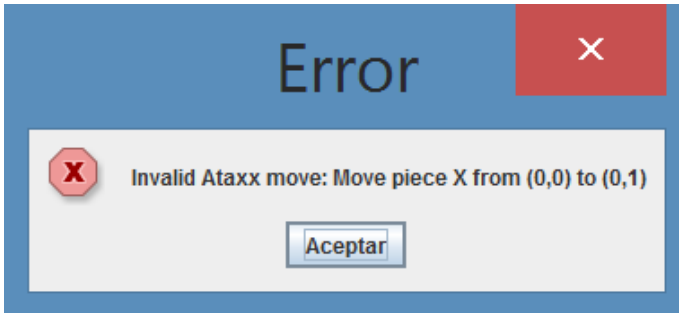
- Antes de salir, debe pararse el juego con el método `stop()` del controlador.
- **Debe estar deshabilitado durante la ejecución de un movimiento y cuando es el turno de un jugador que no está en modo `MANUAL`.**

Quit Button


Board Games: Ataxx
Status Messages
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Turn for X
* Turn for O
* Game Over: Won
* The Winner is: O
Quit
Are sure you want to quit?
Si
No
Player Information
Player
Mode
#Pieces
X
Intelegent
O
Intelegent
Piece Colors
X
Choose Color
Player Modes
O
Intelegent
Set
Automatic Moves
Random
Intelligent
Quit
Restart

