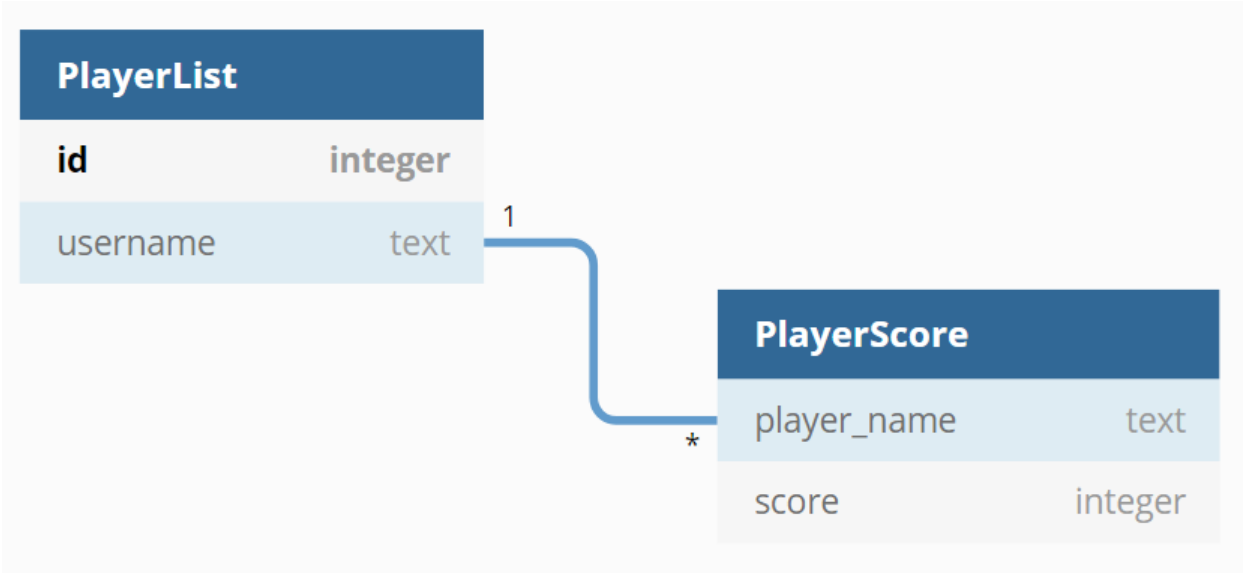
Contents

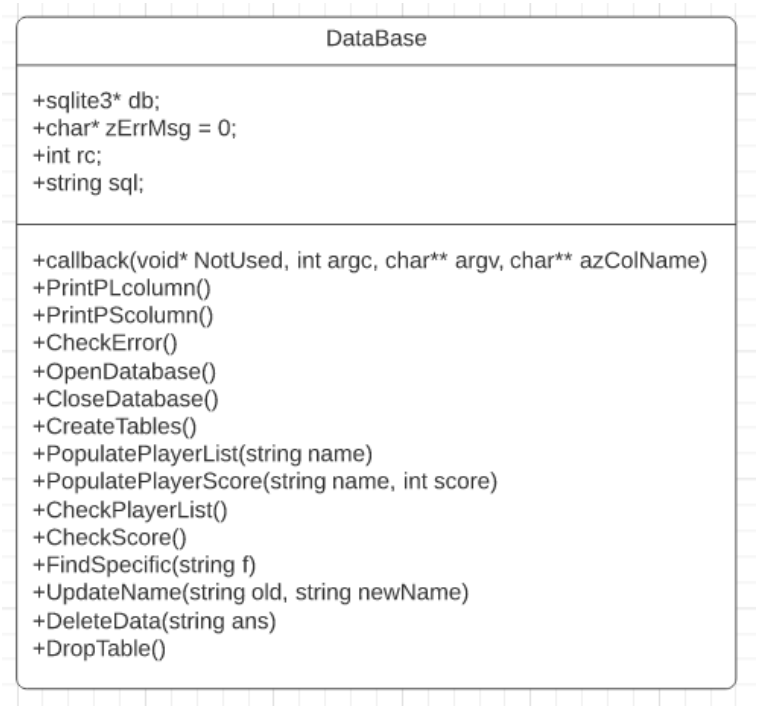
I. Overview

This database is integrated into a simple Snake Game using C++ . It's just a small game so there's not much data that need to process for database, so it's important to make sure there is no error and everything would work as intended.



