BreakoutGame

1.0.0

Generated by Doxygen 1.8.17

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 Ball Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 Ball() [1/2] . . . 8

4.1.2.2 Ball() [2/2] . . . 8

4.1.3 Member Function Documentation . . . 9

4.1.3.1 CollideWithBrick() . . . 9

4.1.3.2 CollideWithPaddle() . . . 9

4.1.3.3 CollideWithWall() . . . 10

4.1.3.4 Draw() . . . 10

4.1.3.5 Update() . . . 11

4.1.4 Member Data Documentation . . . 11

4.1.4.1 position . . . 11

4.1.4.2 rect . . . 11

4.1.4.3 velocity . . . 12

4.2 BreakoutGame Class Reference . . . 12

4.2.1 Detailed Description . . . 14

4.2.2 Constructor & Destructor Documentation . . . 15

4.2.2.1 BreakoutGame() . . . 15

4.2.2.2 ~BreakoutGame() . . . 16

4.2.3 Member Function Documentation . . . 16

4.2.3.1 CheckBrickCollision() . . . 16

4.2.3.2 CheckPaddleCollision() . . . 17

4.2.3.3 CheckWallCollision() . . . 17

4.2.3.4 getSDLRenderer() . . . 18

4.2.3.5 getSDLWindow() . . . 18

4.2.3.6 initBall() . . . 18

4.2.3.7 initGameObjects() . . . 19

4.2.3.8 initPaddle() . . . 19

4.2.3.9 initSDLSystems() . . . 20

4.2.3.10 loadLanguages() . . . 20

4.2.3.11 loadLevels() . . . 21

4.2.3.12 loadResources() . . . 22

4.2.3.13 loop() . . . 23
4.2.3.14 render() . . . 24
4.2.3.15 resetBricks() . . . 24
4.2.3.16 update() . . . 25
4.2.3.17 updateAllTexts() . . . 26
4.2.4 Member Data Documentation . . . 26
4.2.4.1 backgroundMusic . . . 26
4.2.4.2 ball . . . 26
4.2.4.3 brickHitSound . . . 27
4.2.4.4 bricks . . . 27
4.2.4.5 bricksGenerator . . . 27
4.2.4.6 buttons . . . 27
4.2.4.7 configs . . . 27
4.2.4.8 contentFont_ . . . 27
4.2.4.9 gameState . . . 28
4.2.4.10 gRenderer . . . 28
4.2.4.11 gWindow . . . 28
4.2.4.12 languageSelector . . . 28
4.2.4.13 level . . . 28
4.2.4.14 levelNum . . . 28
4.2.4.15 levelText . . . 29
4.2.4.16 livesNum . . . 29
4.2.4.17 livesText . . . 29
4.2.4.18 loseLifeSound . . . 29
4.2.4.19 loseSound . . . 29
4.2.4.20 maxLevel . . . 29
4.2.4.21 menuFont_ . . . 30
4.2.4.22 notificationText . . . 30
4.2.4.23 paddle . . . 30
4.2.4.24 paddleHitSound . . . 30
4.2.4.25 player . . . 30
4.2.4.26 restBricks . . . 30
4.2.4.27 scoreNum . . . 31
4.2.4.28 scoreText . . . 31
4.2.4.29 screenHeight . . . 31
4.2.4.30 screenWidth . . . 31
4.2.4.31 wallHitSound . . . 31
4.2.4.32 wallLeft . . . 31
4.2.4.33 wallRight . . . 32
4.2.4.34 winSound . . . 32
4.3 Brick Class Reference . . . 32
4.3.1 Detailed Description . . . 33

4.3.2 Constructor & Destructor Documentation . . . 34
4.3.2.1 Brick() . . . 34
4.3.3 Member Function Documentation . . . 34
4.3.3.1 Draw() . . . 34
4.3.3.2 getScore() . . . 34
4.3.3.3 isActive() . . . 35
4.3.3.4 setActive() . . . 35
4.3.3.5 setScore() . . . 35
4.3.4 Member Data Documentation . . . 35
4.3.4.1 active . . . 36
4.3.4.2 score . . . 36
4.4 BricksGenerator Class Reference . . . 36
4.4.1 Detailed Description . . . 36
4.4.2 Member Function Documentation . . . 36
4.4.2.1 getLevelBricks() . . . 36
4.4.2.2 getMaxLevel() . . . 37
4.4.2.3 loadOneLevelData() . . . 37
4.4.3 Member Data Documentation . . . 38
4.4.3.1 allLevels . . . 38
4.5 ConfigUtil Class Reference . . . 38
4.5.1 Detailed Description . . . 39
4.5.2 Member Function Documentation . . . 39
4.5.2.1 loadAllVariables() . . . 39
4.5.2.2 loadConfig() . . . 39
4.6 Contact Struct Reference . . . 40
4.6.1 Detailed Description . . . 41
4.6.2 Member Data Documentation . . . 41
4.6.2.1 penetration . . . 41
4.6.2.2 type . . . 41
4.7 Language Struct Reference . . . 41
4.7.1 Detailed Description . . . 41
4.7.2 Member Data Documentation . . . 41
4.7.2.1 data . . . 42
4.8 LanguageSelector Class Reference . . . 42
4.8.1 Detailed Description . . . 42
4.8.2 Member Function Documentation . . . 42
4.8.2.1 getContent() . . . 43
4.8.2.2 loadOneLanguageContent() . . . 43
4.8.2.3 useLanguage() . . . 44
4.8.3 Member Data Documentation . . . 45
4.8.3.1 currentLanguage . . . 45
4.8.3.2 languages . . . 45

4.9 LevelData Struct Reference . . . 45
4.9.1 Detailed Description . . . 46
4.9.2 Member Data Documentation . . . 46
4.9.2.1 data . . . 46
4.10 LTimer Class Reference . . . 46
4.10.1 Detailed Description . . . 47
4.10.2 Constructor & Destructor Documentation . . . 47
4.10.2.1 LTimer() . . . 47
4.10.3 Member Function Documentation . . . 47
4.10.3.1 getTicks() . . . 47
4.10.3.2 isPaused() . . . 48
4.10.3.3 isStarted() . . . 48
4.10.3.4 pause() . . . 48
4.10.3.5 start() . . . 49
4.10.3.6 stop() . . . 49
4.10.3.7 unpause() . . . 49
4.10.4 Member Data Documentation . . . 49
4.10.4.1 mPaused . . . 50
4.10.4.2 mPausedTicks . . . 50
4.10.4.3 mStarted . . . 50
4.10.4.4 mStartTicks . . . 50
4.11 Matrix3D Struct Reference . . . 50
4.11.1 Detailed Description . . . 51
4.11.2 Constructor & Destructor Documentation . . . 51
4.11.2.1 Matrix3D() [1/3] . . . 51
4.11.2.2 Matrix3D() [2/3] . . . 52
4.11.2.3 Matrix3D() [3/3] . . . 52
4.11.3 Member Function Documentation . . . 52
4.11.3.1 column() . . . 52
4.11.3.2 operator"!=() . . . 53
4.11.3.3 operator()() [1/2] . . . 53
4.11.3.4 operator()() [2/2] . . . 53
4.11.3.5 operator==() . . . 53
4.11.3.6 operator[]() [1/2] . . . 54
4.11.3.7 operator[]() [2/2] . . . 54
4.11.4 Member Data Documentation . . . 54
4.11.4.1 n . . . 54
4.12 Paddle Class Reference . . . 54
4.12.1 Detailed Description . . . 55
4.12.2 Constructor & Destructor Documentation . . . 55
4.12.2.1 Paddle() [1/2] . . . 56
4.12.2.2 Paddle() [2/2] . . . 56

4.12.3 Member Function Documentation . . . 56
4.12.3.1 Draw() . . . 56
4.12.3.2 Update() . . . 57
4.12.4 Member Data Documentation . . . 57
4.12.4.1 velocity . . . 57
4.13 Player Class Reference . . . 57
4.13.1 Detailed Description . . . 58
4.13.2 Constructor & Destructor Documentation . . . 58
4.13.2.1 Player() [1/2] . . . 58
4.13.2.2 Player() [2/2] . . . 58
4.13.3 Member Function Documentation . . . 59
4.13.3.1 addScore() . . . 59
4.13.3.2 getLives() . . . 59
4.13.3.3 getScore() . . . 60
4.13.3.4 loseLife() . . . 60
4.13.3.5 setLives() . . . 60
4.13.3.6 setScore() . . . 61
4.13.4 Member Data Documentation . . . 61
4.13.4.1 lives . . . 61
4.13.4.2 score . . . 61
4.14 Rectangle Class Reference . . . 62
4.14.1 Detailed Description . . . 63
4.14.2 Constructor & Destructor Documentation . . . 63
4.14.2.1 Rectangle() [1/2] . . . 63
4.14.2.2 Rectangle() [2/2] . . . 63
4.14.3 Member Function Documentation . . . 63
4.14.3.1 Draw() . . . 63
4.14.4 Member Data Documentation . . . 64
4.14.4.1 position . . . 64
4.14.4.2 rect . . . 64
4.15 ResourceManager Class Reference . . . 64
4.15.1 Detailed Description . . . 66
4.15.2 Constructor & Destructor Documentation . . . 66
4.15.2.1 ResourceManager() [1/2] . . . 66
4.15.2.2 ResourceManager() [2/2] . . . 66
4.15.2.3 ~ResourceManager() . . . 66
4.15.3 Member Function Documentation . . . 67
4.15.3.1 AddResource() [1/3] . . . 67
4.15.3.2 AddResource() [2/3] . . . 67
4.15.3.3 AddResource() [3/3] . . . 68
4.15.3.4 getInstance() . . . 68
4.15.3.5 init() . . . 69

4.15.3.6 LoadResource() . . . 69
4.15.3.7 name() . . . 70
4.15.3.8 operator=() . . . 70
4.15.3.9 RemoveAllResource() . . . 71
4.15.3.10 RemoveResource() . . . 71
4.15.3.11 size() . . . 71
4.15.3.12 startManager() . . . 72
4.15.3.13 stopManager() . . . 72
4.15.4 Member Data Documentation . . . 73
4.15.4.1 cfgFilePath_ . . . 73
4.15.4.2 fileMap_ . . . 73
4.15.4.3 name_ . . . 73
4.15.4.4 resMap_ . . . 74
4.15.4.5 useConfig_ . . . 74
4.16 Text Class Reference . . . 74
4.16.1 Detailed Description . . . 75
4.16.2 Constructor & Destructor Documentation . . . 75
4.16.2.1 Text() [1/2] . . . 75
4.16.2.2 Text() [2/2] . . . 75
4.16.3 Member Function Documentation . . . 76
4.16.3.1 Draw() . . . 76
4.16.3.2 getCenterPosition() . . . 76
4.16.3.3 getHeight() . . . 77
4.16.3.4 getWidth() . . . 77
4.16.3.5 SetCenterPosition() . . . 77
4.16.3.6 SetColor() . . . 78
4.16.3.7 setKeepCentered() . . . 78
4.16.3.8 SetPosition() [1/2] . . . 79
4.16.3.9 SetPosition() [2/2] . . . 79
4.16.3.10 SetText() . . . 79
4.16.4 Member Data Documentation . . . 80
4.16.4.1 color . . . 80
4.16.4.2 font . . . 80
4.16.4.3 keepCentered . . . 81
4.16.4.4 lastText . . . 81
4.16.4.5 rect . . . 81
4.16.4.6 renderer . . . 81
4.16.4.7 surface . . . 81
4.16.4.8 texture . . . 81
4.17 Vector2D Struct Reference . . . 82
4.17.1 Detailed Description . . . 82
4.17.2 Constructor & Destructor Documentation . . . 83

4.17.2.1 Vector2D() [1/2] . . . 83
4.17.2.2 Vector2D() [2/2] . . . 83
4.17.3 Member Function Documentation . . . 83
4.17.3.1 getRotatedVector() . . . 83
4.18 Vector3D Struct Reference . . . 84
4.18.1 Detailed Description . . . 84
4.18.2 Constructor & Destructor Documentation . . . 85
4.18.2.1 Vector3D() [1/2] . . . 85
4.18.2.2 Vector3D() [2/2] . . . 85
4.18.3 Member Function Documentation . . . 85
4.18.3.1 operator"!=() . . . 85
4.18.3.2 operator*=() . . . 85
4.18.3.3 operator+=() . . . 85
4.18.3.4 operator-=() . . . 86
4.18.3.5 operator/=() . . . 86
4.18.3.6 operator==() . . . 86
4.18.3.7 operator[]() [1/2] . . . 86
4.18.3.8 operator[]() [2/2] . . . 86
4.18.4 Member Data Documentation . . . 86
4.18.4.1 x . . . 87
4.18.4.2 y . . . 87
4.18.4.3 z . . . 87
4.19 Wall Class Reference . . . 87
4.19.1 Detailed Description . . . 88
4.19.2 Constructor & Destructor Documentation . . . 88
4.19.2.1 Wall() [1/2] . . . 88
4.19.2.2 Wall() [2/2] . . . 89

5 File Documentation **91**

5.1 assets/fonts/LICENSE_GPLv3.txt File Reference . . . 91
5.1.1 Function Documentation . . . 96
5.1.1.1 Copyright() . . . 96
5.1.2 Variable Documentation . . . 97
5.1.2.1 not . . . 97
5.1.2.2 Version . . . 103
5.2 config/levels/1.txt File Reference . . . 103
5.3 config/levels/2.txt File Reference . . . 103
5.4 config/levels/3.txt File Reference . . . 103
5.5 config/levels/4.txt File Reference . . . 103
5.6 config/levels/5.txt File Reference . . . 103
5.7 include/Ball.hpp File Reference . . . 103
5.7.1 Detailed Description . . . 105

5.8 include/BreakoutGame.hpp File Reference . . . 105
5.8.1 Detailed Description . . . 106
5.9 include/Brick.hpp File Reference . . . 107
5.9.1 Detailed Description . . . 108
5.10 include/BricksGenerator.hpp File Reference . . . 109
5.10.1 Detailed Description . . . 110
5.11 include/Common.h File Reference . . . 110
5.11.1 Detailed Description . . . 112
5.11.2 Enumeration Type Documentation . . . 112
5.11.2.1 Buttons . . . 112
5.11.2.2 CollisionType . . . 113
5.11.2.3 GameState . . . 113
5.11.3 Variable Documentation . . . 113
5.11.3.1 BALL_HEIGHT . . . 114
5.11.3.2 BALL_SPEED . . . 114
5.11.3.3 BALL_START_DEGREE . . . 114
5.11.3.4 BALL_WIDTH . . . 114
5.11.3.5 BRICK_COLUMN . . . 114
5.11.3.6 BRICK_DEFAULT_SCORE . . . 114
5.11.3.7 BRICK_HEIGHT . . . 114
5.11.3.8 BRICK_INTERVAL . . . 114
5.11.3.9 BRICK_ROW . . . 115
5.11.3.10 BRICK_START_HEIGHT . . . 115
5.11.3.11 BRICK_WIDTH . . . 115
5.11.3.12 DEFAULT_FONT_SIZE . . . 115
5.11.3.13 DEFAULT_LEVEL . . . 115
5.11.3.14 GAME_SCENE_LEFT . . . 115
5.11.3.15 GAME_SCENE_RIGHT . . . 115
5.11.3.16 GAME_SCENE_WIDTH . . . 116
5.11.3.17 MENU_FONT_SIZE . . . 116
5.11.3.18 PADDLE_DISTANCE_FROM_BOTTOM . . . 116
5.11.3.19 PADDLE_HEIGHT . . . 116
5.11.3.20 PADDLE_SPEED . . . 116
5.11.3.21 PADDLE_WIDTH . . . 116
5.11.3.22 PLAYER_DEFAULT_LIFE_NUM . . . 116
5.11.3.23 SCREEN_FPS_60 . . . 116
5.11.3.24 SCREEN_TICKS_PER_FRAME_60 . . . 117
5.11.3.25 TICKS_PER_UPDATE . . . 117
5.11.3.26 WALL_WIDTH . . . 117
5.11.3.27 WINDOW_HEIGHT . . . 117
5.11.3.28 WINDOW_WIDTH . . . 117
5.12 include/ConfigUtil.hpp File Reference . . . 117

5.12.1 Detailed Description . . . 118
5.13 include/LanguageManager.hpp File Reference . . . 119
5.13.1 Detailed Description . . . 120
5.14 include/LTimer.h File Reference . . . 120
5.14.1 Detailed Description . . . 121
5.15 include/Paddle.hpp File Reference . . . 122
5.15.1 Detailed Description . . . 123
5.16 include/Player.hpp File Reference . . . 124
5.17 include/Rectangle.hpp File Reference . . . 125
5.17.1 Detailed Description . . . 126
5.18 include/ResourceManager.hpp File Reference . . . 126
5.18.1 Detailed Description . . . 127
5.18.2 Typedef Documentation . . . 127
5.18.2.1 ResFilePath . . . 128
5.18.2.2 ResName . . . 128
5.19 include/Text.hpp File Reference . . . 128
5.19.1 Detailed Description . . . 129
5.20 include/TinyMath.hpp File Reference . . . 130
5.20.1 Detailed Description . . . 131
5.20.2 Enumeration Type Documentation . . . 131
5.20.2.1 Direction . . . 131
5.20.3 Function Documentation . . . 132
5.20.3.1 CrossProduct() . . . 132
5.20.3.2 Dot() . . . 132
5.20.3.3 getUnitVectorFromDegree() . . . 132
5.20.3.4 Magnitude() . . . 133
5.20.3.5 Normalize() . . . 133
5.20.3.6 operator∗() [1/3] . . . 134
5.20.3.7 operator∗() [2/3] . . . 134
5.20.3.8 operator∗() [3/3] . . . 135
5.20.3.9 operator+() . . . 135
5.20.3.10 operator-() [1/2] . . . 135
5.20.3.11 operator-() [2/2] . . . 135
5.20.3.12 operator/() . . . 136
5.20.3.13 Project() . . . 136
5.20.3.14 VectorDirectionSDL() . . . 136
5.21 include/Wall.hpp File Reference . . . 137
5.21.1 Detailed Description . . . 138
5.22 src/BreakoutGame.cpp File Reference . . . 139
5.22.1 Detailed Description . . . 139
5.23 src/common.cpp File Reference . . . 139
5.23.1 Detailed Description . . . 141

5.23.2 Variable Documentation . . . 141
5.23.2.1 BALL_HEIGHT . . . 141
5.23.2.2 BALL_SPEED . . . 141
5.23.2.3 BALL_START_DEGREE . . . 141
5.23.2.4 BALL_WIDTH . . . 142
5.23.2.5 BRICK_COLUMN . . . 142
5.23.2.6 BRICK_DEFAULT_SCORE . . . 142
5.23.2.7 BRICK_HEIGHT . . . 142
5.23.2.8 BRICK_INTERVAL . . . 142
5.23.2.9 BRICK_ROW . . . 142
5.23.2.10 BRICK_START_HEIGHT . . . 142
5.23.2.11 BRICK_WIDTH . . . 142
5.23.2.12 DEFAULT_FONT_SIZE . . . 143
5.23.2.13 DEFAULT_LEVEL . . . 143
5.23.2.14 GAME_SCENE_LEFT . . . 143
5.23.2.15 GAME_SCENE_RIGHT . . . 143
5.23.2.16 GAME_SCENE_WIDTH . . . 143
5.23.2.17 MENU_FONT_SIZE . . . 143
5.23.2.18 PADDLE_DISTANCE_FROM_BOTTOM . . . 143
5.23.2.19 PADDLE_HEIGHT . . . 144
5.23.2.20 PADDLE_SPEED . . . 144
5.23.2.21 PADDLE_WIDTH . . . 144
5.23.2.22 PLAYER_DEFAULT_LIFE_NUM . . . 144
5.23.2.23 SCREEN_FPS_60 . . . 144
5.23.2.24 SCREEN_TICKS_PER_FRAME_60 . . . 144
5.23.2.25 TICKS_PER_UPDATE . . . 144
5.23.2.26 WALL_WIDTH . . . 145
5.23.2.27 WINDOW_HEIGHT . . . 145
5.23.2.28 WINDOW_WIDTH . . . 145
5.24 src/LTimer.cpp File Reference . . . 145
5.24.1 Detailed Description . . . 146
5.24.2 Function Documentation . . . 146
5.24.2.1 isStarted() . . . 146
5.25 src/main.cpp File Reference . . . 146
5.25.1 Detailed Description . . . 147
5.25.2 Function Documentation . . . 147
5.25.2.1 main() . . . 147

Index **149**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Ball . . . 7
BreakoutGame . . . 12
BricksGenerator . . . 36
ConfigUtil . . . 38
Contact . . . 40
Language . . . 41
LanguageSelector . . . 42
LevelData . . . 45
LTimer . . . 46
Matrix3D . . . 50
Player . . . 57
Rectangle . . . 62
  Brick . . . 32
  Paddle . . . 54
  Wall . . . 87
ResourceManager . . . 64
Text . . . 74
Vector3D . . . 84
  Vector2D . . . 82

Chapter 2

Class Index

2.1 Class List

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

Ball
: Represent the ball object in the break out game. Record velocity, position etc. info of the ball . . . 7

BreakoutGame
: Breakout Game main Class, where main logic located . . . 12

Brick
: Brick object Class . . . 32

BricksGenerator
: Helper class to load and generate each level bricks arrangement . . . 36

ConfigUtil
: Help class which use nlohmann::json library to process json format config file . . . 38

Contact
: Collision info struct . . . 40

Language
: Struct to store all key-value pair of game texts . . . 41

LanguageSelector
: Help class to read, load and manage multi language text data . . . 42

LevelData
: Struct to store brick arrangement data of one level . . . 45

LTimer
: The application time based timer . . . 46

Matrix3D
: Matrix 3D represents 3x3 matrices in Math . . . 50

Paddle
: Paddle object class . . . 54

Player
: Player class, store info of the player . . . 57

Rectangle
: A simple rectange class to represnet a rectangle on screen . . . 62

ResourceManager
: Singleton ResourceManager manage resources which loaded from files . . . 64

Text
: A text wrapper class . . . 74

Vector2D . . . 82

Vector3D . . . 84

Wall
: Wall object Class . . . 87

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/Ball.hpp
    Ball object class . . . 103
include/BreakoutGame.hpp
    BreakoutGame main Class Header . . . 105
include/Brick.hpp
    Brick object class . . . 107
include/BricksGenerator.hpp
    Header file for BricksGenerator . . . 109
include/Common.h
    Header file to declare all global variables . . . 110
include/ConfigUtil.hpp
    Helper class to load config from file . . . 117
include/LanguageManager.hpp
    Helper class LanguageSelector . . . 119
include/LTimer.h
    Simple Timer Class . . . 120
include/Paddle.hpp
    Paddle object Class . . . 122
include/Player.hpp . . . 124
include/Rectangle.hpp
    A rectangle class . . . 125
include/ResourceManager.hpp
    ResourceManager class header . . . 126
include/Text.hpp
    Text object class . . . 128
include/TinyMath.hpp
    A tiny math library . . . 130
include/Wall.hpp
    Wall object class header . . . 137
src/BreakoutGame.cpp
    BreakoutGame Class Implementation . . . 139
src/common.cpp
    Definition of all global variables . . . 139
src/LTimer.cpp
    LTimer Class implementation . . . 145
src/main.cpp
    Enter point of the game . . . 146

Chapter 4

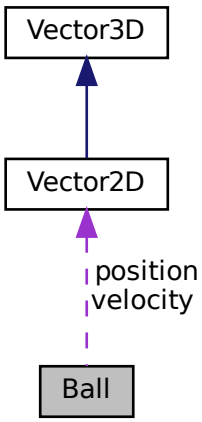
Class Documentation

4.1 Ball Class Reference

Represent the ball object in the break out game. Record velocity, position etc. info of the ball.

