

Platformer 2D

Generated by Doxygen 1.8.18



---

|  |           |
|--|-----------|
| <b>1 My Personal Index Page</b>                | <b>1</b>  |
| 1.1 Introduction . . . . .                     | 1         |
| 1.2 Installation . . . . .                     | 1         |
| 1.2.1 Step 1: Opening the box . . . . .        | 1         |
| <b>2 platformer-2d</b>                         | <b>3</b>  |
| <b>3 Module Index</b>                          | <b>5</b>  |
| 3.1 Modules . . . . .                          | 5         |
| <b>4 Namespace Index</b>                       | <b>7</b>  |
| 4.1 Namespace List . . . . .                   | 7         |
| <b>5 Hierarchical Index</b>                    | <b>9</b>  |
| 5.1 Class Hierarchy . . . . .                  | 9         |
| <b>6 Class Index</b>                           | <b>11</b> |
| 6.1 Class List . . . . .                       | 11        |
| <b>7 File Index</b>                            | <b>15</b> |
| 7.1 File List . . . . .                        | 15        |
| <b>8 Module Documentation</b>                  | <b>19</b> |
| 8.1 Core . . . . .                             | 19        |
| 8.1.1 Detailed Description . . . . .           | 20        |
| 8.1.2 Enumeration Type Documentation . . . . . | 20        |
| 8.1.2.1 Action . . . . .                       | 20        |
| 8.2 Encoder . . . . .                          | 21        |
| 8.2.1 Detailed Description . . . . .           | 21        |
| 8.3 Entities . . . . .                         | 22        |
| 8.3.1 Detailed Description . . . . .           | 23        |
| 8.3.2 Enumeration Type Documentation . . . . . | 23        |
| 8.3.2.1 JumpingState . . . . .                 | 23        |
| 8.3.2.2 MovingState . . . . .                  | 23        |
| 8.4 Events . . . . .                           | 24        |
| 8.4.1 Detailed Description . . . . .           | 24        |
| 8.5 GUI . . . . .                              | 25        |
| 8.5.1 Detailed Description . . . . .           | 26        |
| 8.5.2 Enumeration Type Documentation . . . . . | 26        |
| 8.5.2.1 Btn . . . . .                          | 27        |
| 8.5.3 Function Documentation . . . . .         | 27        |
| 8.5.3.1 mapListOf() . . . . .                  | 27        |
| 8.5.3.2 to_underlying() . . . . .              | 28        |
| 8.5.3.3 toInt() . . . . .                      | 29        |
| 8.5.4 Variable Documentation . . . . .         | 29        |

---

|  |           |
|--|-----------|
| 8.5.4.1 Buttons [1/6] . . . . .                | 30        |
| 8.5.4.2 Buttons [2/6] . . . . .                | 30        |
| 8.5.4.3 Buttons [3/6] . . . . .                | 30        |
| 8.5.4.4 Buttons [4/6] . . . . .                | 30        |
| 8.5.4.5 Buttons [5/6] . . . . .                | 31        |
| 8.5.4.6 Buttons [6/6] . . . . .                | 31        |
| 8.6 Resources . . . . .                        | 32        |
| 8.6.1 Detailed Description . . . . .           | 33        |
| 8.6.2 Enumeration Type Documentation . . . . . | 33        |
| 8.6.2.1 Music . . . . .                        | 33        |
| 8.6.2.2 Sound . . . . .                        | 33        |
| 8.6.2.3 Texture . . . . .                      | 33        |
| 8.7 States . . . . .                           | 35        |
| 8.7.1 Detailed Description . . . . .           | 35        |
| <b>9 Namespace Documentation</b>               | <b>37</b> |
| 9.1 config Namespace Reference . . . . .       | 37        |
| 9.1.1 Detailed Description . . . . .           | 38        |
| 9.1.2 Variable Documentation . . . . .         | 38        |
| 9.1.2.1 blocksCountHeight . . . . .            | 38        |
| 9.1.2.2 blocksCountWidth . . . . .             | 38        |
| 9.1.2.3 defaultJumpKey . . . . .               | 38        |
| 9.1.2.4 defaultMapName . . . . .               | 38        |
| 9.1.2.5 defaultRunLeftKey . . . . .            | 38        |
| 9.1.2.6 defaultRunRightKey . . . . .           | 39        |
| 9.1.2.7 defaultSoundVolume . . . . .           | 39        |
| 9.1.2.8 entityHeight . . . . .                 | 39        |
| 9.1.2.9 entityWidth . . . . .                  | 39        |
| 9.1.2.10 gameMusic . . . . .                   | 39        |
| 9.1.2.11 gravity . . . . .                     | 39        |
| 9.1.2.12 hitCeilingVelocity . . . . .          | 39        |
| 9.1.2.13 horizontalVelocity . . . . .          | 40        |
| 9.1.2.14 initialJumpVelocity . . . . .         | 40        |
| 9.1.2.15 jumpIcon . . . . .                    | 40        |
| 9.1.2.16 keybindsFontName . . . . .            | 40        |
| 9.1.2.17 largeBgIcon . . . . .                 | 40        |
| 9.1.2.18 mapEditorPath . . . . .               | 40        |
| 9.1.2.19 maxFps . . . . .                      | 40        |
| 9.1.2.20 pausedTextImage . . . . .             | 41        |
| 9.1.2.21 playerWon . . . . .                   | 41        |
| 9.1.2.22 programIcon . . . . .                 | 41        |
| 9.1.2.23 rebindingBg . . . . .                 | 41        |

---

|  |    |
|--|----|
| 9.1.2.24 runLeftIcon . . . . .                 | 41 |
| 9.1.2.25 runRightIcon . . . . .                | 41 |
| 9.1.2.26 sideBorder . . . . .                  | 41 |
| 9.1.2.27 terminalVelocity . . . . .            | 42 |
| 9.1.2.28 widgetsFontName . . . . .             | 42 |
| 9.1.2.29 windowHeight . . . . .                | 42 |
| 9.1.2.30 windowWidth . . . . .                 | 42 |
| 9.2 config::bg Namespace Reference . . . . .   | 42 |
| 9.2.1 Variable Documentation . . . . .         | 42 |
| 9.2.1.1 keybinds . . . . .                     | 42 |
| 9.2.1.2 loader . . . . .                       | 43 |
| 9.2.1.3 menu . . . . .                         | 43 |
| 9.2.1.4 options . . . . .                      | 43 |
| 9.2.1.5 paused . . . . .                       | 43 |
| 9.2.1.6 restart . . . . .                      | 43 |
| 9.3 Gui Namespace Reference . . . . .          | 43 |
| 9.3.1 Detailed Description . . . . .           | 43 |
| 9.4 Keybinds Namespace Reference . . . . .     | 44 |
| 9.4.1 Detailed Description . . . . .           | 44 |
| 9.5 lives Namespace Reference . . . . .        | 44 |
| 9.5.1 Detailed Description . . . . .           | 45 |
| 9.5.2 Variable Documentation . . . . .         | 45 |
| 9.5.2.1 borderColor . . . . .                  | 45 |
| 9.5.2.2 borderOffsetX . . . . .                | 45 |
| 9.5.2.3 borderOffsetY . . . . .                | 45 |
| 9.5.2.4 borderThickness . . . . .              | 45 |
| 9.5.2.5 boxSide . . . . .                      | 45 |
| 9.5.2.6 count . . . . .                        | 45 |
| 9.5.2.7 fillColor . . . . .                    | 46 |
| 9.5.2.8 heartOffsetX . . . . .                 | 46 |
| 9.5.2.9 heartOffsetY . . . . .                 | 46 |
| 9.6 Loader Namespace Reference . . . . .       | 46 |
| 9.6.1 Detailed Description . . . . .           | 46 |
| 9.6.2 Enumeration Type Documentation . . . . . | 46 |
| 9.6.2.1 Btn . . . . .                          | 46 |
| 9.7 Menu Namespace Reference . . . . .         | 47 |
| 9.7.1 Detailed Description . . . . .           | 47 |
| 9.7.2 Enumeration Type Documentation . . . . . | 47 |
| 9.7.2.1 Btn . . . . .                          | 47 |
| 9.8 Obj Namespace Reference . . . . .          | 48 |
| 9.8.1 Detailed Description . . . . .           | 48 |
| 9.8.2 Enumeration Type Documentation . . . . . | 48 |

---

|   |           |
|---|-----------|
| 9.8.2.1 Entity                                | 48        |
| 9.9 Options Namespace Reference               | 49        |
| 9.9.1 Detailed Description                    | 50        |
| 9.9.2 Enumeration Type Documentation          | 50        |
| 9.9.2.1 Btn                                   | 50        |
| 9.10 Paused Namespace Reference               | 50        |
| 9.10.1 Detailed Description                   | 50        |
| 9.10.2 Enumeration Type Documentation         | 51        |
| 9.10.2.1 Btn                                  | 51        |
| 9.11 res Namespace Reference                  | 51        |
| 9.11.1 Detailed Description                   | 52        |
| 9.12 Restart Namespace Reference              | 52        |
| 9.12.1 Detailed Description                   | 52        |
| 9.12.2 Enumeration Type Documentation         | 52        |
| 9.12.2.1 Btn                                  | 52        |
| 9.13 state Namespace Reference                | 53        |
| 9.13.1 Detailed Description                   | 53        |
| 9.13.2 Variable Documentation                 | 53        |
| 9.13.2.1 aboutID                              | 53        |
| 9.13.2.2 gameID                               | 53        |
| 9.13.2.3 keybindsID                           | 54        |
| 9.13.2.4 loaderID                             | 54        |
| 9.13.2.5 menuID                               | 54        |
| 9.13.2.6 optionsID                            | 54        |
| 9.13.2.7 pausedID                             | 54        |
| 9.13.2.8 restartID                            | 54        |
| <b>10 Class Documentation</b>                 | <b>55</b> |
| 10.1 ActionMap Class Reference                | 55        |
| 10.1.1 Constructor & Destructor Documentation | 56        |
| 10.1.1.1 ActionMap() [1/2]                    | 56        |
| 10.1.1.2 ~ActionMap()                         | 56        |
| 10.1.1.3 ActionMap() [2/2]                    | 56        |
| 10.1.2 Member Function Documentation          | 57        |
| 10.1.2.1 getActionMap()                       | 57        |
| 10.1.2.2 getActionMapSize()                   | 57        |
| 10.1.2.3 insert()                             | 57        |
| 10.1.2.4 Instance()                           | 58        |
| 10.1.2.5 operator=()                          | 58        |
| 10.1.2.6 set()                                | 58        |
| 10.1.3 Member Data Documentation              | 59        |
| 10.1.3.1 actionMap                            | 59        |

---

|   |    |
|---|----|
| 10.2 AudioCfg Class Reference . . . . .                 | 59 |
| 10.2.1 Detailed Description . . . . .                   | 60 |
| 10.2.2 Member Data Documentation . . . . .              | 60 |
| 10.2.2.1 isFpsEnabled . . . . .                         | 60 |
| 10.2.2.2 isSoundEnabled . . . . .                       | 60 |
| 10.2.2.3 volume . . . . .                               | 60 |
| 10.3 Block Class Reference . . . . .                    | 61 |
| 10.3.1 Detailed Description . . . . .                   | 63 |
| 10.3.2 Constructor & Destructor Documentation . . . . . | 63 |
| 10.3.2.1 Block() . . . . .                              | 63 |
| 10.3.3 Member Function Documentation . . . . .          | 64 |
| 10.3.3.1 draw() . . . . .                               | 64 |
| 10.4 Bmp Struct Reference . . . . .                     | 64 |
| 10.4.1 Detailed Description . . . . .                   | 65 |
| 10.5 BmpReader Class Reference . . . . .                | 65 |
| 10.5.1 Detailed Description . . . . .                   | 68 |
| 10.5.2 Constructor & Destructor Documentation . . . . . | 69 |
| 10.5.2.1 BmpReader() [1/2] . . . . .                    | 69 |
| 10.5.2.2 ~BmpReader() . . . . .                         | 69 |
| 10.5.2.3 BmpReader() [2/2] . . . . .                    | 69 |
| 10.5.3 Member Function Documentation . . . . .          | 69 |
| 10.5.3.1 debugPrint() . . . . .                         | 69 |
| 10.5.3.2 operator=( ) . . . . .                         | 70 |
| 10.5.3.3 readFile() . . . . .                           | 70 |
| 10.5.3.4 readHeight() . . . . .                         | 71 |
| 10.5.3.5 readPixels() . . . . .                         | 71 |
| 10.5.3.6 readWidth() . . . . .                          | 71 |
| 10.5.4 Member Data Documentation . . . . .              | 72 |
| 10.5.4.1 bytesPerPixel . . . . .                        | 72 |
| 10.5.4.2 dataOffset . . . . .                           | 72 |
| 10.5.4.3 height . . . . .                               | 72 |
| 10.5.4.4 heightOffset . . . . .                         | 72 |
| 10.5.4.5 pixelsCount . . . . .                          | 72 |
| 10.5.4.6 width . . . . .                                | 72 |
| 10.5.4.7 widthOffset . . . . .                          | 72 |
| 10.6 Camera::Bot Struct Reference . . . . .             | 73 |
| 10.6.1 Detailed Description . . . . .                   | 73 |
| 10.7 Camera Class Reference . . . . .                   | 73 |
| 10.7.1 Detailed Description . . . . .                   | 76 |
| 10.7.2 Member Enumeration Documentation . . . . .       | 77 |
| 10.7.2.1 CurrentCollision . . . . .                     | 77 |
| 10.7.3 Constructor & Destructor Documentation . . . . . | 78 |

---

---

|   |    |
|---|----|
| 10.7.3.1 Camera() . . . . .                             | 78 |
| 10.7.4 Member Function Documentation . . . . .          | 78 |
| 10.7.4.1 detectCollisionX() . . . . .                   | 78 |
| 10.7.4.2 detectCollisionY() . . . . .                   | 79 |
| 10.7.4.3 getCameraView() . . . . .                      | 80 |
| 10.7.4.4 setCameraConstants() . . . . .                 | 80 |
| 10.7.4.5 setController() . . . . .                      | 81 |
| 10.7.4.6 setControllerConstants() . . . . .             | 81 |
| 10.7.4.7 storeJustSolvedForOtherAxis() . . . . .        | 82 |
| 10.7.4.8 updateX() . . . . .                            | 83 |
| 10.7.4.9 updateY() . . . . .                            | 84 |
| 10.7.5 Member Data Documentation . . . . .              | 84 |
| 10.7.5.1 bottomBorderBoundary . . . . .                 | 85 |
| 10.7.5.2 cameraHeight . . . . .                         | 85 |
| 10.7.5.3 cameraView . . . . .                           | 85 |
| 10.7.5.4 cameraWidth . . . . .                          | 85 |
| 10.7.5.5 controller . . . . .                           | 86 |
| 10.7.5.6 halvedCameraHeight . . . . .                   | 86 |
| 10.7.5.7 halvedCameraWidth . . . . .                    | 86 |
| 10.7.5.8 halvedSpriteHeight . . . . .                   | 86 |
| 10.7.5.9 halvedSpriteWidth . . . . .                    | 86 |
| 10.7.5.10 leftBorderBoundary . . . . .                  | 87 |
| 10.7.5.11 mapHeight . . . . .                           | 87 |
| 10.7.5.12 mapWidth . . . . .                            | 87 |
| 10.7.5.13 previouslySolvedOtherAxis . . . . .           | 87 |
| 10.7.5.14 rightBorderBoundary . . . . .                 | 88 |
| 10.7.5.15 solvedNow . . . . .                           | 88 |
| 10.7.5.16 topBorderBoundary . . . . .                   | 88 |
| 10.8 CollisionEvent Class Reference . . . . .           | 89 |
| 10.8.1 Detailed Description . . . . .                   | 91 |
| 10.8.2 Constructor & Destructor Documentation . . . . . | 92 |
| 10.8.2.1 CollisionEvent() [1/2] . . . . .               | 92 |
| 10.8.2.2 CollisionEvent() [2/2] . . . . .               | 92 |
| 10.8.3 Member Function Documentation . . . . .          | 92 |
| 10.8.3.1 handleCollision() . . . . .                    | 92 |
| 10.8.3.2 operator=() . . . . .                          | 93 |
| 10.8.3.3 playerInRangeOf() . . . . .                    | 93 |
| 10.8.3.4 playerIntersects() . . . . .                   | 94 |
| 10.8.3.5 resolveCollisionAxisX() . . . . .              | 95 |
| 10.8.3.6 resolveCollisionAxisY() . . . . .              | 96 |
| 10.8.3.7 updateAxisX() . . . . .                        | 97 |
| 10.8.3.8 updateAxisY() . . . . .                        | 97 |

---

|  |     |
|--|-----|
| 10.8.4 Member Data Documentation . . . . .               | 98  |
| 10.8.4.1 blocks . . . . .                                | 98  |
| 10.8.4.2 collidableRange . . . . .                       | 98  |
| 10.8.4.3 player . . . . .                                | 98  |
| 10.9 Keybinds::Config Struct Reference . . . . .         | 98  |
| 10.9.1 Detailed Description . . . . .                    | 100 |
| 10.9.2 Constructor & Destructor Documentation . . . . .  | 101 |
| 10.9.2.1 Config() . . . . .                              | 101 |
| 10.9.3 Member Function Documentation . . . . .           | 101 |
| 10.9.3.1 encode() . . . . .                              | 101 |
| 10.9.3.2 init() . . . . .                                | 102 |
| 10.9.4 Member Data Documentation . . . . .               | 102 |
| 10.9.4.1 offsetY . . . . .                               | 102 |
| 10.10 Loader::Config Struct Reference . . . . .          | 103 |
| 10.10.1 Detailed Description . . . . .                   | 104 |
| 10.10.2 Constructor & Destructor Documentation . . . . . | 105 |
| 10.10.2.1 Config() . . . . .                             | 105 |
| 10.10.3 Member Function Documentation . . . . .          | 105 |
| 10.10.3.1 encode() . . . . .                             | 105 |
| 10.10.3.2 init() . . . . .                               | 106 |
| 10.10.4 Member Data Documentation . . . . .              | 106 |
| 10.10.4.1 mapNameColor . . . . .                         | 106 |
| 10.11 Menu::Config Struct Reference . . . . .            | 107 |
| 10.11.1 Detailed Description . . . . .                   | 108 |
| 10.11.2 Constructor & Destructor Documentation . . . . . | 109 |
| 10.11.2.1 Config() . . . . .                             | 109 |
| 10.11.3 Member Function Documentation . . . . .          | 109 |
| 10.11.3.1 encode() . . . . .                             | 109 |
| 10.11.3.2 init() . . . . .                               | 110 |
| 10.11.4 Member Data Documentation . . . . .              | 110 |
| 10.11.4.1 offsetY . . . . .                              | 110 |
| 10.12 Options::Config Struct Reference . . . . .         | 111 |
| 10.12.1 Detailed Description . . . . .                   | 112 |
| 10.12.2 Constructor & Destructor Documentation . . . . . | 113 |
| 10.12.2.1 Config() . . . . .                             | 113 |
| 10.12.3 Member Function Documentation . . . . .          | 113 |
| 10.12.3.1 encode() . . . . .                             | 113 |
| 10.12.3.2 init() . . . . .                               | 114 |
| 10.12.4 Member Data Documentation . . . . .              | 114 |
| 10.12.4.1 offsetY . . . . .                              | 114 |
| 10.13 Paused::Config Struct Reference . . . . .          | 115 |
| 10.13.1 Detailed Description . . . . .                   | 116 |

---

|   |     |
|---|-----|
| 10.13.2 Constructor & Destructor Documentation . . . . .        | 117 |
| 10.13.2.1 Config() . . . . .                                    | 117 |
| 10.13.3 Member Function Documentation . . . . .                 | 117 |
| 10.13.3.1 encode() . . . . .                                    | 117 |
| 10.13.3.2 init() . . . . .                                      | 118 |
| 10.13.4 Member Data Documentation . . . . .                     | 118 |
| 10.13.4.1 offsetY . . . . .                                     | 118 |
| 10.14 Restart::Config Struct Reference . . . . .                | 119 |
| 10.14.1 Detailed Description . . . . .                          | 121 |
| 10.14.2 Constructor & Destructor Documentation . . . . .        | 121 |
| 10.14.2.1 Config() . . . . .                                    | 121 |
| 10.14.3 Member Function Documentation . . . . .                 | 121 |
| 10.14.3.1 encode() . . . . .                                    | 121 |
| 10.14.3.2 init() . . . . .                                      | 122 |
| 10.14.4 Member Data Documentation . . . . .                     | 122 |
| 10.14.4.1 offsetX . . . . .                                     | 122 |
| 10.14.4.2 offsetY . . . . .                                     | 123 |
| 10.14.4.3 spacing . . . . .                                     | 123 |
| 10.15 Gui::Config< Widget > Struct Template Reference . . . . . | 123 |
| 10.15.1 Detailed Description . . . . .                          | 124 |
| 10.15.2 Member Function Documentation . . . . .                 | 124 |
| 10.15.2.1 prepare() . . . . .                                   | 124 |
| 10.15.3 Member Data Documentation . . . . .                     | 124 |
| 10.15.3.1 backgroundColor . . . . .                             | 125 |
| 10.15.3.2 borderColor . . . . .                                 | 125 |
| 10.15.3.3 font . . . . .  | 125 |
| 10.15.3.4 height . . . . .                                      | 125 |
| 10.15.3.5 opacity . . . . .                                     | 125 |
| 10.15.3.6 textSize . . . . .                                    | 125 |
| 10.15.3.7 width . . . . .                                       | 126 |
| 10.16 Encoder< ReaderKey > Class Template Reference . . . . .   | 126 |
| 10.16.1 Detailed Description . . . . .                          | 127 |
| 10.16.2 Constructor & Destructor Documentation . . . . .        | 127 |
| 10.16.2.1 Encoder() . . . . .                                   | 127 |
| 10.16.3 Member Function Documentation . . . . .                 | 127 |
| 10.16.3.1 encode() . . . . .                                    | 127 |
| 10.16.3.2 encodeAll() . . . . .                                 | 128 |
| 10.16.3.3 printDebug() . . . . .                                | 129 |
| 10.16.4 Member Data Documentation . . . . .                     | 129 |
| 10.16.4.1 encodedObjects . . . . .                              | 129 |
| 10.17 Entity Class Reference . . . . .                          | 130 |
| 10.17.1 Detailed Description . . . . .                          | 132 |

---

---

|  |     |
|--|-----|
| 10.17.2 Constructor & Destructor Documentation . . . . . | 133 |
| 10.17.2.1 ~Entity() . . . . .                            | 133 |
| 10.17.2.2 Entity() [1/5] . . . . .                       | 133 |
| 10.17.2.3 Entity() [2/5] . . . . .                       | 133 |
| 10.17.2.4 Entity() [3/5] . . . . .                       | 133 |
| 10.17.2.5 Entity() [4/5] . . . . .                       | 134 |
| 10.17.2.6 Entity() [5/5] . . . . .                       | 134 |
| 10.17.3 Member Function Documentation . . . . .          | 134 |
| 10.17.3.1 bot() . . . . .                                | 134 |
| 10.17.3.2 draw() . . . . .                               | 135 |
| 10.17.3.3 getGlobalBounds() . . . . .                    | 136 |
| 10.17.3.4 getSprite() . . . . .                          | 136 |
| 10.17.3.5 height() . . . . .                             | 137 |
| 10.17.3.6 left() . . . . .                               | 137 |
| 10.17.3.7 move() [1/2] . . . . .                         | 137 |
| 10.17.3.8 move() [2/2] . . . . .                         | 138 |
| 10.17.3.9 operator=() [1/2] . . . . .                    | 138 |
| 10.17.3.10 operator=() [2/2] . . . . .                   | 139 |
| 10.17.3.11 right() . . . . .                             | 139 |
| 10.17.3.12 rotateLeft() . . . . .                        | 140 |
| 10.17.3.13 rotateRight() . . . . .                       | 140 |
| 10.17.3.14 rotateTop() . . . . .                         | 140 |
| 10.17.3.15 setPosition() [1/2] . . . . .                 | 141 |
| 10.17.3.16 setPosition() [2/2] . . . . .                 | 141 |
| 10.17.3.17 top() . . . . .                               | 141 |
| 10.17.3.18 width() . . . . .                             | 142 |
| 10.17.4 Member Data Documentation . . . . .              | 142 |
| 10.17.4.1 position . . . . .                             | 142 |
| 10.17.4.2 sprite . . . . .                               | 143 |
| 10.18 Event Class Reference . . . . .                    | 143 |
| 10.18.1 Detailed Description . . . . .                   | 144 |
| 10.18.2 Constructor & Destructor Documentation . . . . . | 144 |
| 10.18.2.1 ~Event() . . . . .                             | 144 |
| 10.18.3 Member Function Documentation . . . . .          | 144 |
| 10.18.3.1 update() . . . . .                             | 144 |
| 10.19 FileReader< T > Class Template Reference . . . . . | 145 |
| 10.19.1 Detailed Description . . . . .                   | 147 |
| 10.19.2 Constructor & Destructor Documentation . . . . . | 147 |
| 10.19.2.1 ~FileReader() . . . . .                        | 147 |
| 10.19.2.2 FileReader() [1/2] . . . . .                   | 147 |
| 10.19.2.3 FileReader() [2/2] . . . . .                   | 147 |
| 10.19.3 Member Function Documentation . . . . .          | 147 |

---

---

|  |     |
|--|-----|
| 10.19.3.1 getData() [1/2] . . . . .                      | 148 |
| 10.19.3.2 getData() [2/2] . . . . .                      | 148 |
| 10.19.3.3 isOpened() . . . . .                           | 148 |
| 10.19.3.4 operator=() . . . . .                          | 148 |
| 10.19.3.5 readFile() . . . . .                           | 149 |
| 10.19.4 Member Data Documentation . . . . .              | 149 |
| 10.19.4.1 data . . . . .                                 | 149 |
| 10.19.4.2 file . . . . .                                 | 149 |
| 10.19.4.3 fileName . . . . .                             | 149 |
| 10.20 FpsOverlay Class Reference . . . . .               | 150 |
| 10.20.1 Detailed Description . . . . .                   | 152 |
| 10.20.2 Constructor & Destructor Documentation . . . . . | 152 |
| 10.20.2.1 FpsOverlay() . . . . .                         | 152 |
| 10.20.3 Member Function Documentation . . . . .          | 152 |
| 10.20.3.1 calcCurrentFps() . . . . .                     | 153 |
| 10.20.3.2 draw() . . . . .                               | 153 |
| 10.20.3.3 setFont() . . . . .                            | 153 |
| 10.20.3.4 setFps() . . . . .                             | 154 |
| 10.20.3.5 update() . . . . .                             | 154 |
| 10.20.4 Member Data Documentation . . . . .              | 155 |
| 10.20.4.1 font . . . . .                                 | 155 |
| 10.20.4.2 fps . . . . .                                  | 155 |
| 10.21 Game Class Reference . . . . .                     | 156 |
| 10.21.1 Detailed Description . . . . .                   | 158 |
| 10.21.2 Constructor & Destructor Documentation . . . . . | 158 |
| 10.21.2.1 Game() . . . . .                               | 158 |
| 10.21.3 Member Function Documentation . . . . .          | 158 |
| 10.21.3.1 computeDeltaTime() . . . . .                   | 158 |
| 10.21.3.2 createStates() . . . . .                       | 159 |
| 10.21.3.3 draw() . . . . .                               | 159 |
| 10.21.3.4 getResources() . . . . .                       | 160 |
| 10.21.3.5 getStateMachine() . . . . .                    | 160 |
| 10.21.3.6 getWindow() . . . . .                          | 161 |
| 10.21.3.7 isRunning() . . . . .                          | 161 |
| 10.21.3.8 processInput() . . . . .                       | 162 |
| 10.21.3.9 run() . . . . .                                | 162 |
| 10.21.3.10 update() . . . . .                            | 163 |
| 10.21.4 Member Data Documentation . . . . .              | 164 |
| 10.21.4.1 clock . . . . .                                | 164 |
| 10.21.4.2 dt . . . . .                                   | 164 |
| 10.21.4.3 resources . . . . .                            | 165 |
| 10.21.4.4 stateMachine . . . . .                         | 165 |

---

---

|  |     |
|--|-----|
| 10.21.4.5 window . . . . .                               | 165 |
| 10.22 HeartCollectible Class Reference . . . . .         | 165 |
| 10.22.1 Detailed Description . . . . .                   | 168 |
| 10.22.2 Constructor & Destructor Documentation . . . . . | 168 |
| 10.22.2.1 HeartCollectible() . . . . .                   | 168 |
| 10.22.3 Member Function Documentation . . . . .          | 168 |
| 10.22.3.1 draw() . . . . .                               | 168 |
| 10.22.3.2 setCollected() . . . . .                       | 169 |
| 10.22.3.3 wasCollected() . . . . .                       | 169 |
| 10.22.4 Member Data Documentation . . . . .              | 169 |
| 10.22.4.1 collected . . . . .                            | 169 |
| 10.23 IConfig< E > Struct Template Reference . . . . .   | 170 |
| 10.23.1 Detailed Description . . . . .                   | 171 |
| 10.23.2 Member Function Documentation . . . . .          | 171 |
| 10.23.2.1 encode() . . . . .                             | 171 |
| 10.23.3 Member Data Documentation . . . . .              | 171 |
| 10.23.3.1 widgetsNames . . . . .                         | 171 |
| 10.24 ILoader Class Reference . . . . .                  | 172 |
| 10.24.1 Detailed Description . . . . .                   | 173 |
| 10.24.2 Constructor & Destructor Documentation . . . . . | 173 |
| 10.24.2.1 ILoader() [1/3] . . . . .                      | 174 |
| 10.24.2.2 ~ILoader() . . . . .                           | 174 |
| 10.24.2.3 ILoader() [2/3] . . . . .                      | 174 |
| 10.24.2.4 ILoader() [3/3] . . . . .                      | 174 |
| 10.24.3 Member Function Documentation . . . . .          | 174 |
| 10.24.3.1 load() . . . . .                               | 174 |
| 10.24.3.2 operator=() [1/2] . . . . .                    | 175 |
| 10.24.3.3 operator=() [2/2] . . . . .                    | 175 |
| 10.25 InputEvent Class Reference . . . . .               | 175 |
| 10.25.1 Detailed Description . . . . .                   | 177 |
| 10.25.2 Constructor & Destructor Documentation . . . . . | 177 |
| 10.25.2.1 InputEvent() [1/2] . . . . .                   | 177 |
| 10.25.2.2 InputEvent() [2/2] . . . . .                   | 178 |
| 10.25.3 Member Function Documentation . . . . .          | 178 |
| 10.25.3.1 isPressed() . . . . .                          | 178 |
| 10.25.3.2 operator=() . . . . .                          | 178 |
| 10.25.3.3 update() . . . . .                             | 179 |
| 10.25.3.4 updateHorizontal() . . . . .                   | 179 |
| 10.25.3.5 updateVertical() . . . . .                     | 180 |
| 10.25.4 Member Data Documentation . . . . .              | 180 |
| 10.25.4.1 actions . . . . .                              | 181 |
| 10.25.4.2 keys . . . . .                                 | 181 |

---

|  |     |
|--|-----|
| 10.25.4.3 player . . . . .                               | 181 |
| 10.25.4.4 resources . . . . .                            | 181 |
| 10.25.4.5 window . . . . .                               | 181 |
| 10.26 KeybindsView Class Reference . . . . .             | 182 |
| 10.26.1 Detailed Description . . . . .                   | 184 |
| 10.26.2 Constructor & Destructor Documentation . . . . . | 185 |
| 10.26.2.1 KeybindsView() . . . . .                       | 185 |
| 10.26.3 Member Function Documentation . . . . .          | 185 |
| 10.26.3.1 addBackgroundInto() . . . . .                  | 185 |
| 10.26.3.2 addCancelLabelInto() . . . . .                 | 185 |
| 10.26.3.3 addPressKeyLabelInto() . . . . .               | 186 |
| 10.26.3.4 buildGUI() . . . . .                           | 186 |
| 10.26.3.5 createBorders() . . . . .                      | 187 |
| 10.26.3.6 createButtons() . . . . .                      | 188 |
| 10.26.3.7 createButtonsPanel() . . . . .                 | 188 |
| 10.26.3.8 createIcons() . . . . .                        | 189 |
| 10.26.3.9 createJumpIcon() . . . . .                     | 189 |
| 10.26.3.10 createLargeBgIcon() . . . . .                 | 190 |
| 10.26.3.11 createLeftBorder() . . . . .                  | 190 |
| 10.26.3.12 createPanels() . . . . .                      | 190 |
| 10.26.3.13 createRebindingPanel() . . . . .              | 191 |
| 10.26.3.14 createRebindingPanelGroup() . . . . .         | 191 |
| 10.26.3.15 createRightBorder() . . . . .                 | 192 |
| 10.26.3.16 createRunLeftIcon() . . . . .                 | 193 |
| 10.26.3.17 createRunRightIcon() . . . . .                | 193 |
| 10.26.3.18 createTitleLabel() . . . . .                  | 193 |
| 10.26.3.19 createTitlePanel() . . . . .                  | 194 |
| 10.26.3.20 init() . . . . .                              | 194 |
| 10.26.3.21 loadWidget() . . . . .                        | 194 |
| 10.26.4 Member Data Documentation . . . . .              | 195 |
| 10.26.4.1 config . . . . .                               | 195 |
| 10.26.4.2 optionsConfig . . . . .                        | 195 |
| 10.27 Camera::Left Struct Reference . . . . .            | 195 |
| 10.27.1 Detailed Description . . . . .                   | 196 |
| 10.28 Life Class Reference . . . . .                     | 196 |
| 10.28.1 Detailed Description . . . . .                   | 199 |
| 10.28.2 Constructor & Destructor Documentation . . . . . | 199 |
| 10.28.2.1 Life() . . . . .                               | 199 |
| 10.28.3 Member Function Documentation . . . . .          | 200 |
| 10.28.3.1 draw() . . . . .                               | 200 |
| 10.28.3.2 getAliveTexture() . . . . .                    | 200 |
| 10.28.3.3 getDeadTexture() . . . . .                     | 200 |

---

|   |     |
|---|-----|
| 10.28.3.4 <code>isAvailable()</code>                                    | 201 |
| 10.28.3.5 <code>kill()</code>   | 201 |
| 10.28.3.6 <code>revive()</code>   | 201 |
| 10.28.3.7 <code>storeAliveTexture()</code>                              | 201 |
| 10.28.3.8 <code>storeDeadTexture()</code>                               | 202 |
| 10.28.4 Member Data Documentation                                       | 202 |
| 10.28.4.1 <code>aliveTexture</code>                                     | 202 |
| 10.28.4.2 <code>deadTexture</code>                                      | 202 |
| 10.28.4.3 <code>lifeAvailable</code>                                    | 202 |
| 10.29 LivesBorder Class Reference                                       | 203 |
| 10.29.1 Detailed Description  | 204 |
| 10.29.2 Constructor & Destructor Documentation                          | 204 |
| 10.29.2.1 <code>LivesBorder()</code>                                    | 204 |
| 10.29.3 Member Function Documentation                                   | 204 |
| 10.29.3.1 <code>draw()</code>   | 204 |
| 10.29.3.2 <code>setBorder()</code>                                      | 205 |
| 10.29.4 Member Data Documentation                                       | 205 |
| 10.29.4.1 <code>border</code>   | 205 |
| 10.29.4.2 <code>viewport</code>   | 205 |
| 10.30 LivesOverlay Class Reference                                      | 206 |
| 10.30.1 Detailed Description  | 208 |
| 10.30.2 Constructor & Destructor Documentation                          | 208 |
| 10.30.2.1 <code>LivesOverlay()</code>                                   | 208 |
| 10.30.3 Member Function Documentation                                   | 209 |
| 10.30.3.1 <code>decreaseLife()</code>                                   | 209 |
| 10.30.3.2 <code>draw()</code>   | 209 |
| 10.30.3.3 <code>getAvailableLives()</code>                              | 209 |
| 10.30.3.4 <code>hasAllLives()</code>                                    | 210 |
| 10.30.3.5 <code>increaseLife()</code>                                   | 210 |
| 10.30.3.6 <code>isDead()</code>   | 210 |
| 10.30.3.7 <code>livesAvailable()</code>                                 | 211 |
| 10.30.3.8 <code>refillLives()</code>                                    | 211 |
| 10.30.4 Member Data Documentation                                       | 211 |
| 10.30.4.1 <code>availableLives</code>                                   | 211 |
| 10.30.4.2 <code>border</code>   | 211 |
| 10.30.4.3 <code>HUD</code>  | 212 |
| 10.30.4.4 <code>lives</code>  | 212 |
| 10.30.4.5 <code>renderWindow</code>                                     | 212 |
| 10.31 <code>mapListOfHelper&lt; T &gt;</code> Struct Template Reference | 212 |
| 10.31.1 Detailed Description  | 213 |
| 10.31.2 Constructor & Destructor Documentation                          | 213 |
| 10.31.2.1 <code>mapListOfHelper()</code>                                | 213 |

---

|  |     |
|--|-----|
| 10.31.3 Member Function Documentation . . . . .                | 213 |
| 10.31.3.1 operator()() . . . . .                               | 213 |
| 10.31.4 Member Data Documentation . . . . .                    | 213 |
| 10.31.4.1 data . . . . .                                       | 214 |
| 10.32 MapLoader< FileType > Class Template Reference . . . . . | 214 |
| 10.32.1 Detailed Description . . . . .                         | 216 |
| 10.32.2 Constructor & Destructor Documentation . . . . .       | 216 |
| 10.32.2.1 MapLoader() [1/3] . . . . .                          | 217 |
| 10.32.2.2 ~MapLoader() . . . . .                               | 217 |
| 10.32.2.3 MapLoader() [2/3] . . . . .                          | 217 |
| 10.32.2.4 MapLoader() [3/3] . . . . .                          | 217 |
| 10.32.3 Member Function Documentation . . . . .                | 218 |
| 10.32.3.1 analyzeBmpMapData() . . . . .                        | 218 |
| 10.32.3.2 analyzeTxtMapData() . . . . .                        | 218 |
| 10.32.3.3 getData() . . . . .                                  | 218 |
| 10.32.3.4 getQueue() . . . . .                                 | 218 |
| 10.32.3.5 load() . . . . .                                     | 219 |
| 10.32.3.6 operator=() [1/2] . . . . .                          | 219 |
| 10.32.3.7 operator=() [2/2] . . . . .                          | 219 |
| 10.32.4 Member Data Documentation . . . . .                    | 219 |
| 10.32.4.1 blocksNum . . . . .                                  | 219 |
| 10.32.4.2 encoder . . . . .                                    | 220 |
| 10.32.4.3 mapReader . . . . .                                  | 220 |
| 10.32.4.4 queue . . . . .                                      | 220 |
| 10.33 MapLoaderView Class Reference . . . . .                  | 220 |
| 10.33.1 Detailed Description . . . . .                         | 223 |
| 10.33.2 Constructor & Destructor Documentation . . . . .       | 223 |
| 10.33.2.1 MapLoaderView() . . . . .                            | 223 |
| 10.33.3 Member Function Documentation . . . . .                | 224 |
| 10.33.3.1 animateBadMapLabel() . . . . .                       | 224 |
| 10.33.3.2 buildGUI() . . . . .                                 | 224 |
| 10.33.3.3 clearMapNameBox() . . . . .                          | 225 |
| 10.33.3.4 clearMapNameBoxWithPrompt() . . . . .                | 226 |
| 10.33.3.5 createBadMapLabel() . . . . .                        | 226 |
| 10.33.3.6 createButtons() . . . . .                            | 227 |
| 10.33.3.7 createMainPanel() . . . . .                          | 227 |
| 10.33.3.8 createMapNameBox() . . . . .                         | 228 |
| 10.33.3.9 getMapName() . . . . .                               | 228 |
| 10.33.3.10 init() . . . . .                                    | 229 |
| 10.33.3.11 isPromptToEnter() . . . . .                         | 229 |
| 10.33.3.12 loadWidget() . . . . .                              | 230 |
| 10.33.3.13 setBadMapLabelVisible() . . . . .                   | 230 |

---

|  |     |
|--|-----|
| 10.33.3.14 setPromptToEnter() . . . . .                              | 231 |
| 10.33.4 Member Data Documentation . . . . .                          | 231 |
| 10.33.4.1 config . . . . .   | 231 |
| 10.33.4.2 loaderConfig . . . . .                                     | 231 |
| 10.33.4.3 promptToEnterMapName . . . . .                             | 231 |
| 10.34 MapNameValidator Class Reference . . . . .                     | 232 |
| 10.34.1 Detailed Description . . . . .                               | 233 |
| 10.34.2 Constructor & Destructor Documentation . . . . .             | 233 |
| 10.34.2.1 MapNameValidator() . . . . .                               | 233 |
| 10.34.3 Member Function Documentation . . . . .                      | 233 |
| 10.34.3.1 exists() . . . . .   | 234 |
| 10.34.3.2 isBmp() . . . . .  | 234 |
| 10.34.3.3 isTxt() . . . . .  | 234 |
| 10.34.3.4 isValidFormat() . . . . .                                  | 235 |
| 10.34.4 Member Data Documentation . . . . .                          | 235 |
| 10.34.4.1 fileName . . . . .   | 235 |
| 10.34.4.2 mapNamePattern . . . . .                                   | 235 |
| 10.34.4.3 match . . . . .  | 236 |
| 10.34.4.4 re . . . . .   | 236 |
| 10.35 MenuView Class Reference . . . . .                             | 236 |
| 10.35.1 Detailed Description . . . . .                               | 239 |
| 10.35.2 Constructor & Destructor Documentation . . . . .             | 239 |
| 10.35.2.1 MenuView() . . . . .                                       | 239 |
| 10.35.3 Member Function Documentation . . . . .                      | 239 |
| 10.35.3.1 buildGUI() . . . . .                                       | 240 |
| 10.35.3.2 createButtons() . . . . .                                  | 240 |
| 10.35.3.3 createMainPanel() . . . . .                                | 241 |
| 10.35.3.4 init() . . . . .   | 241 |
| 10.35.3.5 loadWidget() . . . . .                                     | 241 |
| 10.35.4 Member Data Documentation . . . . .                          | 242 |
| 10.35.4.1 config . . . . .   | 242 |
| 10.35.4.2 menuConfig . . . . .                                       | 242 |
| 10.36 MissingFont Class Reference . . . . .                          | 243 |
| 10.36.1 Detailed Description . . . . .                               | 244 |
| 10.36.2 Constructor & Destructor Documentation . . . . .             | 244 |
| 10.36.2.1 MissingFont() . . . . .                                    | 244 |
| 10.36.3 Member Function Documentation . . . . .                      | 244 |
| 10.36.3.1 what() . . . . .   | 244 |
| 10.36.4 Member Data Documentation . . . . .                          | 244 |
| 10.36.4.1 msg . . . . .  | 244 |
| 10.37 MissingResource< Resource > Class Template Reference . . . . . | 245 |
| 10.37.1 Detailed Description . . . . .                               | 246 |

---

---

|  |     |
|--|-----|
| 10.37.2 Constructor & Destructor Documentation . . . . . | 246 |
| 10.37.2.1 MissingResource() [1/2] . . . . .              | 246 |
| 10.37.2.2 MissingResource() [2/2] . . . . .              | 247 |
| 10.37.3 Member Function Documentation . . . . .          | 247 |
| 10.37.3.1 initMsg() . . . . .                            | 247 |
| 10.37.3.2 what() . . . . .                               | 248 |
| 10.37.4 Member Data Documentation . . . . .              | 248 |
| 10.37.4.1 msg . . . . .                                  | 248 |
| 10.38 MovementEvent Class Reference . . . . .            | 248 |
| 10.38.1 Detailed Description . . . . .                   | 250 |
| 10.38.2 Constructor & Destructor Documentation . . . . . | 250 |
| 10.38.2.1 MovementEvent() [1/2] . . . . .                | 250 |
| 10.38.2.2 MovementEvent() [2/2] . . . . .                | 251 |
| 10.38.3 Member Function Documentation . . . . .          | 251 |
| 10.38.3.1 operator=() . . . . .                          | 251 |
| 10.38.3.2 updateAxisX() . . . . .                        | 251 |
| 10.38.3.3 updateAxisY() . . . . .                        | 252 |
| 10.38.4 Member Data Documentation . . . . .              | 252 |
| 10.38.4.1 blocks . . . . .                               | 252 |
| 10.38.4.2 player . . . . .                               | 252 |
| 10.38.4.3 velX . . . . .                                 | 253 |
| 10.39 Camera::None Struct Reference . . . . .            | 253 |
| 10.39.1 Detailed Description . . . . .                   | 253 |
| 10.40 Objective Class Reference . . . . .                | 254 |
| 10.40.1 Detailed Description . . . . .                   | 256 |
| 10.40.2 Constructor & Destructor Documentation . . . . . | 256 |
| 10.40.2.1 Objective() . . . . .                          | 256 |
| 10.40.3 Member Function Documentation . . . . .          | 256 |
| 10.40.3.1 draw() . . . . .                               | 256 |
| 10.41 OptionsView Class Reference . . . . .              | 257 |
| 10.41.1 Detailed Description . . . . .                   | 260 |
| 10.41.2 Constructor & Destructor Documentation . . . . . | 260 |
| 10.41.2.1 OptionsView() . . . . .                        | 260 |
| 10.41.3 Member Function Documentation . . . . .          | 261 |
| 10.41.3.1 buildGUI() . . . . .                           | 261 |
| 10.41.3.2 createButtons() . . . . .                      | 262 |
| 10.41.3.3 createButtonsPanel() . . . . .                 | 262 |
| 10.41.3.4 createCheckboxes() . . . . .                   | 263 |
| 10.41.3.5 createFpsCheckBox() . . . . .                  | 263 |
| 10.41.3.6 createFpsLabel() . . . . .                     | 264 |
| 10.41.3.7 createLabels() . . . . .                       | 264 |
| 10.41.3.8 createMainPanel() . . . . .                    | 265 |

---

---

|  |     |
|--|-----|
| 10.41.3.9 createPanels()                       | 265 |
| 10.41.3.10 createSoundCheckBox()               | 266 |
| 10.41.3.11 createSoundLabel()                  | 266 |
| 10.41.3.12 createSoundVolumeSlider()           | 266 |
| 10.41.3.13 getVolume()                         | 267 |
| 10.41.3.14 init()                              | 267 |
| 10.41.3.15 isFpsChecked()                      | 267 |
| 10.41.3.16 isSoundChecked()                    | 268 |
| 10.41.3.17 loadWidget()                        | 268 |
| 10.41.4 Member Data Documentation              | 268 |
| 10.41.4.1 config                               | 268 |
| 10.41.4.2 optionsConfig                        | 269 |
| 10.42 overload< Ts > Struct Template Reference | 269 |
| 10.42.1 Detailed Description                   | 270 |
| 10.43 Overload Struct Reference                | 270 |
| 10.43.1 Detailed Description                   | 271 |
| 10.44 PausedView Class Reference               | 271 |
| 10.44.1 Detailed Description                   | 274 |
| 10.44.2 Constructor & Destructor Documentation | 274 |
| 10.44.2.1 PausedView()                         | 274 |
| 10.44.3 Member Function Documentation          | 274 |
| 10.44.3.1 buildGUI()                           | 275 |
| 10.44.3.2 createButtons()                      | 275 |
| 10.44.3.3 createMainPanel()                    | 276 |
| 10.44.3.4 createPausedTextImage()              | 276 |
| 10.44.3.5 init()                               | 277 |
| 10.44.3.6 loadWidget()                         | 277 |
| 10.44.4 Member Data Documentation              | 277 |
| 10.44.4.1 config                               | 277 |
| 10.44.4.2 menuConfig                           | 278 |
| 10.45 PhysicsEvent Class Reference             | 278 |
| 10.45.1 Detailed Description                   | 279 |
| 10.45.2 Constructor & Destructor Documentation | 279 |
| 10.45.2.1 ~PhysicsEvent()                      | 279 |
| 10.45.3 Member Function Documentation          | 279 |
| 10.45.3.1 updateAxisX()                        | 280 |
| 10.45.3.2 updateAxisY()                        | 280 |
| 10.46 PixelColor Struct Reference              | 280 |
| 10.46.1 Detailed Description                   | 281 |
| 10.46.2 Constructor & Destructor Documentation | 281 |
| 10.46.2.1 PixelColor() [1/2]                   | 281 |
| 10.46.2.2 PixelColor() [2/2]                   | 281 |

---

---

|  |     |
|--|-----|
| 10.46.3 Member Function Documentation . . . . .                          | 282 |
| 10.46.3.1 operator<() . . . . .  | 282 |
| 10.46.4 Friends And Related Function Documentation . . . . .             | 282 |
| 10.46.4.1 operator>> . . . . .   | 282 |
| 10.46.5 Member Data Documentation . . . . .                              | 282 |
| 10.46.5.1 blue . . . . .   | 282 |
| 10.46.5.2 green . . . . .  | 282 |
| 10.46.5.3 red . . . . .  | 283 |
| 10.47 Player Class Reference . . . . .                                   | 283 |
| 10.47.1 Detailed Description . . . . .                                   | 287 |
| 10.47.2 Constructor & Destructor Documentation . . . . .                 | 287 |
| 10.47.2.1 Player() . . . . .   | 287 |
| 10.47.3 Member Function Documentation . . . . .                          | 287 |
| 10.47.3.1 draw() . . . . .   | 288 |
| 10.47.3.2 getGravVelocity() . . . . .                                    | 288 |
| 10.47.3.3 getJumpVelocity() . . . . .                                    | 288 |
| 10.47.3.4 getStartingPosition() . . . . .                                | 289 |
| 10.47.3.5 getVelocityX() . . . . .                                       | 289 |
| 10.47.3.6 gravityFrame() . . . . .                                       | 289 |
| 10.47.3.7 hitCeilingUpdate() . . . . .                                   | 290 |
| 10.47.3.8 isDetectingGround() . . . . .                                  | 291 |
| 10.47.3.9 isIntersecting() . . . . .                                     | 291 |
| 10.47.3.10 jumpFrame() . . . . .   | 292 |
| 10.47.3.11 kill() . . . . .  | 293 |
| 10.47.3.12 landOnGroundUpdate() . . . . .                                | 294 |
| 10.47.3.13 resetGravVelocity() . . . . .                                 | 294 |
| 10.47.3.14 setJumpVelocity() . . . . .                                   | 294 |
| 10.47.3.15 setStartingPosition() . . . . .                               | 295 |
| 10.47.3.16 setToStartingPosition() . . . . .                             | 295 |
| 10.47.3.17 updateVelocity() . . . . .                                    | 296 |
| 10.47.4 Member Data Documentation . . . . .                              | 297 |
| 10.47.4.1 feetDetectorRange . . . . .                                    | 297 |
| 10.47.4.2 gravity . . . . .  | 297 |
| 10.47.4.3 gravVelocity . . . . .   | 297 |
| 10.47.4.4 jumpingState . . . . .   | 297 |
| 10.47.4.5 jumpVelocity . . . . .   | 298 |
| 10.47.4.6 movingState . . . . .  | 298 |
| 10.47.4.7 position . . . . .   | 298 |
| 10.47.4.8 startingPosition . . . . .                                     | 298 |
| 10.47.4.9 velocityX . . . . .  | 298 |
| 10.48 ResourceHolder< Key, Resource > Class Template Reference . . . . . | 299 |
| 10.48.1 Detailed Description . . . . .                                   | 301 |

---

---

|  |     |
|--|-----|
| 10.48.2 Constructor & Destructor Documentation . . . . .               | 301 |
| 10.48.2.1 ResourceHolder() . . . . .                                   | 301 |
| 10.48.3 Member Function Documentation . . . . .                        | 302 |
| 10.48.3.1 erase() . . . . .  | 302 |
| 10.48.3.2 eraseAll() . . . . .   | 302 |
| 10.48.3.3 get() . . . . .  | 302 |
| 10.48.3.4 getResources() . . . . .                                     | 303 |
| 10.48.3.5 getResourcesDir() . . . . .                                  | 303 |
| 10.48.3.6 has() . . . . .  | 303 |
| 10.48.3.7 insert() . . . . .   | 304 |
| 10.48.3.8 operator+=() [1/2] . . . . .                                 | 304 |
| 10.48.3.9 operator+=() [2/2] . . . . .                                 | 305 |
| 10.48.3.10 operator[]() [1/2] . . . . .                                | 305 |
| 10.48.3.11 operator[]() [2/2] . . . . .                                | 306 |
| 10.48.3.12 setResourcesDir() . . . . .                                 | 306 |
| 10.48.4 Member Data Documentation . . . . .                            | 306 |
| 10.48.4.1 resources . . . . .  | 307 |
| 10.48.4.2 resourcesDir . . . . .                                       | 307 |
| 10.49 ResourceInserter< Key, Args > Class Template Reference . . . . . | 307 |
| 10.49.1 Detailed Description . . . . .                                 | 308 |
| 10.49.2 Constructor & Destructor Documentation . . . . .               | 308 |
| 10.49.2.1 ResourceInserter() . . . . .                                 | 308 |
| 10.49.3 Member Data Documentation . . . . .                            | 308 |
| 10.49.3.1 args . . . . .   | 308 |
| 10.49.3.2 fileName . . . . .   | 308 |
| 10.49.3.3 key . . . . .  | 309 |
| 10.50 ResourceManager Class Reference . . . . .                        | 309 |
| 10.50.1 Detailed Description . . . . .                                 | 311 |
| 10.50.2 Constructor & Destructor Documentation . . . . .               | 311 |
| 10.50.2.1 ResourceManager() . . . . .                                  | 311 |
| 10.50.3 Member Function Documentation . . . . .                        | 312 |
| 10.50.3.1 getMusic() . . . . .   | 312 |
| 10.50.3.2 getSounds() . . . . .  | 312 |
| 10.50.3.3 getTextures() . . . . .                                      | 312 |
| 10.50.3.4 loadMusic() . . . . .  | 313 |
| 10.50.3.5 loadResources() . . . . .                                    | 313 |
| 10.50.3.6 loadSounds() . . . . .                                       | 314 |
| 10.50.3.7 loadTextures() . . . . .                                     | 314 |
| 10.50.4 Member Data Documentation . . . . .                            | 315 |
| 10.50.4.1 music . . . . .  | 315 |
| 10.50.4.2 sounds . . . . .   | 315 |
| 10.50.4.3 textures . . . . .   | 315 |

---

---

|  |     |
|--|-----|
| 10.51 RestartView Class Reference . . . . .              | 315 |
| 10.51.1 Detailed Description . . . . .                   | 318 |
| 10.51.2 Constructor & Destructor Documentation . . . . . | 318 |
| 10.51.2.1 RestartView() . . . . .                        | 318 |
| 10.51.3 Member Function Documentation . . . . .          | 319 |
| 10.51.3.1 buildGUI() . . . . .                           | 319 |
| 10.51.3.2 createButtons() . . . . .                      | 320 |
| 10.51.3.3 createButtonsPanel() . . . . .                 | 320 |
| 10.51.3.4 createGameFinishedImage() . . . . .            | 321 |
| 10.51.3.5 createMainPanel() . . . . .                    | 321 |
| 10.51.3.6 createPanels() . . . . .                       | 321 |
| 10.51.3.7 init() . . . . .                               | 322 |
| 10.51.3.8 loadWidget() . . . . .                         | 323 |
| 10.51.3.9 setGameLostTexture() . . . . .                 | 323 |
| 10.51.3.10 setGameWonTexture() . . . . .                 | 323 |
| 10.51.4 Member Data Documentation . . . . .              | 324 |
| 10.51.4.1 config . . . . .                               | 324 |
| 10.51.4.2 gameFinishedImage . . . . .                    | 324 |
| 10.51.4.3 lostTexture . . . . .                          | 324 |
| 10.51.4.4 restartConfig . . . . .                        | 324 |
| 10.51.4.5 wonTexture . . . . .                           | 324 |
| 10.52 Camera::Right Struct Reference . . . . .           | 325 |
| 10.52.1 Detailed Description . . . . .                   | 325 |
| 10.53 Spike Class Reference . . . . .                    | 325 |
| 10.53.1 Detailed Description . . . . .                   | 328 |
| 10.53.2 Constructor & Destructor Documentation . . . . . | 328 |
| 10.53.2.1 Spike() . . . . .                              | 328 |
| 10.53.3 Member Function Documentation . . . . .          | 328 |
| 10.53.3.1 draw() . . . . .                               | 329 |
| 10.53.3.2 setTextureDirection() . . . . .                | 329 |
| 10.53.4 Member Data Documentation . . . . .              | 330 |
| 10.53.4.1 spikeType . . . . .                            | 330 |
| 10.54 State Class Reference . . . . .                    | 330 |
| 10.54.1 Detailed Description . . . . .                   | 331 |
| 10.54.2 Constructor & Destructor Documentation . . . . . | 331 |
| 10.54.2.1 ~State() . . . . .                             | 331 |
| 10.54.3 Member Function Documentation . . . . .          | 332 |
| 10.54.3.1 draw() . . . . .                               | 332 |
| 10.54.3.2 onActivate() . . . . .                         | 332 |
| 10.54.3.3 onCreate() . . . . .                           | 332 |
| 10.54.3.4 onDeactivate() . . . . .                       | 332 |
| 10.54.3.5 onDestroy() . . . . .                          | 333 |

