

Space Type

1.0

Generated by Doxygen 1.9.6

1 Bug List	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 charCount Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 character	7
4.1.2.2 count	7
5 File Documentation	9
5.1 main.c File Reference	9
5.1.1 Detailed Description	10
5.1.2 Function Documentation	10
5.1.2.1 draw_menu()	11
5.1.2.2 main()	11
5.2 spacetype_functions.c File Reference	12
5.2.1 Detailed Description	14
5.2.2 Function Documentation	14
5.2.2.1 draw_background()	14
5.2.2.2 keyboard_highlight()	14
5.2.2.3 pause_screen()	15
5.2.2.4 remove_firstletter()	15
5.2.2.5 reset_counter()	15
5.2.2.6 tutorial_screen()	15
5.3 spacetype_functions.h	16
5.4 spacetype_game.c File Reference	16
5.4.1 Detailed Description	18
5.4.2 Function Documentation	18
5.4.2.1 game()	18
5.4.2.2 text_mover()	19
5.4.3 Variable Documentation	19
5.4.3.1 words	19
5.5 spacetype_game.h	20
5.6 spacetype_test.c File Reference	20
5.6.1 Detailed Description	22
5.6.2 Function Documentation	22
5.6.2.1 test()	22
5.6.2.2 test_menu()	23

5.6.2.3 test_process()	23
5.6.3 Variable Documentation	24
5.6.3.1 cockpitTextureKeyboard	24
5.6.3.2 text1	24
5.7 spacetype_test.h	24
5.8 spacetype_train.c File Reference	24
5.8.1 Detailed Description	26
5.8.2 Function Documentation	26
5.8.2.1 customized_train()	26
5.8.2.2 letter_train()	27
5.8.2.3 mode_trainletters()	28
5.8.2.4 mode_trainwords()	28
5.8.2.5 select_letter()	29
5.8.2.6 select_word()	29
5.8.2.7 train()	29
5.8.2.8 train_menu()	30
5.8.2.9 train_select()	30
5.8.2.10 word_train()	31
5.8.3 Variable Documentation	31
5.8.3.1 cockpitTextureKeyboard	31
5.8.3.2 customizedWords	32
5.9 spacetype_train.h	32
Index	33

Chapter 1

Bug List

File [main.c](#)

: No known Bug.

File [spacetype_functions.c](#)

No Known Bugs

File [spacetype_game.c](#)

No known bug

File [spacetype_test.c](#)

No known Bug

File [spacetype_train.c](#)

No Known Bug

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

charCount	This structure stores mistyped characters and their frequency	7
---------------------------	---	---

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

main.c	Main application file of Space Type	9
spacetype_functions.c	Created functions for this application	12
spacetype_functions.h		??
spacetype_game.c	Word Shooter Game for Spacetype	16
spacetype_game.h		??
spacetype_test.c	Test Mode of Space Type	20
spacetype_test.h		??
spacetype_train.c	Word and Letter Typing train mode	24
spacetype_train.h		??

Chapter 4

Data Structure Documentation

4.1 charCount Struct Reference

This structure stores mistyped characters and their frequency.

```
#include <spacetype_functions.h>
```

Data Fields

- char `character`
- int `count`

4.1.1 Detailed Description

This structure stores mistyped characters and their frequency.

4.1.2 Field Documentation

4.1.2.1 character

`character`

Member 'character' contains the mistyped character

4.1.2.2 count

`count`

Member 'count' contains the the mistyped character's frequency

The documentation for this struct was generated from the following file:

- `spacetype_functions.h`

Chapter 5

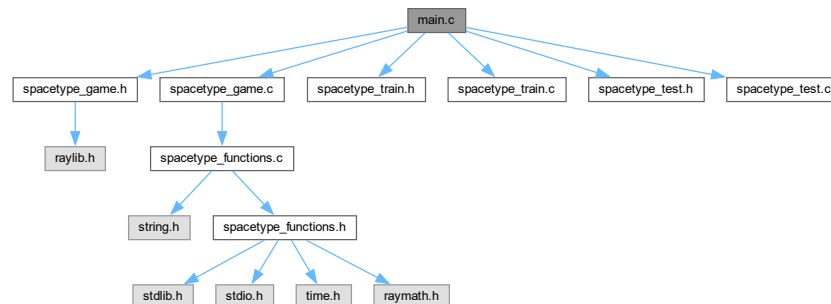
File Documentation

5.1 main.c File Reference

Main application file of Space Type.

```
#include "spacetype_game.h"  
#include "spacetype_game.c"  
#include "spacetype_train.h"  
#include "spacetype_train.c"  
#include "spacetype_test.h"  
#include "spacetype_test.c"
```

Include dependency graph for main.c:



Functions

- `int main ()`
Main function of the application.
- `void draw_menu ()`
Draws main menu for the application.

Variables

Main application variables

- FILE * **wrongChars**
To store mispressed characters to use in customized train mode.
- int **screenWidth**
screen width for graphical operations
- int **screenHeight**
screen height for graphical operations
- bool **exitWindow** = false
To govern main application loop.
- bool **exitGame** = false
To govern game mode loop.
- bool **sorted**
to check if the WrongChars.txt is sorted or not

Main interface variables

Different textures for the main screen and hover, and other graphical components.

- Texture2D **cockpitTexture**
Background Textures.
- Texture2D **cockpitTextureTrain**
- Texture2D **cockpitTextureTest**
- Texture2D **cockpitTextureGame**
- Texture2D **cockpitTextureExit**
Main menu textures.
- Font **retroFont**
- Font **regularFont**
Main application fonts.
- Texture2D **qwertyTexture**
keyboard image for tutorial screen

5.1.1 Detailed Description

Main application file of Space Type.

retro-interface typing trainer and game to help boost your typing speed. This project takes its inspiration from Typeshala, Ztype and aims for the feels similar to retro-console space games like Space Invaders. SPDX-License-Identifier: LGPL-2.1-or-later

Author

Praharsha Adhikari 078bct061.praharsha@pcampus.edu.np
 Mukunda Dev Adhikari 078bct049.mukunda@pcampus.edu.np
 Pragalbha Acharya 078bct060.pragalbha@pcampus.edu.np

Bug : No known Bug.

