INSTITUTO TECNOLÓGICO DE IZTAPALAPA

Sistemas Computacionales

Entrega de proyecto final

PRESENTAN:

ANTONIO ORTEGA BUSTAMANTE 171080148 **25%**

MEJIA RAMOS LUIS ENRIQUE 161080183 **25%**

FABIOLA GUERRERO MENDEZ 16106147 **25%**

FABIAN HERNANDEZ AVILES 16106148 **25%**

TEMA:

JUEGO DE “PIEDRA PAPEL O TIJERA” IMPLEMENTADO EN DEV-C++

LUGAR DE LA REALIZACIÓN

INSTITUTO TECNOLOGICO IZTAPALAPA I

ASESOR INTERNO

ABIEL TOMAS HERNANDEZ PARRA

CMDX JUNIO/2021

Índice

RESUMEN

El presente trabajo es sobre recrear un juego de piedra Papel o tijera en Python, para poder crear este simulador se requiere de una librería que vamos a estar importando durante el proceso llamada random, este simulador lo podremos jugar solamente en nuestra computadora, en donde por una parte el usuario dentro de cmd podrá ejecutarlo y comenzar a jugar por lo tanto si el usuario coloca papel saldrá un mensaje que la PC también selecciono papel y quedaron empates y así sucesivamente el programa arrojar datos de una forma constructiva y entretenida.

.

Introducción

Durante la realización de este proyecto se lleva a cabo la programacion de un clásico de los juegos de destreza y suerte en la que se deriva en el muy conocido Piedra, Papel y Tijera, la programacion se llevo a cabo en la plataforma de PYTHON en la cual es un software libre en el que podemos programar cualquier software o aplicación como lo que es este proyecto.

La elaboración de este proyecto es la interacción de la programacion de una IA con la gente en un entorno en donde la destreza del usuario se desarrolla en competencia con la IA en ganar la partida del juego.

EL mundo de la IA ha evolucionado a una complejidad en la que depende cual sea la situación en la que se usara las IA en el caso de construcciones se desarrollan IA para simuladores, o en las áreas de aprendizajes los chatbot para ayudar a mejorar el habla como en el caso de inglés o autocompletar las frases solicitadas o ingresadas por el usuario. Por ello, la interacción de IA con el mundo se mejora cada día desde videojuegos hasta los smartphones que llevamos hoy en día que nos facilitan la interacción y la comunicación con el mundo en el exterior cada día de nuestras vidas

Objetivo General:

El objetivo es elaborar el clásico juego de piedra, papel o tijera por medio de la plataforma DEV C++ con el fin de completar y entregar el juego como proyecto final para la materia de inteligencia artificial

Objetivos Específicos:

- Comprender la programacion requerida para la elaboración del juego
- Estudiar el comportamiento de la IA al momento de jugar y de programarla
- Utilizar herramientas o elementos necesarios para su programacion en DEV-C++
- Divertirse jugando con la inteligencia artificial e intentar ganarle en repetidas ocasiones

.

Justificación

Para el desarrollo de este videojuego de destreza que se deriva en ganar a una IA programada, se trata del clásico juego de piedra, papel y tijera, pequeño juego que a diferencia de la mayoría de juegos conocidos este se lleva a cabo con la competición de la IA, debido a que la IA se programado con la dificultad de que le ayude usuario interactuar y buscar una manera de lograr ganarle la jugada a la IA con la que compite.

Desarrollando así mentalmente una manera en la que el usuario se divierta y busque analizar la siguiente jugada a escoger para ganar al usuario. La programacion se lleva a cabo en DEV C en que la su programacion es más lógica, de esta manera le da una pequeña ventaja a la IA de ganarle al jugador adivinado su posible movimiento a escoger en la ejecución del juego.

La IA es el factor determinante de algunas historias de éxito significativas:
Una de los éxitos en Ia programada en el área de videojuegos se debe a la exitosa compañía **VALVE** creadora de un sin número de videojuegos las cuales entre ellos destacaron la mayor creación por parte de la empresa, fue el videojuego de survival horror **LEFT 4 DEAD,** la cual se descato por proponer una IA para los enemigos durante la partida.
Se destaca por llevar al jugador a experimentar una partida inigualable en cual fuera su dificultad a la hora de jugar, el juego obligaba al jugador a tomar medidas necesarias para lograr sobrevivir a las hordas de infectados dentro del juego Así se demuestra que las IA pueden ser programadas para darle al jugador una experiencia al jugar o competir en este tipo de juegos de destreza para obtener la victoria.

A continuación, se presentará una tabla definiendo las ventajas y desventajas de la implementacion de una IA en los videojuegos definiendo los puntos importantes en esta implementacion

Ventajas	Desventajas
- Ayudará al usuario a decidir que estrategia usar para ganar la partida - Conocerá la habilidad que pueda alcanzar un a IA programada meticulosamente - La aplicación de una IA interactuando con la gente será un nuevo paso para logar un mayor control sobre el tema - Ser un buen comienzo de en la manera de practica para la introducción al mundo de la programacion de una IA	- Es un proyecto sencillo - Usuarios capacitados no le encontrarían interesante un juego muy viejo y fácil - Su programacion puede ser fácilmente manipulada - En el mercado sería poco probable que no alcance grandes ventas

Tabla 1. Ventajas y desventajas

CAPITULO 1

"MARCO TEORICO"

1.1 Tecnologías Computacionales que se implementan

Lenguajes de programación;

DEV C++

Framework de desarrollo;

Bottle

Plataforma:

Multiplataforma: Unix, Linux, macOS y Windows.