---

|  |     |
|--|-----|
| 10.54.3.6 processInput() . . . . .                       | 333 |
| 10.54.3.7 update() . . . . .                             | 333 |
| 10.55 StateAbout Class Reference . . . . .               | 333 |
| 10.55.1 Detailed Description . . . . .                   | 336 |
| 10.55.2 Constructor & Destructor Documentation . . . . . | 336 |
| 10.55.2.1 StateAbout() . . . . .                         | 336 |
| 10.55.3 Member Function Documentation . . . . .          | 336 |
| 10.55.3.1 draw() . . . . .                               | 336 |
| 10.55.3.2 processInput() . . . . .                       | 337 |
| 10.55.4 Member Data Documentation . . . . .              | 337 |
| 10.55.4.1 aboutInfo . . . . .                            | 337 |
| 10.55.4.2 stateMachine . . . . .                         | 337 |
| 10.56 StateGame Class Reference . . . . .                | 338 |
| 10.56.1 Detailed Description . . . . .                   | 341 |
| 10.56.2 Constructor & Destructor Documentation . . . . . | 341 |
| 10.56.2.1 StateGame() . . . . .                          | 341 |
| 10.56.3 Member Function Documentation . . . . .          | 342 |
| 10.56.3.1 canCollect() . . . . .                         | 342 |
| 10.56.3.2 checkLoseCondition() . . . . .                 | 343 |
| 10.56.3.3 checkWinCondition() . . . . .                  | 343 |
| 10.56.3.4 draw() . . . . .                               | 343 |
| 10.56.3.5 generateWorldFromBmp() . . . . .               | 345 |
| 10.56.3.6 generateWorldFromTxt() . . . . .               | 345 |
| 10.56.3.7 isInDrawRange() . . . . .                      | 346 |
| 10.56.3.8 loadMusic() . . . . .                          | 347 |
| 10.56.3.9 onActivate() . . . . .                         | 347 |
| 10.56.3.10 onCreate() . . . . .                          | 347 |
| 10.56.3.11 onDeactivate() . . . . .                      | 348 |
| 10.56.3.12 onDestroy() . . . . .                         | 348 |
| 10.56.3.13 playerFallToDeath() . . . . .                 | 349 |
| 10.56.3.14 processInput() . . . . .                      | 349 |
| 10.56.3.15 restartGameLevel() . . . . .                  | 350 |
| 10.56.3.16 setBlocksTextures() . . . . .                 | 350 |
| 10.56.3.17 setCollectiblesTextures() . . . . .           | 351 |
| 10.56.3.18 setEntitiesTextures() . . . . .               | 351 |
| 10.56.3.19 setSpikesTextures() . . . . .                 | 352 |
| 10.56.3.20 update() . . . . .                            | 352 |
| 10.56.3.21 updateGainLifeLogic() . . . . .               | 353 |
| 10.56.3.22 updateGameStatus() . . . . .                  | 354 |
| 10.56.3.23 updateLoseLifeLogic() . . . . .               | 354 |
| 10.56.3.24 updatePhysics() . . . . .                     | 355 |
| 10.56.3.25 updatePlayerLife() . . . . .                  | 355 |

---

|  |     |
|--|-----|
| 10.56.4 Member Data Documentation . . . . .              | 356 |
| 10.56.4.1 background . . . . .                           | 356 |
| 10.56.4.2 blocks . . . . .                               | 356 |
| 10.56.4.3 blocksQueue . . . . .                          | 356 |
| 10.56.4.4 bottomBorderHeight . . . . .                   | 356 |
| 10.56.4.5 camera . . . . .                               | 356 |
| 10.56.4.6 collectSound . . . . .                         | 357 |
| 10.56.4.7 collisionEvent . . . . .                       | 357 |
| 10.56.4.8 deathSound . . . . .                           | 357 |
| 10.56.4.9 fps . . . . .                                  | 357 |
| 10.56.4.10 hearts . . . . .                              | 357 |
| 10.56.4.11 inputEvent . . . . .                          | 357 |
| 10.56.4.12 lives . . . . .                               | 357 |
| 10.56.4.13 mapLoader . . . . .                           | 357 |
| 10.56.4.14 movementsEvent . . . . .                      | 358 |
| 10.56.4.15 music . . . . .                               | 358 |
| 10.56.4.16 objective . . . . .                           | 358 |
| 10.56.4.17 player . . . . .                              | 358 |
| 10.56.4.18 resources . . . . .                           | 358 |
| 10.56.4.19 spikes . . . . .                              | 358 |
| 10.56.4.20 stateMachine . . . . .                        | 358 |
| 10.56.4.21 window . . . . .                              | 358 |
| 10.56.4.22 winGameSound . . . . .                        | 359 |
| 10.57 StateKeybinds Class Reference . . . . .            | 359 |
| 10.57.1 Detailed Description . . . . .                   | 361 |
| 10.57.2 Constructor & Destructor Documentation . . . . . | 361 |
| 10.57.2.1 StateKeybinds() . . . . .                      | 361 |
| 10.57.3 Member Function Documentation . . . . .          | 362 |
| 10.57.3.1 draw() . . . . .                               | 362 |
| 10.57.3.2 onActivate() . . . . .                         | 362 |
| 10.57.3.3 onCreate() . . . . .                           | 363 |
| 10.57.3.4 processInput() . . . . .                       | 363 |
| 10.57.3.5 update() . . . . .                             | 363 |
| 10.57.3.6 updateHoverSoundVolume() . . . . .             | 364 |
| 10.57.4 Member Data Documentation . . . . .              | 364 |
| 10.57.4.1 action . . . . .                               | 365 |
| 10.57.4.2 actionMap . . . . .                            | 365 |
| 10.57.4.3 gui . . . . .                                  | 365 |
| 10.57.4.4 onHoverBtnSound . . . . .                      | 365 |
| 10.57.4.5 stateMachine . . . . .                         | 365 |
| 10.57.4.6 window . . . . .                               | 365 |
| 10.58 StateMachine Class Reference . . . . .             | 366 |

---

---

|  |     |
|--|-----|
| 10.58.1 Detailed Description . . . . .                   | 367 |
| 10.58.2 Member Function Documentation . . . . .          | 367 |
| 10.58.2.1 draw() . . . . .                               | 367 |
| 10.58.2.2 erase() . . . . .                              | 367 |
| 10.58.2.3 getPreviousStateID() . . . . .                 | 368 |
| 10.58.2.4 insert() . . . . .                             | 368 |
| 10.58.2.5 operator+=() . . . . .                         | 369 |
| 10.58.2.6 operator=( ) . . . . .                         | 369 |
| 10.58.2.7 processInput() . . . . .                       | 369 |
| 10.58.2.8 setPreviousStateID() . . . . .                 | 370 |
| 10.58.2.9 switchTo() . . . . .                           | 370 |
| 10.58.2.10 switchToDefaultState() . . . . .              | 371 |
| 10.58.2.11 switchToPreviousState() . . . . .             | 371 |
| 10.58.2.12 update() . . . . .                            | 372 |
| 10.58.3 Member Data Documentation . . . . .              | 372 |
| 10.58.3.1 currentInsertedID . . . . .                    | 372 |
| 10.58.3.2 currentState . . . . .                         | 372 |
| 10.58.3.3 previousStateID . . . . .                      | 373 |
| 10.58.3.4 states . . . . .                               | 373 |
| 10.59 StateMapLoader Class Reference . . . . .           | 373 |
| 10.59.1 Detailed Description . . . . .                   | 376 |
| 10.59.2 Constructor & Destructor Documentation . . . . . | 376 |
| 10.59.2.1 StateMapLoader() . . . . .                     | 376 |
| 10.59.3 Member Function Documentation . . . . .          | 377 |
| 10.59.3.1 createGameFrom() . . . . .                     | 377 |
| 10.59.3.2 draw() . . . . .                               | 378 |
| 10.59.3.3 onActivate() . . . . .                         | 378 |
| 10.59.3.4 onCreate() . . . . .                           | 379 |
| 10.59.3.5 printHelp() . . . . .                          | 379 |
| 10.59.3.6 processInput() . . . . .                       | 380 |
| 10.59.3.7 setMapLoaderButton() . . . . .                 | 380 |
| 10.59.3.8 updateHoverSoundVolume() . . . . .             | 381 |
| 10.59.4 Member Data Documentation . . . . .              | 381 |
| 10.59.4.1 gui . . . . .                                  | 381 |
| 10.59.4.2 mapLoader . . . . .                            | 381 |
| 10.59.4.3 onHoverBtnSound . . . . .                      | 381 |
| 10.59.4.4 resources . . . . .                            | 381 |
| 10.59.4.5 stateMachine . . . . .                         | 382 |
| 10.59.4.6 window . . . . .                               | 382 |
| 10.60 StateMenu Class Reference . . . . .                | 382 |
| 10.60.1 Detailed Description . . . . .                   | 385 |
| 10.60.2 Constructor & Destructor Documentation . . . . . | 385 |

---

---

|  |     |
|--|-----|
| 10.60.2.1 StateMenu() . . . . .                          | 385 |
| 10.60.3 Member Function Documentation . . . . .          | 386 |
| 10.60.3.1 draw() . . . . .                               | 386 |
| 10.60.3.2 onActivate() . . . . .                         | 386 |
| 10.60.3.3 onCreate() . . . . .                           | 387 |
| 10.60.3.4 onDeactivate() . . . . .                       | 387 |
| 10.60.3.5 processInput() . . . . .                       | 388 |
| 10.60.3.6 updateHoverSoundVolume() . . . . .             | 388 |
| 10.60.4 Member Data Documentation . . . . .              | 388 |
| 10.60.4.1 gui . . . . .                                  | 388 |
| 10.60.4.2 mapLoader . . . . .                            | 389 |
| 10.60.4.3 onHoverBtnSound . . . . .                      | 389 |
| 10.60.4.4 resources . . . . .                            | 389 |
| 10.60.4.5 stateMachine . . . . .                         | 389 |
| 10.60.4.6 window . . . . .                               | 389 |
| 10.61 StateOptions Class Reference . . . . .             | 390 |
| 10.61.1 Detailed Description . . . . .                   | 392 |
| 10.61.2 Constructor & Destructor Documentation . . . . . | 392 |
| 10.61.2.1 StateOptions() . . . . .                       | 392 |
| 10.61.3 Member Function Documentation . . . . .          | 393 |
| 10.61.3.1 draw() . . . . .                               | 393 |
| 10.61.3.2 onCreate() . . . . .                           | 393 |
| 10.61.3.3 processInput() . . . . .                       | 394 |
| 10.61.3.4 saveOptions() . . . . .                        | 395 |
| 10.61.3.5 updateHoverSoundVolume() . . . . .             | 395 |
| 10.61.3.6 updateSlider() . . . . .                       | 396 |
| 10.61.4 Member Data Documentation . . . . .              | 396 |
| 10.61.4.1 corners . . . . .                              | 396 |
| 10.61.4.2 cornersTextures . . . . .                      | 396 |
| 10.61.4.3 gui . . . . .                                  | 396 |
| 10.61.4.4 onHoverBtnSound . . . . .                      | 397 |
| 10.61.4.5 stateMachine . . . . .                         | 397 |
| 10.61.4.6 window . . . . .                               | 397 |
| 10.62 StatePaused Class Reference . . . . .              | 397 |
| 10.62.1 Detailed Description . . . . .                   | 400 |
| 10.62.2 Constructor & Destructor Documentation . . . . . | 400 |
| 10.62.2.1 StatePaused() . . . . .                        | 400 |
| 10.62.3 Member Function Documentation . . . . .          | 401 |
| 10.62.3.1 draw() . . . . .                               | 401 |
| 10.62.3.2 onActivate() . . . . .                         | 401 |
| 10.62.3.3 onCreate() . . . . .                           | 402 |
| 10.62.3.4 onDeactivate() . . . . .                       | 402 |

---

---

|   |     |
|---|-----|
| 10.62.3.5 processInput()                          | 403 |
| 10.62.4 Member Data Documentation                 | 403 |
| 10.62.4.1 gui                                     | 403 |
| 10.62.4.2 onHoverBtnSound                         | 403 |
| 10.62.4.3 pausedView                              | 403 |
| 10.62.4.4 stateMachine                            | 404 |
| 10.62.4.5 window                                  | 404 |
| 10.63 StateRestart Class Reference                | 404 |
| 10.63.1 Detailed Description                      | 406 |
| 10.63.2 Constructor & Destructor Documentation    | 406 |
| 10.63.2.1 StateRestart()                          | 406 |
| 10.63.3 Member Function Documentation             | 407 |
| 10.63.3.1 draw()                                  | 407 |
| 10.63.3.2 onActivate()                            | 407 |
| 10.63.3.3 onCreate()                              | 408 |
| 10.63.3.4 onDeactivate()                          | 408 |
| 10.63.3.5 processInput()                          | 409 |
| 10.63.4 Member Data Documentation                 | 409 |
| 10.63.4.1 gui                                     | 409 |
| 10.63.4.2 stateMachine                            | 409 |
| 10.63.4.3 window                                  | 409 |
| 10.64 Camera::Top Struct Reference                | 410 |
| 10.64.1 Detailed Description                      | 410 |
| 10.65 Txt Struct Reference                        | 410 |
| 10.65.1 Detailed Description                      | 411 |
| 10.66 TxtReader Class Reference                   | 411 |
| 10.66.1 Detailed Description                      | 414 |
| 10.66.2 Constructor & Destructor Documentation    | 414 |
| 10.66.2.1 TxtReader() [1/2]                       | 414 |
| 10.66.2.2 ~TxtReader()                            | 415 |
| 10.66.2.3 TxtReader() [2/2]                       | 415 |
| 10.66.3 Member Function Documentation             | 415 |
| 10.66.3.1 debugPrint()                            | 415 |
| 10.66.3.2 isValidTxt()                            | 415 |
| 10.66.3.3 operator=()                             | 416 |
| 10.66.3.4 readFile()                              | 416 |
| 10.66.4 Member Data Documentation                 | 416 |
| 10.66.4.1 rawTxtData                              | 416 |
| 10.67 View< TWidgetPtr > Class Template Reference | 417 |
| 10.67.1 Constructor & Destructor Documentation    | 418 |
| 10.67.1.1 View() [1/3]                            | 418 |
| 10.67.1.2 ~View()                                 | 418 |

---

---

|  |            |
|--|------------|
| 10.67.1.3 View() [2/3] . . . . .                         | 419        |
| 10.67.1.4 View() [3/3] . . . . .                         | 419        |
| 10.67.2 Member Function Documentation . . . . .          | 419        |
| 10.67.2.1 buildGUI() . . . . .                           | 419        |
| 10.67.2.2 createBackgroundFrom() . . . . .               | 419        |
| 10.67.2.3 draw() . . . . .                               | 420        |
| 10.67.2.4 getView() . . . . .                            | 420        |
| 10.67.2.5 handleEvent() . . . . .                        | 421        |
| 10.67.2.6 operator=() [1/2] . . . . .                    | 421        |
| 10.67.2.7 operator=() [2/2] . . . . .                    | 421        |
| 10.67.3 Member Data Documentation . . . . .              | 422        |
| 10.67.3.1 buttonsCounter . . . . .                       | 422        |
| 10.67.3.2 view . . . . .                                 | 422        |
| 10.67.3.3 widgets . . . . .                              | 422        |
| 10.68 Window Class Reference . . . . .                   | 422        |
| 10.68.1 Detailed Description . . . . .                   | 423        |
| 10.68.2 Constructor & Destructor Documentation . . . . . | 423        |
| 10.68.2.1 Window() . . . . .                             | 424        |
| 10.68.3 Member Function Documentation . . . . .          | 424        |
| 10.68.3.1 beginDraw() . . . . .                          | 424        |
| 10.68.3.2 close() . . . . .                              | 424        |
| 10.68.3.3 draw() . . . . .                               | 424        |
| 10.68.3.4 endDraw() . . . . .                            | 425        |
| 10.68.3.5 getEvent() . . . . .                           | 426        |
| 10.68.3.6 getHeight() . . . . .                          | 426        |
| 10.68.3.7 getWidth() . . . . .                           | 427        |
| 10.68.3.8 getWindow() . . . . .                          | 427        |
| 10.68.3.9 isOpen() . . . . .                             | 427        |
| 10.68.3.10 setWindowIcon() . . . . .                     | 428        |
| 10.68.3.11 update() . . . . .                            | 428        |
| 10.68.3.12 updateView() . . . . .                        | 428        |
| 10.68.4 Member Data Documentation . . . . .              | 429        |
| 10.68.4.1 events . . . . .                               | 429        |
| 10.68.4.2 window . . . . .                               | 429        |
| <b>11 File Documentation</b> . . . . .                   | <b>431</b> |
| 11.1 Action.h File Reference . . . . .                   | 431        |
| 11.2 ActionMap.cpp File Reference . . . . .              | 432        |
| 11.3 ActionMap.h File Reference . . . . .                | 432        |
| 11.4 Block.cpp File Reference . . . . .                  | 433        |
| 11.5 Block.h File Reference . . . . .                    | 434        |
| 11.6 BmpReader.cpp File Reference . . . . .              | 435        |

---

---

|   |     |
|---|-----|
| 11.7 BmpReader.h File Reference . . . . .           | 435 |
| 11.8 Camera.cpp File Reference . . . . .            | 436 |
| 11.9 Camera.h File Reference . . . . .              | 436 |
| 11.10 CMakeLists.txt File Reference . . . . .       | 438 |
| 11.11 CollisionEvent.cpp File Reference . . . . .   | 438 |
| 11.12 CollisionEvent.h File Reference . . . . .     | 438 |
| 11.13 Config.h File Reference . . . . .             | 439 |
| 11.14 Configuration.cpp File Reference . . . . .    | 440 |
| 11.15 Configuration.h File Reference . . . . .      | 441 |
| 11.16 Encoder.cpp File Reference . . . . .          | 442 |
| 11.17 Encoder.h File Reference . . . . .            | 443 |
| 11.18 Entity.cpp File Reference . . . . .           | 444 |
| 11.19 Entity.h File Reference . . . . .             | 444 |
| 11.20 Event.h File Reference . . . . .              | 446 |
| 11.21 FileReader.cpp File Reference . . . . .       | 446 |
| 11.22 FileReader.h File Reference . . . . .         | 447 |
| 11.23 FpsOverlay.cpp File Reference . . . . .       | 448 |
| 11.24 FpsOverlay.h File Reference . . . . .         | 448 |
| 11.25 Game.cpp File Reference . . . . .             | 449 |
| 11.26 Game.h File Reference . . . . .               | 450 |
| 11.27 HeartCollectible.cpp File Reference . . . . . | 450 |
| 11.28 HeartCollectible.h File Reference . . . . .   | 451 |
| 11.29 IConfig.h File Reference . . . . .            | 452 |
| 11.30 InputEvent.cpp File Reference . . . . .       | 453 |
| 11.31 InputEvent.h File Reference . . . . .         | 453 |
| 11.32 KeybindsConfig.cpp File Reference . . . . .   | 454 |
| 11.33 KeybindsConfig.h File Reference . . . . .     | 455 |
| 11.34 KeybindsView.cpp File Reference . . . . .     | 457 |
| 11.35 KeybindsView.h File Reference . . . . .       | 457 |
| 11.36 Life.cpp File Reference . . . . .             | 458 |
| 11.37 Life.h File Reference . . . . .               | 459 |
| 11.38 LivesBorder.cpp File Reference . . . . .      | 460 |
| 11.39 LivesBorder.h File Reference . . . . .        | 460 |
| 11.40 LivesOverlay.cpp File Reference . . . . .     | 461 |
| 11.41 LivesOverlay.h File Reference . . . . .       | 462 |
| 11.42 LivesOverlayConfig.h File Reference . . . . . | 463 |
| 11.43 main.cpp File Reference . . . . .             | 464 |
| 11.43.1 Function Documentation . . . . .            | 464 |
| 11.43.1.1 main() . . . . .                          | 464 |
| 11.44 MapLoader.cpp File Reference . . . . .        | 465 |
| 11.45 MapLoader.h File Reference . . . . .          | 465 |
| 11.46 MapLoaderConfig.cpp File Reference . . . . .  | 466 |

---

---

|   |     |
|---|-----|
| 11.47 MapLoaderConfig.h File Reference . . . . .    | 467 |
| 11.48 MapLoaderView.cpp File Reference . . . . .    | 469 |
| 11.49 MapLoaderView.h File Reference . . . . .      | 469 |
| 11.50 MapNameValidator.cpp File Reference . . . . . | 470 |
| 11.51 MapNameValidator.h File Reference . . . . .   | 471 |
| 11.52 MenuConfig.cpp File Reference . . . . .       | 471 |
| 11.53 MenuConfig.h File Reference . . . . .         | 472 |
| 11.54 MenuView.cpp File Reference . . . . .         | 474 |
| 11.55 MenuView.h File Reference . . . . .           | 474 |
| 11.56 MovementEvent.cpp File Reference . . . . .    | 475 |
| 11.57 MovementEvent.h File Reference . . . . .      | 476 |
| 11.58 Objective.cpp File Reference . . . . .        | 477 |
| 11.59 Objective.h File Reference . . . . .          | 477 |
| 11.60 OptionsConfig.cpp File Reference . . . . .    | 478 |
| 11.61 OptionsConfig.h File Reference . . . . .      | 479 |
| 11.62 OptionsView.cpp File Reference . . . . .      | 480 |
| 11.63 OptionsView.h File Reference . . . . .        | 480 |
| 11.64 Overload.h File Reference . . . . .           | 481 |
| 11.65 PausedConfig.cpp File Reference . . . . .     | 482 |
| 11.66 PausedConfig.h File Reference . . . . .       | 482 |
| 11.67 PausedView.cpp File Reference . . . . .       | 484 |
| 11.68 PausedView.h File Reference . . . . .         | 484 |
| 11.69 PhysicsEvent.h File Reference . . . . .       | 485 |
| 11.70 PixelColor.h File Reference . . . . .         | 486 |
| 11.71 Player.cpp File Reference . . . . .           | 487 |
| 11.72 Player.h File Reference . . . . .             | 487 |
| 11.73 PlayerStates.h File Reference . . . . .       | 488 |
| 11.74 README.md File Reference . . . . .            | 489 |
| 11.75 ResourceHolder.cpp File Reference . . . . .   | 489 |
| 11.76 ResourceHolder.h File Reference . . . . .     | 489 |
| 11.77 ResourceInserter.h File Reference . . . . .   | 490 |
| 11.77.1 Function Documentation . . . . .            | 491 |
| 11.77.1.1 ResourceInserter() . . . . .              | 491 |
| 11.78 ResourceManager.cpp File Reference . . . . .  | 491 |
| 11.79 ResourceManager.h File Reference . . . . .    | 491 |
| 11.80 Resources.h File Reference . . . . .          | 492 |
| 11.81 RestartConfig.cpp File Reference . . . . .    | 493 |
| 11.82 RestartConfig.h File Reference . . . . .      | 494 |
| 11.83 RestartView.cpp File Reference . . . . .      | 496 |
| 11.84 RestartView.h File Reference . . . . .        | 496 |
| 11.85 Settings.cpp File Reference . . . . .         | 497 |
| 11.85.1 Variable Documentation . . . . .            | 497 |

---

|   |            |
|---|------------|
| 11.85.1.1 audioConfig . . . . .   | 497        |
| 11.86 Settings.h File Reference . . . . .                                   | 498        |
| 11.86.1 Variable Documentation . . . . .                                    | 498        |
| 11.86.1.1 audioConfig . . . . .   | 498        |
| 11.87 Spike.cpp File Reference . . . . .                                    | 499        |
| 11.88 Spike.h File Reference . . . . .                                      | 499        |
| 11.89 State.h File Reference . . . . .                                      | 500        |
| 11.90 StateAbout.cpp File Reference . . . . .                               | 501        |
| 11.91 StateAbout.h File Reference . . . . .                                 | 501        |
| 11.92 StateGame.cpp File Reference . . . . .                                | 502        |
| 11.93 StateGame.h File Reference . . . . .                                  | 502        |
| 11.94 StateID.cpp File Reference . . . . .                                  | 503        |
| 11.95 StateID.h File Reference . . . . .                                    | 504        |
| 11.96 StateKeybinds.cpp File Reference . . . . .                            | 504        |
| 11.97 StateKeybinds.h File Reference . . . . .                              | 505        |
| 11.98 StateMachine.cpp File Reference . . . . .                             | 505        |
| 11.99 StateMachine.h File Reference . . . . .                               | 506        |
| 11.100 StateMapLoader.cpp File Reference . . . . .                          | 507        |
| 11.101 StateMapLoader.h File Reference . . . . .                            | 508        |
| 11.102 StateMenu.cpp File Reference . . . . .                               | 509        |
| 11.103 StateMenu.h File Reference . . . . .                                 | 509        |
| 11.104 StateOptions.cpp File Reference . . . . .                            | 510        |
| 11.105 StateOptions.h File Reference . . . . .                              | 510        |
| 11.106 StatePaused.cpp File Reference . . . . .                             | 511        |
| 11.107 StatePaused.h File Reference . . . . .                               | 511        |
| 11.108 StateRestart.cpp File Reference . . . . .                            | 512        |
| 11.109 StateRestart.h File Reference . . . . .                              | 512        |
| 11.110 TxtReader.cpp File Reference . . . . .                               | 513        |
| 11.111 TxtReader.h File Reference . . . . .                                 | 513        |
| 11.112 Utility.h File Reference . . . . .                                   | 514        |
| 11.113 View.h File Reference . . . . .                                      | 515        |
| 11.114 Window.cpp File Reference . . . . .                                  | 516        |
| 11.115 Window.h File Reference . . . . .                                    | 516        |
| <b>12 Example Documentation</b> . . . . .                                   | <b>519</b> |
| 12.1 C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp . . . . . | 519        |
| 12.2 Calculate . . . . .  | 520        |



# **Chapter 1**

## **My Personal Index Page**

### **1.1 Introduction**

This is the introduction.

### **1.2 Installation**

#### **1.2.1 Step 1: Opening the box**

etc...



## **Chapter 2**

# **platformer-2d**

A customizable crossplatform platformer engine



# Chapter 3

## Module Index

### 3.1 Modules

Here is a list of all modules:

|                     |    |
|---------------------|----|
| Core . . . . .      | 19 |
| Encoder . . . . .   | 21 |
| Entities . . . . .  | 22 |
| Events . . . . .    | 24 |
| GUI . . . . .       | 25 |
| Resources . . . . . | 32 |
| States . . . . .    | 35 |



# Chapter 4

## Namespace Index

### 4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

|            |  |    |
|------------|--|----|
| config     | Compile time customizable settings . . . . .   | 37 |
| config::bg |  | 42 |
| Gui        | Encapsulate common global <a href="#">View Config</a> within an additional namespace . . . . . | 43 |
| Keybinds   | Namespace designated for <a href="#">Keybinds</a> view . . . . .                               | 44 |
| lives      | Encapsulates lives overlay variables . . . . .   | 44 |
| Loader     | Namespace designated for <a href="#">MapLoader</a> view . . . . .                              | 46 |
| Menu       | Namespace designated for <a href="#">Menu</a> view . . . . .                                   | 47 |
| Obj        | Available game entity objects. Used only in <a href="#">StateGame</a> . . . . .                | 48 |
| Options    | Namespace designated for <a href="#">Options</a> view . . . . .                                | 49 |
| Paused     | Namespace designated for <a href="#">Paused</a> view . . . . .                                 | 50 |
| res        | Namespace wrapper over Resources enum IDs . . . . .  | 51 |
| Restart    | Namespace designated for <a href="#">Restart</a> view . . . . .                                | 52 |
| state      | The IDs of the stored states, so the user can access an arbitrary state . . . . .              | 53 |



# Chapter 5

## Hierarchical Index

### 5.1 Class Hierarchy

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

|                                       |     |
|---------------------------------------|-----|
| ActionMap . . . . .                   | 55  |
| AudioCfg . . . . .                    | 59  |
| Bmp . . . . .                         | 64  |
| Camera::Bot . . . . .                 | 73  |
| Camera . . . . .                      | 73  |
| Gui::Config< Widget > . . . . .       | 123 |
| Drawable                              |     |
| Entity . . . . .                      | 130 |
| Block . . . . .                       | 61  |
| HeartCollectible . . . . .            | 165 |
| Life . . . . .                        | 196 |
| Objective . . . . .                   | 254 |
| Player . . . . .                      | 283 |
| Spike . . . . .                       | 325 |
| FpsOverlay . . . . .                  | 150 |
| LivesBorder . . . . .                 | 203 |
| LivesOverlay . . . . .                | 206 |
| Encoder< ReaderKey > . . . . .        | 126 |
| Event . . . . .                       | 143 |
| InputEvent . . . . .                  | 175 |
| exception                             |     |
| MissingFont . . . . .                 | 243 |
| MissingResource< Resource > . . . . . | 245 |
| FileReader< T > . . . . .             | 145 |
| FileReader< int > . . . . .           | 145 |
| TxtReader . . . . .                   | 411 |
| FileReader< PixelColor > . . . . .    | 145 |
| BmpReader . . . . .                   | 65  |
| Game . . . . .                        | 156 |
| IConfig< E > . . . . .                | 170 |
| IConfig< Btn > . . . . .              | 170 |
| Keybinds::Config . . . . .            | 98  |
| Loader::Config . . . . .              | 103 |
| Menu::Config . . . . .                | 107 |

|   |     |
|---|-----|
| Options::Config . . . . .                               | 111 |
| Paused::Config . . . . .                                | 115 |
| Restart::Config . . . . .                               | 119 |
| ILoader . . . . .                                       | 172 |
| MapLoader< FileType > . . . . .                         | 214 |
| Camera::Left . . . . .                                  | 195 |
| mapListOfHelper< T > . . . . .                          | 212 |
| MapNameValidator . . . . .                              | 232 |
| Camera::None . . . . .                                  | 253 |
| Overload . . . . .                                      | 270 |
| PhysicsEvent . . . . .                                  | 278 |
| CollisionEvent . . . . .                                | 89  |
| MovementEvent . . . . .                                 | 248 |
| PixelColor . . . . .                                    | 280 |
| ResourceHolder< Key, Resource > . . . . .               | 299 |
| ResourceHolder< res::Music, sf::Music > . . . . .       | 299 |
| ResourceHolder< res::Sound, sf::SoundBuffer > . . . . . | 299 |
| ResourceHolder< res::Texture, sf::Texture > . . . . .   | 299 |
| ResourceInserter< Key, Args > . . . . .                 | 307 |
| ResourceManager . . . . .                               | 309 |
| Camera::Right . . . . .                                 | 325 |
| State . . . . .   | 330 |
| StateAbout . . . . .                                    | 333 |
| StateGame . . . . .                                     | 338 |
| StateKeybinds . . . . .                                 | 359 |
| StateMapLoader . . . . .                                | 373 |
| StateMenu . . . . .                                     | 382 |
| StateOptions . . . . .                                  | 390 |
| StatePaused . . . . .                                   | 397 |
| StateRestart . . . . .                                  | 404 |
| StateMachine . . . . .                                  | 366 |
| Camera::Top . . . . .                                   | 410 |
| Transformable   |     |
| Entity . . . . .  | 130 |
| Txt . . . . .   | 410 |
| View< TWidgetPtr > . . . . .                            | 417 |
| View<> . . . . .  | 417 |
| KeybindsView . . . . .                                  | 182 |
| MapLoaderView . . . . .                                 | 220 |
| MenuView . . . . .                                      | 236 |
| OptionsView . . . . .                                   | 257 |
| PausedView . . . . .                                    | 271 |
| RestartView . . . . .                                   | 315 |
| Window . . . . .  | 422 |
| Ts  |     |
| overload< Ts > . . . . .                                | 269 |

# Chapter 6

## Class Index

### 6.1 Class List

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

|   |     |
|---|-----|
| ActionMap   | 55  |
| AudioCfg  |     |
| Responsible for storing audio config enable   | 59  |
| Block   | 61  |
| Used to represent collidable game blocks  |     |
| Bmp   | 64  |
| Empty struct used to determine file type in MapLoader variant   |     |
| BmpReader   | 65  |
| Reads the .bmp file into the vector of PixelColor   |     |
| Camera::Bot   | 73  |
| Empty struct denoting previously solved on other axis was Bot collision   |     |
| Camera  | 73  |
| Responsible for viewing only a sensible, displayable part of the World to the screen. Handles all Camera collisions |     |
| CollisionEvent  | 89  |
| CollisionEvent class responsible for solves player<->collidable collisions in <b>axis-independent way</b>           |     |
| Keybinds::Config  | 98  |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to KeybindsView                           |     |
| Loader::Config  | 103 |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to MapLoaderView                          |     |
| Menu::Config  | 107 |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to MenuView                               |     |
| Options::Config   | 111 |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to OptionsView                            |     |
| Paused::Config  | 115 |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to PausedView                             |     |
| Restart::Config   | 119 |
| Deriving from IConfig<Btn>, holds meaningful view configurations specific to RestartView                            |     |
| Gui::Config< Widget >   | 123 |
| A shared gui config. Contains common settings   |     |
| Encoder< ReaderKey >  | 126 |
| Encodes PixelColor (bmp) or int (txt) into Obj::Entity entities, depending on the variant type                      |     |
| Entity  | 130 |
| A base drawable Entity class. All drawable entities (game objects) inherit from it                                  |     |

|                             |   |
|-----------------------------|---|
| Event                       |   |
| Event                       | class with <code>update()</code> pure virtual method handling trivial events . . . . .                                    |
| FileReader< T >             | Abstract File Reader class template . . . . .   |
| FpsOverlay                  | Prints the Fps to the screen . . . . .  |
| Game                        | Container for player actions keybinds . . . . .   |
| HeartCollectible            | Used to represent the collectible hearts (player lives) . . . . .   |
| IConfig< E >                | Every specific Config inherits from this struct . . . . .   |
| ILoader                     | A base interface for a map loader . . . . .   |
| InputEvent                  | Handles the user input within <code>StateGame</code> class. Uses <code>ActionMap</code> . . . . .                         |
| KeybindsView                | <code>View</code> class used to draw gui within <code>StateKeybinds</code> . . . . .                                      |
| Camera::Left                | Empty struct denoting previously solved on other axis was <code>Left</code> collision . . . . .                           |
| Life                        | Used to represent the player life overlay <code>item</code> as a hearts in corner of the screen(player lives) . . . . .   |
| LivesBorder                 | A border around the <code>LivesOverlay</code> . . . . .   |
| LivesOverlay                | Encapsulates player lives overlay . . . . .   |
| mapListOfHelper< T >        | A helper for mapListOf converter . . . . .  |
| MapLoader< FileType >       | Load the map file . . . . .   |
| MapLoaderView               | <code>View</code> class used to draw gui within <code>StateMapLoader</code> . . . . .                                     |
| MapNameValidator            | A utility class validating if a given map file name is valid . . . . .  |
| MenuView                    | <code>View</code> class used to draw gui within <code>StateMenu</code> . . . . .  |
| MissingFont                 | Custom exception class . . . . .  |
| MissingResource< Resource > | Meaningful response error when resource of given type was missing . . . . .   |
| MovementEvent               | <code>MovementEvent</code> class responsible for updating the player movement in an <b>axis-independent way</b> . . . . . |
| Camera::None                | Empty struct denoting there was previously <code>None</code> collision . . . . .  |
| Objective                   | Used to represent the game <code>Objective</code> . Player intersecting <code>Objective</code> wins the game . . . . .    |
| OptionsView                 | <code>View</code> class used to draw gui within <code>StateOptions</code> . . . . .                                       |
| overload< Ts >              |   |
| Overload                    | Overload lambda <i>hack</i> for variant std::visit . . . . .  |
| PausedView                  | <code>View</code> class used to draw gui within <code>StatePaused</code> . . . . .  |
| PhysicsEvent                | Specified Base <code>PhysicsEvent</code> class handling physics-related events on two separate axes . . . . .             |
| PixelColor                  | Container for one .bmp BGR pixel . . . . .  |

|  |   |     |
|--|---|-----|
| <b>Player</b>                                | Controls the player behaviour . . . . .   | 283 |
| <b>ResourceHolder&lt; Key, Resource &gt;</b> | A generic <a href="#">ResourceHolder</a> storing resources of type <a href="#">Resource</a> , accessed by <a href="#">Key</a> . . . . . | 299 |
| <b>ResourceInserter&lt; Key, Args &gt;</b>   | A proxy class used to handle 2+ arguments in operator+=, which takes only 1 argument . . . . .  | 307 |
| <b>ResourceManager</b>                       | <a href="#">ResourceManager</a> is a class responsible for managing all kinds of <a href="#">ResourceHolders</a> . . . . .              | 309 |
| <b>RestartView</b>                           | <a href="#">View</a> class used to draw gui within <a href="#">StateRestart</a> . . . . .   | 315 |
| <b>Camera::Right</b>                         | Empty struct denoting previously solved on other axis was <a href="#">Right</a> collision . . . . .                                     | 325 |
| <b>Spike</b>                                 | Used to represent the spikes. Intersecting spike kills the <a href="#">Player</a> . . . . .   | 325 |
| <b>State</b>                                 | < Pass-through . . . . .  | 330 |
| <b>StateAbout</b>                            | Encapsulates the logic of about menu within one <a href="#">StateAbout</a> class. Displays the about texture .                          | 333 |
| <b>StateGame</b>                             | Encapsulates the main game logic within one <a href="#">StateGame</a> class . . . . .   | 338 |
| <b>StateKeybinds</b>                         | Encapsulates the logic of rebinding menu within one <a href="#">StateKeybinds</a> class . . . . .                                       | 359 |
| <b>StateMachine</b>                          | Container for all game States and holds current state pointer. Updates <a href="#">State</a> logic every frame                          | 366 |
| <b>StateMapLoader</b>                        | Encapsulates the logic of map loader menu within one <a href="#">StateMapLoader</a> class . . . . .                                     | 373 |
| <b>StateMenu</b>                             | Encapsulates the logic of main menu within one <a href="#">StateRestart</a> class . . . . .   | 382 |
| <b>StateOptions</b>                          | Encapsulates the logic of options menu within one <a href="#">StateOptions</a> class . . . . .  | 390 |
| <b>StatePaused</b>                           | Encapsulates the logic of paused menu within one <a href="#">StatePaused</a> class . . . . .  | 397 |
| <b>StateRestart</b>                          | Encapsulates the logic of restart menu within one <a href="#">StateRestart</a> class . . . . .  | 404 |
| <b>Camera::Top</b>                           | Empty struct denoting previously solved on other axis was <a href="#">Top</a> collision . . . . .                                       | 410 |
| <b>Txt</b>                                   | Empty struct used to determine file type in <a href="#">MapLoader</a> variant . . . . .   | 410 |
| <b>TxtReader</b>                             | Reads the .txt file into the vector of int . . . . .  | 411 |
| <b>View&lt; TWidgetPtr &gt;</b>              | . . . . .   | 417 |
| <b>Window</b>                                | Responsible for drawing onto the screen . . . . .   | 422 |



# Chapter 7

## File Index

### 7.1 File List

Here is a list of all files with brief descriptions:

|                      |     |
|----------------------|-----|
| Action.h             | 431 |
| ActionMap.cpp        | 432 |
| ActionMap.h          | 432 |
| Block.cpp            | 433 |
| Block.h              | 434 |
| BmpReader.cpp        | 435 |
| BmpReader.h          | 435 |
| Camera.cpp           | 436 |
| Camera.h             | 436 |
| CollisionEvent.cpp   | 438 |
| CollisionEvent.h     | 438 |
| Config.h             | 439 |
| Configuration.cpp    | 440 |
| Configuration.h      | 441 |
| Encoder.cpp          | 442 |
| Encoder.h            | 443 |
| Entity.cpp           | 444 |
| Entity.h             | 444 |
| Event.h              | 446 |
| FileReader.cpp       | 446 |
| FileReader.h         | 447 |
| FpsOverlay.cpp       | 448 |
| FpsOverlay.h         | 448 |
| Game.cpp             | 449 |
| Game.h               | 450 |
| HeartCollectible.cpp | 450 |
| HeartCollectible.h   | 451 |
| IConfig.h            | 452 |
| InputEvent.cpp       | 453 |
| InputEvent.h         | 453 |
| KeybindsConfig.cpp   | 454 |
| KeybindsConfig.h     | 455 |
| KeybindsView.cpp     | 457 |
| KeybindsView.h       | 457 |
| Life.cpp             | 458 |

|                      |     |
|----------------------|-----|
| Life.h               | 459 |
| LivesBorder.cpp      | 460 |
| LivesBorder.h        | 460 |
| LivesOverlay.cpp     | 461 |
| LivesOverlay.h       | 462 |
| LivesOverlayConfig.h | 463 |
| main.cpp             | 464 |
| MapLoader.cpp        | 465 |
| MapLoader.h          | 465 |
| MapLoaderConfig.cpp  | 466 |
| MapLoaderConfig.h    | 467 |
| MapLoaderView.cpp    | 469 |
| MapLoaderView.h      | 469 |
| MapNameValidator.cpp | 470 |
| MapNameValidator.h   | 471 |
| MenuConfig.cpp       | 471 |
| MenuConfig.h         | 472 |
| MenuView.cpp         | 474 |
| MenuView.h           | 474 |
| MovementEvent.cpp    | 475 |
| MovementEvent.h      | 476 |
| Objective.cpp        | 477 |
| Objective.h          | 477 |
| OptionsConfig.cpp    | 478 |
| OptionsConfig.h      | 479 |
| OptionsView.cpp      | 480 |
| OptionsView.h        | 480 |
| Overload.h           | 481 |
| PausedConfig.cpp     | 482 |
| PausedConfig.h       | 482 |
| PausedView.cpp       | 484 |
| PausedView.h         | 484 |
| PhysicsEvent.h       | 485 |
| PixelColor.h         | 486 |
| Player.cpp           | 487 |
| Player.h             | 487 |
| PlayerStates.h       | 488 |
| ResourceHolder.cpp   | 489 |
| ResourceHolder.h     | 489 |
| ResourceInserter.h   | 490 |
| ResourceManager.cpp  | 491 |
| ResourceManager.h    | 491 |
| Resources.h          | 492 |
| RestartConfig.cpp    | 493 |
| RestartConfig.h      | 494 |
| RestartView.cpp      | 496 |
| RestartView.h        | 496 |
| Settings.cpp         | 497 |
| Settings.h           | 498 |
| Spike.cpp            | 499 |
| Spike.h              | 499 |
| State.h              | 500 |
| StateAbout.cpp       | 501 |
| StateAbout.h         | 501 |
| StateGame.cpp        | 502 |
| StateGame.h          | 502 |
| StateID.cpp          | 503 |
| StateID.h            | 504 |

|                    |     |
|--------------------|-----|
| StateKeybinds.cpp  | 504 |
| StateKeybinds.h    | 505 |
| StateMachine.cpp   | 505 |
| StateMachine.h     | 506 |
| StateMapLoader.cpp | 507 |
| StateMapLoader.h   | 508 |
| StateMenu.cpp      | 509 |
| StateMenu.h        | 509 |
| StateOptions.cpp   | 510 |
| StateOptions.h     | 510 |
| StatePaused.cpp    | 511 |
| StatePaused.h      | 511 |
| StateRestart.cpp   | 512 |
| StateRestart.h     | 512 |
| TxtReader.cpp      | 513 |
| TxtReader.h        | 513 |
| Utility.h          | 514 |
| View.h             | 515 |
| Window.cpp         | 516 |
| Window.h           | 516 |



# Chapter 8

## Module Documentation

### 8.1 Core

Classes that are core to the engine, yet don't belong in a dedicated package.

#### Namespaces

- [config](#)  
*Compile time customizable settings.*

#### Classes

- class [Game](#)  
*Container for player actions keybinds.*
- class [Camera](#)  
*Responsible for viewing only a sensible, displayable part of the World to the screen. Handles all [Camera](#) collisions.*
- struct [Camera::None](#)  
*Empty struct denoting there was previously [None](#) collision.*
- struct [Camera::Left](#)  
*Empty struct denoting previously solved on other axis was [Left](#) collision.*
- struct [Camera::Right](#)  
*Empty struct denoting previously solved on other axis was [Right](#) collision.*
- struct [Camera::Top](#)  
*Empty struct denoting previously solved on other axis was [Top](#) collision.*
- struct [Camera::Bot](#)  
*Empty struct denoting previously solved on other axis was [Bot](#) collision.*
- struct [Overload](#)  
*Overload lambda hack for variant std::visit.*
- class [AudioCfg](#)  
*Responsible for storing audio config enable.*
- class [Window](#)  
*Responsible for drawing onto the screen.*

## Enumerations

- enum `Action` {  
  `Action::Jump`,  
  `Action::GoLeft`,  
  `Action::GoRight`,  
  `Action::NONE` }

*An enum containing available game actions taken by the player during the game.*

### 8.1.1 Detailed Description

Classes that are core to the engine, yet don't belong in a dedicated package.

### 8.1.2 Enumeration Type Documentation

#### 8.1.2.1 Action

```
enum Action [strong]
```

An enum containing available game actions taken by the player during the game.

The default action keybinds are:

- **W** for jumping
- **A** for moving left
- **D** for moving right

#### Note

An action can be rebound to a different key inside [StateKeybinds](#).

#### Enumerator

|                      |  |
|----------------------|--|
| <code>Jump</code>    |  |
| <code>GoLeft</code>  |  |
| <code>GoRight</code> |  |
| <code>NONE</code>    |  |

## 8.2 Encoder

Classes that are responsible for **encoding** the input file ([Bmp](#) or [Txt](#)) into complete game World.

### Namespaces

- [Obj](#)  
*Available game entity objects. Used only in [StateGame](#).*

### Classes

- class [BmpReader](#)  
*Reads the .bmp file into the vector of [PixelColor](#).*
- class [Encoder< ReaderKey >](#)  
*Encodes [PixelColor](#) (bmp) or int (txt) into [Obj::Entity](#) entities, depending on the variant type.*
- class [FileReader< T >](#)  
*Abstract File Reader class template.*
- class [ILoader](#)  
*A base interface for a map loader.*
- struct [Bmp](#)  
*Empty struct used to determine file type in [MapLoader](#) variant.*
- struct [Txt](#)  
*Empty struct used to determine file type in [MapLoader](#) variant.*
- class [MapLoader< FileType >](#)  
*Load the map file.*
- struct [PixelColor](#)  
*Container for one .bmp BGR pixel.*
- class [TxtReader](#)  
*Reads the .txt file into the vector of int.*
- class [MapNameValidator](#)  
*A utility class validating if a given map file name is valid.*

### 8.2.1 Detailed Description

Classes that are responsible for **encoding** the input file ([Bmp](#) or [Txt](#)) into complete game World.

## 8.3 Entities

Drawable game objects, drawn inside [StateGame](#).

### Namespaces

- [lives](#)

*Encapsulates lives overlay variables.*

### Classes

- class [Block](#)

*Used to represent collidable game blocks.*

- class [Entity](#)

*A base drawable [Entity](#) class. All drawable entities (game objects) inherit from it.*

- class [HeartCollectible](#)

*Used to represent the collectible hearts (player lives).*

- class [Life](#)

*Used to represent the player life overlay **item** as a hearts in corner of the screen(player lives).*

- class [Objective](#)

*Used to represent the game [Objective](#). Player intersecting [Objective](#) wins the game.*

- class [MissingFont](#)

*Custom exception class.*

- class [FpsOverlay](#)

*Prints the Fps to the screen.*

- class [LivesBorder](#)

*A border around the [LivesOverlay](#).*

- class [LivesOverlay](#)

*Encapsulates player lives overlay.*

- class [Player](#)

*Controls the player behaviour.*

- class [Spike](#)

*Used to represent the spikes. Intersecting spike kills the [Player](#).*

### Enumerations

- enum [MovingState](#) {

[MovingState::movingLeft](#),  
[MovingState::movingRight](#),  
[MovingState::standing](#) }

*Represents the horizontal MovingState of the player. [Player](#) is in one state at a time.*

- enum [JumpingState](#) {

[JumpingState::onGround](#),  
[JumpingState::jumping](#),  
[JumpingState::gravity](#) }

*Represents the JumpingState of the player. [Player](#) is in one state at a time.*

### 8.3.1 Detailed Description

Drawable game objects, drawn inside [StateGame](#).

### 8.3.2 Enumeration Type Documentation

#### 8.3.2.1 JumpingState

```
enum JumpingState [strong]
```

Represents the JumpingState of the player. [Player](#) is in one state at a time.

Enumerator

|          |  |
|----------|--|
| onGround |  |
| jumping  |  |
| gravity  |  |

#### 8.3.2.2 MovingState

```
enum MovingState [strong]
```

Represents the horizontal MovingState of the player. [Player](#) is in one state at a time.

Enumerator

|             |  |
|-------------|--|
| movingLeft  |  |
| movingRight |  |
| standing    |  |

## 8.4 Events

Events updating the game logic within [StateGame](#) class.

### Classes

- class [CollisionEvent](#)  
*[CollisionEvent](#) class responsible for solves player<->collidable collisions in **axis-independent way**.*
- class [Event](#)  
*[Event](#) class with [update\(\)](#) pure virtual method handling trivial events.*
- class [InputEvent](#)  
*Handles the user input within [StateGame](#) class. Uses [ActionMap](#).*
- class [MovementEvent](#)  
*[MovementEvent](#) class responsible for updating the player movement in an **axis-independent way**.*
- class [PhysicsEvent](#)  
*Specified Base [PhysicsEvent](#) class handling physics-related events on two separate axises.*

### 8.4.1 Detailed Description

Events updating the game logic within [StateGame](#) class.

## 8.5 GUI

Classes responsible for [View](#) layer inside varying Menus.

### Namespaces

- [Keybinds](#)  
*Namespace designated for [Keybinds](#) view.*
- [Loader](#)  
*Namespace designated for [MapLoader](#) view.*
- [Menu](#)  
*Namespace designated for [Menu](#) view.*
- [Options](#)  
*Namespace designated for [Options](#) view.*
- [Paused](#)  
*Namespace designated for [Paused](#) view.*
- [Restart](#)  
*Namespace designated for [Restart](#) view.*
- [Gui](#)  
*Encapsulate common global [View Config](#) within an additional namespace.*

### Classes

- struct [mapListOfHelper< T >](#)  
*A helper for [mapListOf](#) converter.*
- struct [Keybinds::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [KeybindsView](#).*
- struct [Loader::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MapLoaderView](#).*
- struct [Menu::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MenuView](#).*
- struct [Options::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [OptionsMenu](#).*
- struct [Paused::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [PausedView](#).*
- struct [Restart::Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [RestartView](#).*
- struct [Gui::Config< Widget >](#)  
*A shared gui config. Contains common settings.*
- struct [IConfig< E >](#)  
*Every specific Config inherits from this struct.*
- class [KeybindsView](#)  
*View class used to draw gui within [StateKeybinds](#).*
- class [MapLoaderView](#)  
*View class used to draw gui within [StateMapLoader](#).*
- class [MenuView](#)  
*View class used to draw gui within [StateMenu](#).*
- class [OptionsMenu](#)  
*View class used to draw gui within [StateOptions](#).*
- class [PausedView](#)  
*View class used to draw gui within [StatePaused](#).*
- class [RestartView](#)  
*View class used to draw gui within [StateRestart](#).*

## Enumerations

- enum Keybinds::Btn {
 Keybinds::Btn::Jump,
 Keybinds::Btn::RunLeft,
 Keybinds::Btn::RunRight,
 Keybinds::Btn::GoBack,
 Keybinds::Btn::SIZE }

*Keybinds buttons IDs.*

## Functions

- template<typename E >  
constexpr auto to\_underlying (E e) noexcept  
  
*Converts an argument (Enum) to its underlying type.*
- template<typename E >  
constexpr std::optional< int > tolnt (E e) noexcept  
  
*Converts an argument (Enum) to integer.*
- template<typename T >  
constexpr mapListOfHelper< T > mapListOf (T &item)

## Variables

- constexpr std::initializer\_list< Keybinds::Btn > Keybinds::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*
- constexpr std::initializer\_list< Loader::Btn > Loader::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*
- constexpr std::initializer\_list< Menu::Btn > Menu::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*
- constexpr std::initializer\_list< Options::Btn > Options::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*
- constexpr std::initializer\_list< Paused::Btn > Paused::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*
- constexpr std::initializer\_list< Restart::Btn > Restart::Buttons  
  
*An additional initializer list to help mapping Btn names to Strings.*

### 8.5.1 Detailed Description

Classes responsible for [View](#) layer inside varying Menus.

### 8.5.2 Enumeration Type Documentation

### 8.5.2.1 Btn

```
enum Keybinds::Btn [strong]
```

[Keybinds](#) buttons IDs.

[Restart](#) buttons IDs.

[Paused](#) buttons IDs.

[Options](#) buttons IDs.

[Menu](#) buttons IDs.

[MapLoader](#) buttons IDs.

#### Note

C++ standard libraries do not have a simple solution to convert an enum to string

#### Enumerator

|          |  |
|----------|--|
| Jump     |  |
| RunLeft  |  |
| RunRight |  |
| GoBack   |  |
| SIZE     |  |

### 8.5.3 Function Documentation

#### 8.5.3.1 mapListOf()

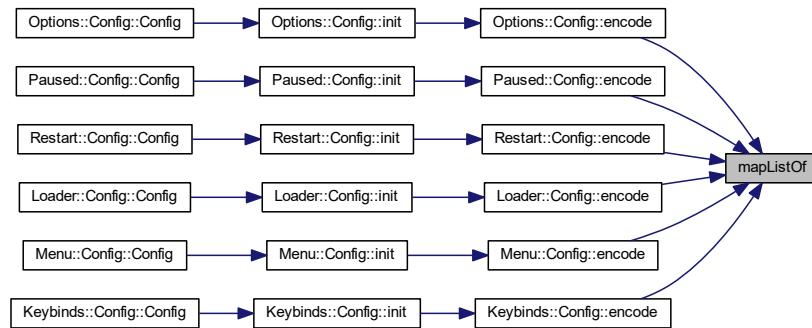
```
template<typename T >
constexpr mapListOfHelper<T> mapListOf (
    T & item ) [constexpr]
```

mapListOf converter, alternatively use Boost implementation.

**Note**

Using Boost implementation adds extra library complexity.

Here is the caller graph for this function:

**8.5.3.2 `to_underlying()`**

```
template<typename E >
constexpr auto to_underlying (
    E e ) [constexpr], [noexcept]
```

Converts an argument (Enum) to its underlying type.

**Parameters**

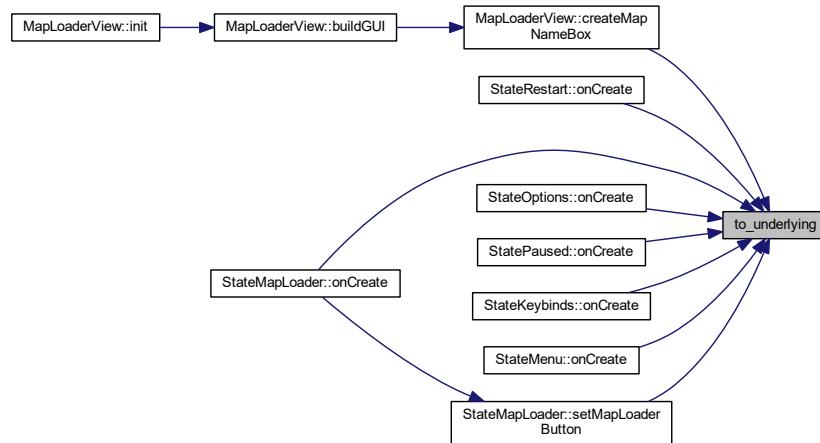
|           |      |
|-----------|------|
| <i>An</i> | enum |
|-----------|------|

**Returns**

An underlying type

Templates meta-programming.

Used to access [View](#) widgets. Here is the caller graph for this function:



### 8.5.3.3 `toInt()`

```
template<typename E >
constexpr std::optional<int> toInt (
    E e ) [constexpr], [noexcept]
```

Converts an argument (Enum) to integer.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <code>an</code> | <code>enum</code> |
|-----------------|-------------------|

#### Returns

an integer value of enum item

#### Note

uses if constexpr

## 8.5.4 Variable Documentation

#### 8.5.4.1 Buttons [1/6]

```
Restart::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::PlayAgain,  
    Btn::Menu  
}
```

An additional initializer list to help mapping Btn names to Strings.

#### 8.5.4.2 Buttons [2/6]

```
Paused::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::resume,  
    Btn::options,  
    Btn::menu  
}
```

An additional initializer list to help mapping Btn names to Strings.

#### 8.5.4.3 Buttons [3/6]

```
Options::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::Keybinds,  
    Btn::GoBack  
}
```

An additional initializer list to help mapping Btn names to Strings.

#### 8.5.4.4 Buttons [4/6]

```
Loader::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::loadMap,  
    Btn::openEditor,  
    Btn::goBack  
}
```

An additional initializer list to help mapping Btn names to Strings.

#### 8.5.4.5 Buttons [5/6]

```
Menu::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::newGame,  
    Btn::loadGame,  
    Btn::options,  
    Btn::about,  
    Btn::exit  
}
```

An additional initializer list to help mapping Btn names to Strings.

#### 8.5.4.6 Buttons [6/6]

```
Keybinds::Buttons [constexpr]
```

**Initial value:**

```
= {  
    Btn::Jump,  
    Btn::RunLeft,  
    Btn::RunRight,  
    Btn::GoBack  
}
```

An additional initializer list to help mapping Btn names to Strings.

## 8.6 Resources

Classes responsible for loading, storing and retrieving the external resources.

### Namespaces

- `res`

*Namespace wrapper over Resources enum IDs.*

### Classes

- class `MissingResource< Resource >`  
*Meaningful response error when resource of given type was missing.*
- class `ResourceHolder< Key, Resource >`  
*A generic `ResourceHolder` storing resources of type `Resource`, accessed by `Key`.*
- class `ResourceInserter< Key, Args >`  
*A proxy class used to handle 2+ arguments in operator+=, which takes only 1 argument.*
- class `ResourceManager`  
*`ResourceManager` is a class responsible for managing all kinds of `ResourceHolders`.*

### Enumerations

- enum `res::Texture {`  
  `res::Texture::BlockRed,`  
  `res::Texture::BlockBlue,`  
  `res::Texture::BlockBrown,`  
  `res::Texture::BlockGray,`  
  `res::Texture::BlockGreen,`  
  `res::Texture::BlockPurple,`  
  `res::Texture::BlockYellow,`  
  `res::Texture::BlockCrate,`  
  `res::Texture::Water1,`  
  `res::Texture::Water2,`  
  `res::Texture::Sign1,`  
  `res::Texture::Sign2,`  
  `res::Texture::Player,`  
  `res::Texture::PlayerLeft,`  
  `res::Texture::PlayerRight,`  
  `res::Texture::Objective,`  
  `res::Texture::Heart,`  
  `res::Texture::HeartEmpty,`  
  `res::Texture::BgGame,`  
  `res::Texture::BgAbout,`  
  `res::Texture::Spike,`  
  `res::Texture::OptionsLeftTopCorner,`  
  `res::Texture::OptionsLeftBotCorner,`  
  `res::Texture::OptionsRightBotCorner,`  
  `res::Texture::OptionsRightTopCorner,`  
  `res::Texture::GameLost,`  
  `res::Texture::GameWon }`

*Available Texture IDs.*

- enum `res::Sound` {  
  `res::Sound::Collect`,  
  `res::Sound::BtnHover`,  
  `res::Sound::Death`,  
  `res::Sound::WinGame` }  
    *Available Sound IDs.*
- enum `res::Music` { `res::Music::Background` }  
    *Available Music IDs.*

### 8.6.1 Detailed Description

Classes responsible for loading, storing and retrieving the external resources.