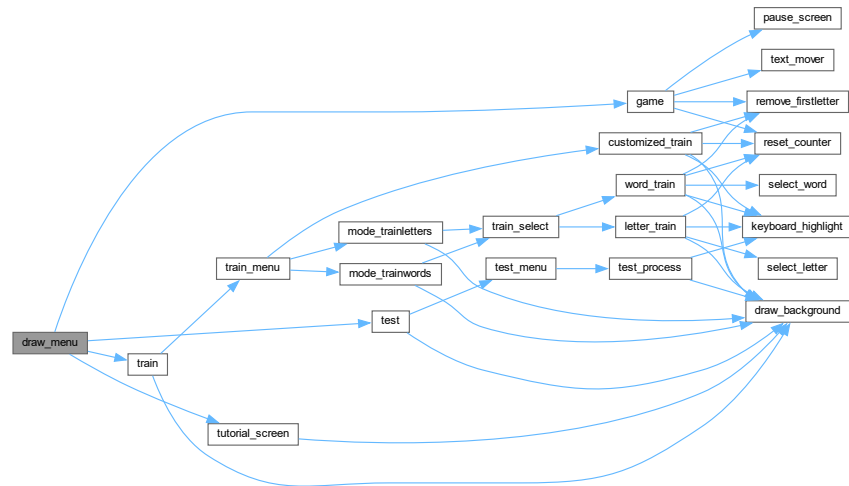
5.1.2 Function Documentation

5.1.2.1 draw_menu()

```
void draw_menu ( )
```

Draws main menu for the application.

Draws the main menu of the application by loading the cockpitTexture with mouse control for navigation. Has indicators for hover and mouse click calls the respective function associated with the menu entry. Here is the call graph for this function:

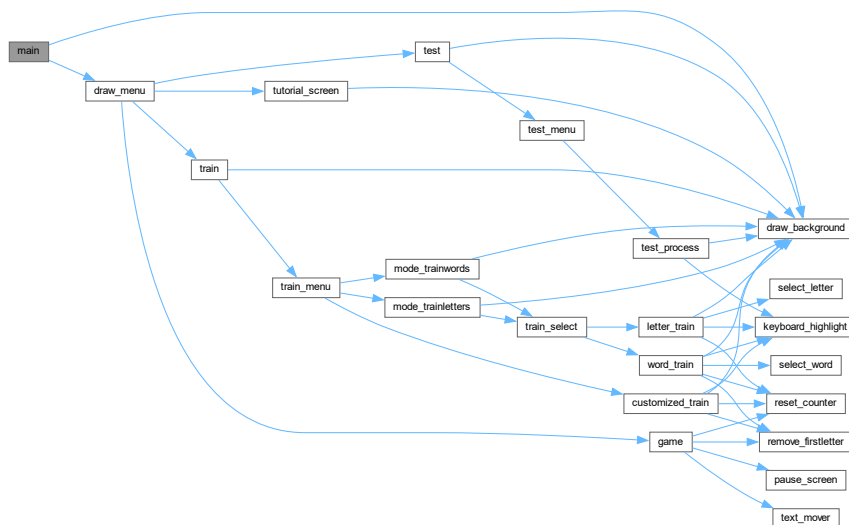


5.1.2.2 main()

```
int main ( )
```

Main function of the application.

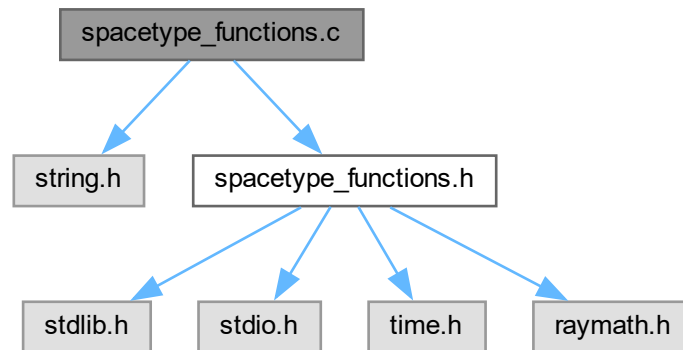
Initializes Screen and audio device. Loads music, fonts and textures. Plays the music, calls the respective function to draw the background and menu. Here is the call graph for this function:



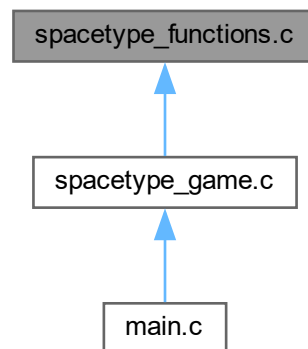
5.2 spacetype_functions.c File Reference

Created functions for this application.

```
#include <string.h>
#include "spacetype_functions.h"
Include dependency graph for spacetype_functions.c:
```



This graph shows which files directly or indirectly include this file:



Functions

- void `remove_firstletter` (char `word`[])
Removes the first letter of the word sent.
- void `draw_background` ()
Draws background in all modes.

- void `reset_counter` ()
resets variables for required statistics
- void `pause_screen` ()
Creates a pause menu.
- void `keyboard_highlight` (char a)
highlights required letter
- void `tutorial_screen` ()
Shows a tutorial screen for touch typing.

Variables

Statistics variables

variables handle counters which are associated with with statistics of Word shooter game and word train mode.

