Space Type

1.0

Generated by Doxygen 1.9.6

Chapter 1

Bug List

File main.c (p. **9**)

: No known Bug.

File spacetype_functions.c (p. **14**)

No Known Bugs

File spacetype_game.c (p. **26**)

No known bug

File spacetype_test.c (p. **35**)

No known Bug

File spacetype_train.c (p. **44**)

No Known Bug

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 charCount Struct Reference

This structure stores mistyped characters and their frequency.

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

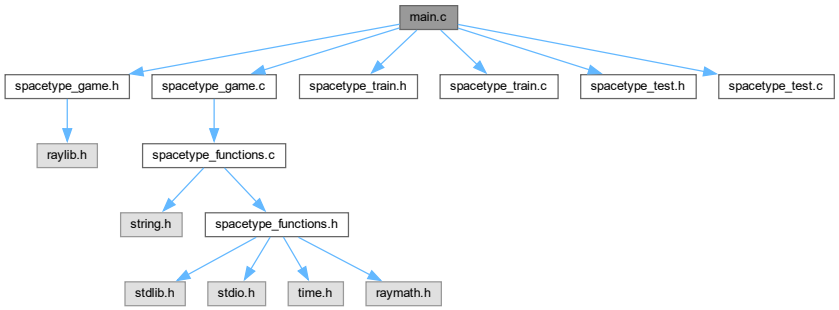
File Documentation

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

- void **draw_menu** ()

  Draws main menu for the application.

- int **main** ()

  Main function of 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 inspration from Typeshala, Ztype and aims for the feels similar to retro-console space games like Space Invadors. 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 078bct049.pragalbha@pcampus.edu.np

Bug : No known Bug.

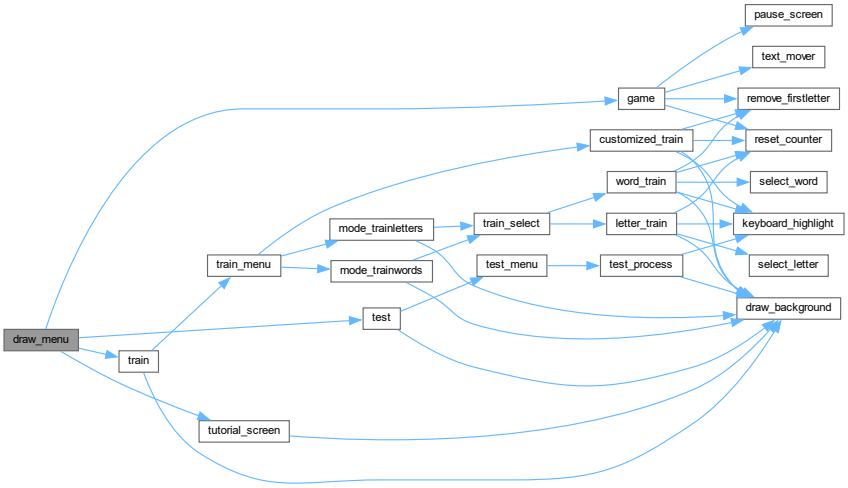
5.1.2 Function Documentation

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:

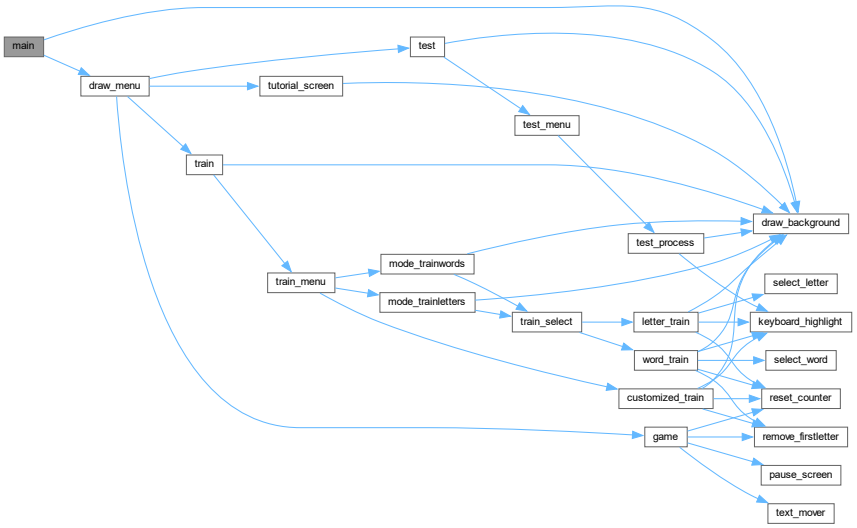


5.1.2.2 main()

```
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:



5.1.3 Variable Documentation

5.1.3.1 cockpitTexture

```
Texture2D cockpitTexture
```

Background Textures.

5.1.3.2 cockpitTextureExit

```
Texture2D cockpitTextureExit
```

Main menu textures.

5.1.3.3 cockpitTextureGame

```
Texture2D cockpitTextureGame
```

5.1.3.4 cockpitTextureTest

```
Texture2D cockpitTextureTest
```

5.1.3.5 cockpitTextureTrain

```
Texture2D cockpitTextureTrain
```

5.1.3.6 exitGame

```
bool exitGame = false
```

To govern game mode loop.

5.1.3.7 exitWindow

```
bool exitWindow = false
```

To govern main application loop.

5.1.3.8 qwertyTexture

```
Texture2D qwertyTexture
```

keyboard image for tutorial screen

5.1.3.9 regularFont

