**UNIVERSIDAD TECNOLOGICA DE SANTIAGO, (UTESA)**

**SISTEMA CORPORATIVO**

**Carrera de Ingeniería en Sistema y Computación**

Logotipo

Descripción generada automáticamente

**PROGRAMACION DE VIDEO JUEGOS**

**TRABAJO A REALIZAR**

Capítulo 1.

**DOCENTE**

Ing. Iván Mendoza

**PRESENTADO POR:**

Alexander Ventura Mercado 2-17-0038

Amaury Francisco Santos Rosario 2-17-1379

República Dominicana, Santiago de los caballeros,

20 de agosto, 2021.

**1.1 Descripción**

El ajedrez es un juego que se juega entre dos oponentes en lados opuestos de un tablero que contiene 64 casillas de colores alternos. Cada jugador tiene 16 piezas: 1 rey, 1 reina, 2 torres, 2 alfiles, 2 caballos y 8 peones. El objetivo del juego es dar jaque mate al otro rey. El jaque mate ocurre cuando el rey está en posición de ser capturado (en jaque) y no puede escapar de la captura

**1.2 Motivación**

Lo que me motivo a crear este juego es que desde siempre me han gustado los juegos de mesa y el ajedrez es uno de mis favoritos, duré mucho tiempo jugando ajedrez en línea hasta que comencé a jugar el género Batlle royale, aunque actualmente ya no juego ajedrez, todavía sigue siendo uno de mis juegos favoritos.

**1.2.1 Originalidad de la idea**

Deep Blue era una computadora para jugar al ajedrez desarrollada por IBM. Fue la primera computadora en ganar tanto una partida de ajedrez como una partida de ajedrez contra un campeón mundial.

**1.2.2 Estado del Arte**

Chess.com comenzó en 2005 cuando dos amigos (Erik Allebest Jay Severson), Decidieron que el mundo necesitaba un sitio web de ajedrez mejor. Se conocieron 10 años antes en la universidad, donde se volvió adicto al ajedrez, y Jay era el presidente del club de ajedrez. Se hicieron amigos gracias a su pasión compartida por el juego.

Había lugares para jugar en línea, pero ninguno se sentía como en su casa. Imaginaron un sitio donde las personas pudieran construir su casa de ajedrez en línea y todo en un solo lugar: para jugar en un ambiente seguro y amigable, encontrar amigos, guardar sus partidas, contar sus historias de ajedrez, compartir ideas y aprender unos de otros.

Tipo de animación: 2D.

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**1.3 Objetivo general**

Poder hacer un juego de ajedrez que funcione igual que un ajedrez real, que cualquier persona no sienta la diferencia.

**1.4 Objetivos específicos**

Poder incluir todas las reglas del juego.

Crear un juego con una interfaz agradable.

Crear un juego estable, fluido y sin errores.

**1.5 Escenario**

Código QR

Descripción generada automáticamente

**1.6 Contenidos**

Juego de ajedrez donde podrás jugar con la maquina o con tu compañero de piso, 2 personas en un pc, en este juego podrás jugar un 1 vs 1 contras tus colegas.

**1.7 Metodología**

Un juego de ajedrez básico donde cada ficha cuenta con sus característicos movimientos y cada persona tiene un turno para mover una ficha. El jugador que haga jaque al otro gana.

**1.8 Arquitectura de la aplicación**

La arquitectura del juego es una donde dividimos los movimientos de cada pieza y bloqueamos algunos movimientos por ocupación o bloqueo de una pieza aliada, en caso de que este ocupada por una pieza enemiga este la pueda sacar del tablero.

Peones: Primer movimiento de 2 o 1 paso, solo puede sacar piezas enemigas cuando estas están en una de las equinas de su posición actual.

Torre: se puede mover el forma vertical y horizontal de forma recta.

Caballo: solo puede moverse en forma de L dando 3 pasos, es el único que puede faltar a otras piezas.

Afil: se mueve solo por la equinas del color de la posición donde están.

Rey: puede moverse en cualquier dirección, pero solo da un paso.

Reina: se mueve en cualquier dirección y da los pasos que quieras.