- char **fastestWord** [20]
- char **slowestWord** [20]
- char **scoreString** [50]
- int **SCORE**
- float **TIME** = 10
- float **planetTime** = 0
- float **keysPressed**
- float **rightKeysPressed**
- float **framesCounterForSession**
- float **framesCounterForWord**
- float **fastestWordFrames**
- float **slowestWordFrames**
- bool **gapMeasured**
- bool **exitPause** = true

Background and main interface variables

handle different elements related to main interface of the program like background textures, music, scale, etc

- float **scale**
Image scale for the uniformity.
- float **movingDown**
Variable to govern infinitely scrolling background.
- float **movingPlanets**
Variable to govern moving planets.
- float **mover**
Variable which controls the speed of the word in game mode.
- Music **music**
background music
- Texture2D **spaceTexture**
Space Background used in infinite scroll background.
- Texture2D **planetTextures** [3]
Array of 3 different planet images to display on background.
- Texture2D **bulletTexture**
Texture of Bullet used in game mode.
- Texture2D **spaceshipTexture**
Texture of spaceship in game mode.
- Texture2D **cockpitTextureKeyboard**
Background Texture.

Extern variables from main

different variables initialized before needed for this portion

- int **screenWidth**
screen width for graphical operations
- int **screenHeight**
screen height for graphical operations
- Font **retroFont**
- bool **exitGame**
To govern game mode loop.
- bool **sorted**
- FILE * **wrongChars**
To store mispressed characters to use in customized train mode.
- Texture2D **qwertyTexture**
keyboard image for tutorial screen

5.2.1 Detailed Description

Created functions for this application.

Author

Praharsha Adhikari 078bct061.praharsha@pcampus.edu.np

Bug No Known Bugs

5.2.2 Function Documentation

5.2.2.1 draw_background()

```
void draw_background ( )
```

Draws background in all modes.

Draws the infinitely scrolling space background with moving planets.

5.2.2.2 keyboard_highlight()

```
void keyboard_highlight (
    char a )
```

highlights requiried letter

the key you need to press in the keyboard for Train mode

Parameters

<i>a</i>	the letter which needs to be highlighted
----------	--

5.2.2.3 pause_screen()

```
void pause_screen ( )
```

Creates a pause menu.

a pause menu when called. Has options to either resume the game or return back to main menu

5.2.2.4 remove_firstletter()

```
void remove_firstletter (
    char word[] )
```

Removes the first letter of the word sent.

Parameters

<code>word[]</code>	the word sent to have its first letter removed
---------------------	--

5.2.2.5 reset_counter()

```
void reset_counter ( )
```

resets variables for required statistics

all the variables like score, fastestword, slowest word, etc. that are used to calculate statistics during word practice and Game mode

5.2.2.6 tutorial_screen()

```
void tutorial_screen ( )
```

Shows a tutorial screen for touch typing.

Shows the touch typing finger placement in keyboard when Help button is pressed in main menu Here is the call graph for this function:



5.3 spacetype_functions.h

```

00001
00002 #include <stdlib.h>
00003 #include <stdio.h>
00004 #include <time.h>
00005 #include <raymath.h>
00006
00007 void draw_background();
00008 void draw_menu();
00009 void reset_counter();
00010 void remove_firstletter(char word[]);
00011 void pause_screen();
00012 void tutorial_screen();
00013 void keyboard_highlight (char a);
00021 struct charCount{
00022     char character;
00023     int count;
00024 } wrongLetters[26];

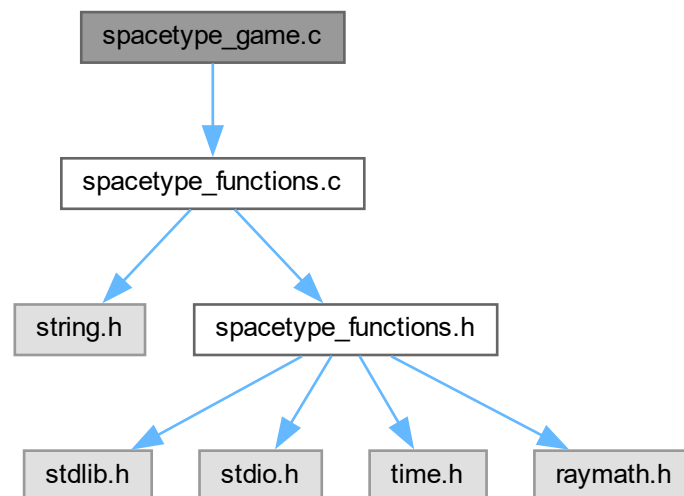
```

5.4 spacetype_game.c File Reference

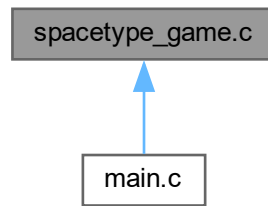
Word Shooter Game for Spacetype.

```
#include "spacetype_functions.c"
```

Include dependency graph for spacetype_game.c:



This graph shows which files directly or indirectly include this file:



Functions

- void `text_mover` (Vector2 *`wordPos`, Rectangle `playerPos`, Vector2 `gap`, float `time`)
Moves the word toward the player.
- void `game` ()
main function for game mode

Variables

Main gameplay variables

These variables handle different aspects related to the main gameplay.

- char * `words` []
List of words to choose from for the game.
- char `word` [20]
word which is presented to type
- char `wordStored` [20]
stores the presented word to later compare with fastestword and slowestword
- int `sizeOfArray` = sizeof(`words`) / sizeof(`words`[0])
for calculating total number of words/*
- Vector2 `gap` = {}
Shortest distance between word and spaceship/.*
- Vector2 `wordPos`
position of word as it falls down
- const int `fontSize` = 25
Fonsize declaration for uniformity of fontsize.
- float `angle` = 0
Angle of word to have it fall towards player.
- float `prevAngle` = 0
Angle of the spaceship.
- float `movingPlanets` = 0
Denotes which planet will be on screen.
- float `movingDown` = 0
Parameters which makes space texture scroll infinitely/.*
- bool `GAME_OVER` = false
boolean to check game over condition

Bullet related variables

These variables handle different aspects related to bullet which destroys the word when it is finished being typed.

