

Alumno: Barrera Jiménez Christian

Grupo: 1058

Documentación del proyecto entregado a candelario para la materia de Introducción a la Ingeniería en Computación.

Proyecto-Intro-ICO

Repositorio del proyecto final del Prof. Candelario subido a git hub, el cual consiste en la elaboración de un juego que permite al usuario interactuar con el juego, afectando las decisiones que toma está en la historia del juego.

Funcionamiento del Código

Menus

Iniciamos con la funcion void mostrarMenu(), y como su mismo nombre lo indica funciona para poder imprimir en pantalla las opciones que dispone el usuario para moverse en el menu.

```
void mostrarMenu(){
    cout << "=== El Peregrinaje de Aztlan a Tenochtitlan ===" << endl << endl;
    cout << "1. Jugar " << endl;
    cout << "2. Instrucciones" << endl;
    cout << "3. Acerca de" << endl;
    cout << "4. Idioma" << endl;
    cout << "5. Salir" << endl;
}
```

Esta funcion tambien se encuentra para el idioma ingles void monstrarMenuIng(), ya que como se puede ver en el codigo anterior una de las opciones existentes es el cambio de idioma.

```
void monstrarMenuIng(){
    cout << "=== The Pilgrimage from Aztlan to Tenochtitlan ===" << endl << endl;
    cout << "1. Play " << endl;
    cout << "2. Instructions" << endl;
    cout << "3. About" << endl;
    cout << "4. Language" << endl;
    cout << "5. Exit" << endl;
}
```

Sistema de elección

Para la posibilidad de seleccionar una de las opciones del menu se utilizo la estructura `switch()` la cual nos permite el funcionamiento para la mayoría del proyecto, aqui podemos ver el funcionamiento de esta estructura para el menu. Para la validacion de datos del usuario se utilizo la estructura `do while` con la condicion de `while(opcion1>3|opcion1<1)`, la hace que el usuario tenga que elegir una opción valida o diferente a la que escogió, debido a que la jugabilidad del juego se basa en la selección de 3 opciones que se limitan a los valores numéricos del 1 al 3. En algunas partes del codigo de utilizan **dobles** `do while`, ya que algunas opciones ofrecidas al jugador no daban continuidad a la historia o tenian condicionales que si el usuario no cumplía no permitía el progreso del juego. Ejemplo del uso de estas estructuras en la opción Idioma del Menu:

```
do{
    system("cls");
    cout<<endl;
    cout<<"==== El idioma actual se encuentra en Espanol. Desea cambiarlo a
Ingles? ====\\n";
    cout<<"1. Si.\\n";
    cout<<"2. No.\\n";
    cout<<">> ";
    cin>>opcIdioma;

    switch(opcIdioma){
        case 1:
            cout<<"The language has been changed to English\\n\\n";
            system("pause");
            idioma=2;
            return main();

        case 2:
            return main();
    }
}while(opcIdioma<1|opcIdioma>2);
```

En este caso al haber solo dos opciones la condición es `while(opcIdioma<1|opcIdioma>2)`.