**1.9 Herramientas de desarrollo**

Unity para el desarrollo del juego, unión de los Sprite con el entorno para la simulación de los movimientos.

Sprite prediseñados de las piezas que nos facilitan la simulación y creación de los movimientos y diseños.

**2.1 Planificación (Diagrama de Gantt)**

**Gráfico

Descripción generada automáticamente**

**2.2 Diagramas y Casos de Uso**

**Diagrama

Descripción generada automáticamente**

**Diagrama

Descripción generada automáticamente**

**2.3 Plataforma**

**PC**

**2.4 Género**

**Juegos de mesa**

**2.5 Clasificación**

**E: Para todos, mayores de 3 años.**

**2.6 Tipo de Animación**

2D

**2.7 Equipo de Trabajo**

Ingenieros de audio: Amaury Santos y Alexander Ventura Mercado

Diseñadores: Amaury Santos

Ilustradores: Amaury Santos

Programadores: Amaury Santos y Alexander Ventura Mercado

Animadores: Amaury Santos

**2.8 Historia**

Juego virtual del clásico Ajedrez donde se busca eliminar al rey del oponente y ganar la batalla. Este juego se hizo para representar las batallas reales como un juego.

**2.9 Guion**

Sin guión.

**2.10 Storyboard**

Sin Storyboard.

2.11 Personajes

Contrincante que será con la persona o máquina que juguemos.

Jugador, nosotros mismo como persona dentro del juego.

**2.12 Niveles**

Fácil

Normal

Difícil

**2.13 Mecánica del Juego**

Este consiste en mover las piezas por un tablero cuadriculado de 2 colores la cuales constituyen 64 posibles posiciones. Al principio del juego cada jugador cuenta con 16 piezas. El juego consiste en eliminar el rey del oponente antes, cada pieza cuenta con unos movimientos específicos que se utilizan para eliminar las piezas del contrincante para así poder arrinconar al rey y ganar la partida. Existe una posibilidad de que el peón llegue al extremo de contrincante pueda cambiarse por una pieza eliminada

**3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)**

**Menús**

Imagen que contiene objeto, pieza de ajedrez, interior, tabla

Descripción generada automáticamente

Imagen que contiene objeto, interior, pieza de ajedrez, tabla

Descripción generada automáticamente

**Sprites.**





Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

**Juego.**

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

**Scripts**

**Menú Scripts.**

public class play : MonoBehaviour

{

    void Start()

    {

    }

    void Update()

    {

    }

    public void empezarjuego()

    {

        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

    }

    public void CerrarJuego (){

        Application.Quit();

        Debug.Log("Salir");

    }

}

**Chessman Script.**

public class Chessman : MonoBehaviour

{

    public GameObject controller;

    public GameObject movePlate;

    private int xBoard = -1;

    private int yBoard = -1;

    private string player;

    public Sprite black\_queen, black\_knight, black\_bishop, black\_king, black\_rook, black\_pawn;

    public Sprite white\_queen, white\_knight, white\_bishop, white\_king, white\_rook, white\_pawn;

    public void Activate()

    {

        controller = GameObject.FindGameObjectWithTag("GameController");

        SetCoords();

        switch (this.name)

        {

            case "black\_queen": this.GetComponent<SpriteRenderer>().sprite = black\_queen; player = "black"; break;

            case "black\_knight": this.GetComponent<SpriteRenderer>().sprite = black\_knight; player = "black"; break;

            case "black\_bishop": this.GetComponent<SpriteRenderer>().sprite = black\_bishop; player = "black"; break;

            case "black\_king": this.GetComponent<SpriteRenderer>().sprite = black\_king; player = "black"; break;

            case "black\_rook": this.GetComponent<SpriteRenderer>().sprite = black\_rook; player = "black"; break;

            case "black\_pawn": this.GetComponent<SpriteRenderer>().sprite = black\_pawn; player = "black"; break;

            case "white\_queen": this.GetComponent<SpriteRenderer>().sprite = white\_queen; player = "white"; break;

            case "white\_knight": this.GetComponent<SpriteRenderer>().sprite = white\_knight; player = "white"; break;

            case "white\_bishop": this.GetComponent<SpriteRenderer>().sprite = white\_bishop; player = "white"; break;

            case "white\_king": this.GetComponent<SpriteRenderer>().sprite = white\_king; player = "white"; break;

            case "white\_rook": this.GetComponent<SpriteRenderer>().sprite = white\_rook; player = "white"; break;

            case "white\_pawn": this.GetComponent<SpriteRenderer>().sprite = white\_pawn; player = "white"; break;

