

Type Through the Bible (C++ Edition)

A Typing Game of Biblical Proportions

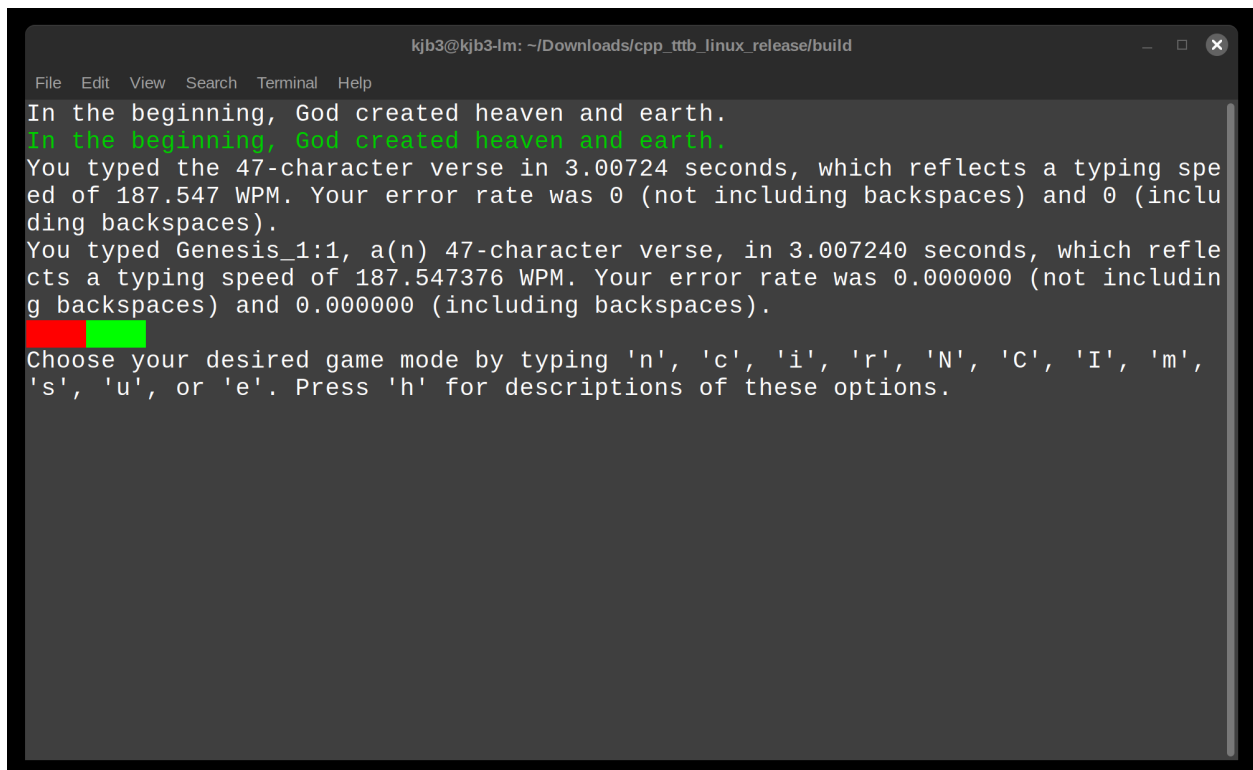
By Ken Burchfiel

Released under the MIT License

[Link to GitHub repository](#)

[Link to game download page on Itch.io](#) (password: `microseconds`)

[Note: this Readme is an early work in progress, and many sections are still incomplete. More information will be provided soon!]