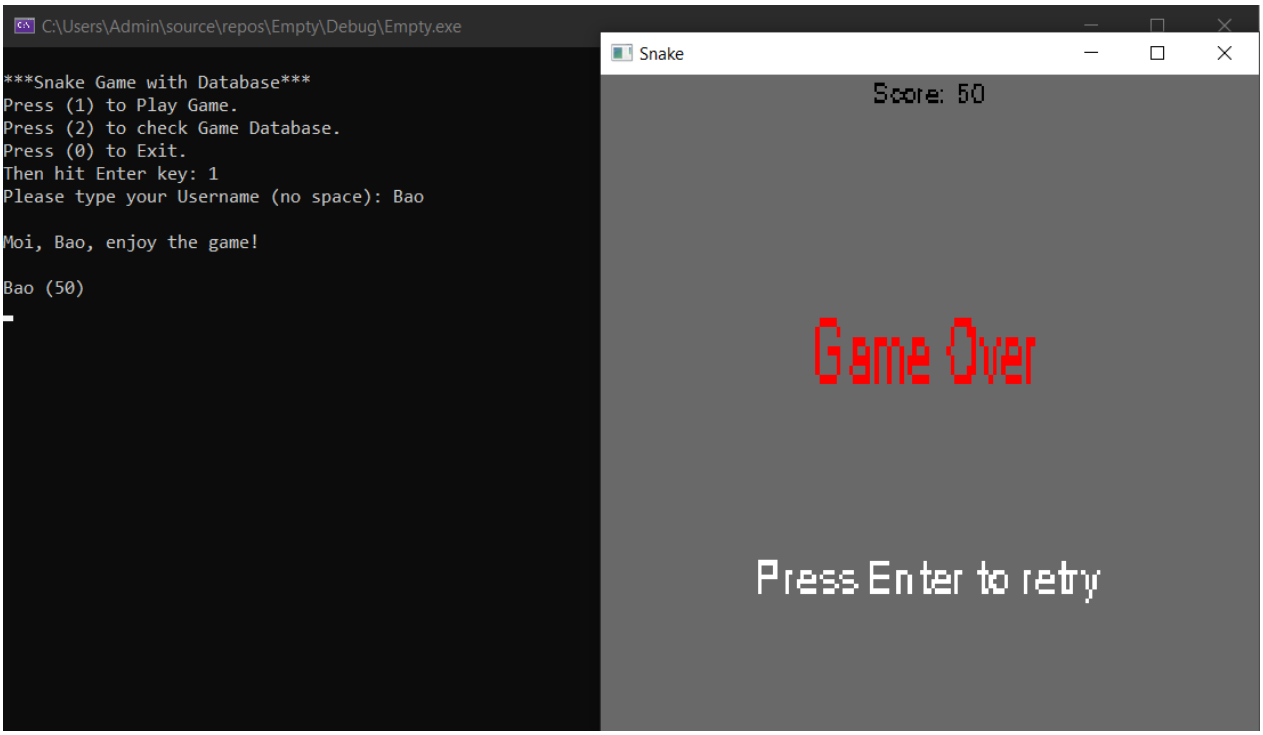
As a one-to-many relationship, the PlayerList table holds id as a primary key (also auto-increment), and the username for each id is unique. Next, the PlayerScore table can hold multiple value of score of each player.

The username in PlayerList also utilizes CASCADE, so each time there's an update or delete occurs with it, the player_name in PlayerScore table will perform the same.



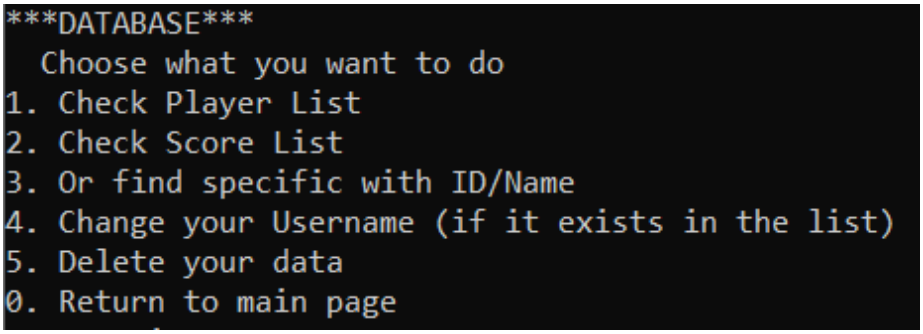
In DataBase class, there are 4 variables that mainly control the database. The sql variable contains the sql statement and uses sqlite3_exec to execute that with the callback function, which print out the table. I also modify this function combine with PrintPLcolumn and PrintPScolumn so it can print out a similar style to a data table with headers.

