Trabajo de Fin de Grado

Aplicación para el juego del bridge

Grado en Ingeniería Informática

Rubén Javier Lee Ramírez

Supervisado:
Agustin Rafael Trujillo Pino
Gabriele Salvatore De Blasio

Julio 2024

Agradecimientos

Querría dar gracias a los amigos y familiares que me han apoyado hasta ahora.

Agradecer bastante a los tutores de mi proyecto D. Agustín Trujillo Pino y Gabriele Salvatore De Blasio por la inmensa paciencia que han tenido conmigo.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Como se suele decir, el Bridge es como el ajedrez en el reino de los juegos de cartas, un desafío constante que combina estrategia, matemáticas y trabajo en equipo. Este juego de cartas, originado en el siglo XIX, ha evolucionado para convertirse en una disciplina que exige tanto habilidad como inteligencia, manteniéndose relevante a lo largo de los años.

"La diferencia entre el genio y la estupidez en el Bridge es que el genio tiene sus límites" de Jared Johnson [5]. En la era digital, el aprendizaje de juegos complejos como el Bridge puede beneficiarse enormemente de las tecnologías modernas. La creación de aplicaciones de software educativas ofrece una oportunidad única para hacer accesibles las estrategias y reglas del Bridge a una audiencia más amplia, superando las barreras tradicionales de acceso y tiempo.

El objetivo principal de este trabajo de fin de grado es desarrollar una aplicación de software interactiva que facilite el aprendizaje del Bridge, combinando elementos educativos con herramientas tecnológicas avanzadas. Esta aplicación está diseñada para guiar a los usuarios a través de las reglas básicas del juego, estrategias avanzadas, y proporcionar prácticas interactivas que refuercen el aprendizaje.

1.1. El Bridge

El Bridge de contrato es un juego de cartas de cuarto personas en parejas de dos. Se juega con una baraja francesa de cincuenta y dos cartas y tiene como objetivo principal que cada pareja consiga la mayor cantidad de puntos posible mediante la obtención de *bazas* durante las manos.

En el Bridge, los jugadores se sientan alrededor de una mesa en las posiciones correspondientes a las cuatro direcciones cardinales: Norte, Sur, Este y Oeste. Los jugadores opuestos entre sí forman parejas que colaboran durante el juego. Las posiciones son cruciales para la estrategia y la comunicación entre compañeros de equipo a lo largo de la partida.

El juego se divide en dos fases principales: la subasta y la defensa. Durante la subasta, los jugadores apuestan un contrato, es decir, el número de *bazas* que deben ganar y el palo que será el dominante o prioritario. Se asigna un jugador que dará la primera voz y luego en sentido horario a los demás jugadores que deberán declarar si quieren aumentar el contrato, proponer otro contrato o pasar. Si se llega el caso de que tres jugadores no suben la apuesta, mediante el voto de pasar, se considera terminada la subasta, la cual se la lleva el jugador que primero ofreció el palo a ese contrato de la pareja.

Para poder saber la apuesta que puedes hacer hay varios criterios a los que seguir:

- Puntos de honor: las cartas de mayor valor tienen una puntuación en la fase de subastas para poder cuantificar cuanto de buena es tu mano. Siendo:
  - As(A): 4
  - Rey(K): 3
  - Reina(Q): 2
  - Basto(J): 1
- Puntos de distribución: Cuando se hacen contratos a palo se puede contar puntos cuando tienes pocas cartas de otros palos, puesto que eso generara que puedas *Fallar* cuando se juegue ese palo y te permita tirar una carta del palo de la apuesta.
  - Semifallo(Tienes 2 cartas de ese palo): 1
  - Dubletón(Tienes 1 carta de ese palo): 2
  - Fallo(No tienes ni una carta de ese palo): 3

  También se pueden añadir puntos de distribución si tienes mas de ocho cartas del palo apostado entre tu compañero y tu. Añadiendo puntos a la diferencia entre la cantidad de cartas del palo menos ocho

En la fase de defensa, los jugadores que ganaron la subasta deberán intentar cumplirla mientras que los contrincantes intentaran hacer que no cumplan el contrato. Comienza la partida el jugador a la izquierda de la persona que se llevo el contrato. Si es la primera carta de la ronda, se podrá jugar cualquier carta de cualquier palo y los demás jugadores deberán seguir el palo que se tiro, en caso de no tener cartas del palo el jugador podrá *Fallar* y tirar una carta de otro palo. Una vez todos los jugadores hayan jugado una carta se analizara cual carta tiene más valor y el jugador que la haya tirado gana esa ronda. La carta que tiene mayor valor depende de dos factores en orden decreciente:

1. El palo al que se esta jugando la apuesta
2. El valor de la carta siendo el 2 la de menor valor y el as como la de mayor valor

Por lo que si se esta jugando a una apuesta de corazones, el dos de corazones le ganaría al as de espadas. En el caso que se este jugando a una apuesta sin triunfo, el palo de la primera carta jugada en cada ronda sera la que tendrá prioridad de las otras.

1.2. Orígenes del Bridge

El Bridge de Contrato, o simplemente "*Contract Bridge*", es una variante del juego de cartas conocido como *"Bridge"*, que se juega con una baraja estándar de 52 cartas y en el que participan cuatro jugadores divididos en dos equipos de dos.

El Bridge es un juego de cartas que ha evolucionado significativamente desde sus inicios. Su desarrollo puede localizarse desde los juegos de *bazas* del siglo XVIII, especialmente el "Whist", que fue popular en Inglaterra durante esa época. A mediados del siglo XIX, el "Bridge Whist"surgió como una variante del Whist [6], introduciendo el concepto de pujas para determinar el número de *bazas* que los jugadores se comprometían a ganar.

La transformación hacia el Bridge moderno comenzó a principios del siglo XX con la creación del "*Contract Bridge*". Este nuevo formato diferenció al juego al permitir a los jugadores hacer ofertas sobre el número de *bazas* que su equipo ganaría y seleccionar un palo como triunfo, o declarar que no habría triunfo. Esta mecánica añadió un nuevo nivel de estrategia y complejidad al juego.

El Contract Bridge fue formalizado y estandarizado en la década de 1930. El año 1932 es significativo en la historia del Bridge porque fue cuando se celebró el primer campeonato mundial de, lo que sentó las bases para la creación de la World Bridge Federation (WBF) en 1958 [7]. La WBF se estableció la para regular y promover el juego a nivel internacional. La estandarización de las reglas y la creación de torneos internacionales contribuyeron a la popularidad del Bridge durante el siglo XX.

Aunque el Bridge sigue siendo un juego respetado, su popularidad general ha disminuido en comparación con décadas anteriores, especialmente en comparación con otros juegos de cartas y actividades recreativas más recientes. Las nuevas generaciones han mostrado menos interés en el Bridge en comparación con juegos de mesa más modernos y videojuegos. La complejidad y el tiempo requerido para aprender y jugar al Bridge pueden ser barreras para su adopción entre los jóvenes.

Sin embargo, el Bridge mantiene una notable presencia en varias regiones del mundo. En Estados Unidos, el juego sigue siendo fuerte gracias a la American Contract Bridge League (ACBL), que organiza varios torneos y eventos. En el Reino Unido, la English Bridge Union (EBU) continúa promoviendo una sólida tradición de Bridge.

En Francia, el Bridge cuenta con una rica tradición respaldada por la Fédération Française de Bridge (FFB), que organiza importantes eventos y competiciones. En los Países Bajos y Bélgica, el Bridge sigue siendo popular, con la Nederlandse Bridge Bond (NBB) y la Belgian Bridge Federation (BBF) desempeñando roles clave en su promoción.

Italia también conserva una destacada presencia en el Bridge, con la Federazione Italiana Bridge (FIB) liderando la organización de torneos y eventos. En China, aunque el interés en el Bridge es más reciente, está en crecimiento gracias a los esfuerzos de la China Contract Bridge Association (CCBA). Además, en Suecia y Alemania, el Bridge mantiene una fuerte presencia, con la Svenska Bridgeförbundet (SBF) y la Deutscher Bridge-Verband (DBV) organizando actividades y competiciones importantes.

Aunque el Bridge enfrenta desafíos en atraer a las nuevas generaciones, los esfuerzos para revitalizar el interés incluyen la promoción del juego en escuelas y universidades, así como el desarrollo de plataformas digitales y aplicaciones que facilitan su aprendizaje y participación. Estos esfuerzos están destinados a asegurar que el Bridge continúe siendo relevante y accesible para futuros jugadores en todo el mundo.

1.3. La digitalización de los Juegos de Mesa

En la actualidad, vivimos inmersos en la era tecnológica, donde los avances digitales han transformado múltiples aspectos de nuestra vida cotidiana, incluyendo el entretenimiento. Este cambio ha generado un notable declive en la popularidad de los juegos tradicionales, como el ajedrez, las cartas o los juegos de mesa, que han sido reemplazados en gran medida por versiones digitales. La conveniencia, la interactividad y la accesibilidad de los juegos digitales han capturado el interés de la generación moderna, que prefiere la inmediatez y la posibilidad de jugar en cualquier momento y lugar.

Con el avance de la tecnología, los juegos tradicionales enfrentan varios desafíos. La creación de juegos modernos han captado a la juventud tanto por los gráficos avanzados y realistas y el frenesí que generan. Las plataformas sociales permiten jugar junto con tus amigos mientras entablas conversaciones a distancia con ellos o con desconocidos.

A pesar de esta tendencia, la tecnología también ha abierto nuevas posibilidades para mantener vivos los juegos tradicionales y llevarlos a una público global. Las plataformas digitales han permitido una reinvención y una extensión de estos juegos. Muchos juegos tradicionales han sido adaptados a formatos digitales, permitiendo a los jugadores disfrutar de ellos en dispositivos electrónicos. Juegos de cartas como el Bridge, el póker y el solitario, así como juegos de mesa como el ajedrez y el backgammon, están disponibles en versiones digitales que no pierden ninguna función con la versión física y además tiene la ventaja de poder jugar en línea con personas de todo el mundo.

Estas reinvenciones de juegos tradicionales permiten atraer a un público mas joven tanto por apariencia como con funcionalidades extras que se ponen, al igual que una ayuda más directa a principiantes con modos de práctica.

1.4. Motivación del proyecto

El desarrollo de videojuegos siempre ha sido un interés al cual he querido probar. Tanto por la parte de la generación de modelos renderizables como la implementación de mecánicas en un juego, o la propia realización de todo el flujo de él. Cuando vi la oferta de trabajo primero me dio una indecisión de si seria un tema muy difícil de realizar en poco tiempo, pero a medida que se iba desarrollando vi, a lo que con el tiempo vi que era posible de realizar toda la base. A esto también se añade el descubrimiento un juego de cartas muy divertido con bastante complejidad y mucha estrategia.

Para un mayor entendimiento del juego Gabriele Salvatore de Blasio ofertaba cursos de Extensión Universitaria con clases de iniciación al Bridge. Al cual junto con su compañero Manuel Fernando Duque de Lara se llego a entender bastante bien las bases del juego y sus reglas.

Capítulo 2

Estado actual y objetivos iniciales

En esta sección, se proporcionará un análisis del estado actual del arte, revisando aplicaciones similares existentes y estableciendo los objetivos iniciales del proyecto. Este enfoque permitirá contextualizar el proyecto dentro del panorama actual y definir claramente las metas que se buscan alcanzar.

2.1. Revisión de aplicaciones similares

En la actualidad, la plataforma online más utilizada para jugar y aprender al Bridge se llama **Bridge Base Online (BBO)** [8], en la Figura 2.1 se puede observar el aspecto visual de la plataforma. BBO ofrece varias funcionalidades, pero también presenta ciertas limitaciones que pueden influir en la experiencia del usuario.

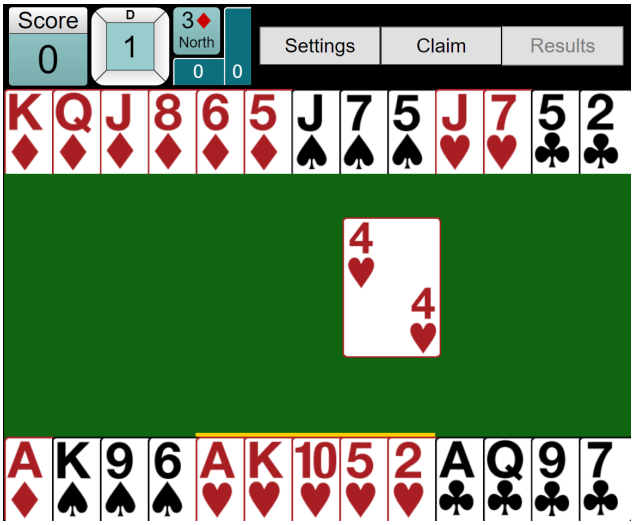

Ilustración 2.1: Captura del Bridge Base Online

Características de Bridge Base Online (BBO):

- **Juego en Tiempo Real:** BBO permite a los usuarios participar en partidas en tiempo real contra jugadores de todo el mundo, así como jugar contra la máquina. Esta característica proporciona una amplia gama de oponentes, lo que puede enriquecer la experiencia de juego, pero la calidad de la competencia puede variar considerablemente.
- **Modos de Juego:** La plataforma incluye varios modos como torneos, mesas de práctica predeterminadas y la opción de ver partidas de jugadores destacados. Aunque estos modos ofrecen distintas oportunidades para jugar y aprender, la interfaz de usuario de BBO puede no ser siempre intuitiva, y la experiencia en algunos de estos modos puede ser menos fluida de lo deseado.
- **Visualización y Análisis de Partidas:** BBO permite a los usuarios observar y analizar partidas de jugadores destacados. Esta función es útil para aquellos que buscan aprender de los expertos, pero la accesibilidad y la profundidad del análisis proporcionado podrían mejorarse.