### 8.6.2 Enumeration Type Documentation

#### 8.6.2.1 Music

```
enum res::Music [strong]
```

Available Music IDs.

Enumerator

|            |                                 |
|------------|---------------------------------|
| Background | <input type="button" value=""/> |
|------------|---------------------------------|

#### 8.6.2.2 Sound

```
enum res::Sound [strong]
```

Available Sound IDs.

Enumerator

|          |                                 |
|----------|---------------------------------|
| Collect  | <input type="button" value=""/> |
| BtnHover | <input type="button" value=""/> |
| Death    | <input type="button" value=""/> |
| WinGame  | <input type="button" value=""/> |

#### 8.6.2.3 Texture

```
enum res::Texture [strong]
```

Available Texture IDs.

Enumerator

|                       |  |
|-----------------------|--|
| BlockRed              |  |
| BlockBlue             |  |
| BlockBrown            |  |
| BlockGray             |  |
| BlockGreen            |  |
| BlockPurple           |  |
| BlockYellow           |  |
| BlockCrate            |  |
| Water1                |  |
| Water2                |  |
| Sign1                 |  |
| Sign2                 |  |
| Player                |  |
| PlayerLeft            |  |
| PlayerRight           |  |
| Objective             |  |
| Heart                 |  |
| HeartEmpty            |  |
| BgGame                |  |
| BgAbout               |  |
| Spike                 |  |
| OptionsLeftTopCorner  |  |
| OptionsLeftBotCorner  |  |
| OptionsRightBotCorner |  |
| OptionsRightTopCorner |  |
| GameLost              |  |
| GameWon               |  |

## 8.7 States

Drawable game objects, drawn inside [StateGame](#).

### Namespaces

- [state](#)  
*The IDs of the stored states, so the user can access an arbitrary state.*

### Classes

- class [State](#)  
*< Pass-through*
- class [StateMachine](#)  
*Container for all game States and holds current state pointer. Updates [State](#) logic every frame.*
- class [StateAbout](#)  
*Encapsulates the logic of about menu within one [StateAbout](#) class. Displays the about texture.*
- class [StateGame](#)  
*Encapsulates the main game logic within one [StateGame](#) class.*
- class [StateKeybinds](#)  
*Encapsulates the logic of rebinding menu within one [StateKeybinds](#) class.*
- class [StateMapLoader](#)  
*Encapsulates the logic of map loader menu within one [StateMapLoader](#) class.*
- class [StateMenu](#)  
*Encapsulates the logic of main menu within one [StateRestart](#) class.*
- class [StateOptions](#)  
*Encapsulates the logic of options menu within one [StateOptions](#) class.*
- class [StatePaused](#)  
*Encapsulates the logic of paused menu within one [StatePaused](#) class.*
- class [StateRestart](#)  
*Encapsulates the logic of restart menu within one [StateRestart](#) class.*

### 8.7.1 Detailed Description

Drawable game objects, drawn inside [StateGame](#).



# Chapter 9

## Namespace Documentation

### 9.1 config Namespace Reference

Compile time customizable settings.

#### Namespaces

- `bg`

#### Variables

- `constexpr static Keyboard::Key defaultJumpKey = Keyboard::W`
- `constexpr static Keyboard::Key defaultRunLeftKey = Keyboard::A`
- `constexpr static Keyboard::Key defaultRunRightKey = Keyboard::D`
- `constexpr static string_view mapEditorPath = "index.html"`
- `constexpr static string_view defaultMapName = "map.bmp"`
- `constexpr static string_view gameMusic = "../resources/music/Background.ogg"`
- `constexpr static auto defaultSoundVolume = 30.0f`
- `constexpr static string_view widgetsFontName = "../resources/Fonts/CascadiaCode.ttf"`
- `constexpr static string_view keybindsFontName = "../resources/Fonts/NeuropolX.ttf"`
- `constexpr static string_view programIcon = "../resources/Icons/Program_Icon.png"`
- `constexpr static string_view jumpIcon = "../resources/Icons/Jump.png"`
- `constexpr static string_view runLeftIcon = "../resources/Icons/Left_Arrow.png"`
- `constexpr static string_view runRightIcon = "../resources/Icons/Right_Arrow.png"`
- `constexpr static string_view largeBgIcon = "../resources/Icons/Keybinds_Large_Icon.png"`
- `constexpr static string_view sideBorder = "../resources/Misc/Options_Side_Border.png"`
- `constexpr static string_view pausedTextImage = "../resources/Misc/Game_Paused_Texture.png"`
- `constexpr static string_view rebindingBg = "../resources/Misc/Rebinding_Key.png"`
- `constexpr static auto horizontalVelocity = 300.0f`
- `constexpr static auto initialJumpVelocity = -680.0f`
- `constexpr static auto terminalVelocity = 1000.0f`
- `constexpr static auto gravity = 2000.0f`
- `constexpr static auto hitCeilingVelocity = 10.0f`
- `int blocksCountWidth = 0`
- `int blocksCountHeight = 0`
- `constexpr static auto entityWidth = 32`
- `constexpr static auto entityHeight = 32`
- `constexpr static auto windowHeight = entityWidth * 20`
- `constexpr static auto windowHeight = entityHeight * 18`
- `constexpr static auto maxFps = 3000`
- `bool playerWon = false`

### 9.1.1 Detailed Description

Compile time customizable settings.

file

### 9.1.2 Variable Documentation

#### 9.1.2.1 blocksCountHeight

```
int config::blocksCountHeight = 0
```

Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

#### 9.1.2.2 blocksCountWidth

```
int config::blocksCountWidth = 0
```

Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

#### 9.1.2.3 defaultJumpKey

```
constexpr static Keyboard::Key config::defaultJumpKey = Keyboard::W [static], [constexpr]
```

#### 9.1.2.4 defaultMapName

```
constexpr static string_view config::defaultMapName = "map.bmp" [static], [constexpr]
```

#### 9.1.2.5 defaultRunLeftKey

```
constexpr static Keyboard::Key config::defaultRunLeftKey = Keyboard::A [static], [constexpr]
```

### 9.1.2.6 defaultRunRightKey

```
constexpr static Keyboard::Key config::defaultRunRightKey = Keyboard::D [static], [constexpr]
```

### 9.1.2.7 defaultSoundVolume

```
constexpr static auto config::defaultSoundVolume = 30.0f [static], [constexpr]
```

### 9.1.2.8 entityHeight

```
constexpr static auto config::entityHeight = 32 [static], [constexpr]
```

### 9.1.2.9 entityWidth

```
constexpr static auto config::entityWidth = 32 [static], [constexpr]
```

### 9.1.2.10 gameMusic

```
constexpr static string_view config::gameMusic = "../resources/music/Background.ogg" [static], [constexpr]
```

### 9.1.2.11 gravity

```
constexpr static auto config::gravity = 2000.0f [static], [constexpr]
```

### 9.1.2.12 hitCeilingVelocity

```
constexpr static auto config::hitCeilingVelocity = 10.0f [static], [constexpr]
```

### 9.1.2.13 horizontalVelocity

```
constexpr static auto config::horizontalVelocity = 300.0f [static], [constexpr]
```

### 9.1.2.14 initialJumpVelocity

```
constexpr static auto config::initialJumpVelocity = -680.0f [static], [constexpr]
```

### 9.1.2.15 jumpIcon

```
constexpr static string_view config::jumpIcon = "../resources/Icons/Jump.png" [static], [constexpr]
```

### 9.1.2.16 keybindsFontName

```
constexpr static string_view config::keybindsFontName = "../resources/Fonts/NeuropolX.ttf"  
[static], [constexpr]
```

### 9.1.2.17 largeBgIcon

```
constexpr static string_view config::largeBgIcon = "../resources/Icons/Keybinds_Large_Icon.  
png" [static], [constexpr]
```

### 9.1.2.18 mapEditorPath

```
constexpr static string_view config::mapEditorPath = "index.html" [static], [constexpr]
```

### 9.1.2.19 maxFps

```
constexpr static auto config::maxFps = 3000 [static], [constexpr]
```

**9.1.2.20 pausedTextImage**

```
constexpr static string_view config::pausedTextImage = "../resources/Misc/Game_Paused_Texture.png" [static], [constexpr]
```

**9.1.2.21 playerWon**

```
bool config::playerWon = false
```

Emphasize frame rate independent movement

**9.1.2.22 programIcon**

```
constexpr static string_view config::programIcon = "../resources/Icons/Program_Icon.png" [static], [constexpr]
```

**9.1.2.23 rebindingBg**

```
constexpr static string_view config::rebindingBg = "../resources/Misc/Rebinding_Key.png" [static], [constexpr]
```

**9.1.2.24 runLeftIcon**

```
constexpr static string_view config::runLeftIcon = "../resources/Icons/Left_Arrow.png" [static], [constexpr]
```

**9.1.2.25 runRightIcon**

```
constexpr static string_view config::runRightIcon = "../resources/Icons/Right_Arrow.png" [static], [constexpr]
```

**9.1.2.26 sideBorder**

```
constexpr static string_view config::sideBorder = "../resources/Misc/Options_Side_Border.png" [static], [constexpr]
```

### 9.1.2.27 terminalVelocity

```
constexpr static auto config::terminalVelocity = 1000.0f [static], [constexpr]
```

### 9.1.2.28 widgetsFontName

```
constexpr static string_view config::widgetsFontName = "../resources/Fonts/CascadiaCode.ttf" [static], [constexpr]
```

### 9.1.2.29 windowHeight

```
constexpr static auto config::windowHeight = entityHeight * 18 [static], [constexpr]
```

### 9.1.2.30 windowWidth

```
constexpr static auto config::windowWidth = entityWidth * 20 [static], [constexpr]
```

## 9.2 config::bg Namespace Reference

### Variables

- constexpr static string\_view `options` = "../resources/Backgrounds/Options.png"
- constexpr static string\_view `keybinds` = "../resources/Backgrounds/Keybinds.png"
- constexpr static string\_view `loader` = "../resources/Backgrounds/Loader.jpg"
- constexpr static string\_view `menu` = "../resources/Backgrounds/Main\_Menu.jpg"
- constexpr static string\_view `paused` = "../resources/Backgrounds/Paused.png"
- constexpr static string\_view `restart` = "../resources/Backgrounds/Restart\_Game.png"

### 9.2.1 Variable Documentation

#### 9.2.1.1 keybinds

```
constexpr static string_view config::bg::keybinds = "../resources/Backgrounds/Keybinds.png" [static], [constexpr]
```

### 9.2.1.2 loader

```
constexpr static string_view config::bg::loader = "../resources/Backgrounds/Loader.jpg" [static],  
[constexpr]
```

### 9.2.1.3 menu

```
constexpr static string_view config::bg::menu = "../resources/Backgrounds/Main_Menu.jpg" [static],  
[constexpr]
```

### 9.2.1.4 options

```
constexpr static string_view config::bg::options = "../resources/Backgrounds/Options.png"  
[static], [constexpr]
```

### 9.2.1.5 paused

```
constexpr static string_view config::bg::paused = "../resources/Backgrounds/Paused.png" [static],  
[constexpr]
```

### 9.2.1.6 restart

```
constexpr static string_view config::bg::restart = "../resources/Backgrounds/Restart_Game.png"  
[static], [constexpr]
```

## 9.3 Gui Namespace Reference

Encapsulate common global [View Config](#) within an additional namespace.

### Classes

- struct [Config](#)  
*A shared gui config. Contains common settings.*

### 9.3.1 Detailed Description

Encapsulate common global [View Config](#) within an additional namespace.

## 9.4 Keybinds Namespace Reference

Namespace designated for [Keybinds](#) view.

### Classes

- struct [Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [KeybindsView](#).*

### Enumerations

- enum [Btn](#) {  
    [Btn::Jump](#),  
    [Btn::RunLeft](#),  
    [Btn::RunRight](#),  
    [Btn::GoBack](#),  
    [Btn::SIZE](#) }

*Keybinds buttons IDs.*

### Variables

- constexpr std::initializer\_list<[Keybinds::Btn](#)> [Buttons](#)

*An additional initializer list to help mapping Btn names to Strings.*

### 9.4.1 Detailed Description

Namespace designated for [Keybinds](#) view.

## 9.5 lives Namespace Reference

Encapsulates lives overlay variables.

### Variables

- constexpr auto [count](#) = 3
- constexpr auto [boxSide](#) = 50.0f
- const sf::Color [fillColor](#) = sf::Color{255, 255, 255, 140}
- const sf::Color [borderColor](#) = sf::Color::Red
- constexpr auto [borderThickness](#) = 2
- constexpr auto [borderOffsetX](#) = 15
- constexpr auto [borderOffsetY](#) = 15
- constexpr auto [heartOffsetX](#) = 25
- constexpr auto [heartOffsetY](#) = 25

### 9.5.1 Detailed Description

Encapsulates lives overlay variables.

### 9.5.2 Variable Documentation

#### 9.5.2.1 **borderColor**

```
const sf::Color lives::borderColor = sf::Color::Red  
  
semi-transparent white
```

#### 9.5.2.2 **borderOffsetX**

```
constexpr auto lives::borderOffsetX = 15 [constexpr]
```

#### 9.5.2.3 **borderOffsetY**

```
constexpr auto lives::borderOffsetY = 15 [constexpr]
```

#### 9.5.2.4 **borderThickness**

```
constexpr auto lives::borderThickness = 2 [constexpr]
```

#### 9.5.2.5 **boxSide**

```
constexpr auto lives::boxSide = 50.0f [constexpr]
```

#### 9.5.2.6 **count**

```
constexpr auto lives::count = 3 [constexpr]
```

### 9.5.2.7 fillColor

```
const sf::Color lives::fillColor = sf::Color{255, 255, 255, 140}
```

### 9.5.2.8 heartOffsetX

```
constexpr auto lives::heartOffsetX = 25 [constexpr]
```

### 9.5.2.9 heartOffsetY

```
constexpr auto lives::heartOffsetY = 25 [constexpr]
```

## 9.6 Loader Namespace Reference

Namespace designated for [MapLoader](#) view.

### Classes

- struct [Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MapLoaderView](#).*

### Enumerations

- enum [Btn](#) {  
    [Btn::loadMap](#),  
    [Btn::openEditor](#),  
    [Btn::goBack](#),  
    [Btn::SIZE](#) }

### Variables

- [constexpr std::initializer\\_list< Loader::Btn > Buttons](#)  
*An additional initializer list to help mapping Btn names to Strings.*

### 9.6.1 Detailed Description

Namespace designated for [MapLoader](#) view.

### 9.6.2 Enumeration Type Documentation

#### 9.6.2.1 Btn

```
enum Loader::Btn [strong]
```

Enumerator

|            |  |
|------------|--|
| loadMap    |  |
| openEditor |  |
| goBack     |  |
| SIZE       |  |

## 9.7 Menu Namespace Reference

Namespace designated for [Menu](#) view.

### Classes

- struct [Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MenuView](#).*

### Enumerations

- enum [Btn](#) {  
    [Btn::newGame](#),  
    [Btn::loadGame](#),  
    [Btn::options](#),  
    [Btn::about](#),  
    [Btn::exit](#),  
    [Btn::SIZE](#) }

### Variables

- [constexpr std::initializer\\_list< Menu::Btn > Buttons](#)  
*An additional initializer list to help mapping Btn names to Strings.*

#### 9.7.1 Detailed Description

Namespace designated for [Menu](#) view.

#### 9.7.2 Enumeration Type Documentation

##### 9.7.2.1 [Btn](#)

```
enum Menu::Btn [strong]
```

## Enumerator

|          |  |
|----------|--|
| newGame  |  |
| loadGame |  |
| options  |  |
| about    |  |
| exit     |  |
| SIZE     |  |

## 9.8 Obj Namespace Reference

Available game entity objects. Used only in [StateGame](#).

### Enumerations

- enum [Entity](#) {  
    [Entity::Empty](#),  
    [Entity::Player](#),  
    [Entity::Objective](#),  
    [Entity::HeartCollectible](#),  
    [Entity::BlockRed](#),  
    [Entity::BlockBlue](#),  
    [Entity::BlockBrown](#),  
    [Entity::BlockGray](#),  
    [Entity::BlockGreen](#),  
    [Entity::BlockPurple](#),  
    [Entity::BlockYellow](#),  
    [Entity::BlockCrate](#),  
    [Entity::Water1](#),  
    [Entity::Water2](#),  
    [Entity::Sign1](#),  
    [Entity::Sign2](#),  
    [Entity::Spike](#),  
    [Entity::SpikeLeft](#),  
    [Entity::SpikeRight](#),  
    [Entity::SpikeTop](#) }

### 9.8.1 Detailed Description

Available game entity objects. Used only in [StateGame](#).

### 9.8.2 Enumeration Type Documentation

#### 9.8.2.1 Entity

```
enum Obj::Entity [strong]
```

**Enumerator**

|                  |  |
|------------------|--|
| Empty            |  |
| Player           |  |
| Objective        |  |
| HeartCollectible |  |
| BlockRed         |  |
| BlockBlue        |  |
| BlockBrown       |  |
| BlockGray        |  |
| BlockGreen       |  |
| BlockPurple      |  |
| BlockYellow      |  |
| BlockCrate       |  |
| Water1           |  |
| Water2           |  |
| Sign1            |  |
| Sign2            |  |
| Spike            |  |
| SpikeLeft        |  |
| SpikeRight       |  |
| SpikeTop         |  |

## 9.9 Options Namespace Reference

Namespace designated for [Options](#) view.

### Classes

- struct [Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [OptionsView](#).*

### Enumerations

- enum [Btn](#) {
   
    [Btn::Keybinds](#),
   
    [Btn::GoBack](#),
   
    [Btn::SIZE](#) }

### Variables

- constexpr std::initializer\_list<[Options::Btn](#)> [Buttons](#)

*An additional initializer list to help mapping Btn names to Strings.*

### 9.9.1 Detailed Description

Namespace designated for [Options](#) view.

### 9.9.2 Enumeration Type Documentation

#### 9.9.2.1 Btn

```
enum Options::Btn [strong]
```

Enumerator

|          |  |
|----------|--|
| Keybinds |  |
| GoBack   |  |
| SIZE     |  |

## 9.10 Paused Namespace Reference

Namespace designated for [Paused](#) view.

### Classes

- struct [Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [PausedView](#).*

### Enumerations

- enum [Btn](#) {  
    [Btn::resume](#),  
    [Btn::options](#),  
    [Btn::menu](#),  
    [Btn::SIZE](#) }

### Variables

- [constexpr std::initializer\\_list< Paused::Btn > Buttons](#)  
*An additional initializer list to help mapping Btn names to Strings.*

### 9.10.1 Detailed Description

Namespace designated for [Paused](#) view.

## 9.10.2 Enumeration Type Documentation

### 9.10.2.1 Btn

```
enum Paused::Btn [strong]
```

#### Enumerator

|         |  |
|---------|--|
| resume  |  |
| options |  |
| menu    |  |
| SIZE    |  |

## 9.11 res Namespace Reference

Namespace wrapper over Resources enum IDs.

### Enumerations

- enum Texture {  
    Texture::BlockRed,  
    Texture::BlockBlue,  
    Texture::BlockBrown,  
    Texture::BlockGray,  
    Texture::BlockGreen,  
    Texture::BlockPurple,  
    Texture::BlockYellow,  
    Texture::BlockCrate,  
    Texture::Water1,  
    Texture::Water2,  
    Texture::Sign1,  
    Texture::Sign2,  
    Texture::Player,  
    Texture::PlayerLeft,  
    Texture::PlayerRight,  
    Texture::Objective,  
    Texture::Heart,  
    Texture::HeartEmpty,  
    Texture::BgGame,  
    Texture::BgAbout,  
    Texture::Spike,  
    Texture::OptionsLeftTopCorner,  
    Texture::OptionsLeftBotCorner,  
    Texture::OptionsRightBotCorner,  
    Texture::OptionsRightTopCorner,  
    Texture::GameLost,  
    Texture::GameWon }

*Available Texture IDs.*

- enum [Sound](#) {  
    [Sound::Collect](#),  
    [Sound::BtnHover](#),  
    [Sound::Death](#),  
    [Sound::WinGame](#) }  
    *Available Sound IDs.*
- enum [Music](#) { [Music::Background](#) }  
    *Available Music IDs.*

### 9.11.1 Detailed Description

Namespace wrapper over Resources enum IDs.

## 9.12 Restart Namespace Reference

Namespace designated for [Restart](#) view.

### Classes

- struct [Config](#)  
*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [RestartView](#).*

### Enumerations

- enum [Btn](#) {  
    [Btn::PlayAgain](#),  
    [Btn::Menu](#),  
    [Btn::SIZE](#) }

### Variables

- constexpr std::initializer\_list<[Restart::Btn](#) > [Buttons](#)  
*An additional initializer list to help mapping Btn names to Strings.*

### 9.12.1 Detailed Description

Namespace designated for [Restart](#) view.

### 9.12.2 Enumeration Type Documentation

#### 9.12.2.1 [Btn](#)

enum [Restart::Btn](#) [strong]

Enumerator

|           |  |
|-----------|--|
| PlayAgain |  |
| Menu      |  |
| SIZE      |  |

## 9.13 state Namespace Reference

The IDs of the stored states, so the user can access an arbitrary state.

### Variables

- int `gameID`
- int `menuID`
- int `loaderID`
- int `optionsID`
- int `keybindsID`
- int `pausedID`
- int `restartID`
- int `aboutID`

#### 9.13.1 Detailed Description

The IDs of the stored states, so the user can access an arbitrary state.

#### 9.13.2 Variable Documentation

##### 9.13.2.1 `aboutID`

```
int state::aboutID
```

##### 9.13.2.2 `gameID`

```
int state::gameID
```

**9.13.2.3 keybindsID**

```
int state::keybindsID
```

**9.13.2.4 loaderID**

```
int state::loaderID
```

**9.13.2.5 menuID**

```
int state::menuID
```

**9.13.2.6 optionsID**

```
int state::optionsID
```

**9.13.2.7 pausedID**

```
int state::pausedID
```

**9.13.2.8 restartID**

```
int state::restartID
```

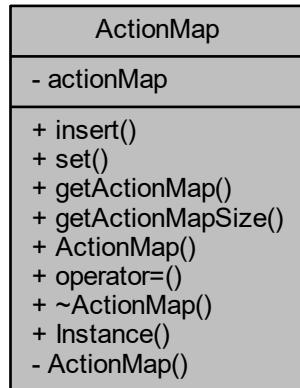
# Chapter 10

## Class Documentation

### 10.1 ActionMap Class Reference

```
#include <ActionMap.h>
```

Collaboration diagram for ActionMap:



### Public Member Functions

- void **insert** (const std::string &action, sf::Keyboard::Key key)  
*Insert {key, value} = {action, key} pair onto the map.*
- void **set** (const std::string &action, sf::Keyboard::Key key)  
*Update a given keybind.*
- auto & **getActionMap** ()  
*Retrieve the **actionMap** keybinds container.*
- auto **getActionMapSize** () const  
*Retrieve the size of the **actionMap** keybinds container.*
- **ActionMap** (**ActionMap** const &) noexcept=delete
- void **operator=** (**ActionMap** const &) noexcept=delete
- **~ActionMap** ()=default

## Static Public Member Functions

- static `ActionMap & Instance ()`  
*Retrieve the singleton instance of `ActionMap`.*

## Private Member Functions

- `ActionMap ()`

## Private Attributes

- `std::unordered_map< std::string, sf::Keyboard::Key > actionMap`  
*Map container for user keybinds. The default keybinds are set in a `Configuration.h` file.*

### 10.1.1 Constructor & Destructor Documentation

#### 10.1.1.1 ActionMap() [1/2]

```
ActionMap::ActionMap (
    ActionMap const & ) [delete], [noexcept]
```

The default move operations are deleted as it is a singleton.

#### Note

Copy operations are implicitly deleted.

*Scott Meyers: make the deleted operations public for better error messages.*

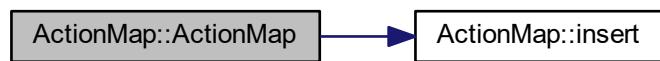
#### 10.1.1.2 ~ActionMap()

```
ActionMap::~ActionMap ( ) [default]
```

#### 10.1.1.3 ActionMap() [2/2]

```
ActionMap::ActionMap ( ) [private]
```

Prohibit Here is the call graph for this function:



## 10.1.2 Member Function Documentation

### 10.1.2.1 `getActionMap()`

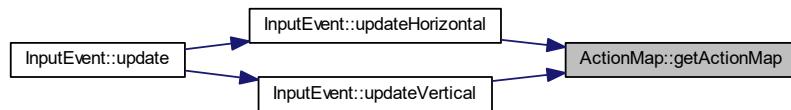
```
auto& ActionMap::getActionMap ( ) [inline]
```

Retrieve the `actionMap` keybinds container.

#### Returns

`Keybinds` map container.

Here is the caller graph for this function:



### 10.1.2.2 `getActionMapSize()`

```
auto ActionMap::getActionMapSize ( ) const [inline]
```

Retrieve the size of the `actionMap` keybinds container.

#### Returns

The size of `actionMap` keybinds container.

### 10.1.2.3 `insert()`

```
void ActionMap::insert (
    const std::string & action,
    sf::Keyboard::Key key )
```

Insert `{key, value} = {action, key}` pair onto the map.

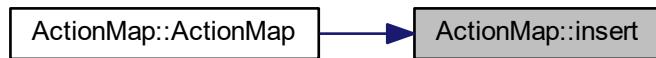
**Parameters**

|               |                        |
|---------------|------------------------|
| <i>action</i> | This is a map's key.   |
| <i>key</i>    | This is a map's value. |

**Warning**

Distinguish between 'key' as a keybind and a 'key' as a general map's data structure key vocabulary.

Here, an action is a *general* map key, while a key is a *general* map's value. Here is the caller graph for this function:

**10.1.2.4 Instance()**

```
ActionMap & ActionMap::Instance () [static]
```

Retrieve the singleton instance of [ActionMap](#).

**Note**

My singleton implementation is **lazy-evaluated, correctly-destroyed, and thread-safe**.

< Guaranteed to be destroyed.

< Instantiated on the first use.

**10.1.2.5 operator=()**

```
void ActionMap::operator= (
    ActionMap const & ) [delete], [noexcept]
```

The default move operations are deleted as it is a singleton.

**Note**

Copy operations are implicitly deleted.

*Scott Meyers: make the deleted operations public for better error messages.*

**10.1.2.6 set()**

```
void ActionMap::set (
    const std::string & action,
    sf::Keyboard::Key key )
```

Update a given keybind.

### Parameters

|               |                               |
|---------------|-------------------------------|
| <i>action</i> | An action to be bound onto... |
| <i>key</i>    | ...the given keybind.         |

Here is the caller graph for this function:



## 10.1.3 Member Data Documentation

### 10.1.3.1 actionMap

`std::unordered_map<std::string, sf::Keyboard::Key> ActionMap::actionMap [private]`

Map container for user keybinds. The default keybinds are set in a [Configuration.h](#) file.

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

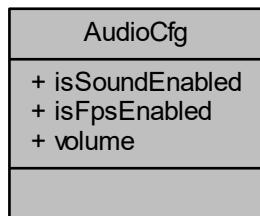
- [ActionMap.h](#)
- [ActionMap.cpp](#)

## 10.2 AudioCfg Class Reference

Responsible for storing audio config enable.

`#include <Settings.h>`

Collaboration diagram for AudioCfg:



## Public Attributes

- bool `isSoundEnabled` = true
- bool `isFpsEnabled` = true
- float `volume` = `config::defaultSoundVolume`

### 10.2.1 Detailed Description

Responsible for storing audio config enable.

#### Note

`volume` default init value is customizable from config namespace.

### 10.2.2 Member Data Documentation

#### 10.2.2.1 `isFpsEnabled`

```
bool AudioCfg::isFpsEnabled = true
```

#### 10.2.2.2 `isSoundEnabled`

```
bool AudioCfg::isSoundEnabled = true
```

#### 10.2.2.3 `volume`

```
float AudioCfg::volume = config::defaultSoundVolume
```

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

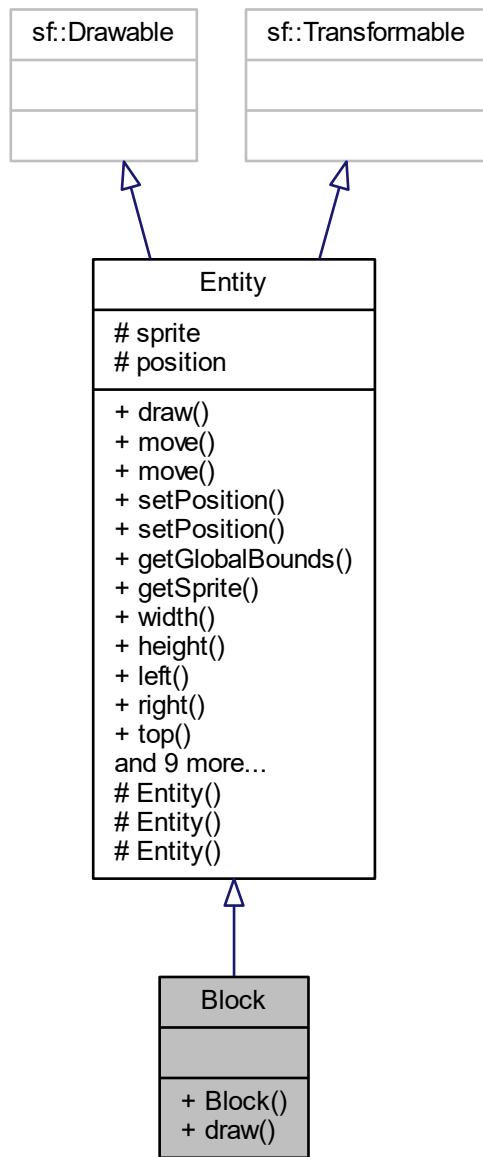
- `Settings.h`

## 10.3 Block Class Reference

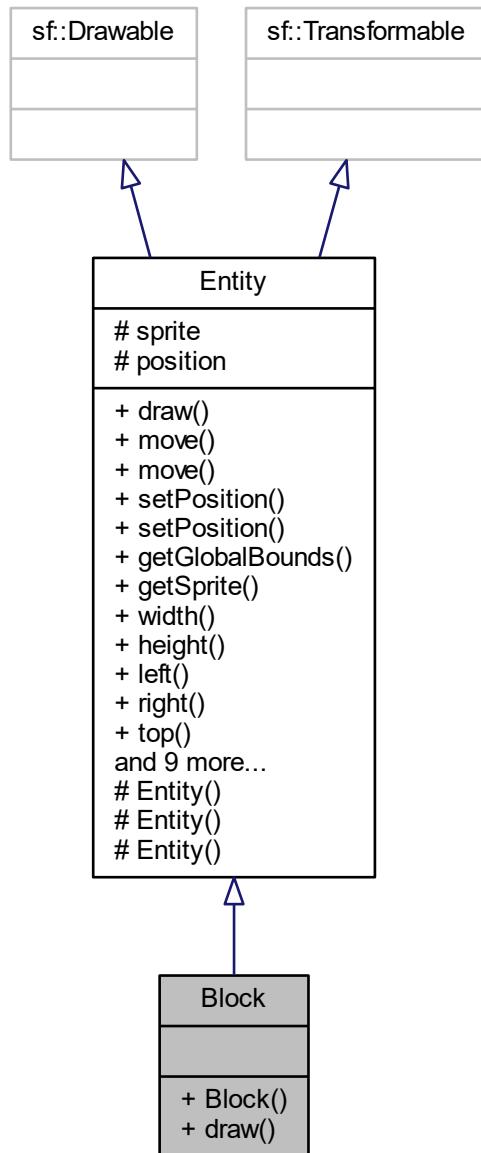
Used to represent collidable game blocks.

```
#include <Block.h>
```

Inheritance diagram for Block:



Collaboration diagram for Block:



## Public Member Functions

- `Block (sf::Vector2f position)`  
*Constructs a collidable `Block`.*
- `void draw (sf::RenderTarget &target, sf::RenderStates states) const override`  
*Draw method used to render `sprite` onto the window.*

## Additional Inherited Members

### 10.3.1 Detailed Description

Used to represent collidable game blocks.

Available collidable [Block](#) types:

- [BlockRed](#),
- [BlockBlue](#),
- [BlockBrown](#),
- [BlockGray](#),
- [BlockGreen](#),
- [BlockPurple](#),
- [BlockYellow](#),
- [BlockCrate](#),
- [Water1](#),
- [Water2](#),
- [Sign1](#),
- [Sign2](#),

#### See also

[Obj::Entity](#)

#### Note

One specific texture e.g. `BlockYellow` is loaded only once to the memory.

A concrete [Entity](#) class overriding the draw method.

### 10.3.2 Constructor & Destructor Documentation

#### 10.3.2.1 [Block\(\)](#)

```
Block::Block ( sf::Vector2f position ) [explicit]
```

Constructs a collidable [Block](#).

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| <i>position</i> | initial position, defaulted to zero. |
|-----------------|--------------------------------------|

**10.3.3 Member Function Documentation****10.3.3.1 draw()**

```
void Block::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render [sprite](#) onto the window.

**Parameters**

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides [Entity](#) draw method.

**Note**

Passing states is optional.

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

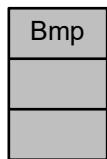
- [Block.h](#)
- [Block.cpp](#)

**10.4 Bmp Struct Reference**

Empty struct used to determine file type in [MapLoader](#) variant.

```
#include <MapLoader.h>
```

Collaboration diagram for Bmp:



### 10.4.1 Detailed Description

Empty struct used to determine file type in [MapLoader](#) variant.

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

- [MapLoader.h](#)

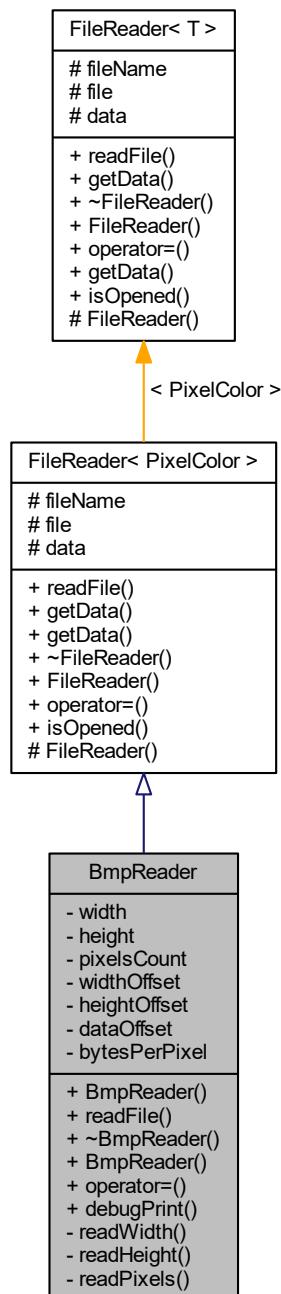
## 10.5 BmpReader Class Reference

Reads the .bmp file into the vector of [PixelColor](#).

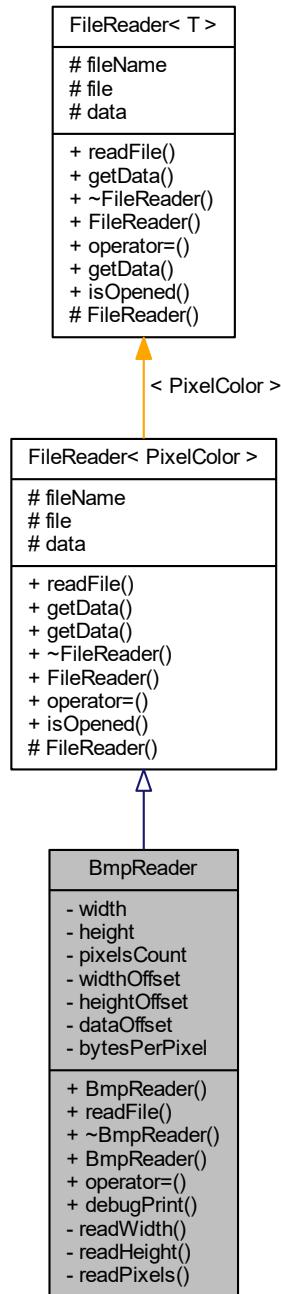
---

```
#include <BmpReader.h>
```

Inheritance diagram for BmpReader:



Collaboration diagram for BmpReader:



## Public Member Functions

- `BmpReader` (const std::string &name, const std::ios\_base::openmode &mode=std::ifstream::ate|std::ifstream::binary)
 

*Create a `BmpReader`.*
- void `readFile` () override
- `~BmpReader` () override=default

- **BmpReader (BmpReader &&)** noexcept=default  
*Enable move operations.*
- **BmpReader & operator= (BmpReader &&)** noexcept=default  
*Enable move operations.*
- **void debugPrint () const**  
*Print formatted file content debug info.*

## Private Member Functions

- **void readWidth ()**  
*Read the .bmp width into `width`. Peeks specific `widthOffset` byte.*
- **void readHeight ()**  
*Read the .bmp height into `height`. Peeks specific `heightOffset` byte.*
- **void readPixels ()**  
*Reads the file `PixelColor` data once it was opened and its `width` and `height` were established.*

## Private Attributes

- **int32\_t width {}**
- **int32\_t height {}**
- **int pixelsCount {}**

## Static Private Attributes

- **static constexpr auto widthOffset {18}**
- **static constexpr auto heightOffset {22}**
- **static constexpr auto dataOffset {54}**
- **static constexpr auto bytesPerPixel {3}**

## Additional Inherited Members

### 10.5.1 Detailed Description

Reads the .bmp file into the vector of `PixelColor`.

#### Warning

Only .bmp files with width that is multiplier of **4** (e.g. 44 but not 45) are correctly encodeable into the World.

Valid .bmp dimensions [width x height]:

- 40 x 23
- 200 x 10
- 404 x 303

Invalid .bmp dimensions [width x height]:

- 101 x 40
- 43 x 44
- 501 x 10

#### Note

## 10.5.2 Constructor & Destructor Documentation

### 10.5.2.1 BmpReader() [1/2]

```
BmpReader::BmpReader (
    const std::string & name,
    const std::ios_base::openmode & mode = std::ifstream::ate | std::ifstream::binary
) [explicit]
```

Create a [BmpReader](#).

#### Parameters

|             |   |
|-------------|---|
| <i>name</i> | Name of the file to be read.              |
| <i>mode</i> | Open mode introduced as a generalization. |

Reads the .bmp file data.

### 10.5.2.2 ~BmpReader()

```
BmpReader::~BmpReader ( ) [override], [default]
```

Default Virtual constructor.

### 10.5.2.3 BmpReader() [2/2]

```
BmpReader::BmpReader (
    BmpReader && ) [default], [noexcept]
```

Enable move operations.

## 10.5.3 Member Function Documentation

### 10.5.3.1 debugPrint()

```
void BmpReader::debugPrint ( ) const
```

Print formatted file content debug info.

### 10.5.3.2 operator=( )

```
BmpReader& BmpReader::operator= (
    BmpReader && ) [default], [noexcept]
```

Enable move operations.

### 10.5.3.3 readFile()

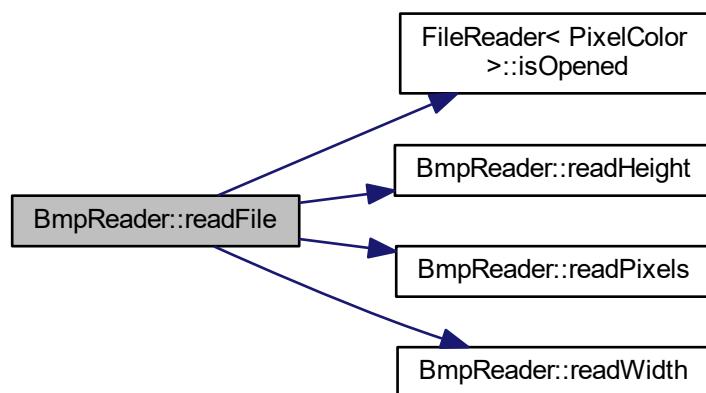
```
void BmpReader::readFile ( ) [override], [virtual]
```

\brief Read the file data into the `PixelColor` vector.

Implements the pure virtual method of `FileReader<T>`.

Implements `FileReader< PixelColor >`.

Here is the call graph for this function:



#### 10.5.3.4 `readHeight()`

```
void BmpReader::readHeight ( ) [private]
```

Read the .bmp height into `height`. Peeks specific `heightOffset` byte.

Here is the caller graph for this function:



#### 10.5.3.5 `readPixels()`

```
void BmpReader::readPixels ( ) [private]
```

Reads the file `PixelColor` data once it was opened and its `width` and `height` were established.

Here is the caller graph for this function:



#### 10.5.3.6 `readWidth()`

```
void BmpReader::readWidth ( ) [private]
```

Read the .bmp width into `width`. Peeks specific `widthOffset` byte.

Here is the caller graph for this function:



## 10.5.4 Member Data Documentation

### 10.5.4.1 bytesPerPixel

```
constexpr auto BmpReader::bytesPerPixel {3} [static], [constexpr], [private]
```

### 10.5.4.2 dataOffset

```
constexpr auto BmpReader::dataOffset {54} [static], [constexpr], [private]
```

### 10.5.4.3 height

```
int32_t BmpReader::height {} [private]
```

### 10.5.4.4 heightOffset

```
constexpr auto BmpReader::heightOffset {22} [static], [constexpr], [private]
```

### 10.5.4.5 pixelsCount

```
int BmpReader::pixelsCount {} [private]
```

### 10.5.4.6 width

```
int32_t BmpReader::width {} [private]
```

### 10.5.4.7 widthOffset

```
constexpr auto BmpReader::widthOffset {18} [static], [constexpr], [private]
```

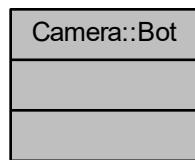
The documentation for this class was generated from the following files:

- [BmpReader.h](#)
- [BmpReader.cpp](#)

## 10.6 Camera::Bot Struct Reference

Empty struct denoting previously solved on other axis was [Bot](#) collision.

Collaboration diagram for Camera::Bot:



### 10.6.1 Detailed Description

Empty struct denoting previously solved on other axis was [Bot](#) collision.

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

- [Camera.h](#)

## 10.7 Camera Class Reference

Responsible for viewing only a sensible, displayable part of the World to the screen. Handles all [Camera](#) collisions.

```
#include <Camera.h>
```

Collaboration diagram for Camera:

| Camera   |
|--|
| <ul style="list-style-type: none"> <li>- previouslySolvedOtherAxis</li> <li>- solvedNow</li> <li>- cameraView</li> <li>- controller</li> <li>- cameraWidth</li> <li>- cameraHeight</li> <li>- halvedCameraWidth</li> <li>- halvedCameraHeight</li> <li>- mapWidth</li> <li>- mapHeight</li> <li>- leftBorderBoundary</li> <li>- rightBorderBoundary</li> <li>- topBorderBoundary</li> <li>- bottomBorderBoundary</li> <li>- halvedSpriteWidth</li> <li>- halvedSpriteHeight</li> </ul> |
| <ul style="list-style-type: none"> <li>+ Camera()</li> <li>+ updateX()</li> <li>+ updateY()</li> <li>+ storeJustSolvedForOtherAxis()</li> <li>+ detectCollisionX()</li> <li>+ detectCollisionY()</li> <li>+ getCameraView()</li> <li>+ setController()</li> <li>- setCameraConstants()</li> <li>- setControllerConstants()</li> </ul>  |

## Classes

- struct [Bot](#)

*Empty struct denoting previously solved on other axis was [Bot](#) collision.*
- struct [Left](#)

*Empty struct denoting previously solved on other axis was [Left](#) collision.*
- struct [None](#)

*Empty struct denoting there was previously [None](#) collision.*
- struct [Right](#)

*Empty struct denoting previously solved on other axis was [Right](#) collision.*
- struct [Top](#)

*Empty struct denoting previously solved on other axis was [Top](#) collision.*

## Public Member Functions

- [Camera](#) (const sf::View &view) noexcept

- Constructs a `Camera` from a `Window`'s view.
- void `updateX () noexcept`  
*Updates the camera on **horizontal X** axis. Moves the camera and resolves the collision if needed.*
- void `updateY () noexcept`  
*Updates the camera on **vertical Y** axis. Moves the camera and resolves the collision if needed.*
- void `storeJustSolvedForOtherAxis (CurrentCollision wasCollision) noexcept`  
*Update the `previouslySolvedOtherAxis` variant state with whatever was just solved collision.*
- float `detectCollisionX () noexcept`  
*Detects if there is a collision on X axis as of **right now**.*
- float `detectCollisionY () noexcept`  
*Detects if there is a collision on Y axis as of **right now**.*
- sf::View & `getCameraView () noexcept`  
*Retrieve the `cameraView`.*
- void `setController (Entity &controllingEntity) noexcept`  
*Set the `Entity` that is controlling the `Camera`.*

## Private Types

- enum `CurrentCollision {`
- `CurrentCollision::None,`
- `CurrentCollision::Left,`
- `CurrentCollision::Right,`
- `CurrentCollision::Top,`
- `CurrentCollision::Bot }`

*Enum holding state of the **currently solving** Collision found on the **same** axis.*

## Private Member Functions

- void `setCameraConstants () noexcept`  
*Store the `Camera` configuration constants.*
- void `setControllerConstants () noexcept`  
*Store player's `halvedSpriteWidth` and `halvedSpriteHeight`.*

## Private Attributes

- std::variant< `None, Left, Right, Top, Bot` > `previouslySolvedOtherAxis {None{}}`  
*Holds solved collision from a previous, **different** axis, so proper response to corners-collision can be performed.*
- `CurrentCollision solvedNow {CurrentCollision::None}`  
*Whatever was just solved on the same axis.*
- sf::View `cameraView`  
*Viewable screen region drawn onto the window.*
- sf::Sprite \* `controller {nullptr}`  
*Controls the camera's position. Does **not** allocate dynamic memory.*
- float `cameraWidth {}`  
*Camera width is same as `Window` width.*
- float `cameraHeight {}`  
*Camera height is same as `Window` height.*
- float `halvedCameraWidth {}`
- float `halvedCameraHeight {}`

- float `mapWidth` {}  
*For calculating `rightBorderBoundary`.*
- float `mapHeight` {}  
*For calculating `bottomBorderBoundary`.*
- float `leftBorderBoundary` {}  
*Precomputed solved X coord for `Left` collision.*
- float `rightBorderBoundary` {}  
*Precomputed solved X coord for `Right` collision.*
- float `topBorderBoundary` {}  
*Precomputed solved Y coord for `Top` collision.*
- float `bottomBorderBoundary` {}  
*Precomputed solved Y coord for `Bot` collision.*
- float `halvedSpriteWidth` {}
- float `halvedSpriteHeight` {}

### 10.7.1 Detailed Description

Responsible for viewing only a sensible, displayable part of the World to the screen. Handles all `Camera` collisions.

`Camera` is bound to the `Player` (sprite) position, `Player` is controller, but it can be rebound to any `Entity` (sprite) controller.

`Camera`'s view width and height is equal to the `Window` width and height.

#### Note

`Camera` can collide with the World borders **in a few different ways**.

#### Possible collision types seen from a *user's perspective*:

- ***none***
- ***left***
- ***right***
- ***bottom***
- ***top***
- ***left-top corner***
- ***left-bottom corner***
- ***right-top corner***
- ***right-bot corner***

#### Real collision states at the implementation level:

- ***none***
- ***left***
- ***right***
- ***bottom***
- ***top***

*Warning*

*Note*

*My Camera algorithm updates its position on one axis at a time.*

*Algorithm:*

- 1. Move Player (and Camera) on X axis.
- 2. Check if camera is colliding on X axis? Store Left, Right or None accordingly.
- 3. Calculate correct X position.
- 4. Resolve that X collision based on whatever was previous Y collision state and calculated X position.
- 5. Store solvedNow inside previouslySolvedOtherAxis for \*\*updateY().
- 6. Move Player (and Camera) on Y axis.
- 7. Check if camera is colliding on Y axis?
- 8. Calculate correct Y position.
- 9. Resolve that Y collision, based on whatever was previous X collision state and calculated Y position.
- 10. Store solvedNow inside previouslySolvedOtherAxis for updateX().

*Note*

*Corollary: camera is not simultaneously colliding e.g. left-top. At the implementation level, it is only in one colliding state at a time.*

*Warning*

*Structs are used for std::variant (storing previously solved collision on the other axis), while Enum is used as a helper for storing current axis collision.*

## 10.7.2 Member Enumeration Documentation

### 10.7.2.1 CurrentCollision

```
enum Camera::CurrentCollision [strong], [private]
```

Enum holding state of the **currently solving** Collision found on the **same** axis.

## Enumerator

|       |  |
|-------|--|
| None  |  |
| Left  |  |
| Right |  |
| Top   |  |
| Bot   |  |

**10.7.3 Constructor & Destructor Documentation****10.7.3.1 Camera()**

```
Camera::Camera (
    const sf::View & view ) [explicit], [noexcept]
```

Constructs a [Camera](#) from a [Window](#)'s view.

## Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>view</i> | <a href="#">Window</a> 's view. |
|-------------|---------------------------------|

Also sets relevant camera constants through `setCamersConstants()`.

## Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

**10.7.4 Member Function Documentation****10.7.4.1 detectCollisionX()**

```
float Camera::detectCollisionX ( ) [noexcept]
```

Detects if there is a collision on X axis as of **right now**.

## Note

[Right](#) before calling this method, the [Player](#) controller (and hence: [Camera](#) view too!) was just moved on X axis.

If the collision was [None](#) (there is no collision), then X position was already correct. If the collision was [Left](#), then correct X position is [leftBorderBoundary](#). If the collision was [Right](#), then correct X position is [rightBorderBoundary](#).

**Note**

Store **Left**, **Right** or **None** for the future **updateY()**

**Returns**

Correct (solved) collision-free X position.

Check X position after sprite was moved on X axis.

Calculate correct resolved non-colliding X position.

< There was a **Right** collision, so store that fact for possible corner-collision response.

< There was a **Left** collision, so store that fact for possible corner-collision response.

< There was **None** collision. For sure not a corner collision.

**Examples**

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

Here is the caller graph for this function:

**10.7.4.2 detectCollisionY()**

```
float Camera::detectCollisionY ( ) [noexcept]
```

Detects if there is a collision on Y axis as of **right now**.

**Note**

**Right** before calling this method, the **Player** controller (and hence: **Camera** view too!) was just moved on Y axis.

If the collision was **None** (there is no collision), then Y position was already correct. If the collision was **Top**, then correct Y position is **topBorderBoundary**. If the collision was **Bot**, then correct Y position is **bottomBorderBoundary**.

**Note**

Store **Top**, **Bot** or **None** for the future **updateX()** method (next game frame).

**Returns**

Correct (solved) collision-free Y position.

**Examples**

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](#).

Here is the caller graph for this function:



#### 10.7.4.3 getCameraView()

```
sf::View & Camera::getCameraView ( ) [noexcept]
```

Retrieve the [cameraView](#).

**Returns**

cameraView Displayed camera view.

**Examples**

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](#).

Here is the caller graph for this function:



#### 10.7.4.4 setCameraConstants()

```
void Camera::setCameraConstants ( ) [private], [noexcept]
```

Store the [Camera](#) configuration constants.

**Examples**

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](#).

#### 10.7.4.5 setController()

```
void Camera::setController (
    Entity & controllingEntity ) [noexcept]
```

Set the [Entity](#) that is controlling the [Camera](#).

##### Parameters

|                                |  |
|--------------------------------|--|
| <code>controllingEntity</code> | the entity that will take control over <a href="#">cameraView</a> position/movement. |
|--------------------------------|--|

On an implementation level, this simply binds to a sf::Sprite.

##### Note

In this game [Player](#) is controlling [Camera](#), but any [Entity](#) could do.

##### Examples

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](#).

Here is the caller graph for this function:



#### 10.7.4.6 setControllerConstants()

```
void Camera::setControllerConstants () [private], [noexcept]
```

Store player's [halvedSpriteWidth](#) and [halvedSpriteHeight](#).

Stored so the camera is kept exactly in the center of the screen.

##### Examples

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](#).

#### 10.7.4.7 `storeJustSolvedForOtherAxis()`

```
void Camera::storeJustSolvedForOtherAxis (
    CurrentCollision wasCollision ) [noexcept]
```

Update the `previouslySolvedOtherAxis` variant state with whatever was just solved collision.

##### Note

This method is called on exit from `updateX()` or `updateY()`, so it can store the just solved collision.

If exiting `updateX()` method, `previouslySolvedOtherAxis` will be set to one of:

- `Left`
- `Right`
- `None` If exiting `updateY()` method, `previouslySolvedOtherAxis` will be set to one of:
  - `Top`
  - `Bot`
  - `None`

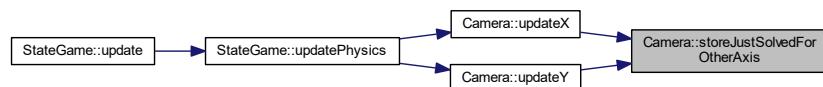
##### Note

Note that a `cameraView` is always moving **on one axis at a time**.

##### Examples

`C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp.`

Here is the caller graph for this function:



#### 10.7.4.8 updateX()

```
void Camera::updateX( ) [noexcept]
```

Updates the camera on **horizontal X** axis. Moves the camera and resolves the collision if needed.

UpdateX knows if there was a **Top**, **Bot** or **None** collision in a previous frame **updateY()** call, because it was stored in **previouslySolvedOtherAxis**.

Therefore: **updateX()** can respond properly to corner collisions.

#### Note

**Camera** is colliding only on one axis at a time, even if it is "corner collision".

Detect **Left**, **Right** or **None** collision. Store it in **solvedNow** and return correct #solvedX position.

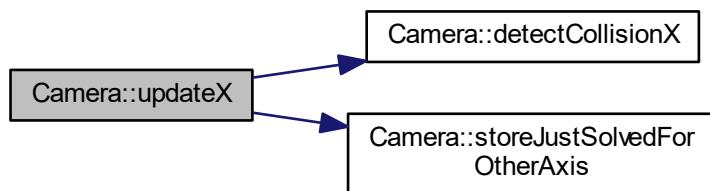
Based on **previouslySolvedOtherAxis** computed in **updateY()**, which is either **Bot**, **Top** or **None** - we can respond correctly to the corners and set correct **Camera** position.

Upon exit, update **previouslySolvedOtherAxis** with what we just **solvedNow**: either **Top**, **Bot** or **None**.

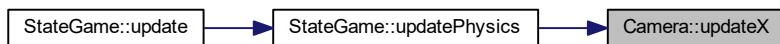
#### Examples

C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.7.4.9 updateY()

```
void Camera::updateY( ) [noexcept]
```

Updates the camera on **vertical Y** axis. Moves the camera and resolves the collision if needed.

UpdateY knows if there was a **Left**, **Right** or **None** collision in a previous frame **updateX()** call, because it was stored in **previouslySolvedOtherAxis**.

Therefore: **updateY()** can respond properly to corner collisions.

#### Note

**Camera** is colliding with only one axis at a time, even if it is "corner collision".

Detect **Top**, **Bot** or **None** collision. Store it in **solvedNow** and return correct #solvedY position.

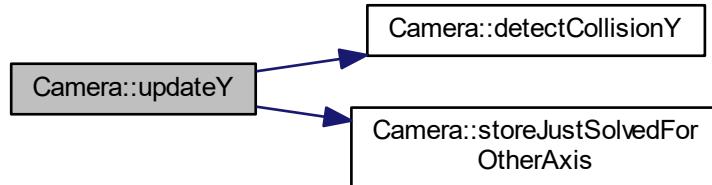
Based on **previouslySolvedOtherAxis** computed in **updateX()**, which is either **Left**, **Right** or **None** - we can respond correctly to the corners and set correct **Camera** position.

Upon exit, update **previouslySolvedOtherAxis** with what we just **solvedNow**: either **Top**, **Bot** or **None**.

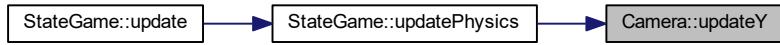
#### Examples

[C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp](https://github.com/wenox/platformer-2d/blob/main/src/Core/Camera.cpp).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.7.5 Member Data Documentation

### 10.7.5.1 bottomBorderBoundary

```
float Camera::bottomBorderBoundary {} [private]
```

Precomputed solved Y coord for [Bot](#) collision.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.2 cameraHeight

```
float Camera::cameraHeight {} [private]
```

[Camera](#) height is same as [Window](#) height.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.3 cameraView

```
sf::View Camera::cameraView [private]
```

Viewable screen region drawn onto the window.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.4 cameraWidth

```
float Camera::cameraWidth {} [private]
```

[Camera](#) width is same as [Window](#) width.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.5 controller

```
sf::Sprite* Camera::controller {nullptr} [private]
```

Controls the camera's position. Does **not** allocate dynamic memory.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.6 halvedCameraHeight

```
float Camera::halvedCameraHeight {} [private]
```

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.7 halvedCameraWidth

```
float Camera::halvedCameraWidth {} [private]
```

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.8 halvedSpriteHeight

```
float Camera::halvedSpriteHeight {} [private]
```

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.9 halvedSpriteWidth

```
float Camera::halvedSpriteWidth {} [private]
```

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.10 leftBorderBoundary

```
float Camera::leftBorderBoundary {} [private]
```

Precomputed solved X coord for [Left](#) collision.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.11 mapHeight

```
float Camera::mapHeight {} [private]
```

For calculating [bottomBorderBoundary](#).

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.12 mapWidth

```
float Camera::mapWidth {} [private]
```

For calculating [rightBorderBoundary](#).

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

### 10.7.5.13 previouslySolvedOtherAxis

```
std::variant<None, Left, Right, Top, Bot> Camera::previouslySolvedOtherAxis {None{}} [private]
```

Holds solved collision from a previous, **different** axis, so proper response to corners-collision can be performed.

A camera can be only in one collision state at a time.

#### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

#### 10.7.5.14 rightBorderBoundary

```
float Camera::rightBorderBoundary {} [private]
```

Precomputed solved X coord for [Right](#) collision.

##### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

#### 10.7.5.15 solvedNow

```
CurrentCollision Camera::solvedNow {CurrentCollision::None} [private]
```

Whatever was just solved on the same axis.

##### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

#### 10.7.5.16 topBorderBoundary

```
float Camera::topBorderBoundary {} [private]
```

Precomputed solved Y coord for [Top](#) collision.

##### Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

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

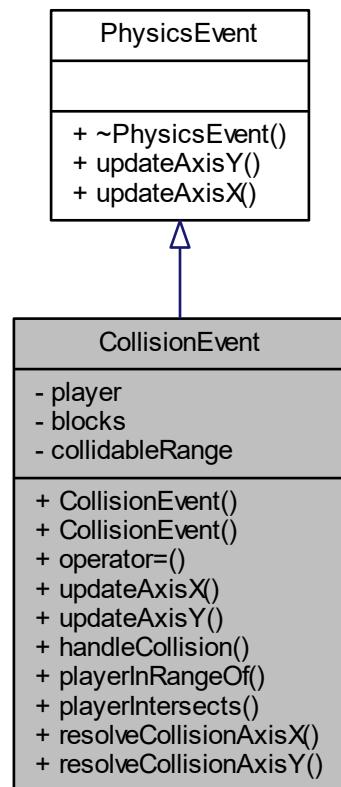
- [Camera.h](#)
- [Camera.cpp](#)

## 10.8 CollisionEvent Class Reference

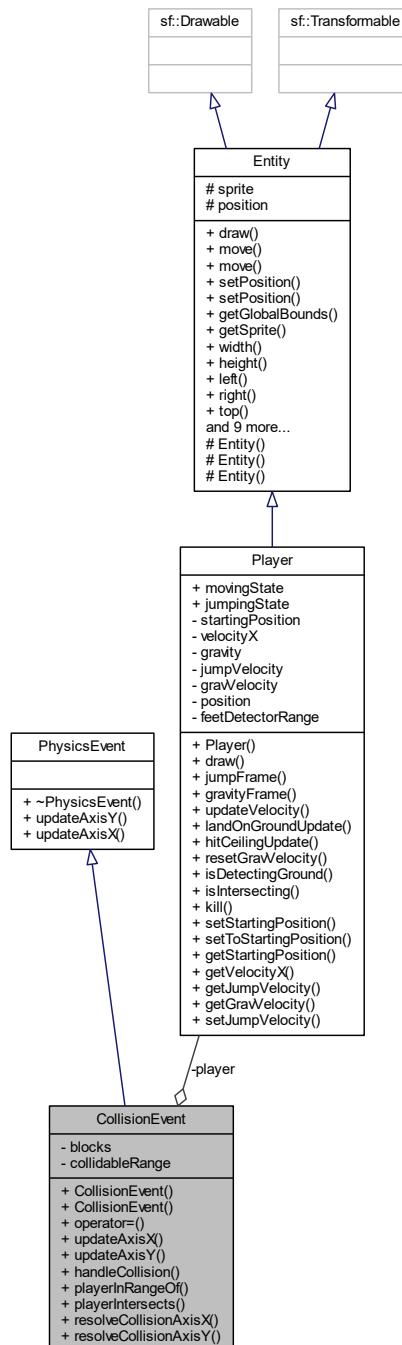
[CollisionEvent](#) class responsible for solves player<->collidable collisions in **axis-independent way**.