Escritorio, web, móvil; Escritorio

Algoritmo de I.A. a implementar en su proyecto:

Algoritmos de búsqueda local

1.2 El algoritmo nos permite:

de una solución y va realizando modificaciones locales Inicio GENERA (Solución Inicial), Solución Actual ← Solución Inicial; Mejor Solución ← Solución Actual; Repetir GENERA (Solución Actual); %Generación Aleatoria

Si Objetivo (Solución Actual) es mejor que Objetivo (Mejor Solución)

entonces Mejor Solución ← Solución Actual; Hasta (Criterio de parada); DEVOLVER (Mejor Solución); Fin

En esta sección pretendemos estudiar el comportamiento de la búsqueda aleatoria, realizando un estudio de su eficacia y eficiencia, el cual justificará el uso de los procedimientos de búsqueda local.

1.3 Tecnologías computacionales que se implementaron en nuestro proyecto.

1.3.1 Lenguaje de programación.

Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas.

1.3.2 DEV C.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

1.3.3 Framework.

En el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.3.4 Multiplataforma.

En informática, se denomina multiplataforma a un atributo conferido a programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas informáticas.

El software multiplataforma puede dividirse en dos grandes tipos o clases:

Uno requiere una compilación individual para cada plataforma que le da soporte, y el otro se puede ejecutar directamente en cualquier plataforma, sin preparación especial.

1.4 Para el desarrollo del programa en DEV C.

1.4.1 Condiciones.

Las sentencias condicionales son estructuras de código que indican que, para que cierta parte del programa se ejecute, deben cumplirse ciertas premisas.

Estos condicionantes por lo general solo se ejecutan una vez a lo largo del programa.

- If: Indica una condición para que se ejecute una parte del programa.
- Else if: Siempre va precedido de un "If" e indica una condición para que se ejecute una parte del programa siempre que no cumpla la condición del if previo y sí se cumpla con la que el "else if" especifique.
- Else: Siempre precedido de "If" y en ocasiones de "Else If". Indica que debe ejecutarse cuando no se cumplan las condiciones previas.

1.4.2 Bucles.

Los bucles son parientes cercanos de los condicionantes, pero ejecutan constantemente un código mientras se cumpla una determinada condición. Los más frecuentes son:

- For: Ejecuta un código mientras una variable se encuentre entre 2 determinados parámetros.
- While: Ejecuta un código mientras que se cumpla la condición que solicita.

1.4.2 Funciones.

Las funciones se crearon para evitar tener que repetir constantemente fragmentos de código. Una función podría considerarse como una variable que encierra código dentro de si. Por lo tanto, cuando accedemos a dicha variable (la función) en realidad lo que estamos haciendo es ordenar al programa que ejecute un determinado código predefinido anteriormente.

1.4.3 Librerías.

En programación, una librería es un archivo o conjunto de archivos que se utilizan para facilitar la programación. Las librerías, también llamadas "framework", consisten en archivos de código a los que llamamos al principio de la página.

1.4.4 Listas.

Una lista es una estructura dinámica de datos que contiene una colección de elementos homogéneos (del mismo tipo) de manera que se establece entre ellos un orden. Es decir, cada elemento, menos el primero, tiene un predecesor, y cada elemento, menos el último, tiene un sucesor.

1.5 ¿Qué es la inteligencia artificial (IA)?

En términos sencillos, inteligencia artificial (IA) se refiere a los sistemas o las máquinas que imitan la inteligencia humana para realizar tareas y que tienen la capacidad de mejorar iterativamente a partir de la información que recopilan. La IA se manifiesta de varias formas. Algunos ejemplos son:

Los bots conversacionales que utilizan IA para comprender más rápido los problemas de los clientes y proporcionar respuestas más eficientes
Los asistentes inteligentes utilizan la IA para analizar información crítica proveniente de grandes conjuntos de datos de texto libre para mejorar la programación Los motores de recomendación pueden proporcionar recomendaciones automatizadas para programas de TV según los hábitos de visualización de los usuarios
La IA se trata mucho más sobre el proceso y la capacidad de pensamiento superpoderado y el análisis de datos que sobre cualquier formato o función en particular. Aunque la IA muestra imágenes de robots de aspecto humano de alto funcionamiento que se apoderan del mundo, la IA no pretende reemplazar a los humanos. Su objetivo es mejorar significativamente las capacidades y contribuciones humanas. Eso la convierte en un activo comercial muy valioso.

1.6 Términos de inteligencia artificial

La IA se ha convertido en un término general para las aplicaciones que realizan tareas complejas que antes requerían aportes humanos, como la comunicación en línea con los clientes o jugar al ajedrez. El término a menudo se usa indistintamente con sus subcampos, que incluyen el aprendizaje autónomo y el aprendizaje profundo. Sin embargo, hay ciertas diferencias. Por ejemplo, el aprendizaje automático se centra en la creación de sistemas que aprenden o mejoran su rendimiento en función de los datos que consumen. Es importante tener en cuenta que aunque todo aprendizaje automático es IA, no toda IA es aprendizaje automático.

Para obtener el valor completo de la IA, muchas empresas están haciendo inversiones significativas en equipos de ciencia de datos. La ciencia de datos, un campo interdisciplinario que usa métodos científicos y de otro tipo para extraer valor de los datos, combina conocimientos de campos como la

estadística y la informática con el conocimiento empresarial para analizar los datos que se recopilan de múltiples fuentes.