Consideraciones a Mejorar:

- **Interfaz de Usuario y Diseño:** El diseño de BBO puede parecer algo tradicional y menos moderno en comparación con otras plataformas más recientes. Esta apariencia puede no ser tan atractiva para los nuevos jugadores y podría beneficiarse de una actualización visual para mejorar la experiencia general.
- **Experiencia del Usuario:** Aunque BBO ofrece una gama de funcionalidades, algunos usuarios han encontrado que la plataforma puede no siempre ser la más intuitiva o fácil de usar, especialmente para aquellos que se están iniciando en el juego.

Es decir, **BBO** es una plataforma establecida con una serie de herramientas y modos para jugar y aprender Bridge. Sin embargo, presenta algunas áreas que podrían beneficiarse de mejoras, especialmente en términos de diseño y facilidad de uso, para atraer y retener a una mayor audiencia.

Además de las plataformas que ofrecen una experiencia completa de juego, existen servicios que se especializan en la práctica de la fase de subasta, una etapa fundamental en el Bridge que influye significativamente en el desarrollo de toda la partida. La fase de subasta es crucial porque establece las condiciones bajo las cuales se jugará el contrato, y una comprensión sólida de esta fase puede marcar la diferencia entre una partida ganadora y una perdedora.

Un ejemplo representativo de estos servicios especializados es el **Standard American Yellow Card bidding practice (SAYC)** [9]. Este servicio se enfoca específicamente en la práctica y perfeccionamiento de la fase de subasta bajo el sistema SAYC. La Figura 2.2 muestra la interfaz de la plataforma, proporcionando una vista de las herramientas y recursos que ofrece a los usuarios.

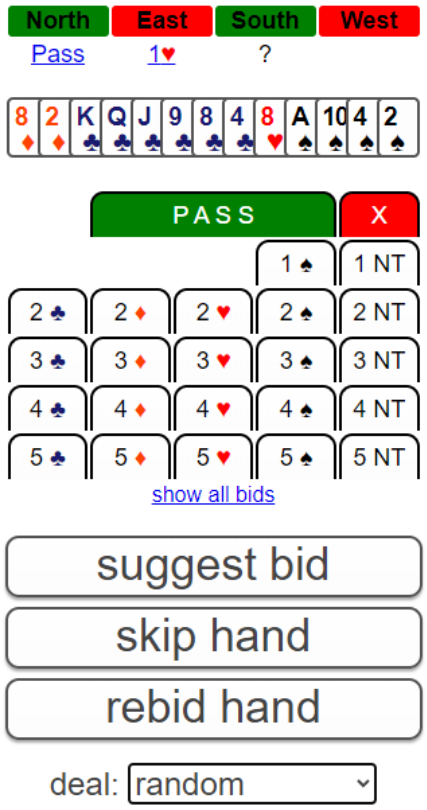

Ilustración 2.2: Captura Bidding Practice - SAYC Bridge

Características del SAYC:

- **Enfoque en la Subasta:** SAYC se centra en la práctica de la fase de subasta bajo el sistema Standard American Yellow Card. La plataforma ofrece ejercicios y simulaciones para familiarizarse con las convenciones y estrategias del sistema.
- **Interfaz de Usuario:** La interfaz está diseñada en la práctica intensiva de la subasta, permitiendo a los jugadores experimentar con diferentes escenarios y tomar decisiones en tiempo real.

Ventajas y Limitaciones:

- **Ventajas:** Al centrarse en la subasta, SAYC permite a los jugadores mejorar su comunicación con su compañero y tomar decisiones más informadas en esta fase crucial del juego.
- **Limitaciones:** Dado que SAYC se enfoca solo en la subasta, puede no ser ideal para quienes buscan una experiencia de juego más completa o para principiantes que necesiten más orientación.

En resumen, el SAYC es útil para perfeccionar la fase de subasta en Bridge, ayudando a los jugadores a entender mejor las comunicaciones con su compañero y a tomar decisiones más efectivas en esta etapa crítica. Aunque el enfoque de SAYC está específicamente en la

subasta, lo que permite una práctica intensiva en este aspecto, puede no cubrir todas las necesidades de quienes buscan una experiencia de juego más completa.

Otro servicio multi plataforma que ofrece toda el aspecto del Bridge es **Funbridge**[10] el cual también se puede aprender en él. La figura 2.3 muestra la aplicación junto con una de sus funcionalidades.

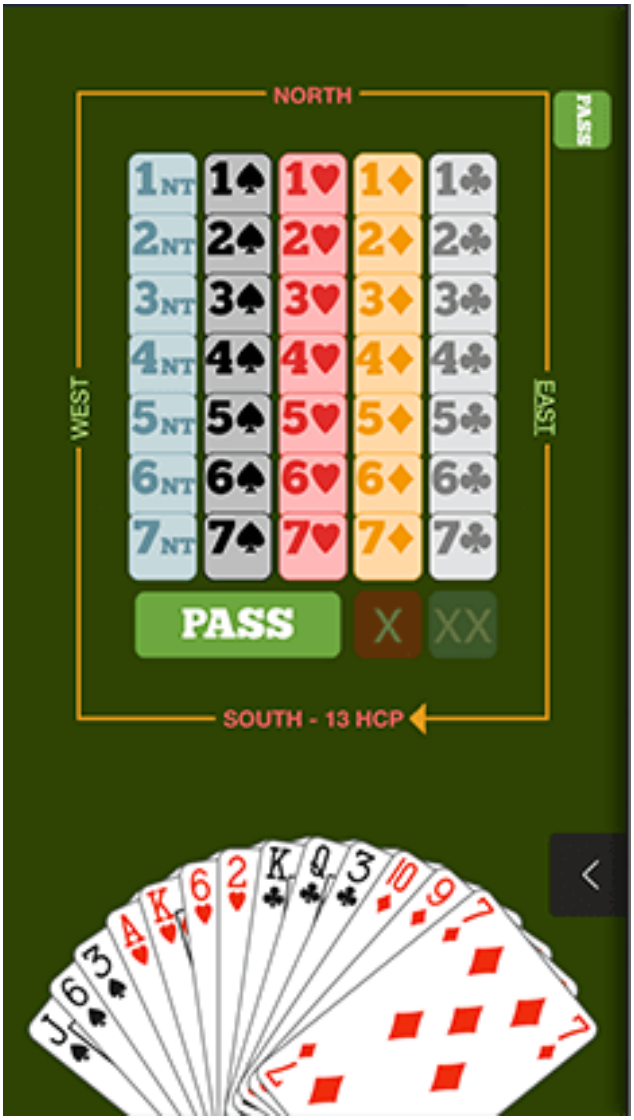

Ilustración 2.3: Captura del Funbridge

Características del Funbridge:

- **Juego solo contra maquina:** En Funbridge hay un sistema de amigos pero muy limitada, el juego es casi exclusivo contra maquina.
- **Interfaz de Usuario:** Tiene un estilo moderno en la que acepta ambas orientaciones de pantalla y diferentes dispositivos.

Ventajas y Limitaciones:

- **Ventajas:** Tiene herramientas separadas de aprendizaje con bastante utilidad. por ejemplo, hay un decodificador de apuestas en la que se centra en decirte el significado de las apuestas a medida que se van haciendo con sus características de cada palo y si tienen algún significado especifico como se observa en la figura 2.4.

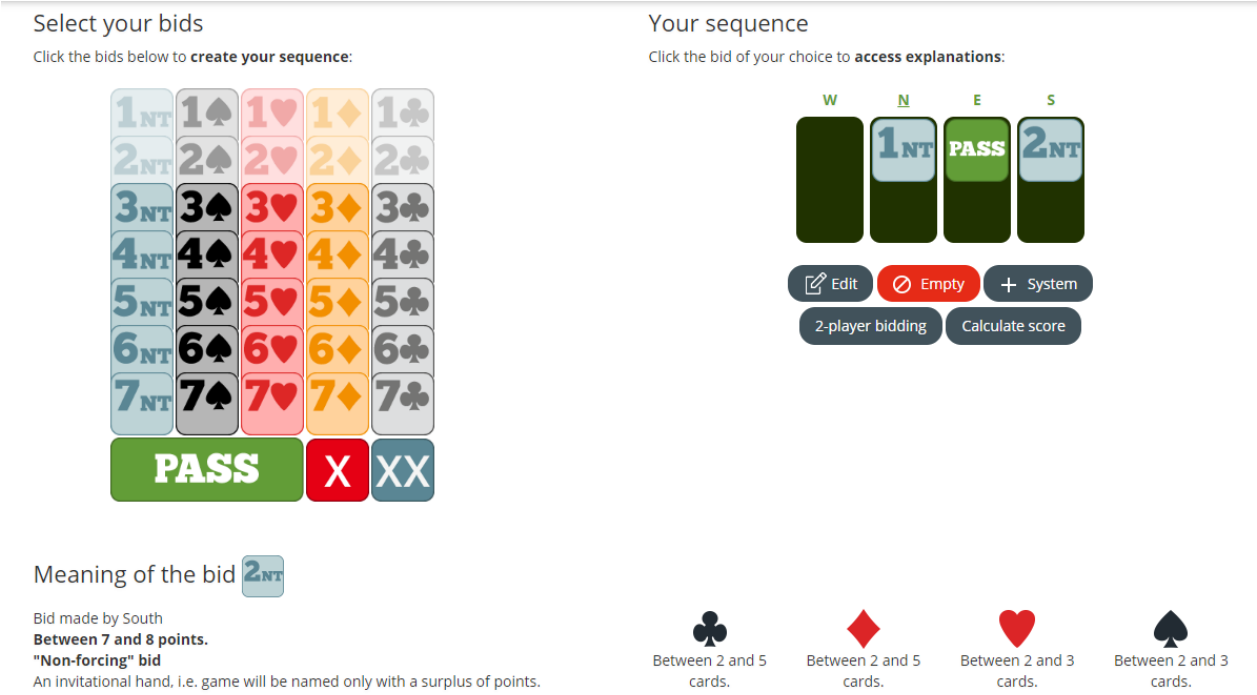

Ilustración 2.4: Captura del decodificador de subasta

- **Limitaciones:** Funbridge es mayoritariamente de pago en formato de subscripción. La cuenta gratuita solo te deja jugar dos manos al día, se pueden comprar mas partidas con una moneda de pago. También las funciones de torneo cuestan dinero.

2.2. Objetivos del proyecto

Este proyecto de fin de título surge por el interes del juego del Bridge de contrato y de la oportunidad de probar nuevas tecnologías. Con este propósito en mente, se han definido los siguientes objetivos:

- **Aprendizaje de lenguaje y utilización de motor de videojuegos:** Dado que se trata de una aplicación para un juego de mesa, se evaluarán distintos motores de videojuegos para determinar cuál es el más adecuado para este proyecto. Además, se considerarán los lenguajes utilizados por cada motor y la cantidad de documentación disponible, así como las librerías externas que ofrecen.

- **Desarrollo de una aplicación de Bridge de contrato:** Crear una aplicación que permita a los usuarios jugar al Bridge de contrato. La interfaz de usuario será diseñada para ser intuitiva y fácil de usar, con el objetivo de ofrecer una experiencia de juego accesible tanto para principiantes como para jugadores experimentados. La aplicación debe garantizar una navegación fluida y ofrecer una presentación clara de la información del juego.
- **Implementación de parámetros personalizables para la fase de subasta:** Proporcionar a los jugadores la capacidad de ingresar parámetros personalizados antes de comenzar una partida. Esto permitirá iniciar el juego con determinadas condiciones predefinidas durante la fase de subasta, facilitando así la práctica de diferentes estrategias y apuestas. Esta funcionalidad está pensada para mejorar las habilidades de los jugadores, ofreciendo un entorno controlado donde pueden experimentar con diversas tácticas.
- **Incorporación de un sistema de ayuda interactivo:** Añadir un sistema de asistencia que ofrezca sugerencias y pistas al jugador cuando este lo requiera. Este sistema estará diseñado para detectar momentos en los que el jugador podría necesitar orientación y proporcionará recomendaciones sobre posibles movimientos a realizar. El objetivo es apoyar al jugador en su proceso de aprendizaje, ayudándole a comprender mejor las reglas y estrategias del juego.

Estos objetivos buscan no solo crear una herramienta útil para la práctica y mejora en el juego de Bridge de contrato, sino también aprovechar el desarrollo de esta aplicación como una oportunidad para experimentar con diferentes tecnologías y metodologías de desarrollo de software. Se espera que el proyecto no solo beneficie a los jugadores de Bridge, sino que también contribuya al desarrollo profesional durante su creación.

Capítulo 3

Competencias específicas y aportaciones del trabajo

Este proyecto tiene como objetivo no solo facilitar el aprendizaje del Bridge, sino también demostrar cómo las herramientas digitales pueden revolucionar la enseñanza de juegos de mesa complejos. Al combinar pedagogía y tecnología, se espera proporcionar una solución accesible y efectiva para todos aquellos interesados en adentrarse en el fascinante mundo del Bridge. Antes de detallar el desarrollo del proyecto, se procederá a describir las competencias y aportaciones cubiertas por el mismo.

