

Bitter Birds

Generated by Doxygen 1.9.1

1 Angry Birds	1
1.1 Group	1
1.2 Repository organization	1
1.3 Project Implementation	1
1.4 Working practices	2
1.5 Source code documentation	2
1.6 TODOs (updated 27.10.2023)	2
1.6.0.1 Tasks	2
2 Contents	3
3 Meeting Notes	5
3.1 Meeting 27.10.2023 16:00	5
3.1.1 Summary of works	5
3.1.2 Challenges	5
3.1.3 Actions	5
3.1.4 Project status	6
3.1.4.1 TODOs	6
3.2 Meeting 8.11.2023 14:30	6
3.2.1 Summary of works	6
3.2.2 Challenges	6
3.2.3 Actions	6
3.2.4 Project status	6
3.2.4.1 TODOs	6
3.3 Meeting 20.11.2023 14:55	7
3.3.1 Summary of works	7
3.3.2 Challenges	7
3.3.3 Actions	7
3.3.4 Project status	7
3.3.4.1 TODOs	7
3.4 Meeting 10.12.2023 14:30	7
3.4.1 Summary of works	8
3.4.2 Challenges	8
3.4.3 Actions	8
3.4.4 Project status	8
3.4.4.1 TODOs	8
4 Contents	9
5 Bitter Birds savefile format	11
5.1 Bolded characters are headers:	11
5.1.1 SST:	11
5.1.2 SEN:	11
5.1.3 EST:	11

5.1.4 EEN:	11
5.1.5 BST:	11
5.1.6 BEN:	11
5.1.7 NME:	12
5.1.8 BGR:	12
5.1.9 STD:	12
5.1.10 UNKNOWN:	12
5.2 File format:	12
6 Source content	13
7 Hierarchical Index	15
7.1 Class Hierarchy	15
8 Class Index	17
8.1 Class List	17
9 File Index	19
9.1 File List	19
10 Class Documentation	21
10.1 Bird Class Reference	21
10.1.1 Detailed Description	22
10.1.2 Constructor & Destructor Documentation	22
10.1.2.1 Bird()	22
10.1.2.2 ~Bird()	23
10.1.3 Member Function Documentation	23
10.1.3.1 get_bird_type()	23
10.1.3.2 getTime()	23
10.1.3.3 resetTime()	23
10.1.4 Member Data Documentation	23
10.1.4.1 spawnTime	24
10.2 Color Struct Reference	24
10.2.1 Detailed Description	24
10.2.2 Member Data Documentation	24
10.2.2.1 b	25
10.2.2.2 g	25
10.2.2.3 r	25
10.3 Enemy Class Reference	25
10.3.1 Constructor & Destructor Documentation	26
10.3.1.1 Enemy()	26
10.3.2 Member Function Documentation	27
10.3.2.1 contains()	27
10.3.2.2 get_type()	27

10.4 Entity Class Reference	28
10.4.1 Detailed Description	29
10.4.2 Constructor & Destructor Documentation	29
10.4.2.1 Entity()	30
10.4.3 Member Function Documentation	30
10.4.3.1 contains()	30
10.4.3.2 get_health_points()	31
10.4.3.3 get_rotation()	31
10.4.3.4 get_type()	31
10.4.3.5 get_x()	31
10.4.3.6 get_y()	32
10.4.3.7 is_destructible()	32
10.4.3.8 is_movable()	32
10.5 ExplosionData Struct Reference	33
10.5.1 Detailed Description	33
10.5.2 Member Data Documentation	33
10.5.2.1 position	33
10.5.2.2 time	33
10.5.2.3 type	34
10.6 GUI Class Reference	34
10.6.1 Detailed Description	35
10.6.2 Constructor & Destructor Documentation	35
10.6.2.1 GUI()	35
10.6.2.2 ~GUI()	35
10.6.3 Member Function Documentation	35
10.6.3.1 button_released()	36
10.6.3.2 button_state()	36
10.6.3.3 close()	36
10.6.3.4 cursor_position()	36
10.6.3.5 draw_button()	36
10.6.3.6 draw_rect()	37
10.6.3.7 draw_sprite()	37
10.6.3.8 draw_text()	38
10.6.3.9 get_aspect_ratio()	38
10.6.3.10 key_state()	38
10.6.3.11 play_sound()	39
10.6.3.12 run()	39
10.6.3.13 scroll_delta()	39
10.6.3.14 set_scene()	39
10.6.3.15 set_viewport()	40
10.7 Image Class Reference	40
10.7.1 Constructor & Destructor Documentation	41

10.7.1.1 Image()	41
10.7.2 Friends And Related Function Documentation	41
10.7.2.1 GUI	41
10.8 Level Class Reference	41
10.8.1 Constructor & Destructor Documentation	42
10.8.1.1 Level()	42
10.8.2 Member Function Documentation	42
10.8.2.1 add_score()	43
10.8.2.2 get_background()	43
10.8.2.3 get_birds()	43
10.8.2.4 get_entities()	43
10.8.2.5 get_name()	44
10.8.2.6 get_save_name()	44
10.8.2.7 get_scores()	44
10.9 LevelEditorScene Class Reference	45
10.9.1 Detailed Description	47
10.9.2 Constructor & Destructor Documentation	47
10.9.2.1 LevelEditorScene()	47
10.9.2.2 ~LevelEditorScene()	47
10.9.3 Member Function Documentation	47
10.9.3.1 render()	47
10.9.3.2 update()	48
10.10 LevelInfo Struct Reference	49
10.10.1 Detailed Description	49
10.10.2 Member Data Documentation	49
10.10.2.1 name	50
10.10.2.2 path	50
10.11 MenuScene Class Reference	50
10.11.1 Detailed Description	52
10.11.2 Constructor & Destructor Documentation	52
10.11.2.1 MenuScene()	52
10.11.2.2 ~MenuScene()	52
10.11.3 Member Function Documentation	52
10.11.3.1 render()	52
10.11.3.2 update()	52
10.12 NameEntryScene Class Reference	53
10.12.1 Detailed Description	55
10.12.2 Constructor & Destructor Documentation	55
10.12.2.1 NameEntryScene()	55
10.12.2.2 ~NameEntryScene()	55
10.12.3 Member Function Documentation	55
10.12.3.1 on_input()	55

10.12.3.2 render()	56
10.12.3.3 update()	56
10.13 NormalBird Class Reference	56
10.13.1 Detailed Description	58
10.13.2 Constructor & Destructor Documentation	58
10.13.2.1 NormalBird()	58
10.13.3 Member Function Documentation	58
10.13.3.1 get_bird_type()	58
10.14 PlayScene Class Reference	59
10.14.1 Detailed Description	61
10.14.2 Constructor & Destructor Documentation	61
10.14.2.1 PlayScene()	61
10.14.2.2 ~PlayScene()	61
10.14.3 Member Function Documentation	61
10.14.3.1 render()	61
10.14.3.2 update()	62
10.15 ReaderWriter Class Reference	63
10.15.1 Detailed Description	64
10.15.2 Constructor & Destructor Documentation	64
10.15.2.1 ReaderWriter()	64
10.15.3 Member Function Documentation	64
10.15.3.1 get_levels()	64
10.15.3.2 get_next_level()	64
10.15.3.3 read_file()	64
10.15.3.4 write_file()	65
10.16 Scene Class Reference	65
10.16.1 Detailed Description	67
10.16.2 Constructor & Destructor Documentation	67
10.16.2.1 Scene()	67
10.16.2.2 ~Scene()	67
10.16.3 Member Function Documentation	67
10.16.3.1 on_input()	67
10.16.3.2 render()	68
10.16.3.3 update()	68
10.16.4 Member Data Documentation	68
10.16.4.1 gui_	68
10.17 ScoreBoardEntry Struct Reference	68
10.17.1 Member Data Documentation	69
10.17.1.1 name	69
10.17.1.2 score	69
10.18 SpecialBird1 Class Reference	69
10.18.1 Detailed Description	70

10.18.2 Constructor & Destructor Documentation	70
10.18.2.1 SpecialBird1()	70
10.18.3 Member Function Documentation	71
10.18.3.1 get_bird_type()	71
10.19 SpecialBird2 Class Reference	71
10.19.1 Detailed Description	72
10.19.2 Constructor & Destructor Documentation	72
10.19.2.1 SpecialBird2()	72
10.19.3 Member Function Documentation	73
10.19.3.1 get_bird_type()	73
10.20 Structure Class Reference	73
10.20.1 Detailed Description	74
10.20.2 Constructor & Destructor Documentation	75
10.20.2.1 Structure()	75
10.20.3 Member Function Documentation	75
10.20.3.1 contains()	75
10.20.3.2 get_height()	76
10.20.3.3 get_type()	76
10.20.3.4 get_width()	76
10.21 TestAssertionException Class Reference	77
10.21.1 Constructor & Destructor Documentation	78
10.21.1.1 TestAssertionException()	78
10.21.2 Member Function Documentation	78
10.21.2.1 what()	78
10.22 userDataStruct Struct Reference	78
10.22.1 Detailed Description	79
10.22.2 Member Data Documentation	79
10.22.2.1 bird	79
10.22.2.2 entity	79
10.22.2.3 hp	79
10.22.2.4 image	79
10.22.2.5 type	79
11 File Documentation	81
11.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference	81
11.1.1 Macro Definition Documentation	81
11.1.1.1 __has_include	82
11.1.1.2 ARCHITECTURE_ID	82
11.1.1.3 C_VERSION	82
11.1.1.4 COMPILER_ID	82
11.1.1.5 DEC	82
11.1.1.6 HEX	82

11.1.1.7 PLATFORM_ID	83
11.1.1.8 STRINGIFY	83
11.1.1.9 STRINGIFY_HELPER	83
11.1.2 Function Documentation	83
11.1.2.1 main()	83
11.1.3 Variable Documentation	83
11.1.3.1 info_arch	83
11.1.3.2 info_compiler	83
11.1.3.3 info_language_extensions_default	84
11.1.3.4 info_language_standard_default	84
11.1.3.5 info_platform	84
11.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference	84
11.2.1 Macro Definition Documentation	85
11.2.1.1 __has_include	85
11.2.1.2 ARCHITECTURE_ID	85
11.2.1.3 COMPILER_ID	85
11.2.1.4 CXX_STD	85
11.2.1.5 DEC	85
11.2.1.6 HEX	86
11.2.1.7 PLATFORM_ID	86
11.2.1.8 STRINGIFY	86
11.2.1.9 STRINGIFY_HELPER	86
11.2.2 Function Documentation	86
11.2.2.1 main()	86
11.2.3 Variable Documentation	86
11.2.3.1 info_arch	87
11.2.3.2 info_compiler	87
11.2.3.3 info_language_extensions_default	87
11.2.3.4 info_language_standard_default	87
11.2.3.5 info_platform	87
11.3 doc/readme.md File Reference	88
11.4 plan/readme.md File Reference	88
11.5 src/readme.md File Reference	88
11.6 Meeting-notes.md File Reference	88
11.7 README.md File Reference	88
11.8 res/README.md File Reference	88
11.9 SavefileFormat.md File Reference	88
11.10 src/bird.cpp File Reference	88
11.11 src/bird.hpp File Reference	89
11.11.1 Enumeration Type Documentation	90
11.11.1.1 birdType	90
11.12 src/enemy.cpp File Reference	90

11.13 src/enemy.hpp File Reference	90
11.14 src/entity.cpp File Reference	91
11.15 src/entity.hpp File Reference	92
11.15.1 Enumeration Type Documentation	92
11.15.1.1 bodyType	92
11.16 src/GUI.cpp File Reference	92
11.17 src/GUI.hpp File Reference	93
11.17.1 Enumeration Type Documentation	94
11.17.1.1 Alignment	94
11.18 src/image.cpp File Reference	94
11.19 src/image.hpp File Reference	95
11.20 src/level.cpp File Reference	96
11.21 src/level.hpp File Reference	96
11.22 src/levelEditorScene.cpp File Reference	97
11.23 src/levelEditorScene.hpp File Reference	98
11.24 src/main.cpp File Reference	99
11.24.1 Function Documentation	100
11.24.1.1 main()	100
11.25 tests/main.cpp File Reference	100
11.25.1 Function Documentation	101
11.25.1.1 dummytest1()	101
11.25.1.2 main()	101
11.26 src/menuScene.cpp File Reference	101
11.27 src/menuScene.hpp File Reference	101
11.27.1 Enumeration Type Documentation	102
11.27.1.1 MenuState	102
11.28 src/nameEntryScene.cpp File Reference	103
11.29 src/nameEntryScene.hpp File Reference	103
11.30 src/playScene.cpp File Reference	104
11.31 src/playScene.hpp File Reference	105
11.31.1 Enumeration Type Documentation	106
11.31.1.1 explosionType	106
11.31.1.2 gameState	106
11.32 src/readerWriter.cpp File Reference	107
11.32.1 Macro Definition Documentation	107
11.32.1.1 RW_LOG	107
11.32.2 Function Documentation	107
11.32.2.1 split_string()	108
11.33 src/readerWriter.hpp File Reference	108
11.33.1 Enumeration Type Documentation	109
11.33.1.1 Header	109
11.34 src/scene.cpp File Reference	109

11.35 src/scene.hpp File Reference	110
11.36 src/structure.cpp File Reference	111
11.37 src/structure.hpp File Reference	111
11.38 tests/reader_writer_test.hpp File Reference	112
11.38.1 Function Documentation	113
11.38.1.1 test_ReaderWriter_read_file()	113
11.38.1.2 test_ReaderWriter_write_file()	114
11.38.1.3 tests_reader_writer()	114
11.39 tests/test_utils.hpp File Reference	114
11.39.1 Function Documentation	115
11.39.1.1 verify()	115
Index	117

Chapter 1

Angry Birds

1.1 Group

- Luukas Karihtala
- Ray Sointula
- Onni Komulainen
- Daniel Granström

1.2 Repository organization

Your project implementation should follow the skeleton organization in this repository. See `readme.md` files in each folder.

1.3 Project Implementation

You must use git repository for the work on the project, making frequent enough commits so that the project group (and course staff) can follow the progress.

The completed project work will be demonstrated to the group's advisor at a demo session. The final demonstrations are arranged on week 50. After the final demonstrations project group evaluates another project, and self-evaluates own project. In addition, project members will give a confidential individual assessment of each group member

The course staff should be able to easily compile the project work using makefile and related instructions provided in the git repository. The final output should be in the **master branch** of the git repository.

1.4 Working practices

Each project group is assigned an advisor from the project teaching personnel. There will be a dedicated Teams channel for each project topic to facilitate discussion between the groups in the same topic and the advisor.

The group should meet weekly. The weekly meeting does not need to be long if there are no special issues to discuss, and can be taken remotely as voice/video chat on the group Teams channel (or Zoom or other similar tool), preferably at a regular weekly time. In the meeting the group updates:

- What each member has done during the week
- Are there challenges or problems? Discuss the possible solutions
- Plan for the next week for everyone
- Deviations and changes to the project plan, if any
- After the meetings, the meeting notes will be committed to the project repository in the `Meeting-notes.md` file.
 - The commits within the week should have some commit messages referring to the meeting notes so that the project advisor can follow the progress.
 - **The meeting notes should be in English.**

Everyone may not be able to participate to all meetings, but at least a couple of members should be present in each meeting. Regular absence from meetings will affect in individual evaluation.

1.5 Source code documentation

The source code is documented using doxygen style comments. If you do not have doxygen yet installed it can be installed using the following command.

```
apt-get install doxygen
apt-get install graphviz
apt-get install texlive-latex-base
apt-get install texlive-fonts-recommended
apt-get install texlive-fonts-extra
apt-get install texlive-latex-extra
```

Generating the documentation can be done by running
`doxygen Doxyfile`

Open the html document `html/index.html` generated by doxygen in your browser to view the documentation.

To generate a PDF file

```
cd latex
make
```

Now there is a PDF `latex/refman.pdf`

1.6 TODOs (updated 27.10.2023)

List of todo tasks in the following format

- [TASK]. Assigned to <Member name>

1.6.0.1 Tasks

- ...

Chapter 2

Contents

The actual project documentation in PDF format must be committed in this folder before the deadline. Separate PDF document needs to be provided also if your project uses Doxygen for inline documentation.

The document should contain the following parts:

1. **Overview:** what the software does, what it doesn't do? (this can be taken/updated from the project plan)
2. **Software structure:** overall architecture, class relationships (diagram very strongly recommended), interfaces to external libraries
3. **Instructions** for building and using the software
4. **How to compile the program** ('make' should be sufficient), as taken from git repository. If external libraries are needed, describe the requirements here
5. How to use the software: a basic user guide
6. **Testing:** how the different modules in software were tested, description of the methods and outcomes
7. **Work log:** This might be a simplified/restructured version of the weekly meeting notes file.
8. Detailed description of division of work and everyone's responsibilities
9. For each week, description of what was done and roughly how many hours were used, for each project member.

Chapter 3

Meeting Notes

In this file, you are required to take notes for your weekly meetings. In each meeting, you are required to discuss:

1. What each member has done during the week?
2. Are there challenges or problems? Discuss the possible solutions
3. Plan for the next week for everyone
4. Deviations and changes to the project plan, if any

3.1 Meeting 27.10.2023 16:00

Participants:

1. Luukas Karihtala
2. Ray Sointula
3. Onni Komulainen
4. Daniel Granström

3.1.1 Summary of works

Started work on project by developing project plan. Initial work environment with SFML and box2d for building project.

3.1.2 Challenges

1. Game architecture

3.1.3 Actions

1. Team: Complete the project plan

3.1.4 Project status

Working on initial project plan

3.1.4.1 TODOs

1. Complete the project plan

3.2 Meeting 8.11.2023 14:30

Participants:

1. Luukas Karihtala
2. Ray Sointula
3. Daniel Granström

3.2.1 Summary of works

Participated in plan review with our advisor.

