Proyecto 2: Blackjack

Juan M. Alberola, Jose I. Pastor

jalberola@dsic.upv.es

Grado en Tecnologías Interactivas

2 sesiones

Introducción

El **Blackjack** es un juego de cartas con un funcionamiento bastante simple pero que a su vez, cuenta con multitud de variantes dependiendo de la zona en donde se juegue. El objetivo del juego es conseguir la máxima puntuación posible sin pasarse de 21 puntos. Para el objetivo de este proyecto, seguiremos las indicaciones que se detallan a continuación.

En este juego, utilizaremos una baraja francesa de 52 cartas, que está formada por 4 palos (tréboles, corazones, picas y rombos) con 13 cartas cada uno. Las cartas del 2 al 10, tienen una puntuación equivalente al valor de su carta. Las cartas de figuras (J, Q, K), tienen una puntuación de 10 puntos. Finalmente, los ases pueden puntuar como 1 o como 11, dependiendo de si se pasa de 21 puntos o no.

El juego empieza con el croupier (**dealer**) repartiendo dos cartas al jugador y otras dos para él mismo. El jugador conocerá el valor de sus dos cartas, pero sólo tendrá descartada una de las dos cartas del dealer. A continuación, el jugador podrá plantarse o pedir cartas de una en una, teniendo en cuenta que si recibe una carta que hace que el valor de su mano sea mayor que 21, el jugador habrá perdido.

Cuando el jugador se planta, el dealer debe seguir unas reglas fijas:

- Está obligado a pedir una carta mientras tenga una puntuación de 16 o inferior.
- Está obligado a plantarse si tiene una puntuación de 17 o superior.

Además de esto, hay una situación especial que define la mejor mano, y es obtener 21 puntos en las dos primeras cartas. Esta jugada, se conoce como **Blackjack**, y gana la partida el que lo ha obtenido. Teniendo en cuenta todo esto, podemos resumir la dinámica del juego de la siguiente manera:

1. Se reparten dos cartas al jugador y otras dos al dealer, de las cuáles, una está descartada.
2. Si alguno obtiene un Blackjack en estas primeras dos cartas, gana la partida.
3. Si no hay Blackjack, el jugador puede pedir cartas de una en una o plantarse.

4. Si el jugador supera los 21 puntos, el jugador pierde.
5. Si el jugador se planta, empieza el turno del dealer, que se regirá por las dos reglas fijas de arriba.
6. Si el dealer se pasa de 21 puntos, el jugador gana.
7. Si el dealer se planta, gana la partida el que tenga una puntuación mayor.
8. En caso de igual puntuación entre el dealer y el jugador, hay un empate.

Descripción del proyecto

En este proyecto, deberás implementar un juego con **Unity** a partir del código inicial que se proporciona en el siguiente repositorio:

https://gitlab.com/Alberola/blackjack.git

La parte gráfica del juego ya está proporcionada, pero tendrás que implementar los algoritmos que definen la dinámica del juego así como las probabilidades de obtener algunas manos.

Como puedes ver, este proyecto consta de diferentes assets, que pasamos a detallar a continuación.

Objetos del juego

Los objetos del juego más relevantes están representados por:

- **Card**: hace referencia a una carta individual. Este objeto se encuentra en la carpeta **Prefabs**.
- **Dealer**: hace referencia al crupier.
- **Player**: hace referencia al jugador.
- **GameController**: se encarga de gestionar la dinámica del juego.
- **Canvas**: contiene los 3 botones del juego y los mensajes de texto que se muestran.

Además, tenemos tres clases principales dentro de la carpeta **scripts** que se utilizan en el proyecto. A continuación, comentamos los detalles.

Clase CardModel

Esta clase representa un objeto de tipo carta. Cada carta, tendrá una imagen frontal, una trasera y un valor asociado. Esta clase, ya está implementada y no es necesario realizar ningún cambio.

Clase CardHand

Esta clase representa las manos del dealer y del jugador. Como puedes observar, los objetos del juego **Player** y **Dealer** tienen asociado este script. Los atributos que se definen son los siguientes:

- **cards**: listado de cartas que tiene en ese momento.
- **card**: para poder instanciar objetos del juego de tipo carta. Este atributo tiene asignado un objeto **Card**.
- **isDealer**: para identificar quién es el dealer.
- **points**: para llevar un recuento de los puntos.
- **coordY**: para posicionar las cartas en el tablero.