`#include <Ball.hpp>`

Collaboration diagram for Ball:



Public Member Functions

- Ball ()=default

  Construct a default all 0 value Ball object.
- Ball (Vector2D position, Vector2D velocity)

  Construct a new Ball object.
- void Update (float dt)

  Update the state of the ball.

- void Draw (SDL_Renderer *renderer)

  Draw ball on screen.
- void CollideWithPaddle (Contact const &contact)

  Update ball's state when collide with paddle.
- void CollideWithWall (Contact const &contact)

  Update ball's state when collide with wall.
- void CollideWithBrick (Contact const &contact)

  Update ball's state when collide with brick.

Public Attributes

- Vector2D position

  2D vector to store ball's position
- Vector2D velocity

  2D vector to store ball's velocity
- SDL_Rect rect {}

  A rectangle struct with left top position , height, width.

4.1.1 Detailed Description

Represent the ball object in the break out game. Record velocity, position etc. info of the ball.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Ball() [1/2]

```
Ball::Ball ( )  [default]
```

Construct a default all 0 value Ball object.

4.1.2.2 Ball() [2/2]

```
Ball::Ball (
            Vector2D position,
            Vector2D velocity )  [inline]
```

Construct a new Ball object.

Parameters

position	Initial position
velocity	The velocity of the ball

4.1.3 Member Function Documentation

4.1.3.1 CollideWithBrick()

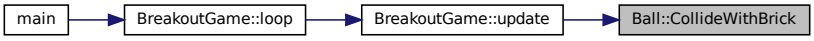
```
void Ball::CollideWithBrick (
            Contact const & contact )  [inline]
```

Update ball's state when collide with brick.

Parameters

contact	Collision info

Here is the caller graph for this function:



4.1.3.2 CollideWithPaddle()

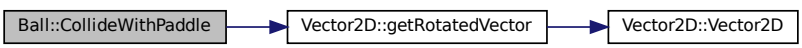
```
void Ball::CollideWithPaddle (
            Contact const & contact )  [inline]
```

Update ball's state when collide with paddle.

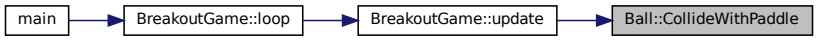
Parameters

contact	Collision info

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.3 CollideWithWall()

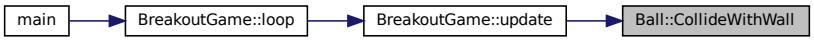
```
void Ball::CollideWithWall (
            Contact const & contact )  [inline]
```

Update ball's state when collide with wall.

Parameters

contact	Collision info

Here is the caller graph for this function:



4.1.3.4 Draw()

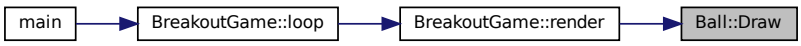
```
void Ball::Draw (
            SDL_Renderer * renderer )  [inline]
```

Draw ball on screen.

Parameters

renderer	Global renderer pointer

Here is the caller graph for this function:



4.1.3.5 Update()

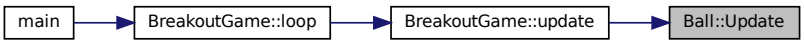
```
void Ball::Update (
            float dt )  [inline]
```

Update the state of the ball.

Parameters

dt	Time in milliseconds passed from last update

Here is the caller graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 position

```
Vector2D Ball::position
```

2D vector to store ball's position

4.1.4.2 rect

```
SDL_Rect Ball::rect {}
```

A rectangle struct with left top position , height, width.

4.1.4.3 velocity

```
Vector2D Ball::velocity
```

2D vector to store ball's velocity

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

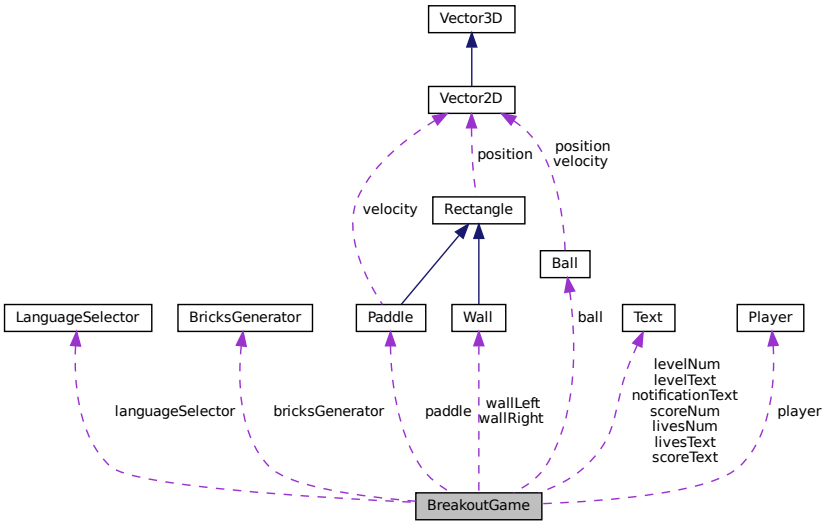
- include/Ball.hpp

4.2 BreakoutGame Class Reference

Breakout Game main Class, where main logic located.

```
#include <BreakoutGame.hpp>
```

Collaboration diagram for BreakoutGame:



Public Member Functions

- BreakoutGame (int w, int h, nlohmann::json js)

  Construct a new Breakout Game object.
- ~BreakoutGame ()

  Destroy the Breakout Game object.
- void update (float dt)

  Update game state.
- void render ()

  Render all things for next frame.
- void loop ()

  Game main loop.
- std::shared_ptr< SDL_Window > getSDLWindow ()

  Get current pointer to SDL_Window object.
- std::shared_ptr< SDL_Renderer > getSDLRenderer ()

  Get current pointer to SDL_Renderer object.

Private Member Functions

- void resetBricks ()

  Reset all current level bricks' state.
- bool initSDLSystems ()

  Init SDL systems.
- void initGameObjects ()

  Init all game Objects.
- bool loadResources ()

  Load all resource which need IO by using ResourceManger.
- void initBall ()

  Init or reset ball state.
- void initPaddle ()

  Init or reset paddle state.
- bool loadLevels ()

  Load level data from files.
- bool loadLanguages ()

  Load multi-language text data from file.
- void updateAllTexts ()

  Update all texts object when language change.
- Contact CheckPaddleCollision (Ball const &ball, Paddle const &paddle)

  Check whether ball collide with paddle.
- Contact CheckWallCollision (Ball const &ball)

  Check whether ball collide with wall.
- Contact CheckBrickCollision (Ball const &ball, Brick const &brick, float dt)

  Check whether ball collide with one brick.

Private Attributes

- nlohmann::json configs

  Json object contains loaded config data.
- GameState gameState

  Current game state.
- bool buttons [2] {}

  Button state array to represent whether left or right key(also 'a' and 'd') is pressed.
- int screenHeight

  Game window height.
- int screenWidth

  Game window width.
- int level = 1

  Current game level.
- int maxLevel

  Max level for this game.
- std::shared_ptr< SDL_Window > gWindow

  SDL window shared pointer.
- std::shared_ptr< SDL_Renderer > gRenderer

  SDL renderer shared pointer.
- std::shared_ptr< TTF_Font > contentFont_

  Font of score,player life,level.
- std::shared_ptr< TTF_Font > menuFont_

Font of all notification.

- std::shared_ptr< Mix_Chunk > wallHitSound

  Sound of ball hit wall.
- std::shared_ptr< Mix_Chunk > paddleHitSound

  Sound of ball hit paddle.
- std::shared_ptr< Mix_Chunk > winSound

  Sound of player win one level.
- std::shared_ptr< Mix_Chunk > loseSound

  Sound of player lose all lives.
- std::shared_ptr< Mix_Chunk > brickHitSound

  Sound of ball hit brick.
- std::shared_ptr< Mix_Chunk > loseLifeSound

  Sound of player lose one life.
- std::shared_ptr< Mix_Chunk > backgroundMusic

  Background music.
- Text scoreText

  Score Text.
- Text scoreNum

  Score number Text.
- Text livesText

  Lives Text.
- Text livesNum

  Player life number Text.
- Text levelText

  Level Text.
- Text levelNum

  Level number Text.
- Text notificationText

  Notification Text.
- Player player

  Player Object store player state.
- Wall wallLeft

  Wall object on the left.
- Wall wallRight

  Wall object on the right.
- Ball ball

  Ball object.
- Paddle paddle

  Paddle object.
- std::vector< Brick > bricks

  Array of all bricks in current level.
- BricksGenerator bricksGenerator

  Helper class to load level config and generate bricks for each level.
- LanguageSelector languageSelector

  Helper class to load different language texts.
- int restBricks

4.2.1 Detailed Description

Breakout Game main Class, where main logic located.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BreakoutGame()

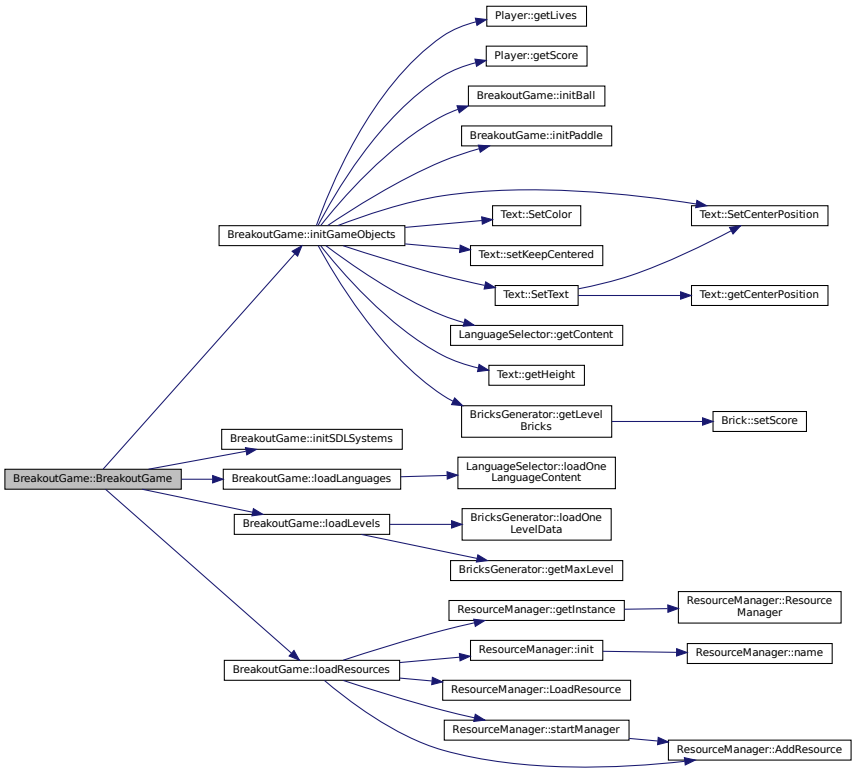
```
BreakoutGame::BreakoutGame (
            int w,
            int h,
            nlohmann::json js )
```

Construct a new Breakout Game object.

Parameters

w	Game program window width
h	Game program window height
js	Config json object

Here is the call graph for this function:

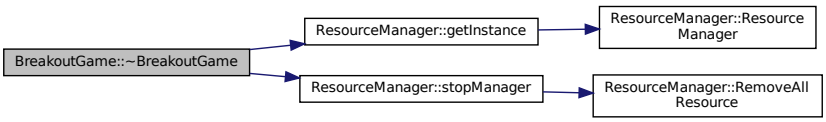


4.2.2.2 ~BreakoutGame()

```
BreakoutGame::~BreakoutGame ( )
```

Destroy the Breakout Game object.

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 CheckBrickCollision()

```
Contact BreakoutGame::CheckBrickCollision (
            Ball const & ball,
            Brick const & brick,
            float dt )  [private]
```

Check whether ball collide with one brick.

Parameters

ball	Ball object
brick	Brick object
dt	Milliseconds passed from last update

Returns

Contact Collision info

Here is the caller graph for this function:

4.2.3.2 CheckPaddleCollision()

```
Contact BreakoutGame::CheckPaddleCollision (
            Ball const & ball,
            Paddle const & paddle )  [private]
```

Check whether ball collide with paddle.

Parameters

ball	Ball object
paddle	Paddle object

Returns

Contact Collision info

Here is the caller graph for this function:

4.2.3.3 CheckWallCollision()

```
Contact BreakoutGame::CheckWallCollision (
            Ball const & ball )  [private]
```

Check whether ball collide with wall.

Parameters

ball	Ball object

Returns

Contact Collision info

Here is the caller graph for this function:

4.2.3.4 getSDLRenderer()

```
std::shared_ptr< SDL_Renderer > BreakoutGame::getSDLRenderer ( )
```

Get current pointer to SDL_Renderer object.

Returns

std::shared_ptr<SDL_Renderer> Shared pointer to SDL_Renderer

4.2.3.5 getSDLWindow()

```
std::shared_ptr< SDL_Window > BreakoutGame::getSDLWindow ( )
```

Get current pointer to SDL_Window object.

Returns

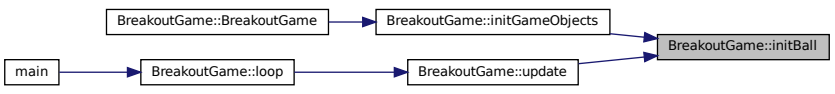
std::shared_ptr<SDL_Window> Shared pointer to SDL_Window

4.2.3.6 initBall()

```
void BreakoutGame::initBall ( )  [private]
```

Init or reset ball state.

Here is the caller graph for this function:

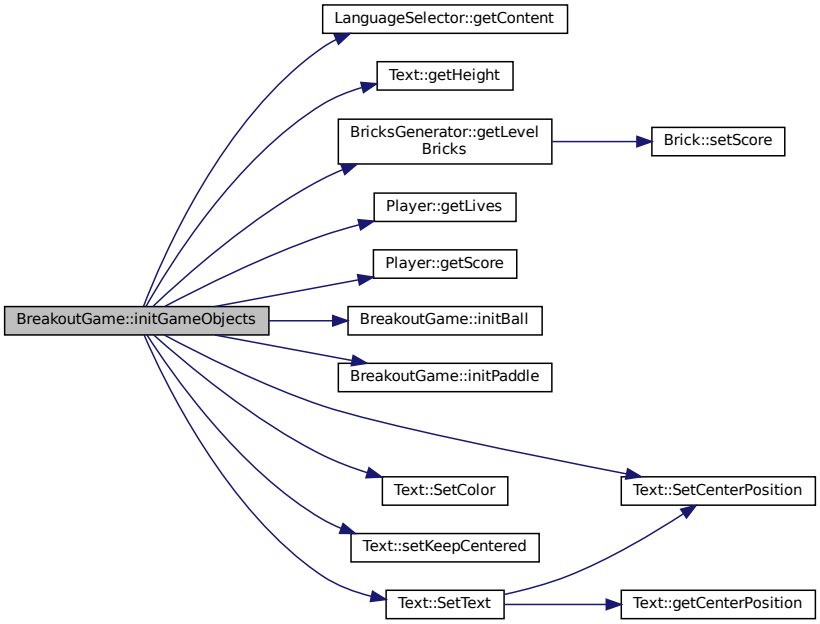


4.2.3.7 initGameObjects()

```
void BreakoutGame::initGameObjects ( )  [private]
```

Init all game Objects.

Here is the call graph for this function:



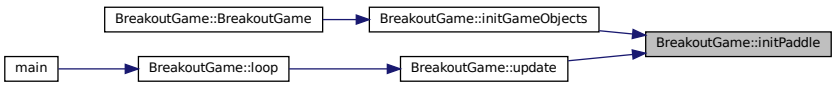
Here is the caller graph for this function:

4.2.3.8 initPaddle()

```
void BreakoutGame::initPaddle ( )  [private]
```

Init or reset paddle state.

Here is the caller graph for this function:



4.2.3.9 initSDLSystems()

```
bool BreakoutGame::initSDLSystems ( )  [private]
```

Init SDL systems.

Returns

true Init SDL and extra systems successfully

false Error occurs

Here is the caller graph for this function:

4.2.3.10 loadLanguages()

```
bool BreakoutGame::loadLanguages ( )  [private]
```

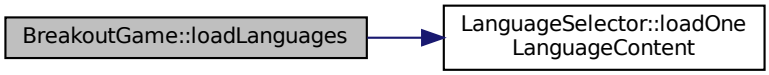
Load multi-language text data from file.

Returns

true Read and load all data successfully

false Error occurs

Here is the call graph for this function:



Here is the caller graph for this function:

4.2.3.11 loadLevels()

```
bool BreakoutGame::loadLevels ( )  [private]
```

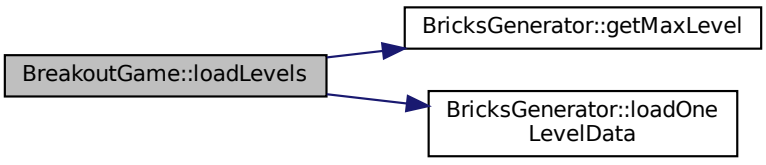
Load level data from files.

Returns

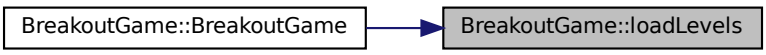
true Read and load level data successfully

false Error occurs

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.12 loadResources()

```
bool BreakoutGame::loadResources ( )  [private]
```

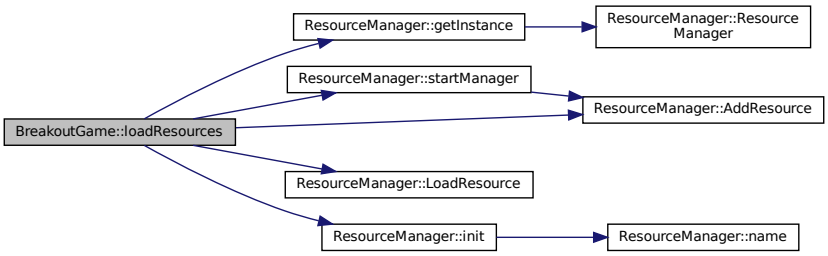
Load all resource which need IO by using ResourceManger.

Returns

true Load all resources successfully

false Error occurs

Here is the call graph for this function:



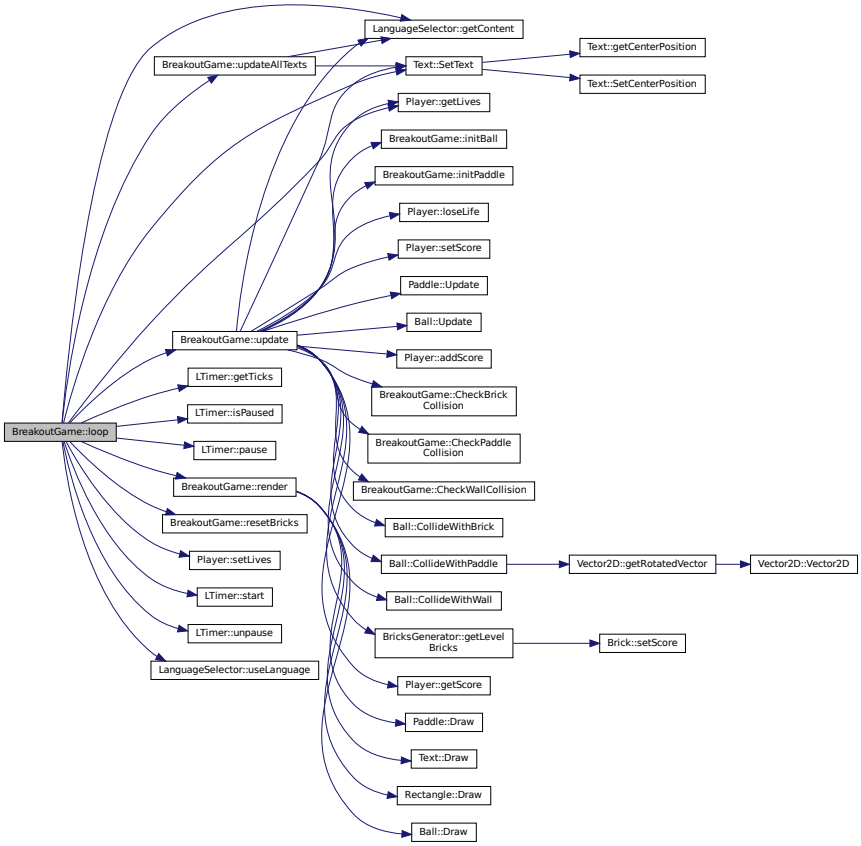
Here is the caller graph for this function:

4.2.3.13 loop()

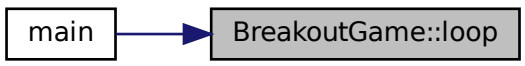
```
void BreakoutGame::loop ( )
```

Game main loop.