A screenshot of a terminal window titled "kjb3@kjb3-lm: ~/Downloads/cpp_ttb_linux_release/build". The terminal shows the game's output for a typing test. It displays the text "In the beginning, God created heaven and earth." twice, with the second instance in green. It then provides statistics: "You typed the 47-character verse in 3.00724 seconds, which reflects a typing speed of 187.547 WPM. Your error rate was 0 (not including backspaces) and 0 (including backspaces)." and "You typed Genesis_1:1, a(n) 47-character verse, in 3.007240 seconds, which reflects a typing speed of 187.547376 WPM. Your error rate was 0.000000 (not including backspaces) and 0.000000 (including backspaces)." Below this, there is a red and green progress bar. The prompt "Choose your desired game mode by typing 'n', 'c', 'i', 'r', 'N', 'C', 'I', 'm', 's', 'u', or 'e'. Press 'h' for descriptions of these options." is shown. The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help".

```
kjb3@kjb3-lm: ~/Downloads/cpp_ttb_linux_release/build
File Edit View Search Terminal Help
In the beginning, God created heaven and earth.
In the beginning, God created heaven and earth.
You typed the 47-character verse in 3.00724 seconds, which reflects a typing speed of 187.547 WPM. Your error rate was 0 (not including backspaces) and 0 (including backspaces).
You typed Genesis_1:1, a(n) 47-character verse, in 3.007240 seconds, which reflects a typing speed of 187.547376 WPM. Your error rate was 0.000000 (not including backspaces) and 0.000000 (including backspaces).
Choose your desired game mode by typing 'n', 'c', 'i', 'r', 'N', 'C', 'I', 'm', 's', 'u', or 'e'. Press 'h' for descriptions of these options.
```

Introduction

Type Through the Bible (TTTB) allows you to practice your keyboarding skills by typing the Bible verse by verse. It contains both single-player and multiplayer modes, and also offers (via a complementary Python script) a wide variety of interactive visualizations.

Download instructions

Linux, Windows, and OSX copies of TTTB are available [at this page](#) on itch.io . Until the documentation for the game is more complete, the page will be password-protected; you can access it by entering the password `microseconds`.

The game is free to download, but donations (while not expected) are greatly appreciated.

Make sure to download both the zipped folder for your operating system *and* the corresponding README (e.g. the file you're reading right now). Once you've unzipped the folder on your local computer, you can move it to a location of your choice—or simply keep it within the downloads folder. Nothing needs to be installed in order for you to begin playing the game.

(If multiple people will be playing TTTB' single-player mode on your computer, I recommend creating a separate copy of the game for each of them. This will make it much easier for each player to keep track of his or her progress.)

Because TTTB is released under the MIT license, you are free to modify its underlying C++ and Python code, then share your revisions with others (even under a proprietary license). In this case, you'll want to download and compile the source code rather than use these underlying binaries. See 'Compilation instructions' below for more details.

Updating your default configuration settings

Once you've downloaded the game, you may want to update your default configuration settings before you begin. That way, certain default values (like your name) will get stored within every test; you'll also have the option to update these settings within single-player games. You can add these default options within the second row (e.g. the row right below the headers) of the `game_config.csv` file.

Here's what my default settings look like: (I use `Tag_1` to store the keyboard I'm using for a test and `Tag_2` to store my location. You can choose not to enter any default tags, but I'd recommend at least storing your name within the 'Player option.')

	A	B	C	D	E
1	Player	Tag_1	Tag_2	Tag_3	Notes
2	KJB3	Cherry_Red	Home		
3					

TTTB's folder directory

Here is how TTTB's folders and files are organized:

`cpp_tttb/` [the root folder for the game; your exact name may differ]

—**build/**

—`_internal/` [contains Pyinstaller-related files; these are crucial to include in order for the corresponding Python executable to work correctly, but you shouldn't need to modify them]

—`tttb` [`tttb.exe` on Windows; the main TTTB executable that you'll launch in order to start the game]

—`tttb_py_complement` [`tttb_py_complement` on Windows; the Python executable that TTTB uses to (1) create single-player and multiplayer visualizations and (2) combine various multiplayer files into single files. This executable can be called either within TTTB or as a standalone executable]

—**Files/**

—`MP_Test_Result_Files_To_Combine/`

—`MP_Word_Result_Files_To_Combine/`

—`Multiplayer/` [multiplayer results will be stored here.]

—`CPDB_for_TTTB.csv` [A local copy of the Catholic Public Domain Bible. Your overall progress in typing through the entire Bible will get stored within this file.]

—`game_config.csv` [This file stores default configuration options, such as your name and various gameplay tags.]

—`test_results.csv` [This file provides detailed data on all of your test results, including WPM and accuracy data; start and end times; and verse-related information.]

——word_results.csv [This file provides word-level WPM and accuracy data.]

——**Visualizations/**

——Multiplayer/ [Multiplayer visualizations will get stored here.]