```
#include <CollisionEvent.h>
```

Inheritance diagram for CollisionEvent:



Collaboration diagram for CollisionEvent:



## Public Member Functions

- `CollisionEvent (Player &player, std::vector< std::unique_ptr< Entity >> &blocks)`  
*Create a `CollisionEvent` instance.*
- `CollisionEvent (const CollisionEvent &)=delete`  
*Delete copy operations.*
- `CollisionEvent & operator= (const CollisionEvent &)=delete`

- Delete copy operations.
- void `updateAxisX` (float) override
  - Handle horizontal collisions.*
- void `updateAxisY` (float) override
  - Handle vertical collisions.*
- void `handleCollision` (const Entity &block, const std::function< void(const Entity &) > &resolveCollision)
  - Axis-independent collision handler, invokes collision resolver.*
- bool `playerInRangeOf` (const Entity &entity) const
  - Checks if Player is in range of a given entity.*
- bool `playerIntersects` (const Entity &entity) const
  - Checks if Player intersects a given entity.*
- void `resolveCollisionAxisX` (const Entity &block)
  - Solve the collision on X axis. Passed as an argument to std::function.*
- void `resolveCollisionAxisY` (const Entity &block)
  - Solve the collision on X axis. Passed as an argument to std::function.*

## Private Attributes

- Player & `player`
  - Controller.*
- std::vector< std::unique\_ptr< Entity > > & `blocks`
  - Collidable blocks.*
- float `collidableRange`
  - The entities do not have to be squares; in such a case, collidableRange is the longer dimension.*

### 10.8.1 Detailed Description

`CollisionEvent` class responsible for solving player<->collidable collisions in **axis-independent way**.

#### Note

My general collision solving idea:

- 1. Move player on X axis
- 2. Check for collision
- 3. Resolve that collision (move back the player on Y axis as little as possible, but so that he no longer collides)
- 4. Move player on Y axis
- 5. Check for collision
- 6. Resolve that collision (move back the player on Y axis as little as possible, but so that he no longer collides)

This collision algorithm was tested heavily and should never disappoint in such game, unless the player is passing-through entities due to very slow FPS, absurd high velocity, or both. This is never a case in my game as it runs in 300-3000 FPS (depending on the map size) on my *slow* laptop (assuming FPS limiter is unlocked).

Problem would rise at around or below 10 FPS, or *teleporting* velocity.

Alternatives are e.g. *Sweep and prune* algorithm, or raycasting.

#### Note

Implements pure virtual methods from base `PhysicsEvent` class.

## 10.8.2 Constructor & Destructor Documentation

### 10.8.2.1 CollisionEvent() [1/2]

```
CollisionEvent::CollisionEvent (
    Player & player,
    std::vector< std::unique_ptr< Entity >> & blocks )
```

Create a [CollisionEvent](#) instance.

#### Parameters

|               |  |
|---------------|--|
| <i>player</i> | - controller   |
| <i>blocks</i> | - entities for the controller to be matched collisions against |

Collidable entity does not have to be a square. In such a case, collidable range is the longer side

### 10.8.2.2 CollisionEvent() [2/2]

```
CollisionEvent::CollisionEvent (
    const CollisionEvent & ) [delete]
```

Delete copy operations.

## 10.8.3 Member Function Documentation

### 10.8.3.1 handleCollision()

```
void CollisionEvent::handleCollision (
    const Entity & block,
    const std::function< void(const Entity &) > & resolveCollision )
```

Axis-independent collision handler, invokes collision resolver.

#### Parameters

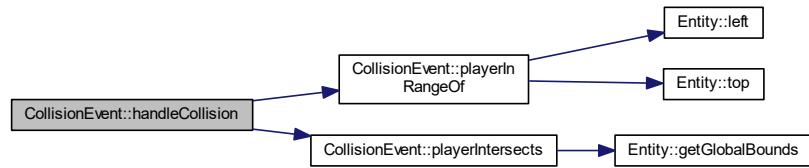
|                         |   |
|-------------------------|---|
| <i>block</i>            | Collidable entity player is matched against.                          |
| <i>resolveCollision</i> | will be passed either resolveCollisionAxisX, or resolveCollisionAxisY |

#### Note

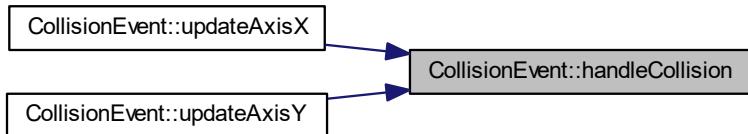
Collision is tried to be resolved only if:

- player is in range of block (checked first)
- player intersects a block (checked if the above was true)

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.8.3.2 operator=( )

```
CollisionEvent& CollisionEvent::operator= (
    const CollisionEvent & ) [delete]
```

Delete copy operations.

### 10.8.3.3 playerInRangeOf( )

```
bool CollisionEvent::playerInRangeOf (
    const Entity & entity ) const
```

Checks if [Player](#) is in range of a given entity.

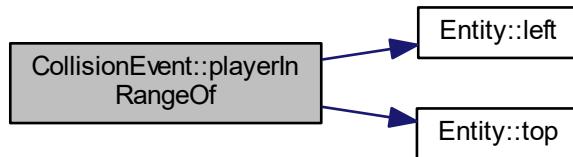
#### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>entity</code> | entity to be checked against. |
|---------------------|-------------------------------|

**Returns**

Was the [Player](#) in range of that entity?

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.8.3.4 `playerIntersects()`

```
bool CollisionEvent::playerIntersects ( const Entity & entity ) const
```

Checks if [Player](#) intersects a given entity.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>entity</code> | entity to be checked against. |
|---------------------|-------------------------------|

**Returns**

Was the player intersecting that entity?

Here is the call graph for this function:



Here is the caller graph for this function:

**10.8.3.5 resolveCollisionAxisX()**

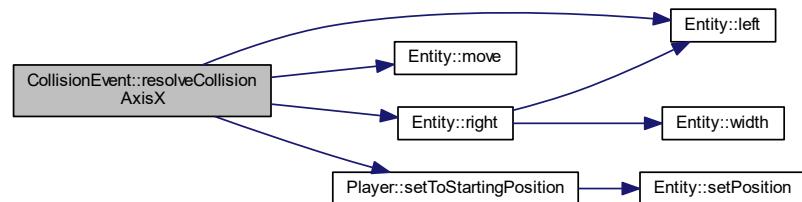
```
void CollisionEvent::resolveCollisionAxisX (
    const Entity & block )
```

Solve the collision on X axis. Passed as an argument to std::function.

**Parameters**

|              |  |
|--------------|--|
| <i>block</i> | A block collision is resolved against. Needed to compute resolved <a href="#">Player</a> position. |
|--------------|--|

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.8.3.6 resolveCollisionAxisY()

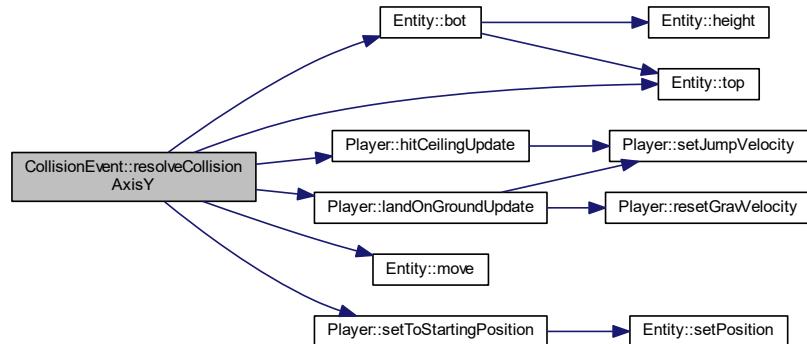
```
void CollisionEvent::resolveCollisionAxisY (
    const Entity & block )
```

Solve the collision on X axis. Passed as an argument to std::function.

##### Parameters

|              |  |
|--------------|--|
| <i>block</i> | A block collision is resolved against. Needed to compute resolved <a href="#">Player</a> position. |
|--------------|--|

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.8.3.7 updateAxisX()

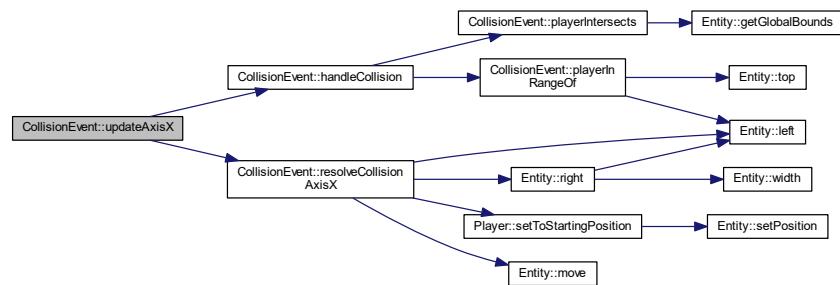
```
void CollisionEvent::updateAxisX (
    float ) [override], [virtual]
```

Handle horizontal collisions.

Implements [PhysicsEvent](#) pure virtual method.

Implements [PhysicsEvent](#).

Here is the call graph for this function:



### 10.8.3.8 updateAxisY()

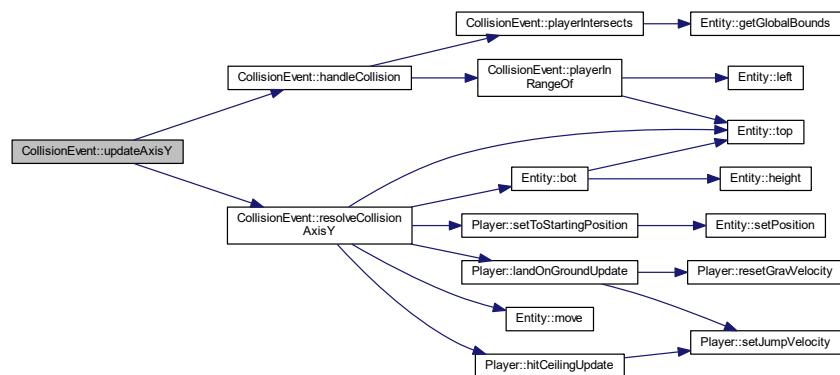
```
void CollisionEvent::updateAxisY (
    float ) [override], [virtual]
```

Handle vertical collisions.

Implements [PhysicsEvent](#) pure virtual method.

Implements [PhysicsEvent](#).

Here is the call graph for this function:



## 10.8.4 Member Data Documentation

### 10.8.4.1 blocks

```
std::vector<std::unique_ptr<Entity>> & CollisionEvent::blocks [private]
```

Collidable blocks.

### 10.8.4.2 collidableRange

```
float CollisionEvent::collidableRange [private]
```

The entities do not have to be squares; in such a case, collidableRange is the longer dimension.

### 10.8.4.3 player

```
Player& CollisionEvent::player [private]
```

Controller.

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

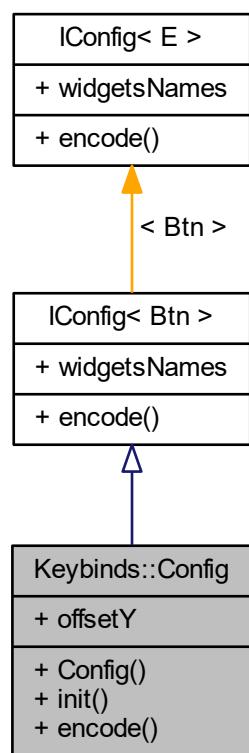
- [CollisionEvent.h](#)
- [CollisionEvent.cpp](#)

## 10.9 Keybinds::Config Struct Reference

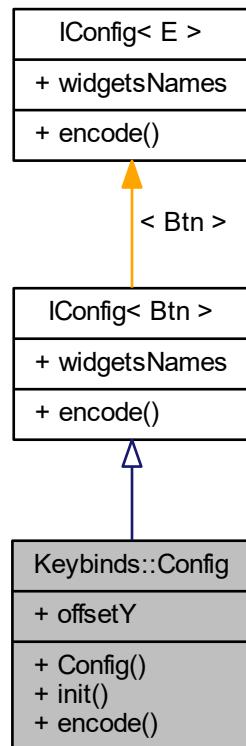
Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [KeybindsView](#).

```
#include <KeybindsConfig.h>
```

Inheritance diagram for Keybinds::Config:



Collaboration diagram for Keybinds::Config:



## Public Member Functions

- `Config ()`
- `void init ()`
- `void encode () override`

## Static Public Attributes

- `constexpr static auto offsetY = 288`  
*Starting buttons drawing offset.*

## Additional Inherited Members

### 10.9.1 Detailed Description

Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `KeybindsView`.

Make sure Initializer list size is equal to Enum

## 10.9.2 Constructor & Destructor Documentation

### 10.9.2.1 Config()

```
Config::Config ()
```

Create a [Config](#). Here is the call graph for this function:



## 10.9.3 Member Function Documentation

### 10.9.3.1 encode()

```
void Config::encode () [override], [virtual]
```

Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.9.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.9.4 Member Data Documentation

#### 10.9.4.1 offsetY

```
constexpr static auto Keybinds::Config::offsetY = 288 [static], [constexpr]
```

Starting buttons drawing offset.

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

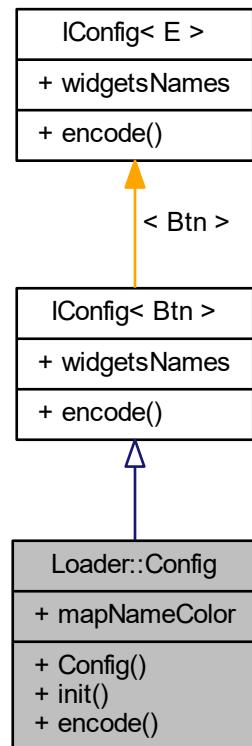
- [KeybindsConfig.h](#)
- [KeybindsConfig.cpp](#)

## 10.10 Loader::Config Struct Reference

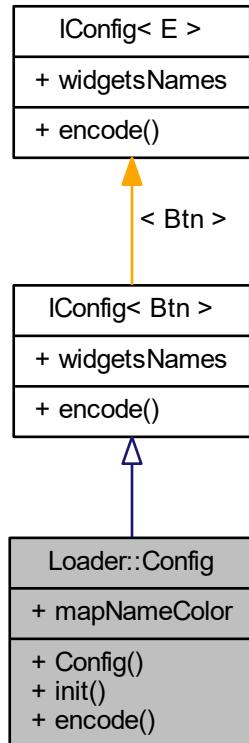
Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `MapLoaderView`.

```
#include <MapLoaderConfig.h>
```

Inheritance diagram for Loader::Config:



Collaboration diagram for Loader::Config:



## Public Member Functions

- [Config \(\)](#)
- void [init \(\)](#)
- void [encode \(\)](#) override

## Public Attributes

- tgui::Color [mapNameColor = {196, 0, 0}](#)

*The color of map name text font.*

### 10.10.1 Detailed Description

Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to [MapLoaderView](#).

Make sure Initializer list size is equal to Enum

### 10.10.2 Constructor & Destructor Documentation

#### 10.10.2.1 Config()

```
Config::Config ()
```

Create a [Config](#). Here is the call graph for this function:



### 10.10.3 Member Function Documentation

#### 10.10.3.1 encode()

```
void Config::encode () [override], [virtual]
```

Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.10.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.10.4 Member Data Documentation

#### 10.10.4.1 mapNameColor

```
tgui::Color Loader::Config::mapNameColor = {196, 0, 0}
```

The color of map name text font.

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

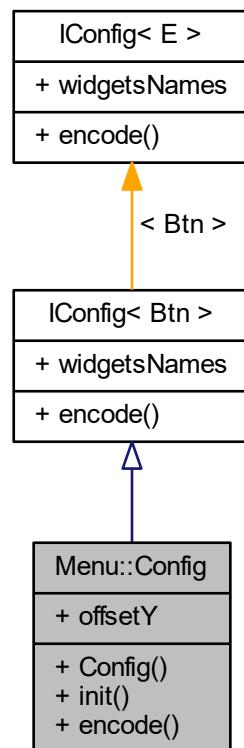
- [MapLoaderConfig.h](#)
- [MapLoaderConfig.cpp](#)

## 10.11 Menu::Config Struct Reference

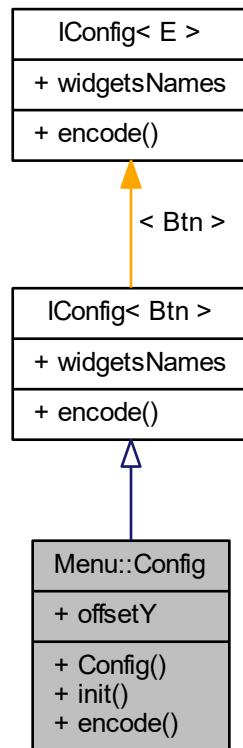
Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `MenuView`.

```
#include <MenuConfig.h>
```

Inheritance diagram for `Menu::Config`:



Collaboration diagram for Menu::Config:



## Public Member Functions

- [Config \(\)](#)
- void [init \(\)](#)
- void [encode \(\)](#) override

## Static Public Attributes

- constexpr static auto [offsetY](#) = 45  
*Starting buttons drawing offset.*

## Additional Inherited Members

### 10.11.1 Detailed Description

Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MenuView](#).

Make sure Initializer list size is equal to Enum

## 10.11.2 Constructor & Destructor Documentation

### 10.11.2.1 Config()

```
Config::Config ()
```

Create a [Config](#). Here is the call graph for this function:



## 10.11.3 Member Function Documentation

### 10.11.3.1 encode()

```
void Config::encode () [override], [virtual]
```

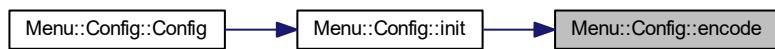
Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



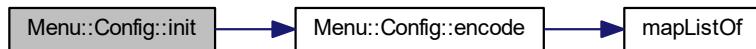
Here is the caller graph for this function:



### 10.11.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



## 10.11.4 Member Data Documentation

### 10.11.4.1 offsetY

```
constexpr static auto Menu::Config::offsetY = 45 [static], [constexpr]
```

Starting buttons drawing offset.

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

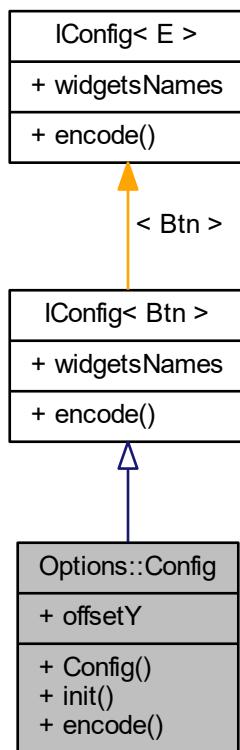
- [MenuConfig.h](#)
- [MenuConfig.cpp](#)

## 10.12 Options::Config Struct Reference

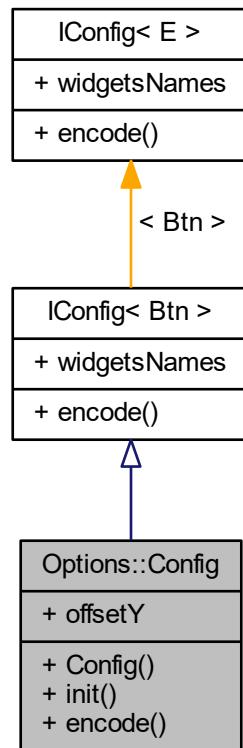
Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to [OptionsView](#).

```
#include <OptionsConfig.h>
```

Inheritance diagram for Options::Config:



Collaboration diagram for Options::Config:



## Public Member Functions

- `Config ()`
- `void init ()`
- `void encode () override`

## Static Public Attributes

- `constexpr static auto offsetY = 350`

*Starting buttons drawing offset.*

## Additional Inherited Members

### 10.12.1 Detailed Description

Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to [OptionsView](#).

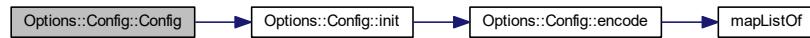
Make sure Initializer list size is equal to Enum

## 10.12.2 Constructor & Destructor Documentation

### 10.12.2.1 Config()

```
Config::Config ( )
```

Create a [Config](#). Here is the call graph for this function:



## 10.12.3 Member Function Documentation

### 10.12.3.1 encode()

```
void Config::encode ( ) [override], [virtual]
```

Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.12.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.12.4 Member Data Documentation

#### 10.12.4.1 offsetY

```
constexpr static auto Options::Config::offsetY = 350 [static], [constexpr]
```

Starting buttons drawing offset.

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

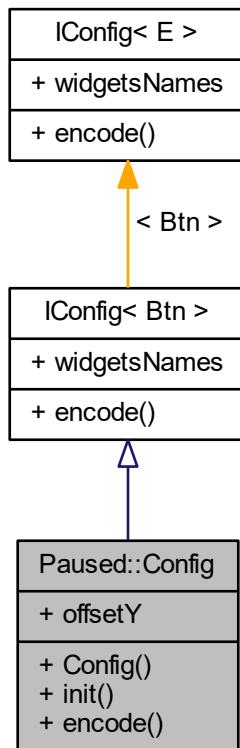
- [OptionsConfig.h](#)
- [OptionsConfig.cpp](#)

## 10.13 Paused::Config Struct Reference

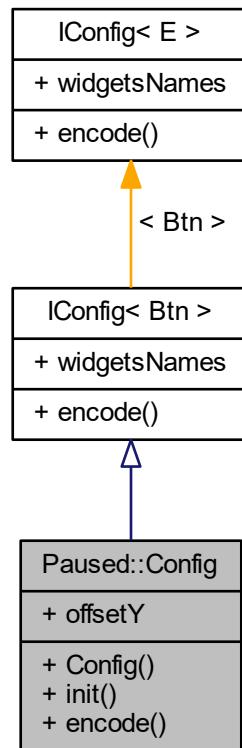
Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `PausedView`.

```
#include <PausedConfig.h>
```

Inheritance diagram for Paused::Config:



Collaboration diagram for Paused::Config:



## Public Member Functions

- `Config ()`
- `void init ()`
- `void encode () override`

## Static Public Attributes

- `constexpr static auto offsetY = 280`  
*Starting buttons drawing offset.*

## Additional Inherited Members

### 10.13.1 Detailed Description

Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `PausedView`.

Make sure Initializer list size is equal to Enum

### 10.13.2 Constructor & Destructor Documentation

#### 10.13.2.1 Config()

```
Config::Config ( )
```

Create a [Config](#). Here is the call graph for this function:



### 10.13.3 Member Function Documentation

#### 10.13.3.1 encode()

```
void Config::encode ( ) [override], [virtual]
```

Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



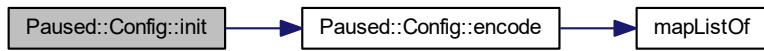
Here is the caller graph for this function:



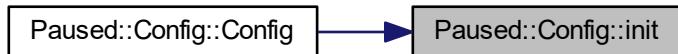
### 10.13.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.13.4 Member Data Documentation

#### 10.13.4.1 offsetY

```
constexpr static auto Paused::Config::offsetY = 280 [static], [constexpr]
```

Starting buttons drawing offset.

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

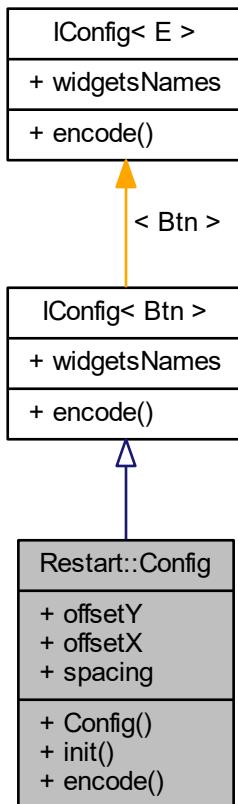
- [PausedConfig.h](#)
- [PausedConfig.cpp](#)

## 10.14 Restart::Config Struct Reference

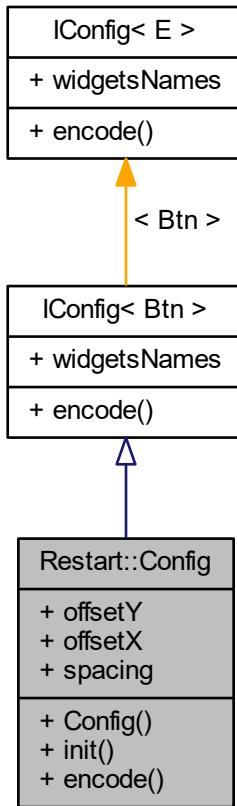
Deriving from `IConfig<Btn>`, holds meaningful view configurations specific to `RestartView`.

```
#include <RestartConfig.h>
```

Inheritance diagram for `Restart::Config`:



Collaboration diagram for Restart::Config:



## Public Member Functions

- `Config ()`
- `void init ()`
- `void encode () override`

## Static Public Attributes

- `constexpr static auto offsetY = 512`

*Starting buttons drawing offset Y.*

- `constexpr static auto offsetX = 60`

*Starting buttons drawing offset X.*

- `constexpr static auto spacing = 20`

*Space between buttons.*

## Additional Inherited Members

### 10.14.1 Detailed Description

Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [RestartView](#).

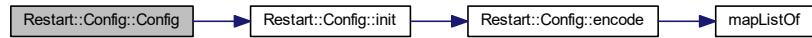
Make sure Initializer list size is equal to Enum

### 10.14.2 Constructor & Destructor Documentation

#### 10.14.2.1 Config()

```
Config::Config ()
```

Create a [Config](#). Here is the call graph for this function:



### 10.14.3 Member Function Documentation

#### 10.14.3.1 encode()

```
void Config::encode () [override], [virtual]
```

Encode button enum accessors to a given strings.

Implements [IConfig< Btn >](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.14.3.2 init()

```
void Config::init ( )
```

Run encode. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.14.4 Member Data Documentation

#### 10.14.4.1 offsetX

```
constexpr static auto Restart::Config::offsetX = 60 [static], [constexpr]
```

Starting buttons drawing offset X.

#### 10.14.4.2 offsetY

```
constexpr static auto Restart::Config::offsetY = 512 [static], [constexpr]
```

Starting buttons drawing offset Y.

#### 10.14.4.3 spacing

```
constexpr static auto Restart::Config::spacing = 20 [static], [constexpr]
```

Space between buttons.

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

- [RestartConfig.h](#)
- [RestartConfig.cpp](#)

## 10.15 Gui::Config< Widget > Struct Template Reference

A shared gui config. Contains common settings.

```
#include <Config.h>
```

Collaboration diagram for Gui::Config< Widget >:



### Public Member Functions

- `void prepare (Widget widget)`

*Prepares a given widget to be ready to be displayed.*

## Public Attributes

- const tgui::Color `borderColor` = {209, 153, 48}
- const tgui::Color `backgroundColor` = {255, 50, 245}

## Static Public Attributes

- constexpr static auto `width` = 250
- constexpr static auto `height` = 50
- constexpr static auto `textSize` = 24
- constexpr static auto `opacity` = 0.95
- static const tgui::Font `font` = {config::widgetsFontName.data()}

### 10.15.1 Detailed Description

```
template<typename Widget = tgui::Widget::Ptr>
struct Gui::Config< Widget >
```

A shared gui config. Contains common settings.

Exemplary constexpr Config settings common to all View classes:

- `Buttonwidth`: `#width` = 250
- `ButtonHeight`: `#height` = 50
- `#testSize` = 24
- `#opacity` = 0.95
- `#font`
- `#borderColor`
- `#backgroundColor`

### 10.15.2 Member Function Documentation

#### 10.15.2.1 `prepare()`

```
template<typename Widget = tgui::Widget::Ptr>
void Gui::Config< Widget >::prepare (
    Widget widget ) [inline]
```

Prepares a given widget to be ready to be displayed.

An extra 'if constexpr' branch is executed for Buttons that sets the `borderColor` and `backgroundColor`.

### 10.15.3 Member Data Documentation

### 10.15.3.1 backgroundColor

```
template<typename Widget = tgui::Widget::Ptr>
const tgui::Color Gui::Config< Widget >::backgroundColor = {255, 50, 245}
```

### 10.15.3.2 borderColor

```
template<typename Widget = tgui::Widget::Ptr>
const tgui::Color Gui::Config< Widget >::borderColor = {209, 153, 48}
```

### 10.15.3.3 font

```
template<typename Widget = tgui::Widget::Ptr>
const tgui::Font Gui::Config< Widget >::font = {config::widgetsFontName.data()}{inline},
[static]
```

### 10.15.3.4 height

```
template<typename Widget = tgui::Widget::Ptr>
constexpr static auto Gui::Config< Widget >::height = 50 [static], [constexpr]
```

### 10.15.3.5 opacity

```
template<typename Widget = tgui::Widget::Ptr>
constexpr static auto Gui::Config< Widget >::opacity = 0.95 [static], [constexpr]
```

### 10.15.3.6 textSize

```
template<typename Widget = tgui::Widget::Ptr>
constexpr static auto Gui::Config< Widget >::textSize = 24 [static], [constexpr]
```

### 10.15.3.7 width

```
template<typename Widget = tgui::Widget::Ptr>
constexpr static auto Gui::Config< Widget >::width = 250 [static], [constexpr]
```

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

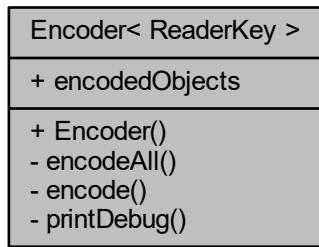
- [Config.h](#)

## 10.16 Encoder< ReaderKey > Class Template Reference

Encodes [PixelColor](#) (bmp) or int (txt) into [Obj::Entity](#) entities, depending on the variant type.

```
#include <Encoder.h>
```

Collaboration diagram for Encoder< ReaderKey >:



### Public Member Functions

- `constexpr Encoder ()`  
*Validates ReaderKey type and creates an [Encoder](#) of that type.*

### Public Attributes

- `std::map< ReaderKey, Obj::Entity > encodedObjects`

### Private Member Functions

- `void encodeAll ()`  
*Encodes all [PixelColor](#) or int into [Obj::Entity](#) entities.*
- `void encode (const ReaderKey &key, Obj::Entity entity)`  
*Encodes a single Readerkey onto [Obj::Entity](#) entity.*
- `void printDebug ()`

### 10.16.1 Detailed Description

```
template<typename ReaderKey>
class Encoder< ReaderKey >
```

Encodes [PixelColor](#) (bmp) or int (txt) into [Obj::Entity](#) entities, depending on the variant type.

Uses custom Mappable concept (C++20) for validating if a ReaderKey is mappable type. In case of C++17 compiler, the Mappable concept is omitted.

### 10.16.2 Constructor & Destructor Documentation

#### 10.16.2.1 Encoder()

```
template<typename ReaderKey >
constexpr Encoder< ReaderKey >::Encoder ( ) [inline], [constexpr]
```

Validates ReaderKey type and creates an [Encoder](#) of that type.

Available ReaderKey types:

- [PixelColor](#) (when map is [Bmp](#))
- int (when map is [Txt](#))

Here is the call graph for this function:



### 10.16.3 Member Function Documentation

#### 10.16.3.1 encode()

```
template<typename ReaderKey >
void Encoder< ReaderKey >::encode (
    const ReaderKey & key,
    Obj::Entity entity ) [inline], [private]
```

Encodes a single Readerkey onto [Obj::Entity](#) entity.

**Parameters**

|               |   |
|---------------|---|
| <i>key</i>    | Map key of type ReaderKey which is a Mappable either <a href="#">PixelColor</a> or int. |
| <i>entity</i> | Value for the key to be mapped to.  |

Here is the caller graph for this function:

**10.16.3.2 encodeAll()**

```
template<typename ReaderKey >
void Encoder< ReaderKey >::encodeAll ( ) [inline], [private]
```

Encodes all [PixelColor](#) or int into [Obj::Entity](#) entities.

**Note**

One of two branches is executed based on constexpr evaluation.

< Light pink

< White

< Light violet

< Dark brown

< Dark blue

< Soft dark blue

< Orange

< Soft orange

< Light grey

< Light grey

< Light grey

< Light greyHere is the call graph for this function:



Here is the caller graph for this function:



### 10.16.3.3 printDebug()

```
template<typename ReaderKey >
void Encoder< ReaderKey >::printDebug ( ) [inline], [private]
```

\bref Prints formatted encoder debug info.

#### Note

Supports only non-trivial [Bmp](#) type.

todo: add support for txt

## 10.16.4 Member Data Documentation

### 10.16.4.1 encodedObjects

```
template<typename ReaderKey >
std::map<ReaderKey, Obj::Entity> Encoder< ReaderKey >::encodedObjects
```

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

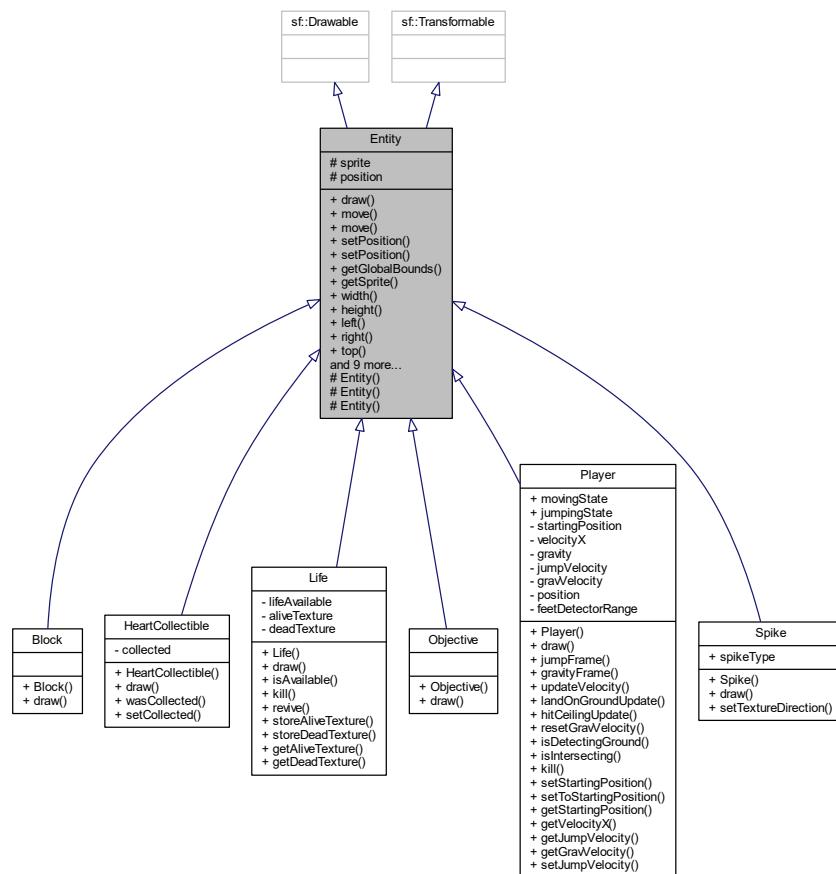
- [Encoder.h](#)

## 10.17 Entity Class Reference

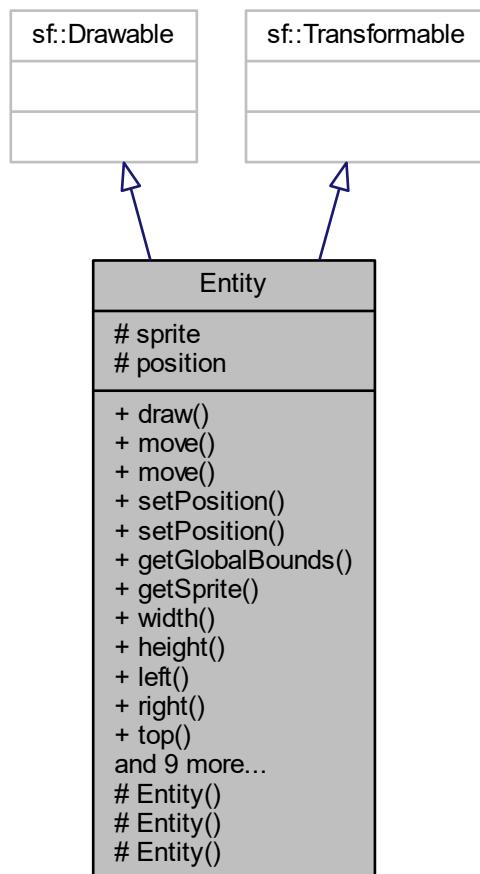
A base drawable [Entity](#) class. All drawable entities (game objects) inherit from it.

```
#include <Entity.h>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



## Public Member Functions

- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override  
*Draws the `sprite` onto the target.*
- void **move** (const sf::Vector2f &offset)  
*Move the entity `sprite` by a float vector.*
- void **move** (float offsetX, float offsetY)  
*Move the entity `sprite` by a explicit float coordinates.*
- void **setPosition** (const sf::Vector2f &position)  
*Sets `sprite` position.*
- void **setPosition** (int x, int y)  
*Sets `sprite` position.*
- sf::FloatRect **getGlobalBounds** () const  
*Retrieve global bounds of the `sprite`.*
- sf::Sprite & **getSprite** ()  
*Retrieve `sprite` itself.*
- float **width** () const

- float `height` () const
  - float `left` () const
    - Retrieve `sprite` x coordinate.*
  - float `right` () const
    - Retrieve `sprite` right side x coordinate.*
  - float `top` () const
    - Retrieve `sprite` y coordinate.*
  - float `bot` () const
    - Retrieve `sprite` bottom side y coordinate.*
- void `rotateLeft` ()
  - Rotate the sprite Left by 90 degrees.*
- void `rotateRight` ()
  - Rotate the sprite Right by 90 degrees.*
- void `rotateTop` ()
  - Rotate the sprite by 180 degrees.*
- virtual ~`Entity` ()=default
  - Enable default destructor.*
- `Entity` (`Entity` const &)=default
  - Enable default copy operations.*
- `Entity` (`Entity` &&) noexcept=default
  - Enable default move operations.*
- `Entity` & operator= (`Entity` const &)=default
  - Enable default copy operations.*
- `Entity` & operator= (`Entity` &&) noexcept=default
  - Enable default move operations.*

## Protected Member Functions

- `Entity` (`sf::Vector2f` position)
- `Entity` (`ResourceHolder< res::Texture, sf::Texture >` &`textureHolder`, `res::Texture` `textureID`, `sf::Vector2f` `position`)
- `Entity` (`ResourceHolder< res::Texture, sf::Texture >` &`textureHolder`, `res::Texture` `textureID`)

## Protected Attributes

- `sf::Sprite` `sprite`
  - Entity sprite.*
- `sf::Vector2f` `position`

### 10.17.1 Detailed Description

A base drawable `Entity` class. All drawable entities (game objects) inherit from it.

Contains helper methods common to all entitie e.g. setting position, retrieving position.

Enables all move and copy operations (Rule of 5).

#### Examples

`C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp`.

## 10.17.2 Constructor & Destructor Documentation

### 10.17.2.1 ~Entity()

```
virtual Entity::~Entity ( ) [virtual], [default]
```

Enable default destructor.

### 10.17.2.2 Entity() [1/5]

```
Entity::Entity ( Entity const & ) [default]
```

Enable default copy operations.

### 10.17.2.3 Entity() [2/5]

```
Entity::Entity ( Entity && ) [default], [noexcept]
```

Enable default move operations.

### 10.17.2.4 Entity() [3/5]

```
Entity::Entity ( sf::Vector2f position ) [explicit], [protected]
```

Creates an entity at a given position

#### Parameters

|                 |                          |
|-----------------|--------------------------|
| <i>position</i> | starting Entity position |
|-----------------|--------------------------|

#### Note

Entity constructor is protected so only deriving classes can create it to be more meaningful.

### 10.17.2.5 Entity() [4/5]

```
Entity::Entity (
    ResourceHolder< res::Texture, sf::Texture > & textureHolder,
    res::Texture textureID,
    sf::Vector2f position ) [protected]
```

Creates an entity at a given position.

#### Parameters

|                      |   |
|----------------------|---|
| <i>textureHolder</i> | textureHolder reference container                     |
| <i>textureID</i>     | ID of the texture to be set to <a href="#">Entity</a> |
| <i>position</i>      | starting <a href="#">Entity</a> position              |

#### Note

This construction uses sets default position to (0, 0)

[Entity](#) constructor is protected so only deriving classes can create it to be more meaningful.

### 10.17.2.6 Entity() [5/5]

```
Entity::Entity (
    ResourceHolder< res::Texture, sf::Texture > & textureHolder,
    res::Texture textureID ) [protected]
```

## 10.17.3 Member Function Documentation

### 10.17.3.1 bot()

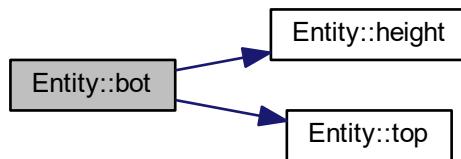
```
float Entity::bot ( ) const
```

Retrieve [sprite](#) bottom side y coordinate.

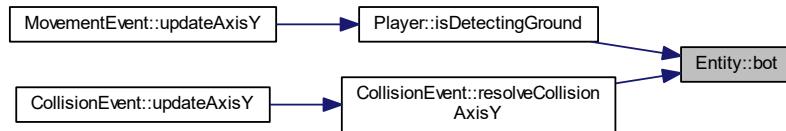
**Returns**

[sprite](#) bottom side y coordinate

Here is the call graph for this function:



Here is the caller graph for this function:

**10.17.3.2 draw()**

```
void Entity::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draws the [sprite](#) onto the target.

**Parameters**

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides pure virtual in [sf::Drawable](#) class.

**Note**

Passing states is optional.

### 10.17.3.3 getGlobalBounds()

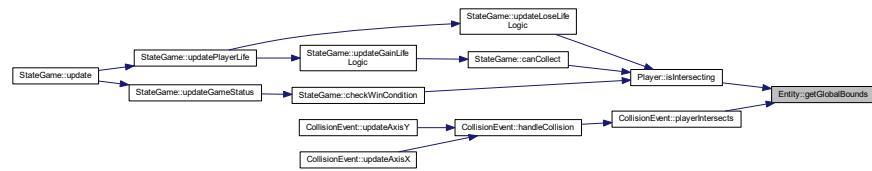
```
sf::FloatRect Entity::getGlobalBounds() const
```

Retrieve global bounds of the [sprite](#).

#### Returns

`sf::FloatRect` [sprite](#) global bounds.

Here is the caller graph for this function:



### 10.17.3.4 getSprite()

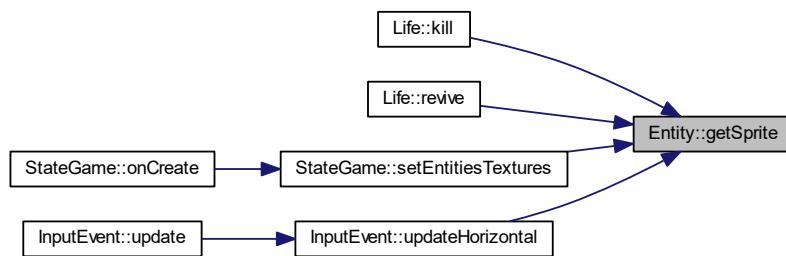
```
sf::Sprite & Entity::getSprite() 
```

Retrieve [sprite](#) itself.

#### Returns

`sf::Sprite&` a reference to a [sprite](#).

Here is the caller graph for this function:



### 10.17.3.5 height()

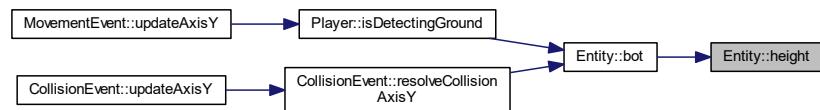
```
float Entity::height () const
```

Retrieve a height of the [sprite](#).

Returns

float [sprite](#) height.

Here is the caller graph for this function:



### 10.17.3.6 left()

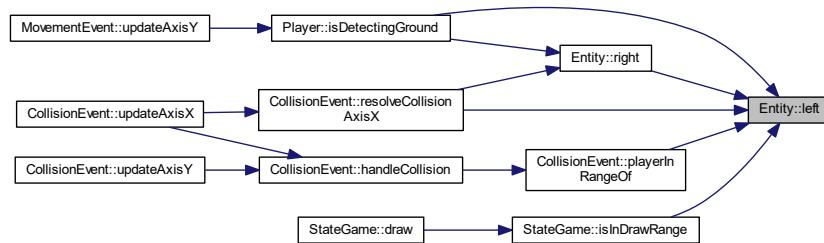
```
float Entity::left () const
```

Retrieve [sprite](#) x coordinate.

Returns

float [sprite](#) x coordinate

Here is the caller graph for this function:



### 10.17.3.7 move() [1/2]

```
void Entity::move (
    const sf::Vector2f & offset )
```

Move the entity [sprite](#) by a float vector.

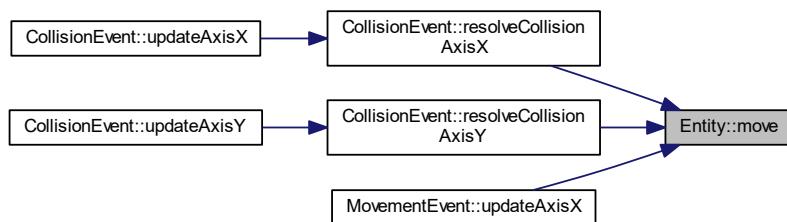
**Parameters**

|               |             |
|---------------|-------------|
| <i>offset</i> | move vector |
|---------------|-------------|

**Note**

Offset is **added** to current sprite position.

Here is the caller graph for this function:

**10.17.3.8 move() [2/2]**

```
void Entity::move (
    float offsetX,
    float offsetY )
```

Move the entity [sprite](#) by a explicit float coordinates.

**Parameters**

|                |  |
|----------------|--|
| <i>offsetX</i> |  |
| <i>offsetY</i> |  |

**Note**

Offset is **added** to current sprite position.

**10.17.3.9 operator=() [1/2]**

```
Entity& Entity::operator= (
    Entity && ) [default], [noexcept]
```

Enable default move operations.

**10.17.3.10 operator=() [2/2]**

```
Entity& Entity::operator= (
    Entity const & )  [default]
```

Enable default copy operations.

**10.17.3.11 right()**

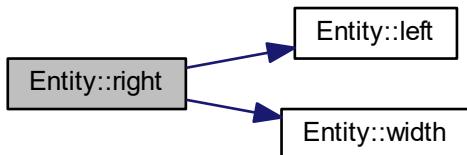
```
float Entity::right ( ) const
```

Retrieve [sprite](#) right side x coordinate.

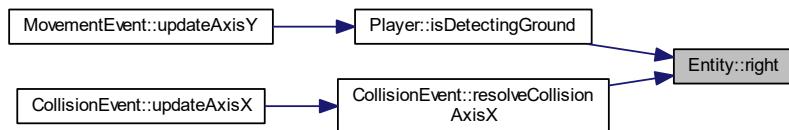
**Returns**

[sprite](#) right side x coordinate

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.17.3.12 `rotateLeft()`

```
void Entity::rotateLeft ( )
```

Rotate the sprite Left by 90 degrees.

Here is the caller graph for this function:

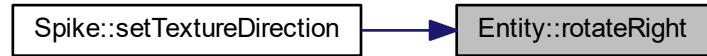


#### 10.17.3.13 `rotateRight()`

```
void Entity::rotateRight ( )
```

Rotate the sprite Right by 90 degrees.

Here is the caller graph for this function:

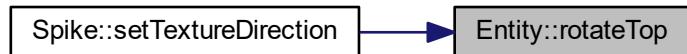


#### 10.17.3.14 `rotateTop()`

```
void Entity::rotateTop ( )
```

Rotate the sprite by 180 degrees.

Here is the caller graph for this function:



**10.17.3.15 setPosition() [1/2]**

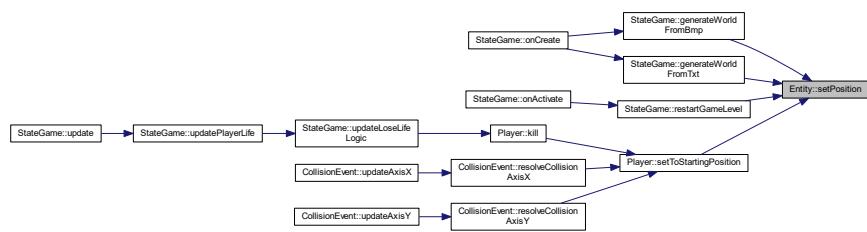
```
void Entity::setPosition (
    const sf::Vector2f & position )
```

Sets [sprite](#) position.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <i>position</i> | given by Vector2f |
|-----------------|-------------------|

Here is the caller graph for this function:

**10.17.3.16 setPosition() [2/2]**

```
void Entity::setPosition (
    int x,
    int y )
```

Sets [sprite](#) position.

**Parameters**

|                 |                                |
|-----------------|--------------------------------|
| <i>position</i> | given by explicit coordinates. |
|-----------------|--------------------------------|

**10.17.3.17 top()**

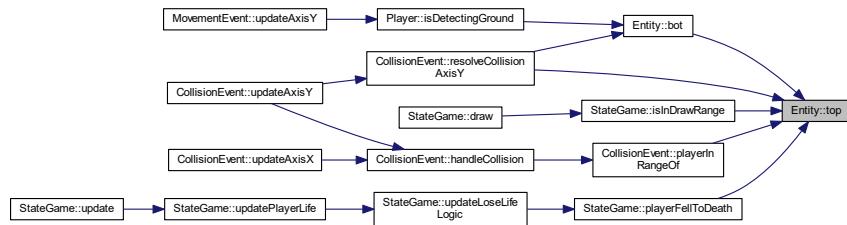
```
float Entity::top ( ) const
```

Retrieve [sprite](#) y coordinate.

**Returns**

```
float sprite y coordinate
```

Here is the caller graph for this function:

**10.17.3.18 width()**

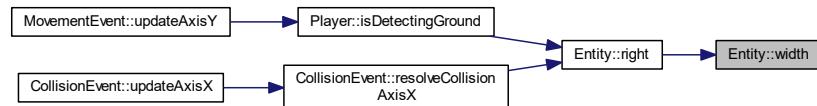
```
float Entity::width() const
```

Retrieve a width of the [sprite](#).

**Returns**

```
float sprite width.
```

Here is the caller graph for this function:

**10.17.4 Member Data Documentation****10.17.4.1 position**

```
sf::Vector2f Entity::position [protected]
```

### 10.17.4.2 sprite

```
sf::Sprite Entity::sprite [protected]
```

[Entity](#) sprite.

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

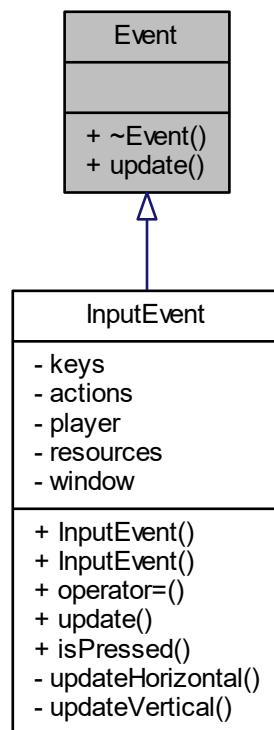
- [Entity.h](#)
- [Entity.cpp](#)

## 10.18 Event Class Reference

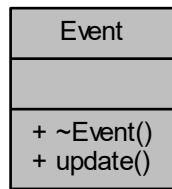
[Event](#) class with [update\(\)](#) pure virtual method handling trivial events.

```
#include <Event.h>
```

Inheritance diagram for Event:



Collaboration diagram for Event:



## Public Member Functions

- virtual `~Event ()=default`  
*Default the destructor.*
- virtual void `update ()=0`  
*Handles simplest events. Pure virtual update method to be overriden by deriving classes.*

### 10.18.1 Detailed Description

Event class with `update()` pure virtual method handling trivial events.

### 10.18.2 Constructor & Destructor Documentation

#### 10.18.2.1 ~Event()

```
virtual Event::~Event ( ) [virtual], [default]
```

Default the destructor.

### 10.18.3 Member Function Documentation

#### 10.18.3.1 update()

```
virtual void Event::update ( ) [pure virtual]
```

Handles simplest events. Pure virtual update method to be overriden by deriving classes.

Implemented in [InputEvent](#).

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

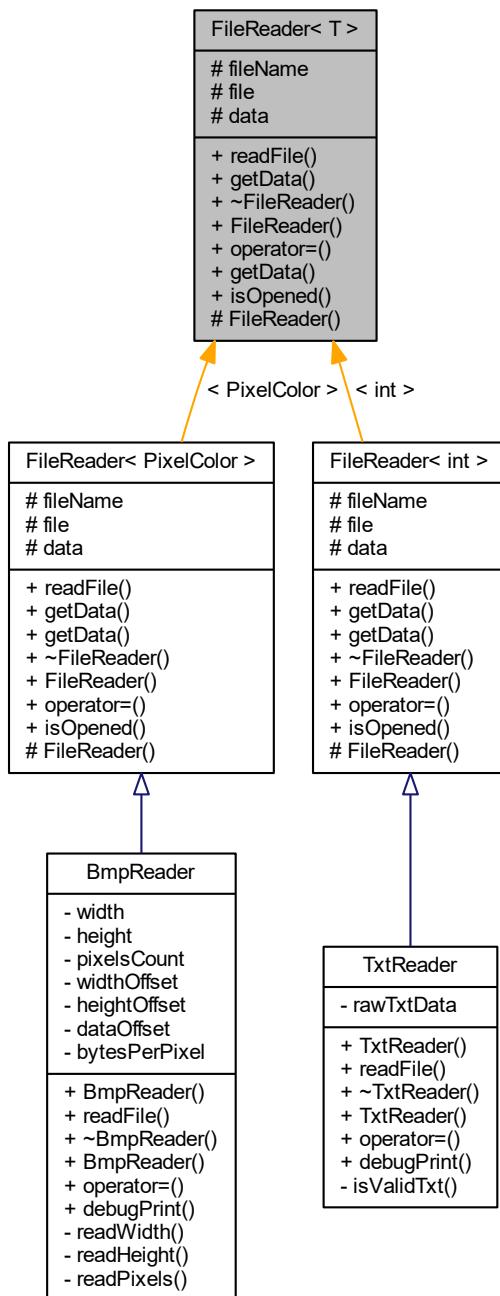
- [Event.h](#)

## 10.19 FileReader< T > Class Template Reference

Abstract File Reader class template.