Here is the call graph for this function:



Here is the caller graph for this function:

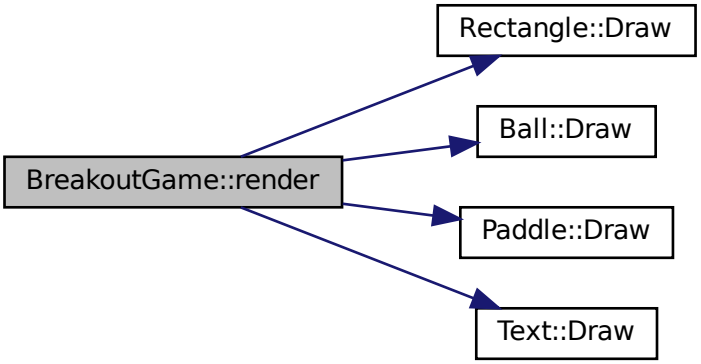


4.2.3.14 render()

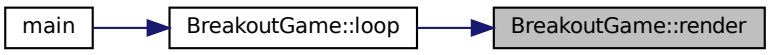
```
void BreakoutGame::render ( )
```

Render all things for next frame.

Here is the call graph for this function:



Here is the caller graph for this function:

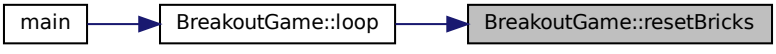


4.2.3.15 resetBricks()

```
void BreakoutGame::resetBricks ( )  [private]
```

Reset all current level bricks' state.

Here is the caller graph for this function:



4.2.3.16 update()

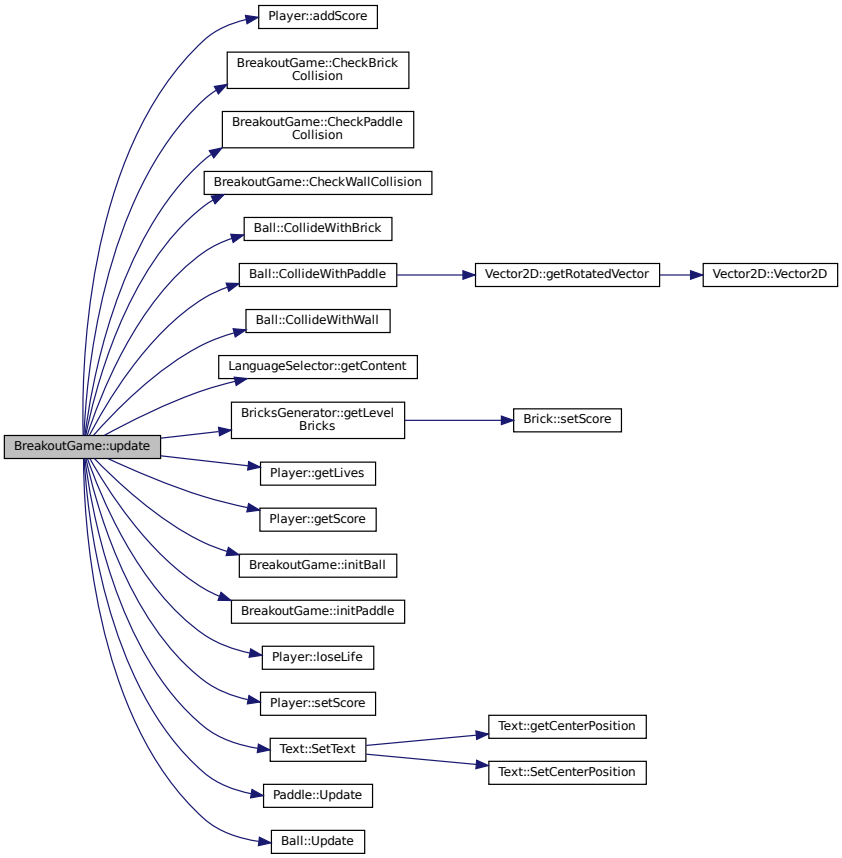
```
void BreakoutGame::update (
            float dt )
```

Update game state.

Parameters

dt	Milliseconds passed from last update

Here is the call graph for this function:



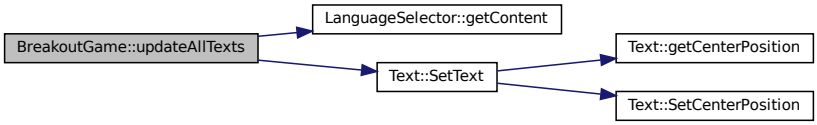
Here is the caller graph for this function:

4.2.3.17 updateAllTexts()

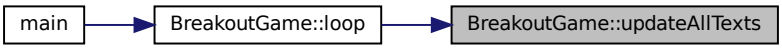
```
void BreakoutGame::updateAllTexts ( )  [private]
```

Update all texts object when language change.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 backgroundMusic

```
std::shared_ptr<Mix_Chunk> BreakoutGame::backgroundMusic  [private]
```

Background music.

4.2.4.2 ball

```
Ball BreakoutGame::ball  [private]
```

Ball object.

4.2.4.3 brickHitSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::brickHitSound  [private]
```

Sound of ball hit brick.

4.2.4.4 bricks

```
std::vector<Brick> BreakoutGame::bricks  [private]
```

Array of all bricks in current level.

4.2.4.5 bricksGenerator

```
BricksGenerator BreakoutGame::bricksGenerator  [private]
```

Helper class to load level config and generate bricks for each level.

4.2.4.6 buttons

```
bool BreakoutGame::buttons[2] {}  [private]
```

Button state array to represent whether left or right key(also 'a' and 'd') is pressed.

4.2.4.7 configs

```
nlohmann::json BreakoutGame::configs  [private]
```

Json object contains loaded config data.

4.2.4.8 contentFont_

```
std::shared_ptr<TTF_Font> BreakoutGame::contentFont_  [private]
```

Font of score,player life,level.

4.2.4.9 gameState

```
GameState BreakoutGame::gameState  [private]
```

Current game state.

4.2.4.10 gRenderer

```
std::shared_ptr<SDL_Renderer> BreakoutGame::gRenderer  [private]
```

SDL renderer shared pointer.

4.2.4.11 gWindow

```
std::shared_ptr<SDL_Window> BreakoutGame::gWindow  [private]
```

SDL window shared pointer.

4.2.4.12 languageSelector

```
LanguageSelector BreakoutGame::languageSelector  [private]
```

Helper class to load different language texts.

4.2.4.13 level

```
int BreakoutGame::level = 1  [private]
```

Current game level.

4.2.4.14 levelNum

```
Text BreakoutGame::levelNum  [private]
```

Level number Text.

4.2.4.15 levelText

```
Text BreakoutGame::levelText  [private]
```

Level Text.

4.2.4.16 livesNum

```
Text BreakoutGame::livesNum  [private]
```

Player life number Text.

4.2.4.17 livesText

```
Text BreakoutGame::livesText  [private]
```

Lives Text.

4.2.4.18 loseLifeSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::loseLifeSound  [private]
```

Sound of player lose one life.

4.2.4.19 loseSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::loseSound  [private]
```

Sound of player lose all lives.

4.2.4.20 maxLevel

```
int BreakoutGame::maxLevel  [private]
```

Max level for this game.

4.2.4.21 menuFont_

```
std::shared_ptr<TTF_Font> BreakoutGame::menuFont_  [private]
```

Font of all notification.

4.2.4.22 notificationText

```
Text BreakoutGame::notificationText  [private]
```

Notification Text.

4.2.4.23 paddle

```
Paddle BreakoutGame::paddle  [private]
```

Paddle object.

4.2.4.24 paddleHitSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::paddleHitSound  [private]
```

Sound of ball hit paddle.

4.2.4.25 player

```
Player BreakoutGame::player  [private]
```

Player Object store player state.

4.2.4.26 restBricks

```
int BreakoutGame::restBricks  [private]
```

4.2.4.27 scoreNum

```
Text BreakoutGame::scoreNum  [private]
```

Score number Text.

4.2.4.28 scoreText

```
Text BreakoutGame::scoreText  [private]
```

Score Text.

4.2.4.29 screenHeight

```
int BreakoutGame::screenHeight  [private]
```

Game window height.

4.2.4.30 screenWidth

```
int BreakoutGame::screenWidth  [private]
```

Game window width.

4.2.4.31 wallHitSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::wallHitSound  [private]
```

Sound of ball hit wall.

4.2.4.32 wallLeft

```
Wall BreakoutGame::wallLeft  [private]
```

Wall object on the left.

4.2.4.33 wallRight

```
Wall BreakoutGame::wallRight  [private]
```

Wall object on the right.

4.2.4.34 winSound

```
std::shared_ptr<Mix_Chunk> BreakoutGame::winSound  [private]
```

Sound of player win one level.

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

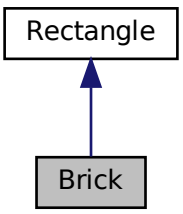
- include/BreakoutGame.hpp
- src/BreakoutGame.cpp

4.3 Brick Class Reference

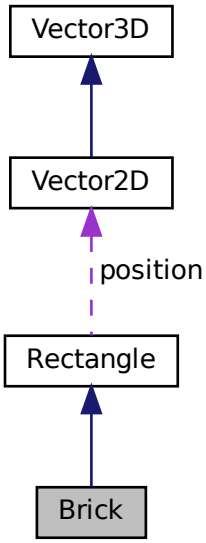
Brick object Class.

```
#include <Brick.hpp>
```

Inheritance diagram for Brick:



Collaboration diagram for Brick:



Public Member Functions

- Brick (Vector2D position, bool active=false)
  Construct a new Brick object.
- bool isActive () const
  Whether active.
- void setActive (bool state)
  Set the active state.
- int getScore () const
  Get the score of this brick.
- void setScore (int score)
  Set the brick's score.
- void Draw (SDL_Renderer *renderer)
  Draw brick on screen.

Private Attributes

- bool active
- int score = BRICK_DEFAULT_SCORE

Additional Inherited Members

4.3.1 Detailed Description

Brick object Class.

Store info of a brick: position,score whether active ...

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Brick()

```
Brick::Brick (
            Vector2D position,
            bool active = false )  [inline]
```

Construct a new Brick object.

Parameters

position	Position
active	Whether active

4.3.3 Member Function Documentation

4.3.3.1 Draw()

```
void Brick::Draw (
            SDL_Renderer * renderer )  [inline]
```

Draw brick on screen.

Parameters

renderer	Global SDL_Renderer pointer

4.3.3.2 getScore()

```
int Brick::getScore ( ) const  [inline]
```

Get the score of this brick.

Returns

int Score value

4.3.3.3 isActive()

```
bool Brick::isActive ( ) const  [inline]
```

Whether active.

Returns

true Active

false Inactive

4.3.3.4 setActive()

```
void Brick::setActive (
            bool state )  [inline]
```

Set the active state.

Parameters

state	New state

4.3.3.5 setScore()

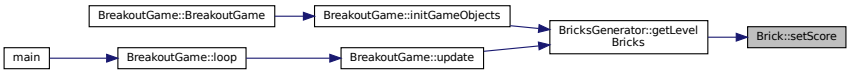
```
void Brick::setScore (
            int score )  [inline]
```

Set the brick's score.

Parameters

score	Score value

Here is the caller graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 active

```
bool Brick::active  [private]
```

4.3.4.2 score

```
int Brick::score = BRICK_DEFAULT_SCORE  [private]
```

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

- include/Brick.hpp

4.4 BricksGenerator Class Reference

Helper class to load and generate each level bricks arrangement.

`#include <BricksGenerator.hpp>`

Public Member Functions

- bool loadOneLevelData (const std::string &path)
  Load one level config data form a file.
- std::vector< Brick > getLevelBricks (int level) const
  Get the bricks info of specific level.
- int getMaxLevel () const
  Get the Max Level.

Private Attributes

- std::vector< LevelData > allLevels
  All level data store in an array.

4.4.1 Detailed Description

Helper class to load and generate each level bricks arrangement.

4.4.2 Member Function Documentation

4.4.2.1 getLevelBricks()

```
std::vector<Brick> BricksGenerator::getLevelBricks (
            int level ) const  [inline]
```

Get the bricks info of specific level.

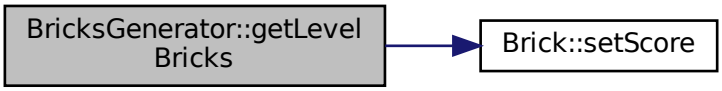
Parameters

level	Level number

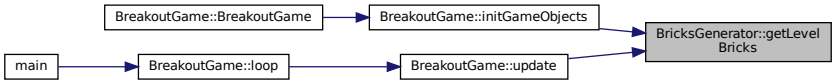
Returns

std::vector<Brick> Bricks array

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.2.2 getMaxLevel()

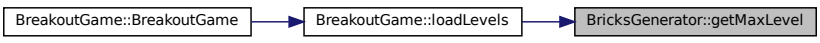
```
int BricksGenerator::getMaxLevel ( ) const  [inline]
```

Get the Max Level.

Returns

int Max Level number

Here is the caller graph for this function:



4.4.2.3 loadOneLevelData()

```
bool BricksGenerator::loadOneLevelData (
            const std::string & path )  [inline]
```

Load one level config data form a file.

Parameters

path	File path

Returns

true Success to load

false Error occurs

Here is the caller graph for this function:

4.4.3 Member Data Documentation

4.4.3.1 allLevels

```
std::vector<LevelData> BricksGenerator::allLevels  [private]
```

All level data store in an array.

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

- include/BricksGenerator.hpp

4.5 ConfigUtil Class Reference

Help class which use nlohmann::json library to process json format config file.

```
#include <ConfigUtil.hpp>
```

Static Public Member Functions

- static nlohmann::json loadConfig (const std::string &path)

  Get json object from file.
- static void loadAllVariables (nlohmann::json &js)

  Update all global variable value from json format config.

4.5.1 Detailed Description

Help class which use nlohmann::json library to process json format config file.

4.5.2 Member Function Documentation

4.5.2.1 loadAllVariables()

```
static void ConfigUtil::loadAllVariables (
            nlohmann::json & js )  [inline], [static]
```

Update all global variable value from json format config.

Parameters

js	Input json object

Here is the caller graph for this function:

4.5.2.2 loadConfig()

```
static nlohmann::json ConfigUtil::loadConfig (
            const std::string & path )  [inline], [static]
```

Get json object from file.

Parameters

path	File path

Returns

nlohmann::json Deserialized json object

Here is the caller graph for this function:

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

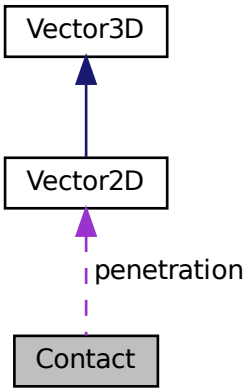
- include/ConfigUtil.hpp

4.6 Contact Struct Reference

Collision info struct.

`#include <Common.h>`

Collaboration diagram for Contact:



Public Attributes

- CollisionType type

  Collision direction.
- Vector2D penetration

  How much the ball has penetrated in hitted object.

4.6.1 Detailed Description

Collision info struct.

4.6.2 Member Data Documentation

4.6.2.1 penetration

```
Vector2D Contact::penetration
```

How much the ball has penetrated in hitted object.

4.6.2.2 type

```
CollisionType Contact::type
```

Collision direction.

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

- include/Common.h

4.7 Language Struct Reference

Struct to store all key-value pair of game texts.

```
#include <LanguageManager.hpp>
```

Public Attributes

- std::unordered_map< std::string, std::string > data

4.7.1 Detailed Description

Struct to store all key-value pair of game texts.

4.7.2 Member Data Documentation

4.7.2.1 data

```
std::unordered_map<std::string, std::string> Language::data
```

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

- include/LanguageManager.hpp

4.8 LanguageSelector Class Reference

Help class to read, load and manage multi language text data.

```
#include <LanguageManager.hpp>
```

Public Member Functions

- bool loadOneLanguageContent (const std::string &path, const std::string &name)

  Read and load one language data file.
- std::string const & getContent (const std::string &key) const
- void useLanguage (std::string const &lan)

  Set current language.

Private Attributes

- std::unordered_map< std::string, Language > languages

  all language text data
- std::string currentLanguage

  current language

4.8.1 Detailed Description

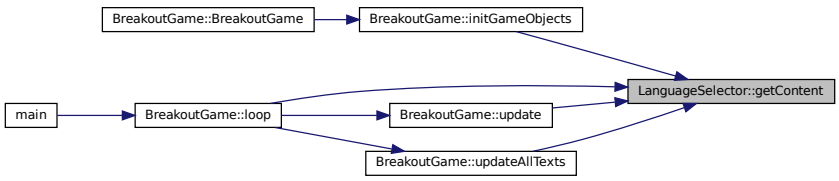
Help class to read, load and manage multi language text data.

4.8.2 Member Function Documentation

4.8.2.1 getContent()

```
std::string const& LanguageSelector::getContent (
            const std::string & key ) const  [inline]
```

Here is the caller graph for this function:



4.8.2.2 loadOneLanguageContent()

```
bool LanguageSelector::loadOneLanguageContent (
            const std::string & path,
            const std::string & name )  [inline]
```

Read and load one language data file.

Parameters

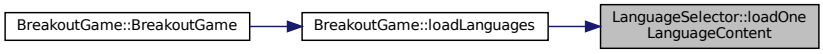
path	File path
name	Language name

Returns

true Success to read and load

false Error occurs

Here is the caller graph for this function:



4.8.2.3 useLanguage()

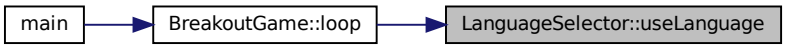
```
void LanguageSelector::useLanguage (
            std::string const & lan )  [inline]
```

Set current language.

Parameters

lan	Language name

Here is the caller graph for this function:



4.8.3 Member Data Documentation

4.8.3.1 currentLanguage

```
std::string LanguageSelector::currentLanguage  [private]
```

current language

4.8.3.2 languages

```
std::unordered_map<std::string, Language> LanguageSelector::languages  [private]
```

all language text data

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

- include/LanguageManager.hpp

4.9 LevelData Struct Reference

Struct to store brick arrangement data of one level.

```
#include <BricksGenerator.hpp>
```

Public Attributes

- std::vector< std::string > data

  Raw data stored in string for each row of bricks.

4.9.1 Detailed Description

Struct to store brick arrangement data of one level.

4.9.2 Member Data Documentation

4.9.2.1 data

```
std::vector<std::string> LevelData::data
```

Raw data stored in string for each row of bricks.

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

- include/BricksGenerator.hpp

4.10 LTimer Class Reference

The application time based timer.

```
#include <LTimer.h>
```

Public Member Functions

- LTimer ()
  Initializes variables.
- void start ()
  Start or reset and restart the timer.
- void stop ()
  Stop the timer.
- void pause ()
  Pause the timer.
- void unpause ()
  Unpause the timer.
- Uint32 getTicks ()
  Gets the timer's time.
- bool isStarted ()
  Check timer is started.
- bool isPaused ()
  Check timer is is paused.

Private Attributes

- Uint32 mStartTicks

  The clock time when the timer started.
- Uint32 mPausedTicks

  The ticks stored when the timer was paused.
- bool mPaused

  Whether paused.
- bool mStarted

  Whether started.

4.10.1 Detailed Description

The application time based timer.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 LTimer()

```
LTimer::LTimer ( )
```

Initializes variables.

4.10.3 Member Function Documentation

4.10.3.1 getTicks()

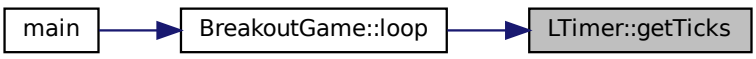
```
Uint32 LTimer::getTicks ( )
```

Gets the timer's time.

Returns

Uint32 Time value

Here is the caller graph for this function:



4.10.3.2 isPaused()

```
bool LTimer::isPaused ( )
```

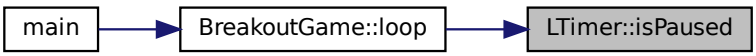
Check timer is is paused.

Returns

true Paused

false Not paused

Here is the caller graph for this function:



4.10.3.3 isStarted()

```
bool LTimer::isStarted ( )
```

Check timer is started.

Returns

true Started

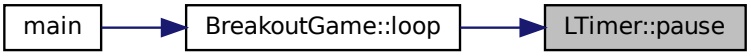
false Not started

4.10.3.4 pause()

```
void LTimer::pause ( )
```

Pause the timer.

Here is the caller graph for this function:

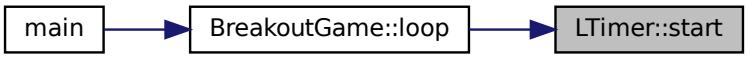


4.10.3.5 start()

```
void LTimer::start ( )
```

Start or reset and restart the timer.

Here is the caller graph for this function:



4.10.3.6 stop()

```
void LTimer::stop ( )
```

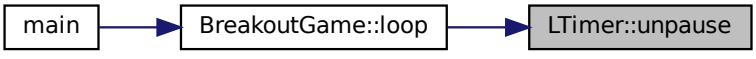
Stop the timer.

4.10.3.7 unpause()

```
void LTimer::unpause ( )
```

Unpause the timer.

Here is the caller graph for this function:



4.10.4 Member Data Documentation

4.10.4.1 mPaused

```
bool LTimer::mPaused  [private]
```

Whether paused.

4.10.4.2 mPausedTicks

```
Uint32 LTimer::mPausedTicks  [private]
```

The ticks stored when the timer was paused.

4.10.4.3 mStarted

```
bool LTimer::mStarted  [private]
```

Whether started.

4.10.4.4 mStartTicks

```
Uint32 LTimer::mStartTicks  [private]
```

The clock time when the timer started.

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

- include/LTimer.h
- src/LTimer.cpp

4.11 Matrix3D Struct Reference

Matrix 3D represents 3x3 matrices in Math.