The OpenDatabase function will open the database called SnakeDatabase.db, and it turns on the foreign key to make sure CASCADE work properly. I call both OpenDatabase and CloseDatabase in almost every function because I usually got sqlite3 malloc error that I am not sure why, but this attempt seems to fix that. The CheckError function is also called with them to check error. The CreateTable function calls everytime the program starts.



When the player starts game, it will ask the username then firstly populate it in PlayerList table. Next, it will then populate the player_name and score to PlayerScore table when game is over.

Other results when accessing database:



1. Check Player List

```
Your option: 1

                ID            Username
        ---------------------------------------
                 1                 Bao
                 2              thuyet
```

2. Check Player Score (ascending)

```
Your option: 2

                ID         Player_Name              Score
        ------------------------------------------------------------
                 1                 Bao                  0
                 1                 Bao                 50
                 2              thuyet                  0
                 2              thuyet                 10
                 2              thuyet                 20
```

3. Find specific

```
What is your ID/Username: Bao
                ID         Player_Name              Score
        ------------------------------------------------------------
                 1                 Bao                  0
                 1                 Bao                 50
```

```
What is your ID/Username: 2
                ID         Player_Name              Score
        ------------------------------------------------------------
                 2              thuyet                  0
                 2              thuyet                 10
                 2              thuyet                 20
```

4. Change Username

```
Please type your ID or your old name: Bao
And your new name is: BaoNguyen

Changed successfully! Your new name is BaoNguyen
```

```
          ID          Player_Name                Score
 ------------------------------------------------------------
           1            BaoNguyen                   0
           1            BaoNguyen                  50
```

```
Please type your ID or your old name: Mai
And your new name is: Bao

YOUR ID OR USERNAME IS NOT EXIST!!!
```

5. Delete data or drop table

```
Please type the ID or Username you want to delete
Or type MOIMOI to wipe out everything: thuyet

Successfully rip your data :)
```

```
Please type the ID or Username you want to delete
Or type MOIMOI to wipe out everything: MOIMOI

Nice move, see you later!

C:\Users\Admin\source\repos\Empty\Debug\Empty.exe (process 24424) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

II. Database script

```
#pragma once
#include <iostream>
#include <string>
#include <stdio.h>
#include <sqlite3.h>
#include <iomanip>
#include <Windows.h>

using namespace std;

class DataBase {

        public:
        // Pointer to SQLite connection
        sqlite3* db;

        // Save any error messages
        char* zErrMsg = 0;

        // Save the result of opening the file
        int rc;

        // Save any SQL
        string sql;


            // Create a callback function to print out
            static int callback(void* NotUsed, int argc, char** argv, char** azColName) {
            // int argc: holds the number of results
            // (array) azColName: holds each column returned
            // (array) argv: holds each value

            for (int i = 0; i < argc; i++) {

                // Show column name, value, and newline
                cout << setw(20) << argv[i];

            }
            cout << endl;


            return 0;
            }

            // Print column names manually
            // Sorry I tried :(
            void PrintPLcolumn()
            {
            cout << setw(20) << "ID" << setw(20) << "Username" << endl;
            cout << "         ----------------------------------------" << endl;
            }

            void PrintPScolumn()
```

```
            {
            cout << setw(20) << "ID" << setw(20) << "Player_Name" << setw(20) << "Score" << endl;
            cout << "------------------------------------------------------------" << endl;
            }

                void CheckError()
                {
                if (rc != SQLITE_OK) {
                fprintf(stderr, "SQL: %s\n", zErrMsg);
                sqlite3_free(zErrMsg);
                }
                else {
                fprintf(stdout, "Successful\n");
                }
                }

            void OpenDatabase() {
            // Save the result of opening the file
            rc = sqlite3_open("SnakeDatabase.db", &db);

                if (rc) {
                // Show an error message
                cout << "DB Error: " << sqlite3_errmsg(db) << endl;
                }

            // Enable foreign key
            sql = "PRAGMA foreign_keys = ON;";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);

