Programando el juego Go

Oliva Ramírez Adrián Fernando, NUA: 424647, af.olivaramirez@ugto.mx
Fonseca Uribe Oswaldo Emmanuel, NUA: 427289, oe.fonsecauribe@ugto.mx

8 de diciembre de 2021

1. Introducción

El juego de mesa llamado Go, es el juego más antiguo del que se tiene conocimiento. Es un juego de estrategia de dos jugadores y el objetivo es conquistar y controlar más área del tablero que el contrincante. Las reglas del juego son sencillas de entender, pero es difícil jugar bien.

En este proyecto nos apoyaremos del paradigma de Programación Orientada a Objetos para poder programar el juego de Go.

2. Objetivo

Usar `pygame` con gráficos para programar un tablero de Go jugable. También investigar maneras para que se pueda jugar con otras personas en el internet.

3. Justificación

El juego de Go tiene pocas reglas y es sencillo de entender. El problema es que existen detalles, como calcular el puntaje final, que no son sencillos de programar. Incluso apenas en 2015 es que la primera máquina, *AlphaGo*, logró vencer a un jugador profesional. Se necesitó de inteligencia artificial para poder programar a *AlphaGo*.

Las fichas que se usan pueden formar grupos cuando están juntas. Al estar en grupos, empiezan a compartir de sus propiedades y se hacen más fuertes. Este será el principal uso de Programación Orientada a Objetos, pues necesitaremos guardar y manejar dichas propiedades para poder programar el juego.

4. Marco Teórico

Demos una breve introducción de los tecnicismos y algunas reglas de Go. [1]

- **El tablero.** Los tableros son una cuadrícula de 9, 13 o 19 líneas horizontales y verticales de forma estándar. Las fichas son circulares y se ponen en las intersecciones vacías de la cuadrícula. Existen dos jugadores y cada quien escoge un color de ficha, ya sea negro o blanco. El de las fichas de color negro siempre será el que empieza poniendo la primera ficha. Después se intercalan los jugadores para ir poniendo fichas. Véase la figura 1 para ver un ejemplo de tablero de Go de 9×9.
- **Grupos.** Las fichas se pueden agrupar y hacerce más fuertes. Estas solamente se consideran conectadas cuando son adyacentes a sí mismas, o puedes conectarlas pasando por fichas adyacentes del mismo color. Véase la figura 2
- **Libertades.** Lo principal de un grupo son el número de "libertades" que tienen. Las "libertades" son aquellas intersecciones vacías adyacentes al grupo. Véase la figura 3.
- **Captura.** Como se pudo apreciar anteriormente, un grupo de fichas puede perder libertades. Al momento de que un grupo pierda todas las libertades, es capturado. El número de fichas que fueron capturadas restarán puntos, esto principalmente para penalizar al jugador. Véase la figura 4 para un ejemplo más visual.

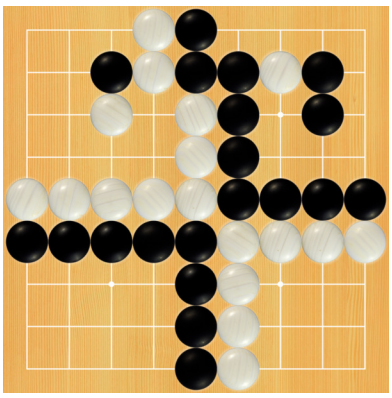
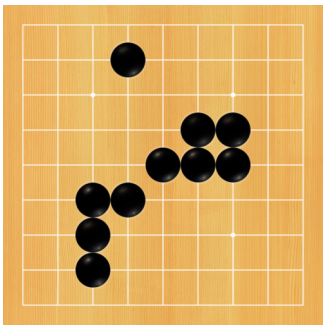
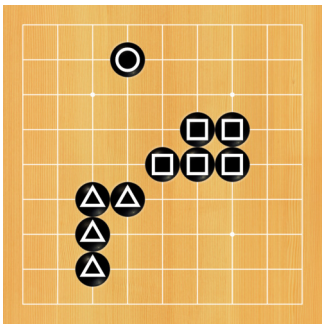


Figura 1: Un ejemplo de juego de Go.



(a) Ejemplo usando solamente fichas negras.

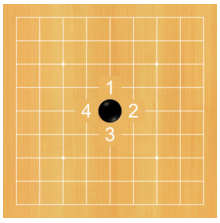


(b) Usando figuras, distinguimos los grupos que hay en el tablero.

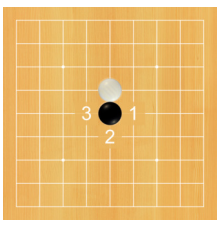
Figura 2: Ejemplificando los grupos.

- **Fin del juego.** Hay dos maneras en que el juego puede terminar. La primera es por resignación de alguno de los jugadores. La otra implica que ambos jugadores "pasaron". Cualquier jugador siempre puede "pasar", es decir, no hace ningún movimiento en su turno. Esto debido a cualquier movimiento que vea le perjudica en lugar de ayudar.

