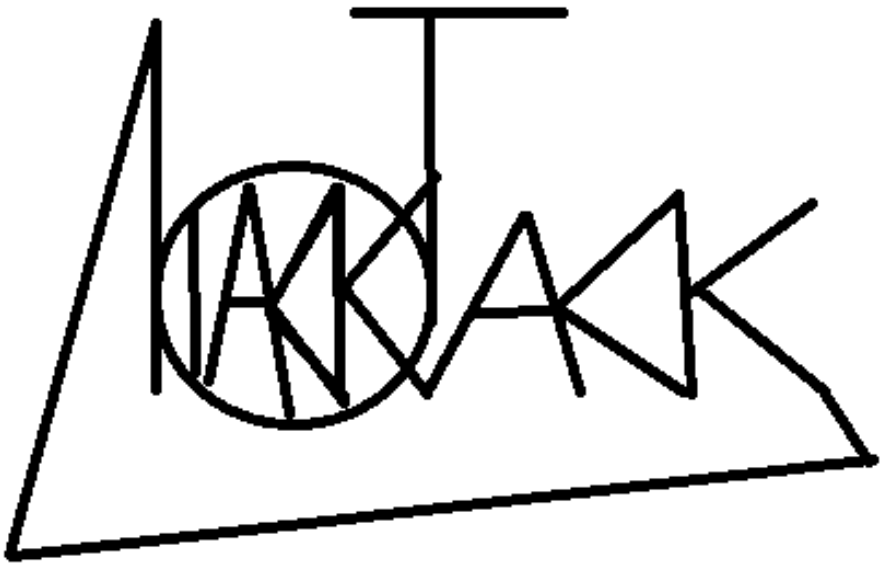
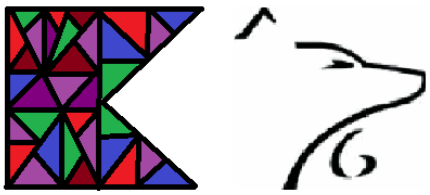
Casino Coyote



Manual técnico



#include <coyotes.c>

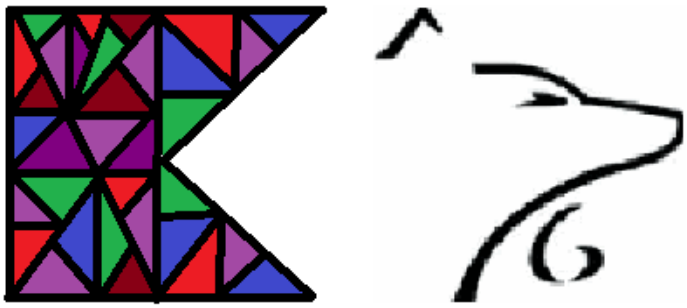
Índice

Introducción

¡Hola, estimado lector!, en este manual hablaremos acerca de cómo desarrollamos nuestro juego, qué obstáculos tuvimos, cómo ideamos las soluciones y cómo las implementamos al juego. Estamos muy emocionados por haber logrado el reto de crear este juego pues fueron muchas horas de dedicación y estudio para poder lograrlo, inclusive en cierto punto, fue hasta un sacrificio físico, como más adelante lo explicaremos.

Esperamos que este manual sirva para despejar tus dudas y que incluso, si lo deseas, puedas llevar acabo modificaciones de nuestro juego a tu gusto ☺.

Y una cosa más, agradecemos la dedicación si analizas todo el código, pues al tener alrededor de 2820 líneas hace que sea una tarea bastante pesada, también por ello este manual te ayudará con ello.



#include <coyotes.c>

Definición y análisis del problema

La idea de cómo llevar acabo nuestro juego surgió desde el primer día que salió la convocatoria del interprepas y algo que teníamos muy claro era que queríamos hacer algo diferente, que no solamente fuera un juego de Blackjack normal, de ahí nos nació la idea de llevar acabo un juego donde tuviera un modo de historia en el cual el usuario tuviera que ir moviéndose por las prepas de la UNAM, pues nosotros además de desarrollar un juego, queríamos sentirnos identificados con él, que nuestros compañeros también lo estuvieran y ,si les enseñábamos el juego, les diera curiosidad por probarlo. Así surgió nuestra idea de crear este juego.