```
#include <TinyMath.hpp>
```

Public Member Functions

- Matrix3D ()=default

  Construct a new Matrix3D object.
- Matrix3D (float n00, float n01, float n02, float n10, float n11, float n12, float n20, float n21, float n22)
- Matrix3D (const Vector3D &a, const Vector3D &b, const Vector3D &c)

  Matrix constructor from three vectors, putting each Vector to each column.
- float & operator() (int i, int j)
- const float & operator() (int i, int j) const
- Vector3D & operator[ ] (int i)

  Return a row from a matrix as a vector.
- const Vector3D & operator[ ] (int i) const

  Return a row from a matrix as a vector.
- bool operator== (const Matrix3D &m) const

  Equal operator.
- bool operator!= (const Matrix3D &m) const

  Unequal operator.
- Vector3D column (int j) const

  Return a column from a matrix as a vector.

Private Attributes

- float n [3][3]

  Store each value of the matrix.

4.11.1 Detailed Description

Matrix 3D represents 3x3 matrices in Math.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Matrix3D() [1/3]

```
Matrix3D::Matrix3D ( )  [default]
```

Construct a new Matrix3D object.

4.11.2.2 Matrix3D() [2/3]

```
Matrix3D::Matrix3D (
            float n00,
            float n01,
            float n02,
            float n10,
            float n11,
            float n12,
            float n20,
            float n21,
            float n22 )  [inline]
```

Matrix constructor with 9 scalar values. Row-major order

4.11.2.3 Matrix3D() [3/3]

```
Matrix3D::Matrix3D (
            const Vector3D & a,
            const Vector3D & b,
            const Vector3D & c )  [inline]
```

Matrix constructor from three vectors, putting each Vector to each column.

4.11.3 Member Function Documentation

4.11.3.1 column()

```
Vector3D Matrix3D::column (
            int j ) const  [inline]
```

Return a column from a matrix as a vector.

Here is the caller graph for this function:

4.11.3.2 operator"!=()

```
bool Matrix3D::operator!= (
            const Matrix3D & m ) const  [inline]
```

Unequal operator.

Here is the call graph for this function:

4.11.3.3 operator()() [1/2]

```
float& Matrix3D::operator() (
            int i,
            int j )  [inline]
```

Index operator with two dimensions Example: M(1,1) returns row 1 and column 1 of matrix M.

4.11.3.4 operator()() [2/2]

```
const float& Matrix3D::operator() (
            int i,
            int j ) const  [inline]
```

Index operator with two dimensions Example: M(1,1) returns row 1 and column 1 of matrix M.

4.11.3.5 operator==()

```
bool Matrix3D::operator== (
            const Matrix3D & m ) const  [inline]
```

Equal operator.

Here is the caller graph for this function:

4.11.3.6 operator[]() [1/2]

```
Vector3D& Matrix3D::operator[] (
            int i )  [inline]
```

Return a row from a matrix as a vector.

4.11.3.7 operator[]() [2/2]

```
const Vector3D& Matrix3D::operator[] (
            int i ) const  [inline]
```

Return a row from a matrix as a vector.

4.11.4 Member Data Documentation

4.11.4.1 n

```
float Matrix3D::n[3][3]  [private]
```

Store each value of the matrix.

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

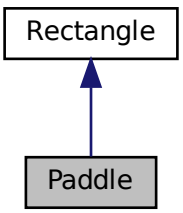
- include/TinyMath.hpp

4.12 Paddle Class Reference

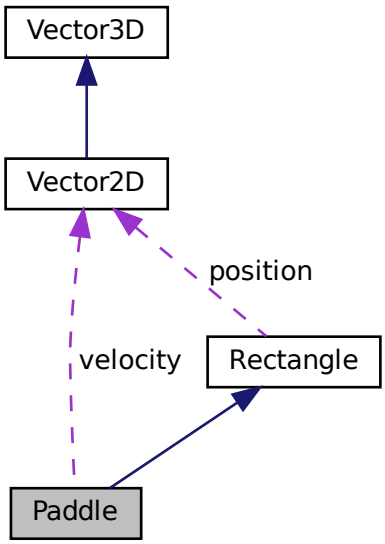
Paddle object class.

```
#include <Paddle.hpp>
```

Inheritance diagram for Paddle:



Collaboration diagram for Paddle:



Public Member Functions

- Paddle ()=default

  Construct a new Paddle object.
- Paddle (Vector2D position, Vector2D velocity)

  Construct a new Paddle object.
- void Update (float dt)

  Update paddle state.
- void Draw (SDL_Renderer ∗renderer)

  Draw the paddle on the screen.

Public Attributes

- Vector2D velocity

  The current velocity of the paddle.

4.12.1 Detailed Description

Paddle object class.

Keep track the info of the paddle in the game

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Paddle() [1/2]

```
Paddle::Paddle ( )  [default]
```

Construct a new Paddle object.

4.12.2.2 Paddle() [2/2]

```
Paddle::Paddle (
            Vector2D position,
            Vector2D velocity )  [inline]
```

Construct a new Paddle object.

Parameters

position	Start position
velocity	Inital velocity

4.12.3 Member Function Documentation

4.12.3.1 Draw()

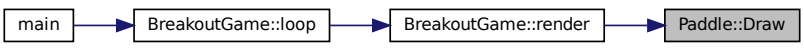
```
void Paddle::Draw (
            SDL_Renderer * renderer )  [inline]
```

Draw the paddle on the screen.

Parameters

renderer	The global SDL_Renderer

Here is the caller graph for this function:



4.12.3.2 Update()

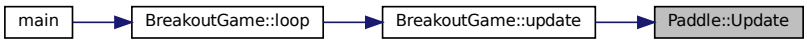
```
void Paddle::Update (
            float dt )  [inline]
```

Update paddle state.

Parameters

dt	Milliseconds from last updated

Here is the caller graph for this function:



4.12.4 Member Data Documentation

4.12.4.1 velocity

```
Vector2D Paddle::velocity
```

The current velocity of the paddle.

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

- include/Paddle.hpp

4.13 Player Class Reference

Player class, store info of the player.

```
#include <Player.hpp>
```

Public Member Functions

- Player ()=default

  Construct a new Player object.
- Player (int s, int l)

  Construct a new Player object.
- void addScore (int s)

  Add score.
- void setScore (int s)

  Set the Score.
- void setLives (int l)

  Set the Lives.
- void loseLife ()

  lose one life
- int getScore () const

  Get the current player's score.
- int getLives () const

  Get the current player's lives.

Private Attributes

- int score = 0

  Curret score of the player.
- int lives = PLAYER_DEFAULT_LIFE_NUM

  Current rest lives of the player.

4.13.1 Detailed Description

Player class, store info of the player.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Player() [1/2]

```
Player::Player ( )  [default]
```

Construct a new Player object.

4.13.2.2 Player() [2/2]

```
Player::Player (
            int s,
            int l )  [inline]
```

Construct a new Player object.

Parameters

s	Default score
l	Default livies

4.13.3 Member Function Documentation

4.13.3.1 addScore()

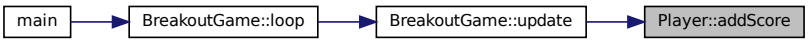
```
void Player::addScore (
            int s )  [inline]
```

Add score.

Parameters

s	Number to add

Here is the caller graph for this function:



4.13.3.2 getLives()

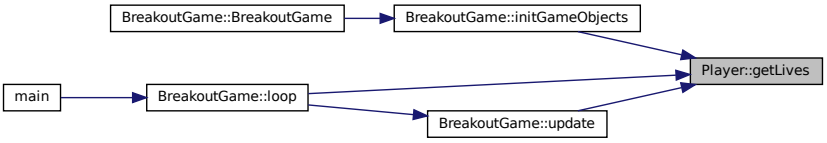
```
int Player::getLives ( ) const  [inline]
```

Get the current player's lives.

Returns

int Lives number

Here is the caller graph for this function:



4.13.3.3 getScore()

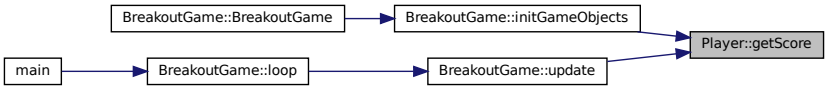
```
int Player::getScore ( ) const  [inline]
```

Get the current player's score.

Returns

int Score number

Here is the caller graph for this function:

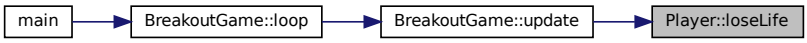


4.13.3.4 loseLife()

```
void Player::loseLife ( )  [inline]
```

lose one life

decrease 1 on lives Here is the caller graph for this function:



4.13.3.5 setLives()

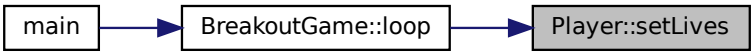
```
void Player::setLives (
            int l )  [inline]
```

Set the Lives.

Parameters

l	Number to set

Here is the caller graph for this function:



4.13.3.6 setScore()

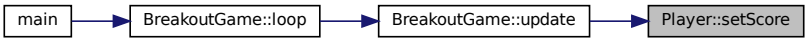
```
void Player::setScore (
            int s )  [inline]
```

Set the Score.

Parameters

s	Number to set

Here is the caller graph for this function:



4.13.4 Member Data Documentation

4.13.4.1 lives

```
int Player::lives = PLAYER_DEFAULT_LIFE_NUM  [private]
```

Current rest lives of the player.

4.13.4.2 score

```
int Player::score = 0  [private]
```

Curret score of the player.

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

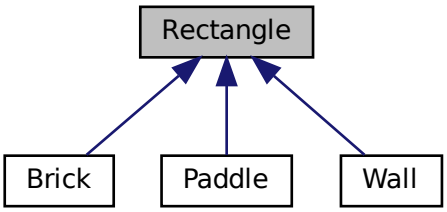
- include/Player.hpp

4.14 Rectangle Class Reference

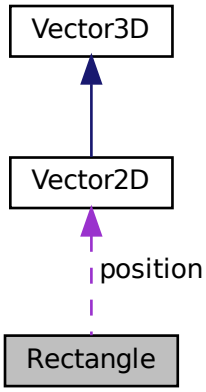
A simple rectange class to represnet a rectangle on screen.

`#include <Rectangle.hpp>`

Inheritance diagram for Rectangle:



Collaboration diagram for Rectangle:



Public Member Functions

- Rectangle ()=default

  Construct a new Rectangle object.
- Rectangle (Vector2D position, int w, int h)

  Construct a new Rectangle object.
- void Draw (SDL_Renderer ∗renderer)

  Draw the rectangle on screen.

Public Attributes

- Vector2D position

  The position of this rectangle.
- SDL_Rect rect {}

  A rectangle struct with left top position , height, width.

4.14.1 Detailed Description

A simple rectange class to represnet a rectangle on screen.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Rectangle() [1/2]

```
Rectangle::Rectangle ( )  [default]
```

Construct a new Rectangle object.

4.14.2.2 Rectangle() [2/2]

```
Rectangle::Rectangle (
            Vector2D position,
            int w,
            int h )  [inline]
```

Construct a new Rectangle object.

Parameters

position	Position of the Rectangle
w	Width of the Rectangle
h	Height of the Rectangle

4.14.3 Member Function Documentation

4.14.3.1 Draw()

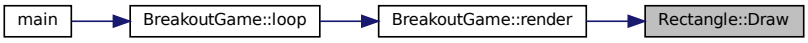
```
void Rectangle::Draw (
            SDL_Renderer * renderer )  [inline]
```

Draw the rectangle on screen.

Parameters

renderer	The pointer to the global SDL_Renderer

Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 position

```
Vector2D Rectangle::position
```

The position of this rectangle.

4.14.4.2 rect

```
SDL_Rect Rectangle::rect {}
```

A rectangle struct with left top position , height, width.

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

- include/Rectangle.hpp

4.15 ResourceManager Class Reference

Singleton ResourceManager manage resources which loaded from files.

```
#include <ResourceManager.hpp>
```

Public Member Functions

- size_t size () const

  Get number of resources.
- std::string name () const

  Get the name of ResourceManager.
- void init (const std::string &name, const std::string &cfgFilePath)

  Init the ResourceManager.
- bool startManager (bool preload=false, void *args=nullptr)

  Start the manager, if preload enable, then load all resource according to the config file.
- bool stopManager ()

  Stop the manager.
- bool AddResource (const ResName &resName, void *args)

  Add a resource by find and load resource in resource-path map.
- bool AddResource (const ResName &resName, std::shared_ptr< SDL_RWops > newRes)

  Directly add a new resource.
- bool AddResource (const ResName &resName, const ResFilePath &resPath, void *args)

  Add a resource from file.
- std::shared_ptr< SDL_RWops > LoadResource (const ResName &resName) const

  Offer a resource to others to use.
- bool RemoveResource (const ResName &resName)

  Remove one resource by name.
- bool RemoveAllResource ()

  Remove all resources.
- ~ResourceManager ()

  Destroy the Resource Manager object.

Static Public Member Functions

- static ResourceManager & getInstance ()

  Get the reference to the ResourceManager Instance.

Private Member Functions

- ResourceManager ()

  Construct a new Resource Manager object.
- ResourceManager (const ResourceManager &other)=delete

  Abandon copy constructor.
- ResourceManager & operator= (const ResourceManager &other)=delete

  Abandon copy assign.

Private Attributes

- std::string name_

  ResourceManager name.
- std::string cfgFilePath_

  The config file which can be automatically loaded.
- bool useConfig_ = false

  Whether load config file.
- std::unordered_map< ResName, ResFilePath > fileMap_

  (Resource name , Resource file path) pair
- std::unordered_map< ResName, std::shared_ptr< SDL_RWops > > resMap_

  (Resource name , Resource pointer) pair

4.15.1 Detailed Description

Singleton ResourceManager manage resources which loaded from files.

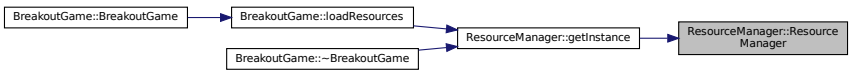
4.15.2 Constructor & Destructor Documentation

4.15.2.1 ResourceManager() [1/2]

```
ResourceManager::ResourceManager ( )  [inline], [private]
```

Construct a new Resource Manager object.

Here is the caller graph for this function:



4.15.2.2 ResourceManager() [2/2]

```
ResourceManager::ResourceManager (
            const ResourceManager & other )  [private], [delete]
```

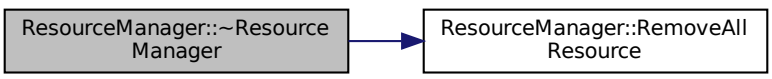
Abandon copy constructor.

4.15.2.3 ~ResourceManager()

```
ResourceManager::~ResourceManager ( )  [inline]
```

Destroy the Resource Manager object.

Here is the call graph for this function:



4.15.3 Member Function Documentation

4.15.3.1 AddResource() [1/3]

```
bool ResourceManager::AddResource (
            const ResName & resName,
            const ResFilePath & resPath,
            void * args )  [inline]
```

Add a resource from file.

Parameters

resName	Resource name
resPath	Resource file path
args	Extra args

Returns

true Success to add

false Fail to add

4.15.3.2 AddResource() [2/3]

```
bool ResourceManager::AddResource (
            const ResName & resName,
            std::shared_ptr< SDL_RWops > newRes )  [inline]
```

Directly add a new resource.

Parameters

resName	Resource name
newRes	Pointer to the resource object

Returns

true Success to add

false Fail to add

4.15.3.3 AddResource() [3/3]

```
bool ResourceManager::AddResource (
            const ResName & resName,
            void * args )  [inline]
```

Add a resource by find and load resource in resource-path map.

Parameters

resName	Resource name
args	Extra args

Returns

true Success to add

false Fail to add

Here is the caller graph for this function:

4.15.3.4 getInstance()

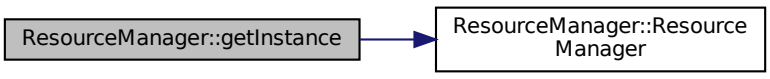
```
static ResourceManager& ResourceManager::getInstance ( )  [inline], [static]
```

Get the reference to the ResourceManager Instance.

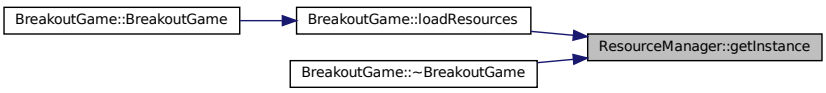
Returns

ResourceManager& The reference to the ResourceManager Instance

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.3.5 init()

```
void ResourceManager::init (
            const std::string & name,
            const std::string & cfgFilePath )  [inline]
```

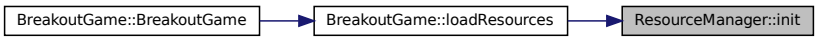
Init the ResourceManager.

Parameters

name	The name of the ResourceManager
cfgFilePath	Config file path, if empty then manage will not load config file

Here is the call graph for this function:

Here is the caller graph for this function:



4.15.3.6 LoadResource()

```
std::shared_ptr<SDL_RWops> ResourceManager::LoadResource (
            const ResName & resName ) const  [inline]
```

Offer a resource to others to use.

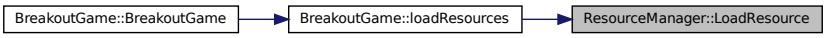
Parameters

resName	Resource name

Returns

std::shared_ptr<SDL_RWops> The pointer to the resource

Here is the caller graph for this function:



4.15.3.7 name()

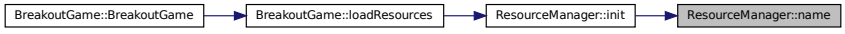
```
std::string ResourceManager::name ( ) const  [inline]
```

Get the name of ResourceManager.

Returns

std::string Name string

Here is the caller graph for this function:



4.15.3.8 operator=()

```
ResourceManager& ResourceManager::operator= (
            const ResourceManager & other )  [private], [delete]
```

Abandon copy assign.

4.15.3.9 RemoveAllResource()

```
bool ResourceManager::RemoveAllResource ( )  [inline]
```

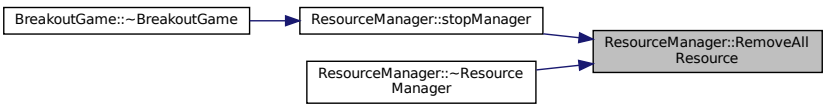
Remove all resources.

Returns

true Success to remove

false Fail to remove

Here is the caller graph for this function:



4.15.3.10 RemoveResource()

```
bool ResourceManager::RemoveResource (
            const ResName & resName )  [inline]
```

Remove one resource by name.

Parameters

resName	Resource name

Returns

true Success to remove

false Fail to remove

4.15.3.11 size()

```
size_t ResourceManager::size ( ) const  [inline]
```

Get number of resources.

Returns

size_t Number of resources

4.15.3.12 startManager()

```
bool ResourceManager::startManager (
            bool preload = false,
            void * args = nullptr )  [inline]
```

Start the manager, if preload enable, then load all resource according to the config file.

Parameters

preload	Whether preload
args	Extra args

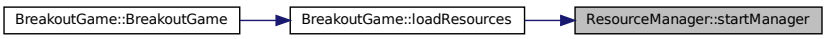
Returns

true Success to start

false Fail to start

Here is the call graph for this function:

Here is the caller graph for this function:



4.15.3.13 stopManager()

```
bool ResourceManager::stopManager ( )  [inline]
```

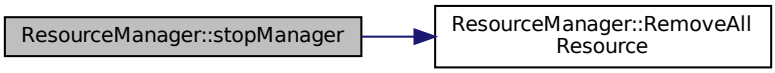
Stop the manager.

Returns

true Success to start

false Fail to start

Here is the call graph for this function:



Here is the caller graph for this function:

4.15.4 Member Data Documentation

4.15.4.1 cfgFilePath_

```
std::string ResourceManager::cfgFilePath_  [private]
```

The config file which can be automatically loaded.

4.15.4.2 fileMap_

```
std::unordered_map<ResName, ResFilePath> ResourceManager::fileMap_  [private]
```

(Resource name , Resource file path) pair

4.15.4.3 name_

```
std::string ResourceManager::name_  [private]
```

ResourceManager name.

4.15.4.4 resMap_

```
std::unordered_map<ResName, std::shared_ptr<SDL_RWops> > ResourceManager::resMap_  [private]
```

(Resource name , Resource pointer) pair

4.15.4.5 useConfig_

```
bool ResourceManager::useConfig_ = false  [private]
```

Whether load config file.

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

- include/ResourceManager.hpp

4.16 Text Class Reference

A text wrapper class.