```
#include <FileReader.h>
```

Inheritance diagram for FileReader< T >:



Collaboration diagram for FileReader< T >:

| FileReader< T >   |
|---|
| # fileName<br># file<br># data  |
| + readFile()<br>+ getData()<br>+ ~FileReader()<br>+ FileReader()<br>+ operator=(<br>+ getData()<br>+ isOpened()<br># FileReader() |

## Public Member Functions

- virtual void `readFile ()=0`  
*Pure virtual method to be overridden by deriving `TxtReader` and `TxtWriter`.*
- auto & `getData ()`
- virtual `~FileReader ()=default`  
*Default Virtual constructor.*
- `FileReader (FileReader &&) noexcept=default`  
*Enable move operations.*
- `FileReader & operator= (FileReader &&) noexcept=default`  
*Enable move operations.*
- auto & `getData () const`
- bool `isOpened () const`  
*Check if a file is successfully opened.*

## Protected Member Functions

- `FileReader (const std::string &name, const std::ios_base::openmode &mode=std::ifstream::in)`

## Protected Attributes

- `std::string fileName`  
*File name.*
- `std::ifstream file`  
*File handle.*
- `std::vector< T > data {}`  
*Containre for data of type T.*

### 10.19.1 Detailed Description

```
template<typename T>
class FileReader< T >
```

Abstract File Reader class template.

#### Note

Inheriting classes: [BmpReader](#) and [TxtReader](#).

### 10.19.2 Constructor & Destructor Documentation

#### 10.19.2.1 ~FileReader()

```
template<typename T >
virtual FileReader< T >::~FileReader ( ) [virtual], [default]
```

Default Virtual constructor.

#### 10.19.2.2 FileReader() [1/2]

```
template<typename T >
FileReader< T >::FileReader (
    FileReader< T > && ) [default], [noexcept]
```

Enable move operations.

#### 10.19.2.3 FileReader() [2/2]

```
template<typename T >
FileReader< T >::FileReader (
    const std::string & name,
    const std::ios_base::openmode & mode = std::ifstream::in ) [inline], [explicit],
[protected]
```

### 10.19.3 Member Function Documentation

**10.19.3.1 `getData()` [1/2]**

```
template<typename T >
auto& FileReader< T >::getData ( ) [inline]
```

Retrieve the data contained in a file.

**Returns**

`std::vector<T>` Vector of data of type T.

**10.19.3.2 `getData()` [2/2]**

```
template<typename T >
auto& FileReader< T >::getData ( ) const [inline]
```

**Note**

: `basic_ifstream` has copy ctor/copy assignment explicitly deleted.

**10.19.3.3 `isOpened()`**

```
template<typename T >
bool FileReader< T >::isOpened ( ) const [inline]
```

Check if a file is successfully opened.

**Returns**

Is file opened?

**10.19.3.4 `operator=()`**

```
template<typename T >
FileReader& FileReader< T >::operator= (
    FileReader< T > && ) [default], [noexcept]
```

Enable move operations.

### 10.19.3.5 `readFile()`

```
template<typename T >
virtual void FileReader< T >::readFile ( ) [pure virtual]
```

Pure virtual method to be overriden by deriving [TxtReader](#) and [TxtReader](#).

Implemented in [BmpReader](#), and [TxtReader](#).

## 10.19.4 Member Data Documentation

### 10.19.4.1 `data`

```
template<typename T >
std::vector<T> FileReader< T >::data {} [protected]
```

Containre for data of type T.

### 10.19.4.2 `file`

```
template<typename T >
std::ifstream FileReader< T >::file [protected]
```

File handle.

### 10.19.4.3 `fileName`

```
template<typename T >
std::string FileReader< T >::fileName [protected]
```

File name.

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

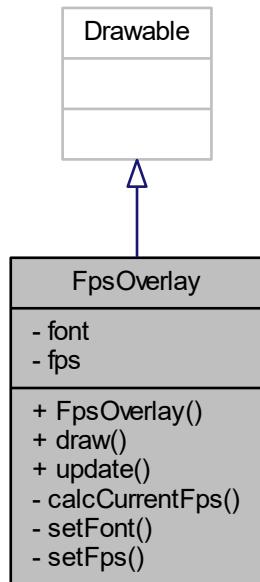
- [FileReader.h](#)

## 10.20 FpsOverlay Class Reference

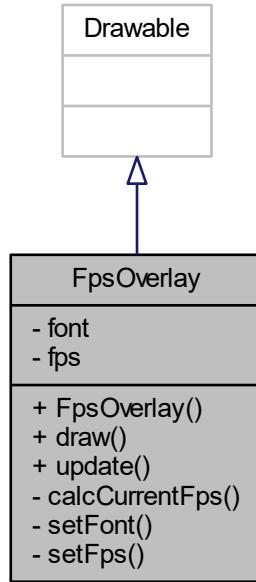
Prints the Fps to the screen.

```
#include <FpsOverlay.h>
```

Inheritance diagram for FpsOverlay:



Collaboration diagram for FpsOverlay:



## Public Member Functions

- `FpsOverlay ()`  
*Constructs an `FpsOverlay`. Calls `setFont()` and `setFps()`.*
- `void draw (sf::RenderTarget &target, sf::RenderStates states) const override`  
*Draw method used to render `fps` onto the window.*
- `void update (float dt)`

## Private Member Functions

- `std::string calcCurrentFps (float dt)`
- `void setFont ()`  
*Sets `font` to customizable `config::widgetsFontName`.*
- `void setFps ()`  
*Sets the `fps` using the `font` and text configurations.*

## Private Attributes

- `sf::Font font`  
*Font used to initialize the text.*
- `sf::Text fps`  
*Fps text number drawn to the screen in left-bottom corner.*

### 10.20.1 Detailed Description

Prints the Fps to the screen.

#### Note

Fps can be turned off and on by game player at any time during run-time.

Updated every game frame.

### 10.20.2 Constructor & Destructor Documentation

#### 10.20.2.1 FpsOverlay()

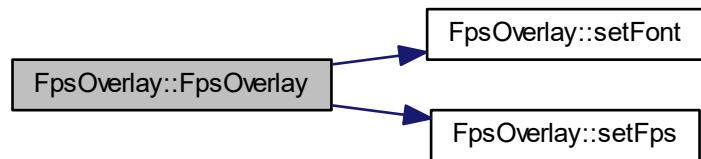
```
FpsOverlay::FpsOverlay ( )
```

Constructs an [FpsOverlay](#). Calls [setFont\(\)](#) and [setFps\(\)](#).

#### Warning

May throw.

Here is the call graph for this function:



### 10.20.3 Member Function Documentation

### 10.20.3.1 calcCurrentFps()

```
std::string FpsOverlay::calcCurrentFps (
    float dt ) [private]
```

Calculate current Fps based on time elapsed since previous frame.

#### Returns

`std::string` number of Fps expressed as a `std::string` for a convenience.

Here is the caller graph for this function:



### 10.20.3.2 draw()

```
void FpsOverlay::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render `fps` onto the window.

#### Parameters

|                     |   |
|---------------------|---|
| <code>target</code> | <code>Window</code> for the <code>fps</code> to be rendered onto. |
| <code>states</code> | Optional render states  |

Overrides [Entity](#) draw method.

#### Note

Passing states is optional.

### 10.20.3.3 setFont()

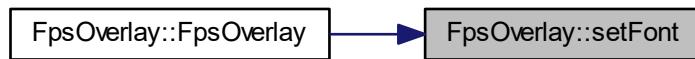
```
void FpsOverlay::setFont ( ) [private]
```

Sets `font` to customizable [config::widgetsFontName](#).

**Warning**

May throw.

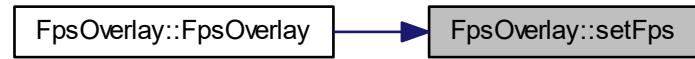
Here is the caller graph for this function:

**10.20.3.4 setFps()**

```
void FpsOverlay::setFps ( ) [private]
```

Sets the [fps](#) using the [font](#) and text configurations.

Here is the caller graph for this function:

**10.20.3.5 update()**

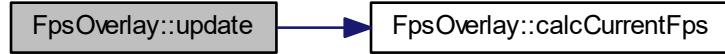
```
void FpsOverlay::update (
    float dt )
```

Update the displayed number of [fps](#) based on time elapsed since previous frame.

**Parameters**

|           |                                    |
|-----------|------------------------------------|
| <i>dt</i> | Time elapsed since previous frame. |
|-----------|------------------------------------|

Here is the call graph for this function:



Here is the caller graph for this function:



## 10.20.4 Member Data Documentation

### 10.20.4.1 font

```
sf::Font FpsOverlay::font [private]
```

Font used to initialize the text.

### 10.20.4.2 fps

```
sf::Text FpsOverlay::fps [private]
```

Fps text number drawn to the screen in left-bottom corner.

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

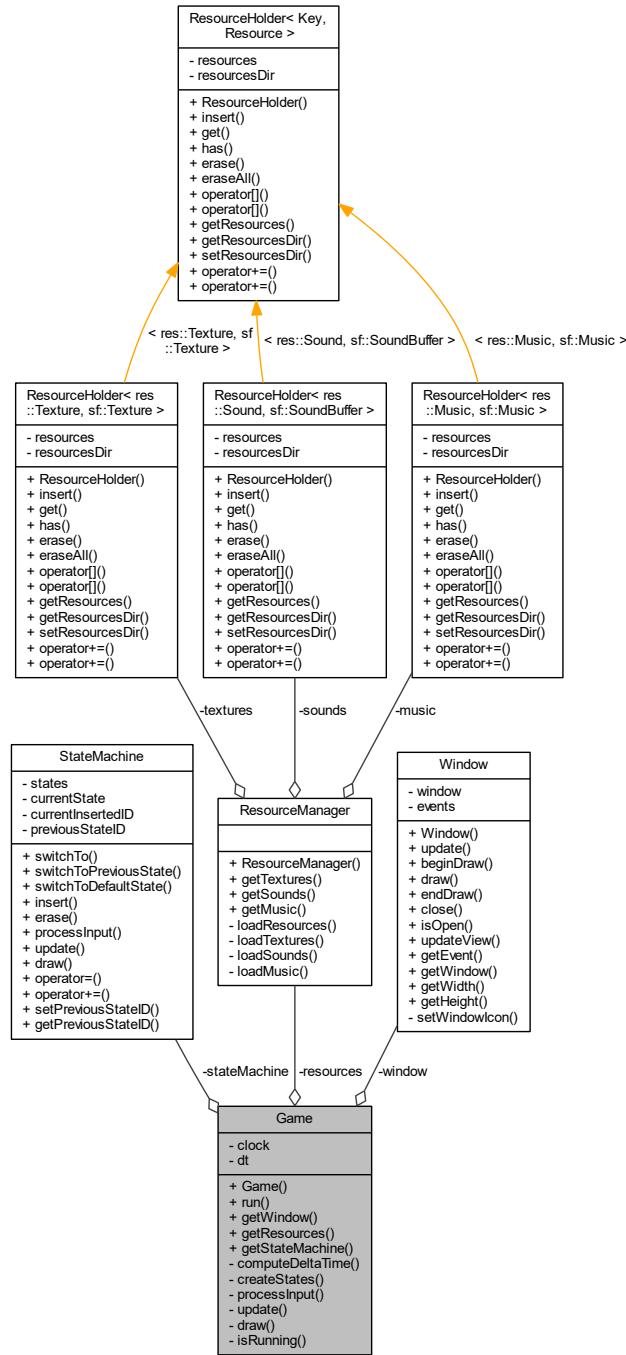
- [FpsOverlay.h](#)
- [FpsOverlay.cpp](#)

## 10.21 Game Class Reference

Container for player actions keybinds.

```
#include <Game.h>
```

Collaboration diagram for Game:



## Public Member Functions

- [Game \(\)](#)  
*Creates a Game.*
- [void run \(\)](#)  
*Runs the game: called from main, this is where the main game loop starts.*
- [Window & getWindow \(\)](#)  
*Get the window instance.*
- [ResourceManager & getResources \(\)](#)  
*Get the resources instance.*
- [StateMachine & getStateMachine \(\)](#)  
*Get the stateMachine instance.*

## Private Member Functions

- [void computeDeltaTime \(\)](#)  
*The function computes the time elapsed since the previous frame. The result is stored as seconds inside dt variable (float).*
- [void createStates \(\)](#)  
*Creates all game states but StateGame during Game construction time.*
- [void processInput \(\)](#)  
*Calls the StateMachine::processInput() method.*
- [void update \(\)](#)  
*Calls the StateMachine::update(float dt) and Window.update() methods.*
- [void draw \(\)](#)  
*Calls the StateMachine.draw() method.*
- [bool isRunning \(\) const](#)  
*Checks if a Game is still running.*

## Private Attributes

- [Window window](#)  
*The Window where the game is drawn.*
- [ResourceManager resources](#)  
*Contains persistent resources needed throughout game loop.*
- [StateMachine stateMachine](#)  
*Manages the game states (See State design pattern)*
- [sf::Clock clock](#)  
*Utility class that measures the elapsed time.*
- [float dt](#)  
*Time measured since previous game frame.*

### 10.21.1 Detailed Description

Container for player actions keybinds.

A fundamental class that runs the game loop. Invoked from `main()`, game execution starts and ends here.

`Keybinds` are held within `#actionMap`.

A given keybind can be bound to only one action, so binding an action to a previously existing keybind invalidates that previous action bind.

#### Note

This is only one singleton class within this whole project; and having just one singleton in a non-trivial project is not necessarily considered harmful. My singleton implementation is **correctly-destroyed, lazy-evaluated, and thread-safe**.

- |See *Singleton design pattern*)

### 10.21.2 Constructor & Destructor Documentation

#### 10.21.2.1 Game()

```
Game::Game ( )
```

Creates a `Game`.

### 10.21.3 Member Function Documentation

#### 10.21.3.1 computeDeltaTime()

```
void Game::computeDeltaTime ( ) [private]
```

The function computes the time elapsed since the previous frame. The result is stored as seconds inside `dt` variable (`float`).

Here is the caller graph for this function:



### 10.21.3.2 `createStates()`

```
void Game::createStates ( ) [private]
```

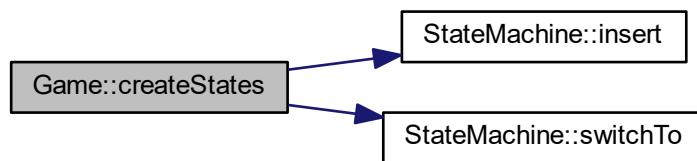
Creates all game states but [StateGame](#) during [Game](#) construction time.

Created states persist until the end of the game (*except for StateGame*).

A list of the available states:

- [StateAbout](#)
- [StateGame](#)
- [StateKeybinds](#)
- [StateMapLoader](#)
- [StateMenu](#)
- [StateOptions](#)
- [StatePaused](#)
- [StateRestart](#)

Here is the call graph for this function:



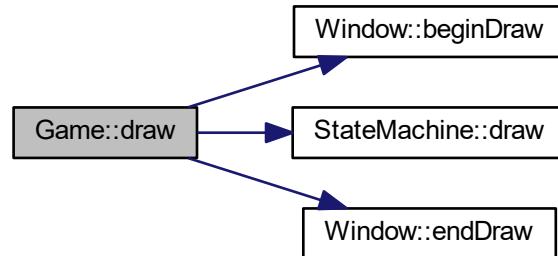
### 10.21.3.3 `draw()`

```
void Game::draw ( ) [private]
```

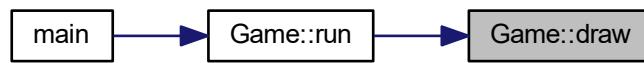
Calls the [StateMachine.draw\(\)](#) method.

This method is continuously executed within a game loop.

(See [State design pattern](#)) Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.21.3.4 `getResources()`

```
ResourceManager & Game::getResources( )
```

Get the `resources` instance.

##### Returns

Reference to an instance of `ResourceManager` class.

#### 10.21.3.5 `getStateMachine()`

```
StateMachine & Game::getStateMachine( )
```

Get the `stateMachine` instance.

##### Returns

Reference to an instance of `StateMachine`.

### 10.21.3.6 getWindow()

```
Window & Game::getWindow ( )
```

Get the [window](#) instance.

#### Returns

Reference to an instance of [Window](#) class.

### 10.21.3.7 isRunning()

```
bool Game::isRunning ( ) const [private]
```

Checks if a [Game](#) is still running.

Boolean condition for ending the game.

#### Note

A [Game](#) is running as long as a [Window](#) is being kept open.

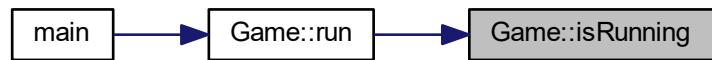
#### Returns

Is the game still running?

Here is the call graph for this function:



Here is the caller graph for this function:



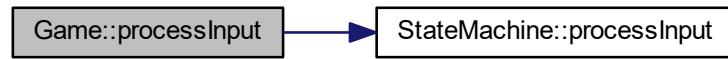
### 10.21.3.8 processInput()

```
void Game::processInput ( ) [private]
```

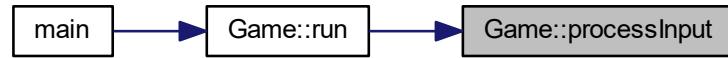
Calls the [StateMachine::processInput\(\)](#) method.

This method is continuously executed within a game loop.

(See [State design pattern](#)) Here is the call graph for this function:



Here is the caller graph for this function:

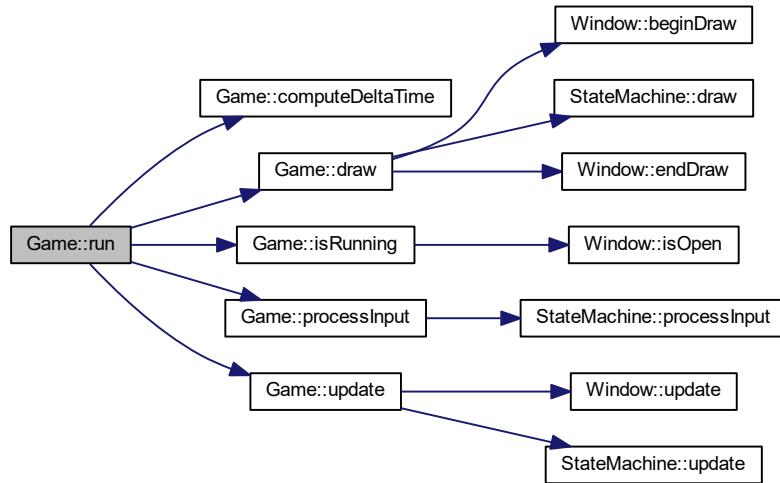


### 10.21.3.9 run()

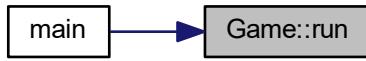
```
void Game::run ( )
```

Runs the game: called from main, this is where the main game loop starts.

Here is the call graph for this function:



Here is the caller graph for this function:



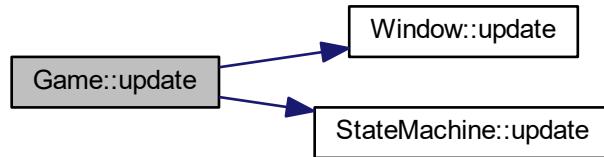
#### 10.21.3.10 `update()`

```
void Game::update ( ) [private]
```

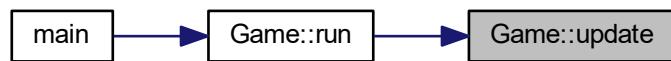
Calls the `StateMachine::update(float dt)` and `Window.update()` methods.

This method is continuously executed within a game loop.

(See [State design pattern](#)) Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.21.4 Member Data Documentation

##### 10.21.4.1 clock

```
sf::Clock Game::clock [private]
```

Utility class that measures the elapsed time.

##### 10.21.4.2 dt

```
float Game::dt [private]
```

Time measured since previous game frame.

#### 10.21.4.3 resources

```
ResourceManager Game::resources [private]
```

Contains persistent resources needed throughout game loop.

#### 10.21.4.4 stateMachine

```
StateMachine Game::stateMachine [private]
```

Manages the game states (See *State design pattern*)

#### 10.21.4.5 window

```
Window Game::window [private]
```

The [Window](#) where the game is drawn.

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

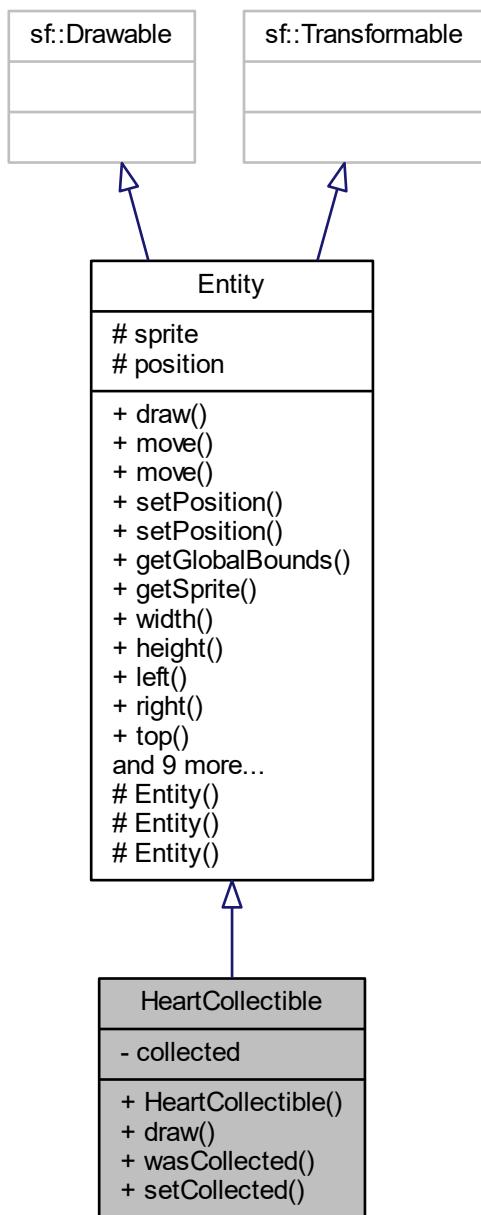
- [Game.h](#)
- [Game.cpp](#)

## 10.22 HeartCollectible Class Reference

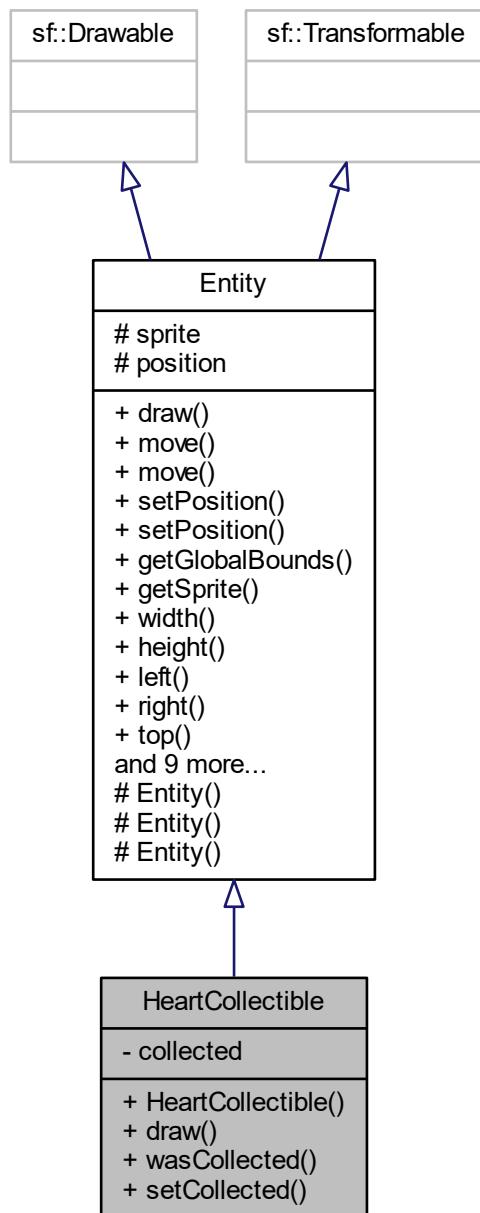
Used to represent the collectible hearts (player lives).

```
#include <HeartCollectible.h>
```

Inheritance diagram for HeartCollectible:



Collaboration diagram for HeartCollectible:



## Public Member Functions

- **HeartCollectible (sf::Vector2f position)**  
*Constructs a HeartCollectible.*
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override  
*Draw method used to render sprite onto the window.*
- bool **wasCollected ()** const
- void **setCollected (bool status)**

## Private Attributes

- bool `collected` = false  
*True if HeartCollectible was collected.*

## Additional Inherited Members

### 10.22.1 Detailed Description

Used to represent the collectible hearts (player lives).

#### Note

A concrete [Entity](#) class overriding the draw method.

### 10.22.2 Constructor & Destructor Documentation

#### 10.22.2.1 HeartCollectible()

```
HeartCollectible::HeartCollectible (
    sf::Vector2f position )  [explicit]
```

Constructs a [HeartCollectible](#).

#### Parameters

|                       |                  |
|-----------------------|------------------|
| <code>position</code> | initial position |
|-----------------------|------------------|

### 10.22.3 Member Function Documentation

#### 10.22.3.1 draw()

```
void HeartCollectible::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const  [override]
```

Draw method used to render [sprite](#) onto the window.

#### Parameters

|                     |  |
|---------------------|--|
| <code>target</code> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <code>states</code> | Optional render states                                     |

Overrides [Entity](#) draw method.

#### Note

Passing states is optional.

### 10.22.3.2 setCollected()

```
void HeartCollectible::setCollected ( bool status )
```

Sets the [collected](#) status to a new one.

#### Parameters

|                     |                                      |
|---------------------|--------------------------------------|
| <code>status</code> | new <a href="#">collected</a> status |
|---------------------|--------------------------------------|

### 10.22.3.3 wasCollected()

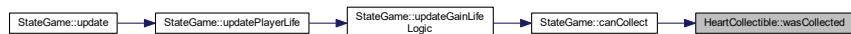
```
bool HeartCollectible::wasCollected ( ) const
```

Checks if the heart was already [collected](#).

#### Returns

Was the heart already [collected](#)?

Here is the caller graph for this function:



## 10.22.4 Member Data Documentation

### 10.22.4.1 collected

```
bool HeartCollectible::collected = false [private]
```

True if [HeartCollectible](#) was collected.

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

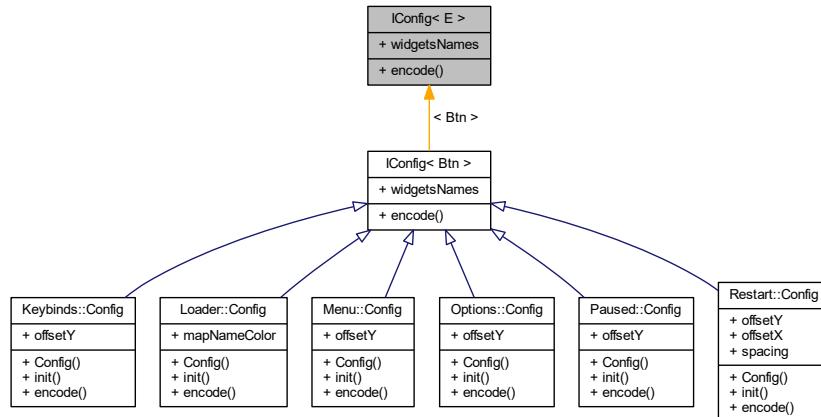
- [HeartCollectible.h](#)
- [HeartCollectible.cpp](#)

## 10.23 IConfig< E > Struct Template Reference

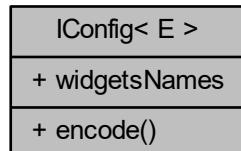
Every specific Config inherits from this struct.

```
#include <IConfig.h>
```

Inheritance diagram for IConfig< E >:



Collaboration diagram for IConfig< E >:



### Public Member Functions

- virtual void `encode ()=0`

### Public Attributes

- `std::map< E, const char * > widgetsNames`

### 10.23.1 Detailed Description

```
template<typename E>
struct IConfig< E >
```

Every specific Config inherits from this struct.

#### Note

Deriving classes must implement encode pure virtual method.

#### Warning

The *E* template type has to be an *Enum*.

### 10.23.2 Member Function Documentation

#### 10.23.2.1 encode()

```
template<typename E >
virtual void IConfig< E >::encode ( ) [pure virtual]
```

C++20: constexpr virtual

Implemented in [Keybinds::Config](#), [Menu::Config](#), [Loader::Config](#), [Options::Config](#), [Paused::Config](#), and [Restart::Config](#).

### 10.23.3 Member Data Documentation

#### 10.23.3.1 widgetsNames

```
template<typename E >
std::map<E, const char*> IConfig< E >::widgetsNames
```

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

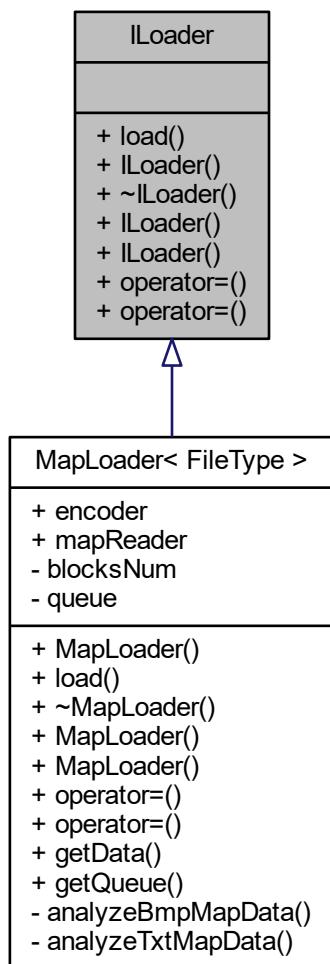
- [IConfig.h](#)

## 10.24 ILoader Class Reference

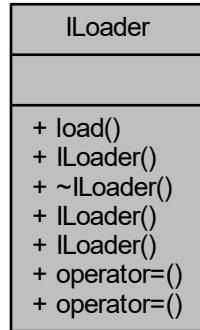
A base interface for a map loader.

```
#include <MapLoader.h>
```

Inheritance diagram for ILoader:



Collaboration diagram for ILoader:



## Public Member Functions

- virtual void `load ()=0`  
*A pure virtual method to be overriden by deriving classes.*
- `ILoader ()=default`  
*Enable default constructor.*
- virtual `~ILoader ()=default`  
*Enable default virtual destructor.*
- `ILoader (const ILoader &)=default`  
*Enable default copy operations.*
- `ILoader (ILoader &&) noexcept=default`  
*Enable default move operations.*
- `ILoader & operator= (const ILoader &)=default`  
*Enable default copy operations.*
- `ILoader & operator= (ILoader &&) noexcept=default`  
*Enable default move operations.*

### 10.24.1 Detailed Description

A base interface for a map loader.

`load()` is a pure virtual method to be overriden by deriving classes.

### 10.24.2 Constructor & Destructor Documentation

#### 10.24.2.1 ILoader() [1/3]

```
ILoader::ILoader () [default]
```

Enable default constructor.

#### 10.24.2.2 ~ILoader()

```
virtual ILoader::~ILoader () [virtual], [default]
```

Enable default virtual destructor.

#### 10.24.2.3 ILoader() [2/3]

```
ILoader::ILoader (
    const ILoader & ) [default]
```

Enable default copy operations.

#### 10.24.2.4 ILoader() [3/3]

```
ILoader::ILoader (
    ILoader && ) [default], [noexcept]
```

Enable default move operations.

### 10.24.3 Member Function Documentation

#### 10.24.3.1 load()

```
virtual void ILoader::load () [pure virtual]
```

A pure virtual method to be overridden by deriving classes.

Implemented in [MapLoader< FileType >](#).

Here is the caller graph for this function:



### 10.24.3.2 operator=( ) [1/2]

```
ILoader& ILoader::operator= (
    const ILoader & ) [default]
```

Enable default copy operations.

### 10.24.3.3 operator=( ) [2/2]

```
ILoader& ILoader::operator= (
    ILoader && ) [default], [noexcept]
```

Enable default move operations.

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

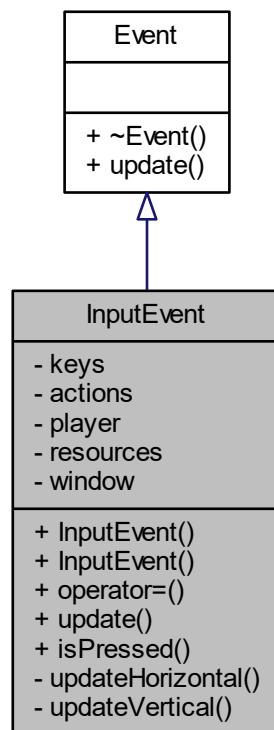
- [MapLoader.h](#)

## 10.25 InputEvent Class Reference

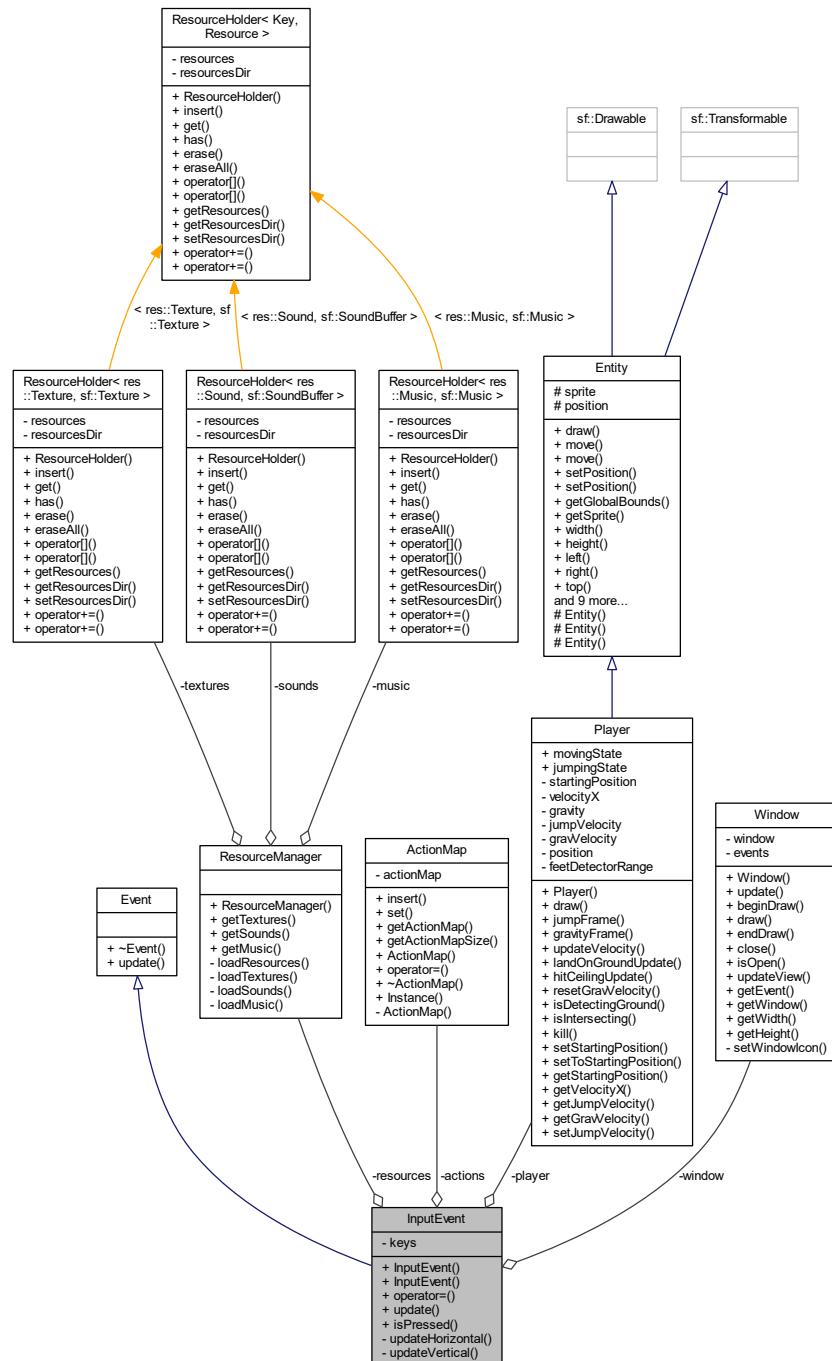
Handles the user input within [StateGame](#) class. Uses [ActionMap](#).

```
#include <InputEvent.h>
```

Inheritance diagram for InputEvent:



Collaboration diagram for InputEvent:



## Public Member Functions

- `InputEvent (Player &player, ResourceManager &resources, Window &window)`  
*Create an `InputEvent` instance.*
- `InputEvent (const InputEvent &)=delete`  
*Delete copy operations.*
- `InputEvent & operator= (const InputEvent &)=delete`

- Delete copy operations.*
- void [update \(\)](#) override
    - Updates horizontal and vertical input events.*

## Static Public Member Functions

- static bool [isPressed \(sf::Keyboard::Key myKeyCode\)](#)

## Private Member Functions

- void [updateHorizontal \(\)](#)  
*Updates Horizontal events.*
- void [updateVertical \(\)](#)  
*Update Vertical events.*

## Private Attributes

- std::unordered\_map< std::string, sf::Keyboard::Key > [keys](#)
- [ActionMap & actions](#) = [ActionMap::Instance\(\)](#)
- [Player & player](#)
- [ResourceManager & resources](#)
- [Window & window](#)

### 10.25.1 Detailed Description

Handles the user input within [StateGame](#) class. Uses [ActionMap](#).

### 10.25.2 Constructor & Destructor Documentation

#### 10.25.2.1 InputEvent() [1/2]

```
InputEvent::InputEvent (
    Player & player,
    ResourceManager & resources,
    Window & window )  [explicit]
```

Create an [InputEvent](#) instance.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <i>player</i>    | player controller       |
| <i>resources</i> | the resources container |
| <i>window</i>    |                         |

**Note**

Populates the `keys` with default-set user keybindings.

### 10.25.2.2 `InputEvent()` [2/2]

```
InputEvent::InputEvent (
    const InputEvent & ) [delete]
```

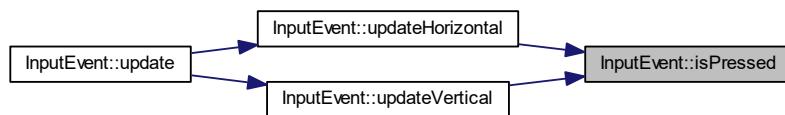
Delete copy operations.

## 10.25.3 Member Function Documentation

### 10.25.3.1 `isPressed()`

```
bool InputEvent::isPressed (
    sf::Keyboard::Key myKeyCode ) [static]
```

Here is the caller graph for this function:



### 10.25.3.2 `operator=()`

```
InputEvent& InputEvent::operator= (
    const InputEvent & ) [delete]
```

Delete copy operations.

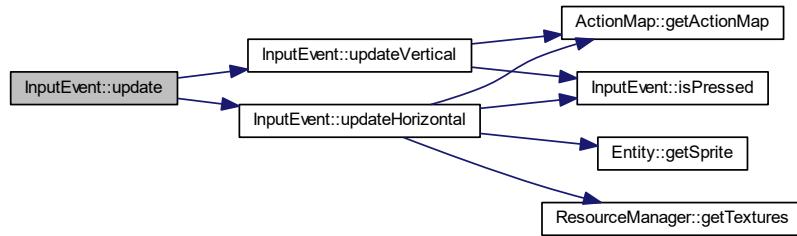
### 10.25.3.3 update()

```
void InputEvent::update () [override], [virtual]
```

Updates horizontal and vertical input events.

Implements [Event](#).

Here is the call graph for this function:



### 10.25.3.4 updateHorizontal()

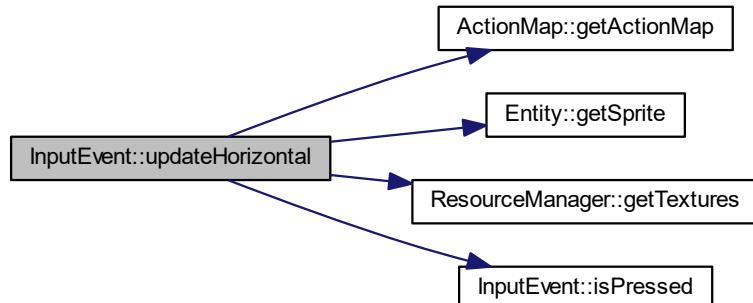
```
void InputEvent::updateHorizontal () [private]
```

Updates Horizontal events.

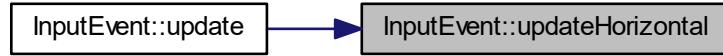
Horizontal actions:

- Running Left
- Running Right
- Standing

Update the [Player](#) texture accordingly. Here is the call graph for this function:



Here is the caller graph for this function:

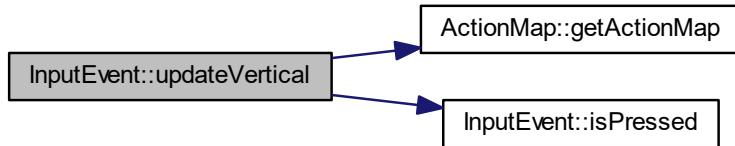


#### 10.25.3.5 updateVertical()

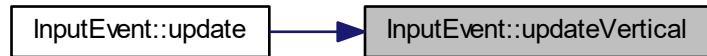
```
void InputEvent::updateVertical () [private]
```

Update Vertical events.

Checks if a [Player](#) is standing on the ground, and only then checks if [Player](#) tries to jump. Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.25.4 Member Data Documentation

#### 10.25.4.1 actions

```
ActionMap& InputEvent::actions = ActionMap::Instance() [private]
```

#### 10.25.4.2 keys

```
std::unordered_map<std::string, sf::Keyboard::Key> InputEvent::keys [private]
```

#### 10.25.4.3 player

```
Player& InputEvent::player [private]
```

#### 10.25.4.4 resources

```
ResourceManager& InputEvent::resources [private]
```

#### 10.25.4.5 window

```
Window& InputEvent::window [private]
```

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

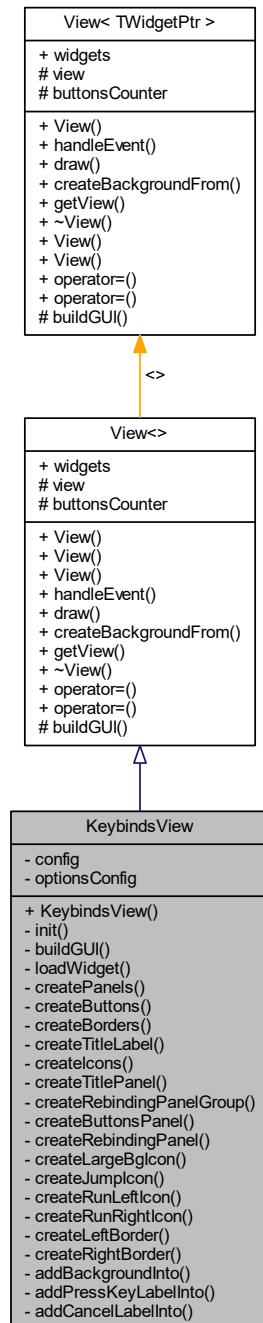
- [InputEvent.h](#)
- [InputEvent.cpp](#)

## 10.26 KeybindsView Class Reference

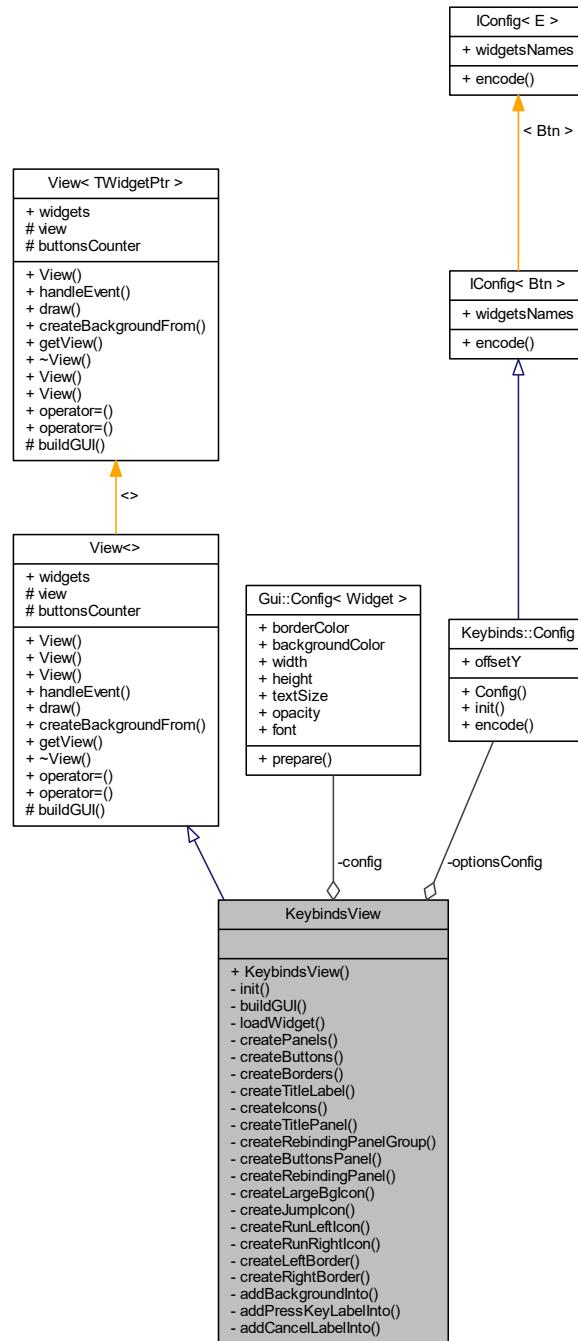
[View](#) class used to draw gui within [StateKeybinds](#).

```
#include <KeybindsView.h>
```

Inheritance diagram for KeybindsView:



Collaboration diagram for KeybindsView:



## Public Member Functions

- [KeybindsView \(Window &window\)](#)

*Constructs the KeybindsView.*

## Private Member Functions

- void `init()`  
*Initializes the [KeybindsView](#). Effectively: calls `buildGUI()`.*
- void `buildGUI()` override
- void `loadWidget(tgui::Widget::Ptr &widget)`  
*Loads a given widget (likely: a button) onto the correct settings and position.*
- void `createPanels()`  
*Creates all panels and adds them to the #gui.*
- void `createButtons()`  
*Creates all buttons and adds them to the #gui.*
- void `createBorders()`  
*Creates all borders and adds them to the #gui.*
- void `createTitleLabel()`
- void `createIcons()`
- void `createTitlePanel()`  
*Creates the title panel and adds it to the #gui.*
- void `createRebindingPanelGroup()`
- void `createButtonsPanel()`
- tgui::Panel::Ptr `createRebindingPanel()`  
*Create Rebinding Panel parent widgets holder.*
- void `createLargeBgIcon()`
- void `createJumpIcon()`  
*Create Jump action icon and add it to the #gui.*
- void `createRunLeftIcon()`  
*Create RunLeft action icon and add it to the #gui.*
- void `createRunRightIcon()`  
*Create RunRight action icon and add it to the #gui.*
- void `createLeftBorder()`  
*Create left side texture border.*
- void `createRightBorder()`  
*Create right side texture border.*

## Static Private Member Functions

- static void `addBackgroundInto(const tgui::Panel::Ptr &parent)`
- static void `addPressKeyLabelInto(const tgui::Panel::Ptr &parent)`
- static void `addCancelLabelInto(const tgui::Panel::Ptr &parent)`

## Private Attributes

- `Gui::Config config`  
*Configuration settings common to all [View](#) classes.*
- `Keybinds::Config optionsConfig`  
*Configuration settings specific to [KeybindsView](#).*

## Additional Inherited Members

### 10.26.1 Detailed Description

[View](#) class used to draw gui within [StateKeybinds](#).

## 10.26.2 Constructor & Destructor Documentation

### 10.26.2.1 KeybindsView()

```
KeybindsView::KeybindsView (
    Window & window ) [explicit]
```

Constructs the [KeybindsView](#).

#### Parameters

|                        |   |
|------------------------|---|
| <a href="#">Window</a> | that is bound to <a href="#">View</a> base. |
|------------------------|---|

## 10.26.3 Member Function Documentation

### 10.26.3.1 addBackgroundInto()

```
void KeybindsView::addBackgroundInto (
    const tgui::Panel::Ptr & parent ) [static], [private]
```

Adds a registering background into the parent widget.

#### Parameters

|               |                        |
|---------------|------------------------|
| <i>parent</i> | of created background. |
|---------------|------------------------|

Here is the caller graph for this function:



### 10.26.3.2 addCancelLabelInto()

```
void KeybindsView::addCancelLabelInto (
    const tgui::Panel::Ptr & parent ) [static], [private]
```

Adds a cancel registering new keybind label prompt message into the parent.

**Parameters**

|               |                   |
|---------------|-------------------|
| <i>parent</i> | of created label. |
|---------------|-------------------|

Here is the caller graph for this function:

**10.26.3.3 addPressKeyLabelInto()**

```

void KeybindsView::addPressKeyLabelInto (
    const tgui::Panel::Ptr & parent )  [static], [private]
  
```

Adds a press key to rebind label prompt message into the parent widget.

**Parameters**

|               |                   |
|---------------|-------------------|
| <i>parent</i> | of created label. |
|---------------|-------------------|

Here is the caller graph for this function:

**10.26.3.4 buildGUI()**

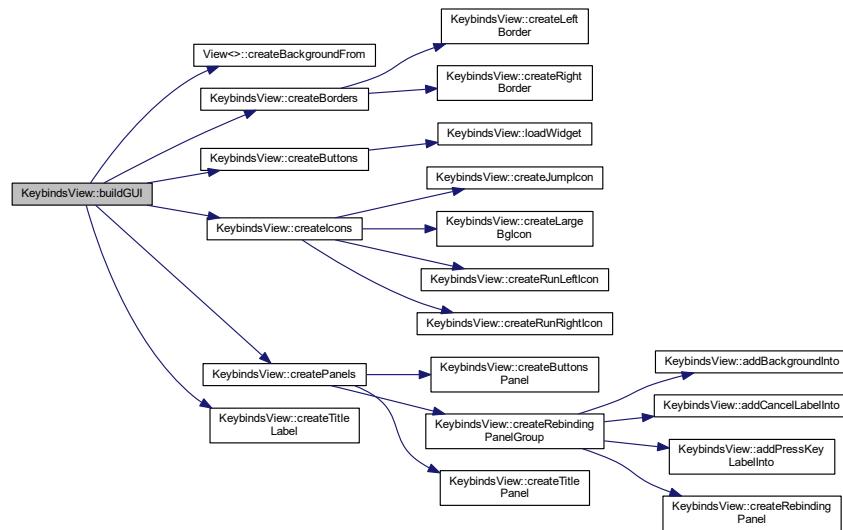
```

void KeybindsView::buildGUI ( )  [override], [private], [virtual]
  
```

Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements [View<>](#).

Here is the call graph for this function:



Here is the caller graph for this function:

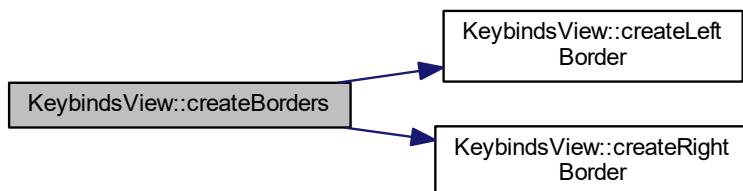


### 10.26.3.5 createBorders()

```
void KeybindsView::createBorders () [private]
```

Creates all borders and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.26.3.6 createButtons()

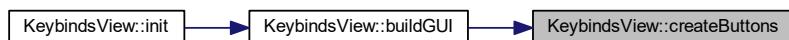
```
void KeybindsView::createButtons () [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.26.3.7 createButtonsPanel()

```
void KeybindsView::createButtonsPanel () [private]
```

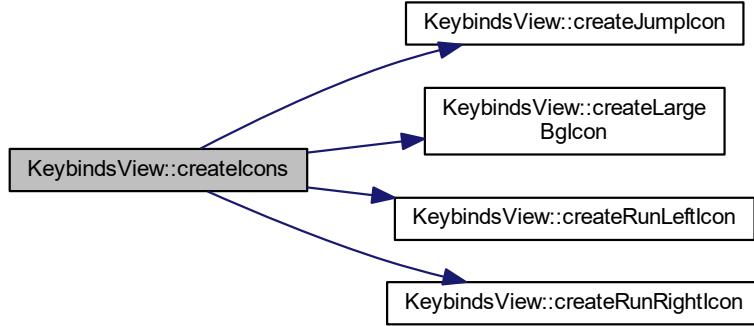
Creates buttons panel and adds it to the #gui. Here is the caller graph for this function:



### 10.26.3.8 createIcons()

```
void KeybindsView::createIcons ( ) [private]
```

Create icons and adds them to the #gui. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.26.3.9 createJumpIcon()

```
void KeybindsView::createJumpIcon ( ) [private]
```

Create Jump action icon and add it to the #gui.

Here is the caller graph for this function:



### 10.26.3.10 `createLargeBgIcon()`

```
void KeybindsView::createLargeBgIcon ( ) [private]
```

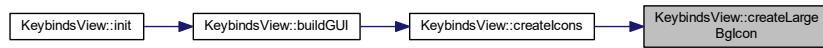
Create large background icon and add it to the #gui.

#### Note

The icon is displayed in the middle of screen, with a show/hide animation.

Icon hides when the sliding RebindingPanel appears.

Here is the caller graph for this function:

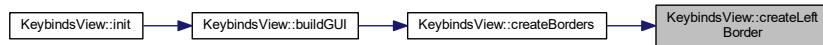


### 10.26.3.11 `createLeftBorder()`

```
void KeybindsView::createLeftBorder ( ) [private]
```

Create left side texture border.

Here is the caller graph for this function:

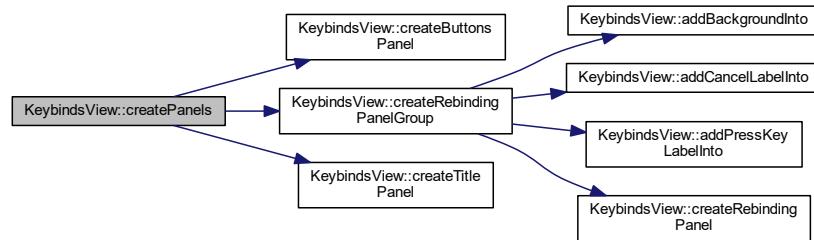


### 10.26.3.12 `createPanels()`

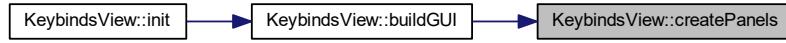
```
void KeybindsView::createPanels ( ) [private]
```

Creates all panels and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.26.3.13 createRebindingPanel()

```
tgui::Panel::Ptr KeybindsView::createRebindingPanel() [private]
```

Create Rebinding Panel parent widgets holder.

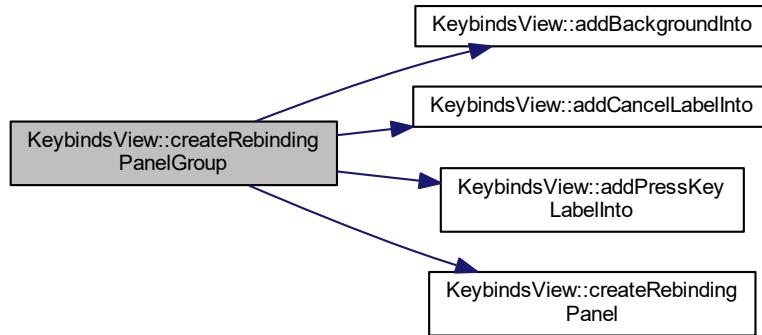
Here is the caller graph for this function:



### 10.26.3.14 createRebindingPanelGroup()

```
void KeybindsView::createRebindingPanelGroup() [private]
```

Creates whole Rebinding panel group and adds it to the #gui. Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.26.3.15 createRightBorder()

```
void KeybindsView::createRightBorder ( ) [private]
```

Create right side texture border.

Here is the caller graph for this function:

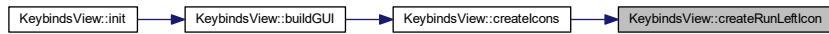


### 10.26.3.16 createRunLeftIcon()

```
void KeybindsView::createRunLeftIcon ( ) [private]
```

Create RunLeft action icon and add it to the #gui.

Here is the caller graph for this function:



### 10.26.3.17 createRunRightIcon()

```
void KeybindsView::createRunRightIcon ( ) [private]
```

Create RunRight action icon and add it to the #gui.

Here is the caller graph for this function:



### 10.26.3.18 createTitleLabel()

```
void KeybindsView::createTitleLabel ( ) [private]
```

Creates the [StateKeybinds](#) title label and adds it to the #gui. Here is the caller graph for this function:

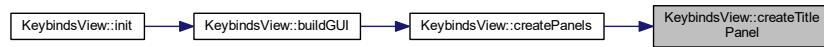


### 10.26.3.19 `createTitlePanel()`

```
void KeybindsView::createTitlePanel ( ) [private]
```

Creates the title panel and adds it to the #gui.

Here is the caller graph for this function:

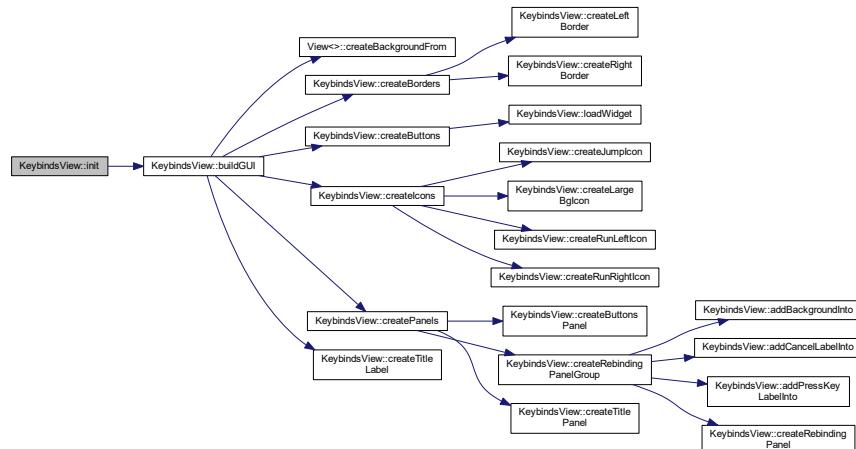


### 10.26.3.20 `init()`

```
void KeybindsView::init ( ) [private]
```

Initializes the [KeybindsView](#). Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



### 10.26.3.21 `loadWidget()`

```
void KeybindsView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position.

**Parameters**

|               |               |
|---------------|---------------|
| <i>widget</i> | to be loaded. |
|---------------|---------------|

**Note**

Can be used in a loop.

