Dungeon Crawler

1.0

Generated by Doxygen 1.6.3

Thu Jun 3 17:50:55 2010

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

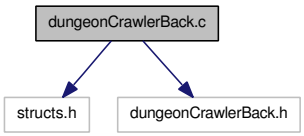
Chapter 2

File Documentation

2.1 dungeonCrawlerBack.c File Reference

```
#include "structs.h"
#include "dungeonCrawlerBack.h"
```

Include dependency graph for dungeonCrawlerBack.c:



Defines

- #define BLOCK 5

Functions

- static int readsImportantPoints (FILE *file, gameDataType *data)

  Reads the important points section. Saves corresponding data in the struct.

- static int validatesText (FILE *file, char *text)

  Reads from File the ammount of characters that where passed by parameter to the function. Then compares the read text with the parameter.

- static int readsProfession (gameDataType *data, FILE *file, int i)

  Reads ONE profession. Saves corresponding data into struct.

- static char * readsDescription (FILE *file)

  Reads text from file until it finds a "<".

- static int readEnemysID (FILE *file)

  Reads and expects a number between <ID> and </ID>.

- static int readsAllRooms (gameDataType *data, FILE *file)

  Reads all rooms. This means that it will only read the ammount of existing rooms in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual room.

- static int readsRooms (gameDataType *data, FILE *file, int i)

  Reads ONE room. Saves read data into corresponding struct.

- static int validatesEnemys (gameDataType *data, FILE *file, int i)

  Reads the enemys present in a room.

- static int validatesDoors (gameDataType *data, int *doorcounter, FILE *file, int i)

  Reads all the doors a room can have.

- static int validatesDestiny (gameDataType *data)

  Validates that the destiny of the doors in a room EXISTS.

- static int readsAllProfessions (gameDataType *data, FILE *file)

  Reads all professions. This means that it will only read the ammount of existing professions in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual profession.

- static int findsRepeatedId (int ID, int arraylength, gameDataType *data, char *check)

  Checks if the ID given is present in the given field.

- static int searchesDestiny (gameDataType *data, int dest)

  Loos up for a given ID in the room's array to verify it exists.

- static int readsEnemys (FILE *file, gameDataType *data, int i)

  Rads ONE enemy.

- static int validatesMinAndMaxDP (gameDataType *data, FILE *file, int i)

  Reads all the "<MinDP-X>%d</MinDP-X>" that are present in an enemy's definition.

- static int readsAllEnemys (FILE *file, gameDataType *data)

  Reads ALL enemys present. This means that it will only read the ammount of existing enemys in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual enemy.

- static int validatesInicialEndRoom (gameDataType *data, int startRoom, int finalRoom)

  Validates that the inicial and end room exist. Variables memoryinicial and memoryfinal do the same. They both save the position in the room's vector in which the searched room is located.

- static void freeMonsterToProfessionDamage (gameDataType *data)

  Frees memory.

- static void freeMonsters (gameDataType *data)
- static void freeProfessions (gameDataType *data)
- static void freeRooms (gameDataType *data)
- static void defineMonsterHP (gameDataType *data, int monsterPosition)

Defines monster's HP.

- static void fightMonster (avatarType ∗avatar, gameDataType ∗data, int actRoom, int(∗waitForEnter)(void), void(∗printHPChanges)(int, gameDataType ∗, avatarType ∗, int, int, int), int enemyQty, int avatarAttacks)
- int creates3DMatrix (int numOfEnemys, int ∗∗∗monsterToProfessionDamage, gameDataType ∗data)

  creates 3 dimension matrix to store the damage each enemy does to each proffesion.

- int fight (gameDataType ∗data, avatarType ∗avatar, int(∗waitForEnter)(void), void(∗printHPChanges)(int, gameDataType ∗, avatarType ∗, int, int, int), void(∗printActualMonster)(gameDataType ∗, int monsterPosition))

  Simulates the fight between the Avatar and one enemy.

- int nameToStruct (avatarType ∗avatar, char ∗auxName)
- FILE ∗ opensFile (int argc, char ∗∗argv, unsigned int ∗seed)
- int saveRoom (gameDataType ∗data, int actualRoom)
- int saveDoor (gameDataType ∗data, int chosenDoor)
- float randomTurn ()
- int randomNumber (int top, int bottom)
- void defineHP (avatarType ∗avatar, gameDataType ∗data)
- void dumpActions (gameDataType ∗data, char ∗fileName)
- void inicializeDumpActions (gameDataType ∗data)
- int readsFile (FILE ∗file, gameDataType ∗data)
- void freeMemory (gameDataType ∗data, avatarType ∗avatar)
- int treasure (avatarType ∗avatar, gameDataType ∗data)

  Generates a random number and depending on this number perfoms different accions. It either reduces avatar's HP, increases it or increases the MaxDP.

2.1.1 Define Documentation

2.1.1.1 #define BLOCK 5

Definition at line 4 of file dungeonCrawlerBack.c.

2.1.2 Function Documentation

2.1.2.1 int creates3DMatrix (int *numOfEnemys*, int ∗∗∗ *monsterToProfessionDamage*, gameDataType ∗ *data*)

creates 3 dimension matrix to store the damage each enemy does to each proffesion.

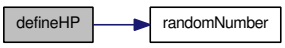
Definition at line 1136 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

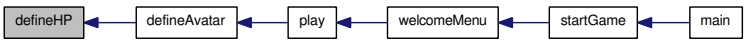
2.1.2.2 void defineHP (avatarType * *avatar*, gameDataType * *data*)

Definition at line 251 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2.3 static void defineMonsterHP (gameDataType * *data*, int *monsterPosition*) [static]

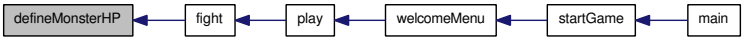
Defines monster's HP.

Receives pointer to data and the position of the struct corresponding to the monster in the Enemy's array

Definition at line 264 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

Here is the caller graph for this function:



2.1.2.4 void dumpActions (gameDataType * *data*, char * *fileName*)

Definition at line 273 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.5 int fight (gameDataType * *data*, avatarType * *avatar*, int(*)(void) *waitForEnter*, void(*)(int, gameDataType *, avatarType *, int, int, int) *printHPChanges*, void(*)(gameDataType *, int monsterPosition) *printActualMonster*)

Simulates the fight between the Avatar and one enemy.