3.1. Competencias

En este trabajo se ha cubierto la competencia **IS04**: Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

Para la realización de esta competencia, se contemplaron las posibles etapas del ciclo de vida del desarrollo de software:

1. **Identificación, análisis y diseño de problemas:** Se emplean herramientas y metodologías como diagramas de casos de uso para tener un abstracto de los objetivos y posibles problemas del programa.
2. **Desarrollo y implementación:** Se investigan las tecnologías disponibles y entornos de desarrollo que se ajusten a los requisitos necesarios para implementar las funcionalidades. necesarias.
3. **Verificación y validación:** Se llevan a cabo revisiones de código con testeo para el correcto funcionamiento del programa.

Este trabajo intenta aportar una alternativa a las posibles prácticas del juego del Bridge. Puesto que no hay actualmente muchos servicios que ayuden en la práctica de las bases del

juego, esta propuesta busca aportar mas recursos y herramientas educativas accesibles para jugadores principiantes.

3.2. Aportaciones al entorno socio-económico

El software que se desarrollará está diseñado específicamente para el aprendizaje de las bases del juego de Bridge. Desde una perspectiva social, el software proporciona un acceso ampliado a recursos de aprendizaje, permitiendo a los jugadores practicar y mejorar sus habilidades en un entorno digital promoviendo así el juego de Bridge y atrayendo a nuevos jugadores.

Desde el punto de vista económico, el software tiene la característica de ser un programa de código abierto, lo que significa que está disponible de manera gratuita para todos los usuarios, eliminando barreras económicas y fomentando una mayor participación. La disponibilidad del código abierto también permite cualquier desarrollador a contribuir al proyecto, mejorando su funcionalidad y adaptándolo a diferentes necesidades, aunque no se espera un impacto económico significativo en términos de ingresos directos debido a su enfoque en fines educativos y de práctica.

Capítulo 4

Metodología y Análisis

En este capítulo se describirá la metodología seguida durante el proyecto. Detallando tanto las herramientas y lenguajes de programación utilizados, así como, explicando las razones detrás de su selección. Para ello se consideraron factores como licencias, la curva de aprendizaje, y las necesidades específicas del proyecto. También, presentaremos un análisis preliminar, que nos servirá para definir los requisitos y funcionalidades necesarias. Los análisis iniciales para un proyecto de software son esenciales para asegurar que el desarrollo sea factible y se alinee con las expectativas y necesidades de los usuarios.

4.1. Metodología

A lo largo del proyecto, se adoptó una metodología de desarrollo ágil, la cual se centra en un enfoque iterativo e incremental. Esto permite que los requisitos y las soluciones se adapten y evolucionen con el tiempo según las necesidades del proyecto. Para implementar esta metodología, se utilizó el marco de trabajo SCRUM como referencia. SCRUM se basa en un modelo de colaboración en equipo en el que se aplican regularmente una serie de buenas prácticas para maximizar la productividad y el éxito del proyecto.

Como seguimiento del flujo de trabajo se utilizo el git como versionado de código a la vez que indicador de realización de funcionalidades.

Al no tener conocimiento de la tecnología y del motor de videojuego fue un desarrollo evolutivo mientras se aprendía tanto por la documentación original [11] como por vídeos publicados en Youtube por la comunidad en la que enseñaban funcionalidades especificas.

Además de la importancia de los aspectos técnicos del proyecto, al tratarse de un juego de mesa, fue indispensable aprender a jugarlo. Para ello, se asistió a clases para comprender el mundo del Bridge, ahí se impartieron los conocimientos fundamentales sobre las reglas del juego, las bases y las estrategias avanzadas. Es decir, aprender a jugar al Bridge fue fundamental para entender sus dinámicas, esto nos ayudó a implementar correctamente las

funcionalidades del software y asegurarnos de que el producto final fuera auténtico y satisfactorio para los jugadores.

4.2. Herramientas de desarrollo

Para la aplicación se buscaba un lenguaje o herramienta que ofreciera capacidades tanto para la creación de interfaces como para el desarrollo de algoritmos. Se optó por un motor de videojuegos, ya que cumplía con estos requisitos y, además, facilitaba la exportación del proyecto a entornos web o a otras plataformas si se deseaba. Al analizar las tendencias comunes, se observó los siguientes motores:

Ilustración 4.1: Logo de Unreal [1]

1. **Unreal engine:** Es un motor bastante potente con lenguaje principal C++ por lo que a la vez se puede generar programas bastante complejos. Ver logo en la Figura 4.1.

Ilustración 4.2: Logo de godot [2]

2. **Godot:** Es un motor de desarrollo de videojuegos de código abierto y gratuito. En los últimos años, ha experimentado un aumento significativo en su popularidad y su comunidad ha crecido rápidamente. Usa GDScript o en C#, aunque se pueden poner módulos escritos en C++. Ver logo en la Figura 4.2.

Ilustración 4.3: Logo de unity [3]

3. **Unity:** Uno de los motores mas populares tanto por su sencillez como por su facilidad de ser utilizado sin tener mucho conocimientos. El lenguaje principal de Unity es C# el cual es un buen lenguaje. Ver logo en la Figura 4.3.

Después de explorar las opciones mencionadas, se llegó a la conclusión de que Godot era la alternativa más adecuada. Unreal Engine fue descartado como opción debido a que, aunque es un motor de desarrollo muy completo, presenta un alto consumo de recursos y no tiene capacidades web ligeras, lo cual no se ajustaba a las necesidades del proyecto.

Por otro lado, en 2023, Unity modificó drásticamente sus términos y condiciones, convirtiéndose gradualmente en una plataforma de pago con una estructura de licenciamiento más restrictiva. Para evitar posibles problemas relacionados con licencias y costos futuros, se decidió no utilizar Unity.

En su lugar, se eligió Godot debido a su equilibrio entre funcionalidad y eficiencia de recursos, así como su capacidad para exportar proyectos a entornos web y otras plataformas sin los inconvenientes de las alternativas descartadas.

Otros aspectos por la que elegir Godot fueron:

- Unreal Engine tiene una curva de aprendizaje bastante alto. Al ser todo en C++ es más propenso a tener fallos por referencias.
- Unity tiene una buena comunidad y su curva de aprendizaje no es muy alta. Sin embargo, C# es un lenguaje que no se tiene conocimiento y la forma de creación de escenas en Unity es algo que no parece ideal.
- Tanto Unreal Engine como Unity piden una cotización dependiendo de la cantidad de descargas del juego. Punto más desfavorable en Unity por sus cambios de condiciones mencionados previamente, puesto que Unreal Engine no tiene presupuestos tan agresivos.

4.2.1. Godot

Godot v4.2.2, tal y como se introdujo en la Sección 4.2, es un motor de desarrollo de videojuegos altamente versátil, de código abierto y gratuito. En los últimos años, ha experimentado un aumento significativo en su popularidad y su comunidad ha crecido rápidamente.

Godot permite la creación de juegos tanto en 2D como en 3D. Sin embargo, es importante reconocer que la madurez y el nivel de desarrollo en el apartado 3D son menos avanzados en comparación con el apartado 2D. A pesar de ello, facilita la exportación a múltiples plataforma de forma simple, con la única excepción de consolas, las cuales requiere compilar el motor con funcionalidades específicas para cada plataforma.

La interfaz de Godot es muy intuitiva, permitiendo arrastrar y soltar elementos en la escena. Estos elementos pueden ser ajustados en todas sus propiedades desde el panel correspondiente. Además, si se añaden propiedades personalizadas, estas pueden integrarse fácilmente en la interfaz del editor, lo que simplifica su modificación 4.4.

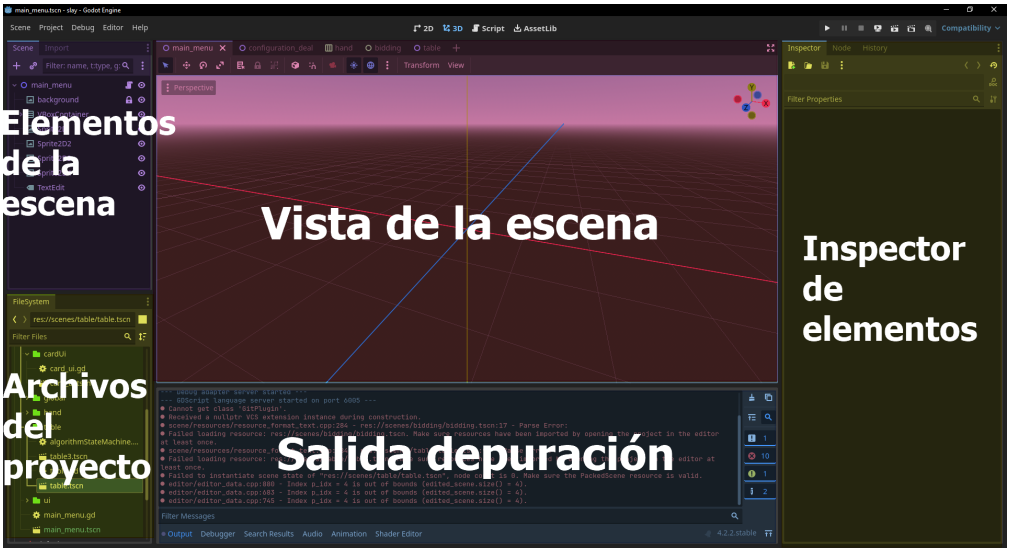

Ilustración 4.4: Interfaz de Godot

Al ser un proyecto de código completamente abierto, cualquier persona es capaz de contribuir a resolver problemas o desarrollar extensiones. Estas extensiones pueden ser publicadas en su biblioteca para que otros usuarios las utilicen.

Aunque Godot está diseñado principalmente para la creación de videojuegos, también se han desarrollado proyectos adicionales, como herramientas para la creación de arte pixelado, como el programa de dibujo Pixelorama, ver Figura 4.5.

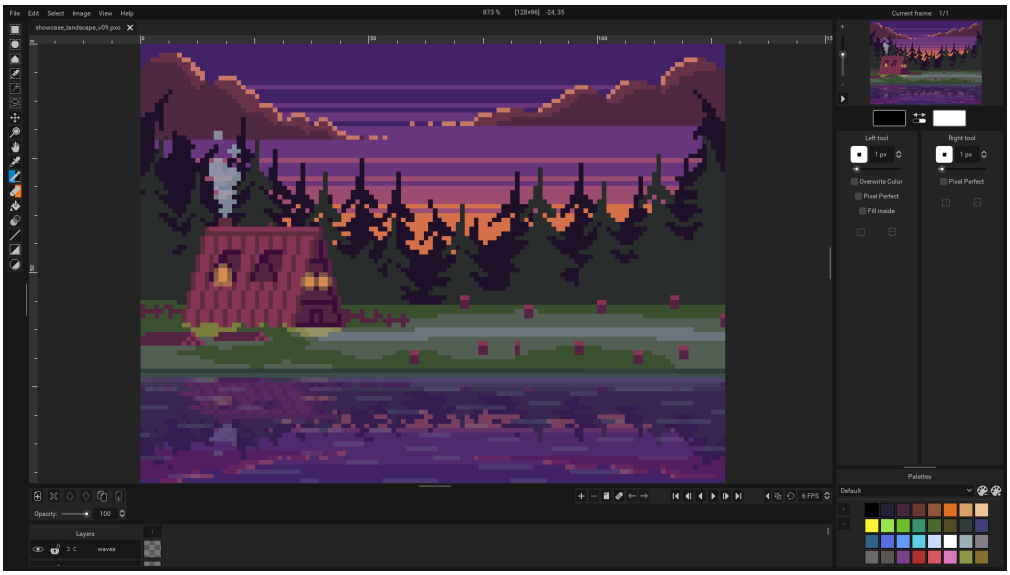

Ilustración 4.5: Pixelorama [4]

Godot esta orientado a una programación orientada a objetos, estos objetos podrán tener cada uno un código relacionado, el cual se ejecuta durante su ciclo de vida o en respuesta a señales provenientes de otros objetos. Estos códigos pueden ser escritos en su lenguaje oficial GDScript o en C#. Se decidió utilizar GDScript debido a que es el que tenia mejor documentación y ejemplos. Godot también permite realizar módulos escritos en C++, en el caso de que se requiera de velocidad de computo se puede realizar.

Fue la primera vez que se emplearon tanto Godot como GDScript, lo que implicó un proceso de aprendizaje continuo a lo largo del desarrollo. A medida que avanzaba el trabajo, se dedicó tiempo a familiarizarse con el uso de Godot y GDScript, superando los obstáculos que surgieron en el camino. Este proceso de aprendizaje resultó muy interesante, dado que la metodología empleada en Godot difiere significativamente de las utilizadas en otros entornos de programación, como el desarrollo web.

4.3. Análisis

En esta sección, analizaremos las necesidades de nuestro programa en detalle para asegurar que se aborden todos los aspectos fundamentales que permitirán su correcto funcionamiento para el usuario. Comenzaremos con una descripción de los requisitos básicos y luego profundizaremos en las funcionalidades específicas que deberá incluir el programa para cumplir con estos requisitos.

- El jugador puede iniciar una partida
- El jugador podrá decidir las condiciones que se cumplirán para una pareja
- El jugador podrá apostar la subasta que él crea que es correcta
- el jugador puede jugar cartas permitidas cuando sea su turno
- El jugador será capaz de volver para generar nuevas condiciones

4.3.1. Casos de uso

En esta subsección, se presentará el diagrama de casos de uso que ilustra las principales interacciones entre los usuarios y la aplicación, ver Figura 4.6. Este diagrama proporciona una representación visual clara de los distintos escenarios en los que los usuarios pueden interactuar con el sistema, destacando las funcionalidades clave y los procesos que permiten alcanzar los objetivos de la aplicación, partiendo del análisis realizado.