Here is the caller graph for this function:



## 10.26.4 Member Data Documentation

### 10.26.4.1 config

`Gui::Config KeybindsView::config [private]`

Configuration settings common to all [View](#) classes.

### 10.26.4.2 optionsConfig

`Keybinds::Config KeybindsView::optionsConfig [private]`

Configuration settings specific to [KeybindsView](#).

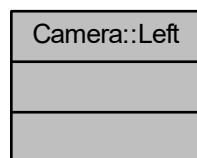
The documentation for this class was generated from the following files:

- [KeybindsView.h](#)
- [KeybindsView.cpp](#)

## 10.27 Camera::Left Struct Reference

Empty struct denoting previously solved on other axis was [Left](#) collision.

Collaboration diagram for Camera::Left:



### 10.27.1 Detailed Description

Empty struct denoting previously solved on other axis was [Left](#) collision.

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

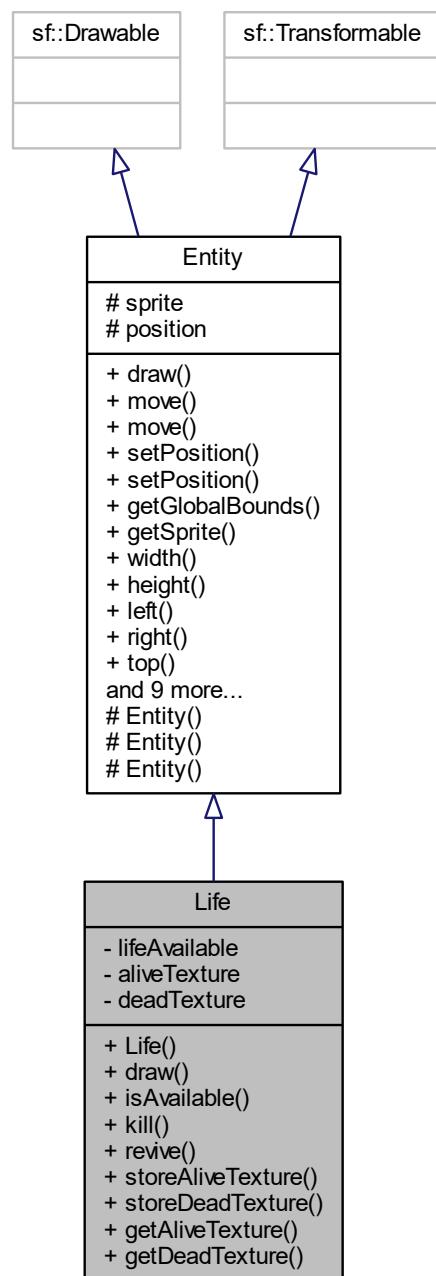
- [Camera.h](#)

## 10.28 Life Class Reference

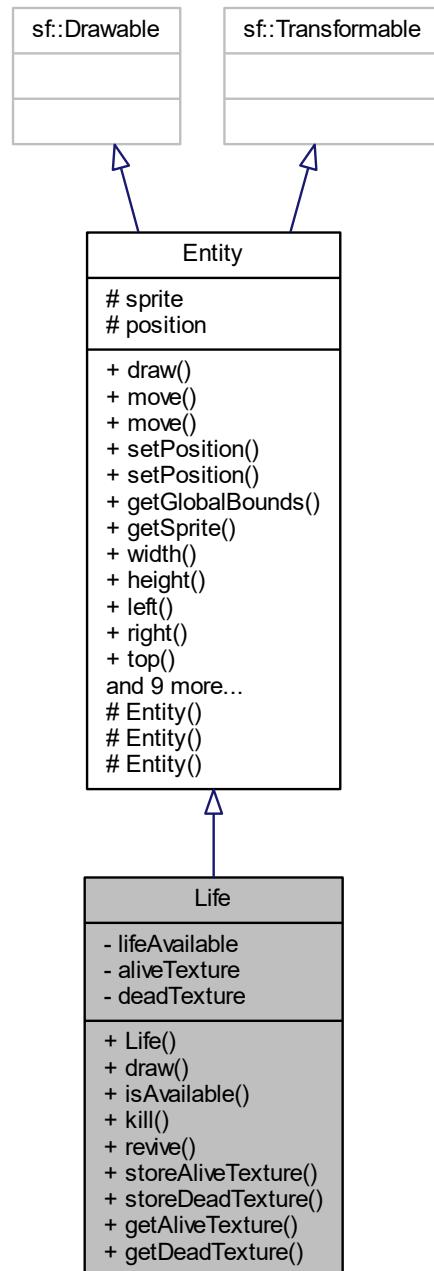
Used to represent the player life overlay **item** as a hearts in corner of the screen(player lives).

```
#include <Life.h>
```

Inheritance diagram for Life:



Collaboration diagram for Life:



## Public Member Functions

- `Life (sf::Vector2f position={0, 0})`  
*Constructs a `Life`.*
- void `draw (sf::RenderTarget &target, sf::RenderStates states) const override`  
*Draw method used to render `sprite` onto the window.*
- bool `isAvailable () const`

- `Is this player Life available?`
- `void kill ()`  
*Removes the life.*
- `void revive ()`  
*Regain the life.*
- `void storeAliveTexture (sf::Texture &texture)`  
*Stores [aliveTexture](#) by binding a pointer to an already existing texture reference.*
- `void storeDeadTexture (sf::Texture &texture)`  
*Stores [deadTexture](#) by binding a pointer to an already existing texture reference.*
- `sf::Texture & getAliveTexture () const`
- `sf::Texture & getDeadTexture () const`

## Private Attributes

- `bool lifeAvailable = true`  
*Is this [Player](#) life available?*
- `sf::Texture * aliveTexture {}`  
*Simply binds to an already existing [aliveTexture](#).*
- `sf::Texture * deadTexture {}`  
*Simply binds to an already existing [deadTexture](#).*

## Additional Inherited Members

### 10.28.1 Detailed Description

Used to represent the player life overlay [item](#) as a hearts in corner of the screen(player lives).

#### Note

Do not confuse an Overlay heart with [HeartCollectible](#) (game object).

A concrete [Entity](#) class overriding the draw method.

#### Note

[aliveTexture](#) is displayed when [Life](#) is available (*red heart*), while [deadTexture](#) is displayed when life is gone (*red-gray heart*).

### 10.28.2 Constructor & Destructor Documentation

#### 10.28.2.1 [Life\(\)](#)

```
Life::Life (
    sf::Vector2f position = {0, 0} ) [explicit]
```

Constructs a [Life](#).

**Parameters**

|                 |                                      |
|-----------------|--------------------------------------|
| <i>position</i> | initial position, defaulted to zero. |
|-----------------|--------------------------------------|

**10.28.3 Member Function Documentation****10.28.3.1 draw()**

```
void Life::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render [sprite](#) onto the window.

**Parameters**

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides [Entity](#) draw method.

**Note**

Passing states is optional.

**10.28.3.2 getAliveTexture()**

```
sf::Texture& Life::getAliveTexture () const [inline]
```

Retrieves the reference to an [aliveTexture](#)# by dereferencing it.

**Returns**

`sf::Texture&` reference to `aliveTexture`

**10.28.3.3 getDeadTexture()**

```
sf::Texture& Life::getDeadTexture () const [inline]
```

Retrieves the reference to a [deadTexture](#) by dereferencing it.

**Returns**

`sf::Texture&` reference to `deadTexture`

#### 10.28.3.4 isAvailable()

```
bool Life::isAvailable () const [inline]
```

Is this player [Life](#) available?

Returns

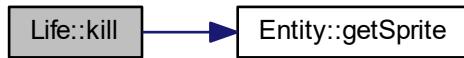
Is this player life available?

#### 10.28.3.5 kill()

```
void Life::kill () [inline]
```

Removes the life.

[lifeAvailable](#) is set to false and [sprite](#) is set to [deadTexture](#) Here is the call graph for this function:

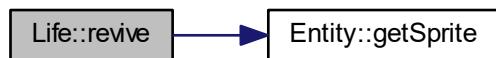


#### 10.28.3.6 revive()

```
void Life::revive () [inline]
```

Regain the life.

[lifeAvailable](#) is set to true and [sprite](#) is set to [aliveTexture](#) Here is the call graph for this function:



#### 10.28.3.7 storeAliveTexture()

```
void Life::storeAliveTexture (
    sf::Texture & texture ) [inline]
```

Stores [aliveTexture](#) by binding a pointer to an already existing texture reference.

**Parameters**

|                |   |
|----------------|---|
| <i>texture</i> | aliveTexture is bound to this parameter |
|----------------|---|

**10.28.3.8 storeDeadTexture()**

```
void Life::storeDeadTexture (
    sf::Texture & texture ) [inline]
```

Stores [deadTexture](#) by binding a pointer to an already existing texture reference.

**Parameters**

|                |  |
|----------------|--|
| <i>texture</i> | deadTexture is bound to this parameter |
|----------------|--|

**10.28.4 Member Data Documentation****10.28.4.1 aliveTexture**

```
sf::Texture* Life::aliveTexture {} [private]
```

Simply binds to an already existing aliveTexture.

**10.28.4.2 deadTexture**

```
sf::Texture* Life::deadTexture {} [private]
```

Simply binds to an already existing deadTexture.

**10.28.4.3 lifeAvailable**

```
bool Life::lifeAvailable = true [private]
```

Is this [Player](#) life available?

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

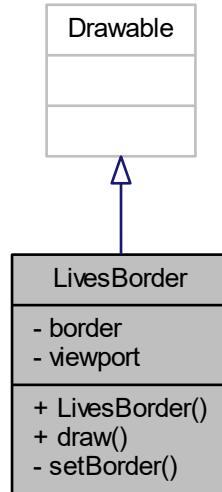
- [Life.h](#)
- [Life.cpp](#)

## 10.29 LivesBorder Class Reference

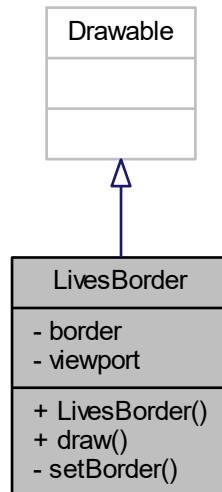
A border around the [LivesOverlay](#).

```
#include <LivesBorder.h>
```

Inheritance diagram for LivesBorder:



Collaboration diagram for LivesBorder:



## Public Member Functions

- [LivesBorder](#) (const sf::Rect< float > &[viewport](#))  
*Constructs a [LivesBorder](#).*
- void [draw](#) (sf::RenderTarget &[target](#), sf::RenderStates [states](#)) const override  
*Draw method used to render [border](#) onto the window.*

## Private Member Functions

- void [setBorder](#) ()

## Private Attributes

- sf::RectangleShape [border](#)  
*The border.*
- const sf::Rect< float > & [viewport](#)  
*The viewport which position [viewport](#) binds to.*

### 10.29.1 Detailed Description

A border around the [LivesOverlay](#).

### 10.29.2 Constructor & Destructor Documentation

#### 10.29.2.1 LivesBorder()

```
LivesBorder::LivesBorder (
    const sf::Rect< float > & viewport ) [explicit]
```

Constructs a [LivesBorder](#).

#### Parameters

|                          |                                     |
|--------------------------|-------------------------------------|
| <a href="#">viewport</a> | viewport position will be bound to. |
|--------------------------|-------------------------------------|

### 10.29.3 Member Function Documentation

#### 10.29.3.1 draw()

```
void LivesBorder::draw (
    sf::RenderTarget & target,
```

```
sf::RenderStates states ) const [override]
```

Draw method used to render [border](#) onto the window.

#### Parameters

|                     |  |
|---------------------|--|
| <code>target</code> | <a href="#">Window</a> for the <a href="#">border</a> to be rendered onto. |
| <code>states</code> | Optional render states.  |

Overrides [Entity](#) draw method.

#### Note

Passing states is optional.

### 10.29.3.2 setBorder()

```
void LivesBorder::setBorder ( ) [private]
```

## 10.29.4 Member Data Documentation

### 10.29.4.1 border

```
sf::RectangleShape LivesBorder::border [private]
```

The border.

### 10.29.4.2 viewport

```
const sf::Rect<float>& LivesBorder::viewport [private]
```

The viewport which position `viewport` binds to.

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

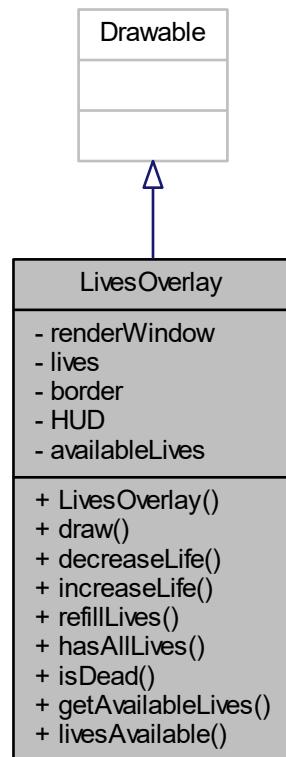
- [LivesBorder.h](#)
- [LivesBorder.cpp](#)

## 10.30 LivesOverlay Class Reference

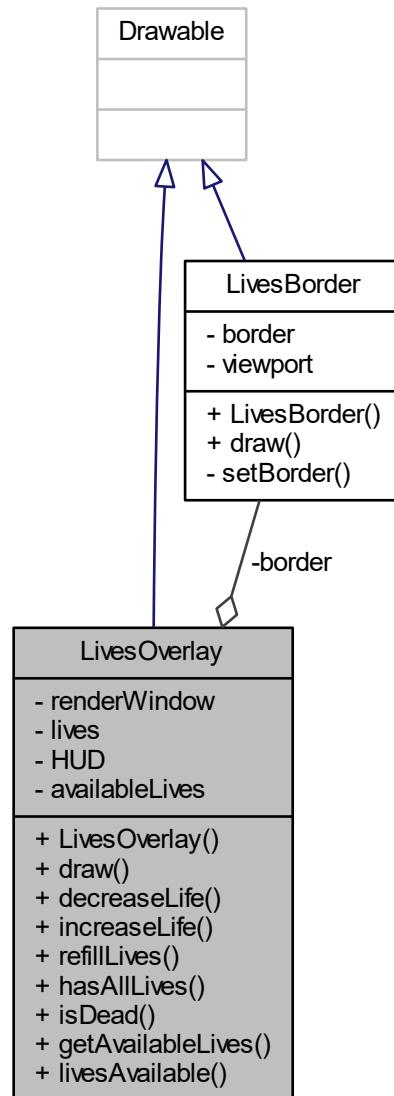
Encapsulates player lives overlay.

```
#include <LivesOverlay.h>
```

Inheritance diagram for LivesOverlay:



Collaboration diagram for LivesOverlay:



## Public Member Functions

- **LivesOverlay** (sf::RenderWindow &`renderWindow`, ResourceHolder< res::Texture, sf::Texture > &`textures`)  
*Constructs a LivesOverlay.*
- void **draw** (sf::RenderTarget &`target`, sf::RenderStates `states`) const override  
*Draw method used to render #fps onto the window.*
- void **decreaseLife** ()  
*Decrease player life by one.*
- void **increaseLife** ()  
*Increase player life by one.*
- void **refillLives** ()

- Refill all player lives.
- bool `hasAllLives () const`  
Checks if `Player` has all lives remaining.
- bool `isDead () const`  
Checks if `Player` has no lives left and should be dead.
- int `getAvailableLives () const`  
Gets the number of available lives remaining.
- bool `livesAvailable () const`  
Checks if `Player` has any lives available.

## Private Attributes

- sf::RenderWindow & `renderWindow`  
Handles the view.
- std::array< `Life`, `lives::count` > `lives`  
Static array of `Player` lives.
- `LivesBorder border`  
Border wrapped around `Player` lives.
- sf::View `HUD`  
Needed for setting the view by `renderWindow`.
- int `availableLives = lives::count`  
Remaining lives.

### 10.30.1 Detailed Description

Encapsulates player lives overlay.

#### Note

This class is relevant only in [StateGame](#).

### 10.30.2 Constructor & Destructor Documentation

#### 10.30.2.1 LivesOverlay()

```
LivesOverlay::LivesOverlay (
    sf::RenderWindow & renderWindow,
    ResourceHolder< res::Texture, sf::Texture > & textures ) [explicit]
```

Constructs a [LivesOverlay](#).

#### Parameters

|                           |  |
|---------------------------|--|
| <code>renderWindow</code> | for settings the view  |
| <code>textures</code>     | reference to container with Heart and HeartEmpty textures (among others) |

### 10.30.3 Member Function Documentation

#### 10.30.3.1 decreaseLife()

```
void LivesOverlay::decreaseLife ( ) [inline]
```

Decrease player life by one.

#### 10.30.3.2 draw()

```
void LivesOverlay::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render #fps onto the window.

##### Parameters

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the #fps to be rendered onto. |
| <i>states</i> | Optional render states                                   |

Overrides [Entity](#) draw method.

##### Note

Passing states is optional.

#### 10.30.3.3 getAvailableLives()

```
int LivesOverlay::getAvailableLives ( ) const [inline]
```

Gets the number of available lives remaining.

##### Returns

A number of player available lives remaining.

#### 10.30.3.4 hasAllLives()

```
bool LivesOverlay::hasAllLives () const [inline]
```

Checks if [Player](#) has all lives remaining.

Returns

does [Player](#) have all lives?

Here is the caller graph for this function:



#### 10.30.3.5 increaseLife()

```
void LivesOverlay::increaseLife () [inline]
```

Increase player life by one.

Here is the caller graph for this function:



#### 10.30.3.6 isDead()

```
bool LivesOverlay::isDead () const [inline]
```

Checks if [Player](#) has no lives left and should be dead.

Note

[Player](#) should be dead if [availableLives](#) are non-positive.

Returns

Is [Player](#) dead?

### 10.30.3.7 livesAvailable()

```
bool LivesOverlay::livesAvailable () const [inline]
```

Checks if [Player](#) has any lives available.

#### Returns

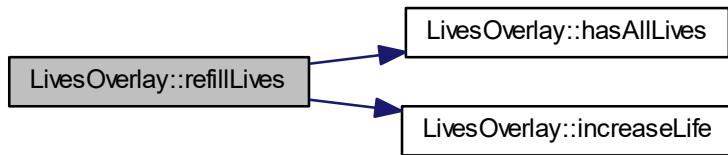
Does the [Player](#) have any lives available?

### 10.30.3.8 refillLives()

```
void LivesOverlay::refillLives () [inline]
```

Refill all player lives.

Here is the call graph for this function:



## 10.30.4 Member Data Documentation

### 10.30.4.1 availableLives

```
int LivesOverlay::availableLives = lives::count [private]
```

Remaining lives.

### 10.30.4.2 border

```
LivesBorder LivesOverlay::border [private]
```

Border wrapped around [Player](#) lives.

#### 10.30.4.3 HUD

```
sf::View LivesOverlay::HUD [private]
```

Needed for setting the view by [renderWindow](#).

#### 10.30.4.4 lives

```
std::array<Life, lives::count> LivesOverlay::lives [private]
```

Static array of [Player](#) lives.

#### 10.30.4.5 renderWindow

```
sf::RenderWindow& LivesOverlay::renderWindow [private]
```

Handles the view.

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

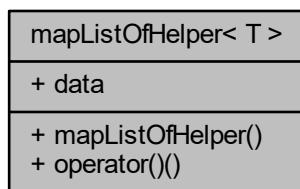
- [LivesOverlay.h](#)
- [LivesOverlay.cpp](#)

## 10.31 mapListOfHelper< T > Struct Template Reference

A helper for mapListOf converter.

```
#include <Utility.h>
```

Collaboration diagram for mapListOfHelper< T >:



## Public Member Functions

- `constexpr mapListOfHelper (T &d)`
- `constexpr mapListOfHelper & operator() (typename T::key_type const &key, typename T::mapped_type const &value)`

## Public Attributes

- `T & data`

### 10.31.1 Detailed Description

```
template<typename T>
struct mapListOfHelper< T >
```

A helper for mapListOf converter.

### 10.31.2 Constructor & Destructor Documentation

#### 10.31.2.1 mapListOfHelper()

```
template<typename T >
constexpr mapListOfHelper< T >::mapListOfHelper (
    T & d ) [inline], [explicit], [constexpr]
```

### 10.31.3 Member Function Documentation

#### 10.31.3.1 operator()()

```
template<typename T >
constexpr mapListOfHelper& mapListOfHelper< T >::operator() (
    typename T::key_type const & key,
    typename T::mapped_type const & value ) [inline], [constexpr]
```

### 10.31.4 Member Data Documentation

#### 10.31.4.1 data

```
template<typename T >  
T& mapListOfHelper< T >::data
```

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

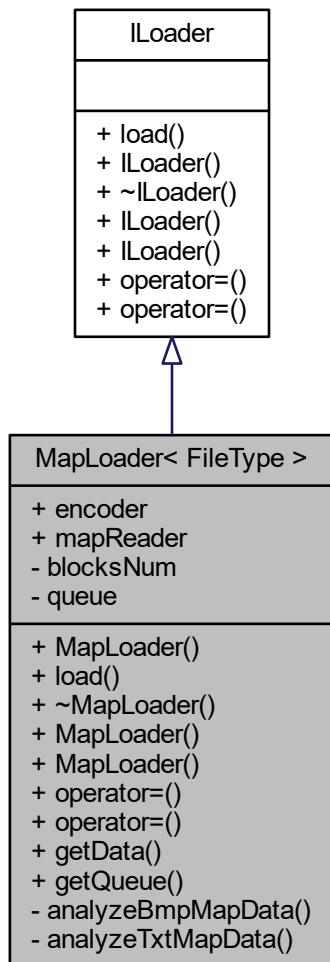
- [Utility.h](#)

## 10.32 MapLoader< FileType > Class Template Reference

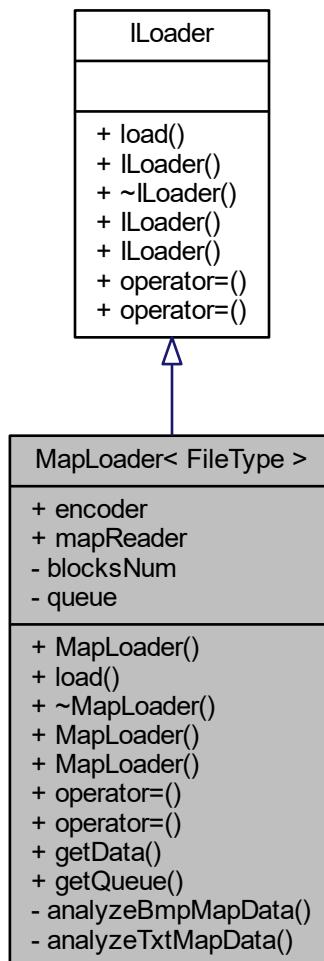
Load the map file.

```
#include <MapLoader.h>
```

Inheritance diagram for MapLoader< FileType >:



Collaboration diagram for MapLoader< FileType >:



## Public Member Functions

- `MapLoader (std::string fileName)`  
*Constructs a map loader of an appropriate either `Txt` or `Bmp` type.*
- void `load ()` override  
*Loads the map data into the `queue`.*
- virtual `~MapLoader ()`=default
- `MapLoader (const MapLoader &)=default`
- `MapLoader (MapLoader &&) noexcept=default`
- `MapLoader & operator= (const MapLoader &)=default`
- `MapLoader & operator= (MapLoader &&) noexcept=default`
- auto & `getData () const`  
*Retrieve the data.*
- auto & `getQueue ()`  
*Retrieve data queue.*

## Public Attributes

- std::variant< std::monostate, Encoder< PixelColor >, Encoder< int > > encoder  
*Encoder variant.*
- std::variant< std::monostate, BmpReader, TxtReader > mapReader  
*MapReader variant.*

## Private Member Functions

- void analyzeBmpMapData ()  
*Counts the number of collidable objects from a **bmp** map data. Inserts them into FIFO queue container.*
- void analyzeTxtMapData ()  
*Counts the number of collidable objects from a **txt** map data. Inserts them into FIFO queue container.*

## Private Attributes

- int blocksNum {}  
*Store the number of collidable blocks.*
- std::queue< res::Texture > queue  
*The queue of textures, FIFO needed when constructing the map to preserve the order.*

### 10.32.1 Detailed Description

```
template<typename FileType>
class MapLoader< FileType >
```

Load the map file.

Most operations determined at compile-time.

A file can be either .bmp or .txt.

This is a class template based on a file type.

### 10.32.2 Constructor & Destructor Documentation

### 10.32.2.1 MapLoader() [1/3]

```
template<typename FileType >
MapLoader< FileType >::MapLoader (
    std::string fileName ) [inline], [explicit]
```

Constructs a map loader of an appropriate either [Txt](#) or [Bmp](#) type.

First analyzes the type of FileType, then loads the map.

#### Note

Uses C++17 'if constexpr' and std::is\_same to determine the type at compile-time.

Here is the call graph for this function:



### 10.32.2.2 ~MapLoader()

```
template<typename FileType >
virtual MapLoader< FileType >::~MapLoader ( ) [virtual], [default]
```

\brief Enable default destructor.

### 10.32.2.3 MapLoader() [2/3]

```
template<typename FileType >
MapLoader< FileType >::MapLoader (
    const MapLoader< FileType > & ) [default]
```

\brief Enable default copy operations.

### 10.32.2.4 MapLoader() [3/3]

```
template<typename FileType >
MapLoader< FileType >::MapLoader (
    MapLoader< FileType > && ) [default], [noexcept]
```

\brief Enable default move operations.

### 10.32.3 Member Function Documentation

#### 10.32.3.1 analyzeBmpMapData()

```
template<typename FileType >
void MapLoader< FileType >::analyzeBmpMapData ( ) [inline], [private]
```

Counts the number of collidable objects from a **bmp** map data. Inserts them into FIFO queue container.

May see improvements in the future.

#### 10.32.3.2 analyzeTxtMapData()

```
template<typename FileType >
void MapLoader< FileType >::analyzeTxtMapData ( ) [inline], [private]
```

Counts the number of collidable objects from a **txt** map data. Inserts them into FIFO queue container.

May see improvements in the future.

#### 10.32.3.3 getData()

```
template<typename FileType >
auto& MapLoader< FileType >::getData ( ) const [inline]
```

Retrieve the data.

Currently not used, may be re-used in the future.

#### 10.32.3.4 getQueue()

```
template<typename FileType >
auto& MapLoader< FileType >::getQueue ( ) [inline]
```

Retrieve data queue.

##### Returns

`std::queue<res::Texture>` queue of Textures

### 10.32.3.5 load()

```
template<typename FileType >
void MapLoader< FileType >::load ( ) [inline], [override], [virtual]
```

Loads the map data into the [queue](#).

Analyzing the data effectively means counting the Collidable block game entities.

#### Note

Uses std::visit with overload hack.

Implements [ILoader](#).

### 10.32.3.6 operator=( ) [1/2]

```
template<typename FileType >
MapLoader& MapLoader< FileType >::operator= (
    const MapLoader< FileType > & ) [default]
```

\brief Enable default copy operations.

### 10.32.3.7 operator=( ) [2/2]

```
template<typename FileType >
MapLoader& MapLoader< FileType >::operator= (
    MapLoader< FileType > && ) [default], [noexcept]
```

\brief Enable default move operations.

## 10.32.4 Member Data Documentation

### 10.32.4.1 blocksNum

```
template<typename FileType >
int MapLoader< FileType >::blocksNum {} [private]
```

Store the number of collidable blocks.

#### 10.32.4.2 encoder

```
template<typename FileType >
std::variant<std::monostate, Encoder<PixelColor>, Encoder<int> > MapLoader< FileType >::encoder
```

Encoder variant.

#### 10.32.4.3 mapReader

```
template<typename FileType >
std::variant<std::monostate, BmpReader, TxtReader> MapLoader< FileType >::mapReader
```

MapReader variant.

#### 10.32.4.4 queue

```
template<typename FileType >
std::queue<res::Texture> MapLoader< FileType >::queue [private]
```

The queue of textures, FIFO needed when constructing the map to preserve the order.

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

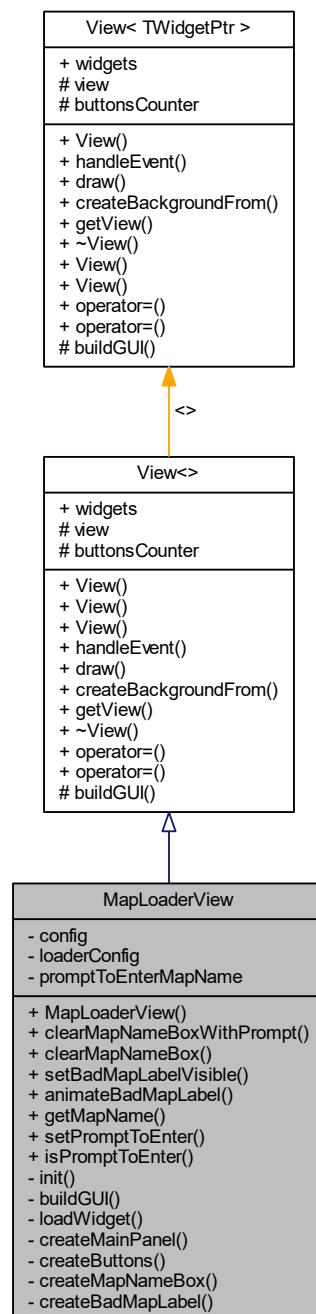
- [MapLoader.h](#)

### 10.33 MapLoaderView Class Reference

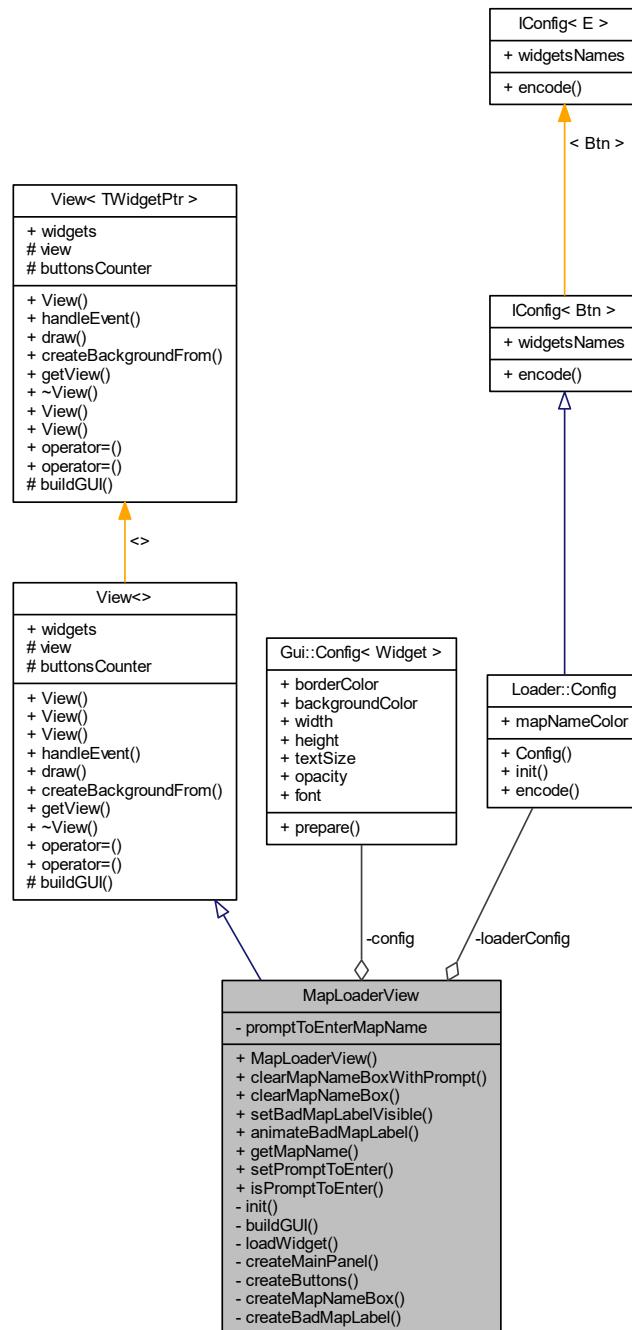
[View](#) class used to draw gui within [StateMapLoader](#).

```
#include <MapLoaderView.h>
```

Inheritance diagram for MapLoaderView:



Collaboration diagram for MapLoaderView:



## Public Member Functions

- `MapLoaderView (Window &window)`  
*Constructs the MapLoaderView.*
- `void clearMapNameBoxWithPrompt ()`
- `void clearMapNameBox ()`
- `void setBadMapLabelVisible (bool visibilityStatus)`

- void [animateBadMapLabel \(\)](#)
- std::string [getMapName \(\) const](#)
- void [setPromptToEnter \(bool status\)](#)
- bool [isPromptToEnter \(\) const](#)

## Private Member Functions

- void [init \(\)](#)  
*Initializes the [MapLoaderView](#). Effectively: calls [buildGUI\(\)](#).*
- void [buildGUI \(\) override](#)
- void [loadWidget \(tgui::Widget::Ptr &widget\)](#)  
*Loads a given widget (likely: a button) onto the correct settings and position.*
- void [createMainPanel \(\)](#)  
*Creates the main panel and adds it to the #gui.*
- void [createButtons \(\)](#)  
*Creates all buttons and adds them to the #gui.*
- void [createMapNameBox \(\)](#)
- void [createBadMapLabel \(\)](#)

## Private Attributes

- Gui::Config [config](#)  
*Configuration settings common to all [View](#) classes.*
- Loader::Config [loaderConfig](#)  
*Configuration settings specific to [MapLoaderView](#).*
- bool [promptToEnterMapName {}](#)  
*Is currently prompt to re-enter map name?*

## Additional Inherited Members

### 10.33.1 Detailed Description

[View](#) class used to draw gui within [StateMapLoader](#).

### 10.33.2 Constructor & Destructor Documentation

#### 10.33.2.1 MapLoaderView()

```
MapLoaderView::MapLoaderView (
    Window & window ) [explicit]
```

Constructs the [MapLoaderView](#).

**Parameters**

|                        |   |
|------------------------|---|
| <a href="#">Window</a> | that is bound to <a href="#">View</a> base. |
|------------------------|---|

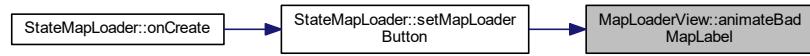
**10.33.3 Member Function Documentation****10.33.3.1 animateBadMapLabel()**

```
void MapLoaderView::animateBadMapLabel( )
```

Slide animation on show and hide of >No such file!< label. Here is the call graph for this function:



Here is the caller graph for this function:

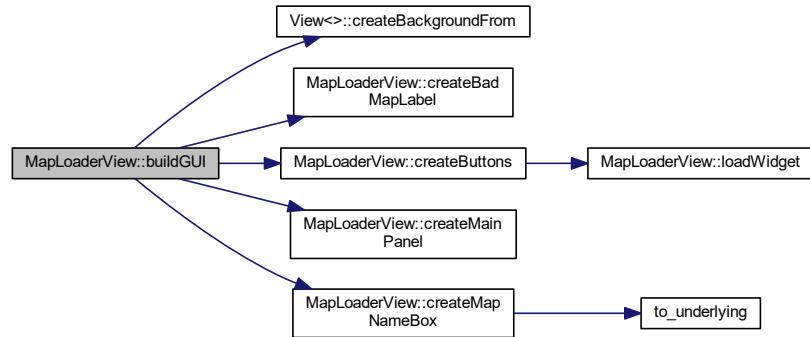
**10.33.3.2 buildGUI()**

```
void MapLoaderView::buildGUI( ) [override], [private], [virtual]
```

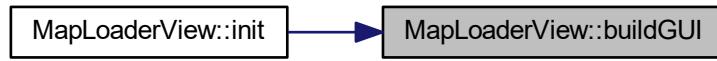
Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements [View<>](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.33.3.3 clearMapNameBox()

```
void MapLoaderView::clearMapNameBox( )
```

Clears the map name edit box. Here is the call graph for this function:



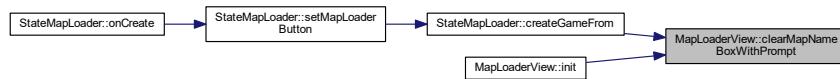
#### 10.33.3.4 clearMapNameBoxWithPrompt()

```
void MapLoaderView::clearMapNameBoxWithPrompt( )
```

Clears the map name edit box with >Enter map name...< message Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.33.3.5 createBadMapLabel()

```
void MapLoaderView::createBadMapLabel( ) [private]
```

Create a label showing >No such file!< when user enters bad map name, and add it to the #gui. Here is the caller graph for this function:



### 10.33.3.6 createButtons()

```
void MapLoaderView::createButtons ( ) [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.33.3.7 createMainPanel()

```
void MapLoaderView::createMainPanel ( ) [private]
```

Creates the main panel and adds it to the #gui.

Here is the caller graph for this function:



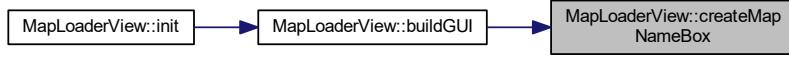
### 10.33.3.8 `createMapNameBox()`

```
void MapLoaderView::createMapNameBox ( ) [private]
```

Create an edit box where user types in his map name to the #gui. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.33.3.9 `getMapName()`

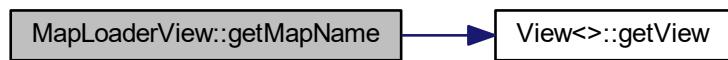
```
std::string MapLoaderView::getMapName ( ) const
```

Retrieve the map name.

#### Returns

the map name

Here is the call graph for this function:



Here is the caller graph for this function:

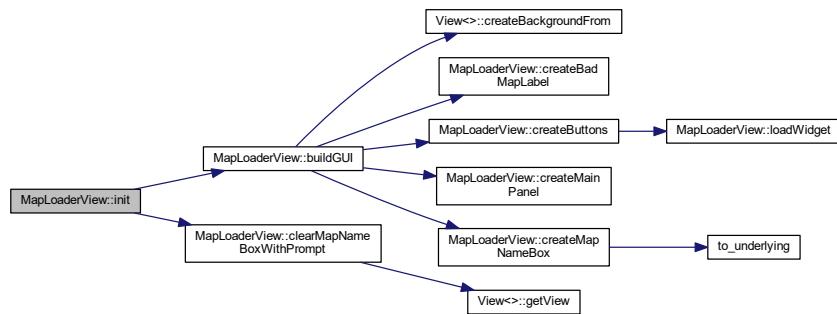


### 10.33.3.10 init()

```
void MapLoaderView::init () [private]
```

Initializes the [MapLoaderView](#). Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



### 10.33.3.11 isPromptToEnter()

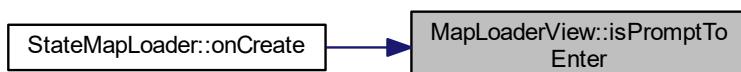
```
bool MapLoaderView::isPromptToEnter () const
```

Check if player is currently [promptToEnterMapName](#).

Returns

Is bad map label prompt visible?

Here is the caller graph for this function:



### 10.33.3.12 loadWidget()

```
void MapLoaderView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position.

#### Parameters

|               |               |
|---------------|---------------|
| <i>widget</i> | to be loaded. |
|---------------|---------------|

#### Note

Can be used in a loop.

< so there is a wide gap between buttonsHere is the caller graph for this function:



### 10.33.3.13 setBadMapLabelVisible()

```
void MapLoaderView::setBadMapLabelVisible (
    bool visibilityStatus )
```

Sets the badMapLabel visibility to a given status.

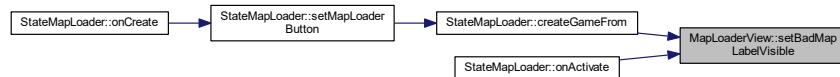
#### Parameters

|                         |  |
|-------------------------|--|
| <i>visibilityStatus</i> |  |
|-------------------------|--|

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.33.3.14 `setPromptToEnter()`

```
void MapLoaderView::setPromptToEnter (
    bool status )
```

Sets `promptToEnterMapName` to a given status.

##### Parameters

|                     |             |
|---------------------|-------------|
| <code>status</code> | new status. |
|---------------------|-------------|

### 10.33.4 Member Data Documentation

#### 10.33.4.1 `config`

```
Gui::Config MapLoaderView::config [private]
```

Configuration settings common to all `View` classes.

#### 10.33.4.2 `loaderConfig`

```
Loader::Config MapLoaderView::loaderConfig [private]
```

Configuration settings specific to `MapLoaderView`.

#### 10.33.4.3 `promptToEnterMapName`

```
bool MapLoaderView::promptToEnterMapName {} [private]
```

Is currently prompt to re-enter map name?

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

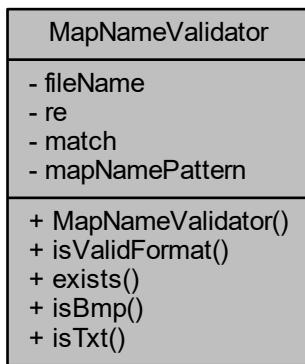
- [MapLoaderView.h](#)
- [MapLoaderView.cpp](#)

## 10.34 MapNameValidator Class Reference

A utility class validating if a given map file name is valid.

```
#include <MapNameValidator.h>
```

Collaboration diagram for MapNameValidator:



### Public Member Functions

- **MapNameValidator (std::string\_view fileName)**  
*Constructs a [MapNameValidator](#) of a given `fileName`.*
- **bool isValidFormat ()**  
*checks if given `fileName` extension is a valid format.*
- **bool exists () const**  
*Checks if a given `fileName` exists.*
- **bool isBmp () const**  
*Check if a `fileName` is a .bmp file. Called only if `isValidFormat()` was true.*
- **bool isTxt () const**  
*Check if a `fileName` is a .txt file. Called only if `isValidFormat()` was true.*

### Private Attributes

- **const std::string\_view fileName**  
*Map name.*
- **std::regex re {}**  
*Run-time FSA compiled (!) even if pattern is known at compile-time.*
- **std::smatch\_results< std::string\_view::const\_iterator > match {}**  
*Stored match results.*

## Static Private Attributes

- `constexpr static std::string_view mapNamePattern = R"(.+?\.(bmp|txt))"`  
*Compile-time basic file matcher pattern.*

### 10.34.1 Detailed Description

A utility class validating if a given map file name is valid.

#### Note

A file is valid if:

- it exists
- is accessible
- has correct file extension

Available file extensions:

- .bmp
- .txt

Uses regex to match file name against the pattern.

### 10.34.2 Constructor & Destructor Documentation

#### 10.34.2.1 MapNameValidator()

```
MapNameValidator::MapNameValidator (
    std::string_view fileName ) [explicit]
```

Constructs a [MapNameValidator](#) of a given fileName.

#### Parameters

|                       |                                |
|-----------------------|--------------------------------|
| <code>fileName</code> | File name of a given game map. |
|-----------------------|--------------------------------|

Throws if a pattern is incorrect, however, the pattern is known at compile time.

#### Note

C++ standard library regexes are compiled to FSA at run-time, even if it is already year 2020 as of *today*.

### 10.34.3 Member Function Documentation

### 10.34.3.1 exists()

```
bool MapNameValidator::exists ( ) const
```

Checks if a given `fileName` exists.

#### Returns

Does the file exist?

### 10.34.3.2 isBmp()

```
bool MapNameValidator::isBmp ( ) const
```

Check if a `fileName` is a .bmp file. Called only if `isValidFormat()` was true.

#### Note

: File extension is stored as a first subexpression (index 1) inside `match`.

#### Returns

Is `fileName` a .bmp file?

### 10.34.3.3 isTxt()

```
bool MapNameValidator::isTxt ( ) const
```

Check if a `fileName` is a .txt file. Called only if `isValidFormat()` was true.

#### Note

: File extension is stored as a first subexpression (index 1) inside `match`.

#### Returns

Is `fileName` a .txt file?

#### 10.34.3.4 isValidFormat()

```
bool MapNameValidator::isValidFormat( )
```

hecks if given `fileName` extension is a valid format.

CCurrently supported game map files:

- .bmp
- .txt

Matches the `fileName` against regular expression `re` generated from `mapNamePattern`.

##### Returns

Is `fileName` a valid map format (.bmp, .txt)?

Here is the caller graph for this function:



#### 10.34.4 Member Data Documentation

##### 10.34.4.1 fileName

```
const std::string_view MapNameValidator::fileName [private]
```

Map name.

##### 10.34.4.2 mapNamePattern

```
constexpr static std::string_view MapNameValidator::mapNamePattern = R"(.+?\.(bmp|txt))" [static],  
[constexpr], [private]
```

Compile-time basic file matcher pattern.

#### 10.34.4.3 match

```
std::match_results<std::string_view::const_iterator> MapNameValidator::match {} [private]
```

Stored match results.

#### 10.34.4.4 re

```
std::regex MapNameValidator::re {} [private]
```

Run-time FSA compiled (!) even if pattern is known at compile-time.

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

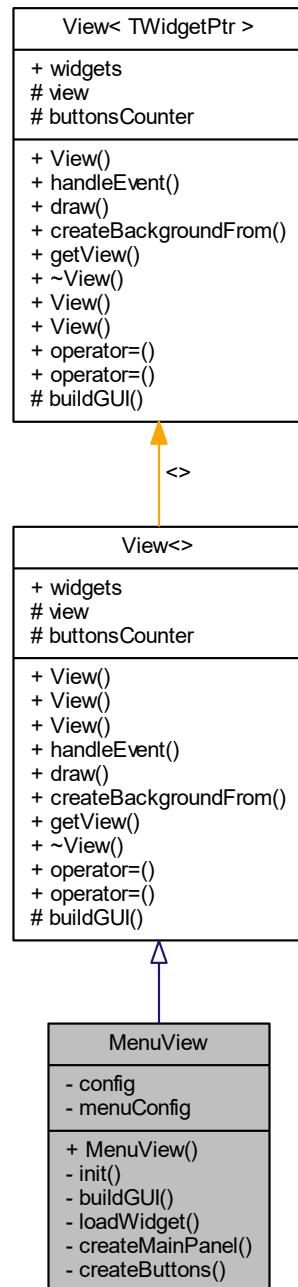
- [MapNameValidator.h](#)
- [MapNameValidator.cpp](#)

## 10.35 MenuView Class Reference

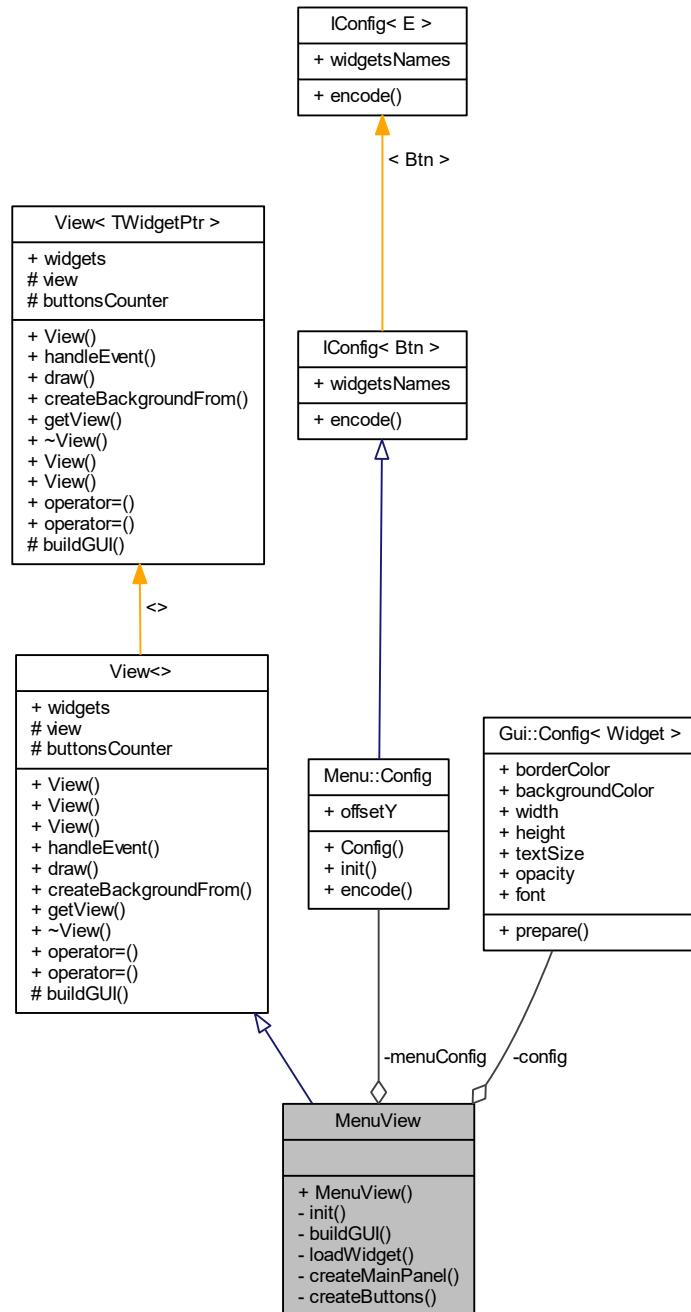
[View](#) class used to draw gui within [StateMenu](#).

```
#include <MenuView.h>
```

Inheritance diagram for MenuView:



Collaboration diagram for MenuView:



## Public Member Functions

- `MenuView (Window &window)`

*Constructs the `MenuView`.*

## Private Member Functions

- void `init ()`  
*Initializes the `MenuView`. Effectively: calls `buildGUI()`.*
- void `buildGUI () override`
- void `loadWidget (tgui::Widget::Ptr &widget)`  
*Loads a given widget (likely: a button) onto the correct settings and position.*
- void `createMainPanel ()`  
*Creates the main panel and adds it to the #gui.*
- void `createButtons ()`  
*Creates all buttons and adds them to the #gui.*

## Private Attributes

- `Gui::Config config`  
*Configuration settings common to all `View` classes.*
- `Menu::Config menuConfig`  
*Configuration settings specific to `MenuView`.*

## Additional Inherited Members

### 10.35.1 Detailed Description

`View` class used to draw gui within `StateMenu`.

### 10.35.2 Constructor & Destructor Documentation

#### 10.35.2.1 `MenuView()`

```
MenuView::MenuView (
    Window & window ) [explicit]
```

Constructs the `MenuView`.

#### Parameters

|                     |  |
|---------------------|--|
| <code>Window</code> | that is bound to <code>View</code> base. |
|---------------------|--|

### 10.35.3 Member Function Documentation

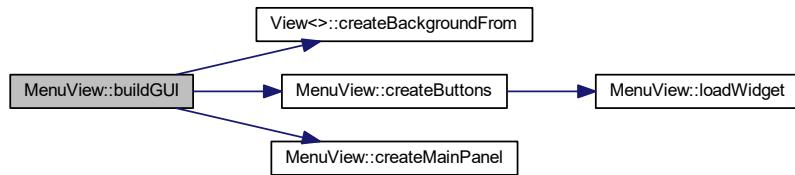
### 10.35.3.1 buildGUI()

```
void MenuView::buildGUI () [override], [private], [virtual]
```

Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements [View<>](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.35.3.2 createButtons()

```
void MenuView::createButtons () [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.35.3.3 createMainPanel()

```
void MenuView::createMainPanel ( ) [private]
```

Creates the main panel and adds it to the #gui.

Here is the caller graph for this function:

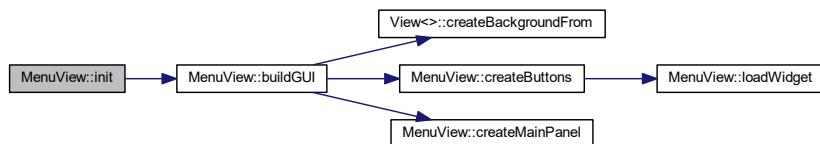


#### 10.35.3.4 init()

```
void MenuView::init ( ) [private]
```

Initializes the [MenuView](#). Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



#### 10.35.3.5 loadWidget()

```
void MenuView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position.

**Parameters**

|                     |               |
|---------------------|---------------|
| <code>widget</code> | to be loaded. |
|---------------------|---------------|

**Note**

Can be used in a loop.

Here is the caller graph for this function:



## 10.35.4 Member Data Documentation

### 10.35.4.1 config

```
Gui::Config MenuView::config [private]
```

Configuration settings common to all [View](#) classes.

### 10.35.4.2 menuConfig

```
Menu::Config MenuView::menuConfig [private]
```

Configuration settings specific to [MenuView](#).

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

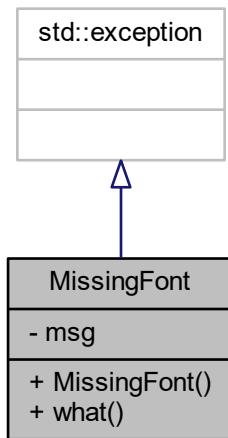
- [MenuView.h](#)
- [MenuView.cpp](#)

## 10.36 MissingFont Class Reference

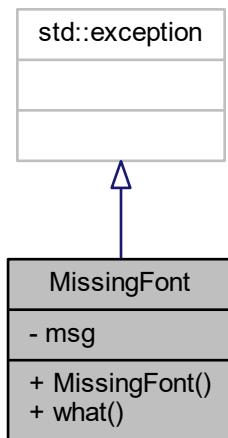
Custom exception class.

```
#include <FpsOverlay.h>
```

Inheritance diagram for MissingFont:



Collaboration diagram for MissingFont:



### Public Member Functions

- `MissingFont (std::string msg="Missing font")`
- `virtual const char * what () const noexcept override`

## Private Attributes

- std::string [msg](#)

### 10.36.1 Detailed Description

Custom exception class.

### 10.36.2 Constructor & Destructor Documentation

#### 10.36.2.1 MissingFont()

```
MissingFont::MissingFont (
    std::string msg = "Missing font" ) [inline]
```

Constructor with default parameter value.

### 10.36.3 Member Function Documentation

#### 10.36.3.1 what()

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

Overriden [what\(\)](#) method.

### 10.36.4 Member Data Documentation

#### 10.36.4.1 msg

```
std::string MissingFont::msg [private]
```

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

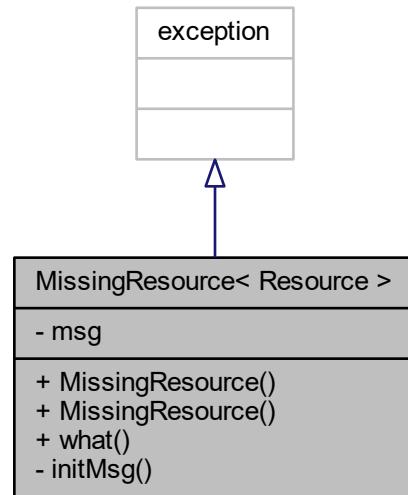
- [FpsOverlay.h](#)

## 10.37 MissingResource< Resource > Class Template Reference

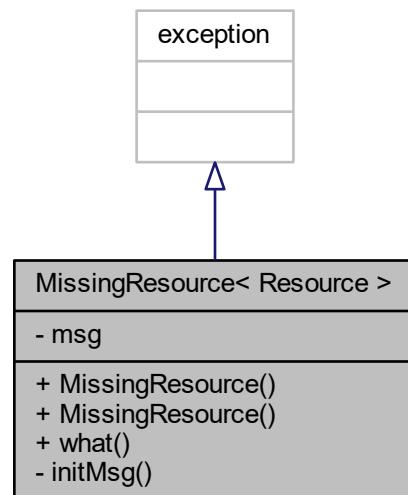
Meaningful response error when resource of given type was missing.

```
#include <ResourceHolder.h>
```

## Inheritance diagram for MissingResource< Resource >:



## Collaboration diagram for MissingResource< Resource >:



## Public Member Functions

- `MissingResource ()=default`  
*Enable default constructor.*
- `MissingResource (std::string_view fileName)`
- `const char * what () const noexcept override`  
*Overridden `what()` method used in catch block, displaying full error message.*

## Static Private Member Functions

- `static std::string initMsg (std::string_view fileName)`  
*Make a meaningful error-message using typeid of Resource.*

## Private Attributes

- `std::string msg {}`  
*Message to be returned.*

### 10.37.1 Detailed Description

```
template<typename Resource>
class MissingResource< Resource >
```

Meaningful response error when resource of given type was missing.

### 10.37.2 Constructor & Destructor Documentation

#### 10.37.2.1 MissingResource() [1/2]

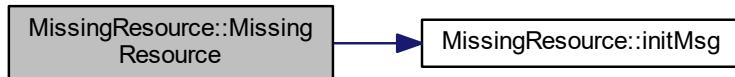
```
template<typename Resource >
MissingResource< Resource >::MissingResource ( ) [default]
```

Enable default constructor.

### 10.37.2.2 MissingResource() [2/2]

```
template<typename Resource >
MissingResource< Resource >::MissingResource (
    std::string_view fileName ) [inline]
```

brief Builds up meaningful error message based on fileName and Resource type. Here is the call graph for this function:



## 10.37.3 Member Function Documentation

### 10.37.3.1 initMsg()

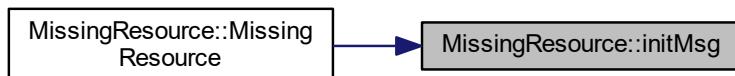
```
template<typename Resource >
static std::string MissingResource< Resource >::initMsg (
    std::string_view fileName ) [inline], [static], [private]
```

Make a meaningful error-message using typeid of Resource.

#### Note

The message: "Failed loading resource: " "{ Type: " + std::string(typeid(Resource).name()) + ", " "File name: " + fileName.data() + " }";

Here is the caller graph for this function:



### 10.37.3.2 what()

```
template<typename Resource >
const char* MissingResource< Resource >::what ( ) const [inline], [override], [noexcept]
Overriden what\(\) method used in catch block, displaying full error message.
```

## 10.37.4 Member Data Documentation

### 10.37.4.1 msg

```
template<typename Resource >
std::string MissingResource< Resource >::msg {} [private]
```

Message to be returned.