Daniel: Made some textures for game and started work on soundtrack. Luukas: Created [GUI](#), [Scene](#) and [MenuScene](#) classes Ray: Created individual issues for tasks. Worked level [ReaderWriter](#) class. Onni: Worked on [Level](#), [Entity](#) and [Bird](#) classes

3.2.2 Challenges

Ray can't continue on readerwriter until other work gets done. Finding meeting times difficult (everyone is not available at the same times).

3.2.3 Actions

Start documentation process with doxygen. Implement testing infrastructure and start writing tests for features.

3.2.4 Project status

Windowing works. Simple button UI for main menu. Beginnings of ReadWriter class. Beginnings of level and entity classes.

3.2.4.1 TODOs

1. Start working on issues described in gitlab.
2. Tests
3. Documentation

3.3 Meeting 20.11.2023 14:55

Participants:

1. Luukas Karihtala
2. Ray Sointula
3. Daniel Granström
4. Onni Komulainen

3.3.1 Summary of works

Luukas has implemented some [PlayScene](#) features

3.3.2 Challenges

Scheduling and communication has been challenging/lacking.

3.3.3 Actions

1. Increase work effort
2. Make progress in class structures
3. Plan work-session(s)
4. Finish [ReaderWriter](#)

3.3.4 Project status

Start of playScene, mainMenu and physics.

3.3.4.1 TODOs

1. Continue work on GitLab issues.

3.4 Meeting 10.12.2023 14:30

Participants:

1. Luukas Karihtala
2. Ray Sointula

3.4.1 Summary of works

Doxygen setup/tweaking and commenting code Camera animations, better textures

3.4.2 Challenges

-

3.4.3 Actions

-

3.4.4 Project status

Unaccounting for some finishing touches, the project is done.

3.4.4.1 TODOs

Finishing touches, documentation etc.

Chapter 4

Contents

Project plan is a PDF document describing the scope of the project, major architectural decisions, preliminary schedule and distribution of roles in the group, design rationale and so on. The document should be roughly five pages long, with a couple of diagrams illustrating the program design (for example, the planned class relationships).

You are required commit your project plan in this folder before the deadline. The plan should contain the following information:

- Scope of the work: what features and functionalities will be implemented, how is the program used, and how does it work
- High-level structure of the software: main modules, main classes (according to current understanding)
- Planned use of external libraries
- Division of work and responsibilities between the group
- Planned schedule and milestones before the final deadline of the project

It is not uncommon that as the project progresses, there may be changes relative to project plan, and that is fine. The final outcome will be described in the final documentation, that can be based on the project plan.

Chapter 5

Bitter Birds savefile format

This file explains the structure and formats of saved levels in this project.

5.1 Bolded characters are headers:

5.1.1 SST:

SoundFX vector Start, indicates that starting from the next line until line with SEN are soundFX paths

5.1.2 SEN:

SoundFX vector End, indicates end of soundFX paths

5.1.3 EST:

[Entity](#) vector Start, indicates that starting from the next line until line with EEN are Entities

5.1.4 EEN:

[Entity](#) vector End, indicates end of [Entity](#) vector

5.1.5 BST:

Birds vector Start, indicates that starting from the next line until line with BEN are Birds

5.1.6 BEN:

Birds vector End, indicates end of Birds vector

5.1.7 NME:

[Level](#) name, [Level](#) constructor variable name

5.1.8 BGR:

Background image path, [Level](#) constructor variable backgroundPath

5.1.9 STD:

Soundtrack path, [Level](#) constructor variable soundtrackPath

5.1.10 UNKNOWN:

For when something goes wrong

5.2 File format:

```
BitterBirds v.0
SST
<sound1Path>
<sound2Path>
...
<soundNPath>
SEN
EST
<Entity1>
<Entity2>
...
<EntityN>
EEN
BST
<Bird1Type>
<Bird2Type>
...
<BirdNType>
BEN
NME<String>
BGR<Path>
SDT<Path>
```

[Entity](#) and [Bird](#) formats will be discussed in this file when they are done. [Bird](#) format, N refers to birds type, N = 0, 1, 2, ...:

N

BGR and SDT contain paths which will be strings.

Chapter 6

Source content

This folder should contain only `hpp/cpp` files of your implementation. You can also place `hpp` files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Bird	21
NormalBird	56
SpecialBird1	69
SpecialBird2	71
Color	24
Entity	28
Enemy	25
Structure	73
std::exception	
TestAssertionException	77
ExplosionData	33
GUI	34
Image	40
Level	41
LevelInfo	49
ReaderWriter	63
Scene	65
LevelEditorScene	45
MenuScene	50
NameEntryScene	53
PlayScene	59
ScoreBoardEntry	68
userDataStruct	78

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Bird	Base class for Bird objects	21
Color	Color object r is red in range 0 to 1 g is green in range 0 to 1 b is blue in range 0 to 1	24
Enemy	25
Entity	Entity class to handle entity data loaded from levels	28
ExplosionData	Structure to store data for each explosion instance	33
GUI	GUI Class to create window and run the game	34
Image	40
Level	41
LevelEditorScene	A scene for editing the playable levels in the game	45
LevelInfo	Information about levels	49
MenuScene	Scene to handle the main menu	50
NameEntryScene	A scene for the user to enter their name from	53
NormalBird	Normal bird class No special effects	56
PlayScene	The game play scene to play the levels in the game	59
ReaderWriter	A class for loading/reading and saving/writing levels in forms of Level objects and .txt files . . .	63
Scene	Base Scene class to represent different states the game can be in. Ex. Play state can be implemented with PlayScene . Menu can be implemented with MenuScene etc	65
ScoreBoardEntry	68
SpecialBird1	Special Bird 1 class Has speed up effect	69
SpecialBird2	Special Bird 2 class Has explosion effect	71

Structure	
Structure	type to represent walls in the game 73
TestAssertionException 77
userDataStruct	
Physics entities hold a copy of userDataStruct they can reference 78

Chapter 9

File Index

9.1 File List

Here is a list of all files with brief descriptions:

build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c	81
build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp	84
src/bird.cpp	88
src/bird.hpp	89
src/enemy.cpp	90
src/enemy.hpp	90
src/entity.cpp	91
src/entity.hpp	92
src/GUI.cpp	92
src/GUI.hpp	93
src/image.cpp	94
src/image.hpp	95
src/level.cpp	96
src/level.hpp	96
src/levelEditorScene.cpp	97
src/levelEditorScene.hpp	98
src/main.cpp	99
src/menuScene.cpp	101
src/menuScene.hpp	101
src/nameEntryScene.cpp	103
src/nameEntryScene.hpp	103
src/playScene.cpp	104
src/playScene.hpp	105
src/readerWriter.cpp	107
src/readerWriter.hpp	108
src/scene.cpp	109
src/scene.hpp	110
src/structure.cpp	111
src/structure.hpp	111
tests/main.cpp	100
tests/reader_writer_test.hpp	112
tests/test_utils.hpp	114

Chapter 10

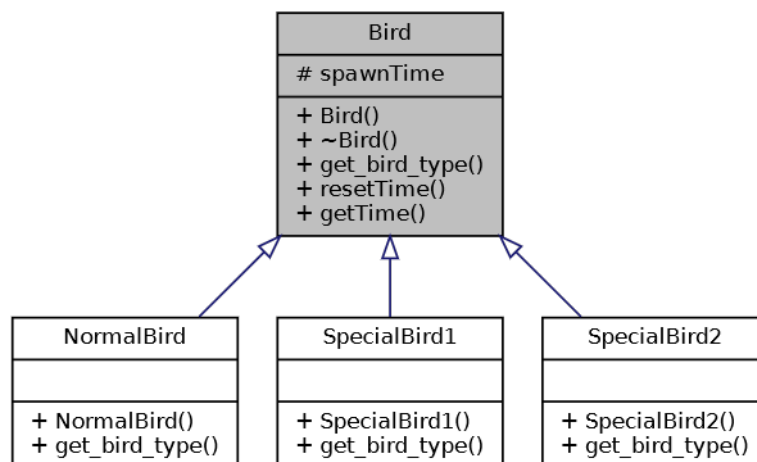
Class Documentation

10.1 Bird Class Reference

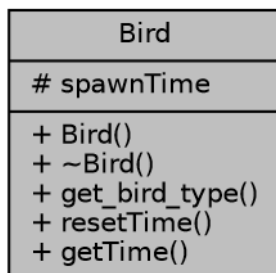
Base class for [Bird](#) objects.

```
#include <bird.hpp>
```

Inheritance diagram for Bird:



Collaboration diagram for Bird:



Public Member Functions

- [Bird](#) ()
Construct a new [Bird](#) object.
- [~Bird](#) ()
Destroy the [Bird](#) object.
- virtual [birdType](#) [get_bird_type](#) () const =0
Get the bird type object.
- void [resetTime](#) ()
Reset spawn timer.
- sf::Time [getTime](#) () const
Get the spawn time object.

Protected Attributes

- sf::Clock [spawnTime](#)

10.1.1 Detailed Description

Base class for [Bird](#) objects.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 Bird()

```
Bird::Bird ( )
```

Construct a new [Bird](#) object.

10.1.2.2 ~Bird()

```
Bird::~~Bird ( )
```

Destroy the [Bird](#) object.

10.1.3 Member Function Documentation

10.1.3.1 get_bird_type()

```
virtual birdType Bird::get_bird_type ( ) const [pure virtual]
```

Get the bird type object.

Returns

[birdType](#)

Implemented in [SpecialBird2](#), [SpecialBird1](#), and [NormalBird](#).

10.1.3.2 getTime()

```
sf::Time Bird::getTime ( ) const
```

Get the spawn time object.

Returns

[sf::Time](#)

10.1.3.3 resetTime()

```
void Bird::resetTime ( )
```

Reset spawn timer.

10.1.4 Member Data Documentation

10.1.4.1 spawnTime

```
sf::Clock Bird::spawnTime [protected]
```

The documentation for this class was generated from the following files:

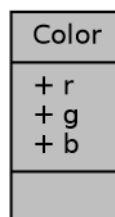
- [src/bird.hpp](#)
- [src/bird.cpp](#)

10.2 Color Struct Reference

[Color](#) object [r](#) is red in range 0 to 1 [g](#) is green in range 0 to 1 [b](#) is blue in range 0 to 1.

```
#include <GUI.hpp>
```

Collaboration diagram for [Color](#):



Public Attributes

- float [r](#)
- float [g](#)
- float [b](#)

10.2.1 Detailed Description

[Color](#) object [r](#) is red in range 0 to 1 [g](#) is green in range 0 to 1 [b](#) is blue in range 0 to 1.

10.2.2 Member Data Documentation

10.2.2.1 b

```
float Color::b
```

10.2.2.2 g

```
float Color::g
```

10.2.2.3 r

```
float Color::r
```

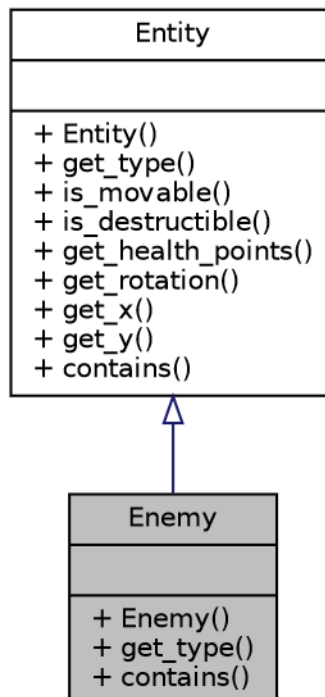
The documentation for this struct was generated from the following file:

- [src/GUI.hpp](#)

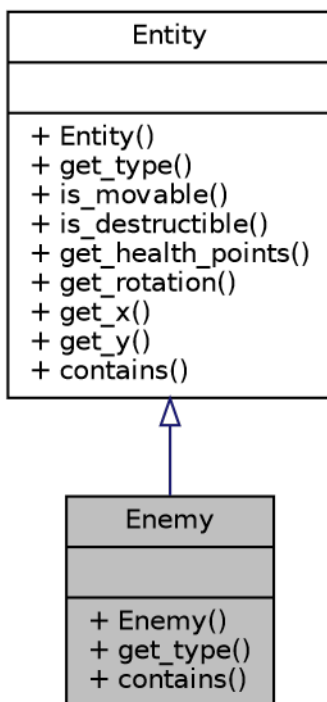
10.3 Enemy Class Reference

```
#include <enemy.hpp>
```

Inheritance diagram for Enemy:



Collaboration diagram for Enemy:



Public Member Functions

- [Enemy](#) (int healthPoints, double initRotation, double x, double y)
Construct a new [Enemy](#) object.
- [bodyType get_type](#) () const override
Get the type object, in this case bodyType::Enemy.
- bool [contains](#) (double x, double y)

10.3.1 Constructor & Destructor Documentation

10.3.1.1 Enemy()

```

Enemy::Enemy (
    int healthPoints,
    double initRotation,
    double x,
    double y )
  
```

Construct a new [Enemy](#) object.

Parameters

<i>healthPoints</i>	Initial HP of this object
<i>initRotation</i>	Initial rotation of this object
<i>x</i>	x-component of this object's location
<i>y</i>	y-component of this object's location

10.3.2 Member Function Documentation

10.3.2.1 contains()

```
bool Enemy::contains (
    double x,
    double y ) [virtual]
```

Parameters

<i>x</i>	
<i>y</i>	

Returns

bool

Implements [Entity](#).

10.3.2.2 get_type()

```
bodyType Enemy::get_type ( ) const [override], [virtual]
```

Get the type object, in this case bodyType::Enemy.

Returns

bodyType

Implements [Entity](#).

The documentation for this class was generated from the following files:

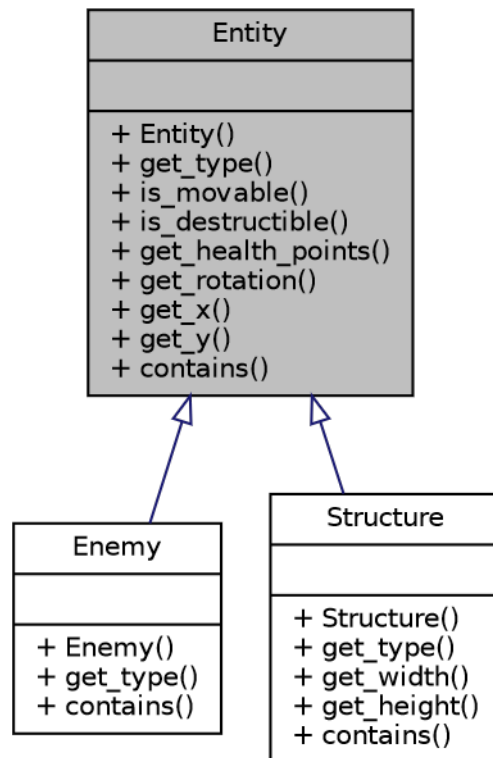
- src/[enemy.hpp](#)
- src/[enemy.cpp](#)

10.4 Entity Class Reference

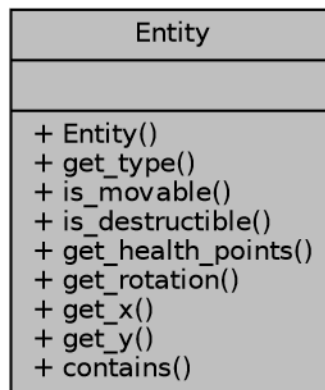
[Entity](#) class to handle entity data loaded from levels.

```
#include <entity.hpp>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



Public Member Functions

- [Entity](#) (bool movable, bool destructible, int healthPoints, double initRotation, double x, double y)
Construct a new [Entity](#) object.
- virtual [bodyType](#) [get_type](#) () const =0
Get the type object.
- bool [is_movable](#) () const
Is entity movable.
- bool [is_destructible](#) () const
Is entity destructible.
- int [get_health_points](#) () const
Get the initial health points.
- double [get_rotation](#) () const
Get the initial rotation.
- double [get_x](#) () const
Get the initial x pos.
- double [get_y](#) () const
Get the initial y pos.
- virtual bool [contains](#) (double x, double y)=0
Is a point contained inside the entity.

10.4.1 Detailed Description

[Entity](#) class to handle entity data loaded from levels.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 Entity()

```
Entity::Entity (
    bool movable,
    bool destructible,
    int healthPoints,
    double initRotation,
    double x,
    double y )
```

Construct a new [Entity](#) object.

Parameters

<i>movable</i>	
<i>destructible</i>	
<i>healthPoints</i>	
<i>initRotation</i>	
<i>x</i>	
<i>y</i>	

10.4.3 Member Function Documentation

10.4.3.1 contains()

```
virtual bool Entity::contains (
    double x,
    double y ) [pure virtual]
```

Is a point contained inside the entity.

Parameters

<i>x</i>	
<i>y</i>	

Returns

true Is contained
false Not contained

Implemented in [Structure](#), and [Enemy](#).

10.4.3.2 `get_health_points()`

```
int Entity::get_health_points ( ) const
```

Get the initial health points.

Returns

int

10.4.3.3 `get_rotation()`

```
double Entity::get_rotation ( ) const
```

Get the initial rotation.

Returns

double

10.4.3.4 `get_type()`

```
virtual bodyType Entity::get_type ( ) const [pure virtual]
```

Get the type object.

Returns

[bodyType](#)

Implemented in [Structure](#), and [Enemy](#).

10.4.3.5 `get_x()`

```
double Entity::get_x ( ) const
```

Get the initial x pos.

Returns

double

10.4.3.6 `get_y()`

```
double Entity::get_y ( ) const
```

Get the initial y pos.

Returns

double

10.4.3.7 `is_destructible()`

```
bool Entity::is_destructible ( ) const
```

Is entity destructible.

Returns

true

false

10.4.3.8 `is_movable()`

```
bool Entity::is_movable ( ) const
```

Is entity movable.

Returns

true

false

The documentation for this class was generated from the following files:

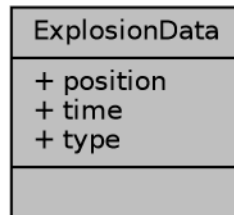
- [src/entity.hpp](#)
- [src/entity.cpp](#)

10.5 ExplosionData Struct Reference

[Structure](#) to store data for each explosion instance.