——Single_Player/ [Single-player visualizations will get stored here.]

Gameplay instructions

Starting the game

To launch TTTB, you'll first want to open up your command prompt or terminal, then navigate to the program's build folder (where the game's 'tttb'/'tttb.exe' executable file is located). On my Linux computer, I can accomplish this via the following steps:

1. Press Ctrl + Alt + T to launch my terminal
2. Enter `cd '/home/kjb3/D1V1/Documents/!Dell64docs/Programming/CPP/cpp_tttb/build'` to navigate to the build folder. (This folder path will of course look different on your end.)

On Windows, I would instead press the Windows button, then enter `cmd` to bring up the command prompt. I would then enter `cd` followed by the path to my build folder (perhaps encased in double quotes).

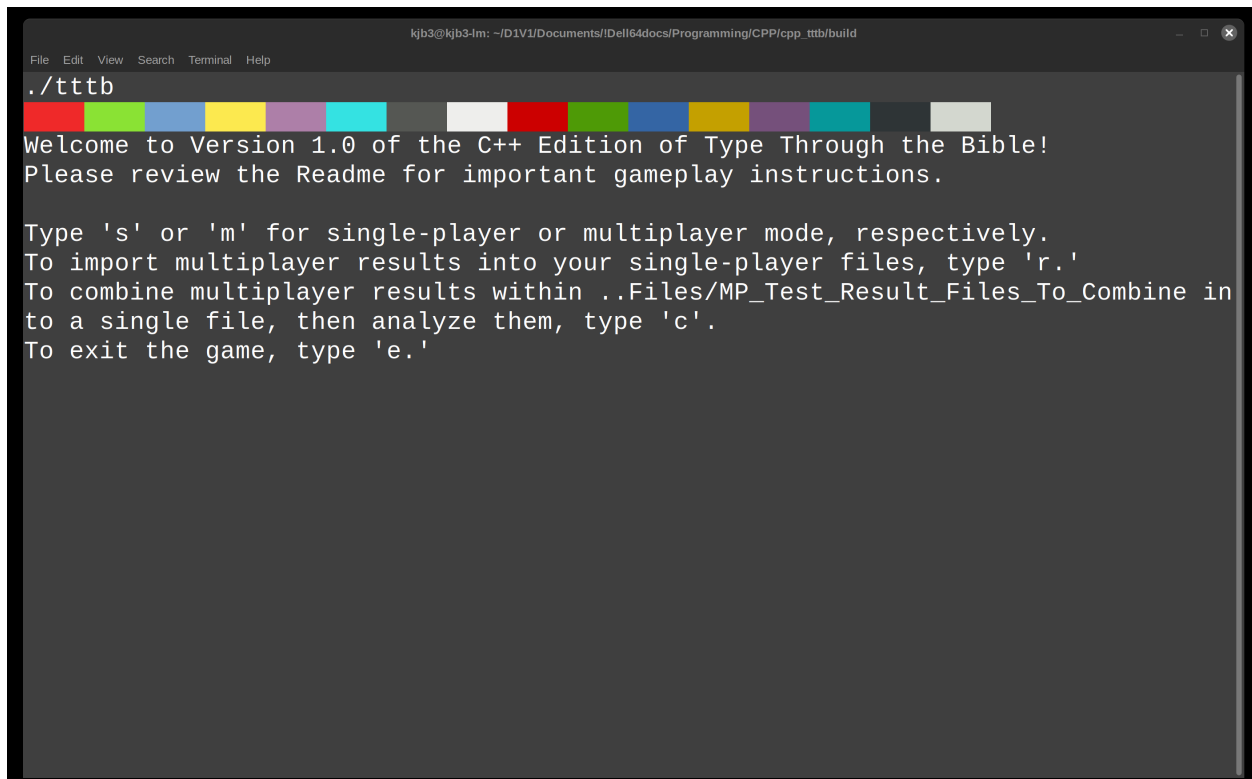
Once you've navigated to this folder, you can launch TTTB by entering the following command (on Linux or OSX):

```
./tttb
```

On Windows, you would instead enter:

```
tttb.exe
```

Once you've completed these steps, you should see the following welcome screen:

A screenshot of a terminal window titled "kjb3@kjb3-lm: ~/D1V1/Documents/!Dell64docs/Programming/CPP/cpp_tttb/build". The terminal shows the command `./tttb` being executed. The output is a colorful ASCII art banner followed by a welcome message: "Welcome to Version 1.0 of the C++ Edition of Type Through the Bible! Please review the Readme for important gameplay instructions." Below this, instructions are provided for single-player and multiplayer modes, importing results, combining files, and exiting the game.

```
kjb3@kjb3-lm: ~/D1V1/Documents/!Dell64docs/Programming/CPP/cpp_tttb/build
File Edit View Search Terminal Help
./tttb
Welcome to Version 1.0 of the C++ Edition of Type Through the Bible!
Please review the Readme for important gameplay instructions.

Type 's' or 'm' for single-player or multiplayer mode, respectively.
To import multiplayer results into your single-player files, type 'r.'
To combine multiplayer results within ..Files/MP_Test_Result_Files_To_Combine in
to a single file, then analyze them, type 'c'.
To exit the game, type 'e.'
```

Playing Type Through the Bible

Type Through the Bible offers both single-player and multiplayer gameplay modes.

Single-player gameplay To launch single-player games, press ‘s’ within the welcome screen. You’ll then be presented with a number of different single-player modes from which to choose:

- **n**: Type the next untyped verse, then return to the gameplay menu. (This, along with ‘N’, is a good starting option.)
- **c**: Type the next verse, then return to the gameplay menu. (The main difference between ‘c’ and ‘n’ is that the former won’t skip over verses that you’ve already typed at least once.)
- **i**: Type a specific verse ID, then return to the gameplay menu.
- **r**: Repeat the verse you just typed, then return to the gameplay menu.
- **N**: Automatically get directed to the next untyped verse. (This option eliminates the need to press ‘n’ after each test if you wish to stay in this mode.)
- **C**: Automatically get directed to the next verse.
- **I**: Select a verse from which to begin, then automatically get directed to the verses that follow it.
- **M**: Enter ‘untyped marathon mode,’ in which you will immediately start a test for the next untyped verse after finishing the current test. (This option, along with ‘S’, can get pretty exhausting—so consider trying it out only for short periods.)
- **S**: Enter ‘sequential marathon mode,’ in which you will immediately start (1) the verse following the one you just typed or (2) (when starting this mode) a verse of your choice.
- **U**: Update game configuration settings.
- **C**: Exit this session and save your progress.

Note: You can exit out of any test, even in marathon mode, by pressing Ctrl+C (not to be confused with Command+C). **Do not** simply exit out of the terminal, as your progress then won’t be saved.

Running typing tests [To be completed soon!]

Multiplayer gameplay [To be completed soon!]

Converting single-player data to multiplayer data (and vice versa) TTTB also lets you combine various copies of results from different computers into a single multiplayer file that you can then analyze.

[To be continued!]

[To be completed soon!]

Compilation instructions

These instructions should work for Linux, OSX, and Windows. Please let me know if you encounter any issues.

1. Download [CMake](#) if you haven’t already.
2. Clone the [cpp-terminal](#) and [csv-parser](#) libraries, both of which are permissively licensed, and build them using CMake. **Note:** I recommend switching the BUILD_SHARED_LIBRARIES option within the cpp-terminal [CMakeLists.txt](#) to OFF for all platforms in order to make your resulting code more portable.

(To build each library using CMake, (a) create a ‘build’ folder within the folder on your computer that contains the library; (b) navigate within this folder using your terminal; (c) run `cmake ..` in preparation for the build; and then, if that command was successful, (d) run `cmake --build .` Don’t forget the space and period at the end!)