```
Font regularFont
```

Main application fonts.

5.1.3.10 retroFont

```
Font retroFont
```

5.1.3.11 screenHeight

```
int screenHeight
```

screen height for graphical operations

5.1.3.12 screenWidth

```
int screenWidth
```

screen width for graphical operations

5.1.3.13 sorted

```
bool sorted
```

to check if the WrongChars.txt is sorted or not

5.1.3.14 wrongChars

```
FILE* wrongChars
```

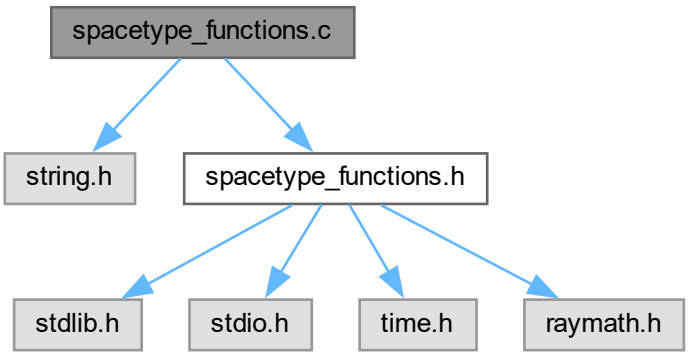
To store mispressed characters to use in customized train mode.

5.2 spacetype_functions.c File Reference

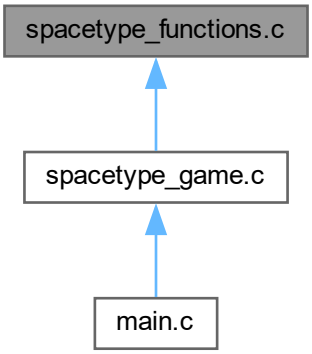
Created functions for this application.

```
#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)

  hightlights requried letter
- void **tutorial_screen** ()

  Shows a tutorial screeen 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 **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 **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 )
```

hightlights requried letter

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

Parameters

a	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

word[]	the word sent to have its first letter removed

5.2.2.5 reset_counter()

```
void reset_counter ( )
```

resets variables for required statistics

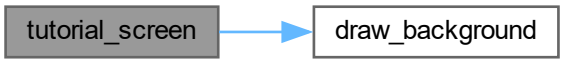
all the variables like score, fastestword, slowest word, etc. that are used to calculate statistics during word practice and Game mode

5.2.2.6 tutorial_screen()

```
void tutorial_screen ( )
```

Shows a tutorial screeen 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:



5.2.3 Variable Documentation

5.2.3.1 bulletTexture

```
Texture2D bulletTexture
```

Texture of Bullet used in game mode.

5.2.3.2 cockpitTextureKeyboard

```
Texture2D cockpitTextureKeyboard
```

Background Texture.

5.2.3.3 exitGame

```
bool exitGame  [extern]
```

To govern game mode loop.

5.2.3.4 exitPause

```
bool exitPause = true
```

5.2.3.5 fastestWord

```
char fastestWord[20]
```

5.2.3.6 fastestWordFrames

```
float fastestWordFrames
```

5.2.3.7 framesCounterForSession

```
float framesCounterForSession
```

5.2.3.8 framesCounterForWord

```
float framesCounterForWord
```

5.2.3.9 gapMeasured

```
bool gapMeasured
```

5.2.3.10 keysPressed

```
float keysPressed
```

5.2.3.11 mover

```
float mover
```

Variable which controls the speed of the word in game mode.

5.2.3.12 movingDown

```
float movingDown
```

Variable to govern infinitely scrolling background.

5.2.3.13 music

```
Music music
```

background music

5.2.3.14 planetTextures

```
Texture2D planetTextures[3]
```

Array of 3 different planet images to display on background.

5.2.3.15 qwertyTexture

```
Texture2D qwertyTexture  [extern]
```

keyboard image for tutorial screen

5.2.3.16 retroFont

```
Font retroFont  [extern]
```

5.2.3.17 rightKeysPressed

```
float rightKeysPressed
```

5.2.3.18 scale

```
float scale
```

Image scale for the uniformity.

5.2.3.19 SCORE

```
int SCORE
```

5.2.3.20 scoreString

```
char scoreString[50]
```

5.2.3.21 screenHeight

```
int screenHeight  [extern]
```

screen height for graphical operations

5.2.3.22 screenWidth

```
int screenWidth  [extern]
```

screen width for graphical operations

5.2.3.23 slowestWord

```
char slowestWord[20]
```

5.2.3.24 slowestWordFrames

```
float slowestWordFrames
```

5.2.3.25 sorted

```
bool sorted
```

5.2.3.26 spaceshipTexture

```
Texture2D spaceshipTexture
```

Texture of spaceship in game mode.

5.2.3.27 spaceTexture

```
Texture2D spaceTexture
```

Space Background used in infinite scroll background.

5.2.3.28 TIME

```
float TIME = 10
```

5.2.3.29 wrongChars

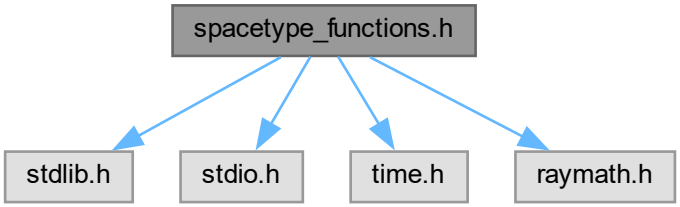
```
FILE* wrongChars  [extern]
```

To store mispressed characters to use in customized train mode.

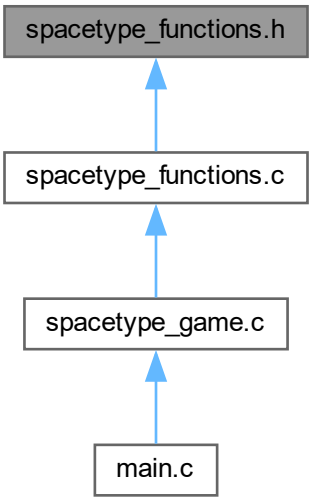
5.3 spacetype_functions.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <raymath.h>
```

Include dependency graph for spacetype_functions.h:



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



Data Structures

- struct **charCount**

  This structure stores mistyped characters and their frequency.