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

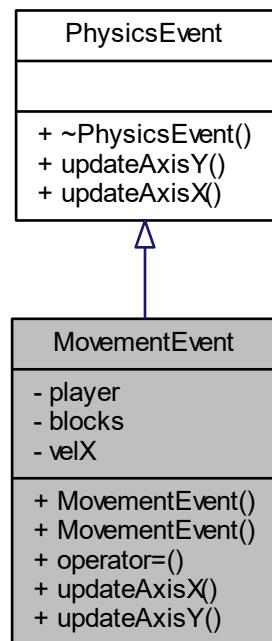
- [ResourceHolder.h](#)

## 10.38 MovementEvent Class Reference

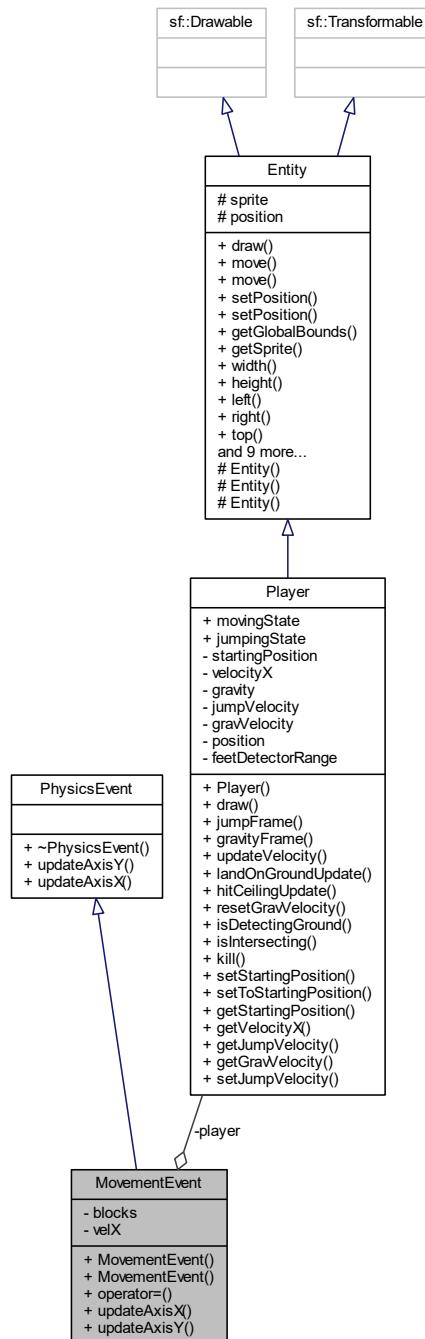
[MovementEvent](#) class responsible for updating the player movement in an **axis-independent way**.

```
#include <MovementEvent.h>
```

Inheritance diagram for MovementEvent:



Collaboration diagram for MovementEvent:



## Public Member Functions

- **MovementEvent (Player &player, std::vector< std::unique\_ptr< Entity >> &blocks)**  
*Create a MovementEvent instance.*
- **MovementEvent (const MovementEvent &)=delete**  
*Delete copy operations.*
- **MovementEvent & operator= (const MovementEvent &)=delete**

- Delete copy operations.*
- void `updateAxisX` (float dt) override  
*Handles horizontal movements on X axis based on MovingState.*
  - void `updateAxisY` (float dt) override  
*Handles all vertical movements on Y axis based on JumpingState.*

## Private Attributes

- `Player & player`
- `std::vector< std::unique_ptr< Entity > >> & blocks`
- const float `velX`  
*Binds during `MovementEvent` construction to minimize getters usage in performance-crucial loop.*

### 10.38.1 Detailed Description

`MovementEvent` class responsible for updating the player movement in an **axis-independent way**.

#### Note

The idea is that the both axes work independently in terms of `MovementEvent`, `CollisionEvent` and `Camera`. It is much easier to work with when working with one axis at a time.

#### Note

Implements pure virtual methods from base `PhysicsEvent` class.

### 10.38.2 Constructor & Destructor Documentation

#### 10.38.2.1 MovementEvent() [1/2]

```
MovementEvent::MovementEvent (
    Player & player,
    std::vector< std::unique_ptr< Entity > >> & blocks )
```

Create a `MovementEvent` instance.

#### Parameters

|                     |  |
|---------------------|--|
| <code>player</code> | - controller   |
| <code>blocks</code> | - entities for the controller to be matched <code>isDetectingGround</code> against |

### 10.38.2.2 MovementEvent() [2/2]

```
MovementEvent::MovementEvent ( 
    const MovementEvent & ) [delete]
```

Delete copy operations.

## 10.38.3 Member Function Documentation

### 10.38.3.1 operator=(\*)

```
MovementEvent& MovementEvent::operator= ( 
    const MovementEvent & ) [delete]
```

Delete copy operations.

### 10.38.3.2 updateAxisX()

```
void MovementEvent::updateAxisX ( 
    float dt ) [override], [virtual]
```

Handles horizontal movements on X axis based on MovingState.

Implements [PhysicsEvent](#) pure virtual method.

@Example

Implements [PhysicsEvent](#).

Here is the call graph for this function:



### 10.38.3.3 updateAxisY()

```
void MovementEvent::updateAxisY (
    float dt ) [override], [virtual]
```

Handles all vertical movements on Y axis based on JumpingState.

[Player](#) can be in one of three states:

- [JumpingStaet::onGround](#)
- [JumpingState::jumping](#)
- [JumpingState::gravity](#)

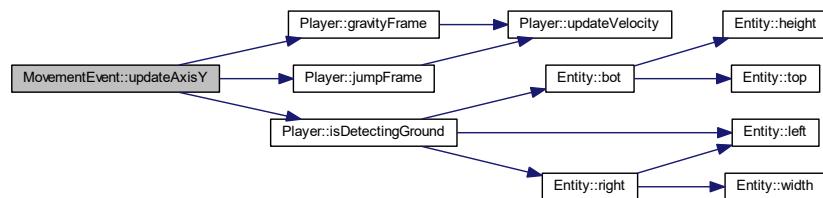
#### Note

If the [Player](#) is onGround, and he no longer detects the blocks beneath his feet, it means he fell off the edge. Therefore passive gravity should be turned on by changing the JumpingState to [JumpingState::gravity](#).

Implements [PhysicsEvent](#) pure virtual method.

Implements [PhysicsEvent](#).

Here is the call graph for this function:



### 10.38.4 Member Data Documentation

#### 10.38.4.1 blocks

```
std::vector<std::unique_ptr<Entity>> & MovementEvent::blocks [private]
```

#### 10.38.4.2 player

```
Player& MovementEvent::player [private]
```

### 10.38.4.3 velX

```
const float MovementEvent::velX [private]
```

Binds during [MovementEvent](#) construction to minimize getters usage in performance-crucial loop.

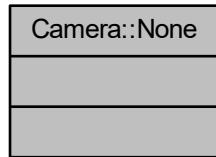
The documentation for this class was generated from the following files:

- [MovementEvent.h](#)
- [MovementEvent.cpp](#)

## 10.39 Camera::None Struct Reference

Empty struct denoting there was previously [None](#) collision.

Collaboration diagram for Camera::None:



### 10.39.1 Detailed Description

Empty struct denoting there was previously [None](#) collision.

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

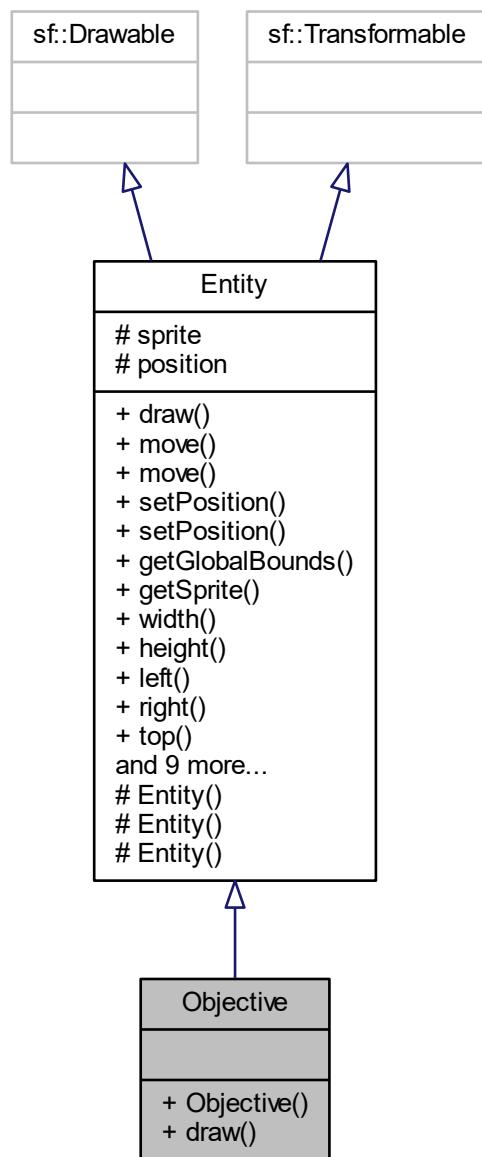
- [Camera.h](#)

## 10.40 Objective Class Reference

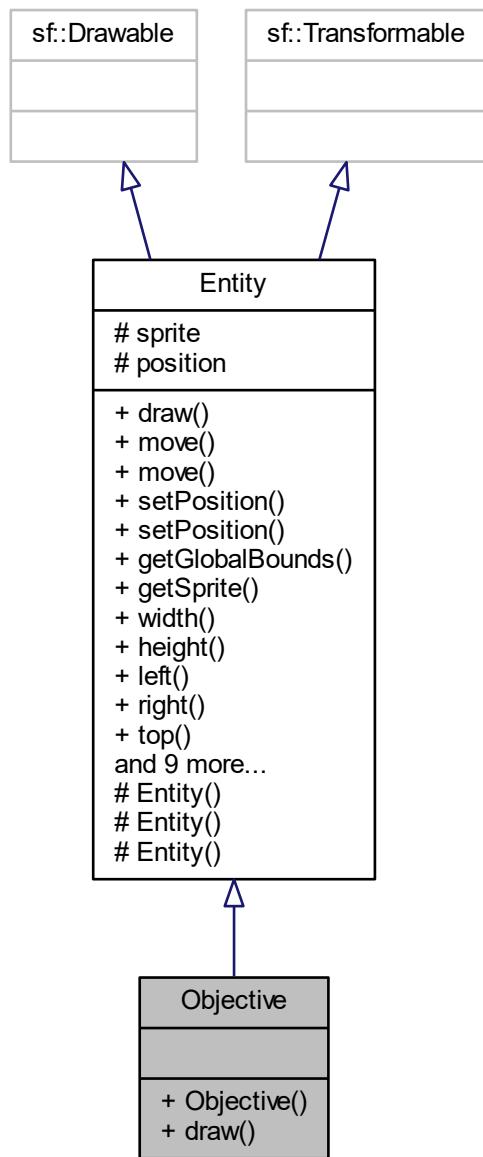
Used to represent the game [Objective](#). [Player](#) intersecting [Objective](#) wins the game.

```
#include <Objective.h>
```

Inheritance diagram for Objective:



Collaboration diagram for Objective:



## Public Member Functions

- `Objective (sf::Vector2f position={0, 0})`  
*Constructs an `Objective`.*
- `void draw (sf::RenderTarget &target, sf::RenderStates states) const override`  
*Draw method used to render `sprite` onto the window.*

## Additional Inherited Members

### 10.40.1 Detailed Description

Used to represent the game [Objective](#). Player intersecting [Objective](#) wins the game.

#### Note

Can the [Player](#) find the [Objective](#)?

A concrete [Entity](#) class overriding the draw method.

### 10.40.2 Constructor & Destructor Documentation

#### 10.40.2.1 Objective()

```
Objective::Objective (
    sf::Vector2f position = {0, 0} ) [explicit]
```

Constructs an [Objective](#).

#### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>position</i> | initial position, defaulted to zero. |
|-----------------|--------------------------------------|

### 10.40.3 Member Function Documentation

#### 10.40.3.1 draw()

```
void Objective::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render [sprite](#) onto the window.

#### Parameters

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides [Entity](#) draw method.

**Note**

Passing states is optional.

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

- [Objective.h](#)
- [Objective.cpp](#)

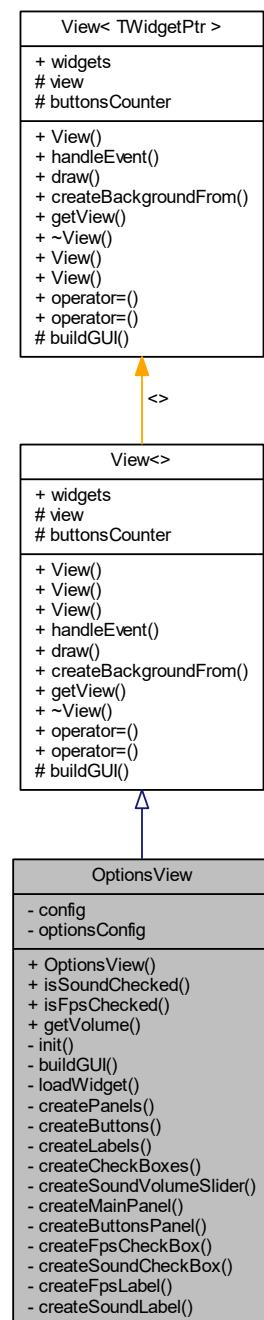
## 10.41 OptionsView Class Reference

[View](#) class used to draw gui within [StateOptions](#).

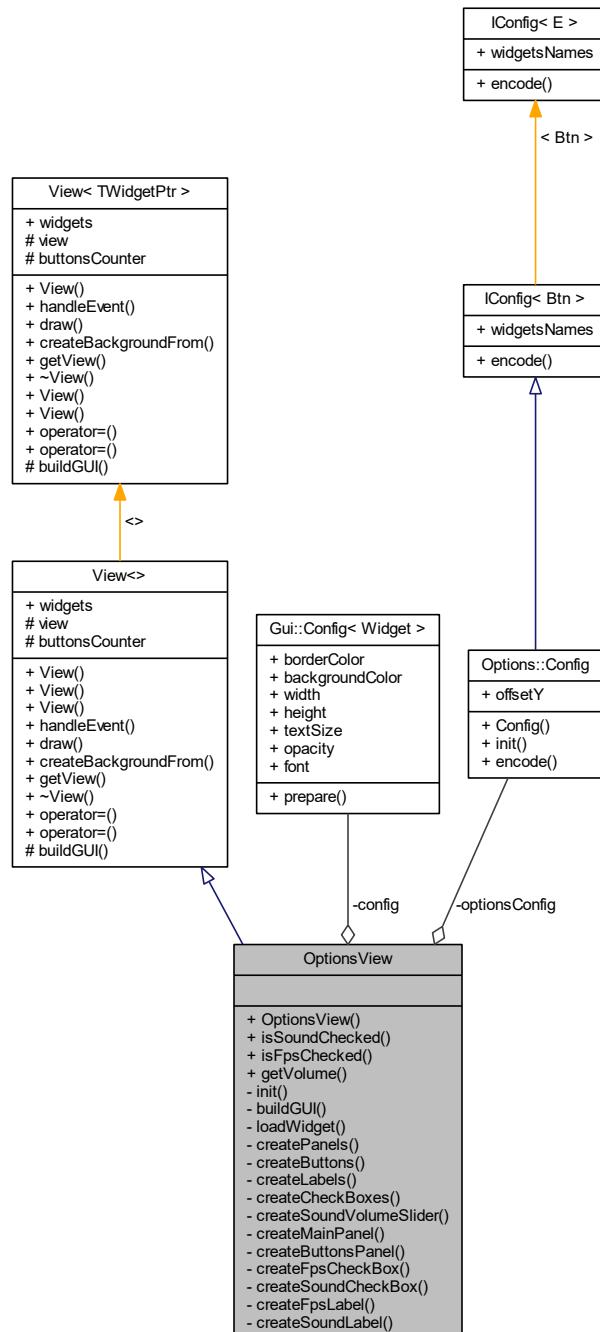
---

```
#include <OptionsView.h>
```

Inheritance diagram for OptionsView:



Collaboration diagram for OptionsView:



## Public Member Functions

- [OptionsView \(Window &window\)](#)  
*Constructs the OptionsView.*
- bool [isSoundChecked \(\) const](#)
- bool [isFpsChecked \(\) const](#)
- float [getVolume \(\) const](#)

## Private Member Functions

- void `init ()`  
*Initializes the [OptionsView](#). Effectively: calls `buildGUI()`.*
- void `buildGUI () override`
- void `loadWidget (tgui::Widget::Ptr &widget)`  
*Loads a given widget (likely: a button) onto the correct settings and position.*
- void `createPanels ()`  
*Creates all panels and adds them to the #gui.*
- void `createButtons ()`  
*Creates all buttons and adds them to the #gui.*
- void `createLabels ()`  
*Creates all labels and adds them to the #gui.*
- void `createCheckboxes ()`  
*Creates all check Boxes and adds them to the #gui.*
- void `createSoundVolumeSlider ()`  
*Creates a sound volume slider and adds it to the #gui.*
- void `createMainPanel ()`  
*Creates the main panel and adds it to the #gui.*
- void `createButtonsPanel ()`  
*Creates the buttons panel and adds it to the #gui.*
- void `createFpsCheckBox ()`  
*Creates FPS enable check box and adds it to the #gui.*
- void `createSoundCheckBox ()`  
*Creates sound enable check box and adds it to the #gui.*
- void `createFpsLabel ()`  
*Creates FPS enabled label and adds it to the #gui.*
- void `createSoundLabel ()`  
*Creates sound enabled label box and adds it to the #gui.*

## Private Attributes

- `Gui::Config config`  
*Configuration settings common to all [View](#) classes.*
- `Options::Config optionsConfig`  
*Configuration settings specific to [OptionsView](#).*

## Additional Inherited Members

### 10.41.1 Detailed Description

[View](#) class used to draw gui within [StateOptions](#).

### 10.41.2 Constructor & Destructor Documentation

#### 10.41.2.1 OptionsView()

```
OptionsView::OptionsView (
    Window & window ) [explicit]
```

Constructs the [OptionsView](#).

## Parameters

|                        |   |
|------------------------|---|
| <a href="#">Window</a> | that is bound to <a href="#">View</a> base. |
|------------------------|---|

### 10.41.3 Member Function Documentation

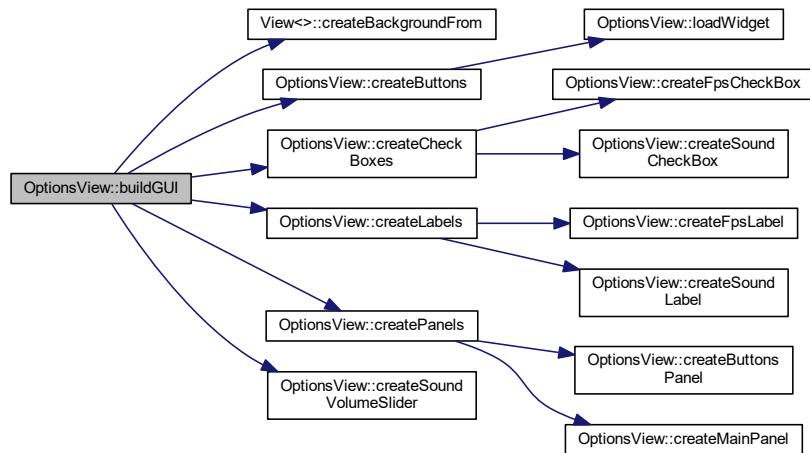
#### 10.41.3.1 buildGUI()

```
void OptionsView::buildGUI ( ) [override], [private], [virtual]
```

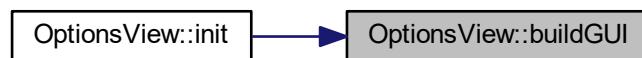
Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements [View<>](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.41.3.2 `createButtons()`

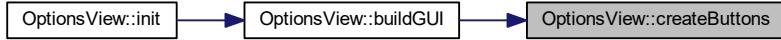
```
void OptionsView::createButtons ( ) [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.41.3.3 `createButtonsPanel()`

```
void OptionsView::createButtonsPanel ( ) [private]
```

Creates the buttons panel and adds it to the #gui.

Here is the caller graph for this function:

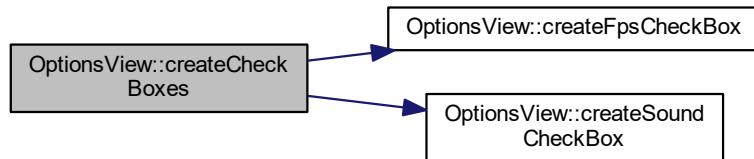


#### 10.41.3.4 createCheckBoxes()

```
void OptionsView::createCheckBoxes ( ) [private]
```

Creates all check Boxes and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:

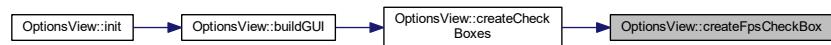


#### 10.41.3.5 createFpsCheckBox()

```
void OptionsView::createFpsCheckBox ( ) [private]
```

Creates FPS enable check box and adds it to the #gui.

Here is the caller graph for this function:

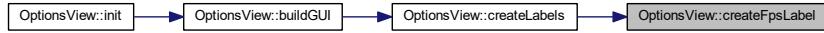


#### 10.41.3.6 `createFpsLabel()`

```
void OptionsView::createFpsLabel ( ) [private]
```

Creates FPS enabled label and adds it to the #gui.

Here is the caller graph for this function:

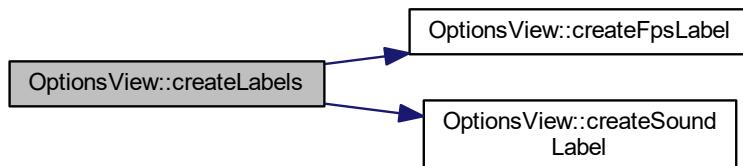


#### 10.41.3.7 `createLabels()`

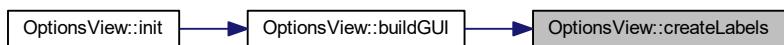
```
void OptionsView::createLabels ( ) [private]
```

Creates all labels and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:

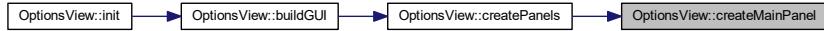


### 10.41.3.8 createMainPanel()

```
void OptionsView::createMainPanel ( ) [private]
```

Creates the main panel and adds it to the #gui.

Here is the caller graph for this function:

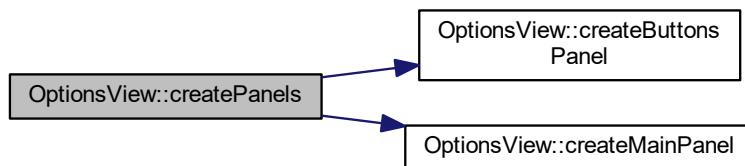


### 10.41.3.9 createPanels()

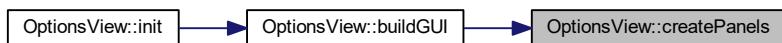
```
void OptionsView::createPanels ( ) [private]
```

Creates all panels and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.41.3.10 createSoundCheckBox()

```
void OptionsView::createSoundCheckBox ( ) [private]
```

Creates sound enable check box and adds it to the #gui.

Here is the caller graph for this function:



#### 10.41.3.11 createSoundLabel()

```
void OptionsView::createSoundLabel ( ) [private]
```

Creates sound enabled label box and adds it to the #gui.

Here is the caller graph for this function:

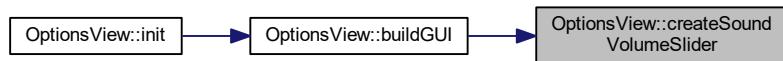


#### 10.41.3.12 createSoundVolumeSlider()

```
void OptionsView::createSoundVolumeSlider ( ) [private]
```

Creates a sound volume slider and adds it to the #gui.

Here is the caller graph for this function:



### 10.41.3.13 `getVolume()`

```
float OptionsView::getVolume ( ) const
```

Here is the caller graph for this function:

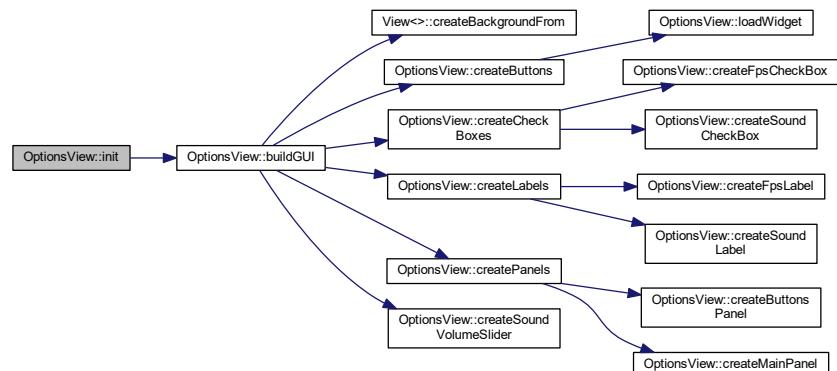


### 10.41.3.14 `init()`

```
void OptionsView::init ( ) [private]
```

Initializes the [OptionsView](#). Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



### 10.41.3.15 `isFpsChecked()`

```
bool OptionsView::isFpsChecked ( ) const
```

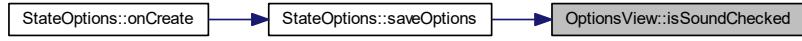
Here is the caller graph for this function:



#### 10.41.3.16 `isSoundChecked()`

```
bool OptionsView::isSoundChecked ( ) const
```

Here is the caller graph for this function:



#### 10.41.3.17 `loadWidget()`

```
void OptionsView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position.

##### Parameters

|                     |               |
|---------------------|---------------|
| <code>widget</code> | to be loaded. |
|---------------------|---------------|

##### Note

Can be used in a loop.

Here is the caller graph for this function:



### 10.41.4 Member Data Documentation

#### 10.41.4.1 `config`

```
Gui::Config OptionsView::config [private]
```

Configuration settings common to all [View](#) classes.

#### 10.41.4.2 optionsConfig

```
Options::Config OptionsView::optionsConfig [private]
```

Configuration settings specific to [OptionsView](#).

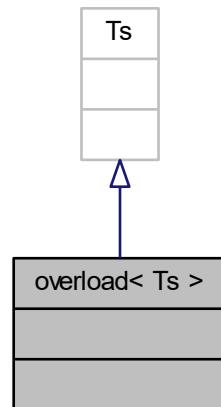
The documentation for this class was generated from the following files:

- [OptionsView.h](#)
- [OptionsView.cpp](#)

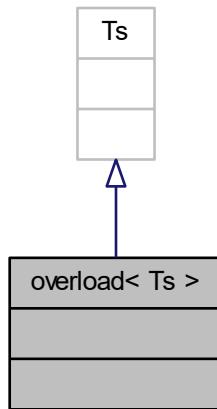
## 10.42 overload< Ts > Struct Template Reference

```
#include <Overload.h>
```

Inheritance diagram for overload< Ts >:



Collaboration diagram for overload< Ts >:



#### 10.42.1 Detailed Description

```
template<class... Ts>
struct overload< Ts >
```

Examples

<C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp>.

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

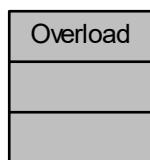
- [Overload.h](#)

### 10.43 Overload Struct Reference

[Overload](#) lambda *hack* for variant std::visit.

```
#include <Overload.h>
```

Collaboration diagram for Overload:



### 10.43.1 Detailed Description

[Overload](#) lambda *hack* for variant std::visit.

•

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

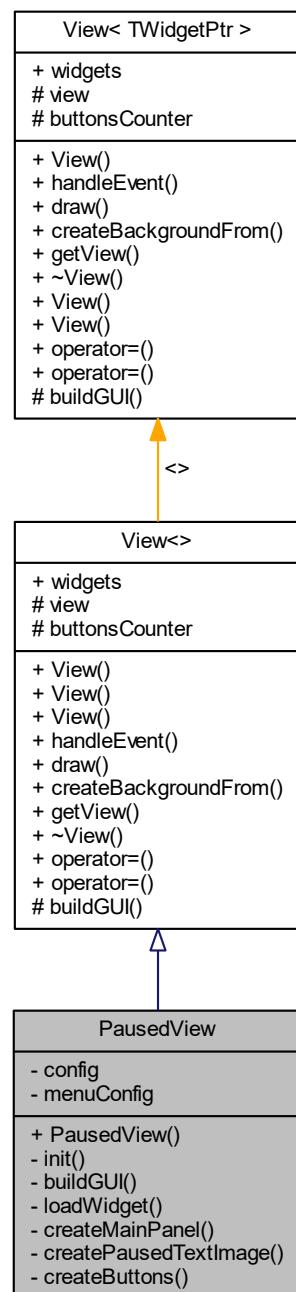
- [Overload.h](#)

## 10.44 PausedView Class Reference

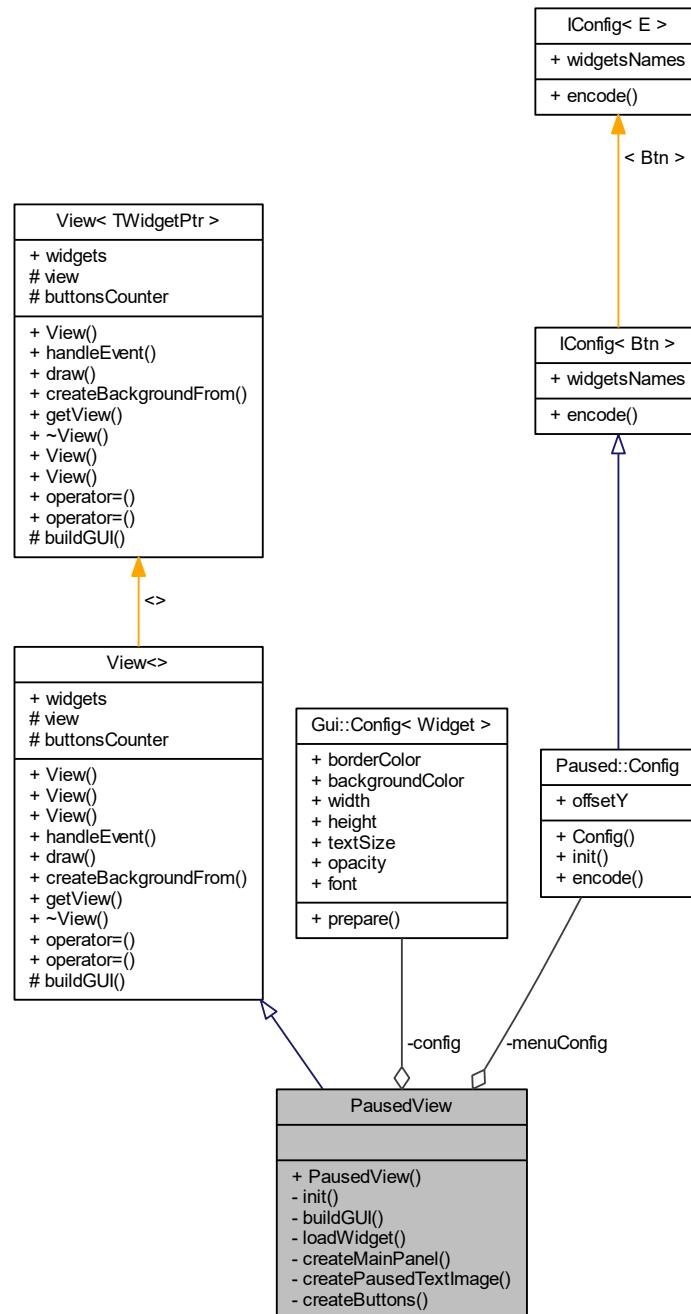
[View](#) class used to draw gui within [StatePaused](#).

```
#include <PausedView.h>
```

Inheritance diagram for PausedView:



Collaboration diagram for PausedView:



## Public Member Functions

- [PausedView \(Window &window\)](#)

*Constructs the PausedView.*

## Private Member Functions

- void `init ()`  
*Initializes the [PausedView](#). Effectively: calls `buildGUI()`.*
- void `buildGUI () override`
- void `loadWidget (tgui::Widget::Ptr &widget)`  
*Loads a given widget (likely: a button) onto the correct settings and position. @widget to be loaded.*
- void `createMainPanel ()`  
*Creates the main panel and adds it to the #gui.*
- void `createPausedTextImage ()`  
*Create the [Paused](#) Text Image widget.*
- void `createButtons ()`  
*Creates all buttons and adds them to the #gui.*

## Private Attributes

- `Gui::Config config`  
*Configuration settings common to all [View](#) classes.*
- `Paused::Config menuConfig`  
*Configuration settings specific to [PausedView](#).*

## Additional Inherited Members

### 10.44.1 Detailed Description

[View](#) class used to draw gui within [StatePaused](#).

### 10.44.2 Constructor & Destructor Documentation

#### 10.44.2.1 `PausedView()`

```
PausedView::PausedView (
    Window & window ) [explicit]
```

Constructs the [PausedView](#).

#### Parameters

|                        |   |
|------------------------|---|
| <a href="#">Window</a> | that is bound to <a href="#">View</a> base. |
|------------------------|---|

### 10.44.3 Member Function Documentation

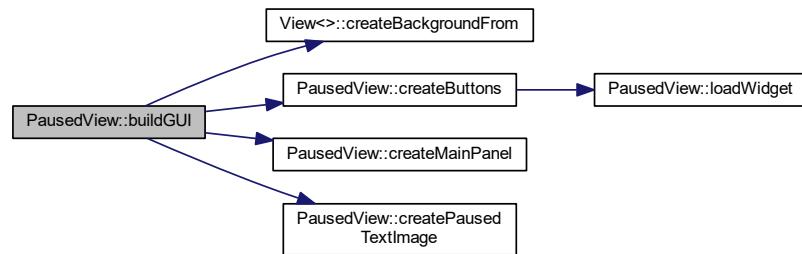
### 10.44.3.1 buildGUI()

```
void PausedView::buildGUI ( ) [override], [private], [virtual]
```

Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements [View<>](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.44.3.2 createButtons()

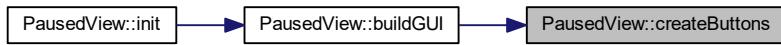
```
void PausedView::createButtons ( ) [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.44.3.3 `createMainPanel()`

```
void PausedView::createMainPanel ( ) [private]
```

Creates the main panel and adds it to the #gui.

Here is the caller graph for this function:



#### 10.44.3.4 `createPausedTextImage()`

```
void PausedView::createPausedTextImage ( ) [private]
```

Create the [Paused](#) Text Image widget.

Here is the caller graph for this function:

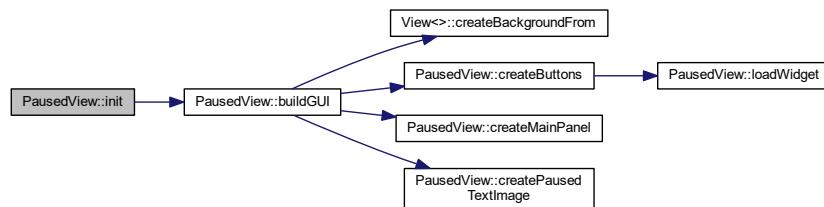


### 10.44.3.5 init()

```
void PausedView::init ( ) [private]
```

Initializes the [PausedView](#). Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



### 10.44.3.6 loadWidget()

```
void PausedView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position. @widget to be loaded.

#### Note

Can be used in a loop.

Here is the caller graph for this function:



## 10.44.4 Member Data Documentation

### 10.44.4.1 config

```
Gui::Config PausedView::config [private]
```

Configuration settings common to all [View](#) classes.

#### 10.44.4.2 menuConfig

```
Paused::Config PausedView::menuConfig [private]
```

Configuration settings specific to [PausedView](#).

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

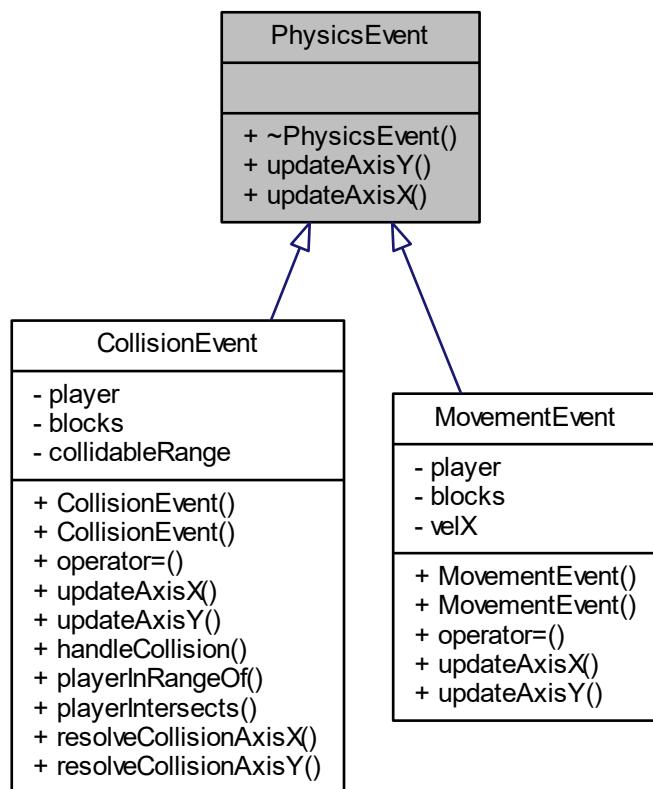
- [PausedView.h](#)
- [PausedView.cpp](#)

## 10.45 PhysicsEvent Class Reference

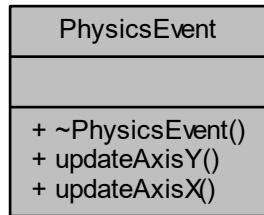
Specified Base [PhysicsEvent](#) class handling physics-related events on two separate axes.

```
#include <PhysicsEvent.h>
```

Inheritance diagram for PhysicsEvent:



Collaboration diagram for PhysicsEvent:



## Public Member Functions

- virtual [~PhysicsEvent \(\)=default](#)  
*Default the destructor.*
- virtual void [updateAxisY \(float dt\)=0](#)  
*Handles the events on X axis.*
- virtual void [updateAxisX \(float dt\)=0](#)  
*Handles the events on Y axis.*

### 10.45.1 Detailed Description

Specified Base [PhysicsEvent](#) class handling physics-related events on two separate axes.

### 10.45.2 Constructor & Destructor Documentation

#### 10.45.2.1 ~PhysicsEvent()

```
virtual PhysicsEvent::~PhysicsEvent ( ) [virtual], [default]
```

Default the destructor.

### 10.45.3 Member Function Documentation

#### 10.45.3.1 updateAxisX()

```
virtual void PhysicsEvent::updateAxisX (
    float dt ) [pure virtual]
```

Handles the events on Y axis.

Implemented in [CollisionEvent](#), and [MovementEvent](#).

#### 10.45.3.2 updateAxisY()

```
virtual void PhysicsEvent::updateAxisY (
    float dt ) [pure virtual]
```

Handles the events on X axis.

Implemented in [CollisionEvent](#), and [MovementEvent](#).

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

- [PhysicsEvent.h](#)

## 10.46 PixelColor Struct Reference

Container for one .bmp BGR pixel.

```
#include <PixelColor.h>
```

Collaboration diagram for PixelColor:



## Public Member Functions

- `PixelColor ()=default`  
*Create default PixelColor.*
- `PixelColor (uint8_t blue, uint8_t green, uint8_t red)`  
*Create PixelColor from given BGR parameters.*
- `bool operator< (const PixelColor &other) const`  
*Overload operator< so that PixelColor can be used as a map key inside Encoder<ReaderKey> class.*

## Public Attributes

- `uint8_t blue`
- `uint8_t green`
- `uint8_t red`

## Friends

- `std::istream & operator>> (std::istream &input, PixelColor &color)`  
*Overload operator>> for convenience when reading .bmp into a data vector.*

### 10.46.1 Detailed Description

Container for one .bmp BGR pixel.

Used mostly for readability and convenience as opposed to necessity.

#### Note

Bear in mind .bmp image format uses BGR color format, not RGB.

### 10.46.2 Constructor & Destructor Documentation

#### 10.46.2.1 PixelColor() [1/2]

```
PixelColor::PixelColor ( ) [default]
```

Create default PixelColor.

#### 10.46.2.2 PixelColor() [2/2]

```
PixelColor::PixelColor (
    uint8_t blue,
    uint8_t green,
    uint8_t red ) [inline]
```

Create PixelColor from given BGR parameters.

**Parameters**

|              |             |
|--------------|-------------|
| <i>blue</i>  | value 0-255 |
| <i>green</i> | value 0-255 |
| <i>red</i>   | value 0-255 |

**10.46.3 Member Function Documentation****10.46.3.1 operator<()**

```
bool PixelColor::operator< (
    const PixelColor & other ) const [inline]
```

[Overload](#) operator< so that `PixelColor` can be used as a map key inside `Encoder<ReaderKey>` class.

**10.46.4 Friends And Related Function Documentation****10.46.4.1 operator>>**

```
std::istream& operator>> (
    std::istream & input,
    PixelColor & color ) [friend]
```

[Overload](#) operator>> for convenience when reading .bmp into a data vector.

**10.46.5 Member Data Documentation****10.46.5.1 blue**

```
uint8_t PixelColor::blue
```

**10.46.5.2 green**

```
uint8_t PixelColor::green
```

### 10.46.5.3 red

```
uint8_t PixelColor::red
```

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

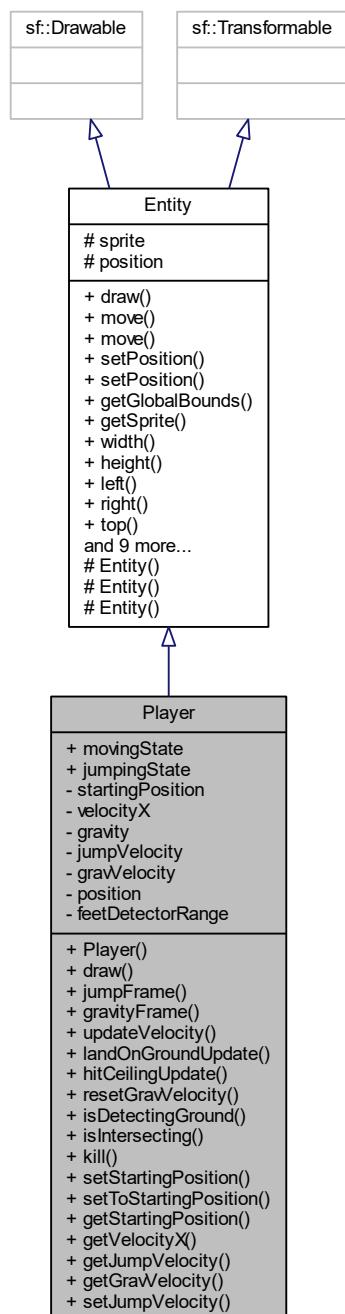
- [PixelColor.h](#)

## 10.47 Player Class Reference

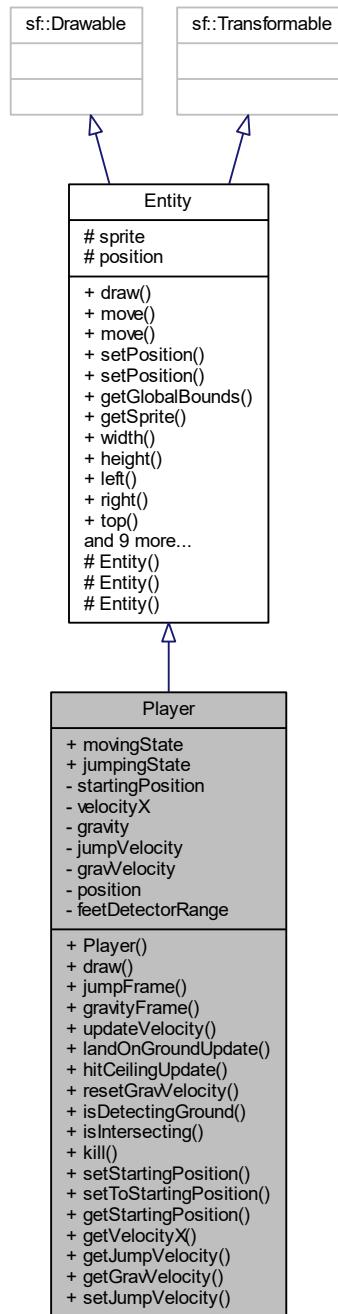
Controls the player behaviour.

```
#include <Player.h>
```

Inheritance diagram for Player:



Collaboration diagram for Player:



## Public Member Functions

- **Player** (`sf::Vector2f position={0, 0}`)  
*Constructs the Player.*
- void **draw** (`sf::RenderTarget &target, sf::RenderStates states`) const override  
*Draw method used to render `sprite` onto the window.*
- void **jumpFrame** (`float dt`)

- `void gravityFrame (float dt)`

*Performs one tick of a jumping logic.*
- `void updateVelocity (float &velocity, float dt) const noexcept`

*Common method for `jumpFrame()` and `gravityFrame()`. Updates the velocity based on dt.*
- `void landOnGroundUpdate ()`

*Update the variables when `Player` lands on ground.*
- `void hitCeilingUpdate ()`

*Update the variables when `Player` hits the ceiling.*
- `void resetGravVelocity ()`

*Nullify the `gravVelocity` when `Player` lands on ground.*
- `bool isDetectingGround (const std::vector< std::unique_ptr< Entity >> &blocks) const`
- `bool isIntersecting (const Entity &entity) const`
- `void kill (LivesOverlay &lives)`
- `void setStartingPosition ()`

*Sets `startingPosition` to wherever the `Player` currently stands.*
- `void setToStartingPosition ()`

*Set `Player` to his `startingPosition`.*
- `sf::Vector2f getStartingPosition () const`

*Retrieve `startingPosition`.*
- `float getVelocityX () const`

*Retrieve horizontal X velocity.*
- `float getJumpVelocity () const`

*Retrieve current jump Y velocity.*
- `float getGravVelocity () const`

*Retrieve current gravitational Y velocity.*
- `void setJumpVelocity (float newVal)`

*Sets a `jumpVelocity` to a new value.*

## Public Attributes

- `MovingState movingState {MovingState::standing}`

*Represents the horizontal MovingState of the player. `Player` is in one state at a time.*
- `JumpingState jumpingState {JumpingState::onGround}`

*Represents the vertical JumpingState of the player. `Player` is in one state at a time.*

## Private Attributes

- `sf::Vector2f startingPosition`

*Position player is set to when he needs to be re-spawned.*
- `const float velocityX = config::horizontalVelocity`

*Horizontal velocity is a constant.*
- `const float gravity = config::gravity`

*Gravity acceleration is a constant.*
- `float jumpVelocity = config::initialJumpVelocity`

*Used when player himself jumped.*
- `float gravVelocity = 0.0f`

*Used when player is free-falling (e.g. slipped off edge)*
- `float position {}`

## Static Private Attributes

- `constexpr static auto feetDetectorRange = 5.0f`  
*Fixed on player's feet to detect the ground.*

## Additional Inherited Members

### 10.47.1 Detailed Description

Controls the player behaviour.

[Player](#) can be in different states:

See also

- [MovingState](#)

See also

- [JumpingState](#)

[Player](#) texture changes based on running direction.

#### Note

A concrete [Entity](#) class overriding the draw method.

### 10.47.2 Constructor & Destructor Documentation

#### 10.47.2.1 Player()

```
Player::Player (
    sf::Vector2f position = {0, 0} ) [explicit]
```

Constructs the [Player](#).

Parameters

|                       |                  |
|-----------------------|------------------|
| <code>position</code> | initial position |
|-----------------------|------------------|

### 10.47.3 Member Function Documentation

### 10.47.3.1 draw()

```
void Player::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render [sprite](#) onto the window.

#### Parameters

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides [Entity](#) draw method.

#### Note

Passing states is optional.

### 10.47.3.2 getGravVelocity()

```
float Player::getGravVelocity ( ) const
```

Retrieve current gravitational Y velocity.

#### Note

Possibly unused.

#### Returns

float horizontal velocity.

### 10.47.3.3 getJumpVelocity()

```
float Player::getJumpVelocity ( ) const
```

Retrieve current jump Y velocity.

#### Note

Possibly unused.

#### Returns

float current jump velocity.

#### 10.47.3.4 getStartingPosition()

```
sf::Vector2f Player::getStartingPosition ( ) const
```

Retrieve [startingPosition](#).

##### Returns

`sf::Vector2f` the player's starting position.

Here is the caller graph for this function:



#### 10.47.3.5 getVelocityX()

```
float Player::getVelocityX ( ) const
```

Retrieve horizontal X velocity.

##### Returns

`float` horizontal velocity.

#### 10.47.3.6 gravityFrame()

```
void Player::gravityFrame ( float dt )
```

Performs one tick of a passive gravity logic.

**Note**

Frame-rate independent gravity. A gravity sequence consists of many consecutive frames. [Player](#) has different [gravVelocity](#) based on the state of jump variables.

Here is the call graph for this function:



Here is the caller graph for this function:

**10.47.3.7 hitCeilingUpdate()**

```
void Player::hitCeilingUpdate( )
```

Update the variables when [Player](#) hits the ceiling.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.47.3.8 isDetectingGround()

```
bool Player::isDetectingGround (
    const std::vector< std::unique_ptr< Entity >> & blocks ) const
```

Check if [Player](#) is standing on the ground.

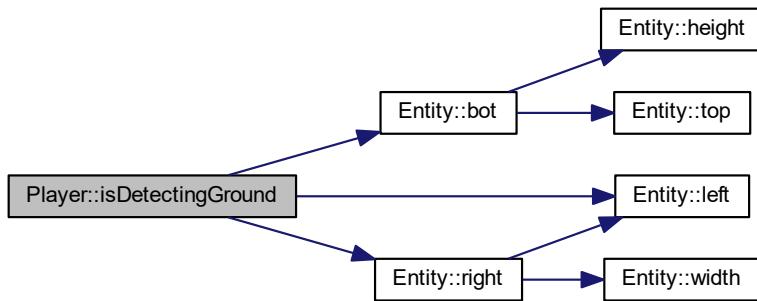
#### Note

[Player](#) has ground detectors fixed below his feet.

#### See also

[feetDetectorRange](#)

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.47.3.9 isIntersecting()

```
bool Player::isIntersecting (
    const Entity & entity ) const
```

Check if [Player](#) is intersecting other entity.

**Parameters**

|               |                           |
|---------------|---------------------------|
| <i>entity</i> | an arbitrary game entity. |
|---------------|---------------------------|

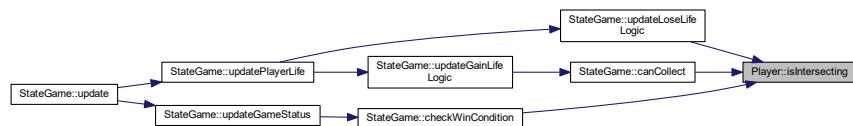
**Returns**

Is player intersecting that entity?

Here is the call graph for this function:



Here is the caller graph for this function:

**10.47.3.10 jumpFrame()**

```
void Player::jumpFrame (
    float dt )
```

Performs one tick of a jumping logic.

**Note**

Frame-rate independent jump.

A jump consists of many consecutive frames. [Player](#) has different [jumpVelocity](#) based on the state of jump variables. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.47.3.11 kill()

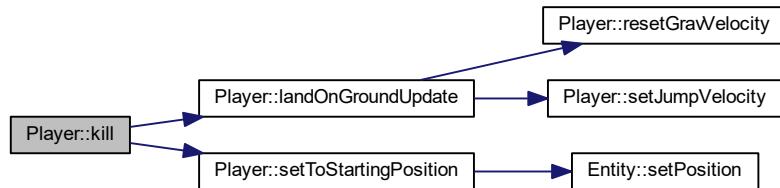
```
void Player::kill (
    LivesOverlay & lives )
```

Kills the player.

#### Parameters

|              |                                      |
|--------------|--------------------------------------|
| <i>lives</i> | an overlay of remaining player lives |
|--------------|--------------------------------------|

Effectively: one player [Life](#) is removed, and [Player](#) re-spawns on the [startingPosition](#). Here is the call graph for this function:



Here is the caller graph for this function:

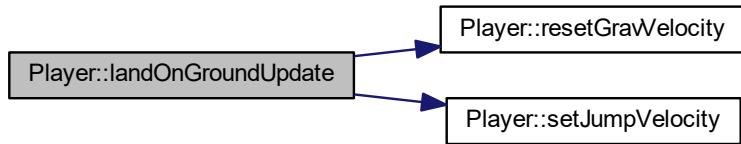


### 10.47.3.12 landOnGroundUpdate()

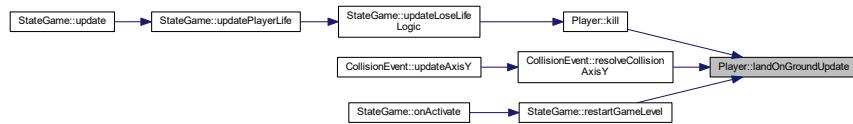
```
void Player::landOnGroundUpdate( )
```

Update the variables when [Player](#) lands on ground.

Here is the call graph for this function:



Here is the caller graph for this function:

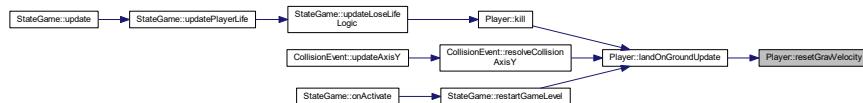


### 10.47.3.13 resetGravVelocity()

```
void Player::resetGravVelocity( )
```

Nullify the [gravVelocity](#) when [Player](#) lands on ground.

Here is the caller graph for this function:



### 10.47.3.14 setJumpVelocity()

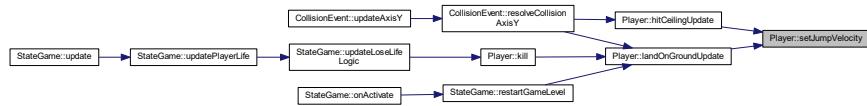
```
void Player::setJumpVelocity(
    float newVal )
```

Sets a [jumpVelocity](#) to a new value.

### Parameters

|                     |   |
|---------------------|---|
| <code>newVal</code> | new value of <a href="#">jumpVelocity</a> to be set |
|---------------------|---|

Here is the caller graph for this function:



### 10.47.3.15 setStartingPosition()

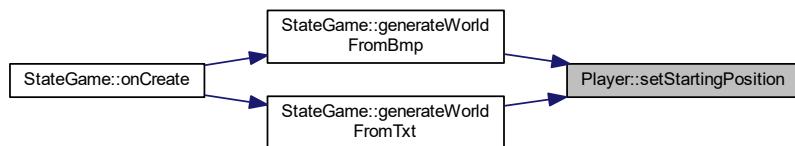
```
void Player::setStartingPosition ( )
```

Sets [startingPosition](#) to wherever the [Player](#) currently stands.

### Warning

Do not confuse with [setToStartingPosition\(\)](#).

Here is the caller graph for this function:



### 10.47.3.16 setToStartingPosition()

```
void Player::setToStartingPosition ( )
```

Set [Player](#) to his [startingPosition](#).

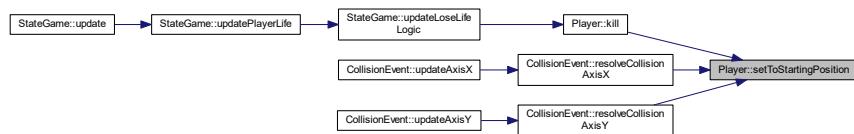
### Warning

Do not confuse with [setStartingPosition\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.47.3.17 updateVelocity()

```
void Player::updateVelocity (
    float & velocity,
    float dt ) const [noexcept]
```

Common method for [jumpFrame\(\)](#) and [gravityFrame\(\)](#). Updates the velocity based on dt.

Terminal velocity is set to infinite velocity growth.

#### Parameters

|                 |   |
|-----------------|---|
| <i>velocity</i> | either <a href="#">gravVelocity</a> or <a href="#">jumpVelocity</a> to be updated |
| <i>dt</i>       | time elapsed since previous frame   |

Here is the caller graph for this function:



## 10.47.4 Member Data Documentation

### 10.47.4.1 feetDetectorRange

```
constexpr static auto Player::feetDetectorRange = 5.0f [static], [constexpr], [private]
```

Fixed on player's feet to detect the ground.

### 10.47.4.2 gravity

```
const float Player::gravity = config::gravity [private]
```

Gravity acceleration is a constant.

### 10.47.4.3 gravVelocity

```
float Player::gravVelocity = 0.0f [private]
```

Used when player is free-falling (e.g. slipped off edge)

### 10.47.4.4 jumpingState

```
JumpingState Player::jumpingState {JumpingState::onGround}
```

Represents the vertical JumpingState of the player. `Player` is in one state at a time.

#### 10.47.4.5 jumpVelocity

```
float Player::jumpVelocity = config::initialJumpVelocity [private]
```

Used when player himself jumped.

Two kinds of vertical velocity

#### 10.47.4.6 movingState

```
MovingState Player::movingState {MovingState::standing}
```

Represents the horizontal MovingState of the player. [Player](#) is in one state at a time.

#### 10.47.4.7 position

```
float Player::position {} [private]
```

#### 10.47.4.8 startingPosition

```
sf::Vector2f Player::startingPosition [private]
```

Position player is set to when he needs to be re-spawned.

#### 10.47.4.9 velocityX

```
const float Player::velocityX = config::horizontalVelocity [private]
```

Horizontal velocity is a constant.