        }

    }

    public void SetCoords()

    {

        float x = xBoard;

        float y = yBoard;

        x \*= 0.66f;

        y \*= 0.66f;

        x += -2.3f;

        y += -2.3f;

        this.transform.position = new Vector3(x, y, -1.0f);

    }

    public int GetXBoard()

    {

        return xBoard;

    }

    public int GetYBoard()

    {

        return yBoard;

    }

    public void SetXBoard(int x)

    {

        xBoard = x;

    }

    public void SetYBoard(int y)

    {

        yBoard = y;

    }

    private void OnMouseUp()

    {

        if (!controller.GetComponent<Game>().IsGameOver() && controller.GetComponent<Game>().GetCurrentPlayer() == player)

        {

            DestroyMovePlates();

            InitiateMovePlates();

        }

    }

    public void DestroyMovePlates()

    {

        GameObject[] movePlates = GameObject.FindGameObjectsWithTag("MovePlate");

        for (int i = 0; i < movePlates.Length; i++)

        {

        }

    }

    public void InitiateMovePlates()

    {

        switch (this.name)

        {

            case "black\_queen":

            case "white\_queen":

                LineMovePlate(1, 0);

                LineMovePlate(0, 1);

                LineMovePlate(1, 1);

                LineMovePlate(-1, 0);

                LineMovePlate(0, -1);

                LineMovePlate(-1, -1);

                LineMovePlate(-1, 1);

                LineMovePlate(1, -1);

                break;

            case "black\_knight":

            case "white\_knight":

                LMovePlate();

                break;

            case "black\_bishop":

            case "white\_bishop":

                LineMovePlate(1, 1);

                LineMovePlate(1, -1);

                LineMovePlate(-1, 1);

                LineMovePlate(-1, -1);

                break;

            case "black\_king":

            case "white\_king":

                SurroundMovePlate();

                break;

            case "black\_rook":

            case "white\_rook":

                LineMovePlate(1, 0);

                LineMovePlate(0, 1);

                LineMovePlate(-1, 0);

                LineMovePlate(0, -1);

                break;

            case "black\_pawn":

                PawnMovePlate(xBoard, yBoard - 1);

                break;

            case "white\_pawn":

                PawnMovePlate(xBoard, yBoard + 1);

                break;

        }

    }

    public void LineMovePlate(int xIncrement, int yIncrement)

    {

        Game sc = controller.GetComponent<Game>();

        int x = xBoard + xIncrement;

        int y = yBoard + yIncrement;

        while (sc.PositionOnBoard(x, y) && sc.GetPosition(x, y) == null)

        {

            MovePlateSpawn(x, y);

            x += xIncrement;

            y += yIncrement;

        }

        if (sc.PositionOnBoard(x, y) && sc.GetPosition(x, y).GetComponent<Chessman>().player != player)

        {

            MovePlateAttackSpawn(x, y);

        }

    }

    public void LMovePlate()

    {

        PointMovePlate(xBoard + 1, yBoard + 2);

        PointMovePlate(xBoard - 1, yBoard + 2);

        PointMovePlate(xBoard + 2, yBoard + 1);

        PointMovePlate(xBoard + 2, yBoard - 1);

        PointMovePlate(xBoard + 1, yBoard - 2);

        PointMovePlate(xBoard - 1, yBoard - 2);

        PointMovePlate(xBoard - 2, yBoard + 1);

        PointMovePlate(xBoard - 2, yBoard - 1);

    }

    public void SurroundMovePlate()

    {

        PointMovePlate(xBoard, yBoard + 1);

        PointMovePlate(xBoard, yBoard - 1);

        PointMovePlate(xBoard - 1, yBoard + 0);

        PointMovePlate(xBoard - 1, yBoard - 1);

        PointMovePlate(xBoard - 1, yBoard + 1);