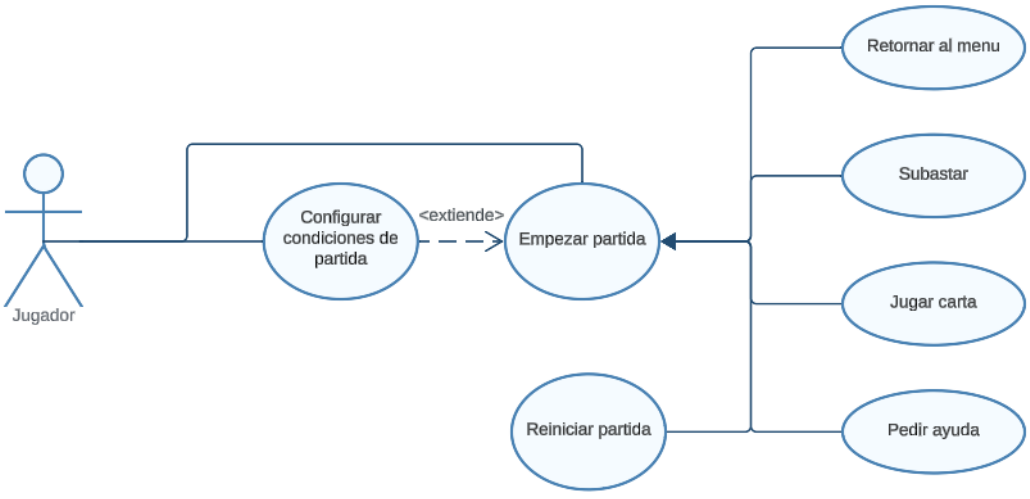

Ilustración 4.6: Diagrama de clases de uso de la aplicación

Al examinar el diagrama, se podrá obtener una comprensión detallada de los requisitos funcionales y las relaciones entre los diversos componentes y usuarios del sistema.

En este caso, el software de Bridge desarrollado no incluye funcionalidades que requieran la permanencia de datos o conexiones externas a internet o alguna base de datos. Debido a esta simplicidad en los requisitos, el diagrama de casos de usos es relativamente sencillo. Esto presenta la ventaja de reducir la complejidad del sistema, lo que a su vez lo hace más intuitivo. Las explicaciones de los casos de usos están ya realizadas en la sección de análisis y la mayoría de casos de usos tienen como requisitos estar en una partida de Bridge.

4.3.2. Especificación de casos de uso

La especificación de casos de uso es un paso crucial en el proceso de desarrollo de software, ya que ya que describe los requisitos funcionales del sistema desde la perspectiva de los usuarios. En esta sección, se detallarán los casos de uso identificados para la aplicación de Bridge, proporcionando una descripción de cómo los usuarios interactúan con el sistema.

Cada caso de uso representa una funcionalidad clave de la aplicación y describe un escenario específico en el que un usuario realiza una acción para alcanzar el resultado deseado. Estos casos de uso no solo muestran las interacciones entre los usuarios y el sistema, sino que también ayudan a identificar los requisitos necesarios para el desarrollo de cada funcionalidad, asegurando que el sistema cumpla con las expectativas de los usuarios y facilite una experiencia de usuario fluida y eficiente.

ID	1
Nombre	Empezar partida
Actor	Jugador
Precondiciones	
Postcondiciones	El jugador sera enviado a la mesa donde se le repartirá las cartas
Flujo	1. El jugador accede a la aplicación 2. El jugador presiona al botón de "*partida rápida*"
Flujos alternativos	

Cuadro 4.1: Especificación de casos de uso: Empezar partida

ID	2
Nombre	Configurar condiciones de partida
Actor	Jugador
Precondiciones	
Postcondiciones	El jugador sera enviado a la mesa donde se le repartirá las cartas con las condiciones aplicadas
Flujo	1. El jugador accede a la aplicación 2. El jugador presiona al botón de "*partida nueva*" 3. El jugador ajusta las condiciones a su gusto 4. El jugador añade la mano a la lista de posibles juegos 5. El jugador le da a "*empezar*"
Flujos alternativos	▪ El jugador no configura ninguna condición. Se empezara un juego con una configuración aleatoria

Cuadro 4.2: Especificación de casos de uso: Configurar condiciones de partida

ID	3
Nombre	Retornar al menú
Actor	Jugador
Precondiciones	El jugador esta observando la mesa de juego
Postcondiciones	El jugador sera enviado al menú de configuración de condiciones
Flujo	1. El jugador accede a la aplicación 2. El jugador procede al caso de uso 1 o 2 3. El jugador le da al botón de "*volver*"
Flujos alternativos	

Cuadro 4.3: Especificación de casos de uso: Retornar al menú

ID	4
Nombre	Subastar
Actor	Jugador
Precondiciones	El jugador esta observando la mesa de juego
Postcondiciones	
Flujo	1. El jugador accede a la aplicación 2. El jugador procede al caso de uso 1 o 2 3. El jugador tendrá la opción de ver sus cartas y subastar la opción que mas le convenga, ya sea aumentar la apuesta o *pasar* para en caso de que no pueda superar una apuesta ya declarada.
Flujos alternativos	

Cuadro 4.4: Especificación de casos de uso: Subastar

ID	5
Nombre	Jugar carta
Actor	Jugador
Precondiciones	El jugador esta observando la mesa de juego
Postcondiciones	
Flujo	1. El jugador accede a la aplicación 2. El jugador procede al caso de uso 1 o 2 3. El jugador a pasado la fase de subasta utilizando el caso de uso 4 4. El jugador podrá jugar una carta, que este permitida para jugar en su turno, mediante un simple toque de ratón en ella.
Flujos alternativos	

Cuadro 4.5: Especificación de casos de uso: Jugar carta

ID	6
Nombre	Pedir ayuda
Actor	Jugador
Precondiciones	El jugador esta observando la mesa de juego
Postcondiciones	
Flujo	1. El jugador accede a la aplicación 2. El jugador procede al caso de uso 1 o 2 3. El jugador a pasado la fase de subasta utilizando el caso de uso 4 4. Una vez sea el turno del jugador, tanto si es de la mano muerta como la suya, él podrá tocar un botón para que una ventana emergente le de una pista de posibles movimientos que le ayuden a mejorar sus posibilidades de ganar.
Flujos alternativos	

Cuadro 4.6: Especificación de casos de uso: Pedir ayuda

ID	7
Nombre	Reiniciar partida
Actor	Jugador
Precondiciones	El jugador esta observando la mesa de juego
Postcondiciones	
Flujo	1. El jugador accede a la aplicación 2. El jugador procede al caso de uso 1 o 2 3. El jugador podra presionar el boton de "*Reiniciar manos*"para que se vuelvan a repartir las cartas con las condiciones establecidas y empezar desde la subasta
Flujos alternativos	

Cuadro 4.7: Especificación de casos de uso: Reiniciar partida

Capítulo 5

Desarrollo

Después del análisis detallado descrito en el Capítulo 4, se desarrolló un prototipo inicial del programa, ver Sección 5.2. Esta primera versión permitió afinar las características del diseño y las interacciones del usuario, además de servir para familiarizarse con Godot. A partir de este prototipo, se avanzó hacia el desarrollo de la versión final del programa, ver Sección 5.3, esta versión incluye un menú inicial, configuraciones y diversas escenas.

5.1. Conceptos bases de Godot

Antes de empezar sobre el desarrollo se explicara algunos elementos de Godot.

5.1.1. Nodos y escenas

Godot, a diferencia de otros motores de videojuegos, va por una metodología más programación orientada por objetos.

En él los nodos se consideran como clases en programación estándar. Estas pueden ser extendidas para meterles más propiedades o alguna señal nueva. También al ser código abierto puedes generar módulos que contengan nodos y hagan uso de librerías terciarias. Un ejemplo es el uso de PhysX[12], una librería en C++ hecha por NVidia para el computo de físicas en tiempo real.

Cada nodo puede tener scripts que definen su comportamiento e interacción con otros nodos. Por ejemplo, un nodo de personaje puede tener un script para movimiento y colisiones, mientras que un nodo de botón puede tener un script para gestionar las acciones al hacer clic.

La forma de trabajar en Godot siempre parte de un nodo raíz, el cual podrá ser del tipo que desees en caso de que la aplicación sea 2D o 3D, y los demás nodos serán hijos de él, ver figura 5.1.

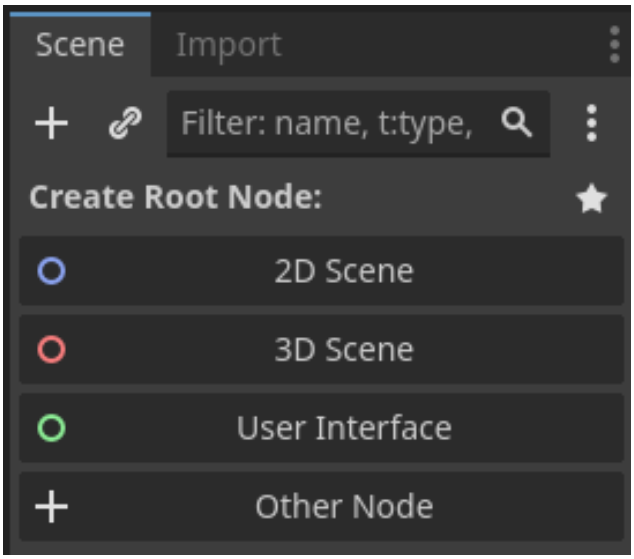

Ilustración 5.1: Inicio de un proyecto

Una escena es conjunto de nodos, la principal funcionalidad de las escenas es que se pueden inicializar, instanciando todos los nodos que la componen, dándole mucha utilidad aparte de una modularización para mayor claridad y limpieza. Estas escenas pueden ser contenidas en otras escenas, que se observan como si fueran nodos, y son indicadas en el arbol de elementos de Godot, ver figura 5.2.

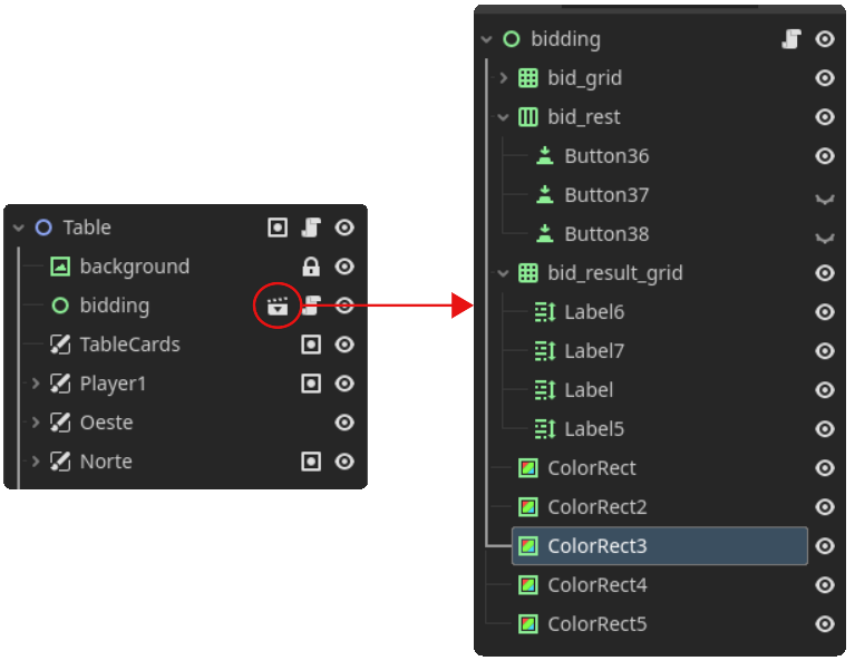

Ilustración 5.2: Indicación de una escena como nodo

5.1.2. Señales

En Godot la comunicación entre nodos se hace mediante señales, en C# lo más parecido seria los delegados o en c++/Python a los callbacks. Las señales se utilizan mayoritariamente en los casos que el nodo hijo quiera transmitirle información o quiere que procese datos al nodo padre, se podría hacer accediendo directamente, desde el nodo hijo, a los métodos del nodo padre pero se considera una mala practica. Para eso Godot permite generar señales, que son conectadas a funciones previamente, que pueden ser emitidas y el receptor procesara la señal. Godot te permite tanto hacer la conexión desde el código o desde su interfaz si son señales genéricas, ver figura 5.3.

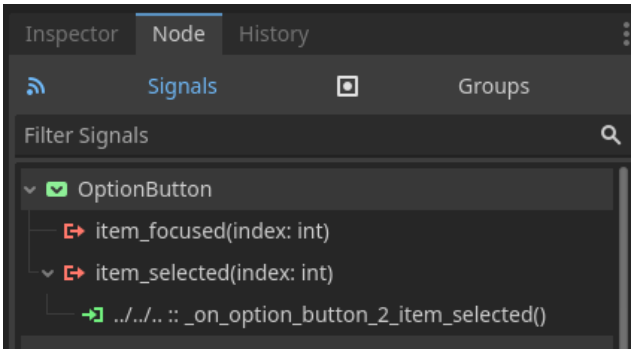

Ilustración 5.3: Ventana de señales

5.2. Prototipo

Para esta primera versión, se decidió implementar una mesa de juego simple. Esto sirvió tanto para familiarizarse con el motor de videojuego como para comprender su funcionamiento. Esta primera interacción es una escena del juego de mesa Bridge para cuatro jugadores, donde las cartas del jugador activo se encuentran en el centro inferior, Figura 5.4. Cada jugador tiene su posición asignada alrededor de la mesa, y ahí se desarrollarán las mecánicas básicas para la distribución y manejo de las cartas.

