Reporte de avance de ambos proyectos.

Alexis Manuel Espinoza Martinez

Angel Uriel Mota Barreto

_

Ajedrez Avance

```
public void NextTurn()
{
    if (currentPlayer == "white")
    {
        currentPlayer = "black";
        GameObject.FindGameObjectWithTag("TurnText").GetComponent<Text>().text = currentPlayer + " turn";
    }
    else
    {
        currentPlayer = "white";
        GameObject.FindGameObjectWithTag("TurnText").GetComponent<Text>().text = currentPlayer + " turn";
    }
}
```

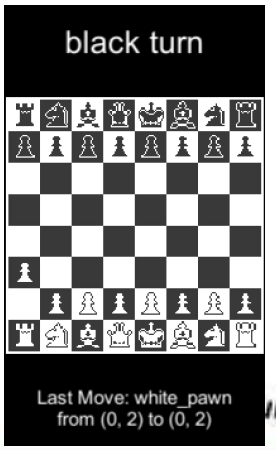
Código para mostrar los turnos

```
public void Winner(string playerWinner)
{
    gameOver = true;
    GameObject.FindGameObjectWithTag("TurnText").GetComponent<Text>().enabled = false;
    GameObject.FindGameObjectWithTag("WinnerText").GetComponent<Text>().enabled = true;
    GameObject.FindGameObjectWithTag("WinnerText").GetComponent<Text>().text = playerWinner + " is the winner";

    GameObject.FindGameObjectWithTag("RestartText").GetComponent<Text>().enabled = true;
}
```

Código para mostrar que jugador gano

```
// Obtener el último movimiento realizado
string lastMove = $"Last Move: {reference.name} from ({reference.GetComponent<Chessman>().GetXBoard()}, {reference.GetComponent<Chessman>().GetYBoard()}) to ({matrixX}, {matrixY})";

// Actualizar el texto en el canvas
controller.GetComponent<Game>().lastMoveText.text = lastMove;
```

Para mostrar el último movimiento(error que ocupa arreglarse)



Compilador Avance

```
static bool VerificarSintaxis(string codigo)
{
    int countLlaveAbierta = 0;
    int countLlaveCerrada = 0;
    int countParentesisAbierto = 0;
    int countParentesisCerrado = 0;
    int countComillas = 0;
    int countPuntoComa = 0;

    foreach (char c in codigo)
    {
        switch (c)
        {
            case '{':
                countLlaveAbierta++;
                break;
            case '}':
                countLlaveCerrada++;
                break;
            case '(':
                countParentesisAbierto++;
                break;
            case ')':
                countParentesisCerrado++;
                break;
            case '"':
                countComillas++;
                break;
            case ';':
                countPuntoComa++;
                break;
        }
    }
```

Se crean variables (contadores) para toda la estructura del ciclo y se aumenta el contador con cada uso de las llaves, comillas, paréntesis, etc.

```
    bool sintaxisCorrecta = true;

    if (countLlaveAbierta != countLlaveCerrada)
    {
        Console.WriteLine("Error: falta de llaves.");
        sintaxisCorrecta = false;
    }

    if (countParentesisAbierto != countParentesisCerrado)
    {
        Console.WriteLine("Error: falta de paréntesis.");
        sintaxisCorrecta = false;
    }

    if (countComillas % 2 != 0)
    {
        Console.WriteLine("Error: falta de comillas.");
        sintaxisCorrecta = false;
    }

    if (countPuntoComa  != 1)
    {
        Console.WriteLine("Error: falta punto y coma.");
        sintaxisCorrecta = false;
    }

    if (sintaxisCorrecta)
    {
        Console.WriteLine("La sintaxis es correcta.");
    }

    return sintaxisCorrecta;
}
```

Se realizan operaciones matemáticas para comprobar que esté bien estructurado el ciclo

```
Contenido del archivo:
i(condicion) { imprimir("hola

No se encontraron ciclos bien estructurados en el archivo.
Error: falta de llaves.
Error: falta de paréntesis.
Error: falta de comillas.
Error: falta punto y coma.
El código tiene errores de sintaxis.
```

Si el programa detecta mal estructuración en el archivo, imprimirán mensajes de error