        }

            void CloseDatabase() {
            sqlite3_close(db);
            }

            // Create tables if not exist
            void CreateTables() {
            OpenDatabase();

            sql = "CREATE TABLE PlayerList (ID INTEGER PRIMARY KEY AUTOINCREMENT, Username TEXT NOT NULL UNIQUE);";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
            sql = "CREATE TABLE PlayerScore (Player_Name TEXT NOT NULL, Score INTEGER NOT NULL, FOREIGN KEY(Player_name) REFERENCES PlayerList (Username) ON UPDATE CASCADE ON DELETE CASCADE);";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);

            CloseDatabase();
            }

            // Populate when game start
            void PopulatePlayerList(string name) {
            cout << "\nMoi, " << name << ", enjoy the game!" <<endl;

```

```
OpenDatabase();

sql = "INSERT INTO PlayerList (Username) VALUES ('" + name + "');";
rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
//CheckError();

CloseDatabase();
}

// Populate when game over
void PopulatePlayerScore(string name, int score) {
cout << endl;
cout << name << " (" << score << ")" << endl;

OpenDatabase();

sql = "INSERT INTO PlayerScore (Player_Name, Score) VALUES ('" + name
  + "', " + to_string(score) + ");";
rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
//CheckError();

CloseDatabase();
}

void CheckPlayerList() {
PrintPLcolumn();

OpenDatabase();

sql = "SELECT * FROM PlayerList;";
rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
//CheckError();

CloseDatabase();
}

void CheckScore() {
PrintPScolumn();

OpenDatabase();

sql = "SELECT PlayerList.ID, PlayerScore.Player_Name,
  PlayerScore.Score FROM PlayerList, PlayerScore WHERE
  PlayerList.Username = PlayerScore.Player_Name ORDER BY
  PlayerList.ID, PlayerScore.Score;";
rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
//CheckError();

CloseDatabase();
}

void FindSpecific(string f) {
PrintPScolumn();

OpenDatabase();
```

```
        sql = "SELECT PlayerList.ID, PlayerScore.Player_Name, PlayerScore.Score FROM PlayerList, PlayerScore WHERE PlayerList.ID =
         '" + f + "' AND PlayerList.Username = PlayerScore.Player_Name ORDER  BY PlayerScore.Score;";
        rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
        //CheckError();

        sql = "SELECT PlayerList.ID, PlayerScore.Player_Name, PlayerScore.Score FROM PlayerList, PlayerScore WHERE PlayerList.Username = '" + f + "' AND PlayerList.Username = PlayerScore.Player_Name ORDER BY PlayerScore.Score;";
        rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
        //CheckError();

        CloseDatabase();
        }

        void UpdateName(string old, string newName) {
        OpenDatabase();

        // Priority for ID when an username is a number same as ID
        sql = "UPDATE PlayerList SET Username = '" + newName + "' WHERE ID = '" + old + "';";
        rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
        //CheckError();

            // If there's no row affected
            if (sqlite3_changes(db) == 0)
            {
            sql = "UPDATE PlayerList SET Username = '" + newName + "' WHERE
              Username = '" + old + "';";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
            //CheckError();

                if (sqlite3_changes(db) == 0)
                {
                cout << "\nYOUR ID OR USERNAME IS NOT EXIST!!!" << endl;
                }
                else
                {
                cout << "\nChanged successfully! Your new name is " << newName << endl;
                }
                }
                else
                {
                cout << "\nChanged successfully! Your new name is " << newName << endl;
                }

        CloseDatabase();
        }

        void DeleteData(string ans) {
```

```
            OpenDatabase();

            // Priority for ID, similar to Update
            sql = "DELETE FROM PlayerList WHERE ID = '" + ans + "';";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
            //CheckError();

                if (sqlite3_changes(db) == 0)
                {
                sql = "DELETE FROM PlayerList WHERE Username = '" + ans + "';";
                rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
                //CheckError();

                    if (sqlite3_changes(db) == 0)
                    {
                    cout << "\nYOUR ID OR USERNAME IS NOT EXIST!!!" << endl;
                    }
                    else
                    {
                    cout << "\nSuccessfully rip your data :)" << endl;
                    }
                    }
                    else
                    {
                    cout << "\nSuccessfully rip your data :)" << endl;
                    }

            CloseDatabase();
            }

            void DropTable() {
            OpenDatabase();

            sql = "DROP TABLE PlayerList; DROP TABLE PlayerScore;";
            rc = sqlite3_exec(db, sql.c_str(), callback, 0, &zErrMsg);
            //CheckError();

            cout << "\nNice move, see you later!" << endl;

            CloseDatabase();
            }
            };
```