```
#include <playScene.hpp>
```

Collaboration diagram for ExplosionData:



Public Attributes

- `b2Vec2` [position](#)
- `float` [time](#)
- `explosionType` [type](#)

10.5.1 Detailed Description

[Structure](#) to store data for each explosion instance.

10.5.2 Member Data Documentation

10.5.2.1 position

```
b2Vec2 ExplosionData::position
```

10.5.2.2 time

```
float ExplosionData::time
```

10.5.2.3 type

`explosionType` ExplosionData::type

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

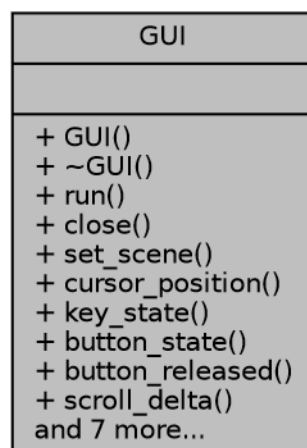
- [src/playScene.hpp](#)

10.6 GUI Class Reference

[GUI](#) Class to create window and run the game.

```
#include <GUI.hpp>
```

Collaboration diagram for GUI:



Public Member Functions

- [GUI](#) ()
Construct a new [GUI](#) object.
- [~GUI](#) ()
Destroy the [GUI](#) object.
- void [run](#) ()
Start the application.
- void [close](#) ()
- template<typename SceneType , typename... Params>
void [set_scene](#) (Params...params)
Set the current [Scene](#) object to be replaced at start of next frame.
- std::pair< float, float > [cursor_position](#) () const

- Get current cursor position.*
- bool [key_state](#) (sf::Keyboard::Key key) const
- Return state of keyboard key.*
- bool [button_state](#) (sf::Mouse::Button btn) const
- Return state of mouse button.*
- bool [button_released](#) (sf::Mouse::Button btn) const
- int [scroll_delta](#) () const
- float [get_aspect_ratio](#) () const
- void [set_viewport](#) (float x, float y, float w, float h)
- void [draw_sprite](#) (float x, float y, float w, float h, float angle, const [Image](#) &img)
- Draw a rectangular sprite on screen.*
- void [draw_rect](#) (float x, float y, float w, float h, float angle, [Color](#) color)
- Draw a rectangle with a color.*
- void [draw_text](#) (float x, float y, float h, const std::string &text, [Alignment](#) align=[Alignment::Center](#), sf::Color color=sf::Color::White)
- Draw text to screen with text centering.*
- bool [draw_button](#) (const std::string &text, float x, float y, float w, float h)
- Draw a button on screen. When mouse is hovered and pressed over button returns true.*
- void [play_sound](#) (std::string path, int vol=100)
- Play sound from given path. Caches sound file inside class.*

10.6.1 Detailed Description

[GUI](#) Class to create window and run the game.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 GUI()

```
GUI::GUI ( )
```

Construct a new [GUI](#) object.

10.6.2.2 ~GUI()

```
GUI::~GUI ( )
```

Destroy the [GUI](#) object.

10.6.3 Member Function Documentation

10.6.3.1 button_released()

```
bool GUI::button_released (
    sf::Mouse::Button btn ) const
```

10.6.3.2 button_state()

```
bool GUI::button_state (
    sf::Mouse::Button btn ) const
```

Return state of mouse button.

Parameters

<i>btn</i>	Button id
------------	-----------

Returns

true Button is pressed
false Button is released

10.6.3.3 close()

```
void GUI::close ( )
```

10.6.3.4 cursor_position()

```
std::pair< float, float > GUI::cursor_position ( ) const
```

Get current cursor position.

Returns

std::pair<float, float> position in [0, 1] range

10.6.3.5 draw_button()

```
bool GUI::draw_button (
    const std::string & text,
    float x,
    float y,
    float w,
    float h )
```

Draw a button on screen. When mouse is hovered and pressed over button returns true.

Parameters

<i>text</i>	
<i>x</i>	
<i>y</i>	
<i>w</i>	
<i>h</i>	
<i>button_image</i>	

Returns

true Button is released

false otherwise

10.6.3.6 draw_rect()

```
void GUI::draw_rect (
    float x,
    float y,
    float w,
    float h,
    float angle,
    Color color )
```

Draw a rectangle with a color.

Parameters

<i>x</i>	
<i>y</i>	
<i>w</i>	
<i>h</i>	
<i>angle</i>	
<i>color</i>	

10.6.3.7 draw_sprite()

```
void GUI::draw_sprite (
    float x,
    float y,
    float w,
    float h,
    float angle,
    const Image & img )
```

Draw a rectangular sprite on screen.

Parameters

<i>x</i>	
<i>y</i>	
<i>w</i>	
<i>h</i>	
<i>angle</i>	
<i>img</i>	

10.6.3.8 draw_text()

```
void GUI::draw_text (
    float x,
    float y,
    float h,
    const std::string & text,
    Alignment align = Alignment::Center,
    sf::Color color = sf::Color::White )
```

Draw text to screen with text centering.

Parameters

<i>x</i>	xpos
<i>y</i>	ypos
<i>h</i>	text scale
<i>text</i>	text string

10.6.3.9 get_aspect_ratio()

```
float GUI::get_aspect_ratio ( ) const
```

10.6.3.10 key_state()

```
bool GUI::key_state (
    sf::Keyboard::Key key ) const
```

Return state of keyboard key.

Parameters

<i>key</i>	
------------	--

Returns

true Key is pressed
false Key is released

10.6.3.11 play_sound()

```
void GUI::play_sound (
    std::string path,
    int vol = 100 )
```

Play sound from given path. Caches sound file inside class.

Parameters

<i>path</i>	Path to file
<i>vol</i>	Volume to play at

10.6.3.12 run()

```
void GUI::run ( )
```

Start the application.

10.6.3.13 scroll_delta()

```
int GUI::scroll_delta ( ) const
```

10.6.3.14 set_scene()

```
template<typename SceneType , typename... Params>
void GUI::set_scene (
    Params... params ) [inline]
```

Set the current [Scene](#) object to be replaced at start of next frame.

Template Parameters

<i>SceneType</i>	scene class type
<i>Params</i>	argument types of SceneType constructor

Parameters

<i>params</i>	arguments for constructing new class of SceneType
---------------	---

10.6.3.15 set_viewport()

```
void GUI::set_viewport (
    float x,
    float y,
    float w,
    float h )
```

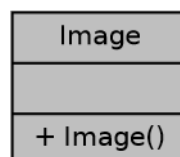
The documentation for this class was generated from the following files:

- [src/GUI.hpp](#)
- [src/GUI.cpp](#)

10.7 Image Class Reference

```
#include <image.hpp>
```

Collaboration diagram for Image:

**Public Member Functions**

- [Image](#) (const std::string &path)
Construct a new [Image](#) object.

Friends

- class [GUI](#)

10.7.1 Constructor & Destructor Documentation

10.7.1.1 Image()

```
Image::Image (
    const std::string & path )
```

Construct a new [Image](#) object.

Parameters

<i>path</i>	Path to image
-------------	---------------

10.7.2 Friends And Related Function Documentation

10.7.2.1 GUI

```
friend class GUI [friend]
```

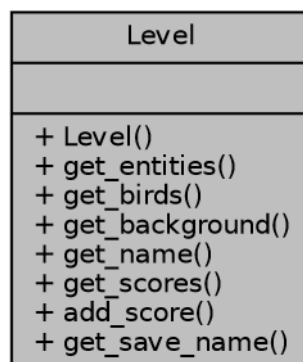
The documentation for this class was generated from the following files:

- [src/image.hpp](#)
- [src/image.cpp](#)

10.8 Level Class Reference

```
#include <level.hpp>
```

Collaboration diagram for Level:



Public Member Functions

- [Level](#) (std::vector< std::shared_ptr< [Entity](#) >> entities, std::vector< std::shared_ptr< [Bird](#) >> birds, std::string [backgroundPath](#), std::string name, std::vector< [ScoreBoardEntry](#) > scores, std::string save_name)
Construct a new [Level](#) object.
- std::vector< std::shared_ptr< [Entity](#) >> [get_entities](#) () const
Get the list of entities in level.
- std::vector< std::shared_ptr< [Bird](#) >> [get_birds](#) () const
Get the list of bird objects.
- std::string [get_background](#) () const
Get path to the background file.
- std::string [get_name](#) () const
Get the name of the [Level](#).
- std::vector< [ScoreBoardEntry](#) > [get_scores](#) () const
Get the scores reached in the level.
- void [add_score](#) (std::string name, int score)
Register a new score to the level.
- std::string [get_save_name](#) () const
Get the path to the level.

10.8.1 Constructor & Destructor Documentation

10.8.1.1 Level()

```
Level::Level (
    std::vector< std::shared_ptr< Entity >> entities,
    std::vector< std::shared_ptr< Bird >> birds,
    std::string backgroundPath,
    std::string name,
    std::vector< ScoreBoardEntry > scores,
    std::string save_name )
```

Construct a new [Level](#) object.

Parameters

<i>entities</i>	List of entities
<i>birds</i>	List of birds
<i>backgroundPath</i>	Path to background
<i>name</i>	Name of level
<i>scores</i>	List of scores reached in level
<i>save_name</i>	Path to save file

10.8.2 Member Function Documentation

10.8.2.1 add_score()

```
void Level::add_score (
    std::string name,
    int score )
```

Register a new score to the level.

Parameters

<i>name</i>	Name of player
<i>score</i>	Score reached

10.8.2.2 get_background()

```
std::string Level::get_background ( ) const
```

Get path to the background file.

Returns

std::string Path to file

10.8.2.3 get_birds()

```
std::vector< std::shared_ptr< Bird > > Level::get_birds ( ) const
```

Get the list of bird objects.

Returns

std::vector<std::shared_ptr<Bird>>

10.8.2.4 get_entities()

```
std::vector< std::shared_ptr< Entity > > Level::get_entities ( ) const
```

Get the list of entities in level.

Returns

std::vector<std::shared_ptr<Entity>> The list of entities

10.8.2.5 `get_name()`

```
std::string Level::get_name ( ) const
```

Get the name of the [Level](#).

Returns

std::string

10.8.2.6 `get_save_name()`

```
std::string Level::get_save_name ( ) const
```

Get the path to the level.

Returns

std::string Path

10.8.2.7 `get_scores()`

```
std::vector< ScoreBoardEntry > Level::get_scores ( ) const
```

Get the scores reached in the level.

Returns

std::vector<ScoreBoardEntry> List of scores

The documentation for this class was generated from the following files:

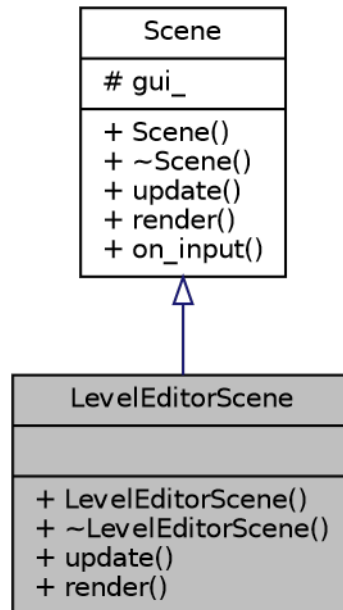
- [src/level.hpp](#)
- [src/level.cpp](#)

10.9 LevelEditorScene Class Reference

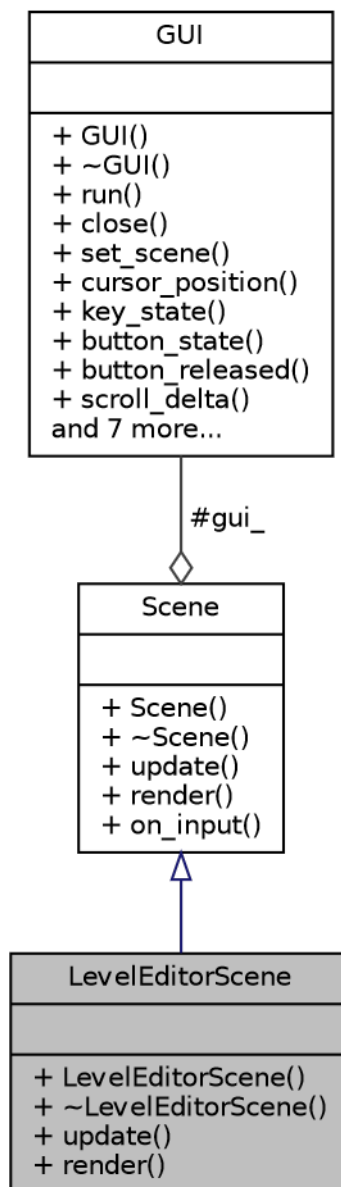
A scene for editing the playable levels in the game.

```
#include <levelEditorScene.hpp>
```

Inheritance diagram for LevelEditorScene:



Collaboration diagram for LevelEditorScene:



Public Member Functions

- [LevelEditorScene](#) ([GUI](#) &gui, [Level](#) &level, const std::string current_player)
Construct a new [Level Editor Scene](#).
- [~LevelEditorScene](#) ()
Destroy the [Level Editor Scene](#) object.
- void [update](#) (float ts) override
Update function to update everything required in level editor.
- void [render](#) () override
Render the level editor.

Additional Inherited Members

10.9.1 Detailed Description

A scene for editing the playable levels in the game.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 LevelEditorScene()

```
LevelEditorScene::LevelEditorScene (
    GUI & gui,
    Level & level,
    const std::string current_player )
```

Construct a new [Level](#) Editor [Scene](#).

Parameters

<i>gui</i>	Reference to GUI
<i>level</i>	Path to level to edit
<i>current_player</i>	Current player name

10.9.2.2 ~LevelEditorScene()

```
LevelEditorScene::~~LevelEditorScene ( )
```

Destroy the [Level](#) Editor [Scene](#) object.

10.9.3 Member Function Documentation

10.9.3.1 render()

```
void LevelEditorScene::render ( ) [override], [virtual]
```

Render the level editor.

Implements [Scene](#).

10.9.3.2 update()

```
void LevelEditorScene::update (  
    float ts )  [override], [virtual]
```

Update function to update everything required in level editor.

Parameters

<i>ts</i>	Timestep
-----------	----------

Implements [Scene](#).

The documentation for this class was generated from the following files:

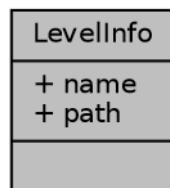
- [src/levelEditorScene.hpp](#)
- [src/levelEditorScene.cpp](#)

10.10 LevelInfo Struct Reference

Information about levels.

```
#include <readerWriter.hpp>
```

Collaboration diagram for LevelInfo:



Public Attributes

- `std::string` [name](#)
- `std::string` [path](#)

10.10.1 Detailed Description

Information about levels.

10.10.2 Member Data Documentation

10.10.2.1 name

```
std::string LevelInfo::name
```

10.10.2.2 path

```
std::string LevelInfo::path
```

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

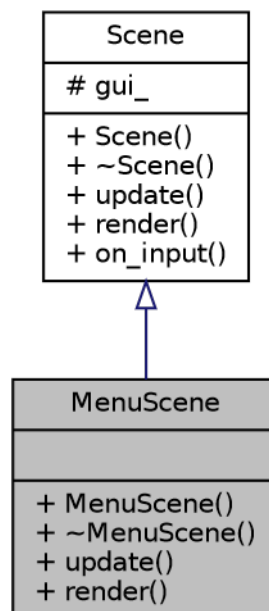
- [src/readerWriter.hpp](#)

10.11 MenuScene Class Reference

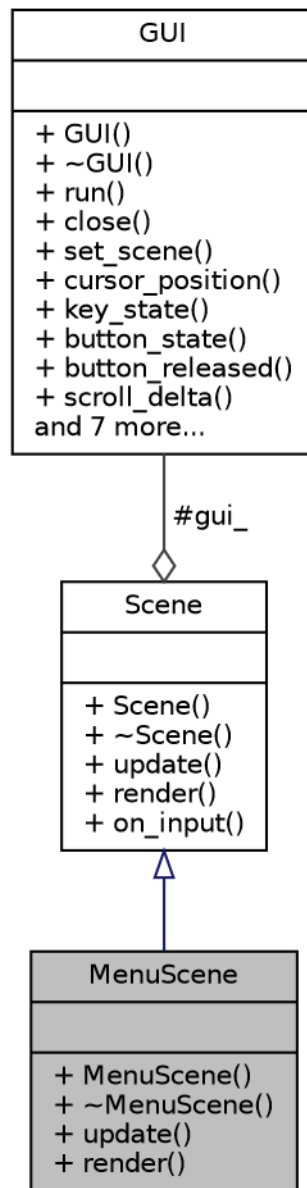
[Scene](#) to handle the main menu.