Pero, desde el primer día tuvimos varias inquietudes, ¿cómo lograríamos que el juego fuese vistoso en un lenguaje de programación de hace más de 30 años que no está orientado al desarrollo de algún videojuego?, ¿cómo obtendríamos las cartas?, ¿cómo lograríamos conseguir unir tantas reglas?, ¿cómo podríamos llevarlo a cabo si hasta ese momento, sólo llevábamos 3 meses programando?.

Podríamos llevarnos cientos de cuartillas hablando de los problemas que nos surgían pero ese no es el objetivo de este manual pues en vez de explicar el código, inclusive podría confundir al lector. Por ello describiremos los 7 problemas que más no fue difícil resolver en el desarrollo de este juego.

1- **Dedicarle tiempo diariamente para poder programar el juego, a pesar de las actividades diarias tanto fuera como dentro de la prepa**

   Quizás esto no es un problema de pensar algún algoritmo para solucionar algún problema en el juego, pero sí es un problema de pensar un algoritmo para la vida diaria, poder programar juntos nos fue complicado, sobre todo por las fechas y el tiempo, pero lo logramos y nos dimos cuenta que la programación siempre será mejor si se trabaja en equipo. Pues tus compañeros la gran mayoría de las veces tendrá una forma de ver los problemas distinta a la tuya, y al compartir ideas, eso hace que se pueda llegar a la solución más eficiente. Y esto sin duda alguna, no solamente pasa en la programación, si no en muchas otras áreas. Muchas veces el principal obstáculo para lograr las cosas somos nosotros mismos y no nos damos cuenta

2- **Desarrollar un algoritmo que cuando el usuario tuviera algún AS en su mano, éste fuera cambiando de valor en conveniencia del que juegue esa mano.**
   Esto fue un gran reto, pues había una gran cantidad de casos posibles y teníamos que pensar en cada uno de ellos, por ello en un cuaderno, hicimos todas las posibles jugadas que se nos ocurrieron, hasta que ya no supimos decir alguna más, después nos pusimos a programar cada caso y

conforme íbamos programando otras funciones y probando el juego, hasta que pudimos condicionar todas en su totalidad.

3- **Crear un algoritmo que generara las cartas del juego, de forma totalmente aleatoria y que nunca fallara**

Este reto nos tomó varios días superarlo, pues el juego al basarse en esto, no podría tener ningún error grave, pues sería difícil de solucionar una vez que se avanzará más en el juego, para resolverlo analizamos que el blackjack al jugarse con 6 o hasta 10 barajas, tiene una aleatoriedad bastante alta y que por ello, un usuario nunca estará seguro de que carta le tocará, pues la mano más grande que podría tener es de 21 ases, si se juega común mazo de 6 o más barajas.

4- **Generar un sistema correcto de apuesta, que fuera como la vida real, con fichas y que fuera cómodo para el usuario**

Este problema costó trabajo realizarlo por la gran cantidad de apuestas diferentes que se pueden hacer en blackjack, con diferentes pagas y que, además, fuera totalmente funcional, por ello nos dimos la tarea de pensar cómo es que se maneja un casino y nos dimos cuenta que si siempre apuesta la misma persona, lo único que varía es la mano, y como su nombre lo indica, sólo eran 3 necesarias 3 variables para poder realizar todas las apuestas.

5- **Desarrollar un algoritmo de movimiento para los mapas**

No teníamos idea de cómo llevarla a cabo, sobre todo porque el personaje tenía que desplegarnos un mensaje y el usuario tendría que ir a ciertas coordenadas, habría un gran marco de error, por ello ideamos crear una zona rectangular, que no es visible, pero delimita los movimientos del jugador y sabemos que en ella no hay obstáculos que afectaran gravemente al juego.

6- **Llevar acabo diseños de cada prepa sin conocerlas y con un lenguaje que sus gráficos no son los mejores**

Para poder dibujar las prepas con la mayor precisión posible, decidimos usar matrices multidimensionales las cuales se meten por archivos de texto y tuvieran caracteres, los cuales imprimieran distintos caracteres gráficos. Y para dibujarlas, buscamos videos e imágenes de ellas para escoger zonas representativas.