        PointMovePlate(xBoard + 1, yBoard + 0);

        PointMovePlate(xBoard + 1, yBoard - 1);

        PointMovePlate(xBoard + 1, yBoard + 1);

    }

    public void PointMovePlate(int x, int y)

    {

        Game sc = controller.GetComponent<Game>();

        if (sc.PositionOnBoard(x, y))

        {

            GameObject cp = sc.GetPosition(x, y);

            if (cp == null)

            {

                MovePlateSpawn(x, y);

            }

            else if (cp.GetComponent<Chessman>().player != player)

            {

                MovePlateAttackSpawn(x, y);

            }

        }

    }

    public void PawnMovePlate(int x, int y)

    {

        Game sc = controller.GetComponent<Game>();

        if (sc.PositionOnBoard(x, y))

        {

            if (sc.GetPosition(x, y) == null)

            {

                MovePlateSpawn(x, y);

            }

            if (sc.PositionOnBoard(x + 1, y) && sc.GetPosition(x + 1, y) != null && sc.GetPosition(x + 1, y).GetComponent<Chessman>().player != player)

            {

                MovePlateAttackSpawn(x + 1, y);

            }

            if (sc.PositionOnBoard(x - 1, y) && sc.GetPosition(x - 1, y) != null && sc.GetPosition(x - 1, y).GetComponent<Chessman>().player != player)

            {

                MovePlateAttackSpawn(x - 1, y);

            }

        }

    }

    public void MovePlateSpawn(int matrixX, int matrixY)

    {

        float x = matrixX;

        float y = matrixY;

        x \*= 0.66f;

        y \*= 0.66f;

        x += -2.3f;

        y += -2.3f;

        GameObject mp = Instantiate(movePlate, new Vector3(x, y, -3.0f), Quaternion.identity);

        MovePlate mpScript = mp.GetComponent<MovePlate>();

        mpScript.SetReference(gameObject);

        mpScript.SetCoords(matrixX, matrixY);

    }

    public void MovePlateAttackSpawn(int matrixX, int matrixY)

    {

        float x = matrixX;

        float y = matrixY;

        x \*= 0.66f;

        y \*= 0.66f;

        x += -2.3f;

        y += -2.3f;

        GameObject mp = Instantiate(movePlate, new Vector3(x, y, -3.0f), Quaternion.identity);

        MovePlate mpScript = mp.GetComponent<MovePlate>();

        mpScript.attack = true;

        mpScript.SetReference(gameObject);

        mpScript.SetCoords(matrixX, matrixY);

    }

}

**Game Scripts.**

   public GameObject piezas;

    private GameObject[,] positions = new GameObject[8, 8];

    private GameObject[] playerBlack = new GameObject[16];

    private GameObject[] playerWhite = new GameObject[16];

    private string currentPlayer = "white";

    private bool gameOver = false;

    public void Start()

    {

        playerWhite = new GameObject[] { Create("white\_rook", 0, 0), Create("white\_knight", 1, 0),

            Create("white\_bishop", 2, 0), Create("white\_queen", 3, 0), Create("white\_king", 4, 0),

            Create("white\_bishop", 5, 0), Create("white\_knight", 6, 0), Create("white\_rook", 7, 0),

            Create("white\_pawn", 0, 1), Create("white\_pawn", 1, 1), Create("white\_pawn", 2, 1),

            Create("white\_pawn", 3, 1), Create("white\_pawn", 4, 1), Create("white\_pawn", 5, 1),

            Create("white\_pawn", 6, 1), Create("white\_pawn", 7, 1) };

        playerBlack = new GameObject[] { Create("black\_rook", 0, 7), Create("black\_knight",1,7),

            Create("black\_bishop",2,7), Create("black\_queen",3,7), Create("black\_king",4,7),

            Create("black\_bishop",5,7), Create("black\_knight",6,7), Create("black\_rook",7,7),

            Create("black\_pawn", 0, 6), Create("black\_pawn", 1, 6), Create("black\_pawn", 2, 6),

            Create("black\_pawn", 3, 6), Create("black\_pawn", 4, 6), Create("black\_pawn", 5, 6),

            Create("black\_pawn", 6, 6), Create("black\_pawn", 7, 6) };

        for (int i = 0; i < playerBlack.Length; i++)