Cómo la IA puede ayudar a las organizaciones

El principio fundamental de la IA es replicar, y luego superar, la forma en que los seres humanos perciben y reaccionan ante el mundo. Se está convirtiendo rápidamente en el cimiento principal de la innovación. La IA, impulsada por varias formas de aprendizaje automático que reconocen patrones en los datos para permitir predicciones, puede agregar valor a su negocio ya que permite:

- Proporcionar una comprensión más completa de la abundancia de datos disponibles
- Usar predicciones para automatizar las tareas excesivamente complejas o prosaicas

1.6.1 Cinco mitos comunes sobre la IA empresarial

Si bien muchas empresas han adoptado con éxito la tecnología de IA, también hay mucha información errónea sobre la inteligencia artificial y lo que puede y no puede hacer. Aquí, exploramos cinco mitos comunes sobre la IA:

Mito 1: La IA empresarial requiere un enfoque de "créelo usted mismo".
Realidad: La mayoría de las empresas adoptan IA combinando soluciones internas y listas para usar. El desarrollo interno de IA les permite a las empresas personalizar según sus necesidades comerciales únicas. Las soluciones de IA prediseñadas le permiten optimizar su implementación con una solución lista para usar para los problemas comerciales más comunes.
Mito 2: La IA entregará resultados mágicos de inmediato.

Realidad: El camino hacia el éxito en IA requiere tiempo, una planificación cuidadosa y una idea clara de los resultados que desea lograr. Necesita un marco estratégico y un enfoque iterativo para evitar entregar un conjunto aleatorio de soluciones de IA desconectadas.

Mito 3: La IA empresarial no requiere que las personas la ejecuten.

Realidad: La IA empresarial no se trata de que los robots se hagan cargo. El valor de la IA es que aumenta las capacidades humanas y libera a sus empleados para realizar tareas más estratégicas. Además, la IA depende de las personas para que proporcionen los datos correctos y trabajen con ellos de manera adecuada.
Mito 4: Cuantos más datos, mejor.
Realidad: La IA empresarial necesita datos inteligentes. Para obtener la información empresarial más efectiva a partir de la IA, sus datos deben ser de alta calidad, estar actualizados y enriquecidos, y tener relevancia.

Mito 5: La IA empresarial solo necesita datos y modelos para tener éxito.
Realidad: Los datos, los algoritmos y los modelos son un comienzo, pero una solución de IA debe ser escalable para satisfacer las cambiantes necesidades comerciales. Hasta la fecha, la mayoría de las soluciones de IA para empresas han sido creadas a mano por científicos de datos. Estas soluciones requieren instalación y mantenimiento detallados y manuales, y no permiten escalar. Para implementar con éxito proyectos de IA, necesita soluciones de inteligencia artificial que puedan escalar a fin de cumplir con los nuevos requisitos a medida que avanza con la IA

1.7 Los beneficios y desafíos de poner en práctica la IA

Existen numerosos casos de éxito que demuestran el valor de la IA. Las organizaciones que suman el aprendizaje automático e interacciones cognitivas a las aplicaciones y a los procesos de negocios tradicionales pueden mejorar en mayor medida la experiencia de usuario e impulsar la productividad.

Sin embargo, la base no se encuentra lo suficientemente sólida. Pocas compañías han desplegado la IA a escala por varias razones. Por ejemplo, si no usan la computación en la nube, los proyectos de IA a menudo son costosos a nivel computacional. También son complejos de construir y requieren experiencia que es muy demandada pero escasa en oferta. Saber cuándo y dónde incorporar la IA, así como cuándo recurrir a un tercero, ayudará a minimizar estas dificultades.

1.8 Casos de éxito de la IA

La IA es el factor determinante de algunas historias de éxito significativas:
Una de los éxitos en Ia programada en el área de videojuegos se debe a la exitosa compañía VALVE creadora de un sin numero de videojuegos las cuales entre ellos destacaron la mayor creación por parte de la empresa, fue el videojuego de survival horror **LEFT 4 DEAD,** la cual se descato por proponer una IA para los enemigos durante la partida.
Se destaca por llevar al jugador a experimentar una partida inigualable en cual fuera su dificultad a la hora de jugar, el juego obligaba al jugador a tomar medidas necesarias para lograr sobrevivir a las hordas de infectados dentro del juego, de las siguientes maneras:

- Aumentar el número de infectados y jefes durante la partida
- Posicionar a los enemigos en lugares en donde el jugador se sienta atrapado o dificulte pasar el nivel
- Armas especiales o falta de munición durante la partida

1.9 Los obstáculos que dificultan desarrollar el máximo potencial de la IA

A pesar de las promesas de la IA, muchas empresas no aprovechan todo el potencial del aprendizaje automático y de otras funciones de la IA. ¿Por qué? Irónicamente, resulta que el problema es, en gran parte... las personas. Los flujos de trabajo ineficientes pueden impedir que las empresas obtengan el valor total de sus implementaciones de IA.

Por ejemplo, los científicos de datos pueden enfrentar desafíos para obtener los recursos y datos necesarios para construir modelos de aprendizaje autónomo. Pueden tener problemas para colaborar con sus compañeros de equipo. Además, cuentan con muchas herramientas de código abierto diferentes de administrar, mientras que los desarrolladores de aplicaciones a veces necesitan recodificar por completo los modelos que los científicos de datos desarrollan antes de que puedan integrarlos a sus aplicaciones

CAPITULO 2

“METODOLOGIA”

2.1 METODOLOGÍA MIXTA

En 1950, el matemático John Nash demostró que, en cualquier juego con un número finito de jugadores y un número finito de opciones, existe siempre una combinación de estrategias tal que ningún jugador puede obtener mejores resultados cambiando únicamente su propia táctica. La teoría que describe estos perfiles de estrategias estables, o «equilibrios de Nash», revolucionó el campo de la teoría de juegos, alteró el curso de la economía y cambió la forma en que se estudia y analiza todo, desde los tratados políticos hasta el tráfico en Internet. Y, además, le valió a Nash el premio Nobel de economía en 1994.

2.1 ESTRATEGIAS PURAS Y MIXTAS

Supongamos que B adopta la estrategia —no muy inteligente— de sacar papel todas las veces. Tras ganar, perder y empatar algunas rondas, es probable que A se percate de esta táctica y adopte una contra estrategia ganadora, consistente en sacar siempre tijera. Llamemos a este perfil de estrategias [tijera, papel]. Si en cada ronda se produce este enfrentamiento, el jugador A ganará siempre.

