Juego de Piedra, Papel o Tijera
IA vs jugador

Autora: Andrea Loo
Institución: Universidad Latina de Panamá
Fecha: 22 de agosto del 2023

Resumen

En el informe se desarrollará la implementación del juego de Piedra, Papel, o Tijera en Python 3 utilizando la biblioteca tkinter. Este juego se desarrollará en una interfaz gráfica en el que el jugador compite contra la computadora, esta aprende y adapta su estrategia a medida que avanza el juego. El juego usa una clase de LearningComputer para la IA y clase GameGUI para la interfaz gráfica de juego.

1. Introducción

El juego de Piedra, Papel o Tijera es un juego popular que se ha jugado por mucho tiempo. El juego trata de agilidad mental y decisiones estratégicas, se juega entre dos personas y cada uno de los jugadores tiene tres (3) opciones para elegir: “piedra”, “papel” o “tijera”. La piedra le gana a la tijera, la tijera al papel y el papel a la piedra. Este juego simple se lleva de la realidad a lo digital a través de un código.

En esta versión digital, se reemplazó a uno de los jugadores por inteligencia artificial (IA) en forma de una instancia de la clase “LearningComputer”. La IA está programada para realizar elecciones de manera aleatoria, a parte de eso también aprende de las decisiones del jugador y adapta su estrategia según lo que aprende.

2. Palabras claves

- LearningComputer
- GameGUI
- Tkinter

3. Objetivos

- Utilizar los conocimientos obtenidos en clase y aplicarlos en un proyecto mediante el lenguaje de programación python y probar que funcione.
- Explorar la interacción entre la programación orientada a objetos, la adaptación de estrategias y la creación de interfaces gráficas.
- Demostrar cómo una IA sencilla puede mejorar la experiencia del juego al crear un oponente desafiante que evoluciona constantemente.

4. Contenido

4.1 ¿Qué es clase “LearningComputer”?

La clase LearningComputer (LC) representa la inteligencia artificial (IA) que asume el papel del oponente del jugador. A diferencia de una IA convencional que toma decisiones de manera estática, la LearningComputer aprende y adapta su estrategia según las elecciones previas del jugador.

Funcionamiento de la clase

A continuación, se describen los componentes clave de esta clase:

- moves: lista de los posibles movimientos del juego, es decir, "piedra", "papel" y "tijera". Estos movimientos son utilizados tanto por el jugador como por la LC.
- history: diccionario que almacena el historial de movimientos del jugador y la LC. Cada movimiento se registra junto con el recuento de veces que la LC ha respondido a ese movimiento específico con cada uno de sus movimientos posibles.
- prev_player_move: guarda el movimiento previo realizado por el jugador. Esto es importante para que la LC pueda aprender y adaptarse.
- make_move(): determina el movimiento que la LC realizará basado en su historial y las elecciones del jugador. Si es la primera ronda o si no ha habido suficientes datos históricos, la LC elige un movimiento aleatorio. De lo contrario, analiza el historial y responde basándose en la opción que ha tenido más éxito frente al movimiento previo del jugador.
- beat(move): devuelve el movimiento que vence al movimiento proporcionado. Por ejemplo, si move es "piedra", este método devolverá "papel".
- learn(player_move): actualiza el historial de movimientos basado en las elecciones del jugador. Esto permite que la LC ajuste su estrategia de los patrones observados.

```
class LearningComputer:
    def __init__(self):
        self.moves = ['piedra', 'papel', 'tijera']
        self.history = {'piedra': {'piedra': 0, 'papel': 0,
'tijera': 0},
                        'papel': {'piedra': 0, 'papel': 0,
'tijera': 0},
                        'tijera': {'piedra': 0, 'papel': 0,
'tijera': 0}}
        self.prev_player_move = None

    def make_move(self):
        if self.prev_player_move is None or
sum(self.history[self.prev_player_move].values()) == 0:
            return random.choice(self.moves)

        last_move = max(self.history[self.prev_player_move],
key=lambda key: self.history[self.prev_player_move][key])
        return self.beat(last_move)

    def beat(self, move):
        return {'piedra': 'papel', 'papel': 'tijera',
'tijera': 'piedra'}[move]

    def learn(self, player_move):
        if self.prev_player_move is not None:
            self.history[self.prev_player_move][player_move]
+= 1
        self.prev_player_move = player_move
```

Adaptación Dinámica

La clave de la LC radica en su capacidad para aprender y adaptarse a lo largo del juego. Mediante el seguimiento de las elecciones del jugador y el análisis de las tendencias históricas, la LC es capaz de modificar su estrategia para mejorar sus posibilidades de ganar.