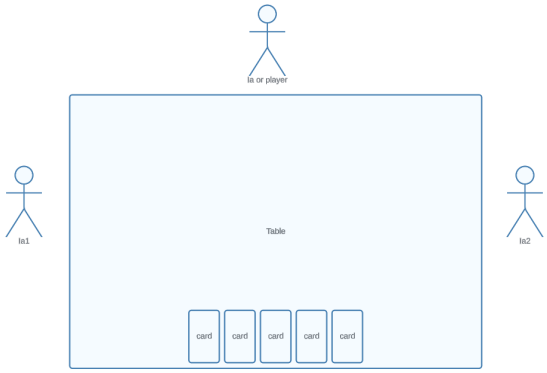

Ilustración 5.4: Croquis de primera iteración de la mesa

5.2.1. Nodos

Para este prototipo, al ser una aproximación inicial de la aplicación, se creó una escena simple. En la Figura 5.5 podemos observar el árbol de la escena, y en la Figura 5.6 el diagrama de clases de la escena.

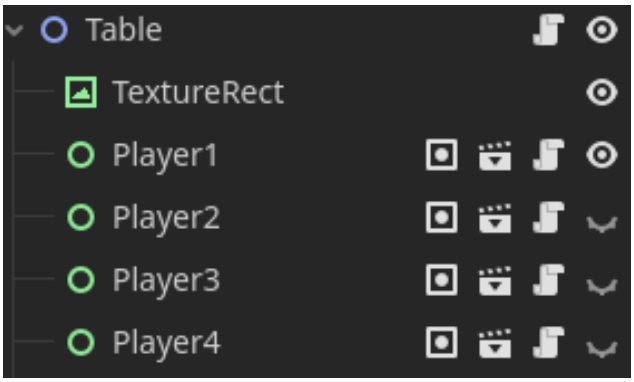

Ilustración 5.5: Árbol de la escena de la mesa de juego, cada ítem del árbol representa un nodo.

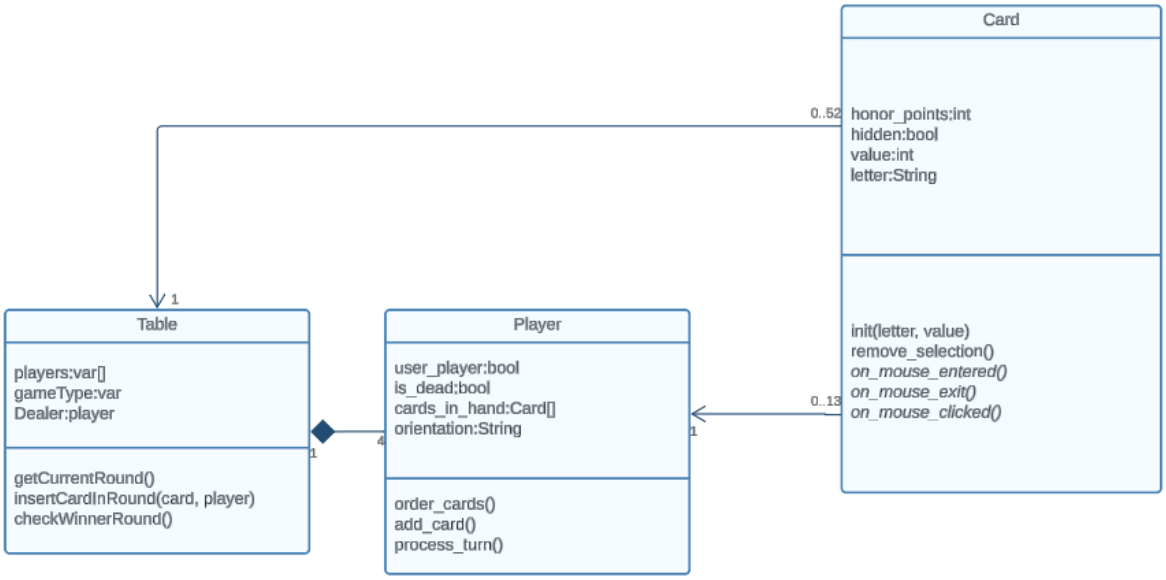

Ilustración 5.6: Diagrama de clases de la escena

Esta escena esta compuesta por lo siguiente:

- Un nodo raíz que representa la mesa y se encarga de repartir las cartas al comenzar el juego.
- Cuatro nodos, cada uno de los cuales representa a un jugador que contiene cartas en la escena.
- Un nodo de textura simple que sirve como fondo del juego.

Las cartas son entidades que son creadas y asignadas durante la ejecución del juego. Los jugadores deben poder interactuar con sus propias cartas. Una vez instanciados todos estos elementos, la primera versión de la escena se ve como se muestra en la Figura 5.7

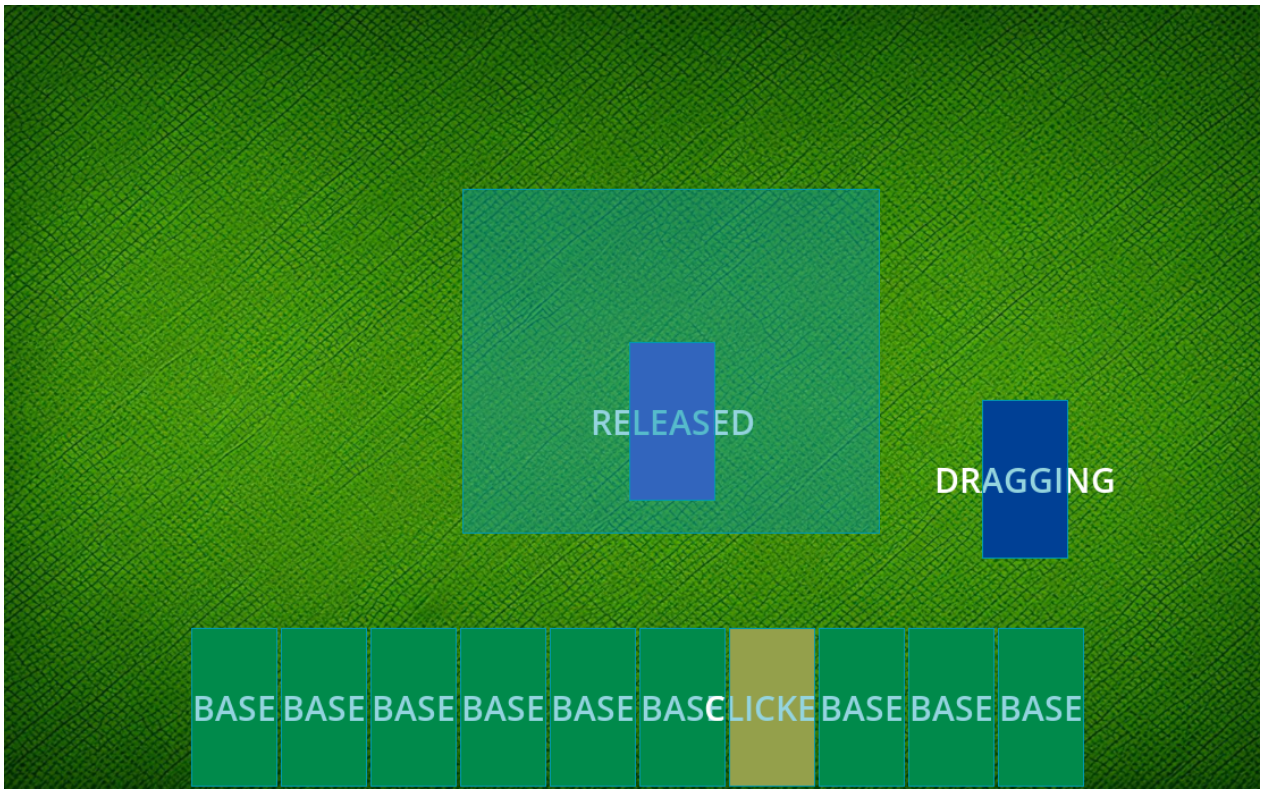

Ilustración 5.7: Escena del prototipo inicial

5.2.2. Maquina de estados

Para permitir la interacción con las cartas, se implementó una máquina de estados, ya que es la metodología más eficiente para gestionar el estado de cada carta, ver Figura 5.8.

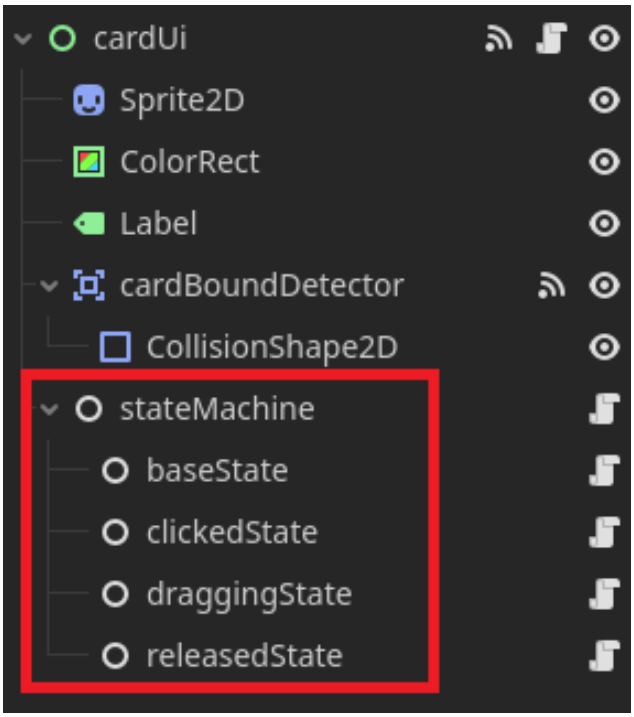

Ilustración 5.8: Componentes de la carta

Las maquinas de estados no es algo exclusivo de Godot, es un modelo de diseño que gestiona el comportamiento de un sistema en respuesta a eventos o condiciones específicas. Se compone de un conjunto de estados definidos, transiciones entre estos estados, y acciones que se ejecutan en respuesta a dichas transiciones. En nuestro caso desarrollamos 4 estados que se interactúan entre ellos, ver Figura 5.9. Estos estados son:

- **Base:** el estado por defecto en el que empiezan las cartas mientras están en la mano del jugador.
- **Clicked:** Una vez tocado la carta deseada se cambia a este estado para ser liberado de la mano.
- **Dragging:** Este estado es para que la carta siga el movimiento del ratón mientras vigila que se suelte el botón del ratón. En el caso que se toque el botón derecho del ratón se considera que el jugador decide cancelar el movimiento y se revierte al estado base.
- **Released:** Una vez soltado el ratón se comprueba la posición del ratón y si esta en una zona de colisión como se observa en la Figura 5.7 se aceptará la carta para ser jugada.

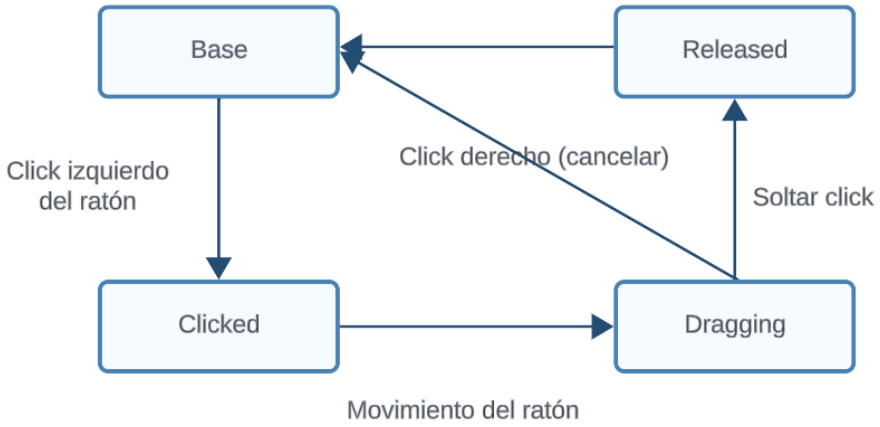

Ilustración 5.9: Máquina de estados del prototipo

5.3. Aplicación final

Tras el análisis realizado con el prototipo se procedió al desarrollo de la aplicación final. El juego consta de tres escenas principales, que irán cambiando a medida que el usuario avanza en la aplicación:

1. **Menú principal:** Este menú se muestra al iniciar el juego por primera vez. Desde aquí, podrás iniciar una partida rápida, comenzar un juego personalizado según tus preferencias, o salir de la aplicación. Figura 5.10.

Ilustración 5.10: Menú principal

2. **Menú de configuración de condiciones:** En este menú se podrá indicar cuantos puntos de honor tiene una pareja, determinar quién será la *primera voz*, y elegir si se jugará algún palo específico o *sin triunfo*. Además, se podrá configurar múltiples manos para el juego, seleccionándose una de manera aleatoria al iniciar cada partida. Figura 5.11.

Ilustración 5.11: Menú de configuración

3. **Mesa de juego:** En esta escena se desarrollan las dos fases principales del juego de Bridge. Aquí se llevan a cabo tanto la subasta como el juego de cartas, permitiendo a los jugadores interactuar y tomar decisiones estratégicas durante ambas etapas del juego.

   a) En la fase de subasta, que es la primera etapa, el jugador podrá elegir la oferta que considere adecuada en función de las cartas que tiene, ver Figura 5.12. La aplicación realizará sus propias ofertas siguiendo el criterio de *puntos de honor* y analizando su mano. Una vez concluida la fase de subasta, se anunciará la oferta ganadora y se determinará quién será el jugador que iniciará la ronda.