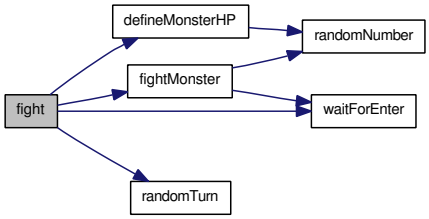
Returns

Returns 0 if the Avatar was killed.
Returns -1 if the Avatar has to fight with another enemy.
Returns any other value if the avatar has defeated all enemys.

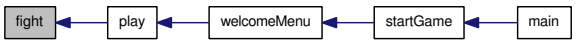
Receives as parameters the functions encharged of printing HP changes during fight and the one encharged of printing the enemys left in the room

Definition at line 49 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



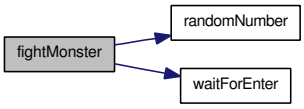
Here is the caller graph for this function:



2.1.2.6 static void fightMonster (avatarType * *avatar*, gameDataType * *data*, int *actRoom*, int(*)(void) *waitForEnter*, void(*)(int, gameDataType *, avatarType *, int, int, int) *printHPChanges*, int *enemyQty*, int *avatarAttacks*) [static]

Definition at line 96 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.1.2.7 static int findsRepeatedId (int *ID*, int *arraylength*, gameDataType * *data*, char * *check*) [static]

Checks if the ID given is present in the given field.

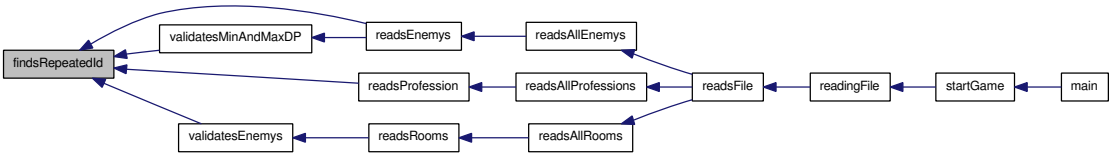
Parameters

check expects either "Monsters" or "Professions"

Returns the position where it was found of -1 if not found

Definition at line 887 of file dungeonCrawlerBack.c.

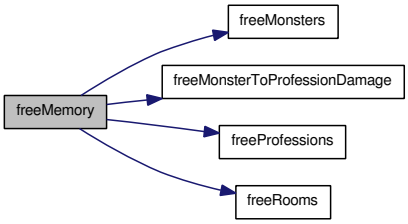
Here is the caller graph for this function:



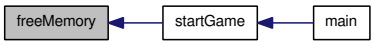
2.1.2.8 void freeMemory (gameDataType * *data*, avatarType * *avatar*)

Definition at line 1224 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2.9 static void freeMonsters (gameDataType * *data*) [static]

Definition at line 1250 of file dungeonCrawlerBack.c.

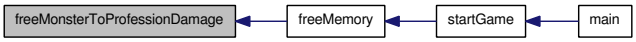
Here is the caller graph for this function:

2.1.2.10 static void freeMonsterToProfessionDamage (gameDataType * *data*) [static]

Frees memory.

Definition at line 1235 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:



2.1.2.11 static void freeProfessions (gameDataType * *data*) [static]

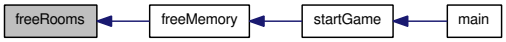
Definition at line 1263 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.12 static void freeRooms (gameDataType * *data*) [static]

Definition at line 1273 of file dungeonCrawlerBack.c.

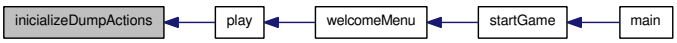
Here is the caller graph for this function:



2.1.2.13 void inicializeDumpActions (gameDataType * *data*)

Definition at line 293 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:



2.1.2.14 int nameToStruct (avatarType * *avatar*, char * *auxName*)

Definition at line 143 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.15 FILE* opensFile (int *argc*, char ** *argv*, unsigned int * *seed*)

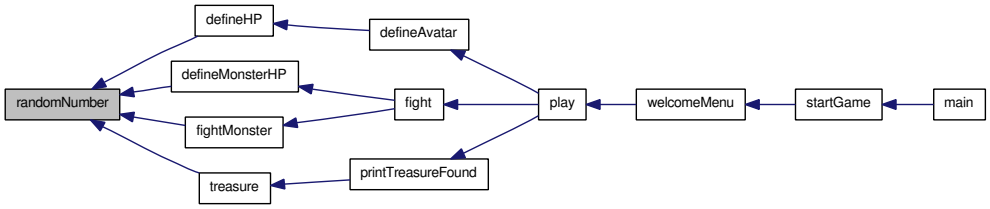
Definition at line 159 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.16 int randomNumber (int *top*, int *bottom*)

Definition at line 243 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:



2.1.2.17 float randomTurn ()

Definition at line 235 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.18 static int readEnemysID (FILE * *file*) [static]

Reads and expects a number between <ID> and </ID>.

Returns

ID in case there was no ERROR. Returns -1 if there was an ERROR

Definition at line 531 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

Here is the caller graph for this function:

2.1.2.19 static int readsAllEnemys (FILE * *file*, gameDataType * *data*) [static]

Reads ALL enemys present. This means that it will only read the ammount of existing enemys in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual enemy.

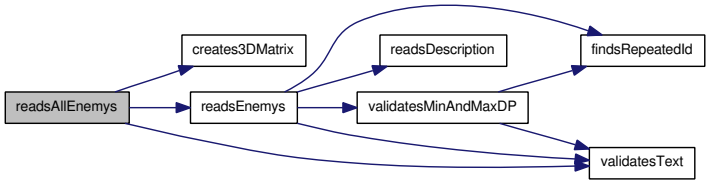
Calls a specific function encharged of creating the 3d matrix

Returns

Returns 1 if error

Definition at line 1087 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

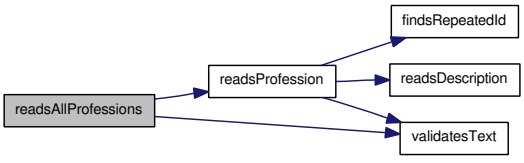
2.1.2.20 static int readsAllProfessions (gameDataType * *data*, FILE * *file*) [static]

Reads all professions. This means that it will only read the ammount of existing professions in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual profession.

Returns 1 if ERROR.

