Typing Speed Game - Viva Questions & Answers

Q: What is your project about?

A: It is a Typing Speed Game where words fall from the top, and the player types them before they reach the bottom.

Q: How does the game work from start to end?

A: The player selects a mode, types words to gain score, loses health if missed, and the game ends when health reaches zero.

Q: What makes your version unique?

A: It includes power-ups, custom words, animations, and a high-score leaderboard.

Q: What are the main features of your game?

A: Random or custom words, health system, score tracking, power-ups, sound toggle, and high scores.

Q: How do you handle user input in the game?

A: Using kbhit() and getch() to detect and read keyboard input in real-time.

Q: What happens when a word reaches the bottom?

A: Health decreases by one, sound plays, and the word is removed.

Q: How are the high scores stored and displayed?

A: Stored in a text file and displayed at game over using graphics functions.

Q: How does the game detect correct typing?

A: It compares the typed string to active words and removes the match.

Q: How does the game end?

A: When health reaches zero, it shows the Game Over screen and leaderboard.

Q: Which library did you use for graphics? Why?

A: WinBGI graphics.h library because it is simple for 2D graphics in C++.

Q: Explain setactivepage() and setvisualpage().

A: They control double buffering: one page is drawn on while the other is displayed to avoid flicker.

Q: How do you spawn new words?

A: Every few frames, if active word count is below a limit, a random word is added.

Q: How is the falling speed of words controlled?

A: By the fallSpeed variable, which increases as score increases.

Q: What is the purpose of countMatchingPrefix()?

A: To highlight correctly typed letters in a falling word.

Q: How do you erase a correctly typed word?

A: It is removed from the vector and score is updated.

Q: How are power-ups implemented?

A: After a streak of 10 correct words, a random power-up is activated for a few seconds.

Q: How do you toggle sound on/off?

A: Pressing F1 changes a soundEnabled boolean and stops/plays music accordingly.

Q: What is the role of handleTyping()?

A: It processes key presses, updates typed text, and checks for matches or mistakes.

Q: How do you load words from a file?

A: Using ifstream to read each line into a vector<string> dictionary.

Q: Where is the high score file stored and in what format?

A: In file/highscore.txt as name-score pairs.

Q: How do you save high scores?

A: Using ofstream to write name-score pairs back to the file.

Q: How does the custom word feature work?

A: The player types a sentence, which is split into words and saved to a file for gameplay.

Q: How do you draw shapes in your game?

A: Using custom Midpoint Circle, Midpoint Ellipse, and Bresenham's Line algorithms.

Q: Why use Bresenham's line algorithm?

A: It is faster and uses only integer calculations, avoiding floating point.

Q: Why use vector instead of array?

A: Vector resizes dynamically and is easier for adding/removing words.

Q: What is struct and why used?

A: A struct groups related variables; used to store word properties like text, position, color.

Q: Difference between struct and class?

A: Struct members are public by default, class members are private by default.

Q: Why srand(time(0)) is used?

A: To seed rand() with the current time for different random sequences each run.

Q: Why const in function parameters?

A: To prevent modification of arguments and allow passing temporary objects safely.

Q: How is clock() used in your game?

A: For measuring durations of power-ups and animations.

Q: Why check if(kbhit()) getch()?

A: To check for and process key presses without blocking the game loop.

Q: What are limitations of your game?

A: Dependent on WinBGI, not optimized for high resolutions, single-player only.

Q: If given more time, what improvements would you add?

A: Better graphics library, multiplayer mode, more animations.

Q: Could this game run on SDL/SFML?

A: Yes, but graphics and input handling code would need rewriting.

Q: How would you make this game multiplayer?

A: Add network or split-screen input handling and shared game state.

Q: Did you face bugs/freezes?

A: Yes, screen freeze due to graphics library message handling; fixed by processing events.

Q: How to prevent lag?

A: Limit active objects, use efficient drawing algorithms, and process events regularly.

Q: What if word file is missing?

A: Game would load an empty dictionary; should handle by showing an error message.