```
#include <menuScene.hpp>
```

Inheritance diagram for MenuScene:



Collaboration diagram for MenuScene:



Public Member Functions

- [MenuScene](#) ([GUI](#) &gui, std::string current_player)
Construct a new Menu [Scene](#).
- [~MenuScene](#) ()
Destroy the Menu [Scene](#) object.
- void [update](#) (float ts) override
Update everything required in the main menu.
- void [render](#) () override
Render the [MenuScene](#).

Additional Inherited Members

10.11.1 Detailed Description

[Scene](#) to handle the main menu.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 MenuScene()

```
MenuScene::MenuScene (
    GUI & gui,
    std::string current_player )
```

Construct a new Menu [Scene](#).

Parameters

<i>gui</i>	Reference to GUI
<i>current_player</i>	Name of current player

10.11.2.2 ~MenuScene()

```
MenuScene::~MenuScene ( )
```

Destroy the Menu [Scene](#) object.

10.11.3 Member Function Documentation

10.11.3.1 render()

```
void MenuScene::render ( ) [override], [virtual]
```

Render the [MenuScene](#).

Implements [Scene](#).

10.11.3.2 update()

```
void MenuScene::update (
    float ts ) [override], [virtual]
```

Update everything required in the main menu.

Parameters

<i>ts</i>	Timestep
-----------	----------

Implements [Scene](#).

The documentation for this class was generated from the following files:

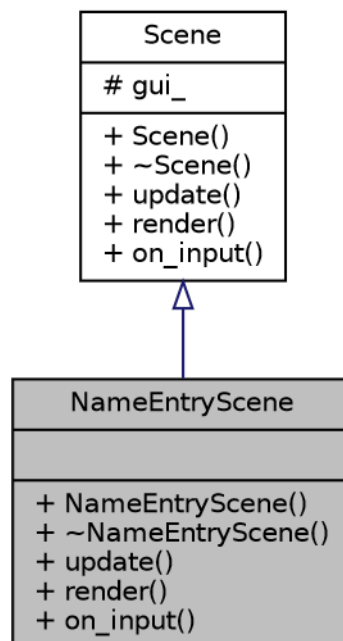
- [src/menuScene.hpp](#)
- [src/menuScene.cpp](#)

10.12 NameEntryScene Class Reference

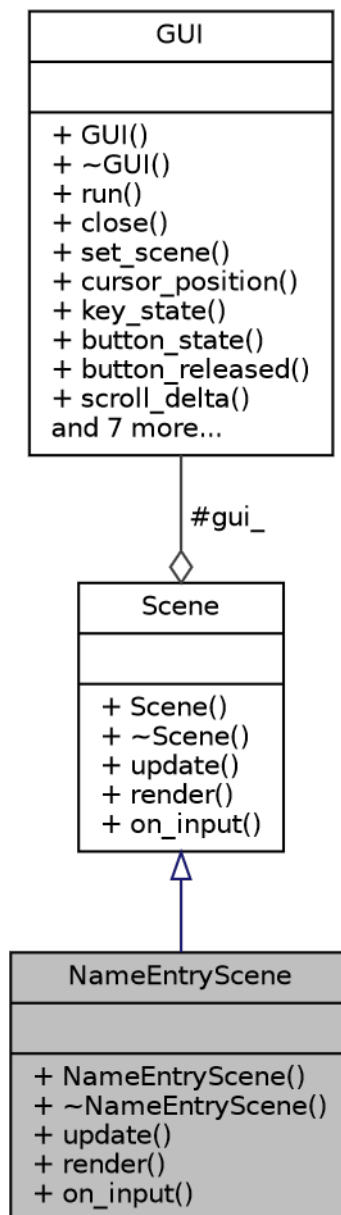
A scene for the user to enter their name from.

```
#include <nameEntryScene.hpp>
```

Inheritance diagram for NameEntryScene:



Collaboration diagram for NameEntryScene:



Public Member Functions

- [NameEntryScene](#) ([GUI](#) &gui)
Construct a new Name Entry [Scene](#) object.
- [~NameEntryScene](#) ()
Destroy the Name Entry [Scene](#) object.
- void [update](#) (float ts) override
Function to update required.

- void `render` () override
Function to render scene.
- void `on_input` (char c) override
Handle input events.

Additional Inherited Members

10.12.1 Detailed Description

A scene for the user to enter their name from.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 NameEntryScene()

```
NameEntryScene::NameEntryScene (  
    GUI & gui )
```

Construct a new Name Entry `Scene` object.

Parameters

<code>gui</code>	Reference to <code>GUI</code>
------------------	-------------------------------

10.12.2.2 ~NameEntryScene()

```
NameEntryScene::~~NameEntryScene ( )
```

Destroy the Name Entry `Scene` object.

10.12.3 Member Function Documentation

10.12.3.1 on_input()

```
void NameEntryScene::on_input (  
    char c ) [override], [virtual]
```

Handle input events.

Parameters

<i>c</i>	Character entered
----------	-------------------

Reimplemented from [Scene](#).

10.12.3.2 render()

```
void NameEntryScene::render ( ) [override], [virtual]
```

Function to render scene.

Implements [Scene](#).

10.12.3.3 update()

```
void NameEntryScene::update (
    float ts ) [override], [virtual]
```

Function to update required.

Parameters

<i>ts</i>	Timestep
-----------	----------

Implements [Scene](#).

The documentation for this class was generated from the following files:

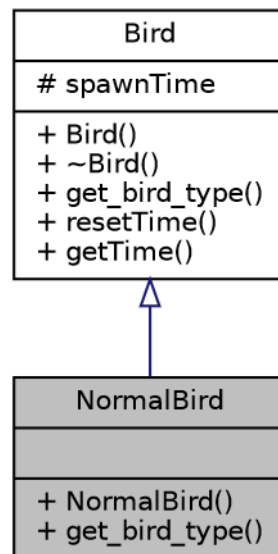
- [src/nameEntryScene.hpp](#)
- [src/nameEntryScene.cpp](#)

10.13 NormalBird Class Reference

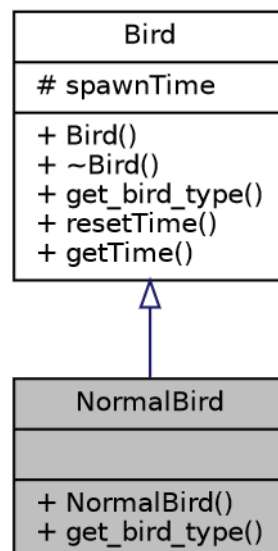
Normal bird class No special effects.

```
#include <bird.hpp>
```

Inheritance diagram for NormalBird:



Collaboration diagram for NormalBird:



Public Member Functions

- [NormalBird](#) ()
Construct a new Normal [Bird](#) object.
- [birdType](#) [get_bird_type](#) () const override
Get the bird type object.

Additional Inherited Members

10.13.1 Detailed Description

Normal bird class No special effects.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 NormalBird()

```
NormalBird::NormalBird ( )
```

Construct a new Normal [Bird](#) object.

10.13.3 Member Function Documentation

10.13.3.1 get_bird_type()

```
birdType NormalBird::get_bird_type ( ) const [override], [virtual]
```

Get the bird type object.

Returns

[birdType::normal](#)

Implements [Bird](#).

The documentation for this class was generated from the following files:

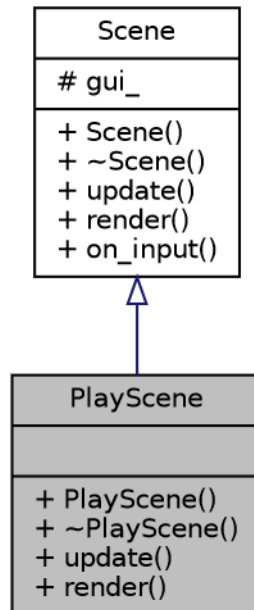
- [src/bird.hpp](#)
- [src/bird.cpp](#)

10.14 PlayScene Class Reference

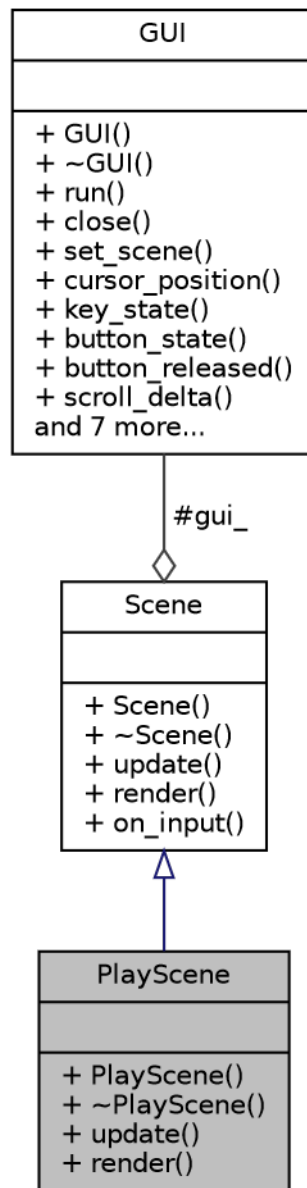
The game play scene to play the levels in the game.

```
#include <playScene.hpp>
```

Inheritance diagram for PlayScene:



Collaboration diagram for PlayScene:



Public Member Functions

- `PlayScene (GUI &gui, const Level &level, std::string current_player)`
Construct a new *Play Scene* object.
- `~PlayScene ()`
Destroy the *Play Scene* object.
- `void update (float ts) override`
Update everything required by the *PlayScene*.
- `void render () override`
Render everything in the *PlayScene*.

Additional Inherited Members

10.14.1 Detailed Description

The game play scene to play the levels in the game.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 PlayScene()

```
PlayScene::PlayScene (
    GUI & gui,
    const Level & level,
    std::string current_player )
```

Construct a new Play [Scene](#) object.

Parameters

<i>gui</i>	Reference to GUI
<i>level</i>	Level object to play
<i>current_player</i>	Name of current player

10.14.2.2 ~PlayScene()

```
PlayScene::~~PlayScene ( )
```

Destroy the Play [Scene](#) object.

10.14.3 Member Function Documentation

10.14.3.1 render()

```
void PlayScene::render ( ) [override], [virtual]
```

Render everything in the [PlayScene](#).

Implements [Scene](#).

10.14.3.2 update()

```
void PlayScene::update (  
    float ts )  [override], [virtual]
```

Update everything required by the [PlayScene](#).

Parameters

<i>ts</i>	Timestep
-----------	----------

Implements [Scene](#).

The documentation for this class was generated from the following files:

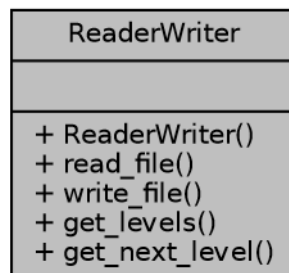
- [src/playScene.hpp](#)
- [src/playScene.cpp](#)

10.15 ReaderWriter Class Reference

A class for loading/reading and saving/writing levels in forms of [Level](#) objects and .txt files.

```
#include <readerWriter.hpp>
```

Collaboration diagram for ReaderWriter:



Public Member Functions

- [ReaderWriter](#) ()
Constructor.
- `std::optional< Level > read_file (std::string fileName) const`
Given the file name as the parameter loads the level.
- `void write_file (Level level, std::string fileName) const`
Saves the provided level into a file named fileName.
- `std::vector< LevelInfo > get_levels () const`
Returns a list of levels that can be loaded.
- `std::optional< std::string > get_next_level (std::string) const`
Get the name of the next level in sequence.

10.15.1 Detailed Description

A class for loading/reading and saving/writing levels in forms of [Level](#) objects and .txt files.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 ReaderWriter()

```
ReaderWriter::ReaderWriter ( ) [inline]
```

Constructor.

10.15.3 Member Function Documentation

10.15.3.1 get_levels()

```
std::vector< LevelInfo > ReaderWriter::get_levels ( ) const
```

Returns a list of levels that can be loaded.

10.15.3.2 get_next_level()

```
std::optional< std::string > ReaderWriter::get_next_level (
    std::string path ) const
```

Get the name of the next level in sequence.

Returns

std::optional<std::string> Name of level if there is a next level

10.15.3.3 read_file()

```
std::optional< Level > ReaderWriter::read_file (
    std::string fileName ) const
```

Given the file name as the parameter loads the level.

Returns

std::optional<Level> Loaded level

10.15.3.4 write_file()

```
void ReaderWriter::write_file (
    Level level,
    std::string fileName ) const
```

Saves the provided level into a file named fileName.

The documentation for this class was generated from the following files:

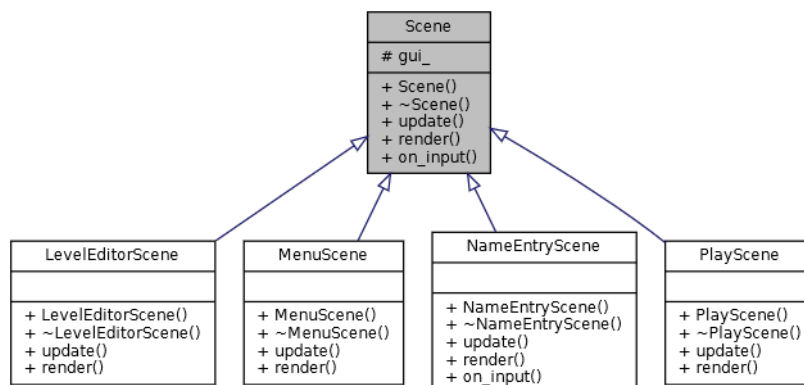
- [src/readerWriter.hpp](#)
- [src/readerWriter.cpp](#)

10.16 Scene Class Reference

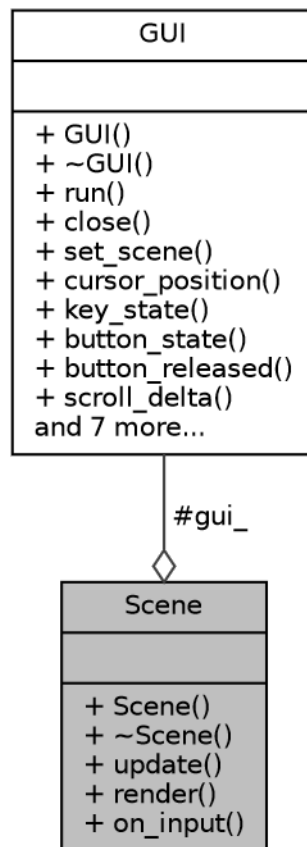
Base [Scene](#) class to represent different states the game can be in. Ex. Play state can be implemented with [PlayScene](#). Menu can be implemented with [MenuScene](#) etc.

```
#include <scene.hpp>
```

Inheritance diagram for Scene:



Collaboration diagram for Scene:



Public Member Functions

- [Scene](#) ([GUI](#) &gui)
Construct a new [Scene](#).
- virtual [~Scene](#) ()
Destroy the [Scene](#) object.
- virtual void [update](#) (float ts)=0
Called on an interval to update state of scene.
- virtual void [render](#) ()=0
- virtual void [on_input](#) (char c)
Called when input has been entered and should be passed to scene.

Protected Attributes

- [GUI](#) & [gui_](#)

10.16.1 Detailed Description

Base [Scene](#) class to represent different states the game can be in. Ex. Play state can be implemented with [PlayScene](#). Menu can be implemented with [MenuScene](#) etc.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 Scene()

```
Scene::Scene (
    GUI & gui )
```

Construct a new [Scene](#).

Parameters

<i>gui</i>	Reference to GUI
------------	----------------------------------

10.16.2.2 ~Scene()

```
Scene::~Scene ( ) [virtual]
```

Destroy the [Scene](#) object.

10.16.3 Member Function Documentation

10.16.3.1 on_input()

```
virtual void Scene::on_input (
    char c ) [inline], [virtual]
```

Called when input has been entered and should be passed to scene.

Parameters

<i>c</i>	Character
----------	-----------

Reimplemented in [NameEntryScene](#).

10.16.3.2 render()

```
virtual void Scene::render ( ) [pure virtual]
```

Implemented in [PlayScene](#), [NameEntryScene](#), [MenuScene](#), and [LevelEditorScene](#).

10.16.3.3 update()

```
virtual void Scene::update (
    float ts ) [pure virtual]
```

Called on an interval to update state of scene.

Parameters

<i>ts</i>	Timestep since last update
-----------	----------------------------

Implemented in [PlayScene](#), [NameEntryScene](#), [MenuScene](#), and [LevelEditorScene](#).

10.16.4 Member Data Documentation

10.16.4.1 gui_

```
GUI& Scene::gui_ [protected]
```

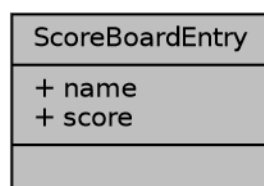
The documentation for this class was generated from the following files:

- [src/scene.hpp](#)
- [src/scene.cpp](#)

10.17 ScoreBoardEntry Struct Reference

```
#include <level.hpp>
```

Collaboration diagram for ScoreBoardEntry:



Public Attributes

- std::string [name](#)
- int [score](#)

10.17.1 Member Data Documentation

10.17.1.1 name

```
std::string ScoreBoardEntry::name
```

10.17.1.2 score

```
int ScoreBoardEntry::score
```

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

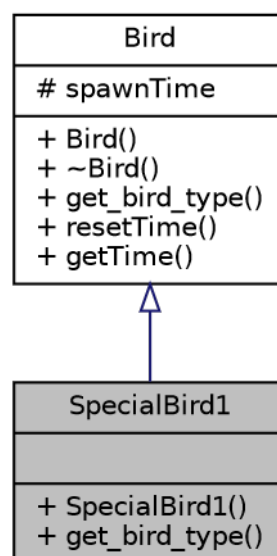
- src/[level.hpp](#)

10.18 SpecialBird1 Class Reference

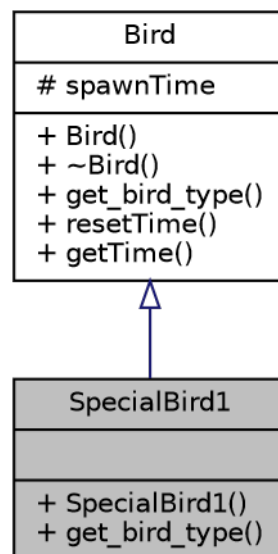
Special [Bird](#) 1 class Has speed up effect.

