Introducción a la creación de videojuegos

Consideraciones IMPORTANTES:

1. El proyecto se entregará vía GitHub: https://classroom.github.com/a/_fv7OOEK
2. La gestión de versiones es **obligatoria para que el proyecto sea evaluado**:
    a. Debemos ver varios commits para comprobar la evolución del proyecto.
    b. Proyectos realizados por arte de magia, con un único commit final, varios commits con el mismo contenido, pocos commits o commits en un espacio muy breve de tiempo, se entenderán como no realizados por el alumno.
    c. Si no se cumple este requisito, la calificación del proyecto es de cero (0) puntos.
3. La defensa del proyecto es obligatoria. Si no se realiza la defensa, la calificación del proyecto es de cero (0) puntos.

Descripción del Proyecto:

En la empresa de diseño web en la que hemos entrado a trabajar no paran de innovar e intentar mejorar su propia página web para impresionar a sus clientes. Esta vez han pensado añadir un juego de introducción a Blackjack usando Javascript. Tú serás la persona encargada de realizar dicho proyecto. Además de a ti, han contratado a otra persona para hacer otro juego al mismo tiempo que tú, pero parece que no le gusta demasiado la programación y la empresa, que confía plenamente en tus asombrosas habilidades, ha optado por que una vez terminada tu parte revises el código de su juego para mejorarlo y asegurarte de que funciona antes de poder usarlo en la web.

Tarea inicial:

1. **Elaboración de un diagrama de flujo:** En este proyecto se realizará un diagrama de flujo de una versión introductoria del juego **Blackjack,** al que llamaremos **Blackjack simplificado.** Es decir, una versión simplificada del juego para sólo 1 jugador contra la máquina (crupier) en el que no se tendrá en cuenta que la jugada que consigue 21 puntos con solo dos cartas (este movimiento se conoce como Blackjack) gana sobre cualquier otra jugada que consiga 21 puntos

de otro modo. Es decir, si el jugador y el crupier consiguen 21 puntos, gana el jugador, no importa con qué movimiento se haya hecho. El resto de reglas permanecen igual. En este diagrama debes contemplar todo el flujo de trabajo del juego, reflejando las interacciones del usuario, los procesos internos del juego, las respuestas que se generan al jugador, las posibles decisiones que se pueden tomar, etc. Debe ser predecido y perfectamente claro el comportamiento del juego en cualquier momento y reflejar todos los posibles flujos de trabajo.

2. **Uso de ramas en nuestro repositorio:**
    - **a.** Durante el desarrollo de todo el proyecto (especialmente durante los posteriores incrementos) se deberán usar diferentes ramas. De este modo, en la rama principal siempre tendremos una versión "funcional" con cosas finalizadas y para ir avanzando por los distintos apartados del proyecto usaremos ramas adicionales que crearemos nosotros, por ejemplo "test", "features" o "develop".
    - **b.** Podéis crear las ramas que consideréis oportunas y asignarles los nombres que queráis, pero en la rama "main" no debería haber ejercicios a medias.
    - **c.** Cuando llegue el momento, empuja al remoto todas las ramas para poder ver todo tu trabajo. No pasa nada si hay ejercicios incompletos o erróneos en las ramas que no sean la principal, justamente es una de sus utilidades, pero en la rama principal deberá haber sólo cosas terminadas.
3. **Generación de documentación:**
    - **a.** Utiliza el documento README.md para realizar una documentación mediante Markdown. En este documento debes reflejar los cambios de los commits más importantes (no es necesario reflejar el 100%, busca un equilibrio entre commits frecuentes y commit tras cada línea de código).
    - **b.** También debes explicar las distintas ramas que vayas a crear y cuál es el propósito de cada una, o dicho de otro modo, para qué las creas y por qué es útil tenerlas.
    - **c.** Cuando generes tu propio código más adelante, puedes añadirlo al documento y si no viene comentado, deberías explicar sus partes en esta documentación. En caso de que ya lo hayas comentado en el propio código, puede valer con simplemente añadirlo a la documentación. Igual

en el caso de la modificación del código proporcionado al final del proyecto.

d. En la documentación puedes explicar las funciones más complejas que hayas trabajado, si no lo has hecho con los comentarios, y explicar qué parámetros reciben esas funciones, qué valores devuelves, un poco el funcionamiento interno en general de cada función, etc.

Primer incremento:

4. **Programación del juego del Blackjack simplificado:** Apoyándote en el diagrama como base y punto de partida, se deberá programar dicho juego y comprobar su correcto funcionamiento. Para programarlo usarás el lenguaje de programación que has estudiado en programación. Puedes ayudarte usando tutoriales o cualquier otra herramienta que consideres, pero recuerda que tendrás que hacer el juego tú mismo y defenderlo más adelantes, además debe reflejar el flujo de trabajo que diseñaste en tu diagrama.

Segundo incremento:

5. **Mejora y optimización de código:**

    a. Se te proporciona el código bastante mejorable de otro juego, el cual tienes que corregir para que funcione y optimizarlo. Presta atención: que funcione no significa que sea correcto. Debes comprobar que el juego hace lo que debe hacer correctamente.

    b. Aplica todas las mejoras posibles de las vistas en clase: uso de nombres de variables intuitivos, uso de las estructuras de control más adecuadas, dar retroalimentación al usuario, uso de la programación modular (funciones) para simplificar el código, etc.