Functions

- void **draw_background** ()

  Draws background in all modes.
- void **reset_counter** ()

  resets variables for required statistics
- void **remove_firstletter** (char **word**[ ])

  Removes the first letter of the word sent.
- void **pause_screen** ()

  Creates a pause menu.
- void **tutorial_screen** ()

  Shows a tutorial screeen for touch typing.
- void **keyboard_highlight** (char a)

  hightlights requried letter

Variables

- struct **charCount** **wrongLetters** [26]

5.3.1 Function Documentation

5.3.1.1 draw_background()

```
void draw_background ( )
```

Draws background in all modes.

Draws the infinitely scrolling space background with moving planets.

5.3.1.2 keyboard_highlight()

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

hightlights requried letter

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

Parameters

a	the letter which needs to be highlighted

5.3.1.3 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

5.3.1.4 remove_firstletter()

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

Removes the first letter of the word sent.

Parameters

word[ ]	the word sent to have its first letter removed

5.3.1.5 reset_counter()

```
void reset_counter ( )
```

resets variables for required statistics

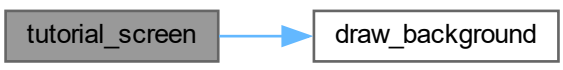
all the variables like score, fastestword, slowest word, etc. that are used to calculate statistics during word practice and Game mode

5.3.1.6 tutorial_screen()

```
void tutorial_screen ( )
```

Shows a tutorial screeen 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:



5.3.2 Variable Documentation

5.3.2.1 wrongLetters

```
struct  charCount wrongLetters[26]
```

5.4 spacetype_functions.h

Go to the documentation of this file.

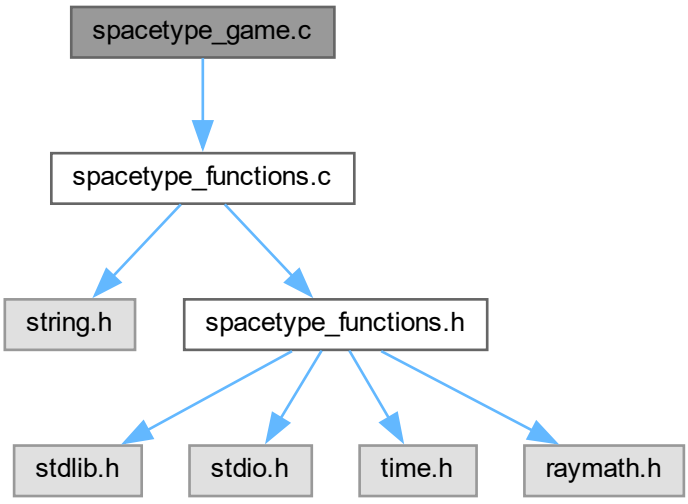
```
00001
00002 #include <stdlib.h>
00003 #include <stdio.h>
00004 #include <time.h>
00005 #include <raymath.h>
00006
00007 void draw_background();
00008 void reset_counter();
00009 void remove_firstletter(char word[]);
00010 void pause_screen();
00011 void tutorial_screen();
00012 void keyboard_highlight (char a);
00020 struct charCount{
00021     char character;
00022     int count;
00023 } wrongLetters[26];
```

5.5 spacetype_game.c File Reference

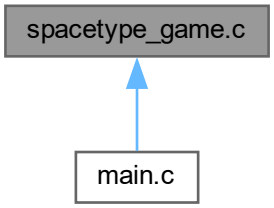
Word Shooter Game for Spacetype.

```
#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**

5.5.1 Detailed Description

Word Shooter Game for Spacetype.

Author

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

Bug No known bug

5.5.2 Function Documentation

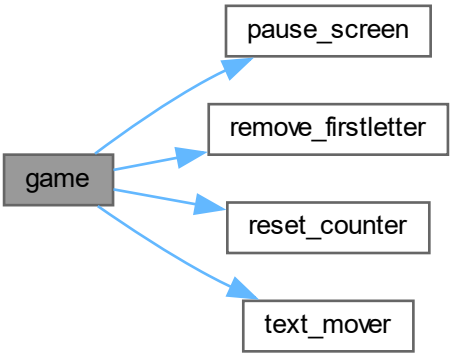
5.5.2.1 game()

```
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:



5.5.2.2 text_mover()

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

Moves the word toward the player.

Parameters

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

5.5.3 Variable Documentation

5.5.3.1 angle

```
float angle = 0
```

Angle of word to have it fall towards player.

5.5.3.2 bullet

```
bool bullet = false
```

indicates whether bullet needs to be shoot or not

5.5.3.3 bulletAngle

```
float bulletAngle
```

Determine angle at which bullet needs to be thrown depending word position.

5.5.3.4 bulletMover

```
float bulletMover = 0
```

To indicate speed of the bullet.