Sin embargo, B no tarda en percatarse de lo disparatado de este perfil de estrategias y, viendo la confianza que deposita A en la tijera, pasa a sacar siempre piedra. Este perfil de estrategias, [tijera, piedra], hace que B comience a ganar. Pero, evidentemente, A cambiará entonces al papel. Durante estos tramos, ambos jugadores están empleando lo que se conoce como estrategias «puras»: una única estrategia que se selecciona y se emplea repetidamente.

Está claro que en este caso no se alcanzará ningún equilibrio: para cualquier estrategia pura (como sacar siempre piedra) se puede adoptar una contraestrategia (sacar siempre papel) que provocará un nuevo cambio de tácticas. Los jugadores continuarán persiguiéndose eternamente el uno al otro alrededor de este círculo de estrategias.

Sin embargo, podemos también intentar usar una estrategia mixta. Supongamos que, en vez de escoger una única opción, en cada ronda podemos elegir aleatoriamente una de las estrategias puras. En lugar de sacar siempre piedra, una estrategia mixta podría consistir en sacar piedra la mitad de las veces y tijera la otra mitad. Nash demostró que, cuando se permite este tipo de combinaciones, todos los juegos de este estilo tienen al menos un punto de equilibrio. Tratemos de encontrarlo.

2.2 ¿QUÉ ESTRATEGIA MIXTA RESULTARÍA APROPIADA EN EL JUEGO DE PIEDRA, PAPEL O TIJERA?

Una intuición razonable sería sacar piedra, papel o tijera con la misma probabilidad, táctica que denotaremos (1/3, 1/3, 1/3). Esto significa que elegimos piedra, papel o tijera con una probabilidad de 1/3. ¿Es ventajoso?

Supongamos que B opta por sacar siempre piedra: una estrategia pura que podemos representar como (1, 0, 0). ¿Cómo se desarrollará el juego con el perfil de estrategias (1/3, 1/3, 1/3) para A y (1, 0, 0) para B?

Para visualizar lo que ocurre, construyamos una tabla que muestre la probabilidad de cada uno de los 9 resultados que pueden producirse en cada ronda: A saca piedra y B saca piedra; A saca piedra y B saca papel; etcétera. En el cuadro que reproducimos aquí, la fila superior indica la elección del jugador B, y la columna de la izquierda, la de A:

A | B	Piedra	Papel	Tijera
Piedra	$\frac{1}{3}$	0	0
Papel	$\frac{1}{3}$	0	0
Tijera	$\frac{1}{3}$	0	0

Figura 1. Modalidad de juego

Cada entrada indica la probabilidad de que se produzca el correspondiente par de elecciones en una ronda cualquiera, la cual no es más que el producto de las probabilidades asociadas a cada jugador. Por ejemplo, la probabilidad de que A elija papel es 1/3; al mismo tiempo, la de que B opte por piedra es 1. Por tanto, la probabilidad de que ocurran ambas viene dada por 1/3 × 1 = 1/3. Por otro lado, la probabilidad de que A saque papel y B tijera es 1/3 × 0 = 0, etcétera.

¿Cuánto se beneficia el jugador A de este perfil de estrategias? Ganará una tercera parte de las veces ([papel, piedra]), perderá otra tercera parte ([tijera, piedra]) y empatará en otro tercio de las ocasiones ([piedra, piedra]). Podemos calcular el número de puntos que obtendrá en promedio el jugador A en cada ronda sumando los productos de cada resultado por la probabilidad correspondiente:

Esto quiere decir que A obtendrá una media de 0 puntos por ronda: ganará, perderá y empatará con la misma probabilidad. En promedio, el número de

victorias se compensará con el de derrotas y los jugadores estarán básicamente abocados a un empate.

Pero, como hemos avanzado, A puede obtener mejores resultados cambiando de estrategia, suponiendo que el oponente no cambie la suya. Si pasa a la estrategia (0, 1, 0), sacar siempre papel, la tabla de probabilidades quedará así.

2.3 EQUILIBRIOS INALCANZABLES

el hecho de que para todos los juegos de este tipo exista un equilibrio es importante por distintos motivos. Uno de ellos es que hay muchas situaciones del mundo real que pueden modelizarse como si fueran juegos. Cada vez que varias personas se ven atrapadas en el tira y afloja que implica contraponer el beneficio personal y el colectivo, como ocurre durante una negociación o en la competencia por recursos compartidos, cada parte considerará distintas estrategias y evaluará los beneficios. La naturaleza ubicua de este modelo matemático explica la gran repercusión del trabajo de Nash.

Otra razón es que un equilibrio de Nash constituye, en cierto sentido, un resultado positivo para todos los jugadores. Cuando se alcanza, nadie puede mejorar cambiando su propia estrategia. Tal vez existan mejores resultados colectivos, los cuales podrían alcanzarse si todos los jugadores actuaran en perfecta cooperación. Pero, si lo único que podemos controlar es lo que hacemos nosotros, terminar en un equilibrio de Nash es lo mejor a lo que podemos aspirar.

Así pues, podríamos esperar que en «juegos» como los que aparecen en los paquetes de incentivos económicos, los sistemas tributarios, los tratados o los diseños de redes se acaben alcanzando equilibrios de Nash, donde todos los participantes, actuando sin más que en su propio interés, terminan contentos y donde los resultados son estables. Pero ¿es razonable suponer que los jugadores llegarán de manera natural a un equilibrio de Nash?