Los métodos de esta clase están completos. Cabe mencionar el método **Push**, que se encarga de realizar la acción de reparto de una carta. Para ello, se crea un nuevo objeto del juego de tipo **Card**, se añade al listado y se calcula la nueva puntuación.
Esta clase, ya está implementada y no es necesario realizar ningún cambio.

Clase Deck

Esta clase representa la baraja de cartas y el control del juego. Como puedes observar, el objeto **GameController** tiene asociado este script. Los atributos que se definen son los siguientes:

- **faces**: listado de las 52 imágenes de las cartas. Este atributo ya está asignado por defecto.
- **dealer**: referencia al objeto del juego **Dealer**. Este atributo ya está asignado por defecto.
- **player**: referencia al objeto del juego **Player**. Este atributo ya está asignado por defecto.
- **hitButton**, **stickButton**, **playAgainButton**: referencia a los objetos del juego que representan los botones. Estos atributos ya están asignados por defecto.
- **finalMessage**, **probMessage**: referencia a los objetos del juego que representan el mensaje final y el de probabilidades. Estos atributos ya están asignados por defecto.
- **values**: para asignar un valor a cada una de las 52 cartas.
- **cardIndex**: para mantener el contador de las cartas repartidas.

Además, se definen los siguientes métodos que representan la dinámica del juego:

- **InitCardValues**. Este método se encarga de inicializar los valores de las 52 cartas. En principio, la posición de los valores se deberán corresponder con la posición de faces. En este sentido, si la carta de la posición 1 de faces es el 2 de corazones, el valor de la posición 1 debería ser un 2.
- **ShuffleCards**. Este método se encarga de barajar las cartas de manera aleatoria.
- **StartGame**. Este método se encarga de repartir las dos primeras manos.

- **CalculateProbabilities**. Este método se encarga de calcular varias probabilidades en función de las cartas que ya hay sobre la mesa.
- **PushDealer**. Este método se encarga de repartir una carta al Dealer.
- **PushPlayer**. Este método se encarga de repartir una carta al Dealer.
- **Hit**. Este método se asocia al botón **Hit** e implementa la lógica de pedir carta.
- **Stand**. Este método se asocia al botón **Stand** e implementa la lógica de plantarse.
- **PlayAgain**. Este método se asocia al botón **PlayAgain** y se encarga de inicializar los valores para volver a jugar.

En esta clase, deberás implementar los algoritmos que se indican en el código. Puedes definir más atributos y métodos si los necesitas.

Parte opcional para llegar a la máxima nota

Para llegar a la máxima nota, deberás incluir una funcionalidad para cumplir los siguientes puntos:

- El usuario parte de una banca de 1000 euros.
- El usuario puede hacer una apuesta inicial de múltiplos de 10 mientras tenga banca disponible.
- Si gana, obtendrá el doble del dinero de su apuesta y si pierde, perderá el dinero de su apuesta.
- Se mostrará en todo momento, la banca que tiene el jugador.

Entrega y exposición

Se realizará la entrega del proyecto en la plataforma dentro de la fecha límite. Además, se tendrá que mostrar el correcto funcionamiento en la sesión indicada por el profesor, el cuál realizará preguntas y podrá proponer modificaciones para comprobar que se entiende.

Será requisito también, enseñar el repositorio local de Git con almenos, los siguientes commits:

- Implementado método InitCardValues.
- Implementado método ShuffleCards.
- Implementado método CalculateProbabilities.
- Implementado método Hit.
- Implementado método Stand.

Criterios de evaluación

- Funcionamiento completo incluyendo opcional: 10.
- Funcionamiento completo sin incluir opcional: 8.
- Conoce y expone los pasos realizados de una forma clara y estructurada: sin penalización.
- Conoce los pasos pero su exposición no es clara ni estructurada: hasta 2 puntos menos.
- Hay partes que no entiende o no explica correctamente: hasta 5 puntos menos.
- No expone: entre 5 y 10 puntos menos.
- Realiza modificaciones con seguridad y claridad: sin penalización.
- Tiene dudas con algunas modificaciones: hasta 2 puntos menos.
- Tiene dudas con todas las modificaciones planteadas: hasta 5 puntos menos.
- Hay algún error leve: entre 0 y 2 puntos menos.
- Hay algún error grave: entre 1 y 5 puntos puntos.
- No se expone ni se entrega: 0 puntos.
- Cualquier modificación de estos criterios, se informará con suficiente antelación