5.5.3.5 bulletPos

```
Rectangle bulletPos
```

Bullet Position.

5.5.3.6 fontSize

```
const int fontSize = 25
```

Fonsize declaration for uniformity of fontsize.

5.5.3.7 GAME_OVER

```
bool GAME_OVER = false
```

boolean to check game over condition

5.5.3.8 gap

```
Vector2 gap = {}
```

Shortest distance between word and spaceship∗/.

5.5.3.9 movingDown

```
float movingDown = 0
```

Parameters which makes space texture scroll infinitely∗/.

5.5.3.10 movingPlanets

```
float movingPlanets = 0
```

Denotes which planet will be on screen.

5.5.3.11 music

```
Music music  [extern]
```

background music

5.5.3.12 prevAngle

```
float prevAngle = 0
```

Angle of the spaceship.

5.5.3.13 retroFont

```
Font retroFont  [extern]
```

5.5.3.14 screenHeight

```
int screenHeight  [extern]
```

screen height for graphical operations

5.5.3.15 screenWidth

```
int screenWidth  [extern]
```

screen width for graphical operations

5.5.3.16 shoot

```
Sound shoot
```

Audio played when bullet is used.

5.5.3.17 sizeOfArray

```
int sizeOfArray = sizeof( words) / sizeof( words[0])
```

for calculating total number of words∗/

5.5.3.18 word

```
char word[20]
```

word which is presented to type

5.5.3.19 wordPos

```
Vector2 wordPos
```

position of word as it falls down

5.5.3.20 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.

5.5.3.21 wordStored

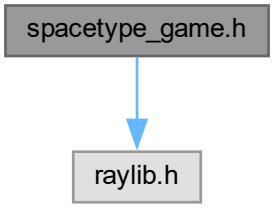
```
char wordStored[20]
```

stores the presented word to later compare with fastestword and slowestword

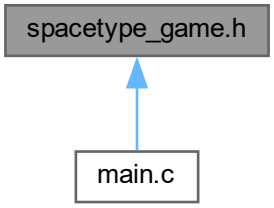
5.6 spacetype_game.h File Reference

```
#include <raylib.h>
```

Include dependency graph for spacetype_game.h:



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



Functions

- void **game** ()

  main function for game mode
- void **text_mover** (Vector2 * **wordPos**, Rectangle playerPos, Vector2 **gap**, float time)

  Moves the word toward the player.

5.6.1 Function Documentation

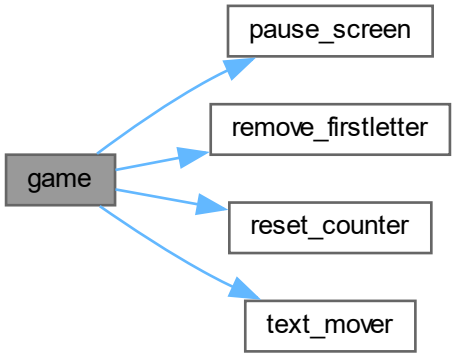
5.6.1.1 game()

```
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:



5.6.1.2 text_mover()

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

Moves the word toward the player.

Parameters

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

5.7 spacetype_game.h

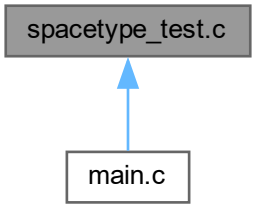
Go to the documentation of this file.

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

5.8 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** [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."}

  Passages for Test Mode.
- char **text2** [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 **text3** [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 **text4** [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 **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** [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 **text7** [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."}
- char **text8** [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 **text9** [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!"}
- char **text10** [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."}

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**

5.8.1 Detailed Description

Test Mode of Space Type.

Author

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

Bug No known Bug

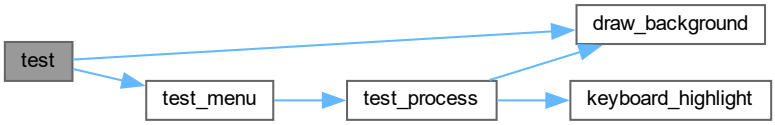
5.8.2 Function Documentation

5.8.2.1 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:

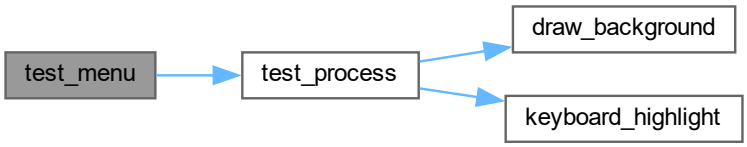


5.8.2.2 test_menu()

```
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:



5.8.2.3 test_process()

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

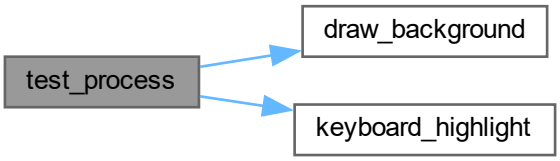
Handles the typing test process.

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 enterred correctly. Statistics like typing speed, accuracy are shown at the end.

Parameters

test_test	Indicates which passage is to be displayed

Here is the call graph for this function:



5.8.3 Variable Documentation

5.8.3.1 check

```
int check
```

5.8.3.2 cockpitTexture

```
Texture2D cockpitTexture  [extern]
```

5.8.3.3 cockpitTextureKeyboard

```
Texture2D cockpitTextureKeyboard
```

Textures for background.

Background Texture.