- Sound **shoot**
Audio played when bullet is used.
- Rectangle **bulletPos**
Bullet Position.
- bool **bullet** = false
indicates whether bullet needs to be shoot or not
- float **bulletAngle**
Determine angle at which bullet needs to be thrown depending word position.
- float **bulletMover** = 0
To indicate speed of the bullet.

Extern variables from main

different variables initialized before needed for this portion

- Music **music**
background music
- int **screenWidth**
screen width for graphical operations
- int **screenHeight**
screen height for graphical operations
- Font **retroFont**

5.4.1 Detailed Description

Word Shooter Game for Spacetype.

Author

Praharsha Adhikari 078bct061.praharsha@pcampus.edu.np

Bug No known bug

5.4.2 Function Documentation

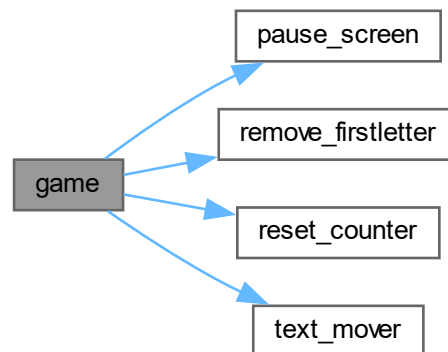
5.4.2.1 game()

```
void game ( )
```

main function for game mode

Handles everything regarding spaceship shoot game. Resets variables specific to the game and general by calling reset_counter. Draws the graphics regarding spaceship falling words and bullets. Handles the gameplay mechanics and shows a gameover screen when required. Resetting in case of new game from menu

Initialize variables and texturesHere is the call graph for this function:



5.4.2.2 text_mover()

```

void text_mover (
    Vector2 * wordPos,
    Rectangle playerPos,
    Vector2 gap,
    float time )
  
```

Moves the word toward the player.

Parameters

<i>wordPos</i>	Position of generated word
<i>playerPos</i>	Position of spaceship
<i>gap</i>	Shortest distance between word and spaceship
<i>time</i>	Set time for the word to hit the player

5.4.3 Variable Documentation

5.4.3.1 words

```
char* words[ ]
```

Initial value:

```
=
{
    "apple", "ant", "airplane", "banana", "book", "boat", "cat", "cow", "car", "dog", "desk", "dolphin",
    "elephant", "egg", "earth", "fish", "flamingo", "frog", "giraffe", "goat", "grapes", "hat", "horse",
    "house", "igloo", "icecream", "insect", "jacket", "jaguar", "juice", "kangaroo", "kite", "key",
    "lion", "leopard", "lamp", "monkey", "mouse", "mango", "night", "nest", "napkin", "octopus",
    "ostrich", "onion", "pear", "panda", "pig", "queen", "quail", "question", "rabbit", "rhinoceros",
    "ring", "snake", "snail", "sock", "tiger", "taco", "table", "unicorn", "umbrella", "vase",
    "vegetable", "whale", "wolf", "watermelon", "xray", "xylophone", "yak", "yoyo", "zipper", "zoo"
}
```

List of words to choose from for the game.

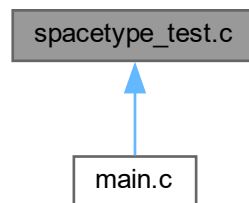
5.5 spacetype_game.h

```
00001 #include <raylib.h>
00002
00003 void game();
00004 void text_mover(Vector2 *wordPos, Rectangle playerPos, Vector2 gap, float time);
```

5.6 spacetype_test.c File Reference

Test Mode of Space Type.

This graph shows which files directly or indirectly include this file:



Functions

- void `test` ()
main function for test mode
- void `test_menu` ()
main menu for test screen
- void `test_process` (char test_text[])
Handles the typing test process.