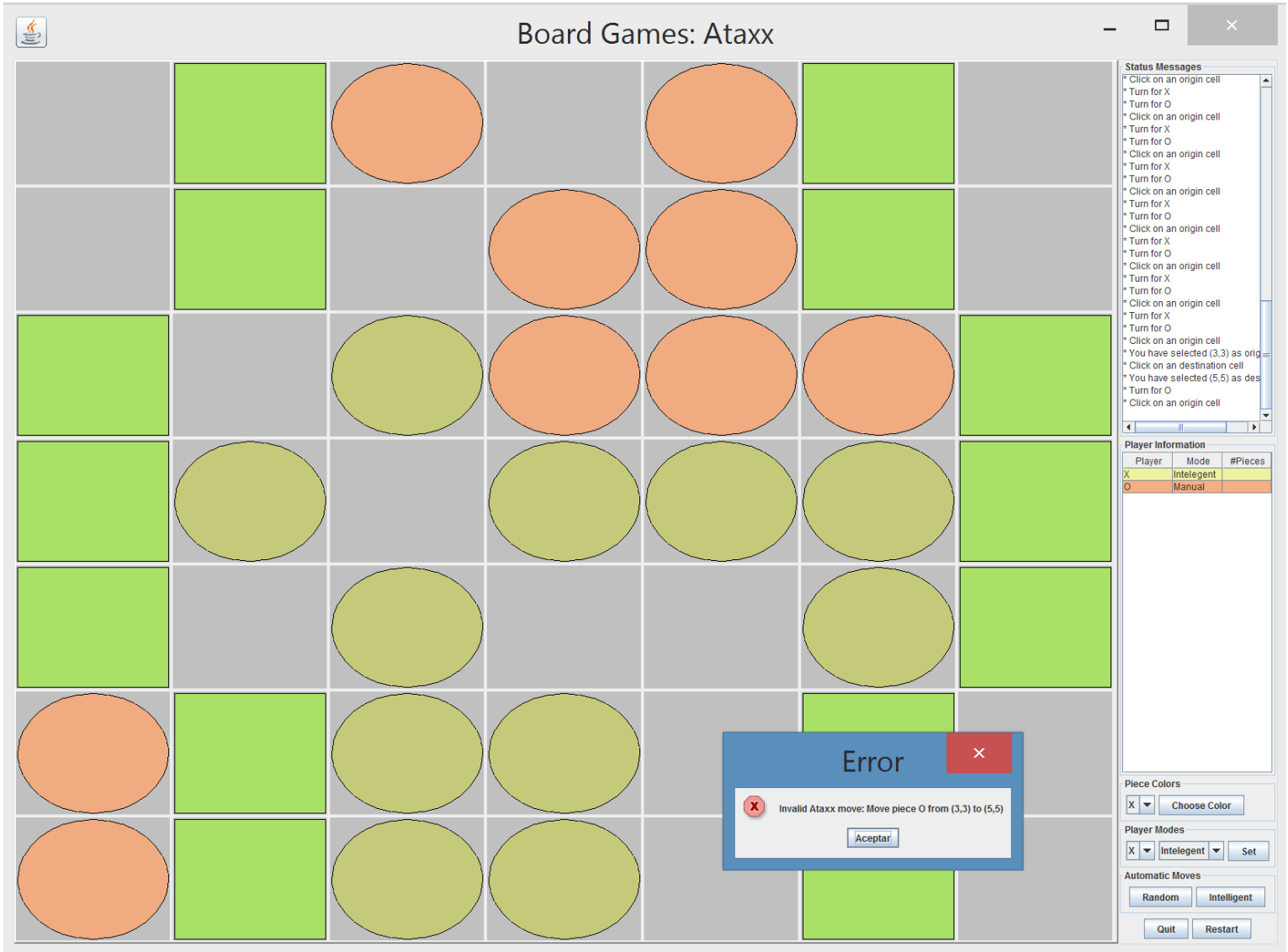
Mensajes de error

- Los mensajes de error, como los recibidos mediante el método `GameObserver.onError(...)`, deben mostrarse utilizando *Dialog boxes*.



- En el modo `múltiples ventanas` los errores que ocurren durante la ejecución de un movimiento solo deben mostrarse en la ventana del jugador que hizo el movimiento.

Mensajes de error



Otros requisitos

- A continuación se indican algunos requisitos adicionales que no corresponden a ninguno de los apartados anteriores:
  1. En *Ataxx*, después de seleccionar una ficha de origen, el usuario puede cancelar la selección utilizando, por ejemplo, el botón derecho del ratón sobre el tablero.
  2. Además, si después de seleccionar la ficha de origen se cambia el modo de juego, se debe cancelar la selección.
  3. **No está permitido utilizar herramientas de generación automática de interfaces gráficos de usuario (como NetBeans).**

¡Importante! Cambio de funcionamiento en *Advanced Tic-Tac-Toe*

Ahora hay dos tipos de movimiento:

- Al principio los jugadores deben ir colocando fichas como en *Tic-Tac-Toe*.
- Cuando un jugador se queda sin fichas, debe **mover las fichas que ya están en el tablero a otras posiciones que estén libres**.

Modificación de las factorías de juego

- La creación de la vista debe realizarse mediante una llamada al método `createSwingView` de la subclase de `GameFactory` que corresponda.
- **Es necesario modificar las factorías de todos los juegos existentes.**
- **No se debe modificar ningún fichero del paquete `basecode`.**
- Debes hacer lo siguiente para cada juego (lo explicamos para *Connect-N*):
  1. Crear un paquete nuevo `practica5.connectN`.
  2. Crear una clase `practica5.connectN.ConnectNFactoryExt` que extienda `ConnectNFactory`.
  3. Definir el método `createSwingView` en `ConnectNFactoryExt`, sobreescribiendo el que está definido en `ConnectNFactory`.
  4. Modificar `Main.java` para que utilice `ConnectNFactoryExt` en lugar de `ConnectNFactory`.
- Es recomendable hacer esto incluso para el juego *Ataxx* que has creado en la práctica 4 para evitar la modificación del código de `practica4` (excepto los errores que hayas corregido en este código).

Modificación de `Main.java`

- En `Main.java` debes modificar el método `startGame` para que se permita el modo de vista `ventana`.
- Debes reemplazar la excepción que se lanza en el caso de elegir el modo ventana por el código que corresponda.
- El *parser* de la línea de órdenes ya procesa las opciones `-v` y `-m`.
- Para crear una vista `ventana` en el caso de `ventana única` puedes hacer lo siguiente:

```
gameFactory.createSwingView(g, c, null,
                    gameFactory.createRandomPlayer(),
                    gameFactory.createAIPlayer(aiPlayerAlg));
```

- Para crear una vista `ventana` para el caso `múltiples ventanas` puedes utilizar lo siguiente:

```
for (Piece p : pieces) {
   gameFactory.createSwingView(g, c, p,
                    gameFactory.createRandomPlayer(),
                    gameFactory.createAIPlayer(aiPlayerAlg));
}
```

Entrega de la práctica

- **La práctica debe entregarse en un único archivo comprimido utilizando el mecanismo de entregas del campus virtual, no más tarde del 22 de abril de 2016 a las 16:00h.**

- El fichero debe tener al menos el siguiente **contenido**:
  - Directorio **src** con el código fuente del programa.
  - Fichero **alumnos.txt** donde se indicará el nombre de los componentes del grupo.
  - Directorio **doc** con la documentación generada automáticamente sobre el código que has implementado.