Definition at line 798 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

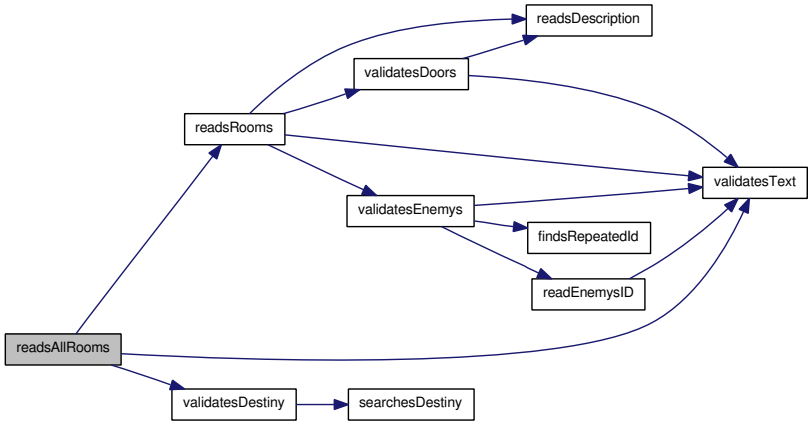
2.1.2.21 static int readsAllRooms (gameDataType * *data*, FILE * *file*) [static]

Reads all rooms. This means that it will only read the ammount of existing rooms in the file (validates the present text before and after their definition) and then calls a specific function encharged of reading each individual room.

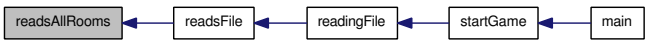
Returns 1 if ERROR

Definition at line 843 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



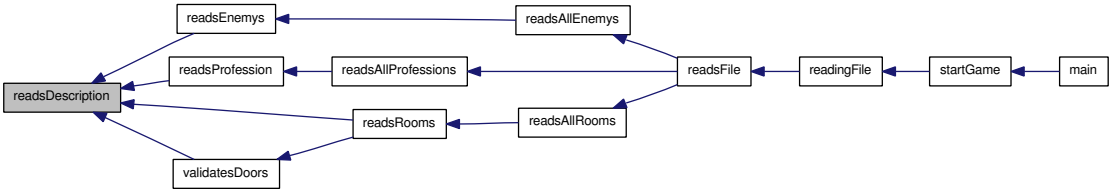
2.1.2.22 static char * readsDescription (FILE * *file*) [static]

Reads text from file until it finds a "<".

Returns a pointer to the read text. NULL if there was an error

Definition at line 491 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:



2.1.2.23 static int readsEnemys (FILE * *file*, gameDataType * *data*, int *i*) [static]

Rads ONE enemy.

Parameters

i parameter represents the position in the enemy's array where the data read must be saved. OF USE:

for(i=0; i<enemyQty; i++)

{

readsEnemys(data, file, i);

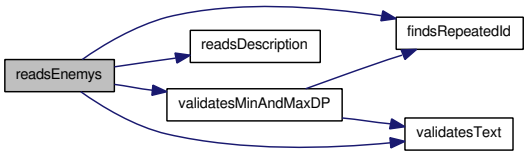
}

Returns

Returns 1 if ERROR.

Definition at line 939 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

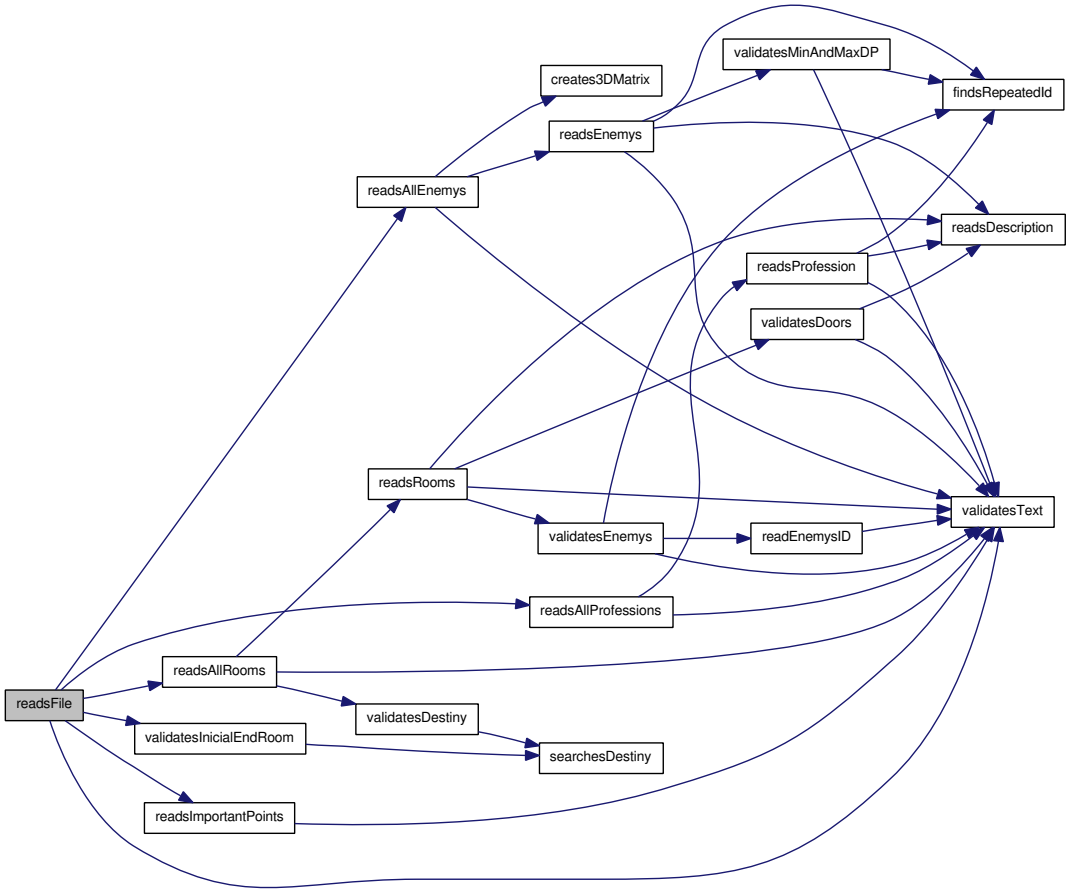


Here is the caller graph for this function:

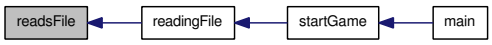
2.1.2.24 int readsFile (FILE * *file*, gameDataType * *data*)

Definition at line 1186 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2.25 static int readsImportantPoints (FILE * *file*, gameDataType * *data*) [static]

Reads the important points section. Saves corresponding data in the struct.

Returns

Returns 1 if there was an error

Definition at line 307 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

Here is the caller graph for this function:

2.1.2.26 static int readsProfession (gameDataType * *data*, FILE * *file*, int *i*) [static]

Reads ONE profession. Saves corresponding data into struct.