A | B	Piedra	Papel	Tijera
Piedra	0	0	0
Papel	1	0	0
Tijera	0	0	0

Figura 2. Tabla de juego

Capítulo 3

"descripción de actividades"

A continuación, se describirán a detalle las actividades necesarias para la elaboración del juego “PIEDRA, PAPEL O TIJERA” con el fin de poder realizar correctamente su programacion en el software “DEV-C++”.

Herramientas

- Descargar DEV-C++ de la pagina oficial

https://www.google.com/search?q=devc%2B%2B&oq=devc%2B%2B&aqs=chrome..69i57j46i433j46i131i433j0i433l4j0j46i433.2005j0j7&sourceid=chrome&ie=UTF-8#

- Abrir y ejecutar como administrador el ejecutable y realizar los pasos correspondientes para la instalación de DEV-C++
- Una vez que se termine de instalar la aplicación, nos mostrara el icono de la aplicación en nuestro escritorio.
- Abrimos el programa y comenzaremos a programar el juego

Figura 3. Icono de DEV-C++

Programacion

1) Inicializaremos el programa DEV-C++ la cual su interfaz de trabajo es demasiado cómoda a comparación a otras plataformas de programacion libre, una vez abierta la interfaz de trabajo nos aparecerá de la siguiente manera.

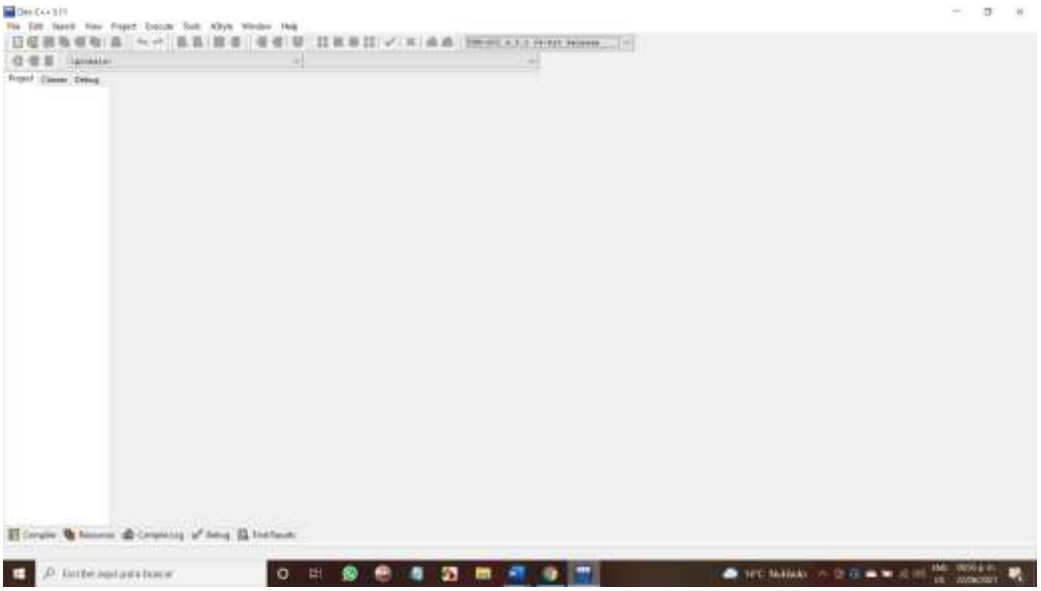

Figura 4. Interfaz de DEV-C++

2) Daremos clic en **file,** seguido en la barra **new y** luego otro clic en **source file**, con esto crearemos un nuevo documento en donde podremos realizar la debida programacion del juego.

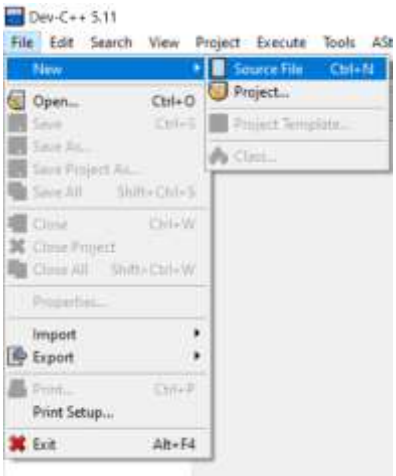

Figura 5. Nuevo documento

3) Comenzamos con escribir las librerías de **#include** las cuáles serán las librerías **<stdio.h>, <conio.h> y <stdlib.h>.** de esta manera las librerías nos apoyaran en que el programa realice las acciones debidas para la programacion, seguido de eso se utilizaran las variables (**jug, maq, op, r),** las cuales serán necesarias para ingresar y guardar los valores ingresados o seleccionados durante el juego.
4) Con ayuda del **int main(void){}** daremos pie a comenzar con el cuerpo del programa, crearemos un menú en el que el jugador podrá escoger las opciones siguientes : **0= piedra, 1= papel, 2= tijeras**.

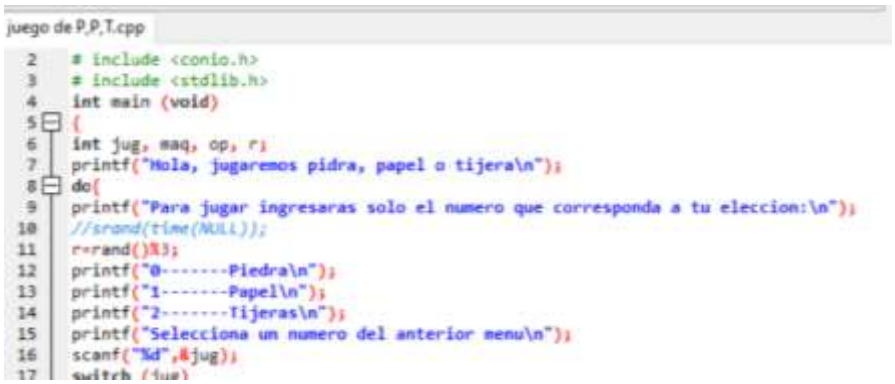

Figura 6. Librerías y menú

5) Con ayuda de las herramientas **do, switch, if y else.** Se creará un menú con el cual según el número u opción que el jugador seleccione encerrando en cada caso las reglas requeridas para el juego por ejemplo si el jugador elije la **opción 0,** en el caso se describirá que si **if(r==0)** se tendrán 3 posibles resultados

