

Space Type: A Retro Themed Typing Trainer

Mukunda Dev Adhikari^{#1}, Praharsha Adhikari^{#2}, Pragalbha Acharya^{#3}

Department of Electronics and Computer Engineering, Pulchowk Engineering Campus

Pulchowk, Lalitpur, Nepal

¹078bct049.mukunda@pcampus.edu.np

²078bct061.praharsha@pcampus.edu.np

³078bct060.pragalbha@pcampus.edu.np

Abstract— This project report outlines the approach and procedure taken while creating Space Type, a typing trainer, and a game in the design of a retro-era space game. Our goal with this was to help boost our typing speed of ourselves and our classmates, which would result in our efficiency in writing code. Space Type is a comprehensive package of various word and letter practice modes, several text passages, and a fun interactive game to test our typing speed. The retro graphics were achieved with the help of Raylib Framework, and the application is written entirely in C programming language.

Keywords— Coding efficiency, Retro, Raylib

NOMENCLATURE

wpm	Words per minute
GUI	Graphical User Interface

I. INTRODUCTION

In this current age where digital tools are actively replacing pen and paper, a good typing speed results in increased efficiency and improved workflow for any working individual. Space Type helps the cause; it is a one-stop solution to help you boost your typing speed.

Space type allows the user to practice typing in a series of different ways. They can either only practice letters of specific rows or letters of multiple rows or all letters entirely. They can also select the option to practice words and choose the keyboard rows for those words. They can also practice the letters that they had the most difficulty typing. They can test how much they have improved over a wide range of text passages of varying difficulty. If they are confident about their typing speed and feel competitive, the game mode is ready to accept their competition.

A typing trainer is nothing new. It's been there since the advent of personal computers and has been continuously improving to help people increase their typing speed. But

most of them have plain graphics. Since people's attention span has been lowering due to the saturation of applications, they aren't appealing to the audience. Our application tackles that problem by disguising itself with retro game-Esque graphics and having engaging background music.

II. OBJECTIVES

The main objectives of our project are:

- To provide a fun learning experience for typing
- To present the users the option to choose the specific area they want to focus their typing practice on
- To help boost the WPM of the users
- To make the user more confident in typing.
- To help make the users' workflow more efficient.

III. MOTIVATION AND SIGNIFICANCE

Higher typing speed reduces cognitive load by allowing you to type at the speed of your thoughts. Research has shown that touch typing improves the user's cognitive function. Touch typing is a mental activity that engages most parts of a person's brain. It helps activate new muscle memory and build more active and strong cognitive connections. This, in turn, will enhance a person's overall brain capacity and function. But then we noticed during the short sessions of MonkeyType that we played with our classmates that our typing speeds were very less. We also realized that due to our low typing speed, we were lacking behind on many other tasks as we were spending a lot of time while typing.

With these benefits of typing in mind, we set out to create Space Type. Space Type provides users with a fun user experience to facilitate maximum utility of the different training, test, and game modes to boost the user's typing speed with attractive graphics, fonts, engaging music and many other things.

IV. RELATED WORKS

A wide variety of typing trainer apps have existed for a long time to help people to learn typing. The most popular ones are:

A. *Typeshala*

Typeshala is the Typing Tutor Software for Windows that users can learn to type in Nepali and English. Typeshala helps to build typing speed, improves the accuracy of typing, and thus makes it more productive. On the completion of each lesson, it monitors the user's typing speed; word per minute, and precision percentage. Nepali Typeshala has a Ramayana game that helps to increase the user's typing skills. There's a new online version that is a typing tutor inspired by DOS-based old typeshala.

B. *Typing Master 2002*

TypingMaster 2002 is a tool that helps users improve their touch typing skills. This application teaches users proper typing using natural finger movements that reduce stress on the wrists, hands, and neck. TypingMaster provides users with customized typing exercises as well as feedback on typing performance, where areas for improvement are pointed out. The feedback includes the number of words per minute, the test duration, and the number of errors made. It can also be used by those who want to evaluate their mastery level and improve their typing skills for more efficiency.

C. *MonkeyType*

MonkeyType is a minimalistic typing test, featuring many modes, an account system to save your typing speed history, and user-configurable features like themes, a smooth caret, and more. This website uses the most common 200 words in the language and generates its test. After completing the test, you will be able to see your wpm, raw wpm, accuracy, character statistics, test length, leaderboard information, and test information.

V. SYSTEM REQUIREMENTS

A. *Software Specifications*

Software dependencies for making our projects are:

1) *Raylib*:

Raylib is an open-source programming library to enjoy video games programming; no fancy interface, no visual helpers, no GUI tools or editors, just coding in the pure spartan-programmers way. Raylib is supported on multiple platforms and can be used with multiple programming languages. It is written in plain C code

(C99) in PascalCase/camelCase notation and has full 3d and audio support.