Parameters

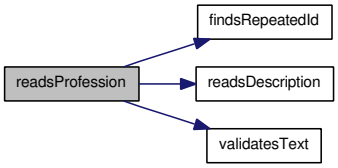
i represents the position in the profession'sarray in which the information read must be saved.
EXAMPLE OF USE:
for(i=0; i<professionQty; i++)
{
readsProfession(data, file, i);
}
Where professionQty is the number of elements that the array where the data will be saved has.

Returns

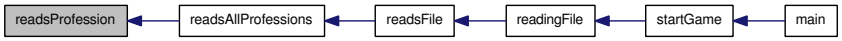
Returns 1 if there was an error

Definition at line 393 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2.27 static int readsRooms (gameDataType * *data*, FILE * *file*, int *i*) [static]

Reads ONE room. Saves read data into corresponding struct.

Parameters

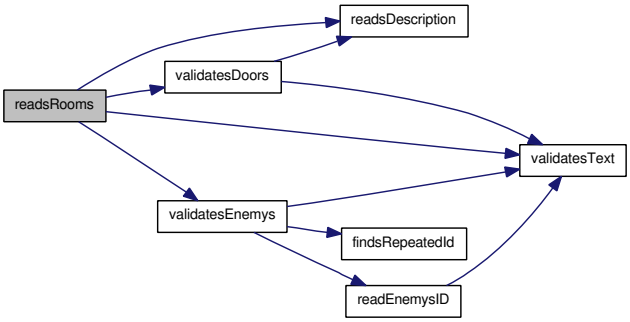
i represents the position of the structure correspondig to the actual room in which the data must be saved. OF USE:
for(i=0; i<roomsQty; i++)
{

readsRooms(data, file, i);
}
Where roomsQty is the ammount of elements that the array where the data will be saved has.

Returns 1 if error

Definition at line 564 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.1.2.28 int saveDoor (gameDataType * *data*, int *chosenDoor*)

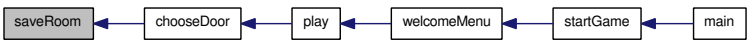
Definition at line 212 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

2.1.2.29 int saveRoom (gameDataType * *data*, int *actualRoom*)

Definition at line 183 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:



2.1.2.30 static int searchesDestiny (gameDataType * *data*, int *dest*) [static]

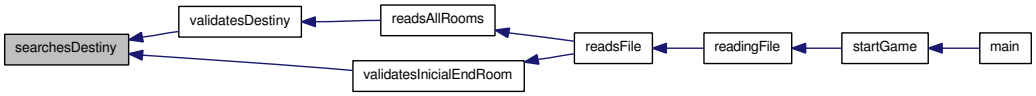
Loos up for a given ID in the room's array to verify it exists.

Returns

Returns -1 if not found

Definition at line 917 of file dungeonCrawlerBack.c.

Here is the caller graph for this function:

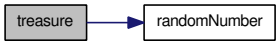


2.1.2.31 int treasure (avatarType * *avatar*, gameDataType * *data*)

Generates a random number and depending on this number perfoms different accions. It either reduces avatar's HP, increases it or increases the MaxDP.

Definition at line 1299 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.1.2.32 static int validatesDestiny (gameDataType * *data*) [static]

Validates that the destiny of the doors in a room EXISTS.

Returns

Returns 1 if error.

Definition at line 770 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

Here is the caller graph for this function:

2.1.2.33 static int validatesDoors (gameDataType * *data*, int * *doorcounter*, FILE * *file*, int *i*) [static]

Reads all the doors a room can have.

Receives an indicator of where should the data be saved in the room's array ("i" parameter). Receives a pointer to a counter that saves the amount of doors present in the room.

OF USE:

for(i=0; i<roomQty; i++)

{

-------- (any other lines that might come before)

if(there are doors)

{

validatesDoors(....., i);

}

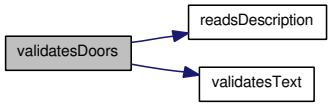
-------- (any other lines that might come after)

}

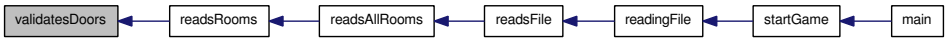
Returns 1 if ERROR

Definition at line 716 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.1.2.34 static int validatesEnemys (gameDataType * *data*, FILE * *file*, int *i*) [static]

Reads the enemys present in a room.

Parameters

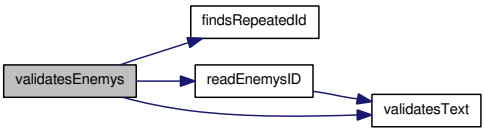
i in the parameters represents the position in the room's array where the data must be saved. OF USE:
for(i=0; i<enemyQty; i++)
{
validatesEnemy(data, file, i);
}
Where enemyQty is the ammount of enemys present in a room.

The variable memory saves the place in which the enemy is located in the enemy's vector. i.e.If the enemy with ID 5 was 3rd in the enemy's array, then memory will have the number 2.

Returns 1 if error

Definition at line 652 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.1.2.35 static int validatesInicialEndRoom (gameDataType * *data*, int *startRoom*, int *finalRoom*) [static]

Validates that the inicial and end room exist. Variables memoryinicial and memoryfinal do the same. They both save the position in the room's vector in which the searched room is located.

Returns

Returns 1 if no error. 0 if there was error

Definition at line 1171 of file dungeonCrawlerBack.c.

Here is the call graph for this function:

Here is the caller graph for this function:

2.1.2.36 static int validatesMinAndMaxDP (gameDataType * *data*, FILE * *file*, int *i*) [static]

Reads all the "<MinDP-X>%d</MinDP-X>" that are present in an enemy's definition.

Parameters

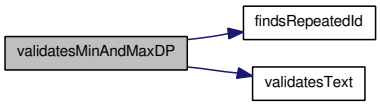
i repesents where to save the data in the 3d array.

Returns

Returns 1 if error

Definition at line 1017 of file dungeonCrawlerBack.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.1.2.37 static int validatesText (FILE * *file*, char * *text*) [static]

Reads from File the ammount of characters that where passed by parameter to the function. Then compares the read text with the parameter.

