

CS101 – Course Project

PROJECT REPORT

Quiz Game

Wednesday Batch
311

TA – Dinesh Patil

Team members

Alok Yadav (Team Leader)	110050043
Ameya Behere(Coordinator)	110010003
Abhilash Gupta	110050058
Nishanth Sai Anchala	110040098
Anirudh Krishnan	11D260008
Madhusudhan	110040103
Aniruddha Samantha	115090023

Introduction

Our project attempts to create a quiz game along the lines of the popular game show “Who Wants To Be A Millionaire”. The player answers multiple choice questions 4 options out of which only 1 is correct, up to a total of 12 questions. Graphic User Interface has been implemented using EZWindows.

The Questions

The questions are selected at random from a question database. The question database currently has 88 questions. Due to a segmentation fault which we were unable to pin point and fix within the given time, we decided to use the first 60 of the 88 questions. We had planned on categorizing the questions on the basis of their difficulty. However in the end we decided to go with a single level. There is no time limit for player to answer any question.

Question Selection

All the questions are maintained in a text file. This question bank is copied into objects of the type question. This is an abstract data type and has been defined in the program. Each question is associated to a single object. A sample question as stored in the text file is as follows -

36, Which is the most spoken language in the world?
A. Hindi
B. Mandarin
C. English
D. Spanish
B

A random picking function picks a question object, checks whether it is a “used” question or not, and then proceeds to display it on the screen along with the 4 options. Also once a question has been selected by the random question selector function, it flags the question as “used”. This flagging procedure is to ensure that not only the user gets the questions at random but also that no question is repeated. All flags are set to “not used” at the beginning of every execution of the program.

A integer array is maintained with all the values initialized to 0. When a question is used, the index corresponding to that question is set to 1. Checking if a question is used consists of checking if the index corresponding to it is 0 or 1. If the question is used, the question selection loop will restart.

Gameplay

On executing the program, the user enters their name in the terminal. A short welcome message is displayed. On pressing OK, the main menu screen is displayed. The game starts when the user selects the START button on the main menu. The user then answers the questions by clicking the option which he/she thinks is correct.

The screens

Main Menu : It is the welcome screen. All other screens are accessed from here. The options are

1. Start - Starts the game.
2. Instructions - Player can refer to this to learn how to play the game.
3. High Score - View the top 5 scores.
4. Credits
5. Exit

Main Gameplay screen : This is the main playing screen. This screen is accessed from the main menu. The questions and options are displayed on the left of the window. The score tree is shown on the right side. A lifeline button and quit button are also displayed at top right.

Congratulations : This is displayed after the Gameplay terminates either via a win or wrong answer. This screen shows the players name and score.

Instructions : Instructions screen. States the rules and format of the game. Explains the lifelines available and the scoring system.

High Scores : High Score screen. Shows the high scores of the top 5 players. This list is updated every time the game runs.

Credits : Shows the credits.

The Lifeline

In case the user is stuck at some question, he/she has the option of using a lifeline. We provide the lifeline 50-50. The user may use this lifeline only once . It is accessed by clicking on the 50-50 lifeline button displayed on the top right corner of the main Gameplay screen. After the user clicks on the button, any 2 wrong options are removed from the screen, leaving the user with only 2 options out of which 1 is correct.

The Timer (discontinued)

We had planned on including a timer in the game. However it was agreed that given the time constraint, it was best that we drop it.

Playing The Game

If a question is answered wrongly the main game loop is broken and the game over screen is displayed. If the question is answered correctly then the question selection and presenting loop will be repeated till the player answers a question wrongly or answers the 12th question correctly. The number of questions correctly answered is tracked using a count variable. The “Game Over” screen is displayed which displays the user name and score. A “Back to main menu” option is provided at most screens to restart the game.

Score Tree

A score tree is shown on the right side of the game screen, which tracks the players progress. The score achieved till current time is indicated by small green squares beside the score.

High Score

A binary files stores the high scores of top 5 users. This file is updated every time the player plays a game. Initially all names are set as “default” and score as 0.

Graphic User Interface and Requirements

This has been implemented using EZWindows. The user can use the mouse to give his response. For this, regions have been defined in the windows. Response given via mouse is recorded according to the option button on which the mouse-click occurred.

Details on how to run the program are included in the User Manual. The User Manual is present in the documentation folder present in the project root folder.