```
#include <bird.hpp>
```

Inheritance diagram for SpecialBird1:



Collaboration diagram for SpecialBird1:



Public Member Functions

- [SpecialBird1](#) ()
Construct a new Special [Bird](#) 1 object.
- [birdType](#) [get_bird_type](#) () const override
Get the bird type object.

Additional Inherited Members

10.18.1 Detailed Description

Special [Bird](#) 1 class Has speed up effect.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 SpecialBird1()

```
SpecialBird1::SpecialBird1 ( )
```

Construct a new Special [Bird](#) 1 object.

10.18.3 Member Function Documentation

10.18.3.1 get_bird_type()

```
birdType SpecialBird1::get_bird_type ( ) const [override], [virtual]
```

Get the bird type object.

Returns

[birdType::special1](#)

Implements [Bird](#).

The documentation for this class was generated from the following files:

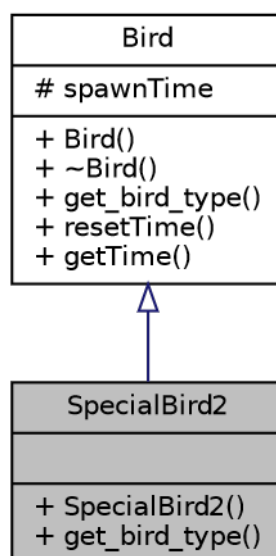
- [src/bird.hpp](#)
- [src/bird.cpp](#)

10.19 SpecialBird2 Class Reference

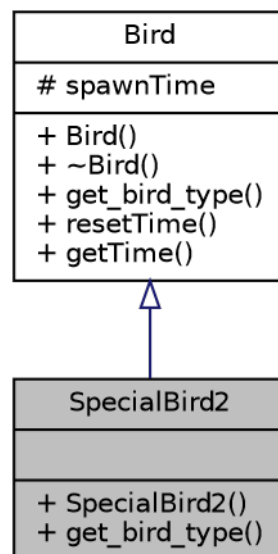
Special [Bird](#) 2 class Has explosion effect.

```
#include <bird.hpp>
```

Inheritance diagram for SpecialBird2:



Collaboration diagram for SpecialBird2:



Public Member Functions

- [SpecialBird2](#) ()
Construct a new Special [Bird](#) 2 object.
- [birdType](#) [get_bird_type](#) () const override
Get the bird type object.

Additional Inherited Members

10.19.1 Detailed Description

Special [Bird](#) 2 class Has explosion effect.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 SpecialBird2()

```
SpecialBird2::SpecialBird2 ( )
```

Construct a new Special [Bird](#) 2 object.

10.19.3 Member Function Documentation

10.19.3.1 `get_bird_type()`

`birdType` `SpecialBird2::get_bird_type () const [override], [virtual]`

Get the bird type object.

Returns

`birdType::special2`

Implements [Bird](#).

The documentation for this class was generated from the following files:

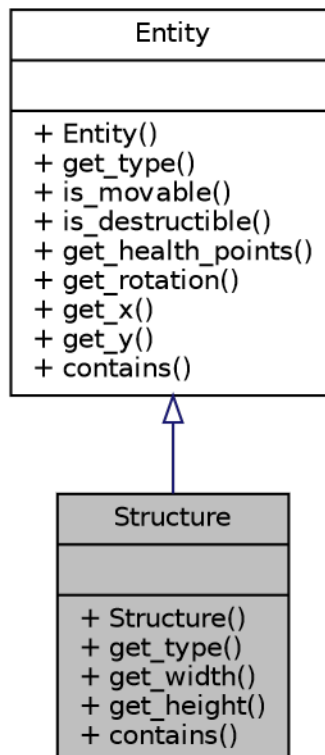
- [src/bird.hpp](#)
- [src/bird.cpp](#)

10.20 Structure Class Reference

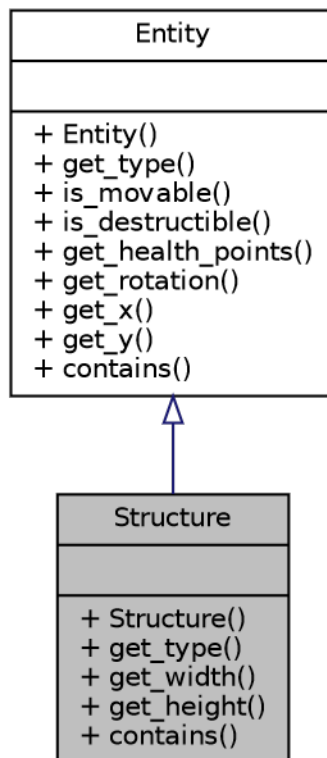
`Structure` type to represent walls in the game.

```
#include <structure.hpp>
```

Inheritance diagram for `Structure`:



Collaboration diagram for Structure:



Public Member Functions

- [Structure](#) (int healthPoints, double initRotation, double x, double y, double height, double width)
Construct a new [Structure](#).
- `bodyType get_type ()` const override
Get the type object.
- `double get_width ()` const
Get the width of structure.
- `double get_height ()` const
Get the height the strucutre.
- `bool contains (double x, double y)`
Returns true if contains the coordinate passed.

10.20.1 Detailed Description

[Structure](#) type to represent walls in the game.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 Structure()

```
Structure::Structure (
    int healthPoints,
    double initRotation,
    double x,
    double y,
    double height,
    double width )
```

Construct a new [Structure](#).

Parameters

<i>healthPoints</i>	HP
<i>initRotation</i>	Rotation
<i>x</i>	X position
<i>y</i>	Y position
<i>height</i>	Height of structure
<i>width</i>	Width of structure

10.20.3 Member Function Documentation

10.20.3.1 contains()

```
bool Structure::contains (
    double x,
    double y ) [virtual]
```

Returns true if contains the coordinate passed.

Parameters

<i>x</i>	X-coord
<i>y</i>	Y-coord

Returns

true Contained
false Not contained

Implements [Entity](#).

10.20.3.2 `get_height()`

```
double Structure::get_height ( ) const
```

Get the height the strucutre.

Returns

double Height

10.20.3.3 `get_type()`

```
bodyType Structure::get_type ( ) const [override], [virtual]
```

Get the type object.

Returns

[bodyType::structure](#)

Implements [Entity](#).

10.20.3.4 `get_width()`

```
double Structure::get_width ( ) const
```

Get the width of structure.

Returns

double Width

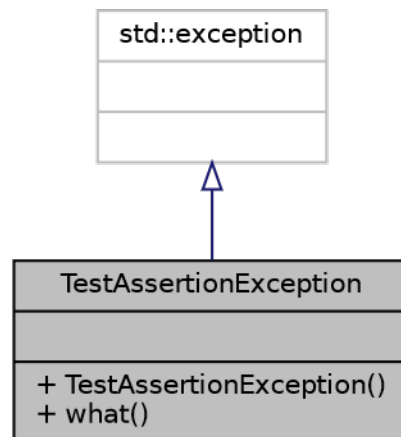
The documentation for this class was generated from the following files:

- [src/structure.hpp](#)
- [src/structure.cpp](#)

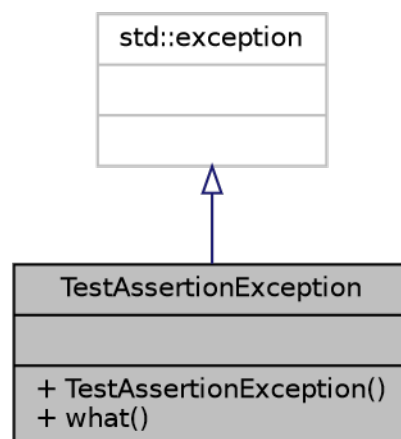
10.21 TestAssertionException Class Reference

```
#include <test_utils.hpp>
```

Inheritance diagram for TestAssertionException:



Collaboration diagram for TestAssertionException:



Public Member Functions

- [TestAssertionException](#) (const std::string &message)
- const char * [what](#) () const noexcept override

10.21.1 Constructor & Destructor Documentation

10.21.1.1 TestAssertionException()

```
TestAssertionException::TestAssertionException (
    const std::string & message ) [inline]
```

10.21.2 Member Function Documentation

10.21.2.1 what()

```
const char* TestAssertionException::what ( ) const [inline], [override], [noexcept]
```

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

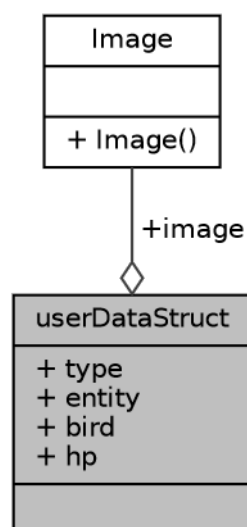
- tests/[test_utils.hpp](#)

10.22 userDataStruct Struct Reference

Physics entities hold a copy of [userDataStruct](#) they can reference.

```
#include <playScene.hpp>
```

Collaboration diagram for userDataStruct:



Public Attributes

- [Image](#) * [image](#)
- [bodyType](#) type
- std::shared_ptr< [Entity](#) > [entity](#)
- std::shared_ptr< [Bird](#) > [bird](#)
- int [hp](#)

10.22.1 Detailed Description

Physics entities hold a copy of [userDataStruct](#) they can reference.

10.22.2 Member Data Documentation

10.22.2.1 [bird](#)

```
std::shared_ptr<Bird> userDataStruct::bird
```

10.22.2.2 [entity](#)

```
std::shared_ptr<Entity> userDataStruct::entity
```

10.22.2.3 [hp](#)

```
int userDataStruct::hp
```

10.22.2.4 [image](#)

```
Image* userDataStruct::image
```

10.22.2.5 [type](#)

```
bodyType userDataStruct::type
```

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

- src/[playScene.hpp](#)

Chapter 11

File Documentation

11.1 build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define C_VERSION`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_standard_default`
- `const char * info_language_extensions_default`

11.1.1 Macro Definition Documentation

11.1.1.1 __has_include

```
#define __has_include(
    x ) 0
```

11.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

11.1.1.3 C_VERSION

```
#define C_VERSION
```

11.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

11.1.1.5 DEC

```
#define DEC(
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

11.1.1.6 HEX

```
#define HEX(
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \
('0' + ((n) >> 24 & 0xF)), \
('0' + ((n) >> 20 & 0xF)), \
('0' + ((n) >> 16 & 0xF)), \
('0' + ((n) >> 12 & 0xF)), \
('0' + ((n) >> 8 & 0xF)), \
('0' + ((n) >> 4 & 0xF)), \
('0' + ((n) & 0xF))
```

11.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

11.1.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

11.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

11.1.2 Function Documentation

11.1.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

11.1.3 Variable Documentation

11.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

11.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

11.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

11.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=  
  "INFO" ":" "standard_default[" C_VERSION "]"
```

11.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

11.2 build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define CXX_STD __cplusplus`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_standard_default`
- `const char * info_language_extensions_default`

11.2.1 Macro Definition Documentation

11.2.1.1 `__has_include`

```
#define __has_include(  
    x ) 0
```

11.2.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

11.2.1.3 `COMPILER_ID`

```
#define COMPILER_ID ""
```

11.2.1.4 `CXX_STD`

```
#define CXX_STD __cplusplus
```

11.2.1.5 `DEC`

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

11.2.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n)>>28 & 0xF)), \  
( '0' + ((n)>>24 & 0xF)), \  
( '0' + ((n)>>20 & 0xF)), \  
( '0' + ((n)>>16 & 0xF)), \  
( '0' + ((n)>>12 & 0xF)), \  
( '0' + ((n)>>8  & 0xF)), \  
( '0' + ((n)>>4  & 0xF)), \  
( '0' + ((n)    & 0xF))
```

11.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

11.2.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

11.2.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

11.2.2 Function Documentation

11.2.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )
```

11.2.3 Variable Documentation

11.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

11.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

11.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

11.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
= "INFO" ":" "standard_default["  
  "98"  
"]"
```

11.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

11.3 doc/readme.md File Reference

11.4 plan/readme.md File Reference

11.5 src/readme.md File Reference

11.6 Meeting-notes.md File Reference

11.7 README.md File Reference

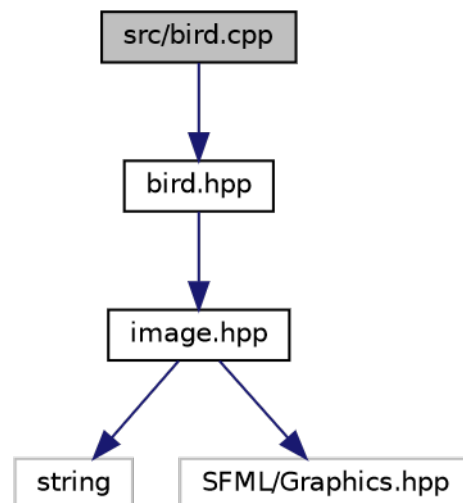
11.8 res/README.md File Reference

11.9 SavefileFormat.md File Reference

11.10 src/bird.cpp File Reference

```
#include "bird.hpp"
```

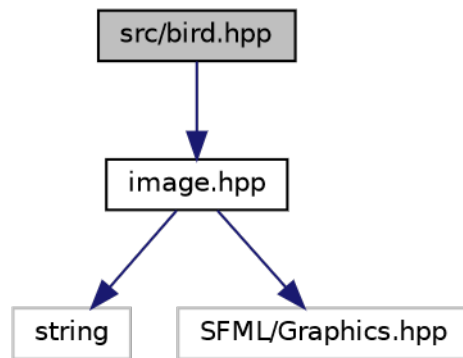
Include dependency graph for bird.cpp:



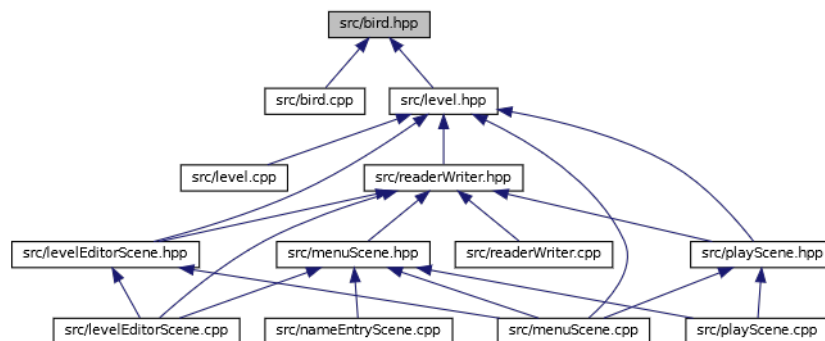
11.11 src/bird.hpp File Reference

```
#include "image.hpp"
```

Include dependency graph for bird.hpp:



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



Classes

- class [Bird](#)
Base class for [Bird](#) objects.
- class [NormalBird](#)
Normal bird class No special effects.
- class [SpecialBird1](#)
Special [Bird](#) 1 class Has speed up effect.
- class [SpecialBird2](#)
Special [Bird](#) 2 class Has explosion effect.

Enumerations

- enum class `birdType` { `normal` , `special1` , `special2` }

Enumeration of all different bird types.

11.11.1 Enumeration Type Documentation

11.11.1.1 `birdType`

```
enum birdType [strong]
```

Enumeration of all different bird types.

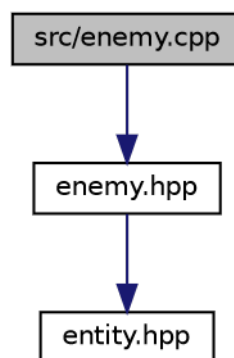
Enumerator

<code>normal</code>	
<code>special1</code>	
<code>special2</code>	

11.12 `src/enemy.cpp` File Reference

```
#include "enemy.hpp"
```

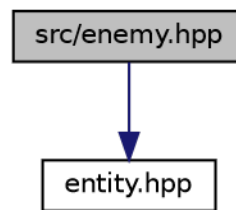
Include dependency graph for `enemy.cpp`:



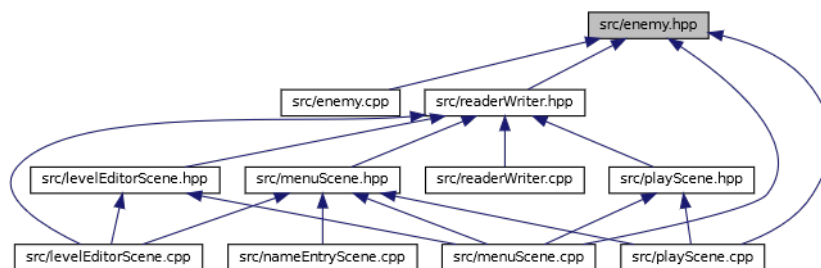
11.13 `src/enemy.hpp` File Reference

```
#include "entity.hpp"
```

Include dependency graph for enemy.hpp:



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



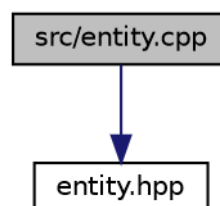
Classes

- class [Enemy](#)

11.14 src/entity.cpp File Reference

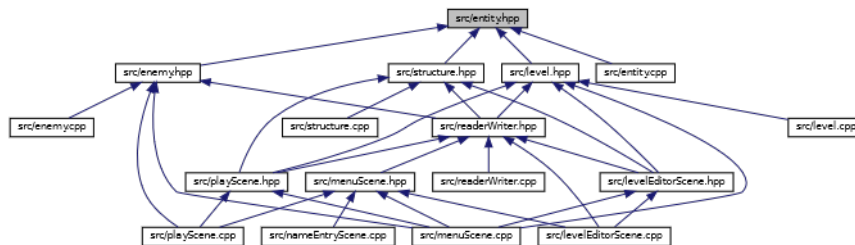
```
#include "entity.hpp"
```

Include dependency graph for entity.cpp:



11.15 src/entity.hpp File Reference

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



Classes

- class [Entity](#)
Entity class to handle entity data loaded from levels.