Variables

Passages for Test mode

- char **text1** [300] = {"Wealth, fame, power. Gold Roger, the King of the Pirates, attained everything this world has to offer. And so, many men head for the Grand Line to find the great treasure he left behind, the One Piece. The world has truly entered a Great Pirate Era!"}
- *Passages for Test Mode.*
- char **text2** [381] = {"The Tale of Jiraiya the Gallant. Now it'll end a bit better, I hope. The final chapter. I'll call it: Frog at the bottom of the well drifts off into the great ocean. Just barely glorious. But glorious indeed. Now I suppose it's about time I put down my pen. Oh, right. What should I name the sequel? I wonder. Let's see: The Tale of Naruto Uzumaki. Yes, that has a nice ring to it."}
- char **text3** [450] = {"A late 20th century trend in typing, primarily used with devices with small keyboards (such as PDAs and Smartphones) is thumbing or thumb typing. This can be accomplished using one or both thumbs. Similar to desktop keyboards and input devices, if a user overuses keys which need hard presses and/or have small and unergonomic layouts, it could cause thumb tendonitis or other repetitive strain injury."}
- char **text4** [450] = {"Today, historians relate that, as a general rule, buying and selling securities was very much unorganized before the year 1792. Every person who owned a security faced the problem of finding interested buyers who might consider the purchase of a debt-free investment. This meant most people were somewhat slow in investing in stocks and bonds because these securities could not readily be converted into money."}
- char **text5** [287] = {"The Master of Business Administration (MBA or M.B.A.) degree originated in the United States. The core courses in an MBA program cover various areas of business such as accounting, applied statistics, business law, finance, managerial economics, management, entrepreneurship & marketing."}
- char **text6** [438] = {"When we talk about motivating others, the justification is the end result (either we want to avoid the pain or go towards pleasure) or what we want to get the person to do. How we achieve the end result, are our alternatives. As a manager, we need to understand the other person's justification and then come up with alternatives. We may then choose the right alternative. Typically people stop at this level of analysis and start to act."}
- char **text7** [456] = {"A data entry clerk is a member of staff employed to enter or update data into a computer system. Data is often entered into a computer from paper documents using a keyboard. The keyboards used can often have special keys - Alt, Ctrl, Fn, Shift - multiple colors to help in the task & speed up the work. Proper ergonomics at the workstation is a common topic considered. The Data Entry Clerk may also use a mouse, and a manually-fed scanner may be involved."}
- char **text8** [530] = {"An ever-growing number of complex rules plus hard-to-cope-with regulations are now being legislated from state to state. Key federal regulations were formulated by the FDA, FTC, and the CPSC. Each of these federal agencies serves a specific mission. One example: Laws sponsored by the Office of the Fair Debt Collection Practices prevent an agency from purposefully harassing clients in serious debt. The Fair Packaging and Labeling Act makes certain that protection from misleading packaging of goods is guaranteed to each buyer."}
- char **text9** [400] = {"Business casual is an ambiguously defined Western dress code that is generally considered casual wear but with smart (in the sense of 'well dressed') components of a proper lounge suit from traditional informal wear, adopted for white-collar workplaces. This interpretation typically including dress shirt, necktie, & trousers, but worn with an odd-colored blazer or a sports coat instead."}
- char **text10** [520] = {"Many touch typists also use keyboard shortcuts or hotkeys when typing on a computer. This allows them to edit their document without having to take their hands off the keyboard to use a mouse. An example of a keyboard shortcut is pressing the Ctrl key + the S key to save a document as they type, or the Ctrl key + the Z key to undo a mistake. Many experienced typists can feel or sense when they have made an error & can hit the Backspace key & make the correction with no increase in time between keystrokes."}

Extern variables from main

different variables initialized before needed for this portion

- Music **music**
background music
- int **screenWidth**

- *screen width for graphical operations*
- int **screenHeight**
screen height for graphical operations
- Font **retroFont**
- Font **regularFont**
- Texture2D **cockpitTexture**
- Texture2D **cockpitTextureKeyboard**
Textures for background.

General variables for Test process

- bool **exitTest**
- bool **exitTestProcess**
- char **input** [550] = {}
- int **check**

5.6.1 Detailed Description

Test Mode of Space Type.

Author

Pragalbha Acharya 078bct060.pragalbha@pcampus.edu.np

Bug No known Bug

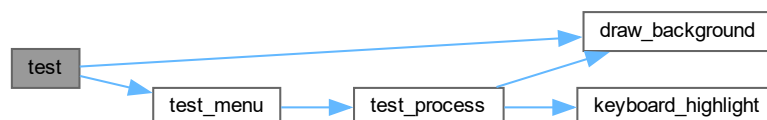
5.6.2 Function Documentation

5.6.2.1 test()

```
void test ( )
```

main function for test mode

Initializes the background texture, continues the background music, draws the background and calls test_menu function for further navigation Here is the call graph for this function:

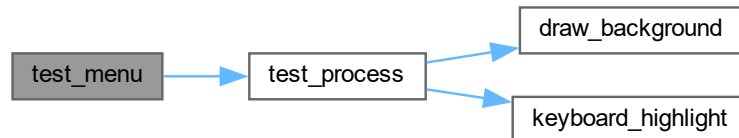


5.6.2.2 test_menu()

```
void test_menu ( )
```

main menu for test screen

Draws the menu for the test mode. Gives the user the option to choose the difficulty that they want their test to be in. Here is the call graph for this function:



5.6.2.3 test_process()

```
void test_process (
    char test_text[] )
```

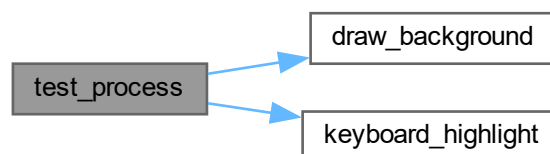
Handles the typing test process.

This function divides the large text into smaller parts, and displays it on the screen to make it easier for the user. Gets the key pressed by the user, checks with the letter from the text, and highlights if its correct. The second part of the text only displayed once the displayed part is entered correctly. Statistics like typing speed, accuracy are shown at the end.

Parameters

<code>test_text</code>	Indicates which passage is to be displayed
------------------------	--

Here is the call graph for this function:



5.6.3 Variable Documentation

5.6.3.1 cockpitTextureKeyboard

Texture2D cockpitTextureKeyboard

Textures for background.

Background Texture.

5.6.3.2 text1

```
char text1[300] = {"Wealth, fame, power. Gold Roger, the King of the Pirates, attained everything  
this world has to offer. And so, many men head for the Grand Line to find the great treasure  
he left behind, the One Piece. The world has truly entered a Great Pirate Era!"}
```

Passages for Test Mode.

These string contains various texts of varying difficulty and types for the test mode.

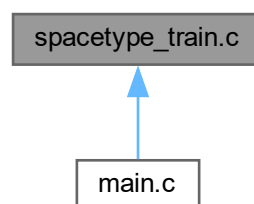
5.7 spacetype_test.h

```
00001 void test_process(char test_text[]);  
00002 void test_menu();  
00003 void test();  
00004
```

5.8 spacetype_train.c File Reference

Word and Letter Typing train mode.