Changes from SRS

Due to time constraints and various other problems, we had to modify the structure of our program from time to time. Thus the game as described in the SRS submitted along with the stage 1 is slightly different from what we have finally submitted. The changes are as follows -

1. No timer feature.
2. No levels of difficulty of questions.
3. The money tree is replaced by a score tree. (Effectively what we earlier referred to as 'money' is now 'score'.)
4. We have made only 1 lifeline (50-50) which the user can use only once.
5. The name of the player is asked right at the start of execution instead of at the game over screen (as was planned earlier).
6. No “Game Won” screen. Even if a player answers all 12 questions correctly, he is shown the same game over screen.

Status of Completion/Known Errors

The project is working satisfactorily. A few minor glitches may be noted.

1. Display of a random symbol after a string.
2. On clicking the correct option, the program shows incorrect.
3. If the player loses on the first question itself, the score screen shows the name but nothing as the score(it should have shown zero).

Both these errors are due to the text file whose format is not completely correct.

Ideas for Future Versions

The following features will be included in the version 2 of this game. Given the time, we would have included these in our final submission

1. More questions.
2. A timer and time limit.
3. Levels of Difficulty.
4. Indicating the correct answer to the player if he/she is wrong.
5. More lifelines such as Skip a question/2 Attempts at a single question.
6. More visually pleasing screens.

Description of the Code

We have created header files that store the different functions required to run

the code. Our main .cpp file which is to be compiled and executed is the screen2.cpp. This program calls all the functions written in the header files. The header files are described as follows.

1. click.h – This header files deals with the main menu. It links the windows to the main menu.
2. click2.h – Takes input from the user during the game play. Works on the main playing window.
3. credits.h – Contains the window that displays the credits.
4. exit.h – If player clicks on the exit button on the main game play screen this takes the user back to main menu after displaying the user's score.
5. fifty_fifty.h – This is the header file containing the code for the fifty fifty lifeline that we have provided for the player.
6. highscore_screen.h – Deals with the High Score screen.
7. hi_score.h – Deals with the coding part of high score. File handling done in this file.
8. Instruction_screen.h – The instruction window code.
9. Kbc.h – The functions setquestion and selectquestion is called in this code.
10. Play.h – The playing screen code.
11. ques_class_definition.h – Contains the class definition and member functions of the class Question.
12. Screen2.cpp – main menu code.
13. select_ques.h – selects the question at random from the array.
14. setques.h – The code that reads the question database and feeds it into the array.
15. setscore.h – sets the user score according to the number of questions the player has answered correctly.

Contribution by Team Members

The work done by each team member is briefly described here. Their work has been included in their folder.

1. Alok Yadav (Team Leader) – As the team leader, called for meetings to be held. Also made decisions when there were multiple codes for the same function written by different team members. Wrote most of the Graphics part of the program along with Abhilash Gupta, Anirudhha Samantha. Input via mouse was done by him. Debugging of the program code. Merging of the elements to produce the final working program. Made the minor adjustments necessary in each team members code so that they are compatible with each other and hence could be joined.
2. Ameya Behere (Coordinator) – Responsible for arranging meetings of the team and overall team coordination and communication. Wrote most of the non Graphic part of the code. Also edited the written code to include indentation and improved overall program readability. Final Report, peer review, Manual written by him.
3. Abhilash Gupta – Was mainly involved with the graphics related part of the project. Designed bitmaps windows and created the mouse-user interface.
4. Nishanth Sai Anchala – Was responsible for the question database and writing the question feeding algorithm that feeds the database into the array of objects along with Madhusudhan. Also wrote a code that selects the random number according to which a question is displayed with Madhusudhan.
5. Madhusudhan - Was responsible for the question database and writing the question feeding algorithm that feeds the database into the array of objects along with Nishanth. Also wrote a code that selects the random number according to which a question is displayed with Nishanth.
6. Anirudh Krishnan – Also contributed to the question database. Also wrote stage 1 report that is SRS.
7. Aniruddha Samantha – Helped to write documentation. Also helped to create the exit and high_score window.

References

Class Notes

Lab handouts

Programming in c++ by Cohoon and Davidson

Object Oriented Programming with C++

Acknowledgement

Our project would not have been successfully completed on time had it not been for these people. We would like to thank them for their advice and suggestions.

Prof. D B Phatak, CS101 Course Instructor

Dinesh Patil, Our JTA

Fellow students