Enumerations

- enum class [bodyType](#) { [structure](#) , [enemy](#) , [ground](#) , [bird](#) }
Type of entity.

11.15.1 Enumeration Type Documentation

11.15.1.1 bodyType

```
enum bodyType [strong]
```

Type of entity.

Enumerator

structure	
enemy	
ground	
bird	

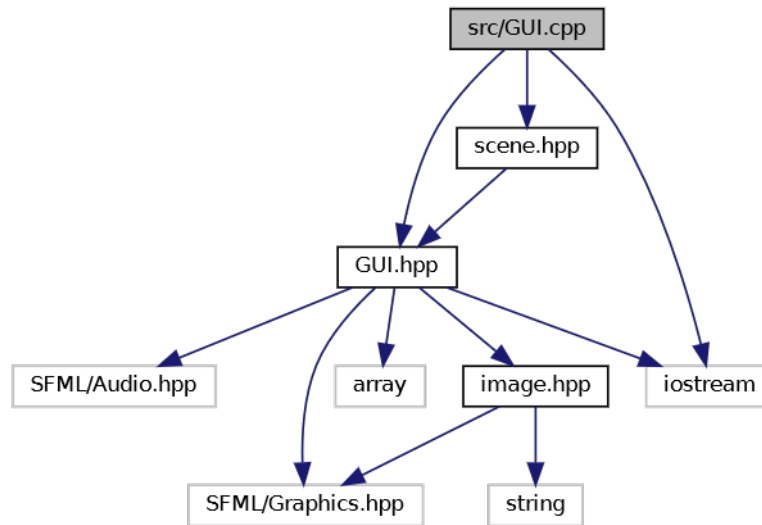
11.16 src/GUI.cpp File Reference

```
#include "GUI.hpp"
#include "scene.hpp"
```



```
#include <iostream>
```

Include dependency graph for GUI.cpp:



11.17 src/GUI.hpp File Reference

```
#include <SFML/Audio.hpp>
```

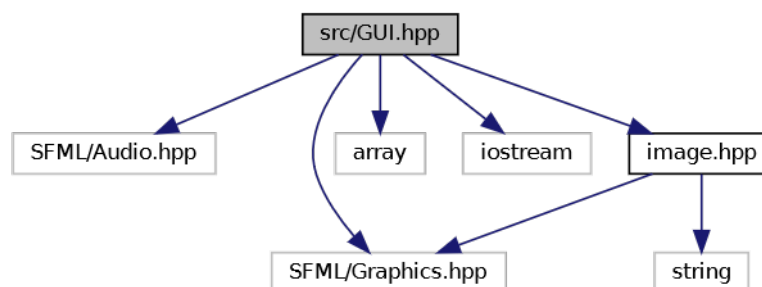
```
#include <SFML/Graphics.hpp>
```

```
#include <array>
```

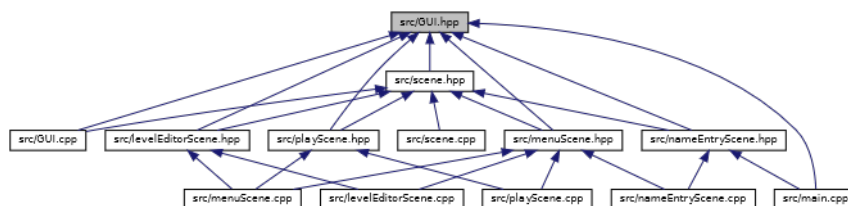
```
#include <iostream>
```

```
#include "image.hpp"
```

Include dependency graph for GUI.hpp:



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



Classes

- struct [Color](#)
Color object *r* is red in range 0 to 1 *g* is green in range 0 to 1 *b* is blue in range 0 to 1.
- class [GUI](#)
GUI Class to create window and run the game.

Enumerations

- enum class [Alignment](#) { [LeftCenter](#) , [Center](#) , [RightCenter](#) }
Enumeration for alignment types.

11.17.1 Enumeration Type Documentation

11.17.1.1 Alignment

```
enum Alignment [strong]
```

Enumeration for alignment types.

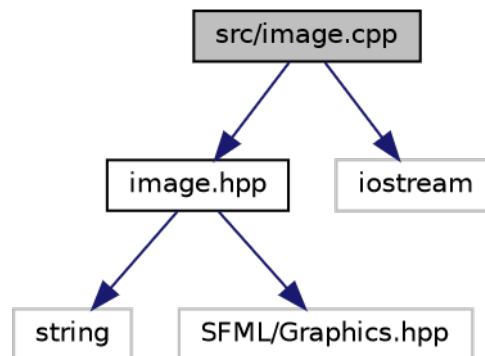
Enumerator

LeftCenter	
Center	
RightCenter	

11.18 src/image.cpp File Reference

```
#include "image.hpp"
#include <iostream>
```

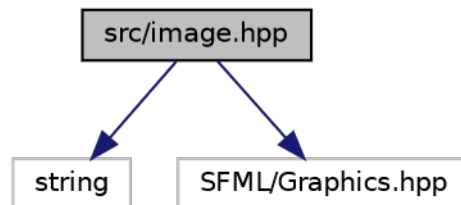
Include dependency graph for image.cpp:



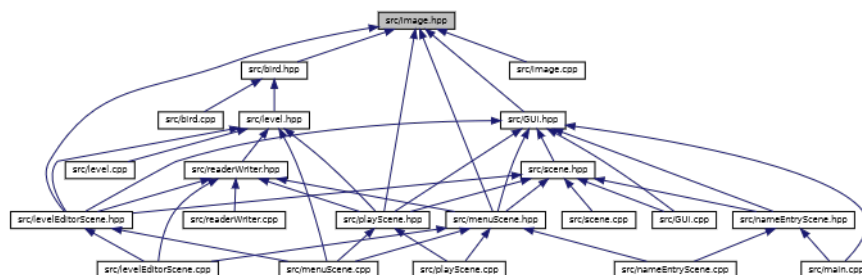
11.19 src/image.hpp File Reference

```
#include <string>
#include <SFML/Graphics.hpp>
```

Include dependency graph for image.hpp:



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



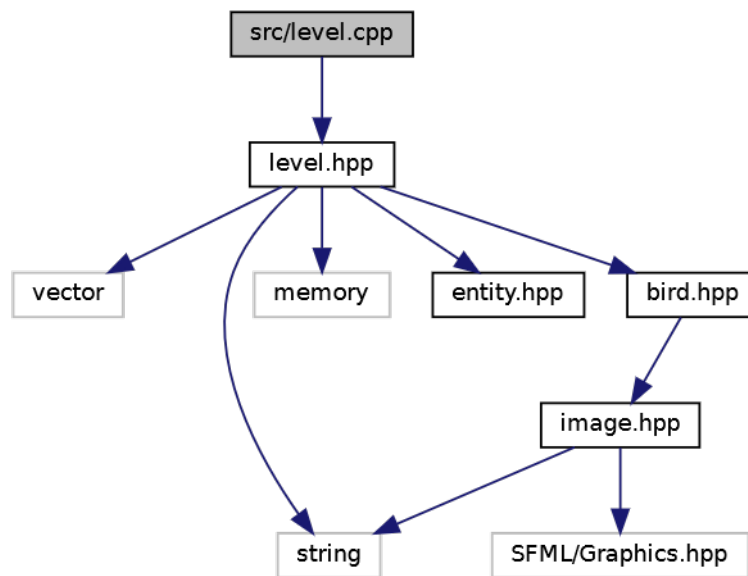
Classes

- class [Image](#)

11.20 src/level.cpp File Reference

```
#include "level.hpp"
```

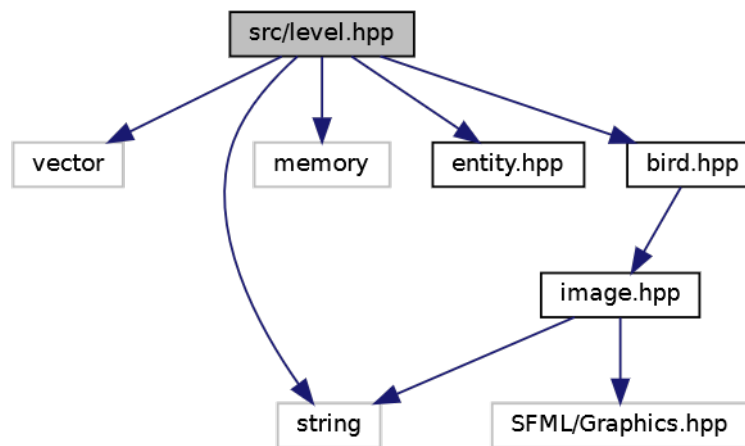
Include dependency graph for level.cpp:



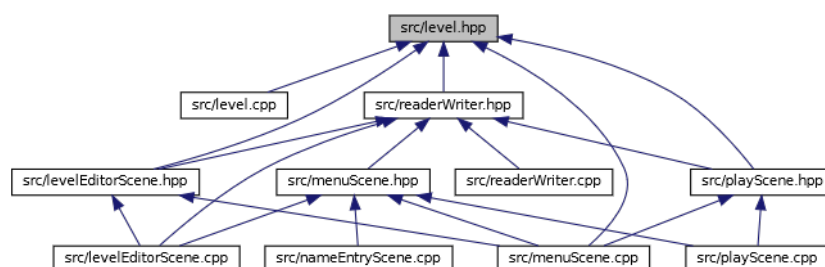
11.21 src/level.hpp File Reference

```
#include <vector>
#include <string>
#include <memory>
#include "entity.hpp"
#include "bird.hpp"
```

Include dependency graph for level.hpp:



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



Classes

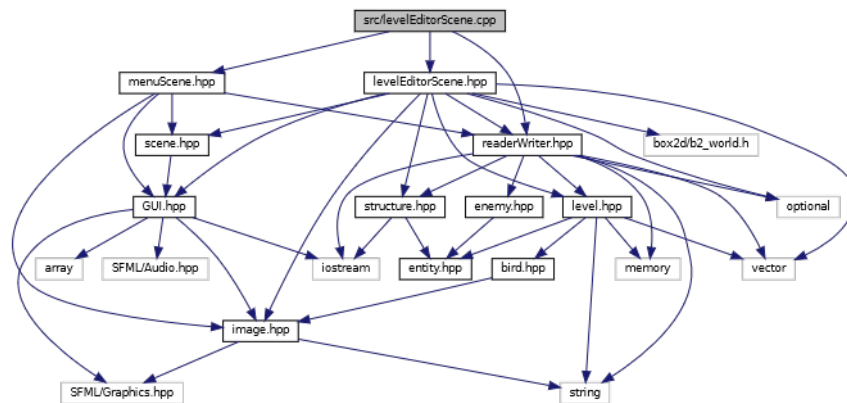
- struct [ScoreBoardEntry](#)
- class [Level](#)

11.22 src/levelEditorScene.cpp File Reference

```
#include "levelEditorScene.hpp"
#include "menuScene.hpp"
```

```
#include "readerWriter.hpp"
```

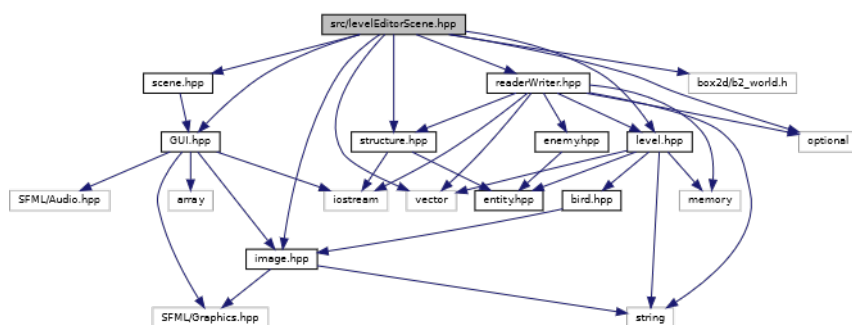
Include dependency graph for levelEditorScene.cpp:



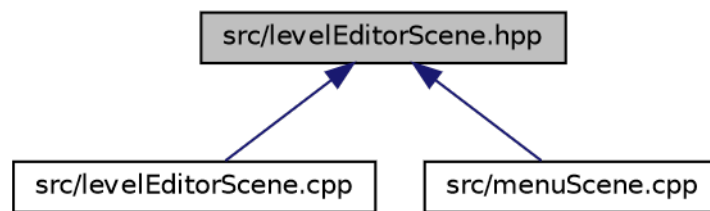
11.23 src/levelEditorScene.hpp File Reference

```
#include "GUI.hpp"
#include "scene.hpp"
#include "image.hpp"
#include "level.hpp"
#include "structure.hpp"
#include "readerWriter.hpp"
#include <box2d/b2_world.h>
#include <optional>
#include <vector>
```

Include dependency graph for levelEditorScene.hpp:



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



Classes

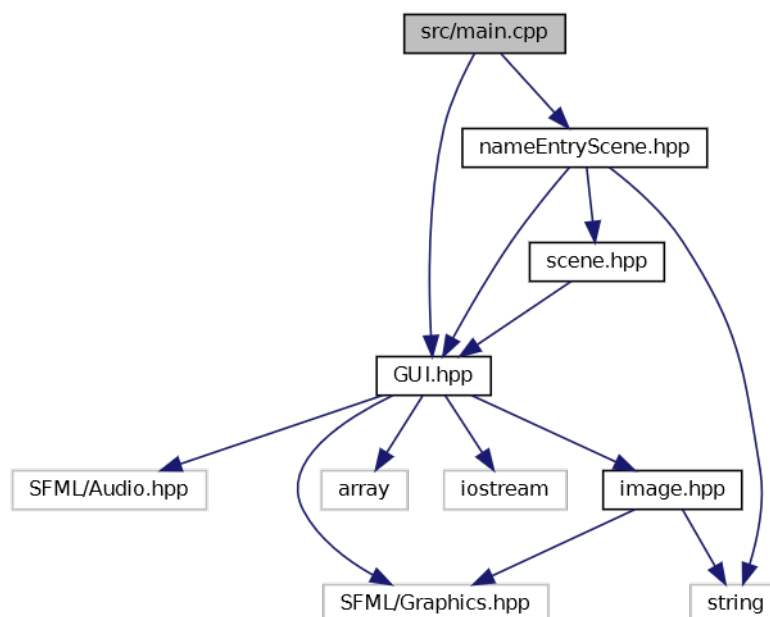
- class [LevelEditorScene](#)

A scene for editing the playable levels in the game.

11.24 src/main.cpp File Reference

```
#include "GUI.hpp"
#include "nameEntryScene.hpp"
```

Include dependency graph for main.cpp:



Functions

- int `main` ()

11.24.1 Function Documentation

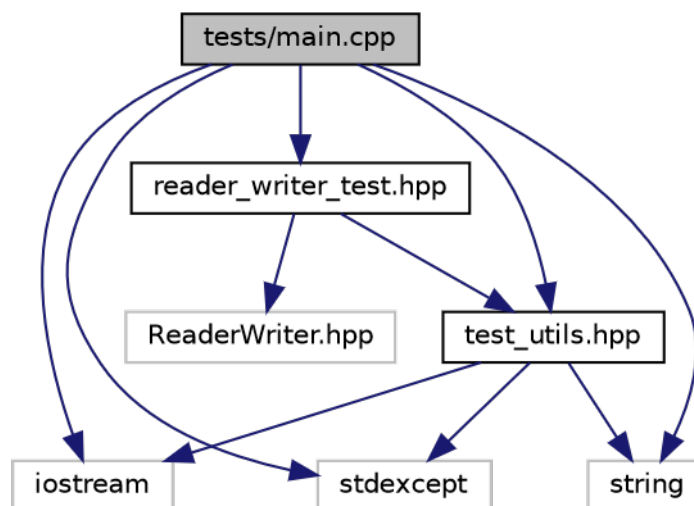
11.24.1.1 `main()`

```
int main ( )
```

11.25 `tests/main.cpp` File Reference

```
#include <iostream>
#include <stdexcept>
#include <string>
#include "test_utils.hpp"
#include "reader_writer_test.hpp"
```

Include dependency graph for `main.cpp`:

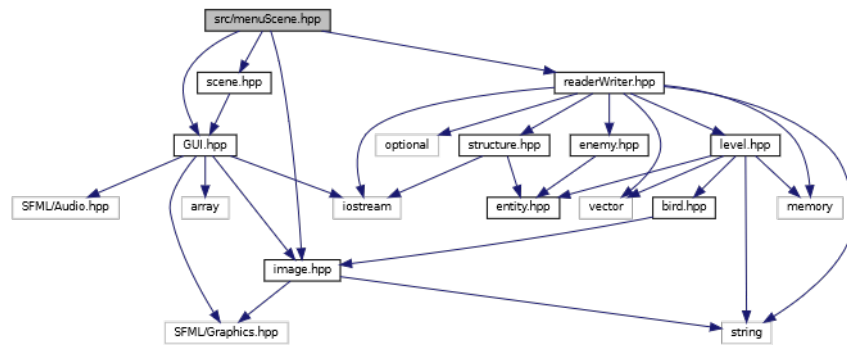


Functions

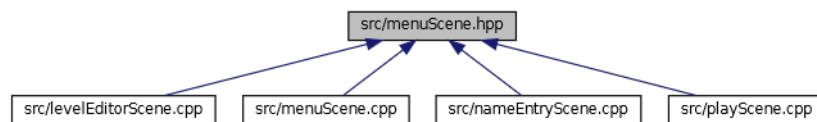
- void `dummytest1` ()
- int `main` ()


```
#include "readerWriter.hpp"
```

Include dependency graph for menuScene.hpp:



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



Classes

- class [MenuScene](#)
Scene to handle the main menu.