Parameters

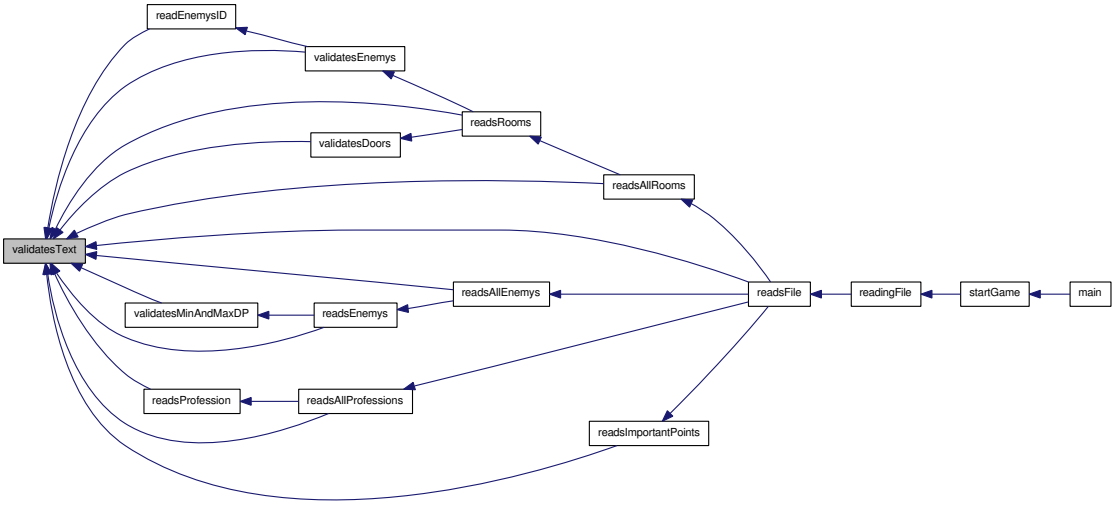
Function gets a pointer to file and a text to read from file.

Returns

Returns 1 if there was an error. 0 if everything went OK

Definition at line 363 of file dungeonCrawlerBack.c.

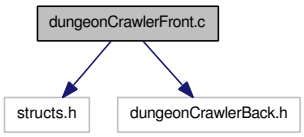
Here is the caller graph for this function:



2.2 dungeonCrawlerFront.c File Reference

```
#include "structs.h"
#include "dungeonCrawlerBack.h"
```

Include dependency graph for dungeonCrawlerFront.c:



Defines

- #define LIMIT_ENGLISH 127
- #define CLEAN_BUFFER while (getchar()!='\n')

Functions

- static int readAvatarName (avatarType *avatar, char *auxName)
- static void asignAvatarName (avatarType *avatar)
- static void printTitle ()
- static int welcomeMenu (avatarType *avatar, gameDataType *data)
- static void defineAvatar (avatarType *avatar, gameDataType *data)
- static void printProfessionsMenu (gameDataType *data)
- static void printAvatarFeatures (avatarType *avatar, gameDataType *data)
- static void printActualState (avatarType *avatar, gameDataType *data)
- static void printAvailableDoors (avatarType *avatar, gameDataType *data)
- static void chooseDoor (avatarType *avatar, gameDataType *data)
- static void play (avatarType *avatar, gameDataType *data)
- static void chooseProfession (avatarType *avatar, gameDataType *data)
- static void printActualMonster (gameDataType *data, int monsterPosition)
- static int readingFile (FILE *file, gameDataType *data)
- static FILE * openingFile (int argc, char **argv, unsigned int *seed)
- static void startGame (avatarType *avatar, gameDataType *data, int argc, char **argv, FILE *file)
- static void printsHPChanges (int avatarAttacks, gameDataType *data, avatarType *avatar, int i, int previousHP, int room)
- static int waitForEnter ()
- static void printTreasureFound (avatarType *avatar, gameDataType *data)
- static void printQuestion (gameDataType *data, avatarType *avatar)
- int main (int argc, char **argv)

2.2.1 Define Documentation

2.2.1.1 #define CLEAN_BUFFER while (getchar()!='\n')

Definition at line 5 of file dungeonCrawlerFront.c.

2.2.1.2 #define LIMIT_ENGLISH 127

Definition at line 4 of file dungeonCrawlerFront.c.

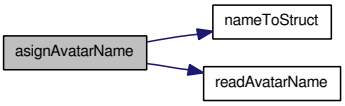
2.2.2 Function Documentation

2.2.2.1 static void asignAvatarName (avatarType * *avatar*) [static]

Calls the function readAvatarName while the name inserted is not valid. If there is a memory error, exits

Definition at line 339 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



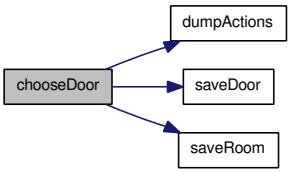
Here is the caller graph for this function:

2.2.2.2 static void chooseDoor (avatarType * *avatar*, gameDataType * *data*) [static]

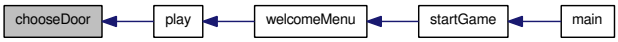
Asks the player for the door to move or dumpActions. If dumpActions is introduced then it reads the file that must be created and calls dumpActions function

Definition at line 251 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.2.2.3 static void chooseProfession (avatarType * *avatar*, gameDataType * *data*) [static]

Reads the profession that the player wants

Definition at line 130 of file dungeonCrawlerFront.c.

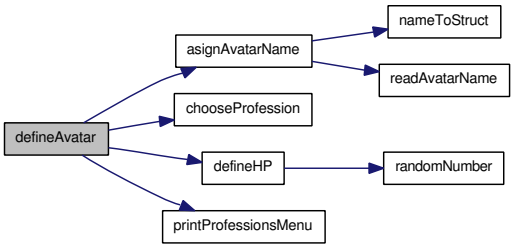
Here is the caller graph for this function:

2.2.2.4 static void defineAvatar (avatarType * *avatar*, gameDataType * *data*) [static]

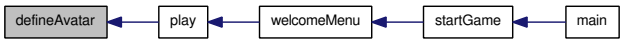
Calls functions needed to define the avatar's features