Cambio de Idioma

Para poder realizar el cambio de idioma en el juego simplemente se repitió el codigo existente y el texto que se imprimia en pantalla se tradijo a ingles, la estructura del juego en ingles es la misma que el que esta en Español y con sus mismas variables. Para poder identificar si el juego debía de cambiar de idioma se utilizó un `if` y la variable `idioma` el cual tenia dos valores **1=Español** y **2=Ingles**.

```
if(idioma==1){
```

```

    mostrarMenu();
    cout<<"\n---Selecciona una opcion";
    cout<<endl<<">> ";
    cin >> opcMenu;

    //Programa en Español
}

if(idioma==2){
    monstrarMenuIng();
    cout<<"\n---Select an option";
    cout<<endl<<">> ";
    cin >> opcMenu;

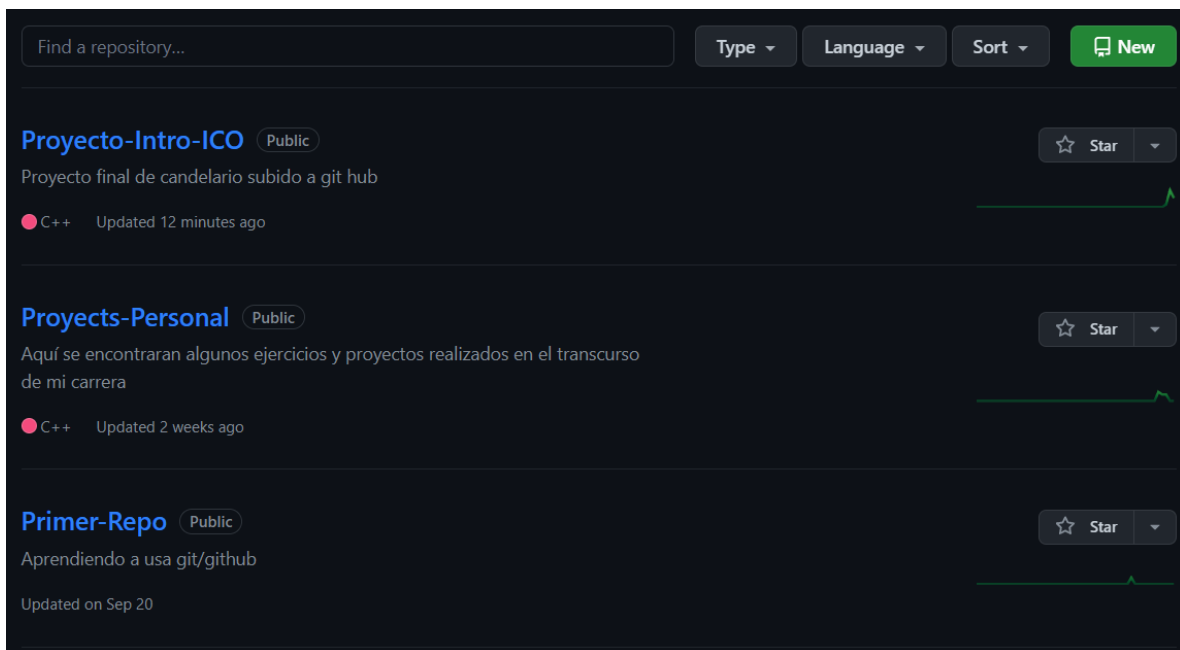
    //Programa en Ingles
}

```

Uso de Git Hub en el proyecto

Inicio del repositorio

Para el seguimiento del progreso del proyecto primero se inició un repositorio mediante Git Hub y para esto era necesario tener una cuenta creda y desde la misma pagina y en la sección de repositorios seleccioanr la opción **New**, la cual nos permite la creación de nuestro repositorio.



Clonación del repositorio

Después de la creación de repositorio es necesario clonar el repositorio en el dispositivo local y si es posible en una carpeta en específico para esto se utiliza el comando `git clone "Url del repositorio a clonar"`.

Actualización del proyecto

Git status

Al ya tener nuestro repositorio ya es posible empezar a trabajar el proyecto en un espacio donde se pueda llevar una gestión de progreso del proyecto. Cuando agregamos nuevos archivos o modificamos los existentes lo primero que podemos hacer para saber el estado del proyecto es utilizar el comando `git status` que nos permite visualizar el estado del proyecto desde nuestro dispositivo local, esto quiere decir que si realizamos algún cambio y no lo subimos este nos avisara o por el contrario si borramos un archivo y no subimos la actualización de esto a git hub tambien nos avisara.

Git add y Git commit

Cuando realizamos cambios en el repositorio es necesario subir estos cambios a Git Hub, ya sea que borremos, agregamos o modifiquemos un documento; y para esto debemos utilizar el comando `git add` que va a agregar los cambios que hayamos realizados, para que se puedan subir los cambios es necesario realizar un commit de los cambios realizados, esto quiere decir un comentario o descripción de que cambios realizamos en el repositorio para poder llevar un control mejor de los cambios que se realizan durante el trabajo en el repositorio, para esto debe utilizarse se utiliza el comando `git commit`.