5.8.3.4 exitTest

```
bool exitTest
```

5.8.3.5 exitTestProcess

```
bool exitTestProcess
```

5.8.3.6 input

```
char input[550] = {}
```

5.8.3.7 music

```
Music music  [extern]
```

background music

5.8.3.8 regularFont

```
Font regularFont
```

5.8.3.9 retroFont

```
Font retroFont  [extern]
```

5.8.3.10 screenHeight

```
int screenHeight  [extern]
```

screen height for graphical operations

5.8.3.11 screenWidth

```
int screenWidth  [extern]
```

screen width for graphical operations

5.8.3.12 text1

```
char text1[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."}
```

Passages for Test Mode.

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

5.8.3.13 text10

```
char text10[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."}
```

5.8.3.14 text2

```
char text2[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."}
```

5.8.3.15 text3

```
char text3[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."}
```

5.8.3.16 text4

```
char text4[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."}
```

5.8.3.17 text5

```
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."}
```

5.8.3.18 text6

```
char text6[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."}
```

5.8.3.19 text7

```
char text7[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."}
```

5.8.3.20 text8

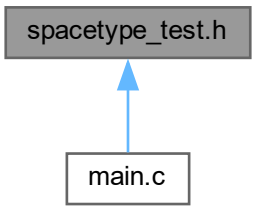
```
char text8[438] = {"When we talk about motivating others, the justification is the end result
(either we want to avoid the pain or go towards pleasure) or what we want to get the person to
do.  How we achieve the end result, are our alternatives.  As a manager, we need to understand
the other person's 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."}
```

5.8.3.21 text9

```
char text9[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!"}
```

5.9 spacetype_test.h File Reference

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



Functions

- void **test_process** (char test_text[ ])

  Handles the typing test process.
- void **test_menu** ()

  main menu for test screen
- void **test** ()

  main function for test mode

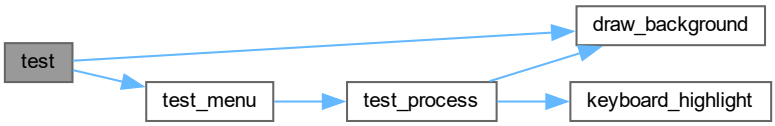
5.9.1 Function Documentation

5.9.1.1 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:

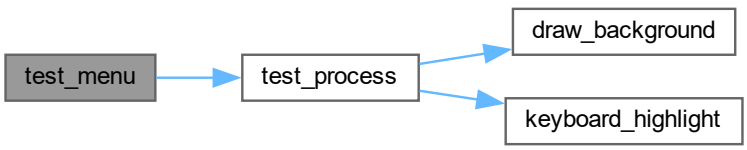


5.9.1.2 test_menu()

```
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:



5.9.1.3 test_process()

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

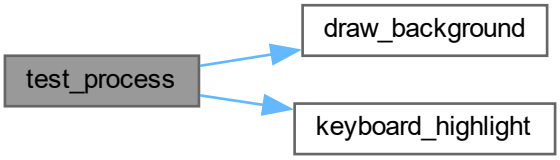
Handles the typing test process.

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 enterred correctly. Statistics like typing speed, accuracy are shown at the end.

Parameters

test_test	Indicates which passage is to be displayed

Here is the call graph for this function:



5.10 spacetype_test.h

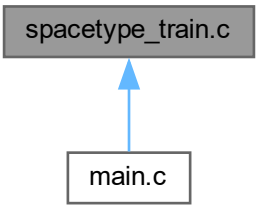
Go to the documentation of this file.

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

5.11 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 cuztomized 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

5.11.1 Detailed Description

Word and Letter Typing train mode.

Author

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

Bug No Known Bug

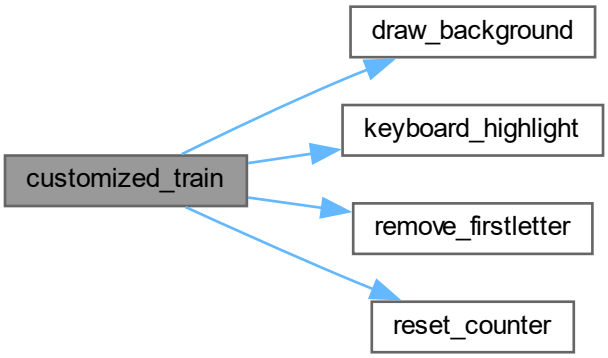
5.11.2 Function Documentation

5.11.2.1 customized_train()

```
void customized_train ( )
```

Handles processes regarding cuztomized 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:

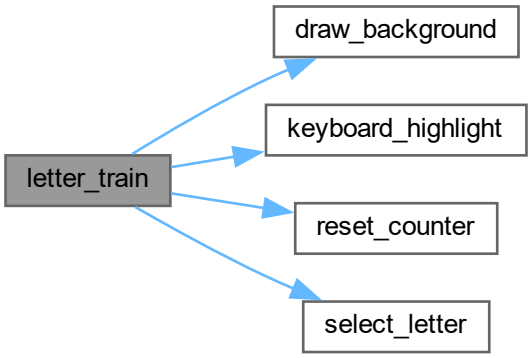


5.11.2.2 letter_train()

```
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:

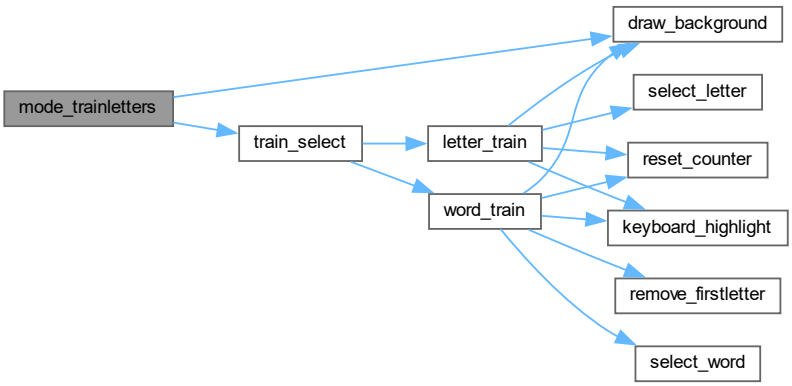


5.11.2.3 mode_trainletters()

```
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:

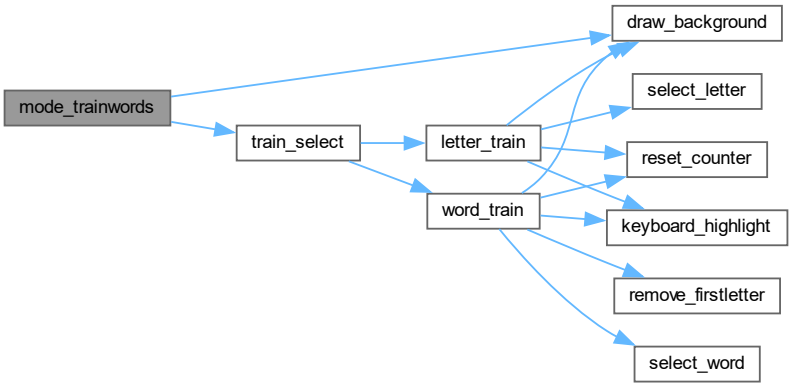


5.11.2.4 mode_trainwords()

```
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:



5.11.2.5 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.

5.11.2.6 select_word()

```
void select_word ( )
```

function which selects words to display for word train process

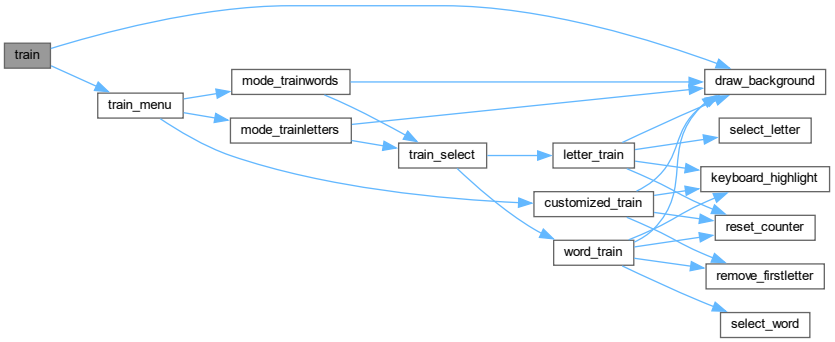
Selects a word using the the three choices, random number generator and a switch case based on the choice and number generated. Stores that word in RequiredWord.

5.11.2.7 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:

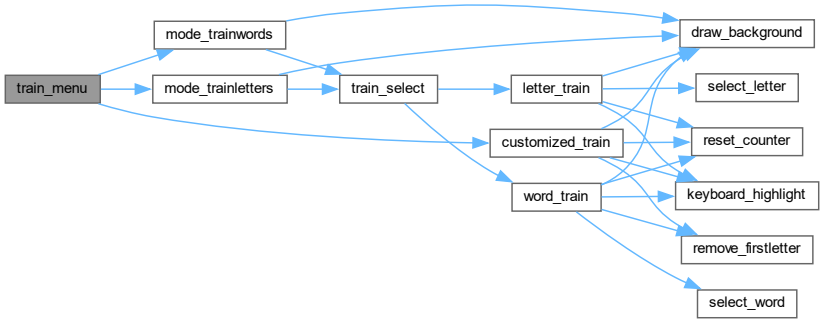


5.11.2.8 train_menu()

```
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:

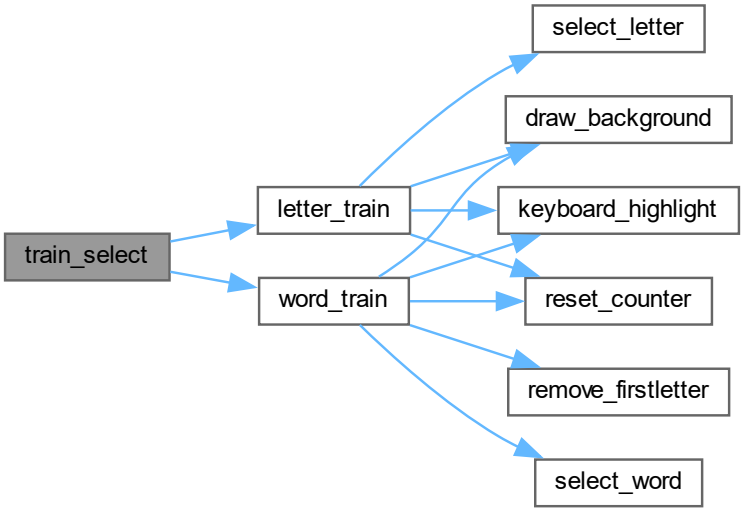


5.11.2.9 train_select()

```
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:

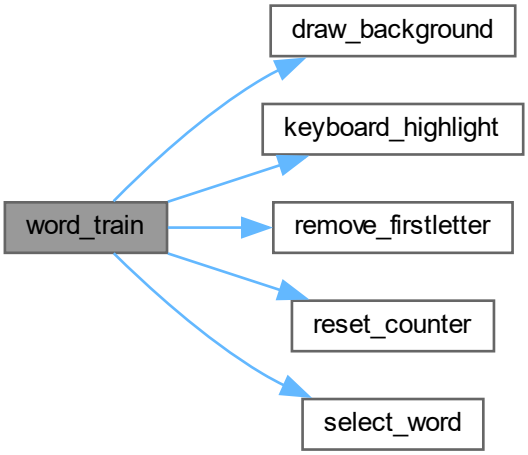


5.11.2.10 word_train()

```
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 enterring 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:



5.11.3 Variable Documentation

5.11.3.1 BottomRow

```
bool BottomRow
```

5.11.3.2 BottomRowLetters

```
char BottomRowLetters[7] = {'z', 'x', 'c', 'v', 'b', 'n', 'm'}
```

5.11.3.3 BottomRowWords

```
char BottomRowWords[][10] = {"zoner", "xerox", "change", "vought", "broom", "noob", "mooncover"}
```

5.11.3.4 cockpitTexture

```
Texture2D cockpitTexture  [extern]
```

Background Textures.

5.11.3.5 cockpitTextureKeyboard

```
Texture2D cockpitTextureKeyboard  [extern]
```

Background Texture.

Background Texture.

5.11.3.6 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"
    }
```

5.11.3.7 exitResult

```
bool exitResult
```

5.11.3.8 exitTrain

```
bool exitTrain
```

5.11.3.9 exitTrainLetters

```
bool exitTrainLetters
```

5.11.3.10 exitTrainProcess

```
bool exitTrainProcess
```

5.11.3.11 exitTrainWords

```
bool exitTrainWords
```

5.11.3.12 letterinput

```
bool letterinput
```

5.11.3.13 MiddleRow

```
bool MiddleRow
```

5.11.3.14 MiddleRowLetters

```
char MiddleRowLetters[9] = {'a', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l'}
```

5.11.3.15 MiddleRowWords

```
char MiddleRowWords[][10] = {"lad", "slade", "glass", "fade", "grade", "hall", "jade", "klaus",
"lathe"}
```

5.11.3.16 music

```
Music music  [extern]
```

background music

5.11.3.17 RequiredLetter

```
char RequiredLetter
```

Stores letter which will be displayed in Letter Train process.

5.11.3.18 RequiredWord

```
char RequiredWord[10]
```

5.11.3.19 retroFont

```
Font retroFont  [extern]
```

5.11.3.20 screenHeight

```
int screenHeight  [extern]
```

screen height for graphical operations

5.11.3.21 screenWidth

```
int screenWidth  [extern]
```

screen width for graphical operations

5.11.3.22 TopRow

```
bool TopRow
```

5.11.3.23 TopRowLetters

```
char TopRowLetters[10] = {'q', 'w', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'}
```

5.11.3.24 TopRowWords

```
char TopRowWords[][10] = {"queer", "wrought", "erode", "trope", "troupe", "youth", "utopia",
"irony", "outhouse", "power"}
```

5.11.3.25 trainMode

```
bool trainMode
```

5.11.3.26 wordinput

```
bool wordinput
```

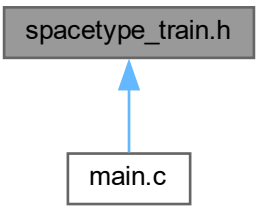
5.11.3.27 wordStored

```
char wordStored[20]  [extern]
```

stores the presented word to later compare with fastestword and slowestword

5.12 spacetype_train.h File Reference

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 **customized_train** ()
  Handles processes regarding cuztomized train mode.
- void **letter_print** ()
- 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
- void **start_trainword** ()
- void **train_select** ()
  Option menu for choosing rows for both letter mode and word mode.

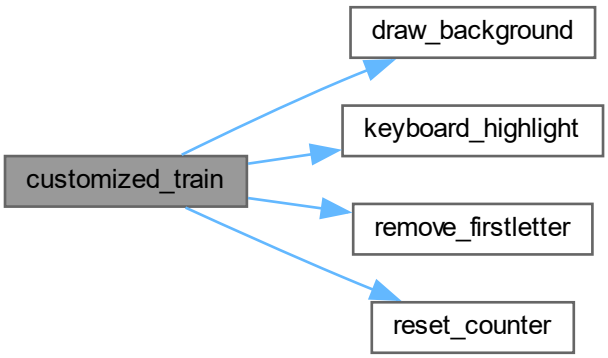
5.12.1 Function Documentation

5.12.1.1 customized_train()

```
void customized_train ( )
```

Handles processes regarding cuztomized 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:



5.12.1.2 letter_print()

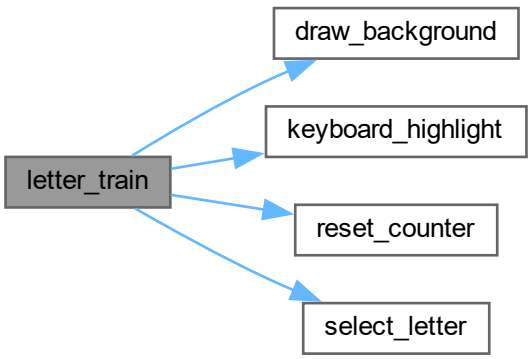
```
void letter_print ( )
```

5.12.1.3 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:

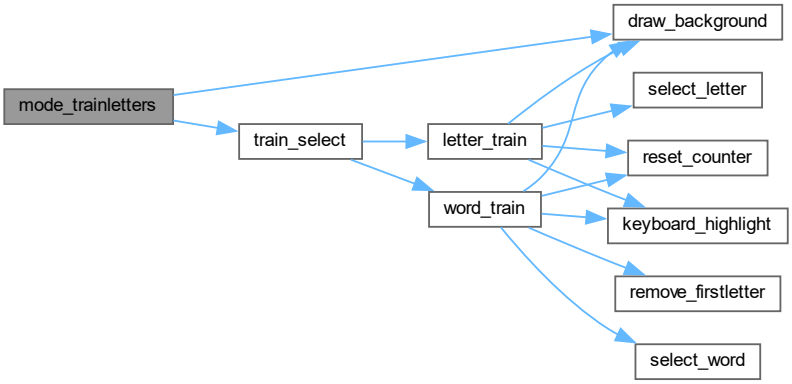


5.12.1.4 mode_trainletters()

```
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:

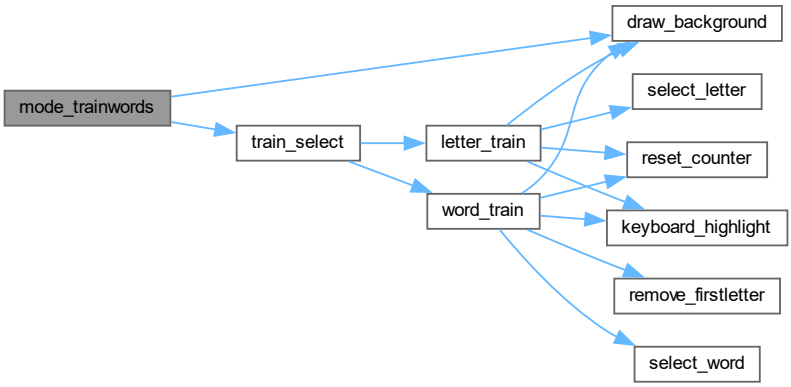


5.12.1.5 mode_trainwords()

```
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:



5.12.1.6 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.

5.12.1.7 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.

5.12.1.8 start_trainword()

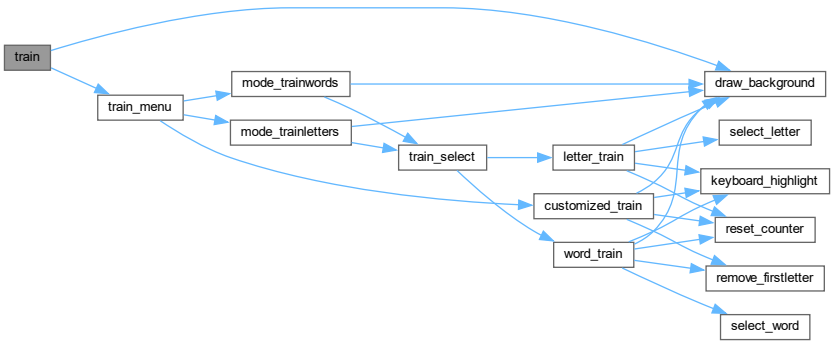
```
void start_trainword ( )
```

5.12.1.9 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:

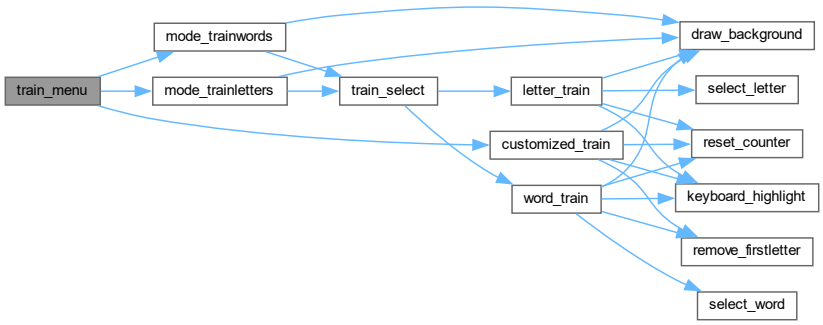


5.12.1.10 train_menu()

```
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:

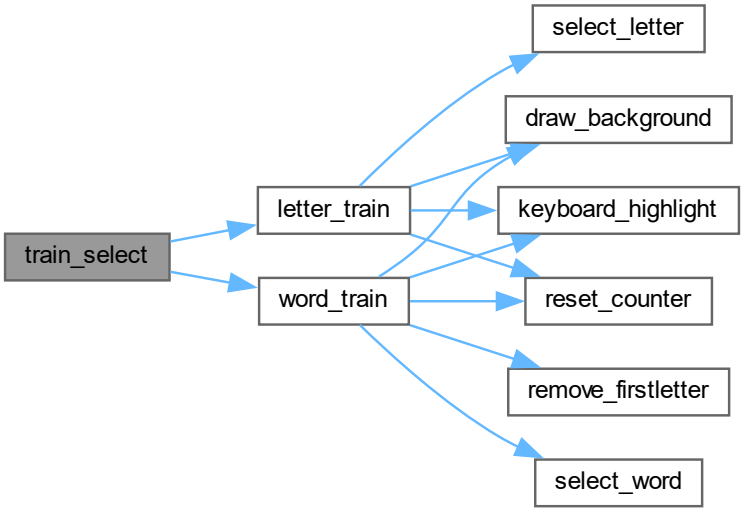


5.12.1.11 train_select()

```
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:



5.13 spacetype_train.h

Go to the documentation of this file.

```
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 letter_print();
00008 void select_letter();
00009 void select_word();
00010 void start_trainword();
00011 void train_select();
00012 void customized_train();
```

Index