La implementación de la clase LearningComputer destaca la flexibilidad y adaptabilidad que puede lograrse mediante la programación orientada a objetos. Además, ilustra cómo incluso las estrategias simples pueden enriquecer la experiencia del juego y proporcionar desafíos para el jugador.

4.2 ¿Qué es GameGUI?

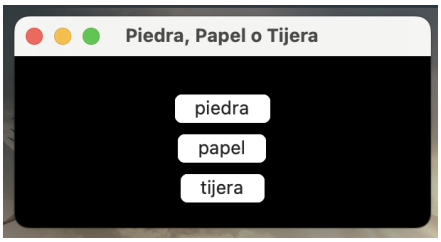
La clase GameGUI crea la interfaz gráfica que permite a los jugadores interactuar con el juego de manera intuitiva. La GameGUI no solo presenta los elementos

visuales del juego, sino que también coordina la interacción entre el jugador y la LC.

Diseño de la Interfaz Gráfica

La clase GameGUI utiliza la biblioteca tkinter de Python para crear una ventana de interfaz gráfica interactiva. En la creación de la ventana, se incluyen los siguientes componentes clave:

- root: La instancia de la ventana principal de tkinter.
- computer: Una instancia de la clase LC, que representa la inteligencia artificial con la que el jugador competirá.
- player_score y computer_score: Variables que almacenan las puntuaciones del jugador y la LC.



Elementos de la Interfaz

La interfaz gráfica de la clase GameGUI se compone de varios elementos interactivos que permiten al jugador interactuar con el juego:

- label: muestra al jugador el mensaje "Elige piedra, papel o tijera".
- buttons: serie de botones, uno para cada movimiento posible del jugador ("piedra", "papel" y "tijera"). Cada botón está asociado con el método play(player_move) que se activa cuando el jugador hace clic en un botón.
- score_label: Una etiqueta que muestra la puntuación actual del jugador y la LC.

Método play(player_move)

El método play(player_move) se activa cuando el jugador hace clic en uno de los botones de movimiento. Coordina el proceso del juego, incluyendo la elección de movimiento de la LC, la determinación del resultado de la ronda y la actualización de la puntuación en la interfaz gráfica.

```
class GameGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Piedra, Papel o Tijera")

        self.computer = LearningComputer()
        self.player_score = 0
        self.computer_score = 0

        self.label = tk.Label(root, text="Elige piedra,
papel o tijera:")
        self.label.pack()

        self.buttons = []
        for move in self.computer.moves:
            button = tk.Button(root, text=move,
command=lambda m=move: self.play(m))
            button.pack()
            self.buttons.append(button)

        self.score_label = tk.Label(root, text="Marcador:
Jugador 0 - Computadora 0")
        self.score_label.pack()

    def play(self, player_move):
        computer_move = self.computer.make_move()

        if player_move == computer_move:
            result = "Empate"
        elif self.computer.beat(player_move) ==
computer_move:
            result = "Gana la computadora"
            self.computer_score += 1
            self.computer.learn(player_move)
        else:
            result = "Ganas tú"
            self.player_score += 1
            self.computer.learn(player_move)

        self.score_label.config(text=f"Marcador: Jugador
{self.player_score} - Computadora {self.computer_score}")
        messagebox.showinfo("Resultado", f"Computadora
eligió {computer_move}\n{result}")
```

Interacción Dinámica

La clase GameGUI no solo presenta los elementos visuales del juego, sino que también interactúa con la LC para asegurar una experiencia de juego dinámica. La puntuación se actualiza en tiempo real y se muestra al jugador el resultado de cada ronda.

4.3 ¿Qué es tkinter?

tkinter es una biblioteca de GUI que forma parte de la distribución estándar de Python. Proporciona un conjunto de módulos y clases que permiten la creación y gestión de elementos gráficos, como ventanas, botones, etiquetas y campos de entrada. Al utilizar tkinter, los desarrolladores pueden crear aplicaciones con interfaces gráficas atractivas y funcionales.