This graph shows which files directly or indirectly include this file:



Functions

- void `train` ()
main function for train mode
- void `train_menu` ()
main menu for train screen
- void `mode_trainletters` ()
Letter Train sub menu.
- void `mode_trainwords` ()
Word Train sub menu.
- void `letter_train` ()
Handles processes regarding training typing letters.
- void `word_train` ()
Handles processes regarding training typing words.
- void `customized_train` ()
Handles processes regarding customized train mode.
- void `train_select` ()
Option menu for choosing rows for both letter mode and word mode.
- void `select_letter` ()
function which selects letter to display for letter train process
- void `select_word` ()
function which selects words to display for word train process

Variables

Letter Train variables

3 character arrays to store characters of respective keyboard rows.

- char **TopRowLetters** [10] = {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'}
- char **MiddleRowLetters** [9] = {'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l'}
- char **BottomRowLetters** [7] = {'z', 'x', 'c', 'v', 'b', 'n', 'm'}
- char **RequiredLetter**
Stores letter which will be displayed in Letter Train process.

Word Train variables

List of Words from toprow, middlerow and bottomrow to choose from when selecting word

- char **TopRowWords** [[10] = {"queer", "wrought", "erode", "trope", "troupe", "youth", "utopia", "irony", "outhouse", "power"}
- char **MiddleRowWords** [[10] = {"lad", "slade", "glass", "fade", "grade", "hall", "jade", "klaus", "lathe"}
- char **BottomRowWords** [[10] = {"zoner", "xerox", "change", "vought", "broom", "noob", "mooncover"}
- char * `customizedWords` []
- char **RequiredWord** [10]

Boolean to control different modes and screen

Different booleans to control training process and also to indicatet which row is chosen for respective train modes

- bool **exitTrainWords**
- bool **exitTrainLetters**
- bool **MiddleRow**
- bool **TopRow**
- bool **BottomRow**
- bool **exitTrainProcess**
- bool **letterinput**

- bool **exitTrain**
- bool **trainMode**
- bool **wordinput**
- bool **exitResult**

Extern variables from main

different variables initialized before that were needed for this portion

- Music **music**
background music
- int **screenWidth**
screen width for graphical operations
- int **screenHeight**
screen height for graphical operations
- Font **retroFont**
- Texture2D **cockpitTexture**
Background Textures.
- Texture2D [cockpitTextureKeyboard](#)
Background Texture.
- char **wordStored** [20]
stores the presented word to later compare with fastestword and slowestword

5.8.1 Detailed Description

Word and Letter Typing train mode.

Author

Mukunda Dev Adhikari 078bct049.mukunda@pcampus.edu.np

Bug No Known Bug

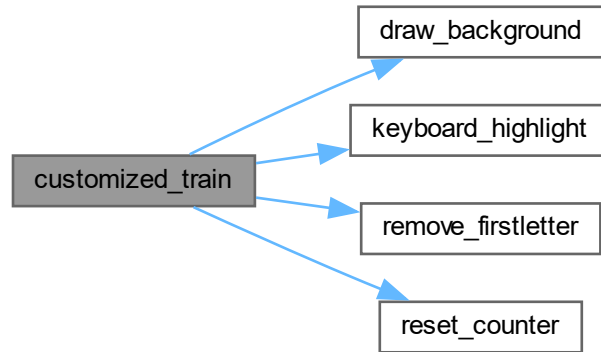
5.8.2 Function Documentation

5.8.2.1 customized_train()

```
void customized_train ( )
```

Handles processes regarding customized train mode.

This function is a child of `word_train`. So, it does everything the `word_train` does but the words it provides to train you are customized and contains the letters that you have typed wrong the most. It is done by storing wrongly typed letters and comparing them with the letters in the word that will be displayed. Here is the call graph for this function:

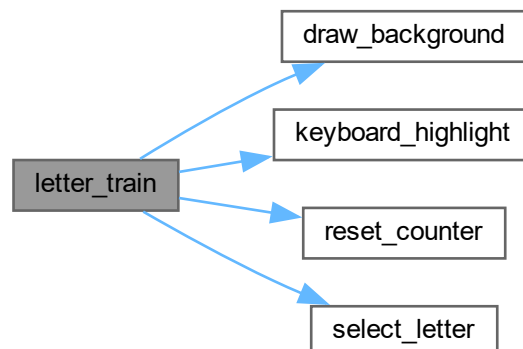


5.8.2.2 letter_train()

```
void letter_train ( )
```

Handles processes regarding training typing letters.

The function that handles resetting of variables by calling `reset_counter`, and contains the loop which selects the new letter (using `select_letter`) and a nested loop to check if the input letter matches with the displayed one. Here is the call graph for this function:

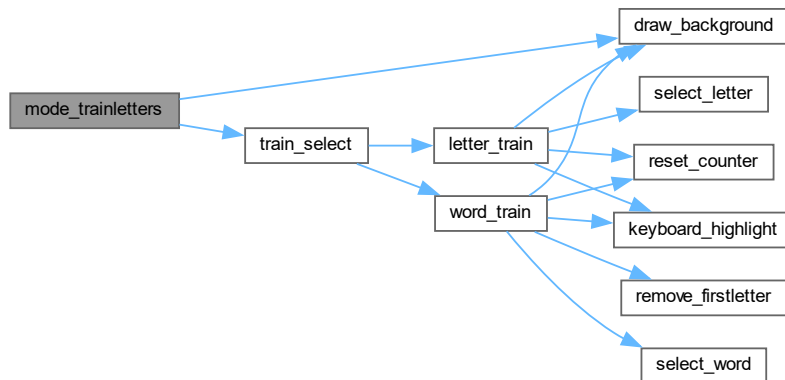


5.8.2.3 mode_trainletters()

```
void mode_trainletters ( )
```

Letter Train sub menu.

function that handles lettere practice mode. Clears all boolean relatetd rows and train loops and calls the function which draws selection menu for rows. Here is the call graph for this function:

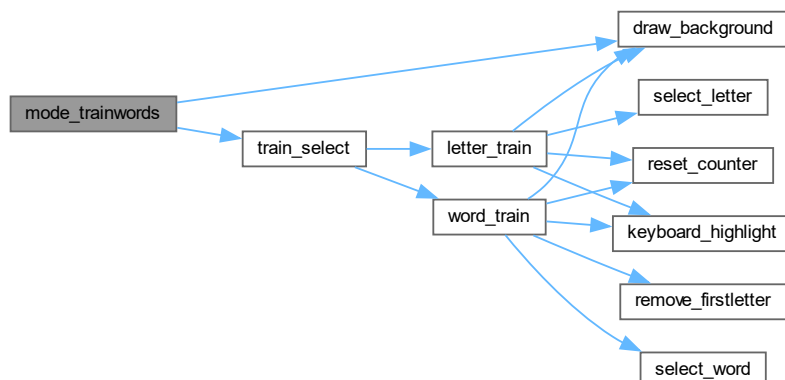


5.8.2.4 mode_trainwords()

```
void mode_trainwords ( )
```

Word Train sub menu.

The function that handles words practice mode. Clears all boolean relatetd rows and train loops and calls the `train_select` function which draws selection menu for rows. Here is the call graph for this function:



5.8.2.5 select_letter()

```
void select_letter ( )
```

function which selects letter to display for letter train process

letter using the the three choices, random number generator and a switch case based on the choice and number generated. Stores that letter in RequiredLetter.

5.8.2.6 select_word()

```
void select_word ( )
```

function which selects words to display for word train process

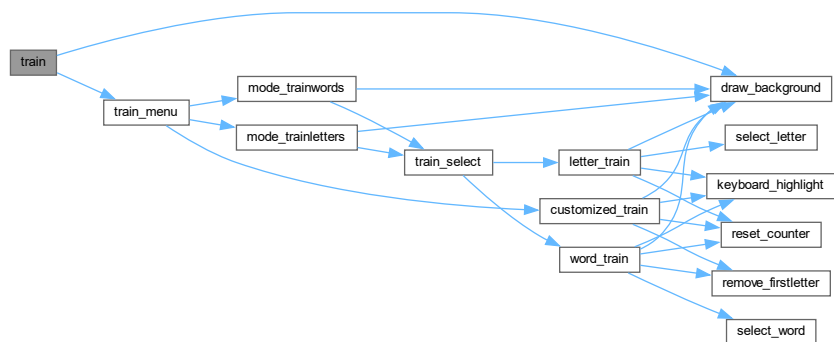
Selects a word using the the three choices, random number generator and a switch case based on the choice and number generated. Stores that word in RequiredWord.

5.8.2.7 train()

```
void train ( )
```

main function for train mode

the background texture, continues the background music and calls train_menu function for further navigation Here is the call graph for this function:

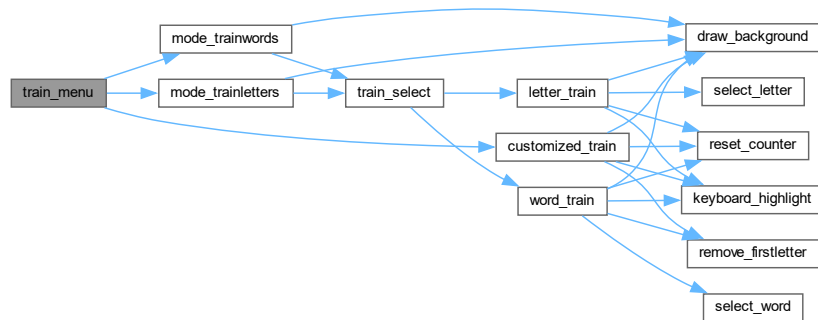


5.8.2.8 train_menu()

```
void train_menu ( )
```

main menu for train screen

Draws the menu for the train mode. Gives the user the option to choose between practicing letters or practicing words. Here is the call graph for this function:

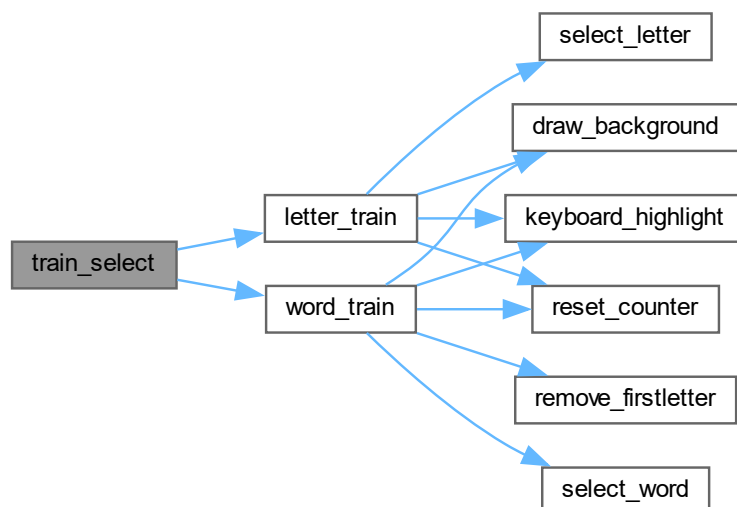


5.8.2.9 train_select()

```
void train_select ( )
```

Option menu for choosing rows for both letter mode and word mode.

This function generates the option menu for the user to choose the rows that he wants to practice. Checkbox indicator are present to indicate the rows chose. letter_train or word_train are called based on the user's previous selection Here is the call graph for this function:

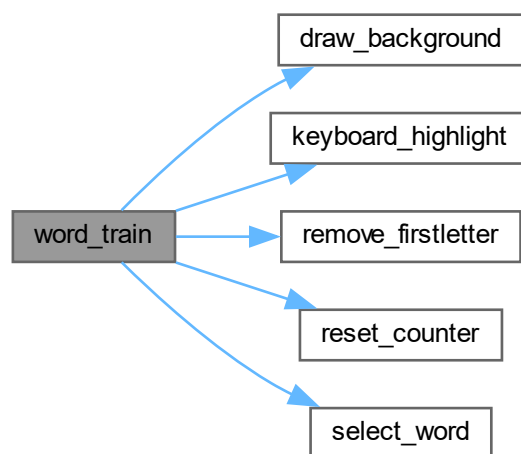


5.8.2.10 word_train()

```
void word_train ( )
```

Handles processes regarding training typing words.