Enumerations

- enum class [MenuState](#) {
 [MainMenu](#) , [PlayLevelSelector](#) , [EditorLevelSelector](#) , [LevelSelector](#) ,
 [Help](#) }
The different states a user can be in the main menu.

11.27.1 Enumeration Type Documentation

11.27.1.1 MenuState

```
enum MenuState [strong]
```

The different states a user can be in the main menu.

Enumerator

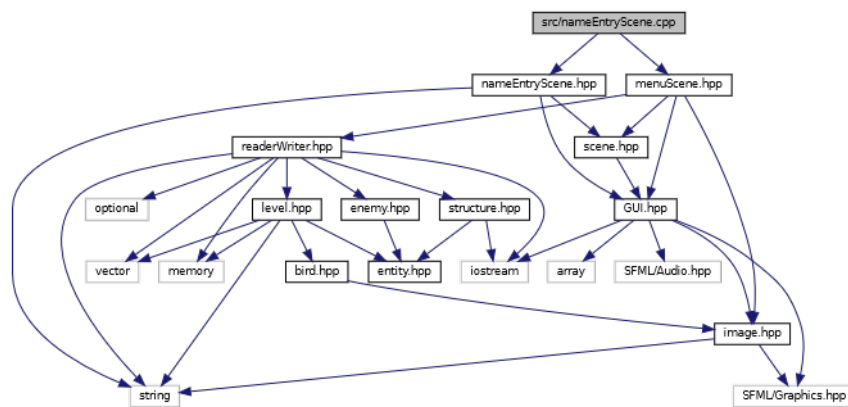
MainMenu	
PlayLevelSelector	
EditorLevelSelector	
LevelSelector	
Help	

11.28 src/nameEntryScene.cpp File Reference

```
#include "nameEntryScene.hpp"
```

```
#include "menuScene.hpp"
```

Include dependency graph for nameEntryScene.cpp:



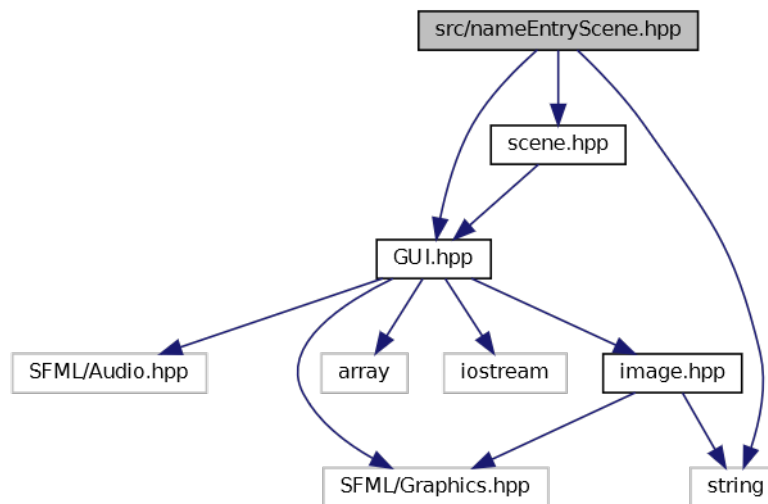
11.29 src/nameEntryScene.hpp File Reference

```
#include "GUI.hpp"
```

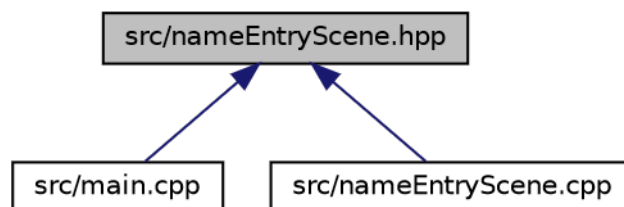
```
#include "scene.hpp"
```

```
#include <string>
```

Include dependency graph for nameEntryScene.hpp:



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



Classes

- class [NameEntryScene](#)

A scene for the user to enter their name from.

11.30 src/playScene.cpp File Reference

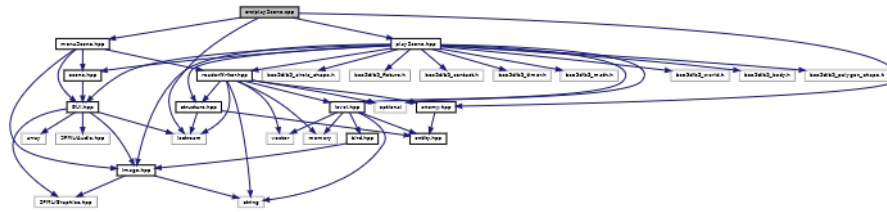
```

#include "playScene.hpp"
#include <iostream>
#include "menuScene.hpp"

```

```
#include "enemy.hpp"
```

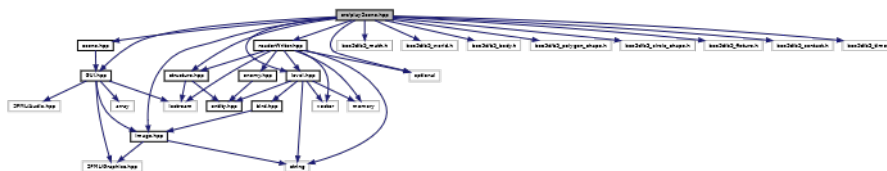
Include dependency graph for playScene.cpp:



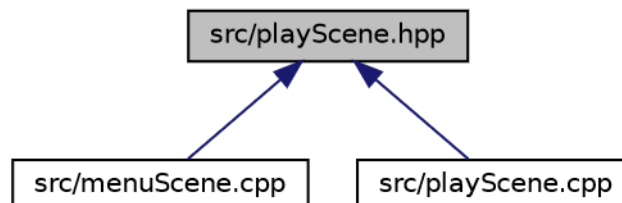
11.31 src/playScene.hpp File Reference

```
#include "GUI.hpp"
#include "scene.hpp"
#include "image.hpp"
#include "level.hpp"
#include "structure.hpp"
#include <optional>
#include <box2d/b2_math.h>
#include <box2d/b2_world.h>
#include <box2d/b2_body.h>
#include <box2d/b2_polygon_shape.h>
#include <box2d/b2_circle_shape.h>
#include <box2d/b2_fixture.h>
#include <box2d/b2_contact.h>
#include <box2d/b2_timer.h>
#include "readerWriter.hpp"
```

Include dependency graph for playScene.hpp:



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



Classes

- struct `userDataStruct`
Physics entities hold a copy of `userDataStruct` they can reference.
- struct `ExplosionData`
Structure to store data for each explosion instance.
- class `PlayScene`
The game play scene to play the levels in the game.

Enumerations

- enum class `gameState` { `won` , `lost` , `playing` }
Different states the game can be in.
- enum class `explosionType` { `cloud` , `fireball` }
Enumeration for different types of explosions in the game.

11.31.1 Enumeration Type Documentation

11.31.1.1 explosionType

```
enum explosionType [strong]
```

Enumeration for different types of explosions in the game.

Enumerator

cloud	
fireball	

11.31.1.2 gameState

```
enum gameState [strong]
```

Different states the game can be in.

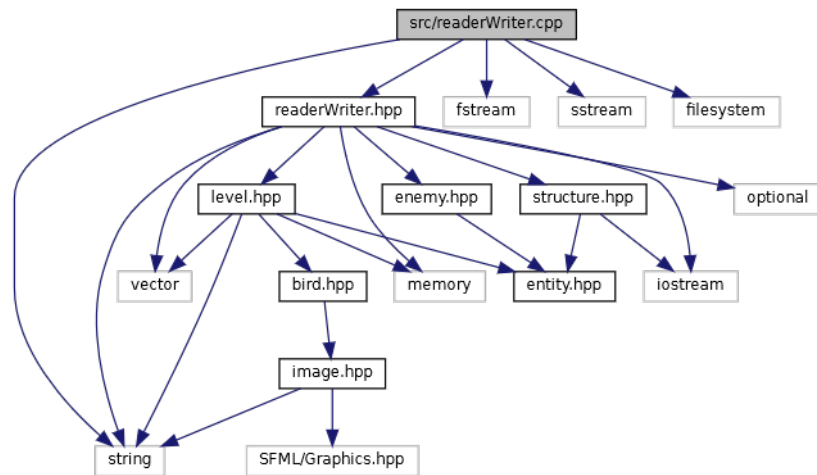
Enumerator

won	
lost	
playing	

11.32 src/readerWriter.cpp File Reference

```
#include "readerWriter.hpp"
#include <fstream>
#include <sstream>
#include <filesystem>
#include <string>
```

Include dependency graph for readerWriter.cpp:



Macros

- `#define RW_LOG(msg)`

Functions

- `std::vector< std::string > split_string (std::string str, char delim=' ')`

11.32.1 Macro Definition Documentation

11.32.1.1 RW_LOG

```
#define RW_LOG(  
    msg )
```

11.32.2 Function Documentation

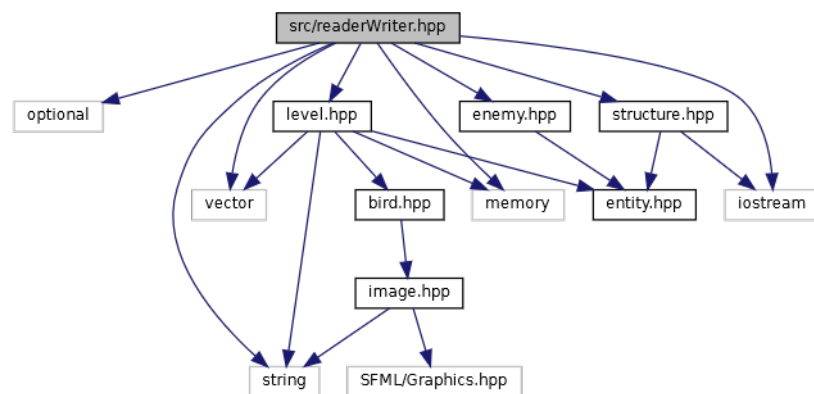
11.32.2.1 split_string()

```
std::vector<std::string> split_string (
    std::string str,
    char delim = ' ' )
```

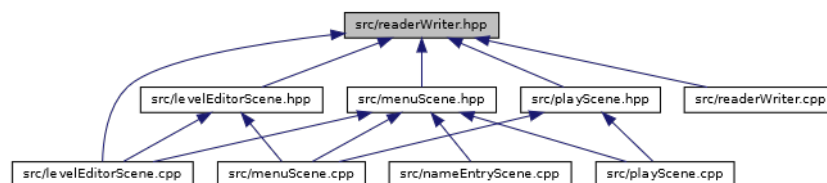
11.33 src/readerWriter.hpp File Reference

```
#include <optional>
#include <string>
#include <vector>
#include <iostream>
#include <memory>
#include "level.hpp"
#include "enemy.hpp"
#include "structure.hpp"
```

Include dependency graph for readerWriter.hpp:



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



Classes

- struct [LevelInfo](#)
Information about levels.
- class [ReaderWriter](#)
A class for loading/reading and saving/writing levels in forms of [Level](#) objects and `.txt` files.

Enumerations

- enum class [Header](#) {
 [levelName](#) , [backgroundPath](#) , [soundtrackPath](#) , [soundFXStart](#) ,
 [soundFXEnd](#) , [EntityStart](#) , [EntityEnd](#) , [BirdsStart](#) ,
 [BirdsEnd](#) , [ScoresStart](#) , [ScoresEnd](#) , [unknown](#) }

Different types of Headers, Allows us to use switch case with headers.

11.33.1 Enumeration Type Documentation

11.33.1.1 Header

```
enum Header [strong]
```

Different types of Headers, Allows us to use switch case with headers.

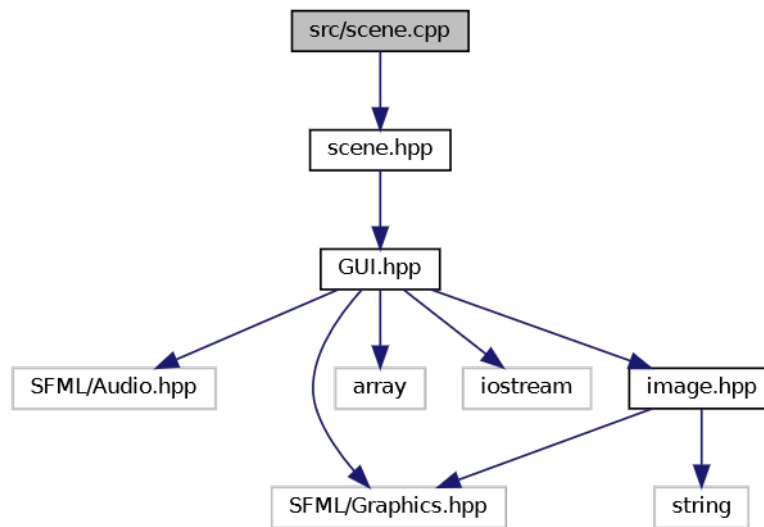
Enumerator

levelName	
backgroundPath	
soundtrackPath	
soundFXStart	
soundFXEnd	
EntityStart	
EntityEnd	
BirdsStart	
BirdsEnd	
ScoresStart	
ScoresEnd	
unknown	

11.34 src/scene.cpp File Reference

```
#include "scene.hpp"
```

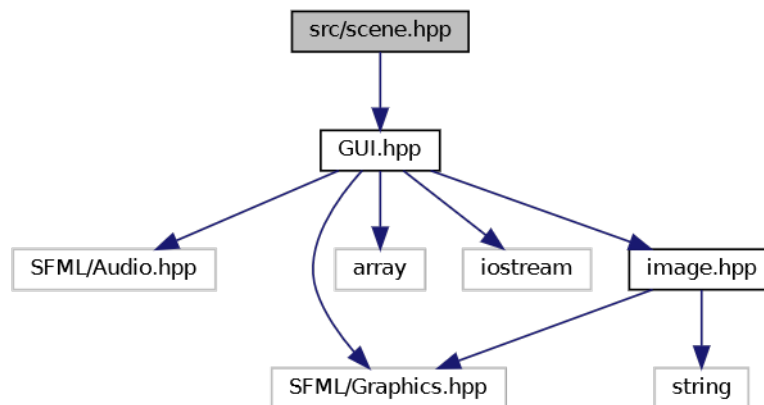
Include dependency graph for scene.cpp:



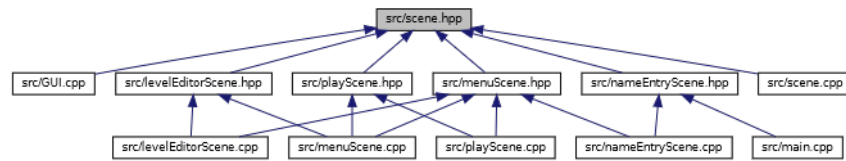
11.35 src/scene.hpp File Reference

```
#include "GUI.hpp"
```

Include dependency graph for scene.hpp:



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



Classes

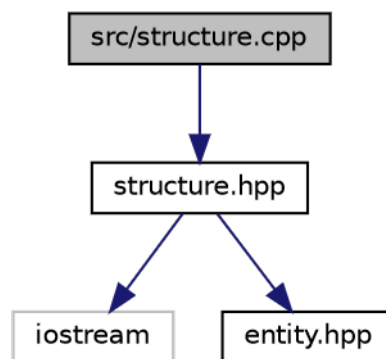
- class [Scene](#)

Base [Scene](#) class to represent different states the game can be in. Ex. Play state can be implemented with [PlayScene](#). Menu can be implemented with [MenuScene](#) etc.

11.36 src/structure.cpp File Reference

```
#include "structure.hpp"
```

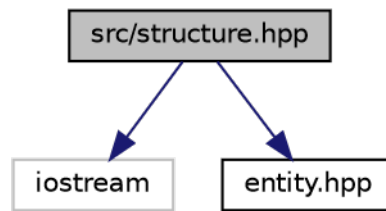
Include dependency graph for structure.cpp:



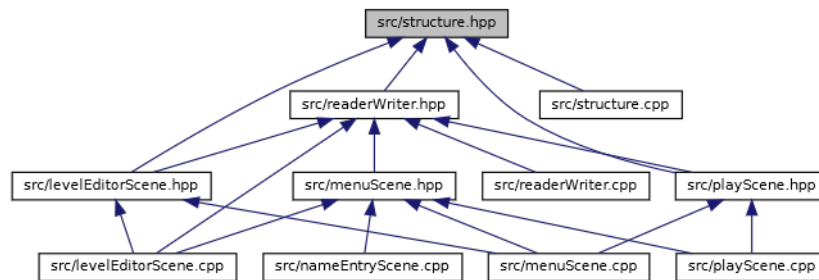
11.37 src/structure.hpp File Reference

```
#include <iostream>
#include "entity.hpp"
```

Include dependency graph for structure.hpp:



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



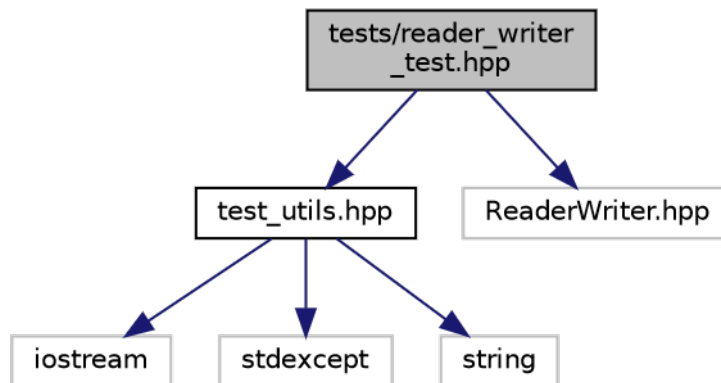
Classes

- class [Structure](#)
Structure type to represent walls in the game.