Ilustración 5.12: Ejemplo de fase de subasta

b) En la fase de carteo, si el jugador ganó la subasta, podrá ver las cartas de la pareja que actuará como *muerto*. En caso de haber perdido la subasta, la máquina jugará como tu pareja, ver Figura 5.13.

Ilustración 5.13: Ejemplo de fase de carteo

5.3.1. Máquina de estados

En la versión final de la aplicación, se llevaron a cabo ajustes significativos en la máquina de estados. Anteriormente, la funcionalidad de arrastre ralentizaba el flujo del juego, por lo que se decidió simplificar la máquina de estados a solo dos estados al eliminar estos controles. Esta modificación no solo agilizó el ritmo del juego, sino que también mejoró la intuición y la facilidad de uso de la aplicación.

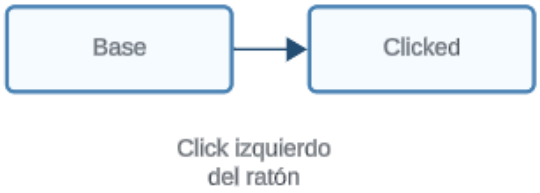

Ilustración 5.14: Máquina de estados de la aplicación final

5.3.2. Interfaz

En Godot cada nodo se puede guardar como una escena personalizada para tenerlo como una instancia separada. Aparte de las 3 escenas principales se generaron las siguientes:

- **Panel de apuesta**, Figura 5.15: Esta escena contiene todos los botones y la información de la apuesta para la fase de subasta. Cada vez que se toca un botón se comunica a la escena de la mesa para que decida que hacer.

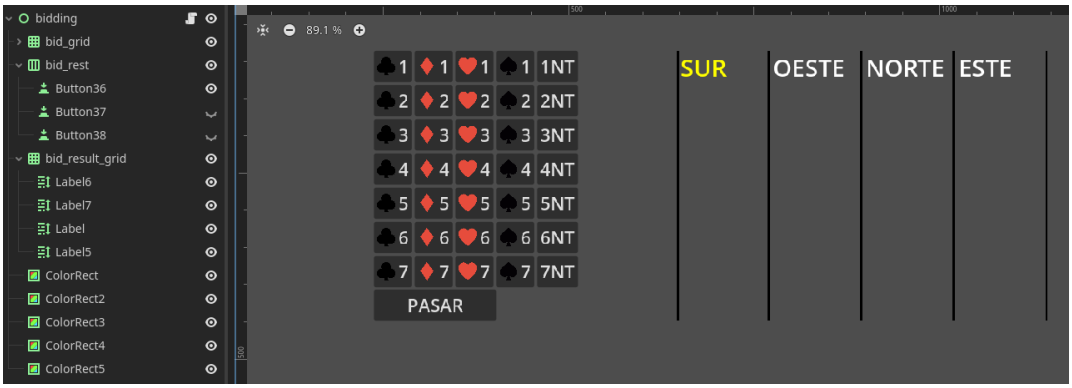

Ilustración 5.15: Escena de apuesta

- **Carta**, Figura 5.16: Cada carta se presenta como una escena personalizada, la cual contiene nodos de indicación visual que marca si la carta es seleccionable durante la jugada. Se utilizan dos barras rojas: una situada por encima de la carta y otra por debajo. Dependiendo de la posición de la carta, se mostrará el indicador correspondiente, facilitando así la selección correcta durante el juego. La mano será la responsable de hacer visible estos indicadores.

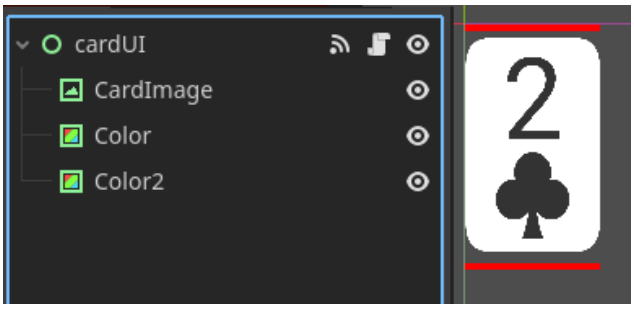

Ilustración 5.16: Escena carta

- **Mano**: La mano está diseñada como escena personalizada puesto que en un principio se diseño con una maquina de estados donde estarían los perfiles dificultad. La estructura se mantiene aunque no esté implementado.

La escena de la tabla es la que alojara las escenas descritas, y otros componentes ocultos de la pantalla. Con elementos ocultos nos referimos aquellos elementos que añaden funcionalidad a la aplicación, pero no deben ser visibles en la interfaz de la escena, para esto se diseñaron fuera de la misma. En la Figura 5.17 podemos ver que los elementos ocultos se encuentran definidos en el recuadro azul.

Ilustración 5.17: Escena tabla

Estos elementos ocultos, aparecerán en pantalla con animaciones puesto que son interfaces de información temporal. Como elementos se definió:

- **Panel de ayuda:** Este aparecerá deslizándose sobre la pantalla, y ofrecerá un breve resumen de la situación actual del jugador, junto con pistas sobre qué jugar a continuación. Este panel proporciona información útil para ayudar al jugador a tomar decisiones informadas durante la partida.

- **Panel de resumen:** Mensaje que aparece tras finalizar la subasta, proporcionando la información sobre la oferta ganadora y el jugador que comenzará la ronda.
- **Nodo de *muerto* para Este/Oeste**, Figura 5.18: Además de la mano normal para Este y Oeste, cuando alguno de estos actúa como *muerto*, se muestra sus cartas programadas con un nodo visual. Dado que este nodo es puramente visual y no contiene código funcional, no se promovió a escena.

Ilustración 5.18: Nodo de *muerto* en Este

5.3.3. Lógica del juego

Para la correcta ejecución de la partida de Bridge se genero un diagrama de actividad, Figura 5.19, con el propósito de ofrecer una visión abstracta y clara de los pasos a seguir. Este diagrama de actividad ayuda a poder descomponer el proceso de una partida de Bridge en una serie de etapas manejables y comprensibles.

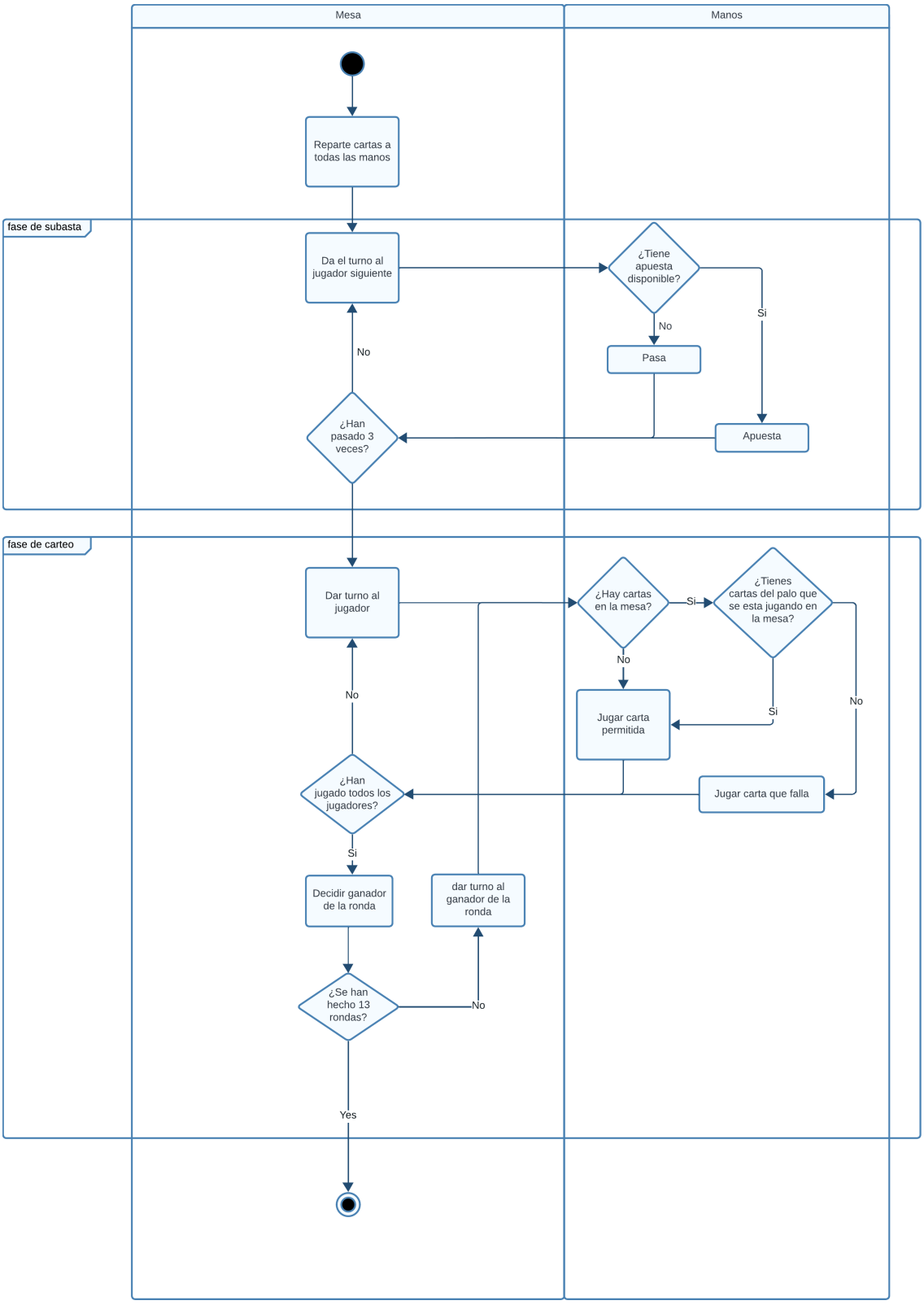

Ilustración 5.19: Diagrama de actividad

5.3.4. Reparto de manos

Para la generación de manos se investigo estándares de repartición de cartas. Una tesis[13] interesante sobre la generación tenia recolectado varias herramientas de generación de manos. Algo que tenían en común la mayoría de estas herramientas, es que utilizaban una metodología de generar y comprobar manos (**G**enerate **A**nd **T**est). La metodología de GAT es la generación aleatoria de manos y después comprobar si las manos cumplen las condiciones establecidas. En caso de no cumplirlas se vuelve a iterar una nueva generación hasta que se cumpla los requisitos necesitados.

Esta forma de generación no es completamente óptima y en unas manos muy especificas no llega a dar la solución, teniendo en cuenta que se suele poner un limite de intentos para que no este infinitamente generándose. Como en nuestro caso no vamos a poner condiciones muy restrictivas decidí implementar una generación por método de GAT.

```
procedure generar_manos()
    foreach asientos:                   //los 4 jugadores
        for 1 hasta 13:                 //Cada jugador tiene 13 cartas
            carta <- baraja.coger_carta_aleatoria
            if carta.puntos > 10
                asientos.punto_de_honor += carta.puntos - 10
            asientos.palo += 1
            asientos.mano <- carta
        end for
        asientos.regular <- comprabar_mano_regular
    end foreach
end procedure
```

En la propia creación de las manos voy añadiendo la información necesaria para la comprobación de cada mano como los *puntos de honor*, *puntos de distribución*, si la mano es *regular*, si la mano es *equilibrada*.

5.3.5. Fase de subasta

En la subasta la computadora deberá poner una apuesta adecuada con la mano que tenga, también se tendrá en cuenta lo que haya podido apostar su compañero u otros jugadores, en este último caso sería para quitarles la subasta a los contrincantes.

En el Bridge las apuestas básicas se rigen por unas pautas sencillas que son fácil de comprobar. Por ejemplo si se intenta comprobar si puedes hacer una apertura con *sin triunfo* lo primero que se debe comprobar es si tienes una mano *equilibrada*.

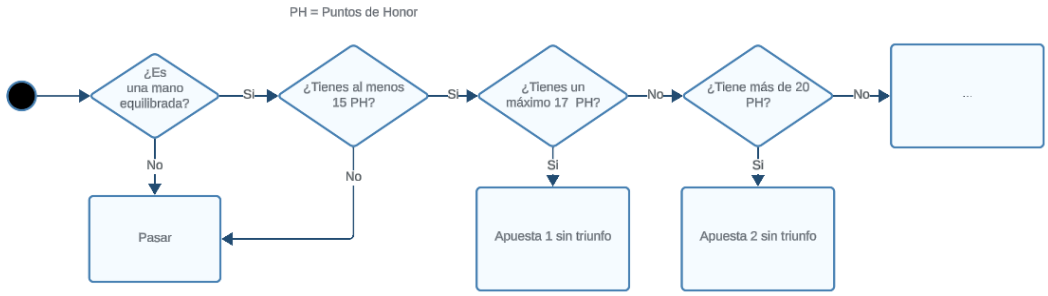

Ilustración 5.20: Pensamiento de apertura sin triunfo extrapolado

Como se ve en la Figura 5.20 el pensamiento para *sin triunfo* no es muy difícil, aunque sea para los 3 primeros niveles de apuesta de *sin triunfo*. En el caso de que el compañero abriera de *sin triunfo*, deberá pensar de una diferente forma puesto que son otros requisitos, Figura 5.21.