- **La computadora elegirá piedra y será un empate**
- **La computadora escogerá papel y ganará la computadora**
- **La computadora escogerá tijeras y el jugador ganará**

Lo mismo ocurrirá para los siguientes casos según cual sea la regla del juego

- **Tijera gana papel**
- **Piedra gana tijera**
- **Papel gana piedra**

Esto incluirá los empates si es que ambos competidores escogen opciones similares **(piedra, piedra), (tijera, tijera), (papel, papel).**

La programacion de los casos de llevara de la siguiente manera:

```
17  switch (jug)
18  {
19  case 0:
20  printf("Elegiste Piedra\n");
21      if (r==0)
22      {
23      printf("\tElegi Piedra\n\n");
24          printf("\t\t\tEmpate\n\n\n");
25      }
26   else
27      {
28       if (r==1)
29       {
30           printf("\tElegi Papel\n\n");
31           printf("\t\t\tGana la computadora\n\n\n");
32       }
33       else
34      {
35       if (r==2)
36       {
37           printf("\tElegi Tijeras\n\n");
38           printf("\t\t\tTu ganas\n\n\n");
39       }
40
41      }
42    }
43  break;
```

Figura 7. Primer caso

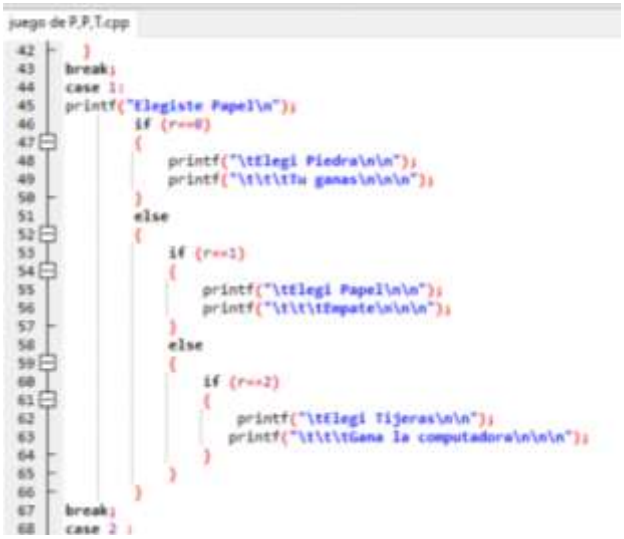

Figura 8. Segundo caso

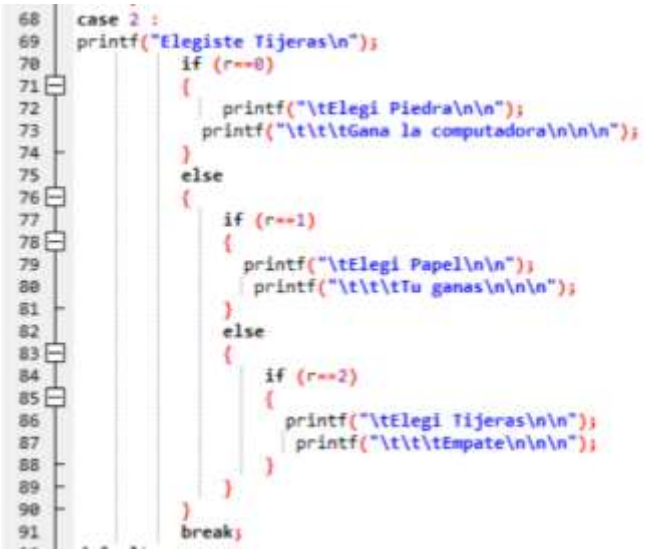

Figura 9. Tercer caso.

Como se puede apreciar en las anteriores imágenes consta de 3 casos comenzando desde el caso 0 hasta el caso 2, cada caso tiene una programacion especial la cual consta con las anteriores reglas anteriormente mencionadas. Cada caso se respeta las reglas en las que puede se puede **ganar, perder o empatar.**

6) Por último, para poder ciclar el juego y jugarlo un sinfín de veces y se vuelva más emotivo se agregó la sentencia de **while** la cual permitirá que el jugador elija entre 1 y 2 con las siguientes opciones:

- **1 repetir y limpiar pantalla**
- **2 salir y finalizar la partida del juego**

```
92  default:
93      printf("Por favor solo introduce los numeros que aparecen en el menu");
94  }
95  printf("Quieres volver a jugar?\n");
96  printf("Si deseas volver a jugar presiona 1\n");
97  printf("Si deseas salir del juego presiona 2\n");
98  scanf("%d", &op);
99  system("cls");
100 }
101 while(op==1);
102 printf("Ok, hasta luego, regresa pronto");
103 getch();
104 }
```

Figura 10 repetir juego.

Resultados

Los resultados fueron los obtenidos, se logro que el juego compile y funcione a la perfección además de agregarle una presentación más técnica y profesional para que el jugador le guste la interfaz presentada para jugar al piedra, papel o tijeras.
Por lo que a continuación se muestran una serie de imágenes las cuales mostrara la jugabilidad de la IA con el jugador al escoger diferentes opciones que el jugador y la maquina pudieron escoger y obtener un ganador, perdedor o un empate.