Git push

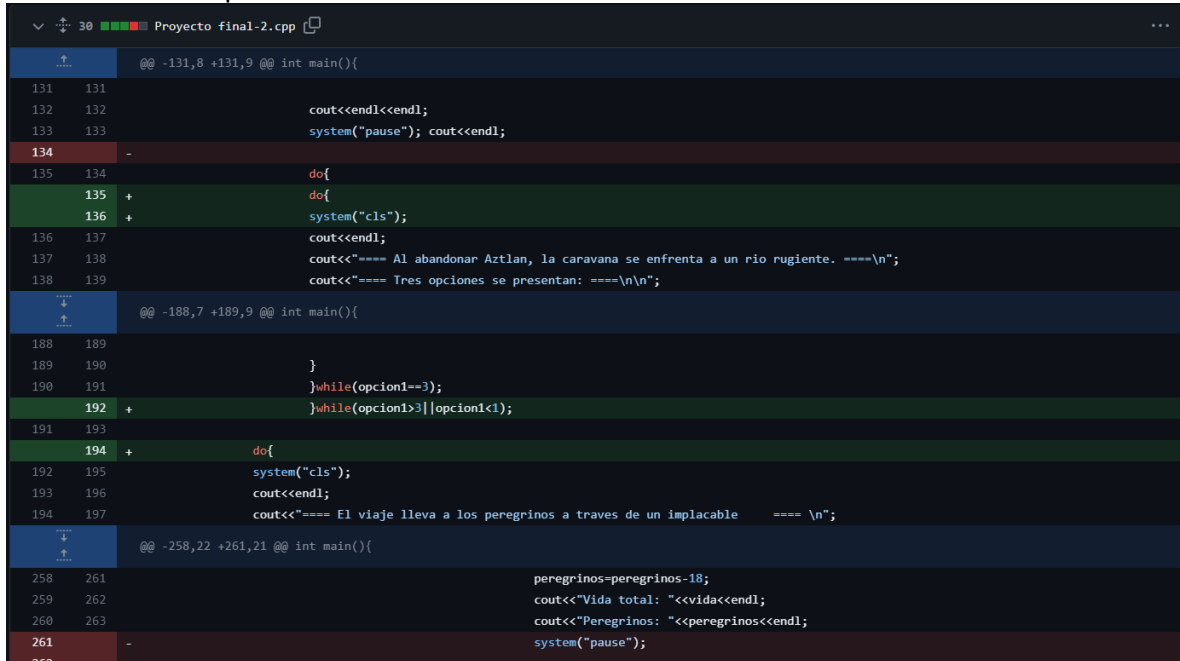
Finalmente para poder subir los cambios realizados se utiliza el comando `git push` que sube todos los archivos al repositorio el Git Hub.

Gestionar las versiones del proyecto

Para poder ver los cambios que se han realizado en el repositorio se debe utilizar el comando `git log` que nos permite ver los commits realizados junto consu

identificador y para poder el repositorio en esa versión debe utilizarse el comando `git checkout "identificador del commit"` para poder ir a esa versión. Aquí un ejemplo de su uso:

Validación de opciones: commit 7f00e31



```
Proyecto final-2.cpp
@@ -131,8 +131,9 @@ int main(){
131 131
132 132         cout<<endl<<endl;
133 133         system("pause"); cout<<endl;
134 -
135 134         do{
135 +         do{
136 +         system("cls");
136 137         cout<<endl;
137 138         cout<<"==== Al abandonar Aztlan, la caravana se enfrenta a un rio rugiente. ==== \n";
138 139         cout<<"==== Tres opciones se presentan: === \n \n";
@@ -188,7 +189,9 @@ int main(){
188 189
189 190         }
190 191         }while(opcion1==3);
191 192 +         }while(opcion1>3||opcion1<1);
191 193
192 194 +         do{
192 195         system("cls");
193 196         cout<<endl;
194 197         cout<<"==== El viaje lleva a los peregrinos a traves de un implacable ==== \n";
@@ -258,22 +261,21 @@ int main(){
258 261         peregrinos=peregrinos-18;
259 262         cout<<"Vida total: "<<vida<<endl;
260 263         cout<<"Peregrinos: "<<peregrinos<<endl;
261 -         system("pause");
262
```

Git pull

`Git pull` nos va a permitir descargar cambios que se hayan realizado en el repositorio pero no en nuestro dispositivo local para mantenernos al día con el estado del proyecto.