3. Copy these folders to a directory of your choice, or leave them in your downloads folder if you prefer.

4. Clone [Type Through the Bible](#) and move it to a directory of your choice.
5. **IMPORTANT:** Within TTTB's [CMakeLists.txt](#) file, you *must* replace the existing paths to my copies of the cpp-terminal and csv-parser libraries within the `add_subdirectory()` calls with paths to *your* copies of these folders (at least for the platform(s) for which you're building TTTB). Within each `add_subdirectory()` call, the first path is to the actual 3rd-party-library's location on your computer; the second path tells CMake where to place some additional files related to that library. (You don't need to create this second folder on your system; CMake will create it automatically.)
6. Navigate into TTTB's `build/` folder.
7. Use CMake to build TTTB. (See above steps for reference if needed.) Once it has been build, copy the resulting executable (e.g. `tttb` or `tttb.exe` on Windows) into your build folder. (Windows will likely place it into a separate subfolder, but it must be placed within your main build folder in order for the relative paths used by the program to work correctly.)
8. Next, you'll need to build an executable version of the `tttb_py_complement.py` file. First, download [Pyinstaller](#) if you don't have it already. Also, download Python (i.e. via [Miniforge](#) if it's not already on your system).
9. Make sure that Pandas, Numpy, and Plotly are all installed within the Python environment that you plan to use to run Pyinstaller. (These can be downloaded via conda-forge.) Otherwise, the executable version of your `tttb_py_complement.py` file won't work correctly.

Note: You may also want to install JupyterLab (e.g. via `conda install jupyterlab`) so that you can modify, if needed, the `.ipynb` notebooks contained within the source code. (Just make sure to export these updated notebooks to a `.py` file so that they can get processed correctly by `create_release_folder.py` and your `pyinstaller` call.)

10. Once you have Python and Pyinstaller set up, run `pyinstaller tttb_py_complement.py` within TTTB's 'build' folder. This will create an executable version of this file, along with an `'_internal'` folder that contains important library components, into a `'dist/tttb_py_complement'` subfolder within your 'build' folder. ****Cut and paste both the `_internal` folder and the `tttb_py_complement` executable into your build folder, as that's where the program will look for them.****

Note: If you encounter issues with Pyinstaller, consider creating a *new* conda environment that contains *only* the libraries required for the code to run (e.g. not even JupyterLab); downloading the latest copies of all libraries; and then rerunning your Pyinstaller command.

10. Navigate up to the main `cpp_tttb` folder (e.g. via `cd ..` in your terminal, assuming you're still in the build folder) and run the `create_release_folder.py` Python file. This will create a new copy of TTTB with blank output files rather than the existing files within the TTTB directory.

Note: This file also has code that can help automate the process of building and (where necessary) moving your C++ and Python executables. (For instance, you can pass `'both'` as a corresponding argument to the file rather than `'neither,'` the default, in order to comopile the executables. See the source code for more documentation and details.) However, I strongly recommend performing these steps outside of this file the first time around so that you can more easily identify and debug any issues.

11. You should now be able to begin playing TTTB!

Development Notes

I had created an earlier Bible typing game in Python, but given my interest in learning how to apply C++ to create games, I figured that a C++ version of this game would be a great learning opportunity. Although I didn't reference the Python version of this game when writing this code, my experiences with that project certainly influenced the current one.

This project was a lot of fun to work on. While it doesn't feature a GUI, it *was* a good opportunity to for me to get reacquainted with C++—and to gain experience with importing and exporting CSV data. I hope that

you enjoy playing it as much as I did coding it!

As with my other GitHub projects, I chose not to use generative AI tools when creating Type Through the Bible. I wanted to learn how to perform these tasks in C++ (or in a C++ library) rather than simply learn how to get an AI tool to create them.

This code also makes extensive use of the following open-source libraries:

1. Vincent La's CSV parser (<https://github.com/vincentlaucsb/csv-parser>)
2. CPP-Terminal (<https://github.com/jupyter-xeus/cpp-terminal>)

In addition, this program uses the Catholic Public Domain Version of the Bible that Ronald L. Conte put together. This Bible can be found at <https://sacredbible.org/catholic/> I last updated my local copy of this Bible (whose text does get updated periodically) around June 10, 2025.

Acknowledgments

I am grateful for Ronald L. Conte Jr. for his work on the [Catholic Public Domain Version](#) of the Bible (the translation that TTTB uses).

This game is dedicated to my wife, Allie. I am very grateful for her patience and understanding as I worked to put it together! My multiplayer gameplay sessions with her (yes, she was kind enough to play it with me) also helped me refine the code and improve the OSX release.

Blessed Carlo Acutis, pray for us!

[More acknowledgments to come!]