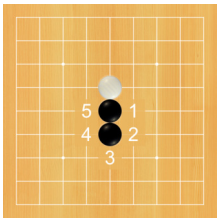
  Si el juego termina porque ambos jugadores pasaron, se cuentan los puntos. Siendo Go un juego sobre conquistar territorio, los puntos son los espacios vacíos del tablero que cada jugador rodea. Las fichas consideradas capturadas se retiran del tablero y son restados al puntaje final. Existe la convención de regalar puntos al jugador con fichas blancas, pues tuvo la desventaja de empezar después. Veamos un ejemplo muy sencillo en la figura .



(a) Un grupo de una sola ficha puede tener hasta 4 libertades.

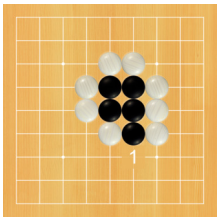


(b) Cuando una ficha enemiga se pone sobre una libertad, dicha libertad se pierde.

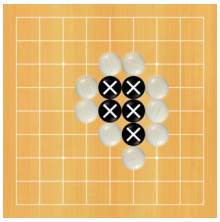


(c) Las fichas en grupo comparten libertades.

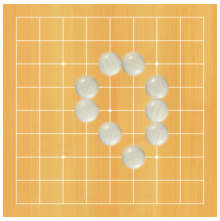
Figura 3: Ejemplificando las libertades.



(a) Cuando a un grupo de fichas solamente le queda una libertad, se puede decir que está en *atari*.

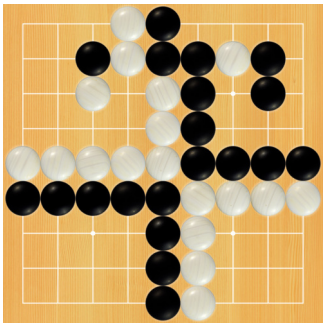


(b) El grupo entero es capturado cuando todas sus libertades se perdieron.

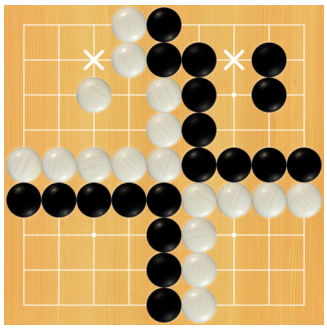


(c) Cuando el grupo es capturado, es removido del tablero. Esta es la única manera de quitar fichas del tablero.

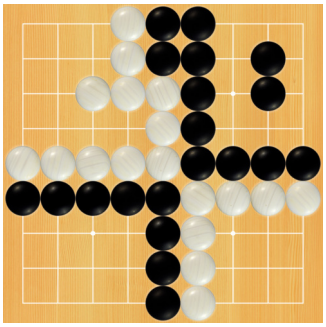
Figura 4: Ejemplificando las libertades.



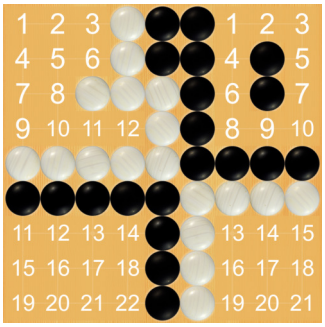
(a) Supongamos que este juego terminó porque ambos jugadores pasaron.



(b) Se quitan del tablero las fichas que ambos jugadores estén de acuerdo que serán capturadas, sin importar qué movimientos hagan.



(c) Una manera sencilla de "restar" las fichas capturadas, es ponerlas en el territorio. Como el juego cuenta el territorio, es como restar los puntos directamente.



(d) Se cuentan las intersecciones vacías del territorio de cada jugador, pudiendo regalar puntos al de fichas blancas por empezar después. Gana quien tenga más puntos.