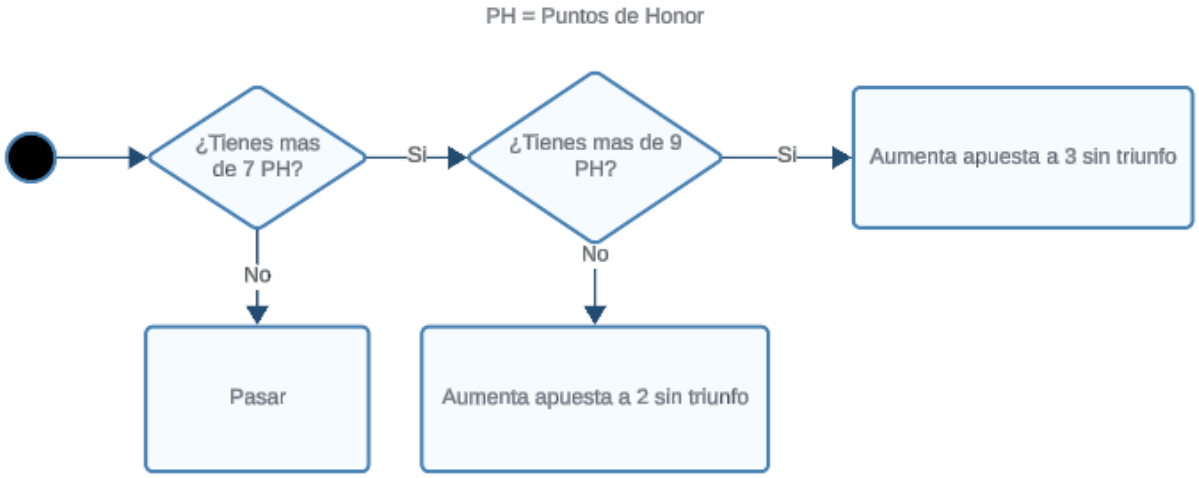

Ilustración 5.21: Pensamiento de continuación de sin triunfo extrapolado

5.3.6. Fase de carteo

Una vez acaba la fase de subasta se guarda en la escena de la mesa el ganador de la subasta y la apuesta. Comienza la mano a la izquierda del que se llevo la subasta siguiendo los puntos cardinales, siendo el jugador humano siempre sur.

Cada vez que la mesa le da el turno a un jugador, esta comprueba si esa mano controlada por la máquina.

- En el caso que la mano sea una controlada por el jugador humano, se procederá a analizar el estado de la ronda para habilitar las cartas que son posibles de jugar.El análisis consiste en comprobar si en la mesa de juego ya se ha jugado alguna carta. En

caso positivo se habilitaran todas las cartas del palo de la primera carta en la mesa. En caso negativo se habilitaran todas las cartas puesto que se considera que va a *fallar*.

- En el caso que la mano sea manejada por la computadora, esta intentara siempre seguir el palo que se ha jugado. Si es el abridor de la ronda, pues jugara la más fuerte que tenga. En el caso que la subasta es a un palo y va a *fallar* tendrá como prioridad jugar una carta del palo de la subasta para intentar ganar la ronda.

Las cartas jugadas en la ronda, pasan de estar en el nodo perteneciente a la *mano* a estar en un nodo de *almacenamiento* de la mesa, es decir, que cambian de tener a la *mano* como padre a tener el nodo *almacenamiento* como padre. Esto se realiza para para tener un acceso fácil a las cartas que hayan sido jugadas accediendo al nodo almacenamiento, que tendrá 4 hijos como máximo, o dicho de otro modo 4 cartas, pues al final de cada ronda se vacían las cartas jugadas en la mesa.

5.3.7. Comprobación de final de ronda

Cada vez que se juega una carta, le llega una señal a la mesa para que compruebe, en el nodo de almacenamiento mencionado en la Sección 5.3.6, si se han jugado ya cuatro cartas. En caso de haber cuatro cartas se comprobara quien es el ganador con un sencillo código:

```
procedure comprobacion_entre_cartas(carta_ganadora, carta_temporal)
    resultado_mayor = false
    if carta_temporal.palo == carta_ganadora.palo
        if carta_temporal.valor > carta_ganadora.valor
            resultado_mayor = true
    result resultado_mayor
end procedure

procedure comprobacion_ronda()
    ganador = cartas_jugadas[0]
    preferencia_palo = false
    for i en rango de 1..3:
        if ganador.palo == palo_subasta:
            if cartas_jugadas[i].palo == palo_subasta
                ganador = cartas_jugadas[i]
            else
                if comprobacion_entre_cartas(ganador, cartas_jugadas[i])
                    ganador = cartas_jugadas[i]
        else
            if comprobacion_entre_cartas(ganador, cartas_jugadas[i])
                ganador = cartas_jugadas[i]
    end for
    return ganador
end procedure
```

Siempre se utilizará la primera carta como la ganadora al principio y se comprobará con el resto de las otras cartas. Por un poco de limpieza se modula para no tener duplicidad de código.

Una vez se analiza que carta gano la ronda, se comprueba a que jugador le pertenece y se añadirá al contador positivo si es de la pareja que hizo la apuesta, al contador negativo en caso de que gane la otra pareja.

Una vez finalizada la ronda, se procede a comprobar si ya se han hecho las trece rondas. Si no es el caso se le dará el turno al siguiente jugador. Si es el caso, se considerará terminada la partida y el jugador tendrá la opción de utilizar el menú inferior para jugar otra partida reiniciando las manos o volviendo al menú de configuración, mostrado en la Figura A.7.

5.3.8. Sistema de ayuda

Cuando es el turno del jugador, este puede darle al botón de "*Ayuda*"para que le aparezca una breve información de la situación actual de la ronda, ver figura 5.22. Esta ventana analizará las cartas jugadas en esa ronda, las cartas que tienes en la mano y las cartas del *muerto* para dar una aproximación de una jugada aceptable.

Ilustración 5.22: Ventana de ayuda

Esta ventana aparece utilizando la funcionalidad de Godot llamada *tween*. El *tween* es un nodo que siempre van acoplados a otros nodos, su principal función es poder animar propiedades del nodo padre en una región de tiempo.

En este caso al panel de ayuda se le añadió un *tween* que modificaba la posición de la ventana de ayuda a lo largo de un segundo, haciendo un efecto de entrada y salida. Figura 5.23.

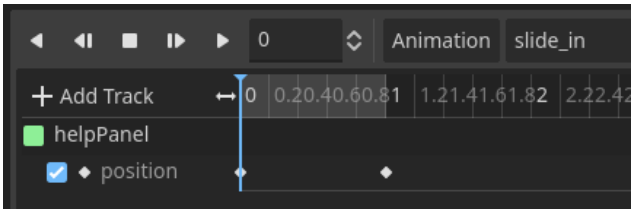

Ilustración 5.23: Panel de edición del *tween*

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusión

Este proyecto fue bastante educacional para comprender como funciona el mundo de la programación para videojuegos. Sin embargo, fue solo un aspecto de todo el trabajo en realizar un videojuego, puesto que el arte fue obtenido gratuitamente de recursos online con licencia abierta y tampoco se desarrollo el apartado de sonido. La experiencia que se ha tenido previamente en la creación de aplicaciones con interfaz discrepa bastante con como se desarrolla en un motor de videojuegos. Dicho esto, GDScript tiene mucha similitud con Python por lo que facilito la comprensión de a la hora de hacer código, como el concepto de señales para ejecución asíncrona.

La aplicación realizada no salio lo mejor estructurada posible tanto por falta de experiencia en realizar estos tipos de aplicación, como posibilidad de mal información en como estructurar. La transición de versiones de Godot 3 al 4 provoco un cambio drástico en la forma de crear código. En internet sigue habiendo mucha información de ambas versiones y es típico encontrar recomendaciones que ya no funcionan en la versión actual.

6.2. Trabajo futuro

El Bridge es un juego de carta la que destaca por su complejidad y sus diferentes formas de jugar.

- Algunas funciones que se podrían añadir que son un nivel intermedio serian:
  - Votación de doblo y redoblo.
  - Añadir perfiles de inteligencia artificial para jugar con diferentes convenciones.
  - Añadir configuraciones avanzadas en la implementación de condiciones de manos.
- A nivel de funciones de programa que se puede implementar:

  - El exportado y importado de condiciones de mano.
  - El exportado y importado de partidas realizadas como un modo de reproducirlas.
  - Implementar funcionalidad por internet como red de pares para que múltiples jugadores puedan jugar en la misma tabla.
  - En la tesis de generación de manos[13] se menciona otra forma de generación más precisa que se podría implementar.

- También la implementación de un resolvedor de mesa creado por Bo Haglund [14], el cual publicó la algoritmia de su librería *dummy dual solver* [15]. Aunque, cabe destacar, que la librería siempre sabrá todas las cartas de la mesa y quien las posee, incluso aquellas que no debería tiene información el jugador, por lo que la máquina a la hora de realizar las jugadas haría trampas al saber información del resto de jugadores. De igual manera, al jugador se le sugerirían jugadas con esa misma información, lo cual no es ideal para practicar movimientos en los que hay probabilidad de que fallen como el *impasse*.

Apéndice A

Manual de usuario

A.1. Menú principal

Controles: Ratón

Ilustración A.1: Botones del menú principal

1. **Partida rápida:** Procede a jugar una mano de Bridge completamente aleatoria
2. **Nueva partida:** Se pasa al menú de configuración de condiciones
3. **Salir:** Salida del juego

A.2. Menú de configuración

Controles: Ratón

Ilustración A.2: Menú de configuración

En este menúA.2 se pueden poner varias condiciones de como repartir la mano para no saber lo que apostar en el caso que hayan varias condiciones.

Ilustración A.3: Selector de predefinidos

Para rellenar rápidamente los campos se permite seleccionar al principio una selección de configuraciónA.3 predefinidas que luego podrán ser modificadas

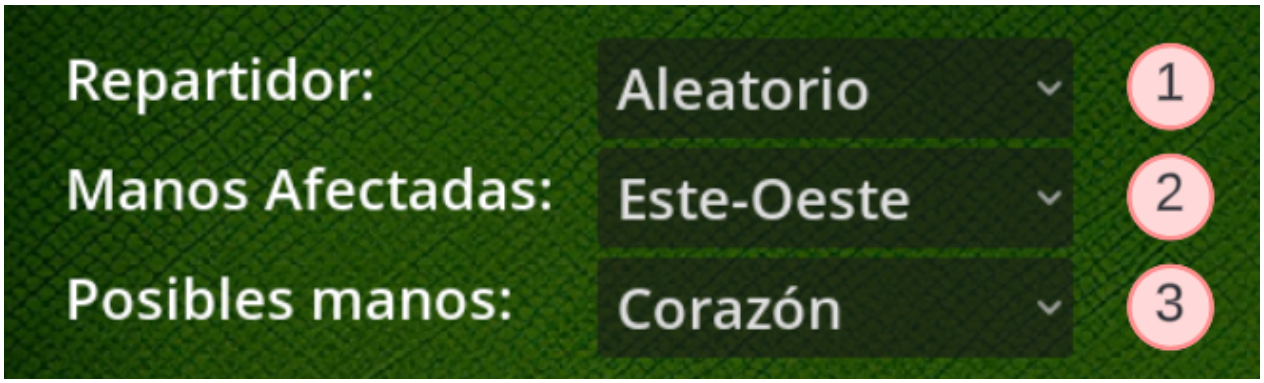

Ilustración A.4: Configuración básica

1. **Repartidor:** selector de quien dará la primera voz
2. **Manos Afectadas:** Se elegirá a que pareja le afecta las condiciones. El jugador siempre sera SUR
3. **Posibles manos:** Selector de que favorecerá las condiciones de jugar a un palo o a sin triunfo

Ilustración A.5: Puntos en las manos

En estos campos podrás indicar cuantos puntos de honor, y en caso de jugar a palo puntos de distribución, tendrá cada jugador de la pareja seleccionada en ”Manos Afectadas”A.4. El botón de ”Añadir mano” lo añadirá a la lista de resumen como una de las condiciones.