2) *C Compiler*:

Our application is entirely programmed in C. To build this project, the PC needs a C compiler installed on it. So a C compiler must be installed on the PC first for testing and building the project. Some of the compilers are MinGW, Dev-C++, Clang, etc.

3) *Operating System*:

Space Type is supported in Windows and Linux-based operating systems and has been tested on Windows 10, Windows 11, and Windows 7. It is supported on both 32-bit as well as 64-bit architecture.

B. *Hardware Specifications*

Our application runs on a system that can run Raylib. The system must support OpenGL to render the Graphical elements properly.

VI. PROCEDURE

After thinking for some time, we decided to take the steps mentioned below for the development of our application.

A. *Research, Analysis, and Decision Making*

At first, we needed to familiarize ourselves with the project. We spent some time learning about the problems people make while typing. We felt that most of the people who type slowly or who make many mistakes while typing are actually typing in "Hunting and Pecking" style, which basically means watching the keyboard while typing to find the specific key that needs to be typed.

The only way to type fast and accurately was touch typing. There were other softwares and websites available to train the user for touch typing but they were bland and boring after some practice. Their interface was simple and not very attractive. Therefore after a long analysis, we decided to make a software that trains and motivates the user for touch typing but entertains them at the same time.

B. *Designing and Prototyping*

Before we started to write the codes, we decided to draw a rough sketch of what our interface would look like. We then created textures depending on those rough sketches. We prepared basic designs for the whole software. We also considered other major details like functionalities, and GUI and then estimated the total time it would take to complete this project.

C. Development and Coding

As a part of the analysis done before developing, we needed to plot out the things that our application would do. All the output features that a good touch typing trainer would have were considered. Other than basic input-output, all the calculations and processes that would take place in the background were considered.

We decided to use the Raylib development library for the graphics. As modular programming is the best way for software creation in a fast and easy manner, we decided to use this approach. We decided to create four modules: A main module that calls other modules and three other modules that handle all the features of the application. We used the C programming language along with the Raylib library to complete this project. Frequent discussions and code sharing were done to track and eliminate any errors that we could find in the code.

D. Compilation

While coding individual modules, we frequently compiled them to see if our codes worked properly and solve any errors found immediately. After we completed our codes, we combined all the modules and then compiled the whole code. The final product was not as good as expected but it was surely usable.

E. Debugging and Testing

Any bugs in the code that were found during the development of individual modules were solved as soon as they were found. After all the visible bugs were solved, we began to test the modules. For that, we executed the same module again and again. We committed some mistakes to see how the results vary. We found a few bugs that were not visible before at first glance. Some of our friends used our module and we took their feedback. On the basis of their feedback and criticism and our experience, we realized he had to improve the interface. We worked to fulfill those expectations that finally made our interface more attractive and desirable.

F. Program Documentation

After the whole compilation of all the modules and debugging and testing the code, we needed to clean it up. Our codes had a large section of unwanted codes that were just commented out, which were the result of programming and testing different approaches we tried for a specific task in our code. We first removed those sections and then removed all those unused variables that had been declared.

Then we worked on documenting our code. For that purpose, we used a tool recommended by the Pulchowk IEEE community called Doxygen. Doxygen is a documentation generator. Doxygen extracts information from specially-formatted comments within the source code.

To prepare documentation using Doxygen, we wrote information like the author, a brief description of what the specific file does, and information on bugs on each C file. Furthermore, we wrote information like why certain header files were included in the code. We grouped the variables using the style of comment Doxygen required and described what the specific group of variables did. In addition to that, we added a usage description of the variables where we found it necessary.

We added a description of what each function does and also descriptions of each parameter used in the functions. We also added the kind of value the parameters required and also what the function returned if it was a return type. We also added comments on the purpose of some blocks of code where we found it necessary.

After finishing writing the formatted comments that Doxygen required, we edited the configuration file and ran Doxygen, and repeated this process several times until we were able to get clear and concise documentation of our code.

VII. FEATURES

The application we created wasn't the same as the one we envisioned. We succeeded in designing a more engaging GUI than the one we had initially planned once we realized the capabilities of the Raylib library. But due to the limitation of the time duration, we had to cut down on the features we had planned for this application. What we finished with was a package with the right blend of engaging graphics and just enough features for this to be a full-fledged typing tutor. Some features of Space Type are:

A. Touch Typing Tutorial:

On all of the menu pages, there is a hamburger icon that shows you how your finger placement should be in a QWERTY keyboard for touch typing. So at any part of the program, you can learn the correct finger placements for touch typing.