        {

            SetPosition(playerBlack[i]);

            SetPosition(playerWhite[i]);

        }

    }

    public GameObject Create(string name, int x, int y)

    {

        GameObject obj = Instantiate(piezas, new Vector3(0, 0, -1), Quaternion.identity);

        Chessman cm = obj.GetComponent<Chessman>();

        cm.name = name;

        cm.SetXBoard(x);

        cm.SetYBoard(y);

        cm.Activate();

        return obj;

    }

    public void SetPosition(GameObject obj)

    {

        Chessman cm = obj.GetComponent<Chessman>();

        positions[cm.GetXBoard(), cm.GetYBoard()] = obj;

    }

    public void SetPositionEmpty(int x, int y)

    {

        positions[x, y] = null;

    }

    public GameObject GetPosition(int x, int y)

    {

        return positions[x, y];

    }

    public bool PositionOnBoard(int x, int y)

    {

        if (x < 0 || y < 0 || x >= positions.GetLength(0) || y >= positions.GetLength(1)) return false;

        return true;

    }

    public string GetCurrentPlayer()

    {

        return currentPlayer;

    }

    public bool IsGameOver()

    {

        return gameOver;

    }

    public void NextTurn()

    {

        if (currentPlayer == "white")

        {

            currentPlayer = "black";

        }

        else

        {

            currentPlayer = "white";

        }

    }

    public void Update()

    {

        if (gameOver == true && Input.GetMouseButtonDown(0))

        {

            gameOver = false;

            SceneManager.LoadScene("Game");

        }

    }

    public void Winner(string playerWinner)

    {

        gameOver = true;

        GameObject.FindGameObjectWithTag("WinnerText").GetComponent<Text>().enabled = true;

        GameObject.FindGameObjectWithTag("WinnerText").GetComponent<Text>().text = playerWinner + " es el ganador";

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

    }

}

**MovePlate Scripts.**

public class MovePlate : MonoBehaviour

{

    public GameObject controller;

    GameObject reference = null;

    int matrixX;

    int matrixY;

    public bool attack = false;

    public void Start()

    {

        if (attack)

        {

            gameObject.GetComponent<SpriteRenderer>().color = new Color(1.0f, 0.0f, 0.0f, 1.0f);

        }

    }

    public void OnMouseUp()

    {

        controller = GameObject.FindGameObjectWithTag("GameController");

        if (attack)

        {

            GameObject cp = controller.GetComponent<Game>().GetPosition(matrixX, matrixY);

            if (cp.name == "white\_king") controller.GetComponent<Game>().Winner("black");

            if (cp.name == "black\_king") controller.GetComponent<Game>().Winner("white");

            Destroy(cp);

        }

        controller.GetComponent<Game>().SetPositionEmpty(reference.GetComponent<Chessman>().GetXBoard(),

            reference.GetComponent<Chessman>().GetYBoard());

        reference.GetComponent<Chessman>().SetXBoard(matrixX);

        reference.GetComponent<Chessman>().SetYBoard(matrixY);

        reference.GetComponent<Chessman>().SetCoords();

        controller.GetComponent<Game>().SetPosition(reference);

        controller.GetComponent<Game>().NextTurn();

        reference.GetComponent<Chessman>().DestroyMovePlates();

    }

    public void SetCoords(int x, int y)

    {

        matrixX = x;

        matrixY = y;

    }

    public void SetReference(GameObject obj)

    {

        reference = obj;

    }

    public GameObject GetReference()

    {

        return reference;

    }

}

**3.2 Prototipos**

Alpha, en esta parte el juego solo consistía en un tablero donde las fichas se podían mover de forma aleatoria y al final del juego no se reiniciaba la partida.

Beta en esta versión el juego cuenta con un menú para bajar y subir el volumen, cambiar el brillo y entrar al modo del juego contra amigo.

**3.3 Perfiles de Usuarios**

El perfil de los usuarios será personas jóvenes adulta de 8 en adelante, con pasión por los juegos de mesa y algo competitivo.