Definition at line 381 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



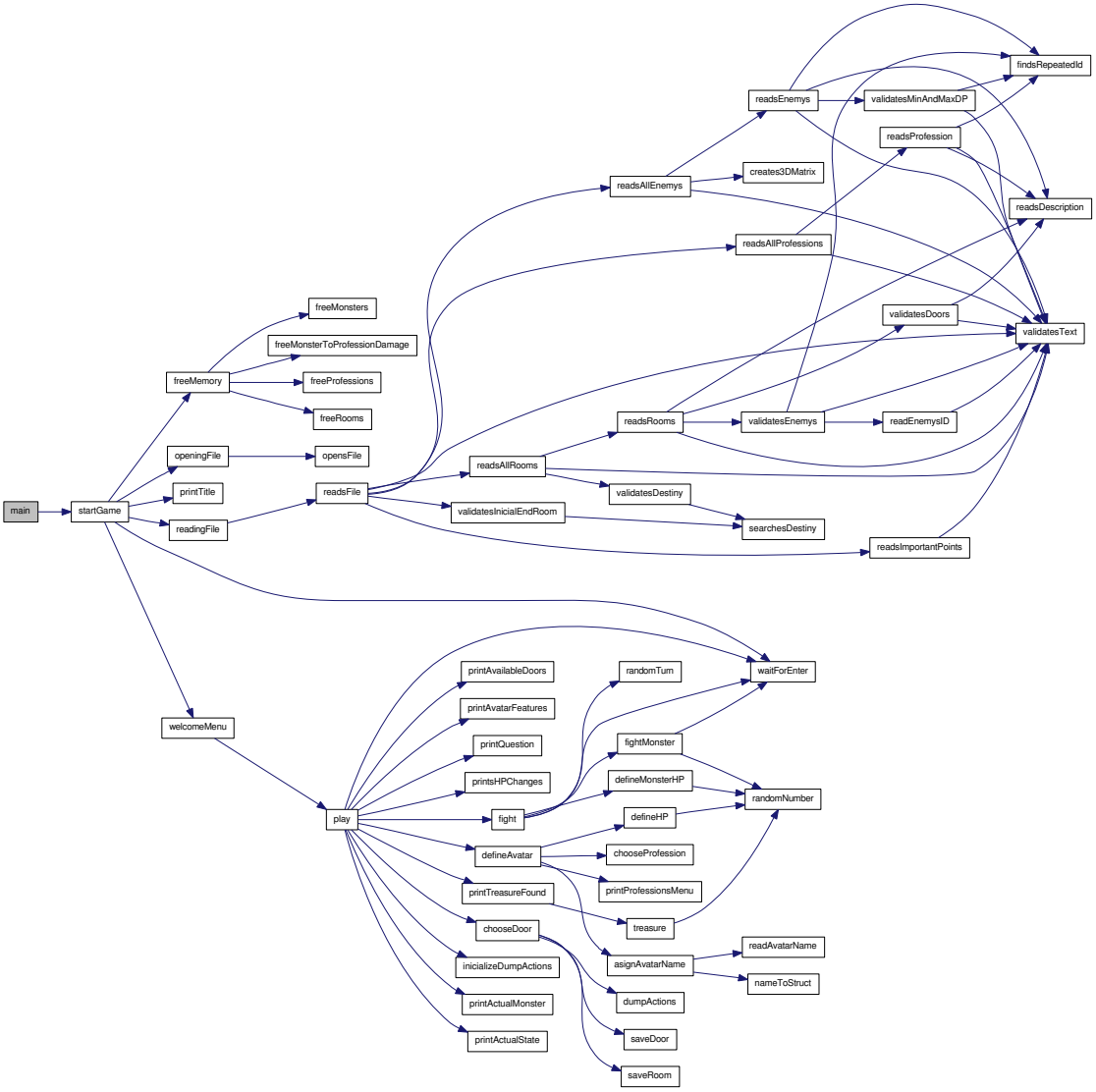
Here is the caller graph for this function:



2.2.2.5 int main (int *argc*, char ** *argv*)

Definition at line 32 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



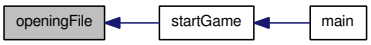
2.2.2.6 static FILE * openingFile (int *argc*, char ** *argv*, unsigned int * *seed*) [static]

Calls a function encharged of reading the file passed by parameter If file does not open prints message and returns NULL. Else returns a pointer to FILE

Definition at line 78 of file dungeonCrawlerFront.c.

Here is the call graph for this function:

Here is the caller graph for this function:

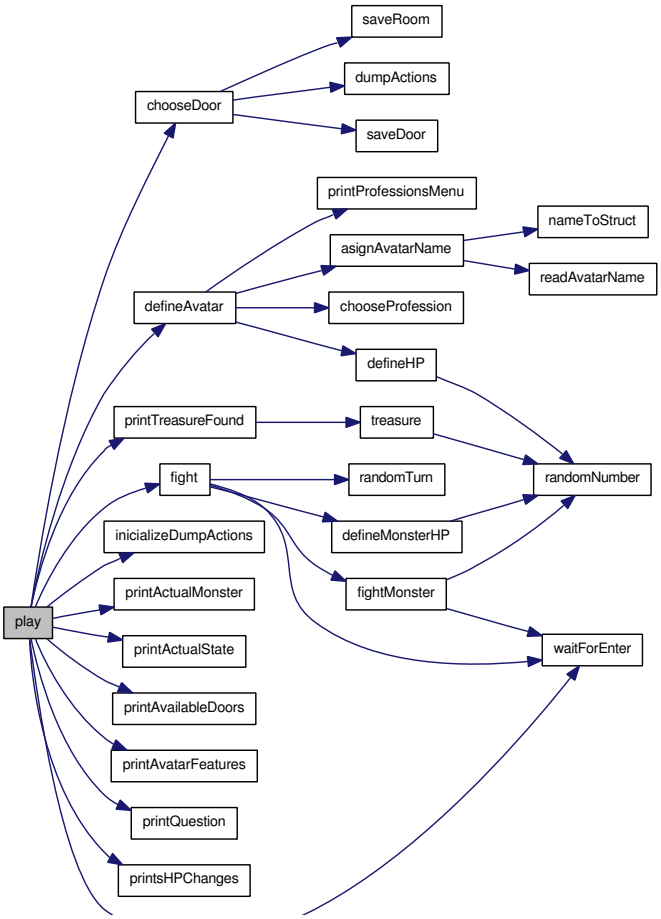


2.2.2.7 static void play (avatarType * *avatar*, gameDataType * *data*) [static]

Calls the functions fight, continue fight, choosedoor and print available doors while the avatar is allive

Definition at line 193 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:

2.2.2.8 void printActualMonster (gameDataType * *data*, int *monsterPosition*) [static]

Prints the monster that comes next in the room

Definition at line 122 of file dungeonCrawlerFront.c.