B. Key Indicator:

As an extension of the feature from before, this feature indicates the key that you need to press on the keyboard while you are in Train mode and Test mode. As soon as that key is pressed, the next key that needs to be pressed appears.

C. *Word and Letter Train mode:*

In this mode, you can practice typing letters and words at your own pace. For both of these modes, you are presented with checkboxes for each keyboard row: Top Row, Middle Row, and Bottom Row. The words and letters that you practice in either of these modes will be from the rows you selected. In some cases, there may be letters from the rows you did not select because the rows you selected could not give a proper word.

D. *“Customized” Mode:*

Each mistyped letter in your train and test modes are stored in a text file locally. This mode checks for the file before making itself clickable. Once you select this mode, the application reads the file and provides you with words focusing on the mistyped letters.

E. *Test Mode:*

In this mode, you can put the typing skills you improved through the Train mode to the test. You can choose the difficulty you want the text to be of, and you will be presented with a random short paragraph. The paragraphs will contain more symbols and text as the difficulty increases. Like any other real-life test, you cannot escape it i.e. you must complete the test to exit the test. At the end of the test, you will be presented with a result screen showing your typing speed, accuracy, and the total time elapsed.

F. *Game Mode:*

This is the most competitive mode in our application. Since our project is spaceship-themed, you are tasked to defend the spaceship against incoming attackers. The attackers are words of various lengths - some are just three letters long while some are more than ten letters. A laser beam from the spaceship destroys the attacking words once you finish typing them. The speed of the words increases the more you keep playing the game. A game over screen with a high score and necessary statistics is presented at the end of the game.

VII. LIMITATIONS

As we mentioned in the earlier section, we had to cut down on many features we had envisioned for this project, some because of lack of time and some because they were too complicated to execute for us. While this program is stable, it is nowhere near perfect. Nevertheless, we have not given up and are trying out different ways to make a more engaging program. Some of the limitations of our projects are:

A. *An Intelligent Train Mode*

Instead of having a separate “Customized” Train mode, we had initially planned to have the succeeding words in the Train mode to address the weak areas until the user performs better. With this, the Train mode would have constantly been changing its difficulty and word selection along with the progress of the typing skills of the user.

B. *More Words in Game Mode*

Right now, the only form of difficulty in the Game mode is the increasing speed at which the words approach the spaceship. While this is fun, this makes for linear gameplay in which, for a certain speed level, your score is sure to fall in a certain range. Our initial plan was to increase the number of attacking words while having the increment of the speed slower than what is in the finished application and provide three life for the spaceship. This would have resulted in more fun gameplay than what we have right now.

C. *Multiplayer Type Race.*

This was another mode we had planned for the application. In this, another user who is connected to the same network could race with each other to test who can type a certain text faster.

D. *Progress Graph*

This was another feature we planned to put in our application. With this, a user would be able to see his/her progress over time in the form of simple graphs. This would help the user to track their progress and be aware of their performance. This would aid them in deciding how to advance in learning how to type fast.

IX. RESULT AND DISCUSSION

Hence, we were able to make a fully working and effective application which is basically a trainer for touch typing. We mainly used basic c programming language and Raylib library for this project. We had to do multiple tests to actually find a bug although there were a lot of bugs and errors that we solved during the development phase of the application. The application is still not what we imagined but we will be working on a better version of Space Type.

X. CONCLUSION

In this way, we were able to create Space Type, a perfect trainer for touch typing. When we started working on this project, we knew that it would be an interesting experience and a great learning opportunity, but we had never anticipated the large number of challenges that would come along. We learned a lot of new things that were very helpful to solve

these new challenges. This project also helped us in gaining skills like time management, leadership, planning, coordination, etc.

In the end, this was a very helpful project that introduced us to the different aspects of programming and made us familiar with the use of Raylib library for graphics handling.

ACKNOWLEDGEMENT

First of all, Space Type wishes to acknowledge Professor Ganesh Gautam who taught us the basics of C programming. Without his guidance and support back in the classes, we would not have been able to complete this project.

We would also like to thank IEEE Pulchowk and Logpoint for providing us with an opportunity to work on this project. We would also like to thank Mr. Sujan Baskota who provided a major layout idea in the application.

We would also like to thank Mr. Nabin Ghartimagar who gave us concepts and ideas when we were stuck on a problem and discovered a major bug in the application.

Last but not the least, we would also like to thank other contributors for supporting and believing in us throughout this journey.

REFERENCES

- [1]Monkeytype website. [Online]. Available: <https://monkeytype.com/about>
- [2]Typeshala website. [Online]. Available: <http://typeshala.shresthasushil.com.np/aboutus>
- [3]Typing Master website. [Online]. Available: <https://www.typingmaster.com>
- [4]Raylib website. [Online]. Available:<https://www.raylib.com/index.html>