```
#include <Text.hpp>
```

Public Member Functions

- Text ()=default
  Construct a new Text object.
- Text (Vector2D position, std::shared_ptr< SDL_Renderer > renderer, std::shared_ptr< TTF_Font > font)
  Construct a new Text object.
- void SetText (const std::string &str)
  Update the text content.
- void Draw ()
  Draw the text on screen.
- void SetPosition (const Vector2D &position)
  Set the Position of the text.
- void SetPosition (int x, int y)
  Set the Position of the text.
- void SetColor (Uint8 r, Uint8 g, Uint8 b, Uint8 a)
  Set the Color of the text.
- void SetCenterPosition (int x, int y)
  Set the Position of the text by given it's Center position.
- Vector2D getCenterPosition () const
  Get the Center Position.
- void setKeepCentered (bool state)
  Set the text maintain the its center position when its size changed.
- int getWidth () const
  Get the Width of the text.
- int getHeight () const
  Get the Height of the text.

Public Attributes

- std::shared_ptr< SDL_Renderer > renderer

  Shared pointer to global SDL_Renderer.
- std::shared_ptr< TTF_Font > font

  Shared pointer to TTF_Font.
- std::shared_ptr< SDL_Surface > surface

  Shared pointer to SDL_Surface.
- std::shared_ptr< SDL_Texture > texture

  Shared pointer to SDL_Texture.
- SDL_Rect rect

  A Rectange store the position and height, with of current text.
- SDL_Color color

  Text's color.
- std::string lastText

  The last text string, used to avoid updating by same string.
- bool keepCentered = false

  Whether keep the center position.

4.16.1 Detailed Description

A text wrapper class.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Text() [1/2]

```
Text::Text ( )  [default]
```

Construct a new Text object.

4.16.2.2 Text() [2/2]

```
Text::Text (
            Vector2D position,
            std::shared_ptr< SDL_Renderer > renderer,
            std::shared_ptr< TTF_Font > font )  [inline]
```

Construct a new Text object.

Parameters

position	The position of the text
renderer	Pinter to the renderer which used to render text
font	Pinter to the font object

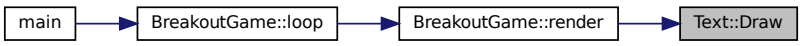
4.16.3 Member Function Documentation

4.16.3.1 Draw()

```
void Text::Draw ( )  [inline]
```

Draw the text on screen.

Here is the caller graph for this function:



4.16.3.2 getCenterPosition()

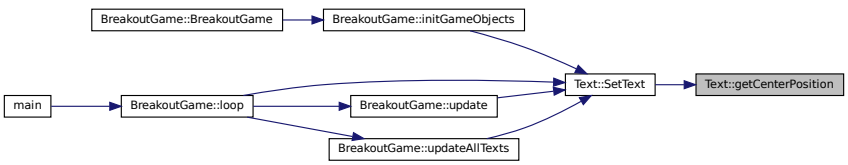
```
Vector2D Text::getCenterPosition ( ) const  [inline]
```

Get the Center Position.

Returns

Vector2D Center Position in a 2D vector

Here is the caller graph for this function:



4.16.3.3 getHeight()

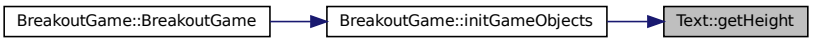
```
int Text::getHeight ( ) const  [inline]
```

Get the Height of the text.

Returns

int Height value

Here is the caller graph for this function:



4.16.3.4 getWidth()

```
int Text::getWidth ( ) const  [inline]
```

Get the Width of the text.

Returns

int Width value

4.16.3.5 SetCenterPosition()

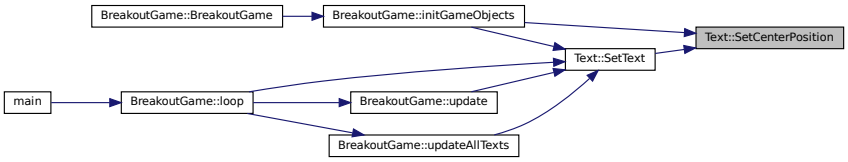
```
void Text::SetCenterPosition (
            int x,
            int y )  [inline]
```

Set the Position of the text by given it's Center position.

Parameters

x	Center position X-coordinate
y	Center position Y-coordinate

Here is the caller graph for this function:



4.16.3.6 SetColor()

```
void Text::SetColor (
            Uint8 r,
            Uint8 g,
            Uint8 b,
            Uint8 a )  [inline]
```

Set the Color of the text.

Parameters

r	Red value
g	Green value
b	Blue value
a	Alpha value

Here is the caller graph for this function:

4.16.3.7 setKeepCentered()

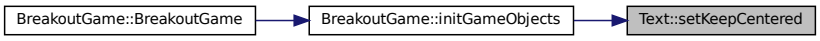
```
void Text::setKeepCentered (
            bool state )  [inline]
```

Set the text maintain the its center position when its size changed.

Parameters

state	Whether maintain

Here is the caller graph for this function:



4.16.3.8 SetPosition() [1/2]

```
void Text::SetPosition (
            const Vector2D & position )  [inline]
```

Set the Position of the text.

Parameters

position	New position

4.16.3.9 SetPosition() [2/2]

```
void Text::SetPosition (
            int x,
            int y )  [inline]
```

Set the Position of the text.

Parameters

x	X-coordinate
y	Y-coordinate

4.16.3.10 SetText()

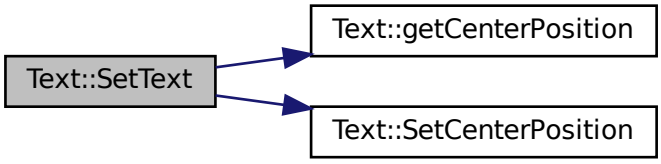
```
void Text::SetText (
            const std::string & str )  [inline]
```

Update the text content.

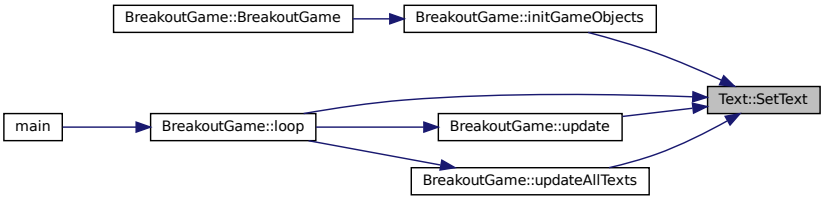
Parameters

str	New string

Here is the call graph for this function:



Here is the caller graph for this function:



4.16.4 Member Data Documentation

4.16.4.1 color

```
SDL_Color Text::color
```

Text's color.

4.16.4.2 font

```
std::shared_ptr<TTF_Font> Text::font
```

Shared pointer to TTF_Font.

4.16.4.3 keepCentered

```
bool Text::keepCentered = false
```

Whether keep the center position.

4.16.4.4 lastText

```
std::string Text::lastText
```

The last text string, used to avoid updating by same string.

4.16.4.5 rect

```
SDL_Rect Text::rect
```

A Rectange store the position and height, with of current text.

4.16.4.6 renderer

```
std::shared_ptr<SDL_Renderer> Text::renderer
```

Shared pointer to global SDL_Renderer.

4.16.4.7 surface

```
std::shared_ptr<SDL_Surface> Text::surface
```

Shared pointer to SDL_Surface.

4.16.4.8 texture

```
std::shared_ptr<SDL_Texture> Text::texture
```

Shared pointer to SDL_Texture.

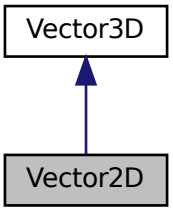
The documentation for this class was generated from the following file:

- include/Text.hpp

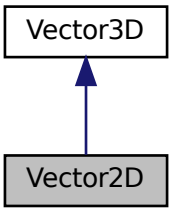
4.17 Vector2D Struct Reference

`#include <TinyMath.hpp>`

Inheritance diagram for Vector2D:



Collaboration diagram for Vector2D:



Public Member Functions

- Vector2D ()=default
- Vector2D (float a, float b)
- Vector2D getRotatedVector (float degree) const

Additional Inherited Members

4.17.1 Detailed Description

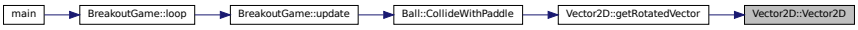
Vector2D performs vector operations with 2-dimensions The purpose of this class is primarily for 2D graphics applications.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Vector2D() [1/2]

```
Vector2D::Vector2D ( )  [default]
```

Here is the caller graph for this function:



4.17.2.2 Vector2D() [2/2]

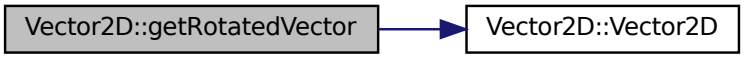
```
Vector2D::Vector2D (
            float a,
            float b )  [inline]
```

4.17.3 Member Function Documentation

4.17.3.1 getRotatedVector()

```
Vector2D Vector2D::getRotatedVector (
            float degree ) const  [inline]
```

Here is the call graph for this function:



Here is the caller graph for this function:

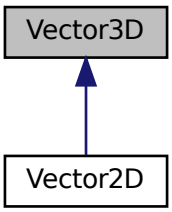
The documentation for this struct was generated from the following file:

- include/TinyMath.hpp

4.18 Vector3D Struct Reference

`#include <TinyMath.hpp>`

Inheritance diagram for Vector3D:



Public Member Functions

- Vector3D ()=default
- Vector3D (float a, float b, float c)
- float & operator[ ] (int i)
- const float & operator[ ] (int i) const
- Vector3D & operator*= (float s)
- Vector3D & operator/= (float s)

  Division Operator.
- Vector3D & operator+= (const Vector3D &v)

  Addition operator.
- Vector3D & operator-= (const Vector3D &v)

  Subtraction operator.
- bool operator== (const Vector3D &v) const

  Equal operator.
- bool operator!= (const Vector3D &v) const

  UnEqual operator.

Public Attributes

- float x
- float y
- float z

4.18.1 Detailed Description

Vector3D performs vector operations with 3-dimensions The purpose of this class is primarily for 3D graphics applications.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Vector3D() [1/2]

```
Vector3D::Vector3D ( )  [default]
```

Default conostrutcor 'why default?' https:///stackoverflow.com/questions/20828907/the-new-keyword-d

4.18.2.2 Vector3D() [2/2]

```
Vector3D::Vector3D (
            float a,
            float b,
            float c )  [inline]
```

The "Real" constructor we want to use. This initializes the values x,y,z

4.18.3 Member Function Documentation

4.18.3.1 operator"!=()

```
bool Vector3D::operator!= (
            const Vector3D & v ) const  [inline]
```

UnEqual operator.

4.18.3.2 operator*=()

```
Vector3D& Vector3D::operator*= (
            float s )  [inline]
```

Multiplication Operator Multiply vector by a uniform-scalar.

4.18.3.3 operator+=()

```
Vector3D& Vector3D::operator+= (
            const Vector3D & v )  [inline]
```

Addition operator.

4.18.3.4 operator-=()

```
Vector3D& Vector3D::operator-= (
            const Vector3D & v )  [inline]
```

Subtraction operator.

4.18.3.5 operator/=()

```
Vector3D& Vector3D::operator/= (
            float s )  [inline]
```

Division Operator.

4.18.3.6 operator==()

```
bool Vector3D::operator== (
            const Vector3D & v ) const  [inline]
```

Equal operator.

4.18.3.7 operator[]() [1/2]

```
float& Vector3D::operator[] (
            int i )  [inline]
```

Index operator, allowing us to access the individual x,y,z components of our vector. x,y,z are stored continuously, so we could recognize x as the first float number of a float[3] array There is no code to change here.

4.18.3.8 operator[]() [2/2]

```
const float& Vector3D::operator[] (
            int i ) const  [inline]
```

Index operator, allowing us to access the individual x,y,z components of our vector. x,y,z are stored continuously, so we could recognize x as the first float number of a float[3] array There is no code to change here.

4.18.4 Member Data Documentation

4.18.4.1 x

```
float Vector3D::x
```

Note: x,y,z are a convention x,y,z could be position, but also any 3-component value.

4.18.4.2 y

```
float Vector3D::y
```

4.18.4.3 z

```
float Vector3D::z
```

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

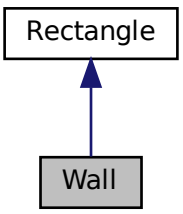
- include/TinyMath.hpp

4.19 Wall Class Reference

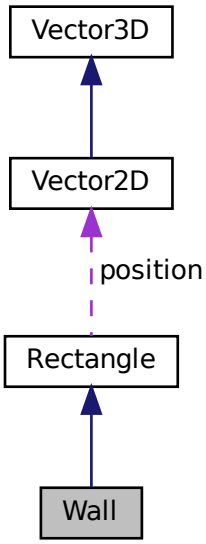
Wall object Class.

```
#include <Wall.hpp>
```

Inheritance diagram for Wall:



Collaboration diagram for Wall:



Public Member Functions

- Wall ()=default

  Construct a new Wall object.
- Wall (Vector2D position, int width)

  Construct a new Wall object.

Additional Inherited Members

4.19.1 Detailed Description

Wall object Class.

Store position info and width of the wall

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Wall() [1/2]

```
Wall::Wall ( )  [default]
```

Construct a new Wall object.

4.19.2.2 Wall() [2/2]

```
Wall::Wall (
            Vector2D position,
            int width )  [inline]
```

Construct a new Wall object.

Parameters

position	Wall position
width	Wall width

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

- include/Wall.hpp

Chapter 5

File Documentation

5.1 assets/fonts/LICENSE_GPLv3.txt File Reference

Functions

- GNU GENERAL PUBLIC LICENSE June Copyright (C) 2007 Free Software Foundation

Variables

- GNU GENERAL PUBLIC LICENSE Version
- GNU GENERAL PUBLIC LICENSE June Inc< http:Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble The GNU General Public License is a free, copyleft license for software and other kinds of works. The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software;it applies also to any other work released this way by its authors. You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software(and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things. To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it:responsibilities to respect the freedom of others. For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. Developers that use the GNU GPL protect your rights with two steps:(1) assert copyright on the software, and(2) offer you this License giving you legal permission to copy, distribute and/or modify it. For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions. Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users. Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those

that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free. The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions. "This License" refers to version 3 of the GNU General Public License. "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks. "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations. To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work. A "covered work" means either the unmodified Program or a work based on the Program. To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution(with or without modification), making available to the public, and in some countries other activities as well. To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that(1) displays an appropriate copyright notice, and(2) tells the user that there is no warranty for the work(except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work. A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language. The "System Libraries" of an executable work include anything, other than the work as a whole, that(a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and(b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component(kernel, window system, and so on) of the specific operating system(if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it. The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and(for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work 's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work. The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source. The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law. You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you. Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed;section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures. When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against

the work 's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies. You may convey verbatim copies of the Program 's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice;keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code;keep intact all notices of the absence of any warranty;and give all recipients a copy of this License along with the Program. You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:a) The work must carry prominent notices stating that you modified it, and giving a relevant date. b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. d) If the work has interactive user interfaces, each must display Appropriate Legal Notices;however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation 's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:a) Convey the object code in, or embodied in, a physical product(including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. b) Convey the object code in, or embodied in, a physical product(including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either(1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or(2) access to copy the Corresponding Source from a network server at no charge. c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. d) Convey the object code by offering access from a designated place(gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server(operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d. A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work. A "User Product" is either(1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or(2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product. "Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the

continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made. If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term(regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product(for example, the work has been installed in ROM). The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network. Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented(and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms. "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions. When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it.(Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission. Notwithstanding any other provision of this License, for material you add to a covered work, you may(if authorized by the copyright holders of that material) supplement the terms of this License with terms:a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License;or b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it;or c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version;or d) Limiting the use for publicity purposes of names of licensors or authors of the material;or e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks;or f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material(or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying. If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms. Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions;the above requirements apply either way. 8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License(including any patent licenses granted under the third paragraph of section 11). However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated(a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and(b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation. Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License(for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice. Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this

License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License. An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party 's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts. You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation(including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents. A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor 's "contributor version". A contributor 's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License. Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor 's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version. In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent(such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party. If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either(1) cause the Corresponding Source to be so available, or(2) arrange to deprive yourself of the benefit of the patent license for this particular work, or(3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient 's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid. If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it. A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license(a) in connection with copies of the covered work conveyed by you(or copies made from those copies), or(b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007. Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom. If conditions are imposed on you(whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered

work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License. The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation. If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy 's public statement of acceptance of a version permanently authorizes you to choose that version for the Program. Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WA↩ RRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOU↩ LD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLI↩ CABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU F↩ OR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM(INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR TH↩ IRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee. END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty;and each file should have at least the "copyright" line and a pointer to where the full notice is found.< one line to give the program 's name and a brief idea of what it does. > Copyright(C)< year >< name of author > This program is free software without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE See the GNU General Public License for more details You should have received a copy of the GNU General Public License along with this program If not

5.1.1 Function Documentation

5.1.1.1 Copyright()

```
GNU GENERAL PUBLIC LICENSE June Copyright (
            C  )
```

5.1.2 Variable Documentation

5.1.2.1 not

GNU GENERAL PUBLIC LICENSE June Inc<http: Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble The GNU General Public License is a free, copyleft license for software and other kinds of works. The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things. To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others. For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it. For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions. Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users. Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free. The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions. "This License" refers to version 3 of the GNU General Public License. "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks. "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations. To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work. A "covered work" means either the unmodified Program or a work based on the Program. To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well. To "convey" a work means any kind of propagation that enables other parties

to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying. An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work. A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language. The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it. The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work. The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source. The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law. You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you. Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures. When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies. You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this

License along with the Program. You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions: a) The work must carry prominent notices stating that you modified it, and giving a relevant date. b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways: a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d. A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work. A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made. If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM). The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network. Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms. "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions. When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission. Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms: a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or d) Limiting the use for publicity purposes of names of licensors or authors of the material; or e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors. All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying. If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms. Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11). However, if you cease all violation of this License, then your license from a particular copyright holder

is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation. Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice. Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License. An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts. You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents. A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version". A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License. Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version. In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party. If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid. If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of

the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it. A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007. Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom. If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License. The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation. If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program. Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHER↩ WISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, T↩ HE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFE↩ CTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL A↩ NY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQU↩ ENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT L↩ IMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THI↩ RD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HO↩ LDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in

connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee. END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author> This program is free software without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE See the GNU General Public License for more details You should have received a copy of the GNU General Public License along with this program If not

5.1.2.2 Version

GNU GENERAL PUBLIC LICENSE Version

5.2 config/levels/1.txt File Reference

5.3 config/levels/2.txt File Reference

5.4 config/levels/3.txt File Reference

5.5 config/levels/4.txt File Reference

5.6 config/levels/5.txt File Reference

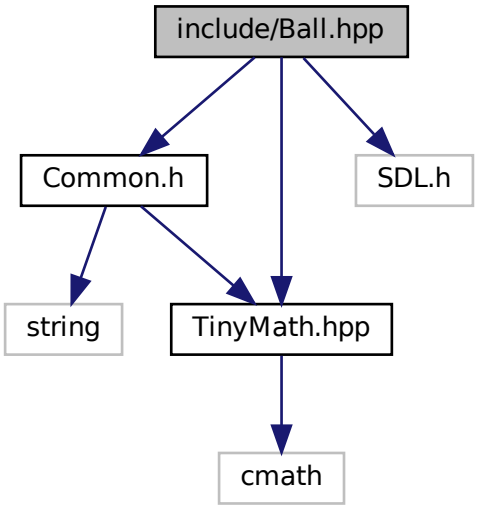
5.7 include/Ball.hpp File Reference

Ball object class.

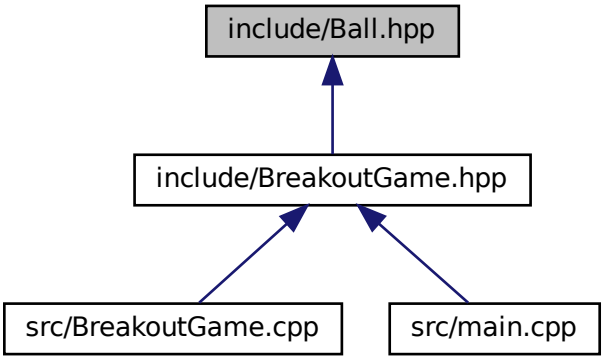
```
#include "Common.h"
#include "TinyMath.hpp"
```

```
#include <SDL.h>
```

Include dependency graph for Ball.hpp:



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



Classes

- class Ball

  Represent the ball object in the break out game. Record velocity, position etc. info of the ball.

5.7.1 Detailed Description

Ball object class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 21:54:19 -08:00

Copyright

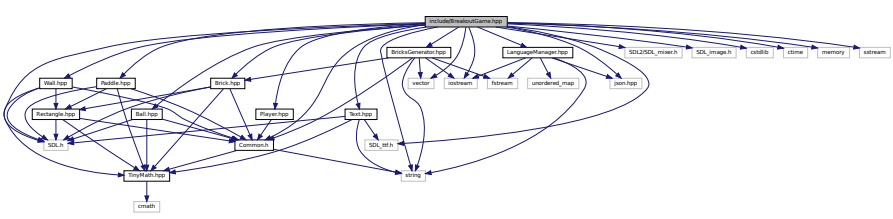
Copyright (c) 2021

5.8 include/BreakoutGame.hpp File Reference

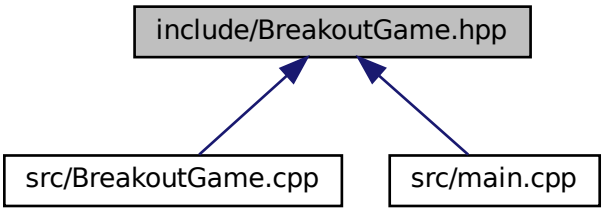
BreakoutGame main Class Header.