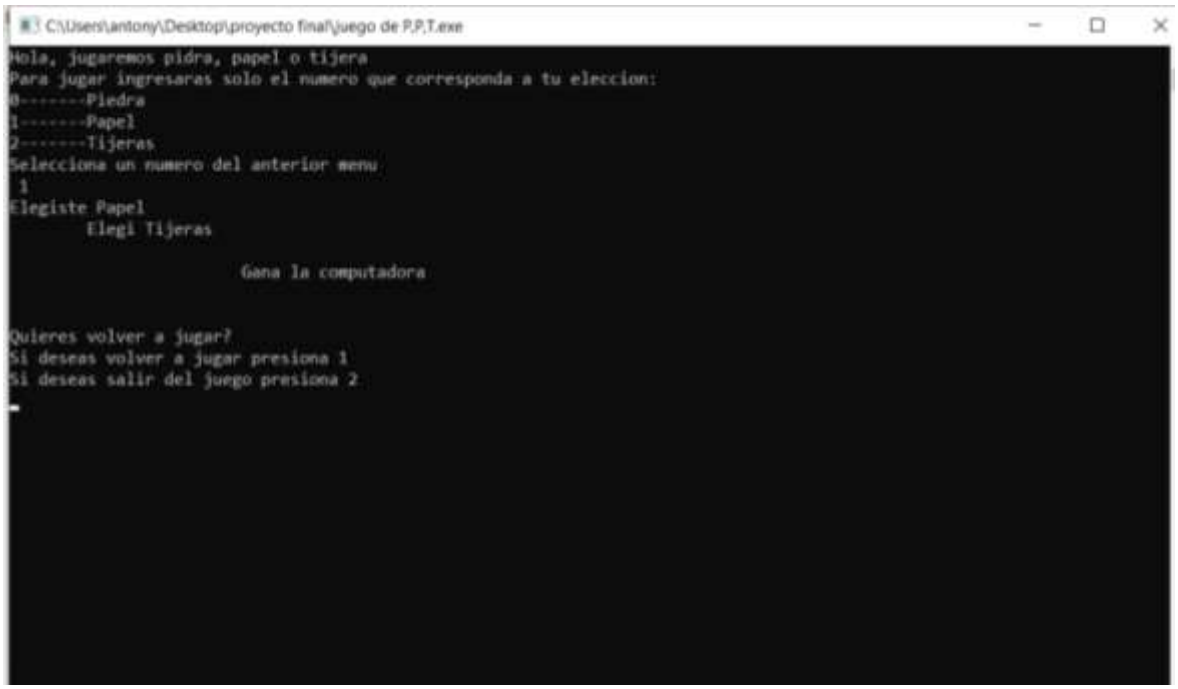

Figura 11. Primera selecion.

Figura 12. Sefunda selecion.

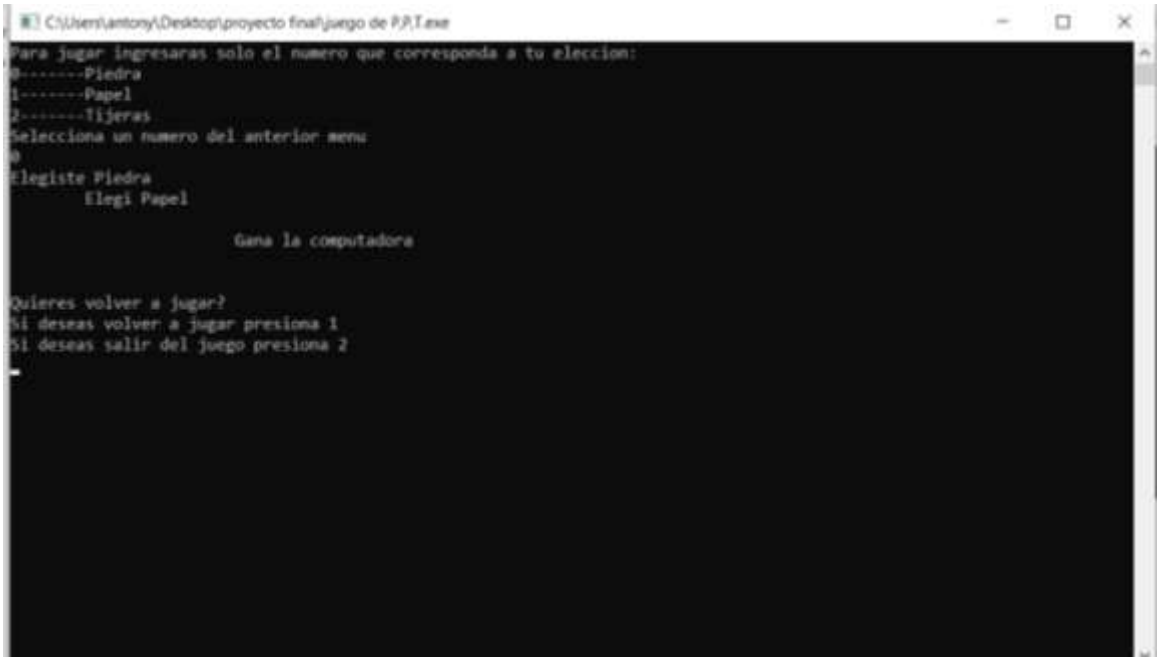

Figura 13. tercer selecion.

CONCLUSIÓN

En el comienzo del proyecto se determinó, elaborar el juego de piedra, papel y tijera por medio de la plataforma, Dev C++ en el cual nos comprometimos a terminar el proyecto entre todos nosotros, para ello se propuso y se desarrollo el juego en dicha plataforma con el fin de que se pueda ejecutar en cualquier computadora, que pueda tener el privilegio de ejecutar la plataforma Dev C++ a lo cual este proyecto pueda ser de gran ayuda para la comunidad y así nosotros podremos aportar un poco de conocimiento para futuras generaciones que quieran entender la Inteligencia Artifician en el lenguaje de programación.