**3.4 Usabilidad**

El juego es fácil de comprender, solo se le dará un click en las piezas que se desea mover y esta presentara los movimientos que puede hacer, en caso de que este bloqueada esta cortara el camino a recorrer. Antes de comenzar el juego podrás elegir entre jugar con otro jugador en el mismo pc u contra la maquina en diferentes modos.

**3.5 Test**

Se probo el juego con 2 personas, las cuales encontraron el juego entretenido, pero no vieron nada único o especial para preferir este modelo en vez de otro, por lo que se pensó en la idea de incluir un sistema de punto para crear un poco de rivalidad, además de la idea de crear pieza única para que los jugadores puedan cambiar en el juego.

**3.6 Versiones de la Aplicación**

Alfa es la prueba donde los desarrolladores probaremos los posibles fallos y solucionaremos. Esta prueba es más específica y buscan errores comunes.

Beta es la prueba donde podremos a los terceros a jugar y ver los posibles fallos para encontrarle solucione. Esta prueba busca encontrar posibles fallos que pasaron por alto en el momento de desarrollarse.

**4.1 Requisitos de la instalación**

Conexión a internet.

Chrome ultima actualización.

**4.2 Instrucciones de Uso**

Para poder mover cada pieza solo hay que darle click con el mouse sobre la pieza que desees mover y marcara la posiciones que tienes permitido mover.

**4.3 Bugs**

Hay veces que el juego se reinicia solo, estos es un error muy problemático ya que en una partida avanzada es muy malo que comience desde cero.

Cuando un peón llega al otro extremo del tablero puede decidir convertirse en cualquier ficha, pero al parecer se queda hay en el medio del otro extremo estorbando.

**4.4 Proyección a Futuro**

Nuestra proyección es crear un juego que suponga un reto a los más grandes jugadores contra una inteligencia artificial, además de presentar una forma fácil para los nuevos jugadores de aprender a jugar ajedrez y comprender sus estrategias, también con contar con un rating para crear una competitividad sana entre los jugadores.

**4.5 Presupuesto**

Tomando en cuenta que los asset de los juegos de ajedrez no son muy complicado y que está desarrollado por uno mismo el presupuesto no es muy alto.

DC: Diseñador Creativo 5 dolaré.

P: Programador 100 dólares.

UT: Usuario Tester 20 dólares

TS: Tecnico de Sonido Entre otros 5 dólares

Total: 130 dólares.

**4.6 Análisis de Mercado**

Los juegos de ajedrez multijugador están volviendo a ser muy populares gracias a las nuevas competiciones, cada día millones de persona juegan partida para demostrar que son los mejores, esto ha creado una comunidad muy competitiva donde quieren mostrar sus habilidades. Esto da una oportunidad a crear un juego con sistema de puntuación donde las personas puedan lucir su conocimiento sobre este juego.

**4.7 Viabilidad**

El juego tiene una gran viabilidad gracia a que su desarrollo no es nada costoso, esto ayuda a que se pueda distribuir con un costo mínimo, además con la nueva demanda de juegos de mesa para demostrar habilidades hace que se muy buscado por personas nuevas y con gran experiencia. El juego será totalmente gratis y podrán acceder a él por una página web, los ingresos será por promociones y nuevos diseños de las fichas del tablero. Las personas cada día buscan nuevos retos y el hecho de poder enfrentarse a más personas del otro lado del mundo hace muy atractivo estos tipos de juegos.

**CONCLUSIONES**

Los juegos de ajedrez están volviendo a ser muy jugado y son varios los desarrolladores que han creado nuevo juegos virtuales sobre esto, pero casi nadie ha desarrollado un IA capaz de poner un reto a los grandes jugadores, aunque el diseño de este juego es algo simple la inteligencia que se tiene que aplicar para que la maquina suponga un reto para los usuarios es más complicado de lo que aparece, también el hecho de que cuando descubre cómo funciona la maquina es fácil derrotarla por lo que la necesidad de que la maquina pueda aprender de su errores para suponer un reto es necesario.

**REFERENCIAS**

Archivo.

<https://github.com/amaurysantos27/proyectofinall>

<https://github.com/VM-Alexander/Proyecto_Video_Juego>

Ejecutable

<https://amaurysantos27.github.io/proyectoFinalWeb/>