```
#include <SDL.h>
#include <SDL2/SDL_mixer.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include <cstdlib>
#include <ctime>
#include <iostream>
#include <memory>
#include <sstream>
#include <string>
#include <vector>
#include "Ball.hpp"
#include "Brick.hpp"
#include "BricksGenerator.hpp"
#include "Common.h"
#include "LanguageManager.hpp"
#include "Paddle.hpp"
#include "Player.hpp"
#include "Text.hpp"
#include "Wall.hpp"
#include "json.hpp"
```

Include dependency graph for BreakoutGame.hpp:



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



Classes

- class BreakoutGame

  Breakout Game main Class, where main logic located.

5.8.1 Detailed Description

BreakoutGame main Class Header.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 21:55:07 -08:00

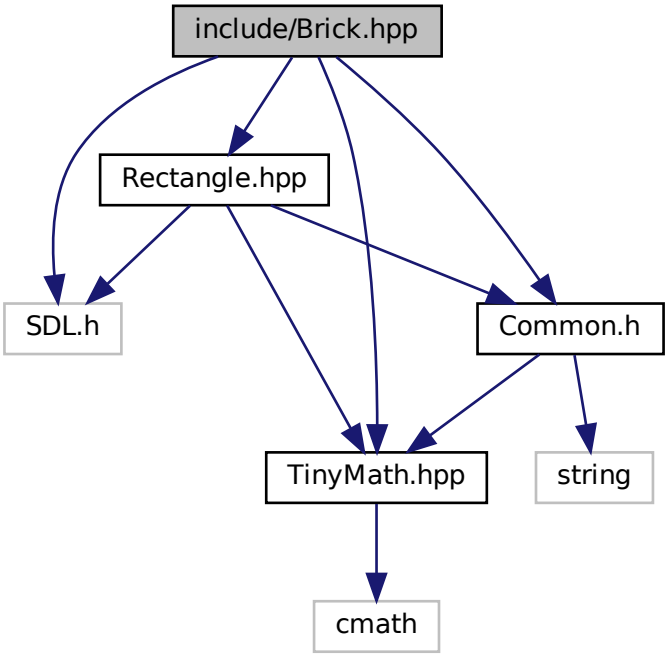
Copyright

5.9 include/Brick.hpp File Reference

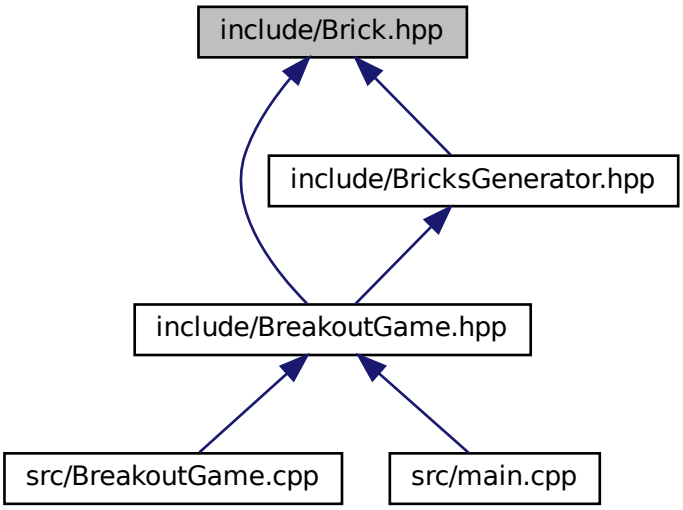
Brick object class.

```
#include <SDL.h>
#include "Common.h"
#include "Rectangle.hpp"
#include "TinyMath.hpp"
```

Include dependency graph for Brick.hpp:



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



Classes

- class Brick

  Brick object Class.

5.9.1 Detailed Description

Brick object class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 22:44:19 -08:00

Copyright

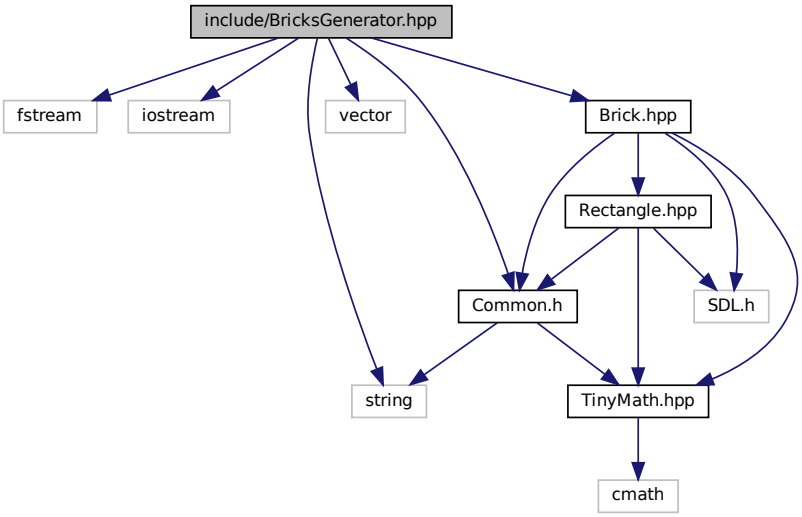
Copyright (c) 2021

5.10 include/BricksGenerator.hpp File Reference

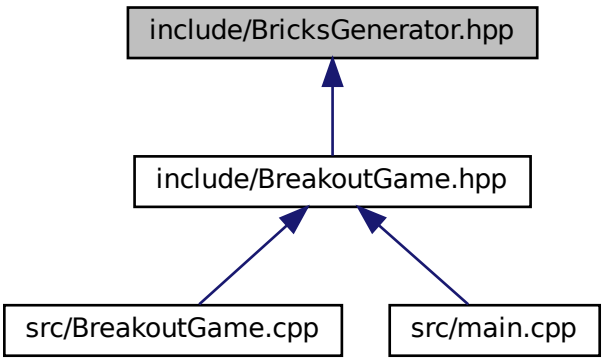
Header file for BricksGenerator.

```
#include <fstream>
#include <iostream>
#include <string>
#include <vector>
#include "Brick.hpp"
#include "Common.h"
```

Include dependency graph for BricksGenerator.hpp:



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



Classes

- struct LevelData

  Struct to store brick arrangement data of one level.
- class BricksGenerator

  Helper class to load and generate each level bricks arrangement.

5.10.1 Detailed Description

Header file for BricksGenerator.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 22:47:39 -08:00

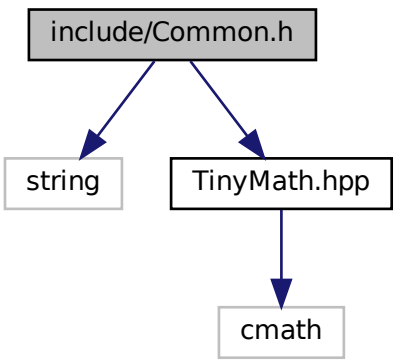
Copyright

5.11 include/Common.h File Reference

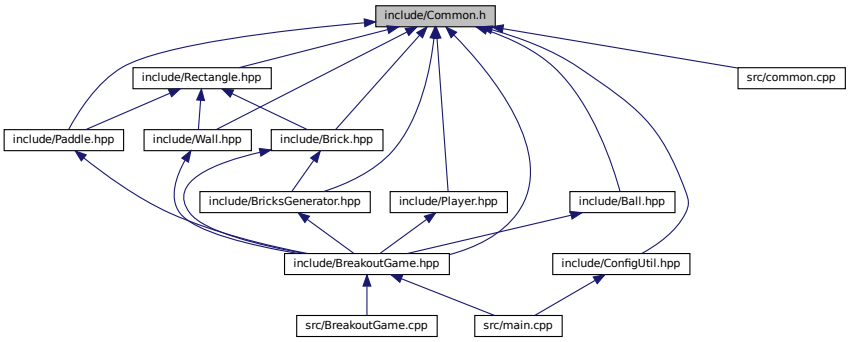
Header file to declare all global variables.

```
#include <string>
#include "TinyMath.hpp"
```

Include dependency graph for Common.h:



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



Classes

- struct Contact

  Collision info struct.

Enumerations

- enum Buttons { PaddleLeft = 0, PaddleRight }

  Indicate paddle moving state.
- enum GameState {
  Initializing, Running, PauseNormal, PauseWin,
  PauseClearGame, PauseLoseGame, PauseLoseLife }

  Game state Enum.
- enum CollisionType {
  CollisionType::None, CollisionType::Top, CollisionType::Middle, CollisionType::Bottom,
  CollisionType::Left, CollisionType::Right }

  Collision direction.

Variables

- int WINDOW_WIDTH
- int WINDOW_HEIGHT
- int GAME_SCENE_WIDTH
- int GAME_SCENE_LEFT
- int GAME_SCENE_RIGHT
- int WALL_WIDTH
- float PADDLE_SPEED
- int PADDLE_WIDTH
- int PADDLE_HEIGHT
- int PADDLE_DISTANCE_FROM_BOTTOM
- float BALL_START_DEGREE
- float BALL_SPEED
- int BALL_WIDTH
- int BALL_HEIGHT
- int BRICK_START_HEIGHT

- int BRICK_WIDTH
- int BRICK_HEIGHT
- int BRICK_INTERVAL
- int BRICK_ROW
- int BRICK_COLUMN
- int BRICK_DEFAULT_SCORE
- int PLAYER_DEFAULT_LIFE_NUM
- int DEFAULT_LEVEL

  Default begin level.
- int DEFAULT_FONT_SIZE
- int MENU_FONT_SIZE
- int SCREEN_FPS_60
- int SCREEN_TICKS_PER_FRAME_60

  Milliseconds between each frame update when fps is 60.
- int TICKS_PER_UPDATE

  Milliseconds between each update.

5.11.1 Detailed Description

Header file to declare all global variables.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 22:52:41 -08:00

Copyright

Copyright (c) 2021

5.11.2 Enumeration Type Documentation

5.11.2.1 Buttons

```
enum Buttons
```

Indicate paddle moving state.

Enumerator

PaddleLeft	Paddle is moving left.
PaddleRight	Paddle is moving right.

5.11.2.2 CollisionType

```
enum CollisionType  [strong]
```

Collision direction.

Enumerator

None	No collision happened.
Top	Ball is now moving towards top.
Middle	Ball hit the middle of the paddle.
Bottom	Ball is now moving towards bottom.
Left	Ball is now moving towards left.
Right	Ball is now moving towards right.

5.11.2.3 GameState

```
enum GameState
```

Game state Enum.

Enumerator

Initializing	Game is under initialization.
Running	Game is running.
PauseNormal	Game paused normally.
PauseWin	Game paused because player just finish one level.
PauseClearGame	Game paused because player finished all levels.
PauseLoseGame	Game paused because player lose all lives.
PauseLoseLife	Game paused because player lose one life.

5.11.3 Variable Documentation

5.11.3.1 BALL_HEIGHT

```
int BALL_HEIGHT
```

5.11.3.2 BALL_SPEED

```
float BALL_SPEED
```

5.11.3.3 BALL_START_DEGREE

```
float BALL_START_DEGREE
```

5.11.3.4 BALL_WIDTH

```
int BALL_WIDTH
```

5.11.3.5 BRICK_COLUMN

```
int BRICK_COLUMN
```

5.11.3.6 BRICK_DEFAULT_SCORE

```
int BRICK_DEFAULT_SCORE
```

5.11.3.7 BRICK_HEIGHT

```
int BRICK_HEIGHT
```

5.11.3.8 BRICK_INTERVAL

```
int BRICK_INTERVAL
```

5.11.3.9 BRICK_ROW

```
int BRICK_ROW
```

5.11.3.10 BRICK_START_HEIGHT

```
int BRICK_START_HEIGHT
```

5.11.3.11 BRICK_WIDTH

```
int BRICK_WIDTH
```

5.11.3.12 DEFAULT_FONT_SIZE

```
int DEFAULT_FONT_SIZE
```

5.11.3.13 DEFAULT_LEVEL

```
int DEFAULT_LEVEL
```

Default begin level.

5.11.3.14 GAME_SCENE_LEFT

```
int GAME_SCENE_LEFT
```

5.11.3.15 GAME_SCENE_RIGHT

```
int GAME_SCENE_RIGHT
```

5.11.3.16 GAME_SCENE_WIDTH

```
int GAME_SCENE_WIDTH
```

5.11.3.17 MENU_FONT_SIZE

```
int MENU_FONT_SIZE
```

5.11.3.18 PADDLE_DISTANCE_FROM_BOTTOM

```
int PADDLE_DISTANCE_FROM_BOTTOM
```

5.11.3.19 PADDLE_HEIGHT

```
int PADDLE_HEIGHT
```

5.11.3.20 PADDLE_SPEED

```
float PADDLE_SPEED
```

5.11.3.21 PADDLE_WIDTH

```
int PADDLE_WIDTH
```

5.11.3.22 PLAYER_DEFAULT_LIFE_NUM

```
int PLAYER_DEFAULT_LIFE_NUM
```

5.11.3.23 SCREEN_FPS_60

```
int SCREEN_FPS_60
```

5.11.3.24 SCREEN_TICKS_PER_FRAME_60

```
int SCREEN_TICKS_PER_FRAME_60
```

Milliseconds between each frame update when fps is 60.

5.11.3.25 TICKS_PER_UPDATE

```
int TICKS_PER_UPDATE
```

Milliseconds between each update.

5.11.3.26 WALL_WIDTH

```
int WALL_WIDTH
```

5.11.3.27 WINDOW_HEIGHT

```
int WINDOW_HEIGHT
```

5.11.3.28 WINDOW_WIDTH

```
int WINDOW_WIDTH
```

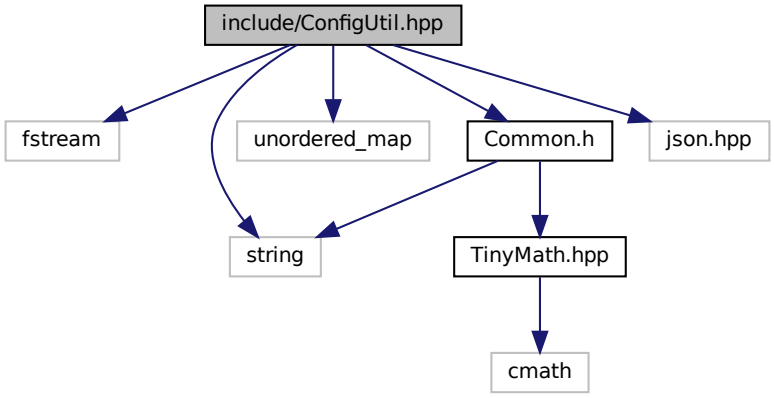
5.12 include/ConfigUtil.hpp File Reference

Helper class to load config from file.

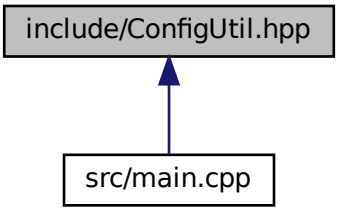
```
#include <fstream>
#include <string>
#include <unordered_map>
#include "Common.h"
```

```
#include "json.hpp"
```

Include dependency graph for ConfigUtil.hpp:



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



Classes

- class ConfigUtil

  Help class which use nlohmann::json library to process json format config file.

5.12.1 Detailed Description

Helper class to load config from file.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:05:14 -08:00

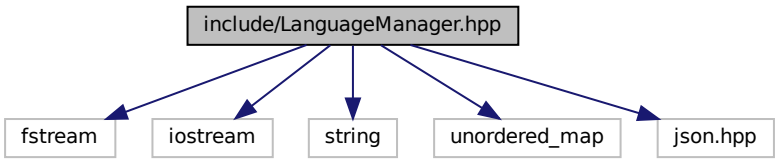
Copyright

5.13 include/LanguageManager.hpp File Reference

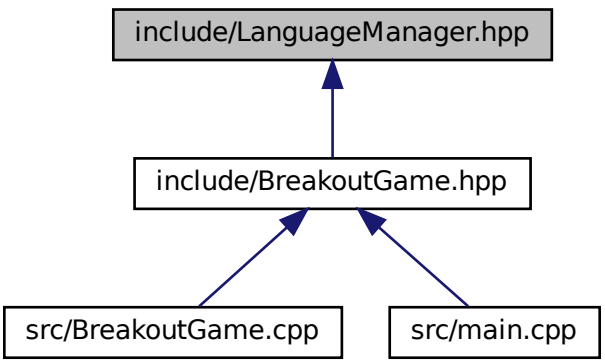
Helper class LanguageSelector.

```
#include <fstream>
#include <iostream>
#include <string>
#include <unordered_map>
#include "json.hpp"
```

Include dependency graph for LanguageManager.hpp:



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



Classes

- struct Language

  Struct to store all key-value pair of game texts.
- class LanguageSelector

  Help class to read, load and manage multi language text data.

5.13.1 Detailed Description

Helper class LanguageSelector.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:15:42 -08:00

Copyright

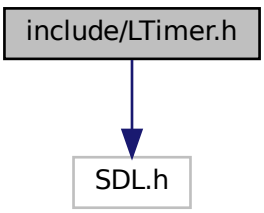
Copyright (c) 2021

5.14 include/LTimer.h File Reference

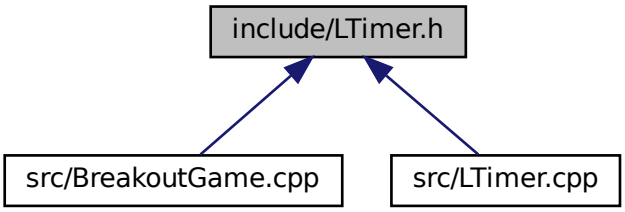
Simple Timer Class.

```
#include <SDL.h>
```

Include dependency graph for LTimer.h:



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



Classes

- class LTimer

  The application time based timer.

5.14.1 Detailed Description

Simple Timer Class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:22:12 -08:00

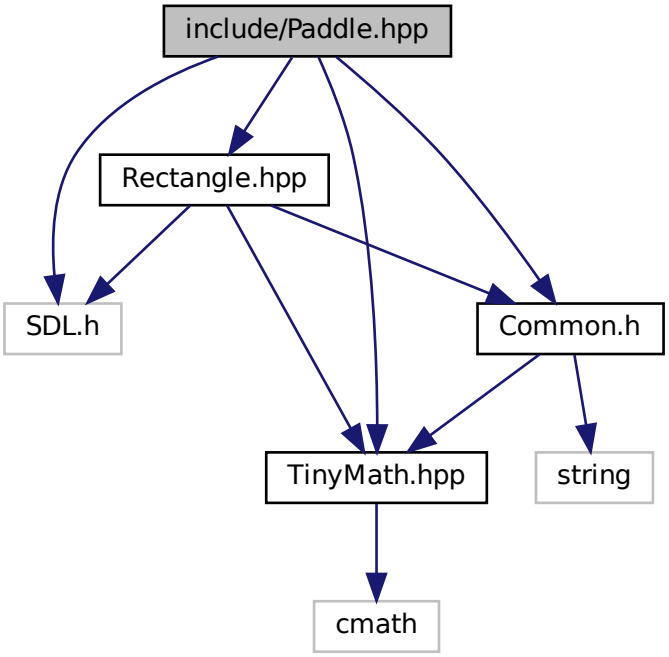
Copyright

5.15 include/Paddle.hpp File Reference

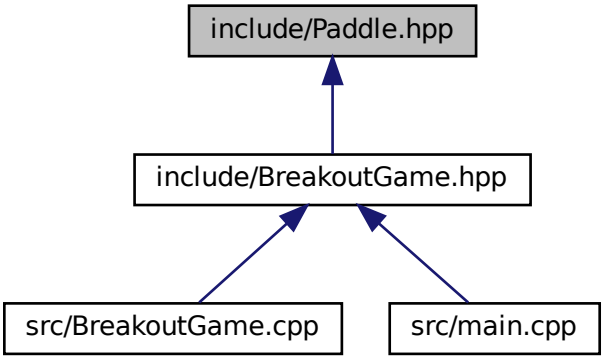
Paddle object Class.

```
#include <SDL.h>
#include "Common.h"
#include "Rectangle.hpp"
#include "TinyMath.hpp"
```

Include dependency graph for Paddle.hpp:



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



Classes

- class Paddle

  Paddle object class.

5.15.1 Detailed Description

Paddle object Class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-23 00:01:56 -08:00

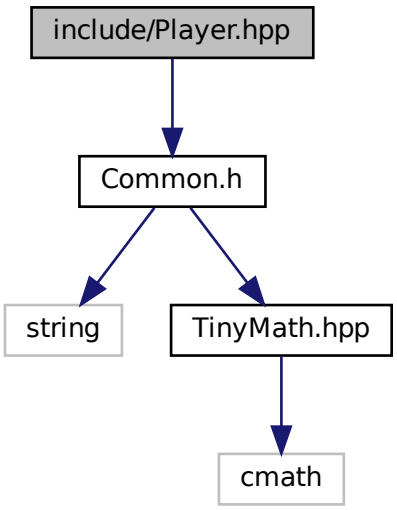
Copyright

Copyright (c) 2021

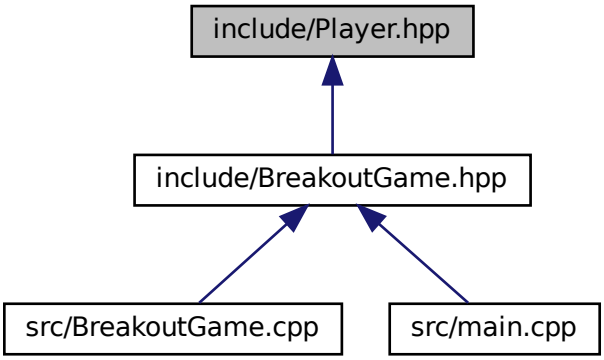
5.16 include/Player.hpp File Reference

```
#include "Common.h"
```

Include dependency graph for Player.hpp:



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



Classes

- class Player

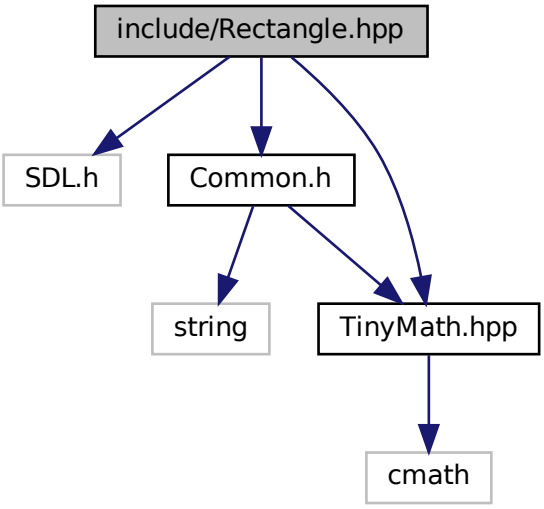
  Player class, store info of the player.

5.17 include/Rectangle.hpp File Reference

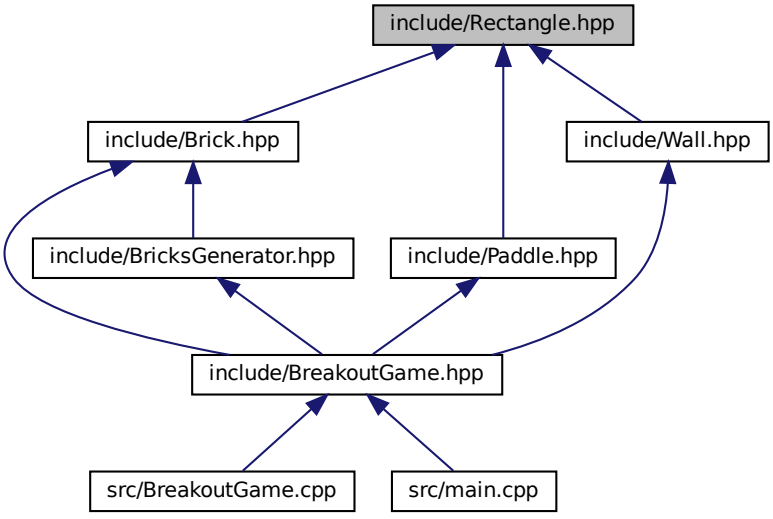
A rectangle class.