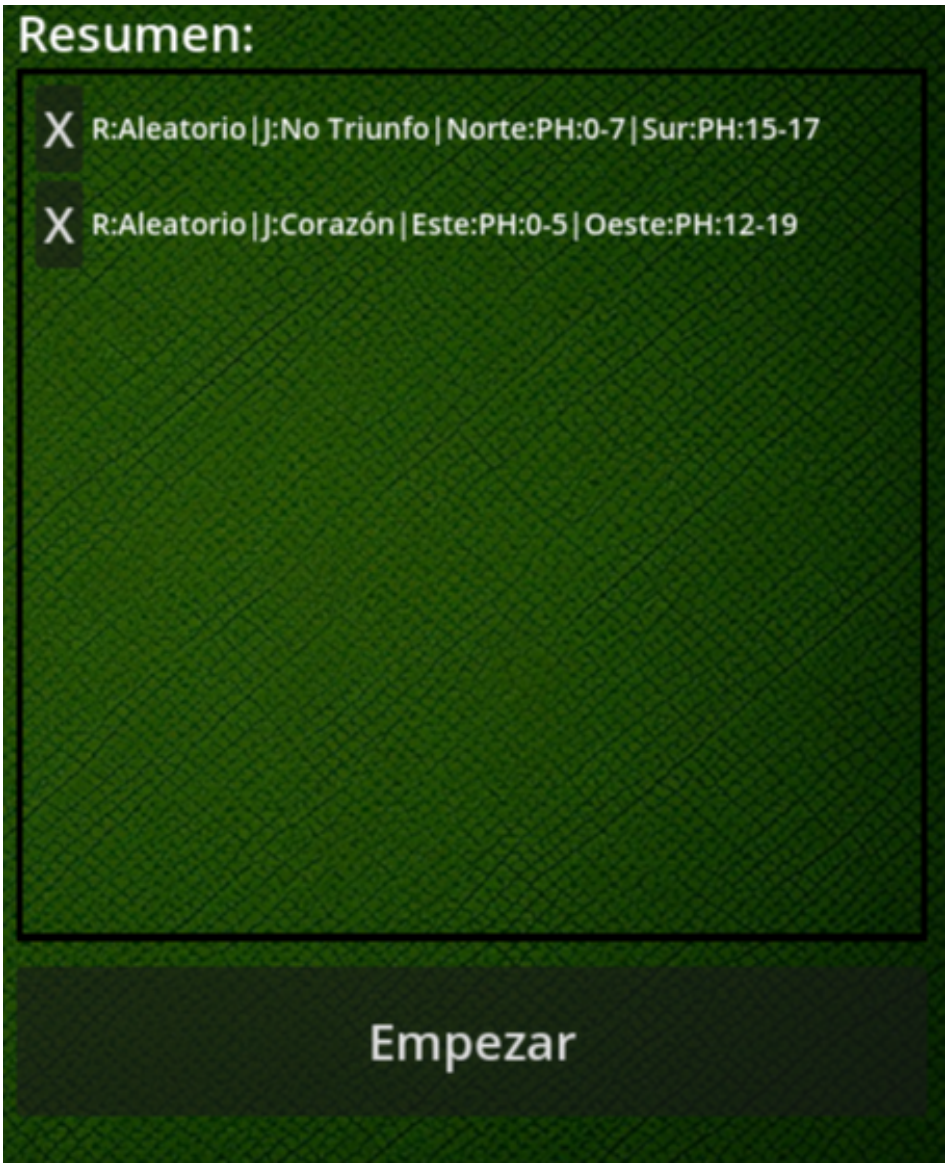

Ilustración A.6: Ventana de resumen

En la ventana se verán todas las configuraciones añadidas con un pequeño resumen de ellas. En caso de querer eliminar alguna solo hay que pulsar el botón al lado de ella. Una vez se haya configurado todo se le da a ”Empezar” para ir al tablero.

A.3. Tablero

Controles: Ratón

Ilustración A.7: Sección general del tablero

En la parte inferior del programa se podrá observar un pequeño menúA.7:

1. Rondas o *bazas* ganadas
2. Rondas o *bazas* perdidas
3. Apuesta declarada
4. **Ayuda:** Te da información de la situación actual y te da pistas
5. **Reiniciar manos:** Se vuelve a repartir las cartas seleccionando al azar todas las configuraciones preparadas
6. **Volver:** Botón para volver al menú de configuración en caso de querer cambiar las condiciones

A.3.1. Fase de subasta

Ilustración A.8: Interfaz de subasta

En esta fase se podrá observar las apuestas realizadas por la maquina y cuando sea tu turno se habilitaran los botones de las apuestas posibles a ejecutar. A.8

A.3.2. Fase de carteo

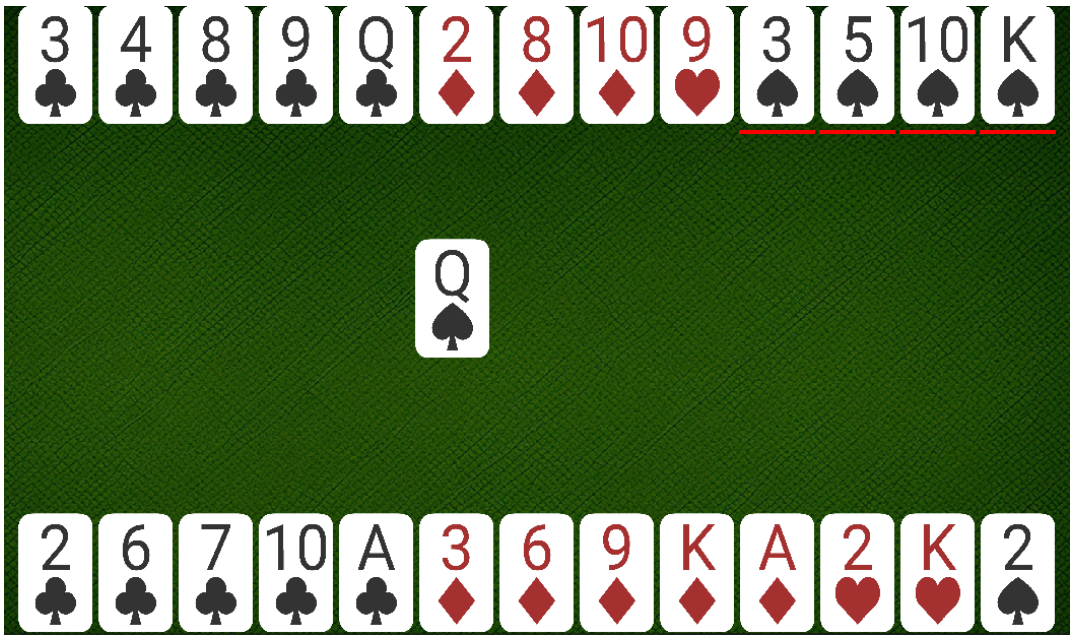

Ilustración A.9: Interfaz de carteo con subasta en Sur

Cuando se este jugando al carteo las únicas cartas posibles para jugar estarán indicadas con una linea roja por debajo, o por encima si son las cartas de NORTE, cuando sea el turno del jugador y haya ganado la apuesta, figura A.9, en caso de que haya ganado la apuesta este o oeste, el muerto se vera en el lado donde esta ubicado, figura A.10.

Ilustración A.10: Interfaz de carteo con subasta ganada en oeste

A.4. Enlaces

El código fuente del proyecto esta alojado en github.

Para probarlo la aplicación esta desplegada en itch.io en el que estará disponible para jugarlo en la propia pagina o se podrá descargar un ejecutable para Windows.

A.5. Ejecución del código

Una vez se tiene el código fuente el único requisito es descargarse Godot, para evitar problemas de compatibilidad con otras versiones se recomienda la 4.2.2.

Se conseguirá un ejecutable portable al que cuando se abra le aparecerá una pantalla de selector de proyectos vacía, figura A.11.

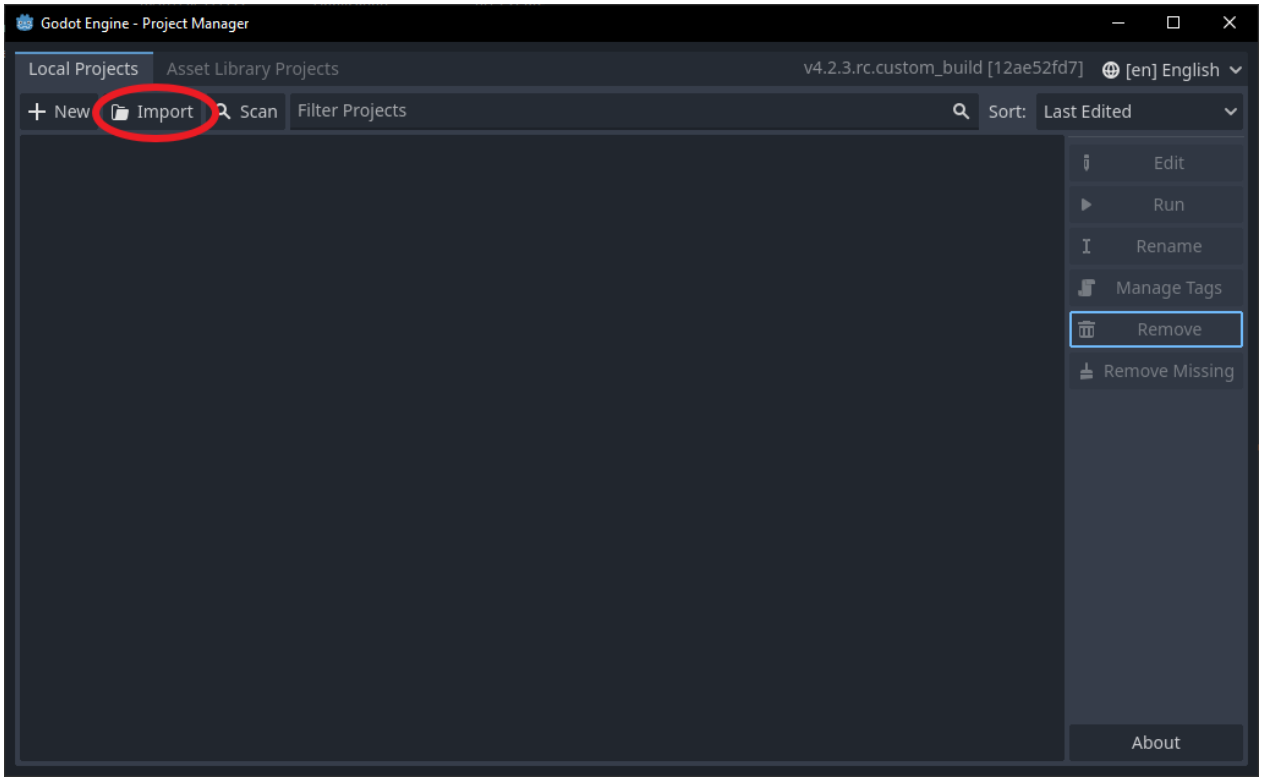

Ilustración A.11: Ventana de selector de proyectos

Se le da al botón de importar y se selecciona el archivo ".godot" dentro del proyecto descargado. Una vez cargado puede darle a la tecla "F5" para ejecutar el juego.

Bibliografía

[1] Tim Sweeney. Unreal engine, 1998. https://www.unrealengine.com/en-US.

[2] Juan Linietsky and Ariel Manzur. Godot engine, 2010. https://godotengine.org/.

[3] Unity Technologies. Unity engine, 2005. https://unity.com/.

[4] Orama interactive. Pixelorama, 2022. https://orama-interactive.itch.io/pixelorama.

[5] Jared Johnson. *Classic Bridge Quotes*. Devyn Pr, 1989.

[6] English Bridge Union. Origins and history of bridge, 2024. https://www.ebu.co.uk/origins-and-history-bridge.

[7] World Bridge Federation. Wbf history, 2024. http://www.worldbridge.org/the-wbf/wbf-history/#:~:text=The%20World%20Bridge%20Federation%20was,was%20elected%20as%20first%20President.

[8] Fred Gitelman. Bridge base online, 2010. https://www.bridgebase.com/.

[9] Eric Seidel. Bidding practice - sayc bridge, 2014. http://www.saycbridge.com/.

[10] GOTO Games. Play bridge - online, 2024. https://www.funbridge.com/.

[11] Juan Linietsky and Ariel Manzur. Godot engine manual, 2010. https://docs.godotengine.org/en/stable/index.html.

[12] NVidia. Nvidia physx, 2024. https://developer.nvidia.com/physx-sdk.

[13] Angel Mario Traversi. Generación automática de manos de bridge con restricciones. https://gestion.dc.uba.ar/media/academic/grade/thesis/traversi.pdf.

[14] Bo Haglund. Double dummy solver, 2018. https://privat.bahnhof.se/wb758135/bridge/index.html.

[15] Bo Haglund and Soren Hein. Search algorithms for a bridge double dummy solver, 2014. https://privat.bahnhof.se/wb758135/bridge/Alg-dds_x.pdf.

Glosario

baza Una baza consiste en las cuatro cartas, una por jugador, que se han jugado en una ronda. 1–3, 49

equilibrada Es una mano regular pero lo único que es diferente es si es la combinación 5332 y el palo que tiene 5 cartas es palo mayor **NO** se considera una mano equilibrada. . 38

fallar Cuando el jugador no tiene cartas del palo que se esta jugando la ronda. 2, 40

fit Se le llama fit cuando en, la fase de apuestas, se sabe por comunicación indirecta que entre las cartas de tu pareja y y las tuyas tenéis como mínimo 8 de un mismo palo. 53

impasse Movimiento con un 50 % de acierto en el que se juega de menor a mayor para obligar a la pareja contrincante a dar la ronda, en el caso que no tenga una carta de mayor valor, de una carta no asegurada. 44

muerto La pareja de la que gano la apuesta muestra las cartas a todos los demás jugadores y no podrá jugar las cartas por decisión propia, sino que jugará las cartas que le indique su compañero. iv, 33, 36, 41

palo mayor se le llama palo mayor al palo de corazones o de picas. 53

primera voz Jugador que empieza la declaración de la subasta. 32, 47

puntos de distribución Puntos que se asignan cuando se juega por palos y hay fit estos puntos puede ser por semifallo, dubletón, fallo o tener más de 8 cartas de un palo. 38, 47

puntos de honor Puntos que se asignan a las cartas para tener cuantificado cuanto de fuerte es tu carta. Los símbolos (basto, reina, rey y as) son los únicos que tienen puntos de honor, todos los demás tienen 0 puntos de honor. 32, 38, 47

red de pares Red de ordenadores comunicadas en la que no hay un servidor por defecto y cualquier cliente puede funcionar como uno. 44

regular Cuando una mano tiene una cantidad de cartas por palos que cumpla alguna de estas distribuciones:

- 4333
- 5332
- 4432

. 38, 53

sin triunfo Apuesta en la que ningún palo tiene preferencia por defecto. El palo de la primera carta que se juegue sera la que tenga el mayor valor en esa ronda. IV, 32, 38, 39