7- **Manejar un código de muchas líneas y no afectar funciones al continuar programando**

Esto nos pasó muchas veces y nos hizo entender la importancia de la claridad y el orden en la programación estructurada, la forma que lo solucionamos fue aprender de cada error que pasaba en estos casos para después no repetirlo.

Diseño e implementación de la solución.

Para el algoritmo de BlackJack creamos el siguiente resumen:

El juego de BlackJack se basa principalmente en las funciones de inicializar, DefineBaraja, ValidaciónAs, animación, dividir, Aseg, CrearCarta, AlgJueComp, AlgJue y Jugar.

Primeramente, la función de inicializar va a definir todos los valores de las estructuras cartas y de la estructura elementos en cero. A excepción del elemento ConteoCartas de la estructura Player. Este elemento propio del Player[0], Player[1], Player[2], los va a definir en función a cuál será el rango de cartas para cada usuario. El Player[0] tendrá del conteo 0 al 15, el Player[1] del 32 al 48 y el Player[2] va a tener del 16 al 31. La variable de ConteoCartas irá aumentando a lo largo de la ejecución del programa para cada jugador y será la que nos indicará que carta mostrar durante la partida. Esta parte se puede ver en la función de CrearCarta, que tiene como uno de sus parámetros de entrada a "numcarta" que representa el número de carta que se mostrará en el tablero, "jugador" que puede tomar el valor de 0 o 1 (0 es para indicar que es el usuario y 1 para indicar que es la computadora) y tiene como último parámetro de entrada "estructuraElementos" que puede tomar valor entre 0-2. Una vez inicializando las cartas, se van a definir sus elementos en la función de DefineBaraja, en donde se le otorgará a cada carta un símbolo aleatorio, un número aleatorio y un color aleatorio. También hay valores que se convertirán en verdadero si es que salen iguales las primeras dos cartas del usuario, que en este caso indicaría que se puede ejecutar la división. Por otra parte está la variable boolAseg que retornará su valor a verdadero si es que la primera carta de la computadora es A o tiene valor de 10. Después tenemos la función de ValidacionAs misma que se manda a llamar dentro de la función de CrearCarta. Cuando el símbolo aleatorio de la carta resultó "1", en la función de CrearCarta, para asignarle valor a la carta, tendrá que pasar por la función ValidacionAs misma que le retornará a esa carta "1" el valor de 1 u 11 según sea necesario. La función de ValidacionAs también torna positivo el valor de las variables BoRs, mismas que durante la ejecución del juego, al ser verdaderas, le indicarán al programa que

puede hacerle la resta de 10 a sumaNum(Esta acción la podemos interpretar como cambiarle el valor de una carta “A” que inicialmente tenía el valor de 11 a 1) .

Por otra parte, la función de CrearCarta va a mandar a desplegar la carta que el usuario esté solicitando. Esta función la va a crear desde cero con los elementos de la carta que se esté mandando a llamar: su símbolo, su color y su número aleatorio.

La función CrearCarta se mandará a llamar dentro de las funciones de AlgJue y AlgJueComp, mismas que controlan la forma en la que el usuario está jugando: pidiendo cartas bajo las condiciones del juego de BlackJack. Finalmente la función de jugar será aquella que resumirá todo el juego del usuario y de la computadora.

Resumen para la apuesta

Nuestra función de apuesta, tiene como parámetros de entrada la apuesta máxima y l apuesta mínima que se solicitara, decidimos meter un menú en el que el usuario tuviera que elegir entre cantidades predeterminadas, inicialmente solicitaba que el usuario tecleara la cantidad, pero desechamos la idea y quedo con un menú vistoso, en el cual el usuario sólo hace uso de 3 teclas para poder apostar, cada vez que se realiza una apuesta, el sistema guarda la apuesta en 3 variables diferentes, n1, n2 y n3. ‘n1’ es la variable que se usa cuando el usuario apuesta en la mano default, n2 sólo se usa en caso de que el usuario divida y n3 sólo si desea asegurar. Una vez terminados los juegos se ejecuta una función llamada dinero, la cual paga al usuario su ganancia o le devuelve su inversión si es que empata, un problema que nos surgió fue tener que pagar 3 a 2 cuando era blackjack, pero lo solucionamos multiplicando la variable int por 1.5 para que así la computadora de forma automática nos redondeara la cantidad deseada.