```
#include <SDL.h>
#include "Common.h"
#include "TinyMath.hpp"
```

Include dependency graph for Rectangle.hpp:



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



Classes

- class Rectangle

  A simple rectange class to represnet a rectangle on screen.

5.17.1 Detailed Description

A rectangle class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-23 00:07:43 -08:00

Copyright

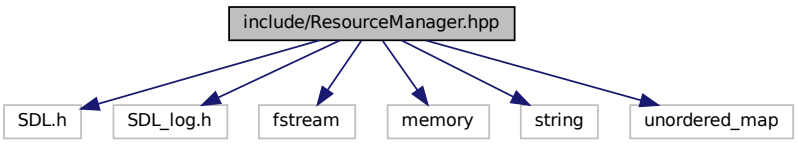
Copyright (c) 2021

5.18 include/ResourceManager.hpp File Reference

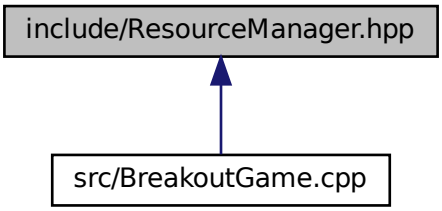
ResourceManager class header.

```
#include <SDL.h>
#include <SDL_log.h>
#include <fstream>
#include <memory>
#include <string>
#include <unordered_map>
```

Include dependency graph for ResourceManager.hpp:



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



Classes

- class ResourceManager

  Singleton ResourceManager manage resources which loaded from files.

Typedefs

- typedef std::string ResName
- typedef std::string ResFilePath

5.18.1 Detailed Description

ResourceManager class header.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:35:03 -08:00

Copyright

5.18.2 Typedef Documentation

5.18.2.1 ResFilePath

```
typedef std::string ResFilePath
```

5.18.2.2 ResName

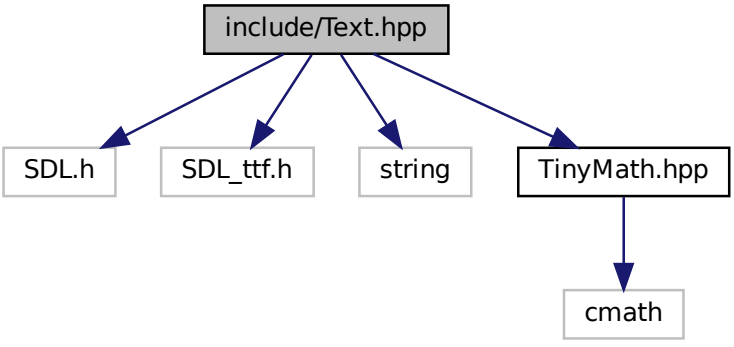
```
typedef std::string ResName
```

5.19 include/Text.hpp File Reference

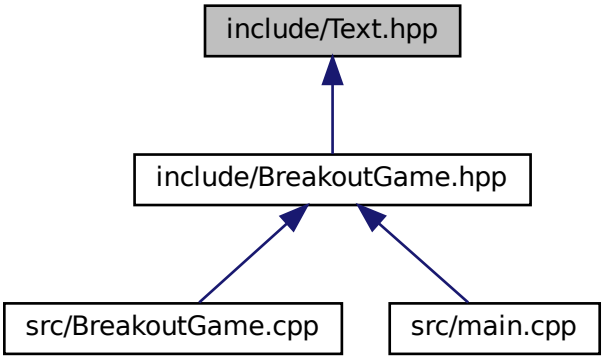
Text object class.

```
#include <SDL.h>
#include <SDL_ttf.h>
#include <string>
#include "TinyMath.hpp"
```

Include dependency graph for Text.hpp:



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



Classes

- class Text

  A text wrapper class.

5.19.1 Detailed Description

Text object class.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:53:09 -08:00

Copyright

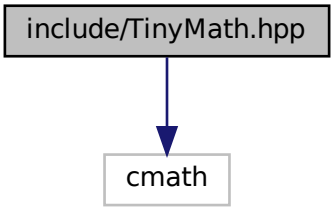
Copyright (c) 2021

5.20 include/TinyMath.hpp File Reference

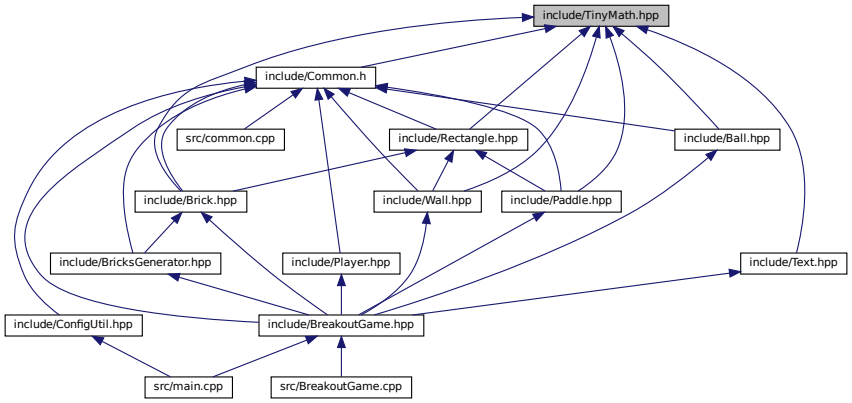
A tiny math library.

```
#include <cmath>
```

Include dependency graph for TinyMath.hpp:



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



Classes

- struct Vector3D
- struct Vector2D
- struct Matrix3D

  Matrix 3D represents 3x3 matrices in Math.

Enumerations

- enum Direction { UP, RIGHT, DOWN, LEFT }

  Vector main direction.

Functions

- float Dot (const Vector3D &a, const Vector3D &b)
  Compute the dot product of a Vector3D.
- Vector3D operator∗ (const Vector3D &v, float s)
  Multiplication of a vector by a scalar values.
- Vector3D operator/ (const Vector3D &v, float s)
  Division of a vector by a scalar value.
- Vector3D operator- (const Vector3D &v)
- float Magnitude (const Vector3D &v)
  Return the magnitude of a vector.
- Vector3D operator+ (const Vector3D &a, const Vector3D &b)
  Add two vectors together.
- Vector3D operator- (const Vector3D &a, const Vector3D &b)
  Subtract two vectors.
- Vector3D Project (const Vector3D &a, const Vector3D &b)
  Vector Projection.
- Vector3D Normalize (const Vector3D &v)
- Vector3D CrossProduct (const Vector3D &a, const Vector3D &b)
- Direction VectorDirectionSDL (Vector2D target)
  Get one 2D vector's main direction.
- Vector2D getUnitVectorFromDegree (float degree)
  Get the Unit Vector From given Degree number.
- Matrix3D operator∗ (const Matrix3D &A, const Matrix3D &B)
  Matrix Multiplication.
- Vector3D operator∗ (const Matrix3D &M, const Vector3D &v)
  Matrix multiply by a vector.

5.20.1 Detailed Description

A tiny math library.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:52:51 -08:00

Copyright

5.20.2 Enumeration Type Documentation

5.20.2.1 Direction

```
enum Direction
```

Vector main direction.

Enumerator

UP	
RIGHT	
DOWN	
LEFT	

5.20.3 Function Documentation

5.20.3.1 CrossProduct()

```
Vector3D CrossProduct (
            const Vector3D & a,
            const Vector3D & b )  [inline]
```

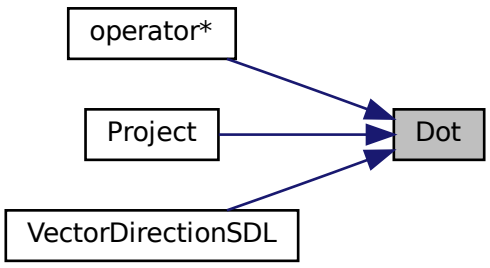
a x b (read: 'a crossed b') Produces a new vector perpendicular to a and b. (So long as a and b are not parallel which returns zero vector)

5.20.3.2 Dot()

```
float Dot (
            const Vector3D & a,
            const Vector3D & b )  [inline]
```

Compute the dot product of a Vector3D.

Here is the caller graph for this function:



5.20.3.3 getUnitVectorFromDegree()

```
Vector2D getUnitVectorFromDegree (
            float degree )  [inline]
```

Get the Unit Vector From given Degree number.

The direction of degree is same to the polar coordinate system

Parameters

degree	Given degree number

Returns

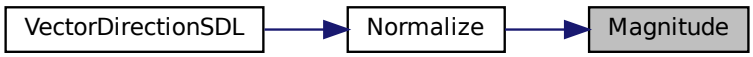
Vector2D The result Unit Vector

5.20.3.4 Magnitude()

```
float Magnitude (
            const Vector3D & v )  [inline]
```

Return the magnitude of a vector.

Here is the caller graph for this function:

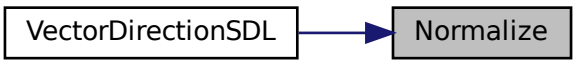


5.20.3.5 Normalize()

```
Vector3D Normalize (
            const Vector3D & v )  [inline]
```

Set a vectors magnitude to 1 Note: This is NOT generating a normal vector Here is the call graph for this function:

Here is the caller graph for this function:

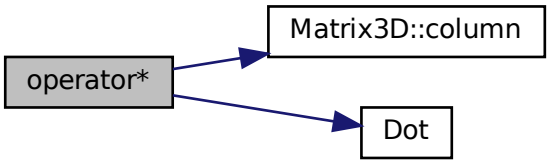


5.20.3.6 operator*() [1/3]

```
Matrix3D operator* (
            const Matrix3D & A,
            const Matrix3D & B )  [inline]
```

Matrix Multiplication.

Here is the call graph for this function:

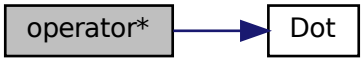


5.20.3.7 operator*() [2/3]

```
Vector3D operator* (
            const Matrix3D & M,
            const Vector3D & v )  [inline]
```

Matrix multiply by a vector.

Here is the call graph for this function:



5.20.3.8 operator*() [3/3]

```
Vector3D operator* (
            const Vector3D & v,
            float s )  [inline]
```

Multiplication of a vector by a scalar values.

5.20.3.9 operator+()

```
Vector3D operator+ (
            const Vector3D & a,
            const Vector3D & b )  [inline]
```

Add two vectors together.

5.20.3.10 operator-() [1/2]

```
Vector3D operator- (
            const Vector3D & a,
            const Vector3D & b )  [inline]
```

Subtract two vectors.

5.20.3.11 operator-() [2/2]

```
Vector3D operator- (
            const Vector3D & v )  [inline]
```

Negation of a vector Use Case: Sometimes it is handy to apply a force in an opposite direction

5.20.3.12 operator/()

```
Vector3D operator/ (
            const Vector3D & v,
            float s )  [inline]
```

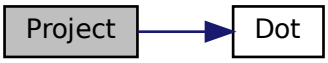
Division of a vector by a scalar value.

5.20.3.13 Project()

```
Vector3D Project (
            const Vector3D & a,
            const Vector3D & b )  [inline]
```

Vector Projection.

Here is the call graph for this function:



5.20.3.14 VectorDirectionSDL()

```
Direction VectorDirectionSDL (
            Vector2D target )  [inline]
```

Get one 2D vector's main direction.

Main direction is the nearest direction close to the given direction, which is the direction which has smallest included angle with the given vector among the 4 axis direction,

Parameters

target	Given 2D vector

Returns

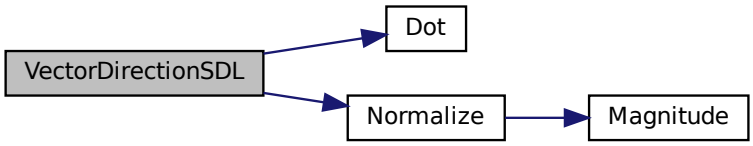
Direction Given 2D vector's main direction

up

right

down

leftHere is the call graph for this function:

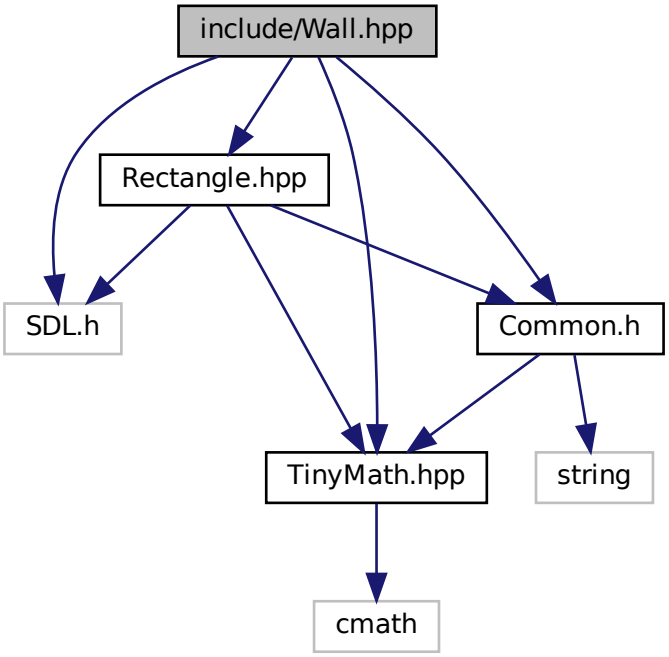


5.21 include/Wall.hpp File Reference

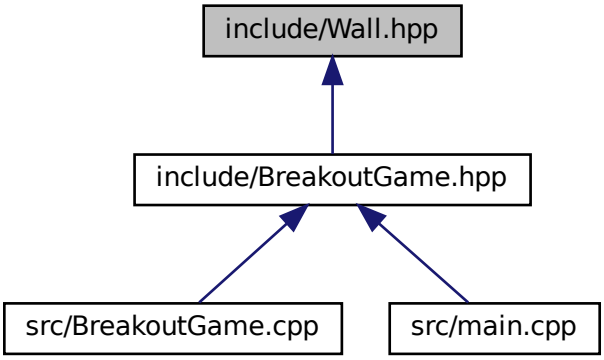
Wall object class header.

```
#include <SDL.h>
#include "Common.h"
#include "Rectangle.hpp"
#include "TinyMath.hpp"
```

Include dependency graph for Wall.hpp:



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



Classes

- class Wall

  Wall object Class.

5.21.1 Detailed Description

Wall object class header.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:26:56 -08:00

Copyright

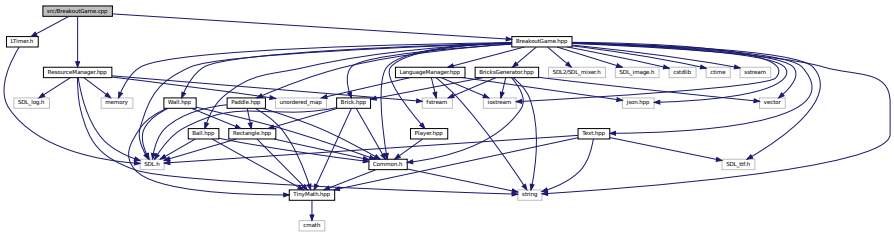
Copyright (c) 2021

5.22 src/BreakoutGame.cpp File Reference

BreakoutGame Class Implementation.

```
#include "BreakoutGame.hpp"
#include "LTimer.h"
#include "ResourceManager.hpp"
```

Include dependency graph for BreakoutGame.cpp:



5.22.1 Detailed Description

BreakoutGame Class Implementation.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 22:43:39 -08:00

Copyright

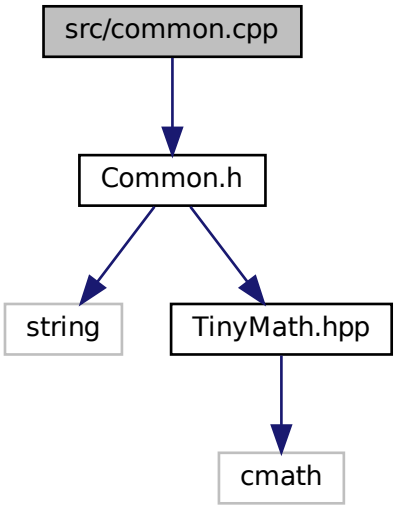
Copyright (c) 2021

5.23 src/common.cpp File Reference

Definition of all global variables.

```
#include "Common.h"
```

Include dependency graph for common.cpp:



Variables

- int WINDOW_WIDTH
- int WINDOW_HEIGHT
- int GAME_SCENE_WIDTH
- int GAME_SCENE_LEFT
- int GAME_SCENE_RIGHT
- int WALL_WIDTH
- float PADDLE_SPEED
- int PADDLE_WIDTH
- int PADDLE_HEIGHT
- int PADDLE_DISTANCE_FROM_BOTTOM
- float BALL_START_DEGREE
- float BALL_SPEED
- int BALL_WIDTH
- int BALL_HEIGHT
- int BRICK_START_HEIGHT
- int BRICK_WIDTH
- int BRICK_HEIGHT
- int BRICK_INTERVAL
- int BRICK_ROW
- int BRICK_COLUMN
- int BRICK_DEFAULT_SCORE
- int PLAYER_DEFAULT_LIFE_NUM
- int DEFAULT_LEVEL

  Default begin level.
- int DEFAULT_FONT_SIZE
- int MENU_FONT_SIZE

- int SCREEN_FPS_60
- int SCREEN_TICKS_PER_FRAME_60

  Milliseconds between each frame update when fps is 60.
- int TICKS_PER_UPDATE

  Milliseconds between each update.

5.23.1 Detailed Description

Definition of all global variables.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:03:50 -08:00

Copyright

Copyright (c) 2021

5.23.2 Variable Documentation

5.23.2.1 BALL_HEIGHT

```
int BALL_HEIGHT
```

5.23.2.2 BALL_SPEED

```
float BALL_SPEED
```

5.23.2.3 BALL_START_DEGREE

```
float BALL_START_DEGREE
```

5.23.2.4 BALL_WIDTH

```
int BALL_WIDTH
```

5.23.2.5 BRICK_COLUMN

```
int BRICK_COLUMN
```

5.23.2.6 BRICK_DEFAULT_SCORE

```
int BRICK_DEFAULT_SCORE
```

5.23.2.7 BRICK_HEIGHT

```
int BRICK_HEIGHT
```

5.23.2.8 BRICK_INTERVAL

```
int BRICK_INTERVAL
```

5.23.2.9 BRICK_ROW

```
int BRICK_ROW
```

5.23.2.10 BRICK_START_HEIGHT

```
int BRICK_START_HEIGHT
```

5.23.2.11 BRICK_WIDTH

```
int BRICK_WIDTH
```

5.23.2.12 DEFAULT_FONT_SIZE

```
int DEFAULT_FONT_SIZE
```

5.23.2.13 DEFAULT_LEVEL

```
int DEFAULT_LEVEL
```

Default begin level.

5.23.2.14 GAME_SCENE_LEFT

```
int GAME_SCENE_LEFT
```

5.23.2.15 GAME_SCENE_RIGHT

```
int GAME_SCENE_RIGHT
```

5.23.2.16 GAME_SCENE_WIDTH

```
int GAME_SCENE_WIDTH
```

5.23.2.17 MENU_FONT_SIZE

```
int MENU_FONT_SIZE
```

5.23.2.18 PADDLE_DISTANCE_FROM_BOTTOM

```
int PADDLE_DISTANCE_FROM_BOTTOM
```

5.23.2.19 PADDLE_HEIGHT

```
int PADDLE_HEIGHT
```

5.23.2.20 PADDLE_SPEED

```
float PADDLE_SPEED
```

5.23.2.21 PADDLE_WIDTH

```
int PADDLE_WIDTH
```

5.23.2.22 PLAYER_DEFAULT_LIFE_NUM

```
int PLAYER_DEFAULT_LIFE_NUM
```

5.23.2.23 SCREEN_FPS_60

```
int SCREEN_FPS_60
```

5.23.2.24 SCREEN_TICKS_PER_FRAME_60

```
int SCREEN_TICKS_PER_FRAME_60
```

Milliseconds between each frame update when fps is 60.

5.23.2.25 TICKS_PER_UPDATE

```
int TICKS_PER_UPDATE
```

Milliseconds between each update.

5.23.2.26 WALL_WIDTH

```
int WALL_WIDTH
```

5.23.2.27 WINDOW_HEIGHT

```
int WINDOW_HEIGHT
```

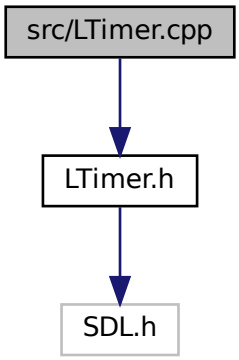
5.23.2.28 WINDOW_WIDTH

```
int WINDOW_WIDTH
```

5.24 src/LTimer.cpp File Reference

LTimer Class implementation.