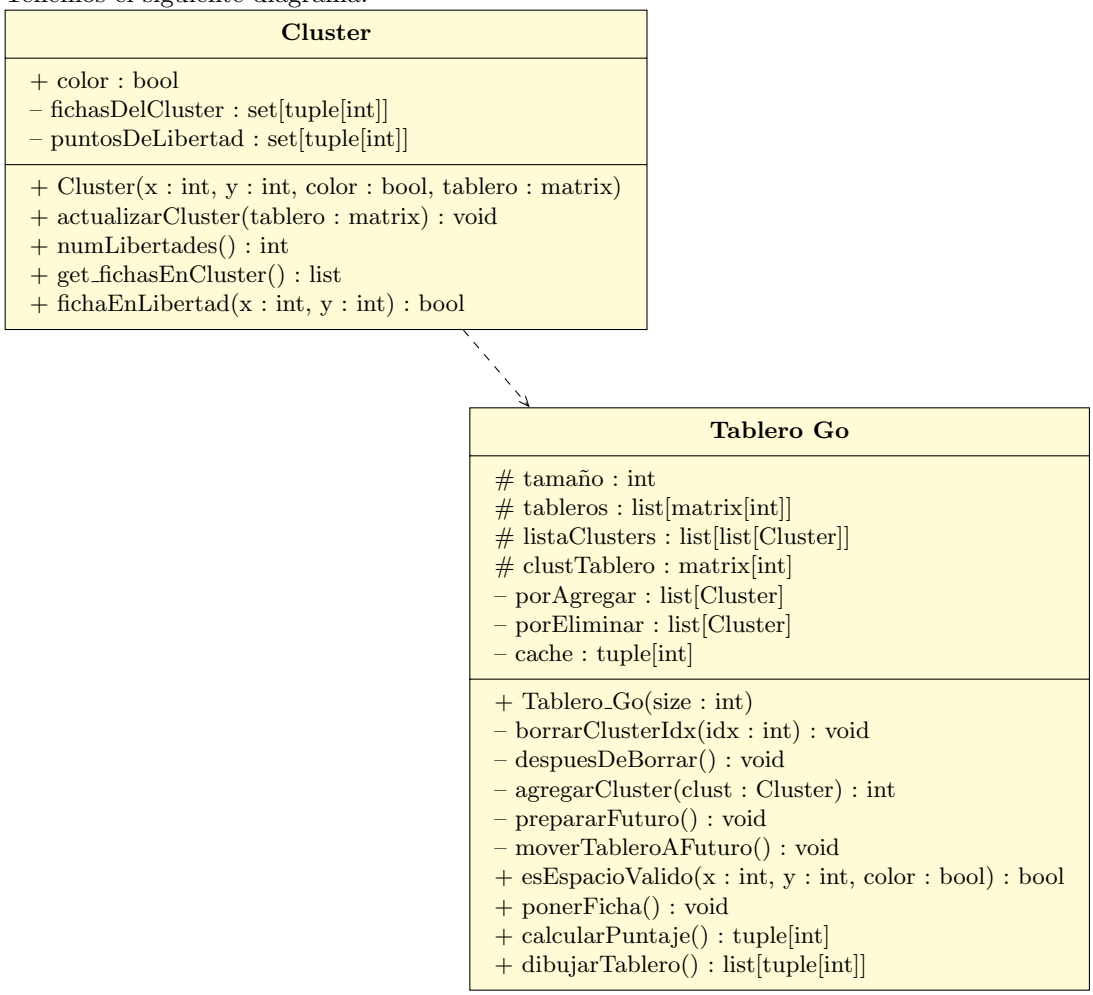
Figura 5: Ejemplificando el final de un juego de Go.

5. Diagrama de Clases

Para facilitar el trabajo, usaremos una clase `Cluster` para guardar los grupos de las fichas. Aquella clase podrá actualizar dichos grupos y llevar registro de las "libertades".

Por otra parte, tendremos la clase del `Tablero Go`. Aquel guardará el tablero entero, los clusters del tablero y se encargará de que todas las reglas de Go sean aplicadas en cada movimiento. Heredaremos las clases `Tablero 19`, `Tablero 13` y `Tablero 9`, donde el número indica el tamaño del tablero. Esto para poder adaptar la función `dibujarTablero()` para cada tablero.

Tenemos el siguiente diagrama:



6. Actividades de Programación

- ☑ Esbozar un diagrama de clases base para la programación. [A]
- ☑ Realizar una investigación detallada sobre el juego GO y sus reglas. [A]
- ☑ Detallar un algoritmo que permita calcular los puntajes de cada jugador al final de la partida. [A]
- ☑ Renderizar imágenes para los menús y objetos del juego. [O]
- ☑ Programar una interfaz usando `pygame` para personalizar una partida y darle comienzo. [O]
- ☑ Realizar la programación para interactuar con la interfaz de juego (Botones y textos) [O]
- ☑ Implementar el método para colocar fichas en el tablero y que actuen conforme a las reglas del juego. [A]
- ☑ Programar una interfaz de seguimiento para los jugadores (Botones de "Pasar", Resignarse", Cronómetro, Turnos) [O]

- ☑ Diseñar e implementar un evento de "Fin de Partida". [A]
- ☐ Implementar un cronómetro a la partida. (No se pudo realizar correctamente)

Referencias

[1] Wikipedia contributors. (2021, Octubre 23). Rules of Go. In *Wikipedia, The Free Encyclopedia*. Visitado el 05:29, diciembre 4, 2021, desde https://en.wikipedia.org/w/index.php?title=Rules_of_Go&oldid=1051465754.

[2] Andrea Carta (2018) *A static method for computing the score of a Go game*, http://www.micini.net/. Visitado el 2021-11-29 en https://www.oipaz.net/Carta.pdf.

[3] Tim Ruscica (2019) *Python Online Multiplayer Game Development Tutorial*, www.freecodecamp.org. Visitado el 2021-11-29 en https://www.youtube.com/watch?v=McoDjOCb2Zo.