Resumen para desplazamiento de mapas

Para hacer claro su funcionamiento, describiremos cada función

void lectur(char nprepa[2]);

Esta función carga el archivo de la prepa seleccionada por el parámetro de la cadena y la imprime en pantalla. Funciona de la siguiente manera:

1) Une la cadena de nprea con la localización del archivo.
2) Después abre el archivo y lo coloca en la matriz tablero(la cual contiene los números que están el archivo)
3) Después a través de un ciclo for y un switch, a cada número almacenado en la matriz se le asigna una coordenada , un carácter , y un color.

Void prepa1 cualquier otra

Decidimos crear una función para cada prepa, para que nos cargara el mapa deseado de cada una.

Se comportan se manera muy similares , ya que primero se resetean las 5 cadenas globales(m1,m2,m3,m4,m5). En caso de que la función prepa9 sea llamada el bprepa9 pasa a ser verdadero. Después se asignan 5 mensajes a las cadenas globales (m1,m2,m3,m4,m5) mediante la función strcpy , y enseguida entra en un ciclo indefinido (do while). Éste clico permite que cuando el usuario salga del casino de una prepa lo retorne al mapa de esa prepa. El ciclo funciona de la siguiente manera:

primero limpia la pantalla, después imprime en pantalla el mapa de la prepa. Posteriormente manda a llamar la función monito(). Después la variable bool Comenzarajugar se cambia a verdadero e inmediantamente terminado eso manda el escenario y comienza el juego. Al finalizar, un condicional valida que tengas el dinero suficiente para permanecer en esa prepa. Cabe mencionar que en el caso de prepa 9, se agrega un condicional, mismo que permite que se juegue contra el tótem siempre y cuando se tenga el dinero para retarlo., de lo contrario te mandará al casino de prepa 9.

void monito(int xmin, int xmax, int ymin, int ymax, int xm1, int xm2, int xm3, int xm4, int xm5, int ym1, int ym2, int ym3, int ym4, int ym5, int color);

Ésta función se encarga de darle movilidad al personaje , desplegar todos los mensajes de los estudiantes de las prepas y también permite acceder al casino de cada prepa .

Los primeros 4 parámetros son los responsables de delimitar el área de movimiento del personaje principal. Para el movimiento, se usó un ciclo indefinido el cual se rompe al encontrar al estudiante con el pase y acceder al casino. El ciclo siempre lee un carácter a través de la función getch, la cual se utiliza en diversas condicionales para sumar o restar las coordenadas Y o X. Posteriormente imprimir al personaje principal en esas coordenadas. Al inicio del ciclo, un for se encarga de borrar la impresión del muñeco con las coordenadas antiguas imprimiendo varios espacios con un color definido en el parámetro color

Conclusiones y comentarios

Al desarrollar este juego, pusimos a prueba nuestra capacidad de trabajo en equipo, responsabilidad y aprendizaje, pero aprendimos muchas cosas que pueden usarse en la vida diaria, que son muy valiosas, también hubo puntos negativos, pues somos conscientes de que varias cosas se podrían haber mejorado aún más, pero por falta de tiempo no lo llevamos a cabo. Otro punto que nos fue negativo es que al intentar ser minuciosos con los dibujos de varias prepas, nos lastimamos los ojos, pues el brillo de las pantallas nos lastimó, pero aprendimos de ello y nos dejó un aprendizaje, que al final eso es lo mejor, disfrutamos mucho hacer este juego y ganamos conocimientos que aplicaremos en nuestra vida diaria, para los 3 integrantes de este equipo, este ha sido el proyecto más largo que hemos hecho y estamos muy felices por el resultado, pues a pesar de que hace dos meses estábamos viendo arreglos, hoy ya los manejamos bastante bien. Tenemos la certeza de que nuestro juego dará una gran competencia.

Anexos

Es fundamental conocer las reglas del Blackjack para poder hablar de él, por ello aquí hay un enlace consultado el 5 de febrero del 2018 con las reglas del juego.

http://www.ete.enp.unam.mx/como_jugar_Blackjack.pdf

¡Gracias por su atención, para cualquier duda acerca del funcionamiento de nuestro juego, usted puede comunicarse vìa correo electrónico a la dirección:
marianosaleronarvaez@gmail.com