Es un juego de manos en el cual existen tres elementos. La piedra que vence a la tijera rompiéndola, la tijera que vencen al papel cortándolo, y el papel que vence a la piedra envolviéndola; dando lugar a un círculo o ciclo cerrado, que caracteriza al juego.

Bibliografías:

FUENTES DE INFORMACIÓN
P., & Programacionpython80889555, V. T. L. E. (2020, 29 octubre). JUGANDO A “PIEDRA, PAPEL O TIJERA” EN PYTHON. El Programador Chapuzas. https://programacionpython80889555.wordpress.com/2018/03/23/jugando-a-piedra-papel-o-tijera-en-pyton/

Recrear el juego de Piedra Papel o Tijera en Python. (2019, 7 julio). YouTube. https://www.youtube.com/watch?v=gqeTkDKjokw

colaboradores de Wikipedia. (2021, 6 junio). Piedra, papel o tijera. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Piedra,_papel_o_tijera

Machuca, F. (2021, 21 mayo). ¿Qué es Python? El lenguaje de programación más popular para aprender en 2021. https://www.crehana.com. https://www.crehana.com/mx/blog/web/que-es-python/

Eduardo Morales Manzanares (2004-11-02) Búsqueda Local (Local Search) https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/node58.html

https://www.google.com/search?q=devc%2B%2B&oq=devc%2B%2B&aqs=chrome..69i57j46i433j46i131i433j0i433l4j0j46i433.2005j0j7&sourceid=chrome&ie=UTF-8#

ANEXOS

Historia

La primera mención de un juego que se parece a la versión actual de Piedra, papel o tijeras se encuentra en Wuzazu, un libro escrito por el escritor chino Xie Zhaozhi, de la dinastía Ming. El autor se refiere a él como shoushiling y data sus orígenes en algún momento durante la dinastía china Han china (206 aC - 220 dC). Li Rihua, un reconocido artista, crítico y burócrata que vivió durante la dinastía Ming, también habla de shoushiling en su libro Note of Liuyanzhai.

Posteriormente podemos encontrar referencias al juego en Japón, un país que a menudo (erróneamente) se menciona como el lugar de nacimiento del juego. En su libro, Some Thoughts on the Ken Game in Japan: From the Viewpoint of Comparative Civilization Studies (1995), Sepp Linhart confirma los orígenes chinos del juego y su consiguiente popularidad en Japón, algo que trajo consigo ligeros cambios en la forma y las reglas que la mayoría de los jugadores habían seguido hasta entonces.

Figura 14. Imagen antigua

A lo largo de la historia japonesa, los juegos similares a Piedra, papel o tijeras siempre se han denominado sansukumi-ken, un término que se traduce libremente en pasatiempos en los que "tres tienen miedo el uno del otro". En pocas palabras, estos juegos se juegan con tres gestos con las manos ("ken" significa puño). Uno de los primeros sansukumi-ken en ganar popularidad en todo el país fue el mushi-ken, donde los jugadores usan sus manos para representar una rana (pulgar), una babosa (dedo meñique) y una serpiente (dedo índice). La rana gana contra la babosa, que a su vez gana contra la serpiente, mientras que esta última gana contra la rana.

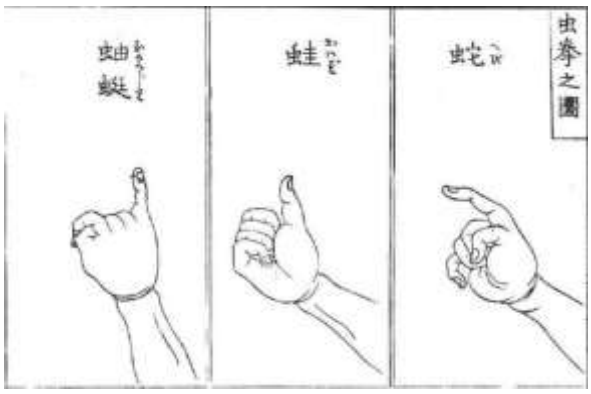

Figura 15 señales de manos

MÁS QUE SUERTE

Aunque es aceptado como un juego de suerte y aleatoriedad, lo cierto es que Piedra, papel o tijera ha sido objeto de muchos estudios de juegos mentales, con expertos que reconocen el uso de al menos alguna estrategia por parte de jugadores profesionales, lo que ha llevado a competiciones mundiales, ligas nacionales y montones de premios en efectivo.

ESTOS SON ESOS 8 GAMBITOS:

- La avalancha (piedra, piedra, piedra)
- El burócrata (papel, papel, papel)
- La caja de herramientas (tijeras, tijeras, tijeras)
- El crescendo (papel, tijeras, piedra)
- El desenlace (piedra, tijeras, papel)
- El puñado de dólares (piedra, papel, papel)
- Las muñecas de papel (tijeras, papel, papel)
- El sandwich de tijeras (papel, tijeras, papel)

Además de estos gambitos, cuando los jugadores profesionales discuten sobre estrategias ganadoras mencionan la importancia de poder predecir el próximo movimiento de un oponente a través de su disposición física, lo que nos lleva al robot Janken.

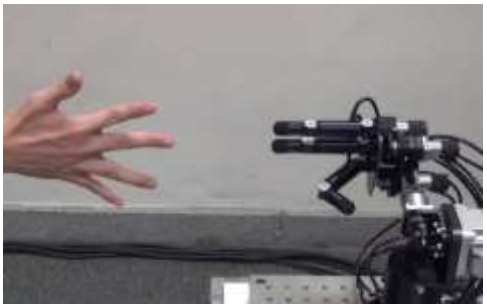

Figura 16. Mano vs maquina