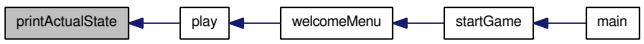
Here is the caller graph for this function:

2.2.2.9 static void printActualState (avatarType * *avatar*, gameDataType * *data*) [static]

Prints the actual location of the avatar

Definition at line 320 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:

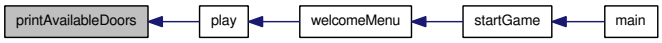


2.2.2.10 static void printAvailableDoors (avatarType * *avatar*, gameDataType * *data*) [static]

Prints the available doors in the room

Definition at line 303 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:

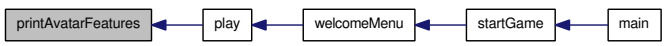


2.2.2.11 static void printAvatarFeatures (avatarType * *avatar*, gameDataType * *data*) [static]

Prints the the avatar's features. i.e. HP, name, profession

Definition at line 330 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:



2.2.2.12 static void printProfessionsMenu (gameDataType * *data*) [static]

Prints all professions and their HP and DP

Definition at line 464 of file dungeonCrawlerFront.c.

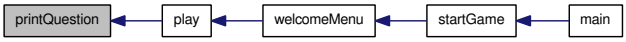
Here is the caller graph for this function:

2.2.2.13 static void printQuestion (gameDataType * *data*, avatarType * *avatar*) [static]

If the HP falls below a certain percentage of the max value, a magician appaears and tries to help the player. This function prints the messages to the user

Definition at line 499 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:



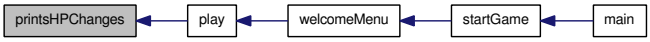
2.2.2.14 static void printsHPChanges (int *avatarAttacks*, gameDataType * *data*, avatarType * *avatar*, int *i*, int *previousHP*, int *room*) [static]

Prints the corresponding message for the attack.

The "i" refers to the enemy that must be printed next

Definition at line 153 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:

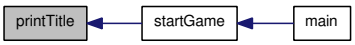


2.2.2.15 static void printTitle () [static]

prints a welcome message when the game starts

Definition at line 438 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:

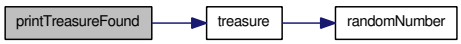


2.2.2.16 static void printTreasureFound (avatarType * *avatar*, gameDataType * *data*) [static]

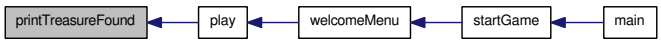
If a treasure is found, this function prints a message telling the player how will he be affected

Definition at line 479 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:



2.2.2.17 static int readAvatarName (avatarType * *avatar*, char * *auxName*) [static]

Reads the avatar's name Returns 0 if there was an error

Definition at line 360 of file dungeonCrawlerFront.c.

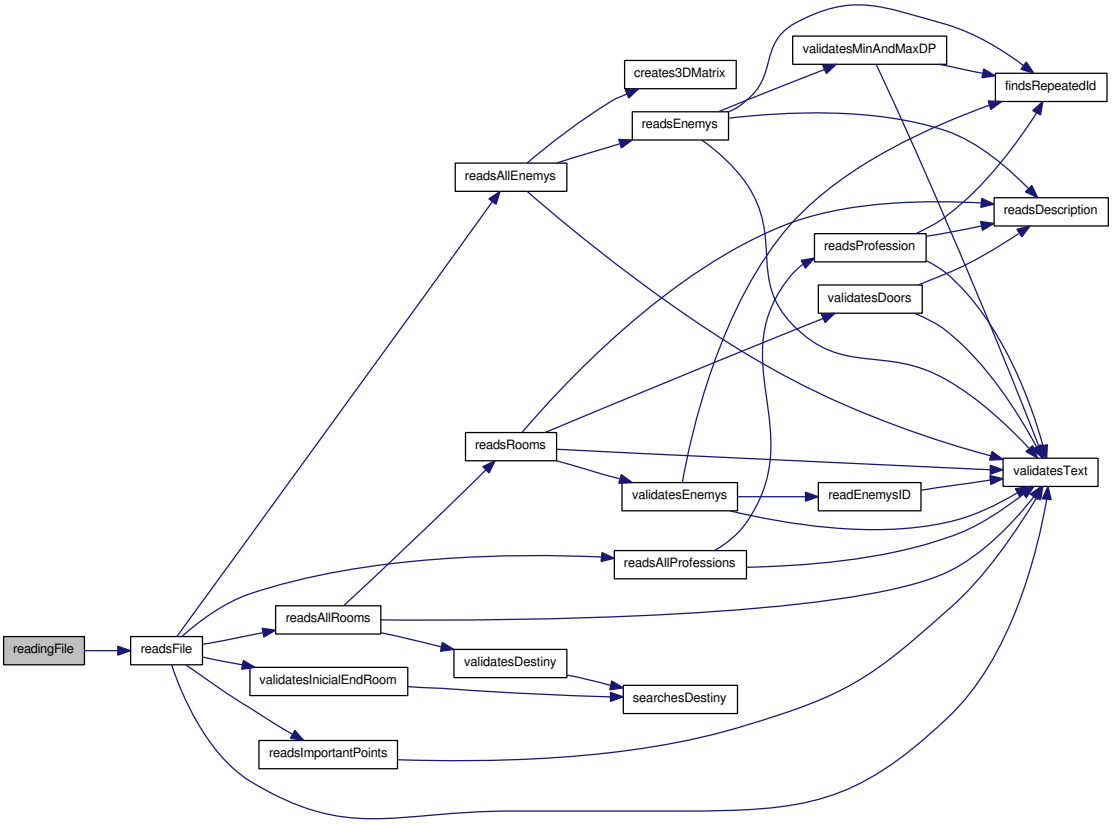
Here is the caller graph for this function:

2.2.2.18 static int readingFile (FILE * *file*, gameDataType * *data*) [static]

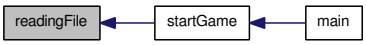
Prints message according to the error returned by readsFile

Definition at line 92 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:

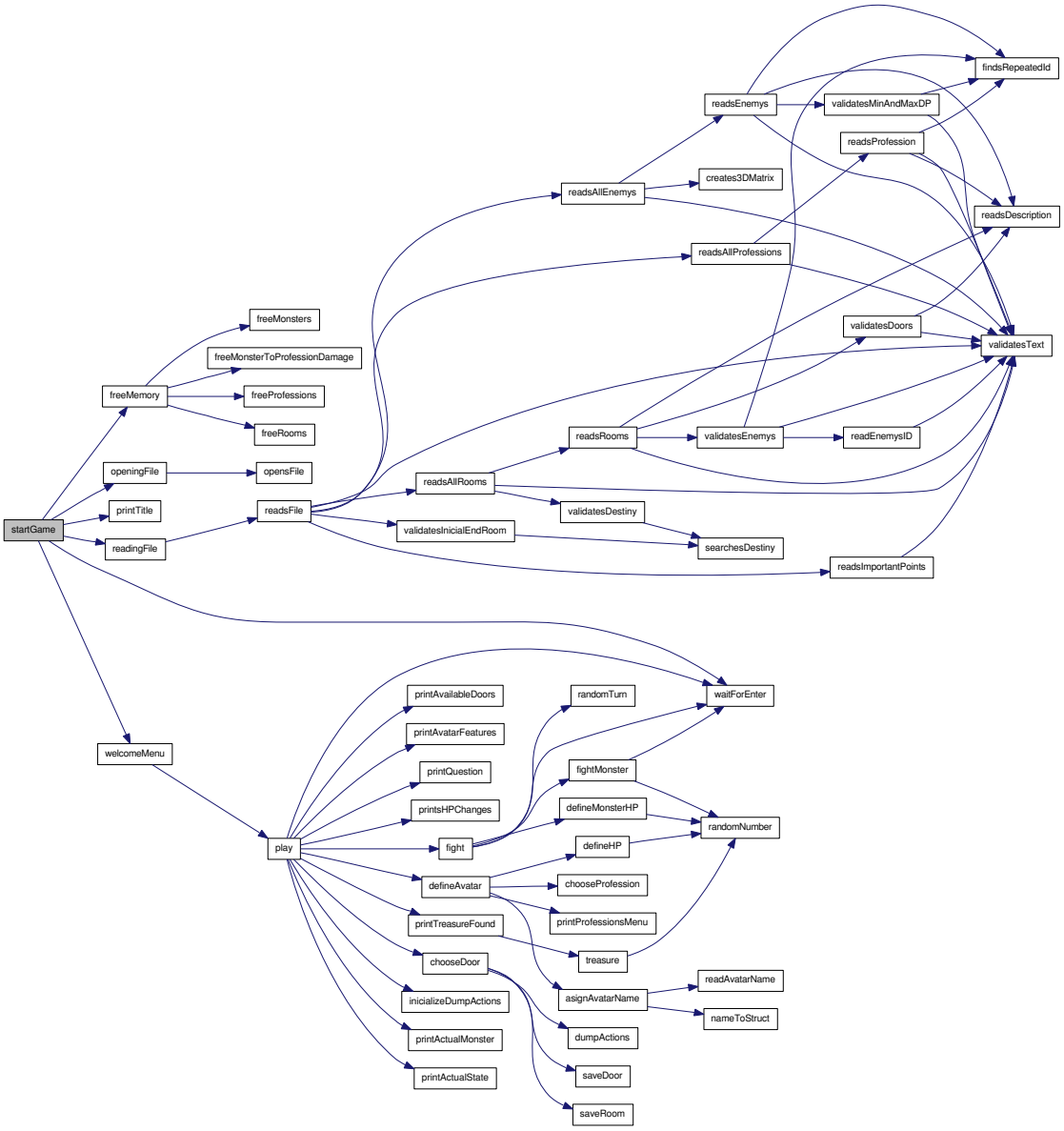


2.2.2.19 static void startGame (avatarType * *avatar*, gameDataType * *data*, int *argc*, char ** *argv*, FILE * *file*) [static]

If file does not open prints message and exits. Else reads calls the function that reads file and then calls the function that prints message

Definition at line 44 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:

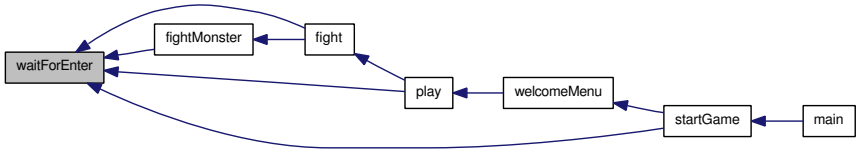
2.2.2.20 static int waitForEnter () [static]

Waits for the user to enter an ’

’ to continue Fighting

Definition at line 172 of file dungeonCrawlerFront.c.

Here is the caller graph for this function:



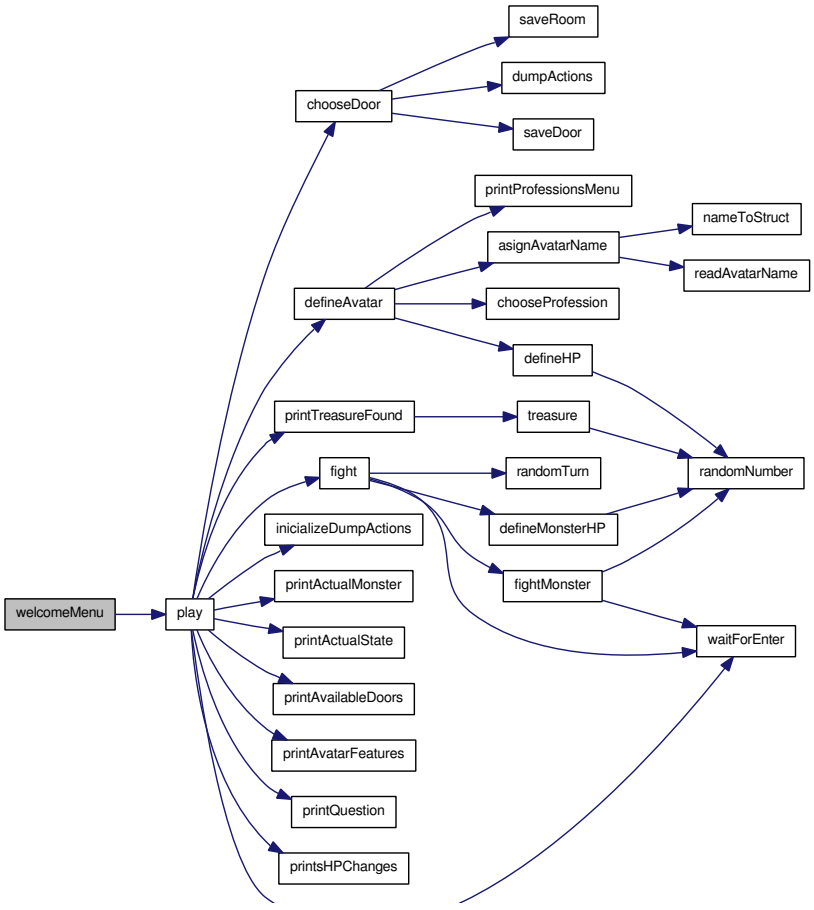
2.2.2.21 static int welcomeMenu (avatarType ∗ *avatar*, gameDataType ∗ *data*) [static]

Prints the welcome Menu. Depending on the option chosen, prints a specific message.

Returns 1 if option chosen was play

Definition at line 394 of file dungeonCrawlerFront.c.

Here is the call graph for this function:



Here is the caller graph for this function:

Index