The function that handles resetting of variables by calling `reset_counter`, and contains the loop which selects the new word (using `select_word`) and a nested loop which ends when the user has finished entering the word. It calls `remove_firstletter` each time the letter input matches the letter displayed. Once the user chooses to leave the train process, by pressing ESC, a stats screen is displayed which shows their WPM, fastest word, slowest word, etc. Here is the call graph for this function:



5.8.3 Variable Documentation

5.8.3.1 cockpitTextureKeyboard

```
Texture2D cockpitTextureKeyboard [extern]
```

Background Texture.

Background Texture.

5.8.3.2 customizedWords

```
char* customizedWords[ ]
```

Initial value:

```
=
{
    "abstract", "conjecture", "elixir", "fervent", "gargantuan", "haphazard", "intrepid", "jubilant",
    "kinetic", "luminous", "maverick", "nocturnal", "orchid", "predator", "quagmire", "resilient",
    "saunter", "turbulent", "unwavering", "vortex", "whimsical", "xenon", "yellow", "zephyr", "allegory",
    "benevolent", "credence", "demeanor", "enigma", "fractal", "gusto", "hiatus", "intricacy",
    "jubilation", "kaleidoscope", "lucid", "magnitude", "nimble", "opulence", "pertinent",
    "quintessential", "responsive", "saturate", "tenacity", "unbridled", "volatile", "whirlwind",
    "xylophone", "yacht", "zodiac", "acumen", "bazaar", "clarity", "diligent", "empathy", "flourish",
    "graceful", "harmony", "intuition", "jovial", "klutz", "leverage", "mystique", "nostalgia",
    "overture", "persistence", "quirk", "radiance", "savvy", "transcend", "unison", "vivid", "whisper",
    "xylograph", "yearning", "zephyr", "affinity", "bucolic", "clandestine", "disparate", "embellish",
    "fluctuate", "glossary", "hiatus", "intrepid", "jubilant", "knick-knack", "legerdemain",
    "magnanimous", "nihilistic", "opulent", "provocative", "quintessence", "rhapsodic", "solitude",
    "tenacity", "unabashed", "versatility", "whirlwind", "xenophobe", "yen"
}
```

5.9 spacetype_train.h

```
00001 void train();
00002 void train_menu();
00003 void mode_trainletters();
00004 void mode_trainwords();
00005 void letter_train();
00006 void customized_train();
00007 void select_letter();
00008 void select_word();
00009 void train_select();
00010 void customized_train();
```

Index

- character
 - charCount, [7](#)
- charCount, [7](#)
 - character, [7](#)
 - count, [7](#)
- cockpitTextureKeyboard
 - spacetype_test.c, [24](#)
 - spacetype_train.c, [31](#)
- count
 - charCount, [7](#)
- customized_train
 - spacetype_train.c, [26](#)
- customizedWords
 - spacetype_train.c, [31](#)
- draw_background
 - spacetype_functions.c, [14](#)
- draw_menu
 - main.c, [10](#)
- game
 - spacetype_game.c, [18](#)
- keyboard_highlight
 - spacetype_functions.c, [14](#)
- letter_train
 - spacetype_train.c, [27](#)
- main
 - main.c, [11](#)
- main.c, [9](#)
 - draw_menu, [10](#)
 - main, [11](#)
- mode_trainletters
 - spacetype_train.c, [27](#)
- mode_trainwords
 - spacetype_train.c, [28](#)
- pause_screen
 - spacetype_functions.c, [15](#)
- remove_firstletter
 - spacetype_functions.c, [15](#)
- reset_counter
 - spacetype_functions.c, [15](#)
- select_letter
 - spacetype_train.c, [28](#)
- select_word
 - spacetype_train.c, [29](#)
- spacetype_functions.c, [12](#)
 - draw_background, [14](#)
 - keyboard_highlight, [14](#)
 - pause_screen, [15](#)
 - remove_firstletter, [15](#)
 - reset_counter, [15](#)
 - tutorial_screen, [15](#)
- spacetype_game.c, [16](#)
 - game, [18](#)
 - text_mover, [19](#)
 - words, [19](#)
- spacetype_test.c, [20](#)
 - cockpitTextureKeyboard, [24](#)
 - test, [22](#)
 - test_menu, [22](#)
 - test_process, [23](#)
 - text1, [24](#)
- spacetype_train.c, [24](#)
 - cockpitTextureKeyboard, [31](#)
 - customized_train, [26](#)
 - customizedWords, [31](#)
 - letter_train, [27](#)
 - mode_trainletters, [27](#)
 - mode_trainwords, [28](#)
 - select_letter, [28](#)
 - select_word, [29](#)
 - train, [29](#)
 - train_menu, [29](#)
 - train_select, [30](#)
 - word_train, [31](#)
- test
 - spacetype_test.c, [22](#)
- test_menu
 - spacetype_test.c, [22](#)
- test_process
 - spacetype_test.c, [23](#)
- text1
 - spacetype_test.c, [24](#)
- text_mover
 - spacetype_game.c, [19](#)
- train
 - spacetype_train.c, [29](#)
- train_menu
 - spacetype_train.c, [29](#)
- train_select
 - spacetype_train.c, [30](#)
- tutorial_screen
 - spacetype_functions.c, [15](#)

word_train

 spacetype_train.c, [31](#)

words

 spacetype_game.c, [19](#)