5. Código

```
import random
import tkinter as tk
from tkinter import messagebox

class LearningComputer:
    def __init__(self):
        self.moves = ['piedra', 'papel', 'tijera']
        self.history = {'piedra': {'piedra': 0, 'papel': 0,
'tijera': 0},
                        'papel': {'piedra': 0, 'papel': 0,
'tijera': 0},
                        'tijera': {'piedra': 0, 'papel': 0,
'tijera': 0}}
        self.prev_player_move = None

    def make_move(self):
        if self.prev_player_move is None or
sum(self.history[self.prev_player_move].values()) == 0:
            return random.choice(self.moves)

        last_move = max(self.history[self.prev_player_move],
key=lambda key: self.history[self.prev_player_move][key])
        return self.beat(last_move)

    def beat(self, move):
        return {'piedra': 'papel', 'papel': 'tijera',
'tijera': 'piedra'}[move]

    def learn(self, player_move):
        if self.prev_player_move is not None:
            self.history[self.prev_player_move][player_move]
+= 1
        self.prev_player_move = player_move

class GameGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Piedra, Papel o Tijera")

        self.computer = LearningComputer()
        self.player_score = 0
        self.computer_score = 0

        self.label = tk.Label(root, text="Elige piedra, papel
o tijera:")
        self.label.pack()

        self.buttons = []
        for move in self.computer.moves:
            button = tk.Button(root, text=move, command=lambda
m=move: self.play(m))
            button.pack()
            self.buttons.append(button)

        self.score_label = tk.Label(root, text="Marcador:
Jugador 0 - Computadora 0")
        self.score_label.pack()

    def play(self, player_move):
        computer_move = self.computer.make_move()

        if player_move == computer_move:
            result = "Empate"
        elif self.computer.beat(player_move) == computer_move:
            result = "Gana la computadora"
            self.computer_score += 1
            self.computer.learn(player_move)
        else:
            result = "Ganas tú"
            self.player_score += 1
            self.computer.learn(player_move)

        self.score_label.config(text=f"Marcador: Jugador
{self.player_score} - Computadora {self.computer_score}")
        messagebox.showinfo("Resultado", f"Computadora eligió
{computer_move}\n{result}")

if __name__ == "__main__":
    root = tk.Tk()
    game = GameGUI(root)
    root.mainloop()
```

6. Resultados y Conclusiones

El juego Piedra, Papel o Tijera utilizando la biblioteca tkinter y la clase LearningComputer ha resultado en una experiencia interactiva y entretenida.

Experiencia de Juego Dinámica

La combinación de la lógica del juego y la interfaz gráfica ha resultado en una

experiencia de juego dinámica y atractiva. Los jugadores pueden seleccionar sus movimientos haciendo clic en botones visuales, y las decisiones de la LearningComputer se reflejan de inmediato en la ventana de juego. Esto crea una sensación de competencia y emoción a medida que avanza el juego.

Adaptación de la LearningComputer

La LearningComputer aprende de las elecciones del jugador y ajusta su estrategia basadas en los resultados observados en el historial de movimientos. A medida que se desarrollan más rondas, la LearningComputer se vuelve más competente en sus elecciones y es capaz de predecir y contrarrestar las decisiones del jugador.

Interfaz Gráfica Intuitiva

La utilización de la biblioteca tkinter ha permitido crear una interfaz gráfica intuitiva y fácil de usar para el juego. Los botones de movimiento, las etiquetas de mensaje y las ventanas emergentes de resultado brindan una experiencia de usuario amigable que facilita la interacción con el juego.

Competencia y Diversión

El juego crea una competencia equilibrada y una sensación de diversión para los jugadores. La LearningComputer ofrece un desafío a medida que se adapta a las elecciones del jugador, lo que lleva a un juego más emocionante y envolvente. Los resultados variados y la capacidad de la LearningComputer para aprender y mejorar a lo largo del juego.

Futuras Extensiones

Si bien la implementación actual del juego es satisfactoria, existen posibles extensiones y mejoras que podrían explorarse en el futuro. Estas podrían incluir la implementación de estrategias de Machine Learning más avanzadas para la LearningComputer, la incorporación de modos de juego adicionales y la optimización de la interfaz gráfica para dispositivos móviles.

7. Referencia

- Documentación de python. Enlace: https://docs.python.org/es/3.11/

- Python GUI Programming With Tkinter. Enlace: https://realpython.com/python-gui-tkinter/

- Learn Python by Building a GUI Guessing Game with Tkinter. Enlace: https://levelup.gitconnected.com/learn-python-by-building-a-gui-guessing-game-with-tkinter-9f82291db6

- Interfaces gráficas de usuario con Tk. Enlace: https://docs.python.org/es/3/library/tk.html#:~:text=tkinter%20es%20un%20conjunto%20de,tutoriales%2C%20un%20libro%20y%20otros.