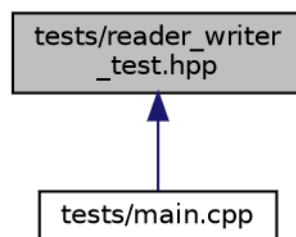
11.38 tests/reader_writer_test.hpp File Reference

```
#include "test_utils.hpp"
#include "ReaderWriter.hpp"
```

Include dependency graph for reader_writer_test.hpp:



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



Functions

- void [test_ReaderWriter_read_file](#) ()
- void [test_ReaderWriter_write_file](#) ()
- void [tests_reader_writer](#) ()

11.38.1 Function Documentation

11.38.1.1 test_ReaderWriter_read_file()

```
void test_ReaderWriter_read_file ( ) [inline]
```

11.38.1.2 test_ReaderWriter_write_file()

```
void test_ReaderWriter_write_file ( ) [inline]
```

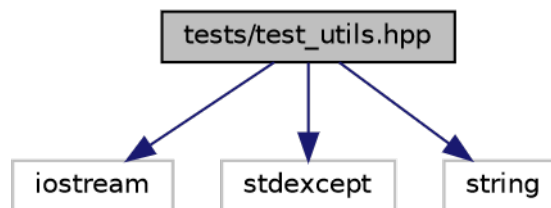
11.38.1.3 tests_reader_writer()

```
void tests_reader_writer ( ) [inline]
```

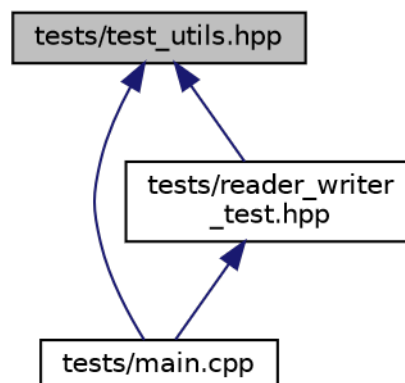
11.39 tests/test_utils.hpp File Reference

```
#include <iostream>
#include <stdexcept>
#include <string>
```

Include dependency graph for test_utils.hpp:



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



Classes

- class [TestAssertionException](#)

Functions

- void [verify](#) (bool condition, const std::string &message)

11.39.1 Function Documentation

11.39.1.1 [verify\(\)](#)

```
void verify (  
    bool condition,  
    const std::string & message ) [inline]
```


Index

- `__has_include`
 - `CMakeCCompilerId.c`, [81](#)
 - `CMakeCXXCompilerId.cpp`, [85](#)
 - `~Bird`
 - `Bird`, [22](#)
 - `~GUI`
 - `GUI`, [35](#)
 - `~LevelEditorScene`
 - `LevelEditorScene`, [47](#)
 - `~MenuScene`
 - `MenuScene`, [52](#)
 - `~NameEntryScene`
 - `NameEntryScene`, [55](#)
 - `~PlayScene`
 - `PlayScene`, [61](#)
 - `~Scene`
 - `Scene`, [67](#)
- `add_score`
 - `Level`, [42](#)
- `Alignment`
 - `GUI.hpp`, [94](#)
- `ARCHITECTURE_ID`
 - `CMakeCCompilerId.c`, [82](#)
 - `CMakeCXXCompilerId.cpp`, [85](#)
- `b`
 - `Color`, [24](#)
- `backgroundPath`
 - `readerWriter.hpp`, [109](#)
- `Bird`, [21](#)
 - `~Bird`, [22](#)
 - `Bird`, [22](#)
 - `get_bird_type`, [23](#)
 - `getTime`, [23](#)
 - `resetTime`, [23](#)
 - `spawnTime`, [23](#)
- `bird`
 - `entity.hpp`, [92](#)
 - `userDataStruct`, [79](#)
- `bird.hpp`
 - `birdType`, [90](#)
 - `normal`, [90](#)
 - `special1`, [90](#)
 - `special2`, [90](#)
- `BirdsEnd`
 - `readerWriter.hpp`, [109](#)
- `BirdsStart`
 - `readerWriter.hpp`, [109](#)
- `birdType`

- `bird.hpp`, [90](#)
- `bodyType`
 - `entity.hpp`, [92](#)
- `build/CMakeFiles/3.22.1/CompilerIdC/CMakeCCompilerId.c`, [81](#)
- `build/CMakeFiles/3.22.1/CompilerIdCXX/CMakeCXXCompilerId.cpp`, [84](#)
- `button_released`
 - `GUI`, [35](#)
- `button_state`
 - `GUI`, [36](#)
- `C_VERSION`
 - `CMakeCCompilerId.c`, [82](#)
- `Center`
 - `GUI.hpp`, [94](#)
- `close`
 - `GUI`, [36](#)
- `cloud`
 - `playScene.hpp`, [106](#)
- `CMakeCCompilerId.c`
 - `__has_include`, [81](#)
 - `ARCHITECTURE_ID`, [82](#)
 - `C_VERSION`, [82](#)
 - `COMPILER_ID`, [82](#)
 - `DEC`, [82](#)
 - `HEX`, [82](#)
 - `info_arch`, [83](#)
 - `info_compiler`, [83](#)
 - `info_language_extensions_default`, [83](#)
 - `info_language_standard_default`, [84](#)
 - `info_platform`, [84](#)
 - `main`, [83](#)
 - `PLATFORM_ID`, [82](#)
 - `STRINGIFY`, [83](#)
 - `STRINGIFY_HELPER`, [83](#)
- `CMakeCXXCompilerId.cpp`
 - `__has_include`, [85](#)
 - `ARCHITECTURE_ID`, [85](#)
 - `COMPILER_ID`, [85](#)
 - `CXX_STD`, [85](#)
 - `DEC`, [85](#)
 - `HEX`, [85](#)
 - `info_arch`, [86](#)
 - `info_compiler`, [87](#)
 - `info_language_extensions_default`, [87](#)
 - `info_language_standard_default`, [87](#)
 - `info_platform`, [87](#)
 - `main`, [86](#)
 - `PLATFORM_ID`, [86](#)

- STRINGIFY, [86](#)
 - STRINGIFY_HELPER, [86](#)
- Color, [24](#)
 - b, [24](#)
 - g, [25](#)
 - r, [25](#)
- COMPILER_ID
 - CMakeCCompilerId.c, [82](#)
 - CMakeCXXCompilerId.cpp, [85](#)
- contains
 - Enemy, [27](#)
 - Entity, [30](#)
 - Structure, [75](#)
- cursor_position
 - GUI, [36](#)
- CXX_STD
 - CMakeCXXCompilerId.cpp, [85](#)
- DEC
 - CMakeCCompilerId.c, [82](#)
 - CMakeCXXCompilerId.cpp, [85](#)
- doc/readme.md, [88](#)
- draw_button
 - GUI, [36](#)
- draw_rect
 - GUI, [37](#)
- draw_sprite
 - GUI, [37](#)
- draw_text
 - GUI, [38](#)
- dummytest1
 - main.cpp, [101](#)
- EditorLevelSelector
 - menuScene.hpp, [103](#)
- Enemy, [25](#)
 - contains, [27](#)
 - Enemy, [26](#)
 - get_type, [27](#)
- enemy
 - entity.hpp, [92](#)
- Entity, [28](#)
 - contains, [30](#)
 - Entity, [29](#)
 - get_health_points, [30](#)
 - get_rotation, [31](#)
 - get_type, [31](#)
 - get_x, [31](#)
 - get_y, [31](#)
 - is_destructible, [32](#)
 - is_movable, [32](#)
- entity
 - userDataStruct, [79](#)
- entity.hpp
 - bird, [92](#)
 - bodyType, [92](#)
 - enemy, [92](#)
 - ground, [92](#)
 - structure, [92](#)
- EntityEnd
 - readerWriter.hpp, [109](#)
- EntityStart
 - readerWriter.hpp, [109](#)
- ExplosionData, [33](#)
 - position, [33](#)
 - time, [33](#)
 - type, [33](#)
- explosionType
 - playScene.hpp, [106](#)
- fireball
 - playScene.hpp, [106](#)
- g
 - Color, [25](#)
- gameState
 - playScene.hpp, [106](#)
- get_aspect_ratio
 - GUI, [38](#)
- get_background
 - Level, [43](#)
- get_bird_type
 - Bird, [23](#)
 - NormalBird, [58](#)
 - SpecialBird1, [71](#)
 - SpecialBird2, [73](#)
- get_birds
 - Level, [43](#)
- get_entities
 - Level, [43](#)
- get_health_points
 - Entity, [30](#)
- get_height
 - Structure, [75](#)
- get_levels
 - ReaderWriter, [64](#)
- get_name
 - Level, [43](#)
- get_next_level
 - ReaderWriter, [64](#)
- get_rotation
 - Entity, [31](#)
- get_save_name
 - Level, [44](#)
- get_scores
 - Level, [44](#)
- get_type
 - Enemy, [27](#)
 - Entity, [31](#)
 - Structure, [76](#)
- get_width
 - Structure, [76](#)
- get_x
 - Entity, [31](#)
- get_y
 - Entity, [31](#)
- getTime
 - Bird, [23](#)

- ground
 - entity.hpp, 92
- GUI, 34
 - ~GUI, 35
 - button_released, 35
 - button_state, 36
 - close, 36
 - cursor_position, 36
 - draw_button, 36
 - draw_rect, 37
 - draw_sprite, 37
 - draw_text, 38
 - get_aspect_ratio, 38
 - GUI, 35
 - Image, 41
 - key_state, 38
 - play_sound, 39
 - run, 39
 - scroll_delta, 39
 - set_scene, 39
 - set_viewport, 40
- GUI.hpp
 - Alignment, 94
 - Center, 94
 - LeftCenter, 94
 - RightCenter, 94
- gui_
 - Scene, 68
- Header
 - readerWriter.hpp, 109
- Help
 - menuScene.hpp, 103
- HEX
 - CMakeCCompilerId.c, 82
 - CMakeCXXCompilerId.cpp, 85
- hp
 - userDataStruct, 79
- Image, 40
 - GUI, 41
 - Image, 41
- image
 - userDataStruct, 79
- info_arch
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 86
- info_compiler
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 87
- info_language_extensions_default
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 87
- info_language_standard_default
 - CMakeCCompilerId.c, 84
 - CMakeCXXCompilerId.cpp, 87
- info_platform
 - CMakeCCompilerId.c, 84
 - CMakeCXXCompilerId.cpp, 87
- is_destructible
 - Entity, 32
- is_movable
 - Entity, 32
- key_state
 - GUI, 38
- LeftCenter
 - GUI.hpp, 94
- Level, 41
 - add_score, 42
 - get_background, 43
 - get_birds, 43
 - get_entities, 43
 - get_name, 43
 - get_save_name, 44
 - get_scores, 44
 - Level, 42
- LevelEditorScene, 45
 - ~LevelEditorScene, 47
 - LevelEditorScene, 47
 - render, 47
 - update, 47
- LevelInfo, 49
 - name, 49
 - path, 50
- levelName
 - readerWriter.hpp, 109
- LevelSelector
 - menuScene.hpp, 103
- lost
 - playScene.hpp, 106
- main
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 86
 - main.cpp, 100, 101
- main.cpp
 - dummysTest1, 101
 - main, 100, 101
- MainMenu
 - menuScene.hpp, 103
- Meeting-notes.md, 88
- MenuScene, 50
 - ~MenuScene, 52
 - MenuScene, 52
 - render, 52
 - update, 52
- menuScene.hpp
 - EditorLevelSelector, 103
 - Help, 103
 - LevelSelector, 103
 - MainMenu, 103
 - MenuState, 102
 - PlayLevelSelector, 103
- MenuState
 - menuScene.hpp, 102

- name
 - LevelInfo, 49
 - ScoreBoardEntry, 69
- NameEntryScene, 53
 - ~NameEntryScene, 55
 - NameEntryScene, 55
 - on_input, 55
 - render, 56
 - update, 56
- normal
 - bird.hpp, 90
- NormalBird, 56
 - get_bird_type, 58
 - NormalBird, 58
- on_input
 - NameEntryScene, 55
 - Scene, 67
- path
 - LevelInfo, 50
- plan/readme.md, 88
- PLATFORM_ID
 - CMakeCCompilerId.c, 82
 - CMakeCXXCompilerId.cpp, 86
- play_sound
 - GUI, 39
- playing
 - playScene.hpp, 106
- PlayLevelSelector
 - menuScene.hpp, 103
- PlayScene, 59
 - ~PlayScene, 61
 - PlayScene, 61
 - render, 61
 - update, 61
- playScene.hpp
 - cloud, 106
 - explosionType, 106
 - fireball, 106
 - gameState, 106
 - lost, 106
 - playing, 106
 - won, 106
- position
 - ExplosionData, 33
- r
 - Color, 25
- read_file
 - ReaderWriter, 64
- reader_writer_test.hpp
 - test_ReaderWriter_read_file, 113
 - test_ReaderWriter_write_file, 113
 - tests_reader_writer, 114
- ReaderWriter, 63
 - get_levels, 64
 - get_next_level, 64
 - read_file, 64
 - ReaderWriter, 64
 - write_file, 64
- readerWriter.cpp
 - RW_LOG, 107
 - split_string, 107
- readerWriter.hpp
 - backgroundPath, 109
 - BirdsEnd, 109
 - BirdsStart, 109
 - EntityEnd, 109
 - EntityStart, 109
 - Header, 109
 - levelName, 109
 - ScoresEnd, 109
 - ScoresStart, 109
 - soundFXEnd, 109
 - soundFXStart, 109
 - soundtrackPath, 109
 - unknown, 109
- README.md, 88
- render
 - LevelEditorScene, 47
 - MenuScene, 52
 - NameEntryScene, 56
 - PlayScene, 61
 - Scene, 67
- res/README.md, 88
- resetTime
 - Bird, 23
- RightCenter
 - GUI.hpp, 94
- run
 - GUI, 39
- RW_LOG
 - readerWriter.cpp, 107
- SavefileFormat.md, 88
- Scene, 65
 - ~Scene, 67
 - gui_, 68
 - on_input, 67
 - render, 67
 - Scene, 67
 - update, 68
- score
 - ScoreBoardEntry, 69
- ScoreBoardEntry, 68
 - name, 69
 - score, 69
- ScoresEnd
 - readerWriter.hpp, 109
- ScoresStart
 - readerWriter.hpp, 109
- scroll_delta
 - GUI, 39
- set_scene
 - GUI, 39
- set_viewport
 - GUI, 40

- soundFXEnd
 - readerWriter.hpp, 109
- soundFXStart
 - readerWriter.hpp, 109
- soundtrackPath
 - readerWriter.hpp, 109
- spawnTime
 - Bird, 23
- special1
 - bird.hpp, 90
- special2
 - bird.hpp, 90
- SpecialBird1, 69
 - get_bird_type, 71
 - SpecialBird1, 70
- SpecialBird2, 71
 - get_bird_type, 73
 - SpecialBird2, 72
- split_string
 - readerWriter.cpp, 107
- src/bird.cpp, 88
- src/bird.hpp, 89
- src/enemy.cpp, 90
- src/enemy.hpp, 90
- src/entity.cpp, 91
- src/entity.hpp, 92
- src/GUI.cpp, 92
- src/GUI.hpp, 93
- src/image.cpp, 94
- src/image.hpp, 95
- src/level.cpp, 96
- src/level.hpp, 96
- src/levelEditorScene.cpp, 97
- src/levelEditorScene.hpp, 98
- src/main.cpp, 99
- src/menuScene.cpp, 101
- src/menuScene.hpp, 101
- src/nameEntryScene.cpp, 103
- src/nameEntryScene.hpp, 103
- src/playScene.cpp, 104
- src/playScene.hpp, 105
- src/readerWriter.cpp, 107
- src/readerWriter.hpp, 108
- src/readme.md, 88
- src/scene.cpp, 109
- src/scene.hpp, 110
- src/structure.cpp, 111
- src/structure.hpp, 111
- STRINGIFY
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 86
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, 83
 - CMakeCXXCompilerId.cpp, 86
- Structure, 73
 - contains, 75
 - get_height, 75
 - get_type, 76
 - get_width, 76
 - Structure, 75
- structure
 - entity.hpp, 92
- test_ReaderWriter_read_file
 - reader_writer_test.hpp, 113
- test_ReaderWriter_write_file
 - reader_writer_test.hpp, 113
- test_utils.hpp
 - verify, 115
- TestAssertionException, 77
 - TestAssertionException, 78
 - what, 78
- tests/main.cpp, 100
- tests/reader_writer_test.hpp, 112
- tests/test_utils.hpp, 114
- tests_reader_writer
 - reader_writer_test.hpp, 114
- time
 - ExplosionData, 33
- type
 - ExplosionData, 33
 - userDataStruct, 79
- unknown
 - readerWriter.hpp, 109
- update
 - LevelEditorScene, 47
 - MenuScene, 52
 - NameEntryScene, 56
 - PlayScene, 61
 - Scene, 68
- userDataStruct, 78
 - bird, 79
 - entity, 79
 - hp, 79
 - image, 79
 - type, 79
- verify
 - test_utils.hpp, 115
- what
 - TestAssertionException, 78
- won
 - playScene.hpp, 106
- write_file
 - ReaderWriter, 64