```
#include "LTimer.h"
```
Include dependency graph for LTimer.cpp:



Functions

- bool isStarted ()

5.24.1 Detailed Description

LTimer Class implementation.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:24:41 -08:00

Copyright

5.24.2 Function Documentation

5.24.2.1 isStarted()

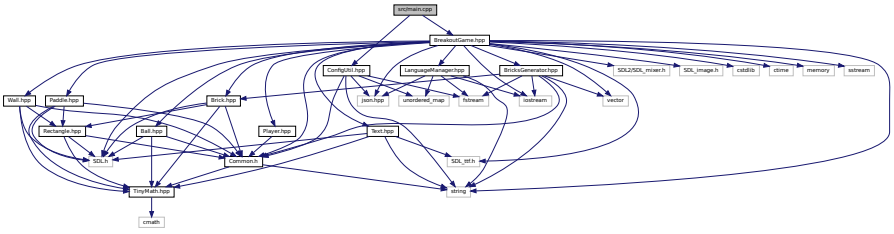
```
bool isStarted ( )
```

5.25 src/main.cpp File Reference

Enter point of the game.

```
#include "BreakoutGame.hpp"
#include "ConfigUtil.hpp"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char **argv)
  Main function of the game.

5.25.1 Detailed Description

Enter point of the game.

Author

Yuxiang Cao (cao.yux@northeastern.edu)

Version

1.0.0

Date

2021-02-22 23:25:35 -08:00

Copyright

Copyright (c) 2021

5.25.2 Function Documentation

5.25.2.1 main()

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

Main function of the game.

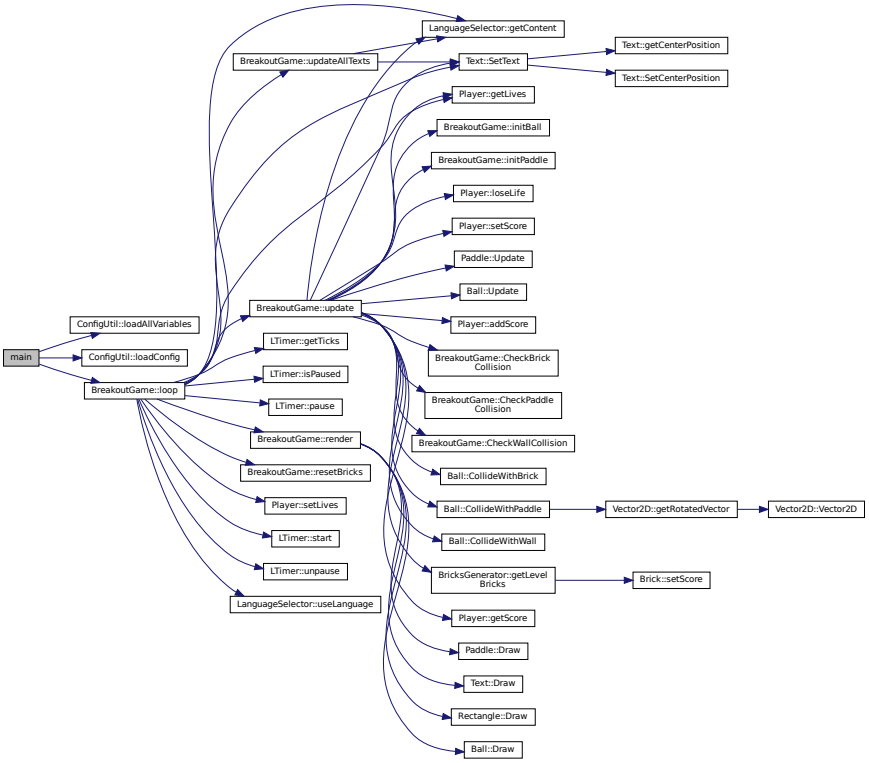
Parameters

argc	Number of argument
argv	Argument string array

Returns

int Program exit state

Here is the call graph for this function:



Index

~BreakoutGame
    BreakoutGame, 15
~ResourceManager
    ResourceManager, 66

active
    Brick, 35
AddResource
    ResourceManager, 67
addScore
    Player, 59
allLevels
    BricksGenerator, 38
assets/fonts/LICENSE_GPLv3.txt, 91

backgroundMusic
    BreakoutGame, 26
Ball, 7
    Ball, 8
    CollideWithBrick, 9
    CollideWithPaddle, 9
    CollideWithWall, 10
    Draw, 10
    position, 11
    rect, 11
    Update, 11
    velocity, 11
ball
    BreakoutGame, 26
BALL_HEIGHT
    common.cpp, 141
    Common.h, 113
BALL_SPEED
    common.cpp, 141
    Common.h, 114
BALL_START_DEGREE
    common.cpp, 141
    Common.h, 114
BALL_WIDTH
    common.cpp, 141
    Common.h, 114
Bottom
    Common.h, 113
BreakoutGame, 12
    ~BreakoutGame, 15
    backgroundMusic, 26
    ball, 26
    BreakoutGame, 15
    brickHitSound, 26
    bricks, 27
    bricksGenerator, 27
    buttons, 27
    CheckBrickCollision, 16
    CheckPaddleCollision, 16
    CheckWallCollision, 17
    configs, 27
    contentFont_, 27
    gameState, 27
    getSDLRenderer, 17
    getSDLWindow, 18
    gRenderer, 28
    gWindow, 28
    initBall, 18
    initGameObjects, 18
    initPaddle, 19
    initSDLSystems, 20
    languageSelector, 28
    level, 28
    levelNum, 28
    levelText, 28
    livesNum, 29
    livesText, 29
    loadLanguages, 20
    loadLevels, 21
    loadResources, 22
    loop, 22
    loseLifeSound, 29
    loseSound, 29
    maxLevel, 29
    menuFont_, 29
    notificationText, 30
    paddle, 30
    paddleHitSound, 30
    player, 30
    render, 23
    resetBricks, 24
    restBricks, 30
    scoreNum, 30
    scoreText, 31
    screenHeight, 31
    screenWidth, 31
    update, 24
    updateAllTexts, 25
    wallHitSound, 31
    wallLeft, 31
    wallRight, 31
    winSound, 32
Brick, 32
    active, 35

Brick, 34
Draw, 34
getScore, 34
isActive, 34
score, 36
setActive, 35
setScore, 35
BRICK_COLUMN
common.cpp, 142
Common.h, 114
BRICK_DEFAULT_SCORE
common.cpp, 142
Common.h, 114
BRICK_HEIGHT
common.cpp, 142
Common.h, 114
BRICK_INTERVAL
common.cpp, 142
Common.h, 114
BRICK_ROW
common.cpp, 142
Common.h, 114
BRICK_START_HEIGHT
common.cpp, 142
Common.h, 115
BRICK_WIDTH
common.cpp, 142
Common.h, 115
brickHitSound
BreakoutGame, 26
bricks
BreakoutGame, 27
BricksGenerator, 36
allLevels, 38
getLevelBricks, 36
getMaxLevel, 37
loadOneLevelData, 37
bricksGenerator
BreakoutGame, 27
Buttons
Common.h, 112
buttons
BreakoutGame, 27

cfgFilePath_
ResourceManager, 73
CheckBrickCollision
BreakoutGame, 16
CheckPaddleCollision
BreakoutGame, 16
CheckWallCollision
BreakoutGame, 17
CollideWithBrick
Ball, 9
CollideWithPaddle
Ball, 9
CollideWithWall
Ball, 10
CollisionType
Common.h, 113
color
Text, 80
column
Matrix3D, 52
common.cpp
BALL_HEIGHT, 141
BALL_SPEED, 141
BALL_START_DEGREE, 141
BALL_WIDTH, 141
BRICK_COLUMN, 142
BRICK_DEFAULT_SCORE, 142
BRICK_HEIGHT, 142
BRICK_INTERVAL, 142
BRICK_ROW, 142
BRICK_START_HEIGHT, 142
BRICK_WIDTH, 142
DEFAULT_FONT_SIZE, 142
DEFAULT_LEVEL, 143
GAME_SCENE_LEFT, 143
GAME_SCENE_RIGHT, 143
GAME_SCENE_WIDTH, 143
MENU_FONT_SIZE, 143
PADDLE_DISTANCE_FROM_BOTTOM, 143
PADDLE_HEIGHT, 143
PADDLE_SPEED, 144
PADDLE_WIDTH, 144
PLAYER_DEFAULT_LIFE_NUM, 144
SCREEN_FPS_60, 144
SCREEN_TICKS_PER_FRAME_60, 144
TICKS_PER_UPDATE, 144
WALL_WIDTH, 144
WINDOW_HEIGHT, 145
WINDOW_WIDTH, 145
Common.h
BALL_HEIGHT, 113
BALL_SPEED, 114
BALL_START_DEGREE, 114
BALL_WIDTH, 114
Bottom, 113
BRICK_COLUMN, 114
BRICK_DEFAULT_SCORE, 114
BRICK_HEIGHT, 114
BRICK_INTERVAL, 114
BRICK_ROW, 114
BRICK_START_HEIGHT, 115
BRICK_WIDTH, 115
Buttons, 112
CollisionType, 113
DEFAULT_FONT_SIZE, 115
DEFAULT_LEVEL, 115
GAME_SCENE_LEFT, 115
GAME_SCENE_RIGHT, 115
GAME_SCENE_WIDTH, 115
GameState, 113
Initializing, 113
Left, 113
MENU_FONT_SIZE, 116

Middle, 113
None, 113
PADDLE_DISTANCE_FROM_BOTTOM, 116
PADDLE_HEIGHT, 116
PADDLE_SPEED, 116
PADDLE_WIDTH, 116
PaddleLeft, 113
PaddleRight, 113
PauseClearGame, 113
PauseLoseGame, 113
PauseLoseLife, 113
PauseNormal, 113
PauseWin, 113
PLAYER_DEFAULT_LIFE_NUM, 116
Right, 113
Running, 113
SCREEN_FPS_60, 116
SCREEN_TICKS_PER_FRAME_60, 116
TICKS_PER_UPDATE, 117
Top, 113
WALL_WIDTH, 117
WINDOW_HEIGHT, 117
WINDOW_WIDTH, 117
config/levels/1.txt, 103
config/levels/2.txt, 103
config/levels/3.txt, 103
config/levels/4.txt, 103
config/levels/5.txt, 103
configs
BreakoutGame, 27
ConfigUtil, 38
loadAllVariables, 39
loadConfig, 39
Contact, 40
penetration, 41
type, 41
contentFont_
BreakoutGame, 27
Copyright
LICENSE_GPLv3.txt, 96
CrossProduct
TinyMath.hpp, 132
currentLanguage
LanguageSelector, 45

data
Language, 41
LevelData, 46
DEFAULT_FONT_SIZE
common.cpp, 142
Common.h, 115
DEFAULT_LEVEL
common.cpp, 143
Common.h, 115
Direction
TinyMath.hpp, 131
Dot
TinyMath.hpp, 132
DOWN
TinyMath.hpp, 132
Draw
Ball, 10
Brick, 34
Paddle, 56
Rectangle, 63
Text, 76

fileMap_
ResourceManager, 73
font
Text, 80

GAME_SCENE_LEFT
common.cpp, 143
Common.h, 115
GAME_SCENE_RIGHT
common.cpp, 143
Common.h, 115
GAME_SCENE_WIDTH
common.cpp, 143
Common.h, 115
GameState
Common.h, 113
gameState
BreakoutGame, 27
getCenterPosition
Text, 76
getContent
LanguageSelector, 42
getHeight
Text, 76
getInstance
ResourceManager, 68
getLevelBricks
BricksGenerator, 36
getLives
Player, 59
getMaxLevel
BricksGenerator, 37
getRotatedVector
Vector2D, 83
getScore
Brick, 34
Player, 59
getSDLRenderer
BreakoutGame, 17
getSDLWindow
BreakoutGame, 18
getTicks
LTimer, 47
getUnitVectorFromDegree
TinyMath.hpp, 132
getWidth
Text, 77
gRenderer
BreakoutGame, 28
gWindow
BreakoutGame, 28

include/Ball.hpp, 103
include/BreakoutGame.hpp, 105
include/Brick.hpp, 107
include/BricksGenerator.hpp, 109
include/Common.h, 110
include/ConfigUtil.hpp, 117
include/LanguageManager.hpp, 119
include/LTimer.h, 120
include/Paddle.hpp, 122
include/Player.hpp, 124
include/Rectangle.hpp, 125
include/ResourceManager.hpp, 126
include/Text.hpp, 128
include/TinyMath.hpp, 130
include/Wall.hpp, 137
init
    ResourceManager, 69
initBall
    BreakoutGame, 18
initGameObjects
    BreakoutGame, 18
Initializing
    Common.h, 113
initPaddle
    BreakoutGame, 19
initSDLSystems
    BreakoutGame, 20
isActive
    Brick, 34
isPaused
    LTimer, 47
isStarted
    LTimer, 48
    LTimer.cpp, 146

keepCentered
    Text, 80

Language, 41
    data, 41
languages
    LanguageSelector, 45
LanguageSelector, 42
    currentLanguage, 45
    getContent, 42
    languages, 45
    loadOneLanguageContent, 43
    useLanguage, 43
languageSelector
    BreakoutGame, 28
lastText
    Text, 81
LEFT
    TinyMath.hpp, 132
Left
    Common.h, 113
level
    BreakoutGame, 28
LevelData, 45
    data, 46
levelNum
    BreakoutGame, 28
levelText
    BreakoutGame, 28
LICENSE_GPLv3.txt
    Copyright, 96
    not, 97
    Version, 103
lives
    Player, 61
livesNum
    BreakoutGame, 29
livesText
    BreakoutGame, 29
loadAllVariables
    ConfigUtil, 39
loadConfig
    ConfigUtil, 39
loadLanguages
    BreakoutGame, 20
loadLevels
    BreakoutGame, 21
loadOneLanguageContent
    LanguageSelector, 43
loadOneLevelData
    BricksGenerator, 37
LoadResource
    ResourceManager, 69
loadResources
    BreakoutGame, 22
loop
    BreakoutGame, 22
loseLife
    Player, 60
loseLifeSound
    BreakoutGame, 29
loseSound
    BreakoutGame, 29
LTimer, 46
    getTicks, 47
    isPaused, 47
    isStarted, 48
    LTimer, 47
    mPaused, 49
    mPausedTicks, 50
    mStarted, 50
    mStartTicks, 50
    pause, 48
    start, 48
    stop, 49
    unpause, 49
LTimer.cpp
    isStarted, 146

Magnitude
    TinyMath.hpp, 133
main
    main.cpp, 147

main.cpp
main, 147
Matrix3D, 50
column, 52
Matrix3D, 51, 52
n, 54
operator!=, 52
operator(), 53
operator==, 53
operator[], 53, 54
maxLevel
BreakoutGame, 29
MENU_FONT_SIZE
common.cpp, 143
Common.h, 116
menuFont_
BreakoutGame, 29
Middle
Common.h, 113
mPaused
LTimer, 49
mPausedTicks
LTimer, 50
mStarted
LTimer, 50
mStartTicks
LTimer, 50

n
Matrix3D, 54
name
ResourceManager, 70
name_
ResourceManager, 73
None
Common.h, 113
Normalize
TinyMath.hpp, 133
not
LICENSE_GPLv3.txt, 97
notificationText
BreakoutGame, 30

operator!=
Matrix3D, 52
Vector3D, 85
operator∗
TinyMath.hpp, 134, 135
operator∗=
Vector3D, 85
operator()
Matrix3D, 53
operator+
TinyMath.hpp, 135
operator+=
Vector3D, 85
operator-
TinyMath.hpp, 135
operator-=
Vector3D, 85
operator/
TinyMath.hpp, 135
operator/=
Vector3D, 86
operator=
ResourceManager, 70
operator==
Matrix3D, 53
Vector3D, 86
operator[]
Matrix3D, 53, 54
Vector3D, 86

Paddle, 54
Draw, 56
Paddle, 55, 56
Update, 56
velocity, 57
paddle
BreakoutGame, 30
PADDLE_DISTANCE_FROM_BOTTOM
common.cpp, 143
Common.h, 116
PADDLE_HEIGHT
common.cpp, 143
Common.h, 116
PADDLE_SPEED
common.cpp, 144
Common.h, 116
PADDLE_WIDTH
common.cpp, 144
Common.h, 116
paddleHitSound
BreakoutGame, 30
PaddleLeft
Common.h, 113
PaddleRight
Common.h, 113
pause
LTimer, 48
PauseClearGame
Common.h, 113
PauseLoseGame
Common.h, 113
PauseLoseLife
Common.h, 113
PauseNormal
Common.h, 113
PauseWin
Common.h, 113
penetration
Contact, 41
Player, 57
addScore, 59
getLives, 59
getScore, 59
lives, 61
loseLife, 60

Player, 58
score, 61
setLives, 60
setScore, 61
player
BreakoutGame, 30
PLAYER_DEFAULT_LIFE_NUM
common.cpp, 144
Common.h, 116
position
Ball, 11
Rectangle, 64
Project
TinyMath.hpp, 136

rect
Ball, 11
Rectangle, 64
Text, 81
Rectangle, 62
Draw, 63
position, 64
rect, 64
Rectangle, 63
RemoveAllResource
ResourceManager, 70
RemoveResource
ResourceManager, 71
render
BreakoutGame, 23
renderer
Text, 81
resetBricks
BreakoutGame, 24
ResFilePath
ResourceManager.hpp, 127
resMap_
ResourceManager, 73
ResName
ResourceManager.hpp, 128
ResourceManager, 64
∼ResourceManager, 66
AddResource, 67
cfgFilePath_, 73
fileMap_, 73
getInstance, 68
init, 69
LoadResource, 69
name, 70
name_, 73
operator=, 70
RemoveAllResource, 70
RemoveResource, 71
resMap_, 73
ResourceManager, 66
size, 71
startManager, 71
stopManager, 72
useConfig_, 74
ResourceManager.hpp
ResFilePath, 127
ResName, 128
restBricks
BreakoutGame, 30
RIGHT
TinyMath.hpp, 132
Right
Common.h, 113
Running
Common.h, 113

score
Brick, 36
Player, 61
scoreNum
BreakoutGame, 30
scoreText
BreakoutGame, 31
SCREEN_FPS_60
common.cpp, 144
Common.h, 116
SCREEN_TICKS_PER_FRAME_60
common.cpp, 144
Common.h, 116
screenHeight
BreakoutGame, 31
screenWidth
BreakoutGame, 31
setActive
Brick, 35
SetCenterPosition
Text, 77
SetColor
Text, 78
setKeepCentered
Text, 78
setLives
Player, 60
SetPosition
Text, 79
setScore
Brick, 35
Player, 61
SetText
Text, 79
size
ResourceManager, 71
src/BreakoutGame.cpp, 139
src/common.cpp, 139
src/LTimer.cpp, 145
src/main.cpp, 146
start
LTimer, 48
startManager
ResourceManager, 71
stop
LTimer, 49
stopManager

ResourceManager, 72
surface
Text, 81

Text, 74
color, 80
Draw, 76
font, 80
getCenterPosition, 76
getHeight, 76
getWidth, 77
keepCentered, 80
lastText, 81
rect, 81
renderer, 81
SetCenterPosition, 77
SetColor, 78
setKeepCentered, 78
SetPosition, 79
SetText, 79
surface, 81
Text, 75
texture, 81
texture
Text, 81
TICKS_PER_UPDATE
common.cpp, 144
Common.h, 117
TinyMath.hpp
CrossProduct, 132
Direction, 131
Dot, 132
DOWN, 132
getUnitVectorFromDegree, 132
LEFT, 132
Magnitude, 133
Normalize, 133
operator∗, 134, 135
operator+, 135
operator-, 135
operator/, 135
Project, 136
RIGHT, 132
UP, 132
VectorDirectionSDL, 136
Top
Common.h, 113
type
Contact, 41

unpause
LTimer, 49
UP
TinyMath.hpp, 132
Update
Ball, 11
Paddle, 56
update
BreakoutGame, 24
updateAllTexts
BreakoutGame, 25
useConfig_
ResourceManager, 74
useLanguage
LanguageSelector, 43

Vector2D, 82
getRotatedVector, 83
Vector2D, 83
Vector3D, 84
operator!=, 85
operator∗=, 85
operator+=, 85
operator-=, 85
operator/=, 86
operator==, 86
operator[], 86
Vector3D, 85
x, 86
y, 87
z, 87
VectorDirectionSDL
TinyMath.hpp, 136
velocity
Ball, 11
Paddle, 57
Version
LICENSE_GPLv3.txt, 103

Wall, 87
Wall, 88
WALL_WIDTH
common.cpp, 144
Common.h, 117
wallHitSound
BreakoutGame, 31
wallLeft
BreakoutGame, 31
wallRight
BreakoutGame, 31
WINDOW_HEIGHT
common.cpp, 145
Common.h, 117
WINDOW_WIDTH
common.cpp, 145
Common.h, 117
winSound
BreakoutGame, 32

x
Vector3D, 86

y
Vector3D, 87

z
Vector3D, 87