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

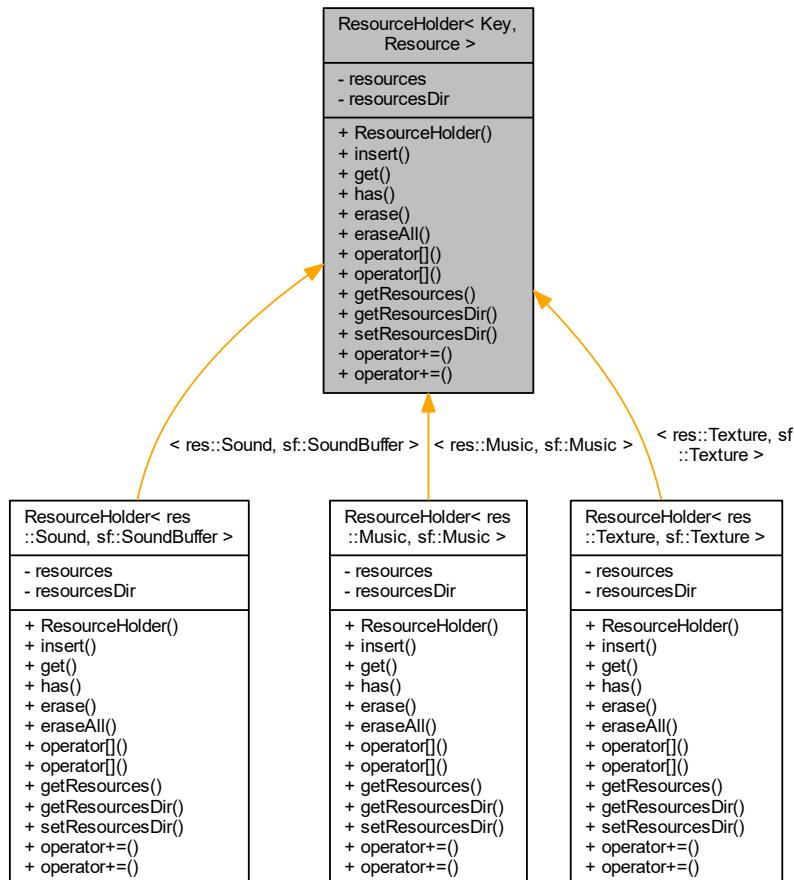
- [Player.h](#)
- [Player.cpp](#)

## 10.48 ResourceHolder< Key, Resource > Class Template Reference

A generic [ResourceHolder](#) storing resources of type Resource, accessed by Key.

```
#include <ResourceHolder.h>
```

Inheritance diagram for ResourceHolder< Key, Resource >:



Collaboration diagram for ResourceHolder< Key, Resource >:

|   |
|---|
| <b>ResourceHolder&lt; Key,<br/>Resource &gt;</b>  |
| - resources<br>- resourcesDir   |
| + ResourceHolder()<br>+ insert()<br>+ get()<br>+ has()<br>+ erase()<br>+ eraseAll()<br>+ operator[]( )<br>+ operator[]( )<br>+ getResources()<br>+ getResourcesDir()<br>+ setResourcesDir()<br>+ operator+=()<br>+ operator+=() |

## Public Member Functions

- **ResourceHolder** (std::string \_view resourcesDir="..../resources/")
 

*Constructs an empty [ResourceHolder](#) with a given resourceDir root.*
- template<typename... Args>  
**void insert** (const Key &key, std::string \_view fileName, Args &&... args)
 

*Insert a resource of type Resource to a map.*
- **Resource & get** (const Key &key) const
 

*Retrieves a reference to an existing resource found by a given ID.*
- **bool has** (const Key &key) const
 

*Check if a [ResourceHolder](#) has a given key stored.*
- **void erase** (const Key &key) noexcept
 

*Erase a Resource found by a given key.*
- **void eraseAll** () noexcept
 

*Erase all existing resources within this [ResourceHolder](#).*
- **const Resource & operator[]** (const Key &key) const
- **Resource & operator[]** (const Key &key)
- **auto & getResources** () const
 

*Retrieve all resources.*
- **auto & getResourcesDir** () const
 

*Retrieve resources root directory.*
- **void setResourcesDir** (std::string newPath)
 

*Set the \$resourceDir# to a given new directory.*
- template<typename... Args>  
**ResourceHolder & operator+=** (const [ResourceInserter](#)< Key, Args... > &inserter)
 

*Append resources from another [ResourceHolder](#) to this one.*

*Use operator+= alternatively to [insert\(\)](#) method.*

- template<typename... Args>  
**ResourceHolder** & **operator+=** (**ResourceInserter< Key, Args... >** &&inserter)  
*Use operator+= alternatively to [insert\(\)](#) method.*

## Private Attributes

- std::unordered\_map< Key, std::unique\_ptr< Resource > > **resources**
- std::string **resourcesDir**

### 10.48.1 Detailed Description

```
template<typename Key, typename Resource>
class ResourceHolder< Key, Resource >
```

A generic **ResourceHolder** storing resources of type **Resource**, accessed by **Key**.

Custom Mappable concept is used only if C++20 flag is enabled.

**ResourceHolder** has a 'if constexpr method branch specialization' if the Reosource is sf::Music.

Uses Arguments parameter pack to suit future needs (number of arguments in [insert\(\)](#) method may vary).

#### Note

Resources can be inserted with [insert\(\)](#) or **operator+=**.

Resources can be accessed with [get\(\)](#) or **operator[]**.

### 10.48.2 Constructor & Destructor Documentation

#### 10.48.2.1 ResourceHolder()

```
template<typename Key , typename Resource >
ResourceHolder< Key, Resource >::ResourceHolder (
    std::string_view resourcesDir = "../resources/" ) [inline], [explicit]
```

Constructs an empty **ResourceHolder** with a given resourceDir root.

#### Parameters

|                     |                          |
|---------------------|--------------------------|
| <i>resourcesDir</i> | resources root directory |
|---------------------|--------------------------|

ResourceDir defaults to "../resources/".

### 10.48.3 Member Function Documentation

#### 10.48.3.1 erase()

```
template<typename Key , typename Resource >
void ResourceHolder< Key, Resource >::erase (
    const Key & key ) [inline], [noexcept]
```

Erase a Resource found by a given key.

##### Parameters

|            |   |
|------------|---|
| <i>key</i> | The key to be used to find a Resource to be erased. |
|------------|---|

#### 10.48.3.2 eraseAll()

```
template<typename Key , typename Resource >
void ResourceHolder< Key, Resource >::eraseAll ( ) [inline], [noexcept]
```

Erase all existing resources within this [ResourceHolder](#).

#### 10.48.3.3 get()

```
template<typename Key , typename Resource >
Resource& ResourceHolder< Key, Resource >::get (
    const Key & key ) const [inline]
```

Retrieves a reference to an existing resource found by a given ID.

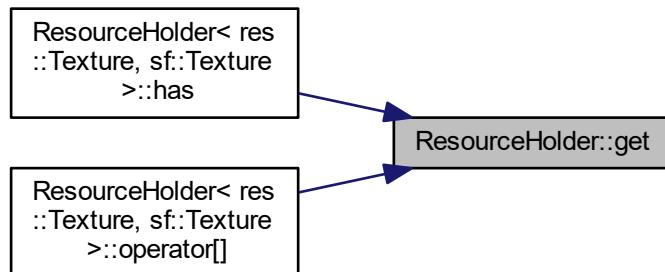
##### Parameters

|            |                      |
|------------|----------------------|
| <i>key</i> | Resource identifier. |
|------------|----------------------|

**Returns**

Resource& a reference to the found, existing resource.

May throw. Here is the caller graph for this function:

**10.48.3.4 getResources()**

```
template<typename Key , typename Resource >
auto& ResourceHolder< Key, Resource >::getResources ( ) const [inline]
```

Retrieve all resources.

**Returns**

a reference to all resources.

**10.48.3.5 getResourcesDir()**

```
template<typename Key , typename Resource >
auto& ResourceHolder< Key, Resource >::getResourcesDir ( ) const [inline]
```

Retrieve resources root directory.

**Returns**

a reference to [resourcesDir](#).

**10.48.3.6 has()**

```
template<typename Key , typename Resource >
bool ResourceHolder< Key, Resource >::has (
    const Key & key ) const [inline]
```

Check if a [ResourceHolder](#) has a given key stored.

**Parameters**

|                  |  |
|------------------|--|
| <code>key</code> | The key to be checked if it is already stored. |
|------------------|--|

**Returns**

Is the key already stored?

**10.48.3.7 `insert()`**

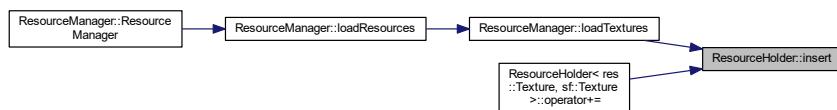
```
template<typename Key , typename Resource >
template<typename... Args>
void ResourceHolder< Key, Resource >::insert (
    const Key & key,
    std::string_view fileName,
    Args &... args ) [inline]
```

Insert a resource of type `Resource` to a map.

**Parameters**

|                       |   |
|-----------------------|---|
| <code>key</code>      | A key stored as a key in map, needs to be remembered to retrieve the resource.              |
| <code>fileName</code> | A file path to the Resource.  |
| <code>args</code>     | Optional parameter pack, could be used in e.g. <code>sf::shader</code> class in the future. |

`sf::Music` has slightly incompatible interface as opposed to other resources, so 'if constexpr' is evaluated. Here is the caller graph for this function:

**10.48.3.8 `operator+=()` [1/2]**

```
template<typename Key , typename Resource >
template<typename... Args>
ResourceHolder& ResourceHolder< Key, Resource >::operator+= (
    const ResourceInserter< Key, Args... > & inserter ) [inline]
```

Use `operator+=` alternatively to `insert()` method.

**Parameters**

|                 |  |
|-----------------|--|
| <i>inserter</i> | const reference to a ResourceInserter<Key, Args...> proxy class/ |
|-----------------|--|

**Returns**

a reference to a newly stored resource.

Uses Proxy pattern to make operator+= compatible with 2 parameters.

**Note**

Const lvalue reference version.

**10.48.3.9 operator+=( ) [2/2]**

```
template<typename Key , typename Resource >
template<typename... Args>
ResourceHolder& ResourceHolder< Key, Resource >::operator+= (
    ResourceInserter< Key, Args... > && inserter ) [inline]
```

Use operator+= alternatively to [insert\(\)](#) method.

**Parameters**

|                 |   |
|-----------------|---|
| <i>inserter</i> | rvalue of a ResourceInserter<Key, Args...> proxy class/ |
|-----------------|---|

**Returns**

a reference to a newly stored resource.

Uses Proxy pattern to make operator+= compatible with 2 parameters.

**Note**

Rvalue reference version.

**10.48.3.10 operator[]( ) [1/2]**

```
template<typename Key , typename Resource >
Resource& ResourceHolder< Key, Resource >::operator[] (
    const Key & key ) [inline]
```

Use operator[] alternatively to retrieve a resource.

**Parameters**

|            |              |
|------------|--------------|
| <i>key</i> | to be found. |
|------------|--------------|

**Returns**

Resource found by a given key.

Non-const version.

**10.48.3.11 operator[]( ) [2/2]**

```
template<typename Key , typename Resource >
const Resource& ResourceHolder< Key, Resource >::operator[ ] (
    const Key & key ) const [inline]
```

Use operator[] alternatively to retrieve a resource.

**Parameters**

|            |              |
|------------|--------------|
| <i>key</i> | to be found. |
|------------|--------------|

**Returns**

Resource found by a given key.

Const version.

**10.48.3.12 setResourcesDir()**

```
template<typename Key , typename Resource >
void ResourceHolder< Key, Resource >::setResourcesDir (
    std::string newPath ) [inline]
```

Set the \$resourceDir# to a given new directory.

**Parameters**

|                |                         |
|----------------|-------------------------|
| <i>newPath</i> | new directory to be set |
|----------------|-------------------------|

**10.48.4 Member Data Documentation**

#### 10.48.4.1 resources

```
template<typename Key , typename Resource >
std::unordered_map<Key, std::unique_ptr<Resource>> ResourceHolder< Key, Resource >::resources
[private]
```

#### 10.48.4.2 resourcesDir

```
template<typename Key , typename Resource >
std::string ResourceHolder< Key, Resource >::resourcesDir [private]
```

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

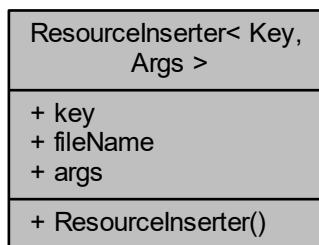
- [ResourceHolder.h](#)

## 10.49 ResourceInserter< Key, Args > Class Template Reference

A proxy class used to handle 2+ arguments in operator+=, which takes only 1 argument.

```
#include <ResourceInserter.h>
```

Collaboration diagram for ResourceInserter< Key, Args >:



### Public Member Functions

- [ResourceInserter \(Key key, std::string\\_view fileName, Args... args\)](#)

### Public Attributes

- Key [key](#)
- std::string [fileName](#)
- std::tuple<Args...> [args](#)

### 10.49.1 Detailed Description

```
template<typename Key, typename... Args>
class ResourceInserter< Key, Args >
```

A proxy class used to handle 2+ arguments in operator+=, which takes only 1 argument.

Uses Mappable custom concept (C++20).

#### Note

There are two ways to insert new resources into: ResourceHolder<Key, Resource> resources;

1: resources.insert(key, args...); 2: resources += [ResourceInserter\(key, args...\);](#)

[ResourceInserter](#) is a proxy class for operator+=, which is an alternative way of inserting resources

### 10.49.2 Constructor & Destructor Documentation

#### 10.49.2.1 ResourceInserter()

```
template<typename Key , typename... Args>
ResourceInserter< Key, Args >::ResourceInserter (
    Key key,
    std::string_view fileName,
    Args... args ) [inline], [explicit]
```

### 10.49.3 Member Data Documentation

#### 10.49.3.1 args

```
template<typename Key , typename... Args>
std::tuple<Args...> ResourceInserter< Key, Args >::args
```

#### 10.49.3.2 fileName

```
template<typename Key , typename... Args>
std::string ResourceInserter< Key, Args >::fileName
```

### 10.49.3.3 key

```
template<typename Key , typename... Args>
Key ResourceInserter< Key, Args >::key
```

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

- [ResourceInserter.h](#)

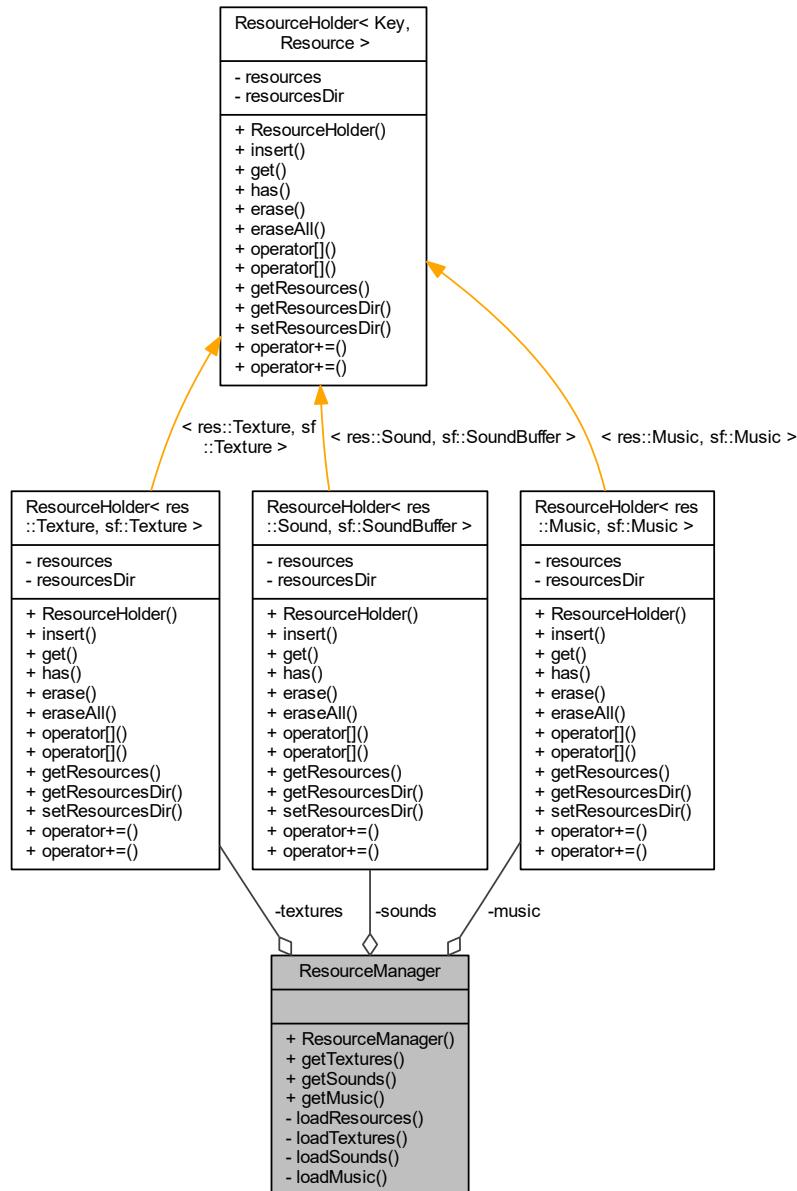
## 10.50 ResourceManager Class Reference

[ResourceManager](#) is a class responsible for managing all kinds of ResourceHolders.

---

```
#include <ResourceManager.h>
```

Collaboration diagram for ResourceManager:



## Public Member Functions

- `ResourceManager ()`  
*Call `loadResources()` which may throw.*
- `auto & getTextures ()`
- `auto & getSounds ()`
- `auto & getMusic ()`

## Private Member Functions

- `void loadResources ()`

- Loads all resources, each different resource type uses different thread to handle its type.*
- void [loadTextures \(\)](#)  
*Loads all texture resources. May throw.*
  - void [loadSounds \(\)](#)  
*Loads all sound resources. May throw.*
  - void [loadMusic \(\)](#)  
*Loads all music resources. May throw.*

## Private Attributes

- [`ResourceHolder< res::Texture, sf::Texture > textures {"..../resources/Textures/"}`](#)  
*textures container*
- [`ResourceHolder< res::Sound, sf::SoundBuffer > sounds {"..../resources/Sounds/"}`](#)  
*sounds container*
- [`ResourceHolder< res::Music, sf::Music > music {"..../resources/Music/"}`](#)  
*music container*

### 10.50.1 Detailed Description

[ResourceManager](#) is a class responsible for managing all kinds of ResourceHolders.

Encapsulates TexturesHolder, SoundsHolder and MusicHolder.

### 10.50.2 Constructor & Destructor Documentation

#### 10.50.2.1 ResourceManager()

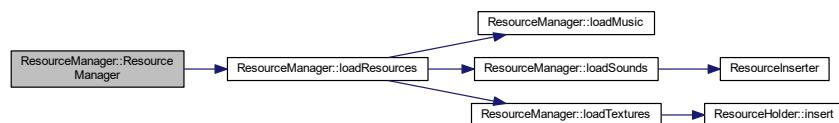
```
ResourceManager::ResourceManager ( )
```

Call [loadResources\(\)](#) which may throw.

#### Warning

Exits with EXIT\_FAILURE on catching an error, because missing resource is a fundamental problem for a game.

Here is the call graph for this function:



### 10.50.3 Member Function Documentation

#### 10.50.3.1 getMusic()

```
auto& ResourceManager::getMusic ( ) [inline]
```

Retrieve a reference to the music holder.

##### Returns

reference to the music holder.

#### 10.50.3.2 getSounds()

```
auto& ResourceManager::getSounds ( ) [inline]
```

Retrieve a reference to the sounds holder.

##### Returns

reference to the sounds holder.

#### 10.50.3.3 getTextures()

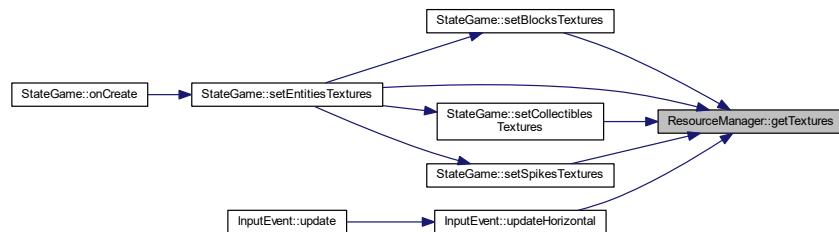
```
auto& ResourceManager::getTextures ( ) [inline]
```

Retrieve a reference to the textures holder.

##### Returns

reference to the textures holder.

Here is the caller graph for this function:



#### 10.50.3.4 loadMusic()

```
void ResourceManager::loadMusic ( ) [private]
```

Loads all music resources. May throw.

Here is the caller graph for this function:



#### 10.50.3.5 loadResources()

```
void ResourceManager::loadResources ( ) [private]
```

Loads all resources, each different resource type uses different thread to handle its type.

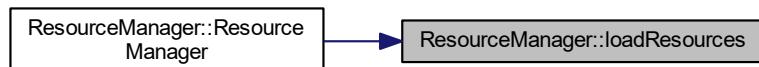
##### Warning

May throw

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.50.3.6 loadSounds()

```
void ResourceManager::loadSounds ( ) [private]
```

Loads all sound resources. May throw.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.50.3.7 loadTextures()

```
void ResourceManager::loadTextures ( ) [private]
```

Loads all texture resources. May throw.

Player

Blocks

Special

Large textures

Corners in the [Options](#) menu

Here is the call graph for this function:



Here is the caller graph for this function:



## 10.50.4 Member Data Documentation

### 10.50.4.1 music

```
ResourceHolder<res::Music, sf::Music> ResourceManager::music {"../resources/Music/"} [private]
```

music container

### 10.50.4.2 sounds

```
ResourceHolder<res::Sound, sf::SoundBuffer> ResourceManager::sounds {"../resources/Sounds/"}
```

[private]

sounds container

### 10.50.4.3 textures

```
ResourceHolder<res::Texture, sf::Texture> ResourceManager::textures {"../resources/Textures/"}
```

[private]

textures container

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

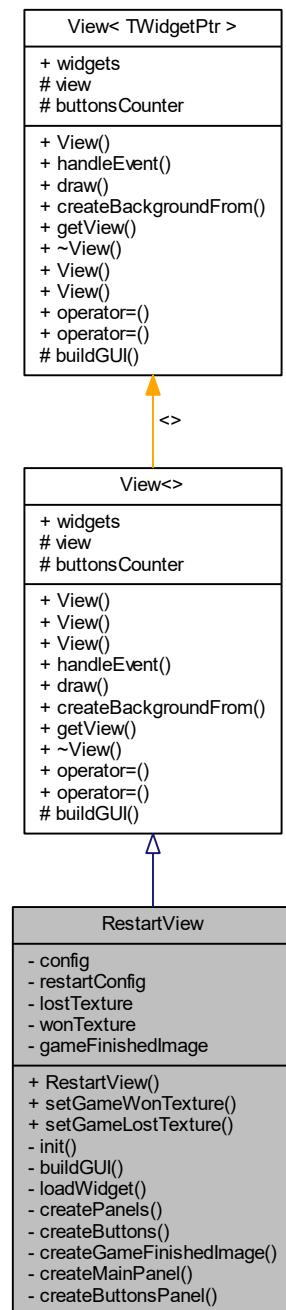
- [ResourceManager.h](#)
- [ResourceManager.cpp](#)

## 10.51 RestartView Class Reference

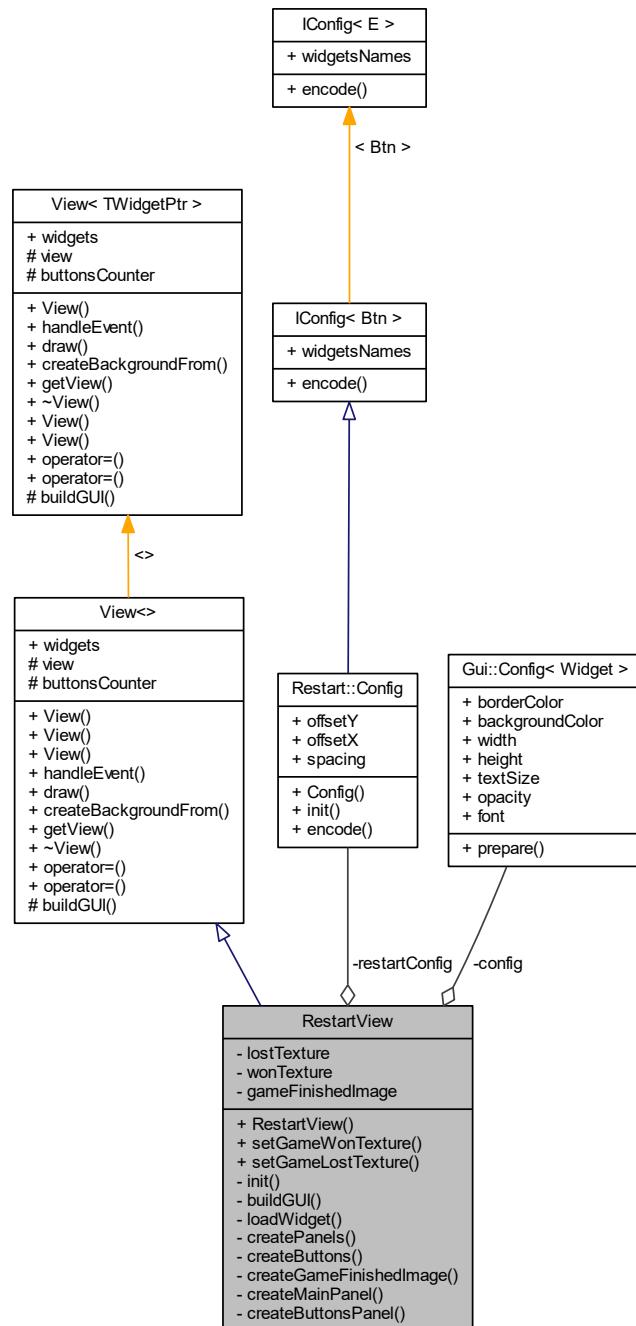
[View](#) class used to draw gui within [StateRestart](#).

```
#include <RestartView.h>
```

Inheritance diagram for RestartView:



Collaboration diagram for RestartView:



## Public Member Functions

- [RestartView \(Window &window, ResourceManager &resourceManager\)](#)  
*Constructs the `RestartView`.*
- [void setGameWonTexture \(\)](#)
- [void setGameLostTexture \(\)](#)

## Private Member Functions

- void `init ()`  
*Initializes the [RestartView](#). Effectively: calls `buildGUI()`.*
- void `buildGUI () override`  
*Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.*
- void `loadWidget (tgui::Widget::Ptr &widget)`  
*Loads a given widget (likely: a button) onto the correct settings and position. @widget to be loaded.*
- void `createPanels ()`  
*Creates all panels and adds them to the #gui.*
- void `createButtons ()`  
*Creates all buttons and adds them to the #gui.*
- void `createGameFinishedImage ()`  
*Creates the gameFinishedImage which is either a `lostTexture` or `wonTexture`.*
- void `createMainPanel ()`  
*Creates the main panel and adds it to the #gui.*
- void `createButtonsPanel ()`  
*Creates the buttons panel and adds it to the #gui.*

## Private Attributes

- `Gui::Config config`  
*Configuration settings common to all [View](#) classes.*
- `Restart::Config restartConfig`  
*Configuration settings specific to [RestartView](#).*
- `const sf::Texture & lostTexture`
- `const sf::Texture & wonTexture`
- `tgui::Picture::Ptr gameFinishedImage`

## Additional Inherited Members

### 10.51.1 Detailed Description

[View](#) class used to draw gui within [StateRestart](#).

### 10.51.2 Constructor & Destructor Documentation

#### 10.51.2.1 `RestartView()`

```
RestartView::RestartView (
    Window & window,
    ResourceManager & resourceManager ) [explicit]
```

Constructs the [RestartView](#).

## Parameters

|                              |  |
|------------------------------|--|
| <code>Window</code>          | that is bound to <code>View</code> base. |
| <code>resourceManager</code> | containing required textures references. |

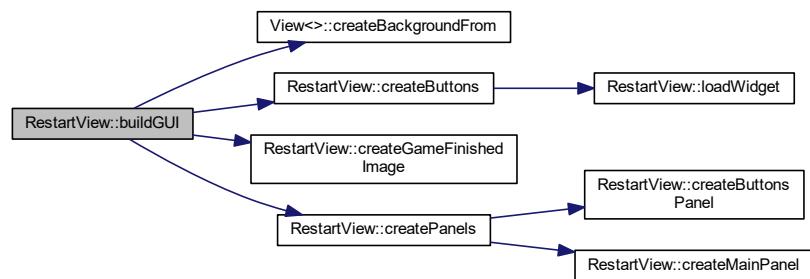
**10.51.3 Member Function Documentation****10.51.3.1 buildGUI()**

```
void RestartView::buildGUI ( ) [override], [private], [virtual]
```

Creates all necessary GUI widgets, sets them correct settings, and add them to #gui.

Implements `View<>`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.51.3.2 `createButtons()`

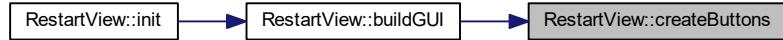
```
void RestartView::createButtons ( ) [private]
```

Creates all buttons and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.51.3.3 `createButtonsPanel()`

```
void RestartView::createButtonsPanel ( ) [private]
```

Creates the buttons panel and adds it to the #gui.

Here is the caller graph for this function:



#### 10.51.3.4 createGameFinishedImage()

```
void RestartView::createGameFinishedImage ( ) [private]
```

Creates the gameFinishedImage which is either a [lostTexture](#) or [wonTexture](#).

##### Note

There is no dynamic memory allocation. The pointers bind to references that already existed.

Here is the caller graph for this function:



#### 10.51.3.5 createMainPanel()

```
void RestartView::createMainPanel ( ) [private]
```

Creates the main panel and adds it to the #gui.

Here is the caller graph for this function:

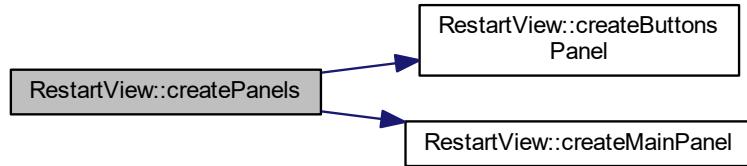


#### 10.51.3.6 createPanels()

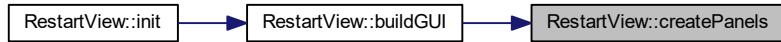
```
void RestartView::createPanels ( ) [private]
```

Creates all panels and adds them to the #gui.

Here is the call graph for this function:



Here is the caller graph for this function:

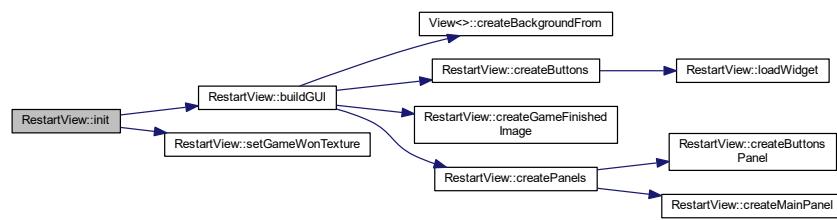


#### 10.51.3.7 `init()`

```
void RestartView::init ( ) [private]
```

Initializes the `RestartView`. Effectively: calls [buildGUI\(\)](#).

Here is the call graph for this function:



### 10.51.3.8 loadWidget()

```
void RestartView::loadWidget (
    tgui::Widget::Ptr & widget ) [private]
```

Loads a given widget (likely: a button) onto the correct settings and position. @widget to be loaded.

#### Note

Can be used in a loop.

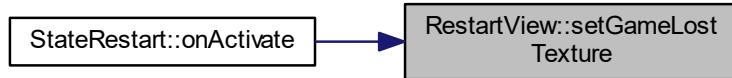
todo: use Restart::Configs without instance of restartConfigHere is the caller graph for this function:



### 10.51.3.9 setGameLostTexture()

```
void RestartView::setGameLostTexture ( )
```

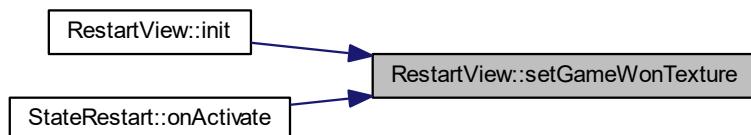
Set [gameFinishedImage](#) to a [lostTexture](#). Here is the caller graph for this function:



### 10.51.3.10 setGameWonTexture()

```
void RestartView::setGameWonTexture ( )
```

Set the [gameFinishedImage](#) to a [wonTexture](#). Here is the caller graph for this function:



## 10.51.4 Member Data Documentation

### 10.51.4.1 config

```
Gui::Config RestartView::config [private]
```

Configuration settings common to all [View](#) classes.

### 10.51.4.2 gameFinishedImage

```
tgui::Picture::Ptr RestartView::gameFinishedImage [private]
```

### 10.51.4.3 lostTexture

```
const sf::Texture& RestartView::lostTexture [private]
```

### 10.51.4.4 restartConfig

```
Restart::Config RestartView::restartConfig [private]
```

Configuration settings specific to [RestartView](#).

### 10.51.4.5 wonTexture

```
const sf::Texture& RestartView::wonTexture [private]
```

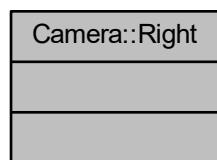
The documentation for this class was generated from the following files:

- [RestartView.h](#)
- [RestartView.cpp](#)

## 10.52 Camera::Right Struct Reference

Empty struct denoting previously solved on other axis was [Right](#) collision.

Collaboration diagram for Camera::Right:



### 10.52.1 Detailed Description

Empty struct denoting previously solved on other axis was [Right](#) collision.

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

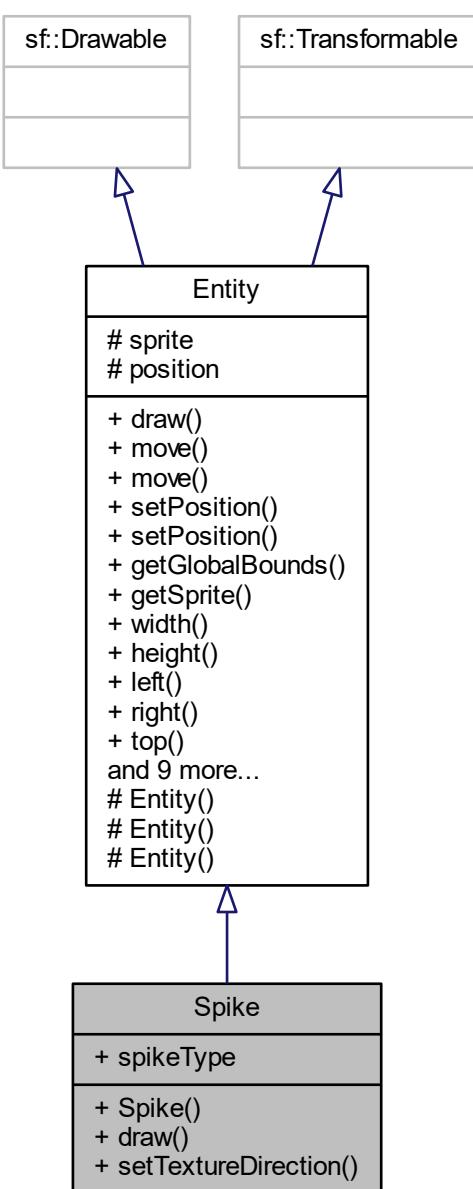
- [Camera.h](#)

## 10.53 Spike Class Reference

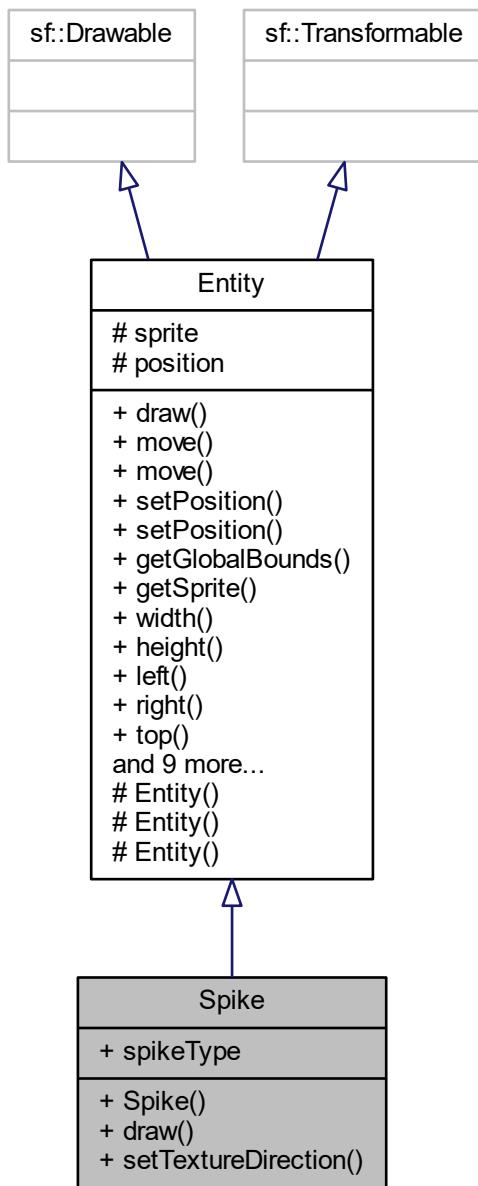
Used to represent the spikes. Intersecting spike kills the [Player](#).

```
#include <Spike.h>
```

Inheritance diagram for Spike:



Collaboration diagram for Spike:



## Public Member Functions

- **Spike (Obj::Entity spikeType, sf::Vector2f position)**  
*Constructs a Spike.*
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override  
*Draw method used to render `sprite` onto the window.*
- void **setTextureDirection** ()  
*Sets the spike `sprite` texture direction.*

## Public Attributes

- Obj::Entity spikeType

## Additional Inherited Members

### 10.53.1 Detailed Description

Used to represent the spikes. Intersecting spike kills the Player.

There are four different possible spike views:

- left-oriented spike
- right-oriented spike
- top-oriented spike
- bot-oriented spike

#### Note

All of the above spike re-use the very same texture.

A concrete Entity class overriding the draw method.

### 10.53.2 Constructor & Destructor Documentation

#### 10.53.2.1 Spike()

```
Spike::Spike (Obj::Entity spikeType, sf::Vector2f position) [explicit]
```

Constructs a Spike.

##### Parameters

|                  |                                |
|------------------|--------------------------------|
| <i>spikeType</i> | the facing direction of spike. |
| <i>position</i>  | initial position.              |

### 10.53.3 Member Function Documentation

### 10.53.3.1 draw()

```
void Spike::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

Draw method used to render [sprite](#) onto the window.

#### Parameters

|               |  |
|---------------|--|
| <i>target</i> | <a href="#">Window</a> for the sprite to be rendered onto. |
| <i>states</i> | Optional render states                                     |

Overrides [Entity](#) draw method.

#### Note

Passing states is optional.

### 10.53.3.2 setTextureDirection()

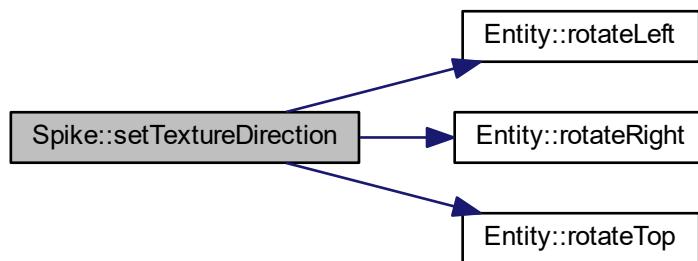
```
void Spike::setTextureDirection ( )
```

Sets the spike [sprite](#) texture direction.

Four possible directions:

- SpikeLeft
- SpikeRight
- SpikeTop
- SpikeBot

The direction is set by rotating [sprite](#). Here is the call graph for this function:



## 10.53.4 Member Data Documentation

### 10.53.4.1 spikeType

`Obj::Entity Spike::spikeType`

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

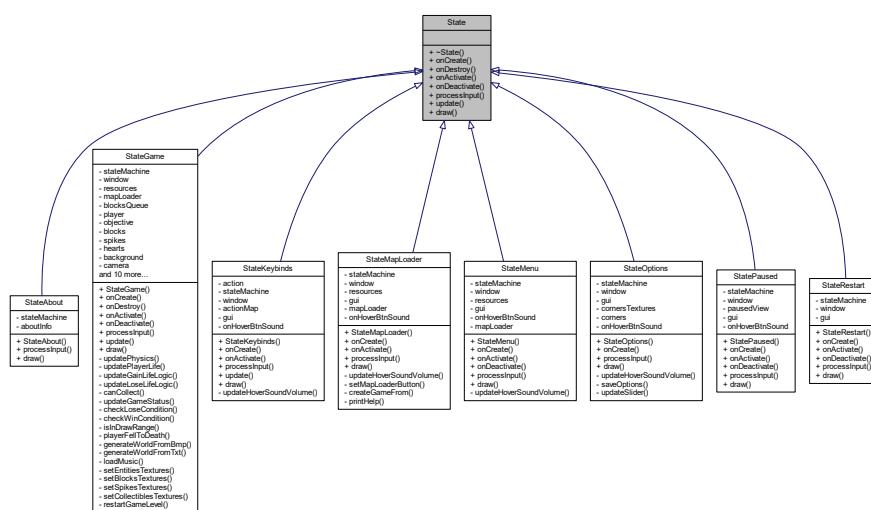
- [Spike.h](#)
- [Spike.cpp](#)

## 10.54 State Class Reference

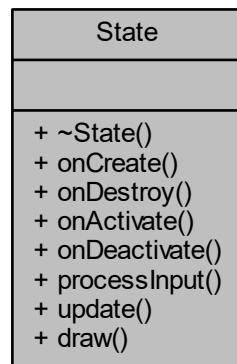
< Pass-through

```
#include <State.h>
```

Inheritance diagram for State:



Collaboration diagram for State:



## Public Member Functions

- virtual `~State ()=default`
- virtual void `onCreate ()`
- virtual void `onDestroy ()`
- virtual void `onActivate ()`
- virtual void `onDeactivate ()`
- virtual void `processInput ()=0`
- virtual void `update (float)`
- virtual void `draw (Window &)=0`

### 10.54.1 Detailed Description

< Pass-through

An abstract class working as a polymorphic interface for Concrete States.

All States have to override this class and implement pure virtual `processInput()` and `draw(Window&)` methods.

Other methods are empty definition body pass-through method - the user can override only the method he needs in his Concrete `State`.

### 10.54.2 Constructor & Destructor Documentation

#### 10.54.2.1 ~State()

```
virtual State::~State ( ) [virtual], [default]
```

Enable default destructor.

### 10.54.3 Member Function Documentation

#### 10.54.3.1 draw()

```
virtual void State::draw (
    Window & ) [pure virtual]
```

At the highest call level, invoked by [Game](#) class.

*Perfect* interface: meaningfully overriden in all 8 States.

Implemented in [StateGame](#), [StateAbout](#), [StateMenu](#), [StateKeybinds](#), [StatePaused](#), [StateMapLoader](#), [StateOptions](#), and [StateRestart](#).

#### 10.54.3.2 onActivate()

```
virtual void State::onActivate ( ) [inline], [virtual]
```

Empty pass-through method called by [StateMachine](#) when [State](#) was transitioned to.

Reimplemented in [StateGame](#), [StateMenu](#), [StateMapLoader](#), [StateKeybinds](#), [StatePaused](#), and [StateRestart](#).

#### 10.54.3.3 onCreate()

```
virtual void State::onCreate ( ) [inline], [virtual]
```

Empty pass-through method called by [StateMachine](#) when inserting new [State](#).

Reimplemented in [StateGame](#), [StateMenu](#), [StateKeybinds](#), [StateOptions](#), [StatePaused](#), [StateMapLoader](#), and [StateRestart](#).

#### 10.54.3.4 onDeactivate()

```
virtual void State::onDeactivate ( ) [inline], [virtual]
```

Empty pass-through method called by [StateMachine](#) when [State](#) was transitioned from.

Reimplemented in [StateGame](#), [StateMenu](#), [StatePaused](#), and [StateRestart](#).

### 10.54.3.5 onDestroy()

```
virtual void State::onDestroy ( ) [inline], [virtual]
```

Empty pass-through method called by [StateMachine](#) when erasing existing [State](#).

Reimplemented in [StateGame](#).

### 10.54.3.6 processInput()

```
virtual void State::processInput ( ) [pure virtual]
```

At the highest call level, invoked by [Game](#) class.

*Perfect* interface: meaningfully overriden in all 8 States.

Implemented in [StateGame](#), [StateMenu](#), [StatePaused](#), [StateKeybinds](#), [StateMapLoader](#), [StateOptions](#), [StateRestart](#), and [StateAbout](#).

### 10.54.3.7 update()

```
virtual void State::update (
    float ) [inline], [virtual]
```

At the highest call level, invoked by [Game](#) class.

#### Parameters

|              |   |
|--------------|---|
| <i>float</i> | Likely a time-elapsed since previous frame. |
|--------------|---|

Reimplemented in [StateGame](#), and [StateKeybinds](#).

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

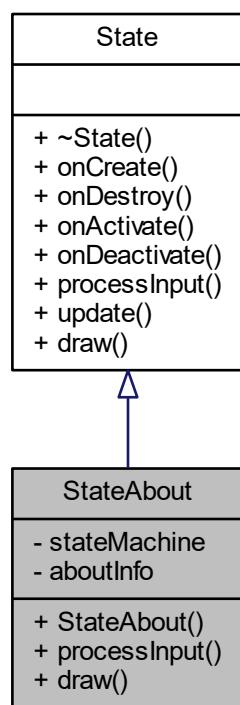
- [State.h](#)

## 10.55 StateAbout Class Reference

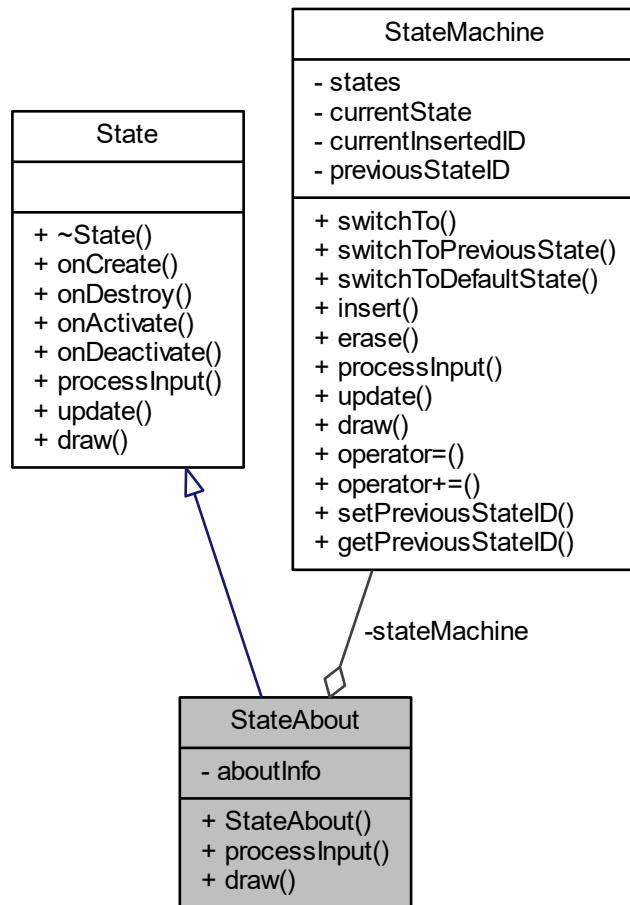
Encapsulates the logic of about menu within one [StateAbout](#) class. Displays the about texture.

```
#include <StateAbout.h>
```

Inheritance diagram for StateAbout:



Collaboration diagram for StateAbout:



## Public Member Functions

- `StateAbout (StateMachine &stateMachine, ResourceManager &resources)`  
*Create the StateAbout.*
- `void processInput () override`  
*Processes all inputs received by #gui.*
- `void draw (Window &window) override`  
*Draws the #gui onto the screen.*

## Private Attributes

- `StateMachine & stateMachine`
- `sf::Sprite aboutInfo`

### 10.55.1 Detailed Description

Encapsulates the logic of about menu within one [StateAbout](#) class. Displays the about texture.

#### Note

The only possible transition from [StateAbout](#) is:

- [StateMenu](#)

Inherits from [State](#) base. Created inside [Game](#) and Stored inside [StateMachine](#). The ID of [StateMenu](#) can be retrieved from `state::menuID`.

### 10.55.2 Constructor & Destructor Documentation

#### 10.55.2.1 [StateAbout\(\)](#)

```
StateAbout::StateAbout (
    StateMachine & stateMachine,
    ResourceManager & resources )
```

Create the [StateAbout](#).

#### Parameters

|                              |                                 |
|------------------------------|---------------------------------|
| <code>stateMachine</code>    | <input type="button" value=""/> |
| <code>resourceManager</code> | <input type="button" value=""/> |

Constructor invoked from [Game](#).

### 10.55.3 Member Function Documentation

#### 10.55.3.1 [draw\(\)](#)

```
void StateAbout::draw (
    Window & window ) [override], [virtual]
```

Draws the #gui onto the screen.

Implements [State](#).

Here is the call graph for this function:



### 10.55.3.2 processInput()

```
void StateAbout::processInput( ) [override], [virtual]
```

Processes all inputs received by #gui.

Implements [State](#).

Here is the call graph for this function:



## 10.55.4 Member Data Documentation

### 10.55.4.1 aboutInfo

```
sf::Sprite StateAbout::aboutInfo [private]
```

### 10.55.4.2 stateMachine

```
StateMachine& StateAbout::stateMachine [private]
```

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

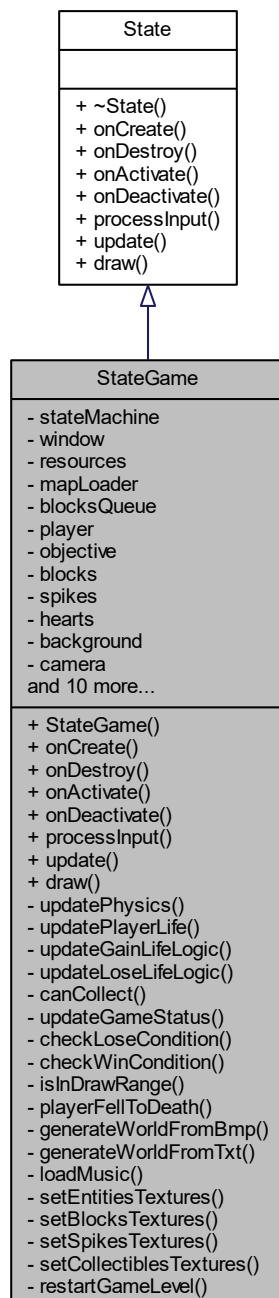
- [StateAbout.h](#)
- [StateAbout.cpp](#)

## 10.56 StateGame Class Reference

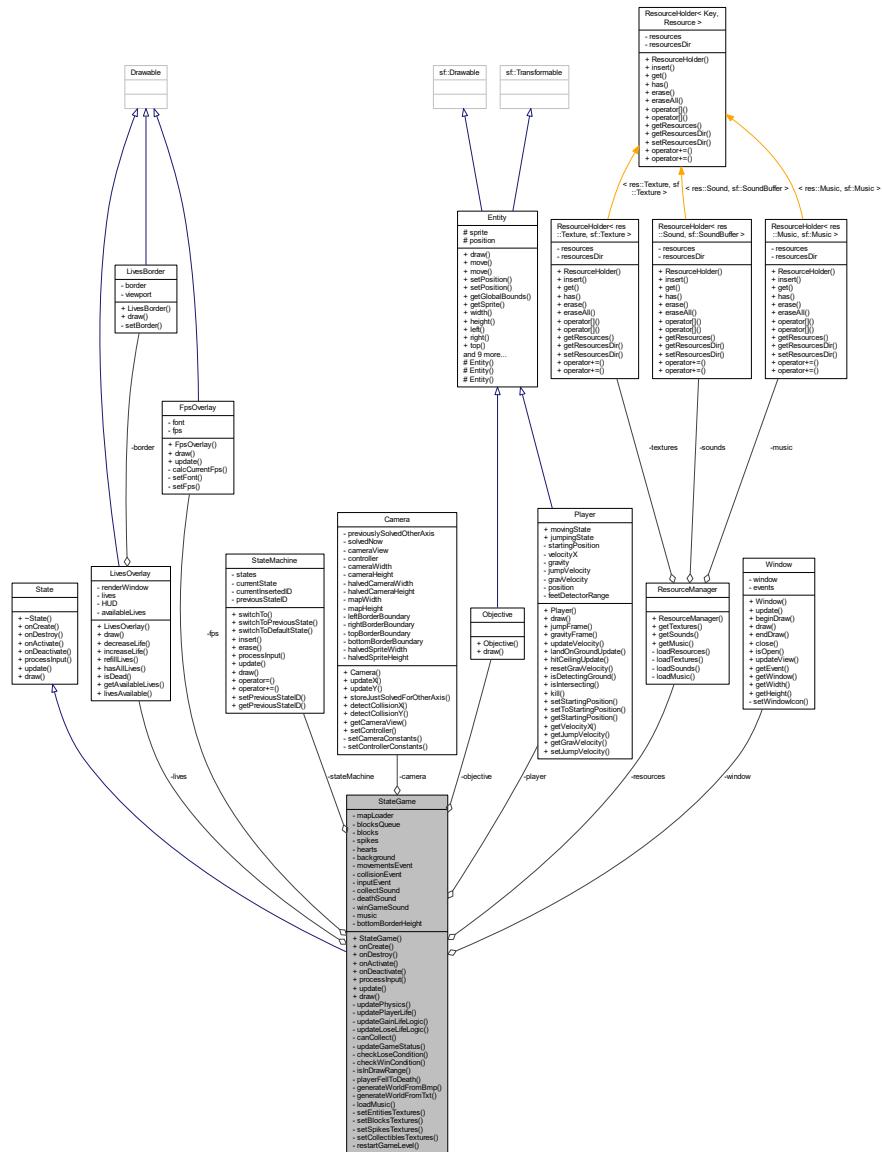
Encapsulates the main game logic within one [StateGame](#) class.

```
#include <StateGame.h>
```

Inheritance diagram for StateGame:



## Collaboration diagram for StateGame:



# Public Member Functions

- StateGame (StateMachine &stateMachine, Window &window, ResourceManager &resources, std::variant< MapLoader< Bmp >, MapLoader< Txt >> &mapLoader)

## Create the *StateGame*.

- void `onCreate()` override

*Called when the State is created. Generates the World by visiting a variant.*

- void `onDestroy()` override

Called when the [StateGame](#) is destroyed, which is either upon [Window](#) close (end [Game](#)); or when new map is loaded.

- void `onActivate()` override

Called when [StateMenu](#) is activated. Updates activation [StateGame](#) logic.

- void `onDeactivate()` override

Called when *StateMenu* is left. Pause the music and store previous state as *state::gameID*.

- void `processInput () override`  
*Call InputEvent->`update()` method, and process activation StatePause keypress input.*
- void `update (float dt) override`  
*Performs a frame cycle of a main game loop.*
- void `draw (Window &window) override`  
*Draws the game onto the screen.*

## Private Member Functions

- void `updatePhysics (float dt)`  
*Update: MovementEvent, CollisionEvent and Camera on axis X, then on axis Y.*
- void `updatePlayerLife ()`  
*Update the status of player life (lose or gain life).*
- void `updateGainLifeLogic ()`  
*Player collects heart if he touches a visible heart and Player is missing lives.*
- void `updateLoseLifeLogic ()`  
*Player loses life if he touches the Spike.*
- bool `canCollect (const HeartCollectible &heart)`  
*Checks if a given heart can be collected.*
- void `updateGameStatus ()`  
*Update the game status: checks if game was won or lost in current frame.*
- void `checkLoseCondition ()`  
*Check if current frame was a losing frame. Lose condition: player lost all lives.*
- void `checkWinCondition ()`  
*Check if current frame was a winning frame. Win condition: reach objective.*
- bool `isInDrawRange (const Entity &entity) const`
- bool `playerFellToDeath () const`
- void `generateWorldFromBmp ()`  
*Generate world from a Bmp. Allocate blocks and set them to proper position.*
- void `generateWorldFromTxt ()`  
*Generate world from a Txt. Allocate blocks and set them to proper position.*
- void `loadMusic ()`  
*Load the music.*
- void `setEntitiesTextures ()`  
*Sets the textures of all Entity objects.*
- void `setBlocksTextures ()`  
*Set the textures of Block entities.*
- void `setSpikesTextures ()`  
*Set the spike texture facing correct direction to each Spike entity.*
- void `setCollectiblesTextures ()`  
*Set the collectible texture to all Collectible entities.*
- void `restartGameLevel ()`

## Private Attributes

- `StateMachine & stateMachine`
- `Window & window`
- `ResourceManager & resources`
- `std::variant< MapLoader< Bmp >, MapLoader< Txt >> & mapLoader`
- `std::queue< res::Texture > blocksQueue`
- `Player player`
- `Objective objective`
- `std::vector< std::unique_ptr< Entity > > blocks`
- `std::vector< std::unique_ptr< Spike > > spikes`
- `std::vector< std::unique_ptr< HeartCollectible > > hearts`
- `sf::Sprite background`
- `Camera camera`
- `std::unique_ptr< MovementEvent > movementsEvent`
- `std::unique_ptr< CollisionEvent > collisionEvent`
- `std::unique_ptr< InputEvent > inputEvent`
- `LivesOverlay lives`
- `FpsOverlay fps`
- `sf::Sound collectSound`
- `sf::Sound deathSound`
- `sf::Sound winGameSound`
- `sf::Music music`
- `const int bottomBorderHeight = config::entityHeight * config::blocksCountHeight`

### 10.56.1 Detailed Description

Encapsulates the main game logic within one `StateGame` class.

#### Note

All possible transitions from `StateGame` are:

- `StateOptions`
- `StatePaused`
- `StateMenu`
- `StateRestart`

Inherits from `State` base. Created inside `Game` and Stored inside `StateMachine`. The ID of `StateMenu` can be retrieved from `state::menuID`.

### 10.56.2 Constructor & Destructor Documentation

#### 10.56.2.1 StateGame()

```
StateGame::StateGame (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resources,
    std::variant< MapLoader< Bmp >, MapLoader< Txt >> & mapLoader )
```

Create the `StateGame`.

**Parameters**

|                        |   |
|------------------------|---|
| <i>stateMachine</i>    |   |
| <i>window</i>          |   |
| <i>resourceManager</i> |   |
| <i>mapLoader</i>       | either a <a href="#">Bmp</a> or <a href="#">Txt</a> variant, depending on map type. |

Sets the [camera](#), [background](#) image and default sounds. Constructor invoked from [Game](#), or [StateMapLoader](#) depending if user chose default or custom map.

### 10.56.3 Member Function Documentation

#### 10.56.3.1 canCollect()

```
bool StateGame::canCollect (
    const HeartCollectible & heart ) [private]
```

Checks if a given heart can be collected.

**Parameters**

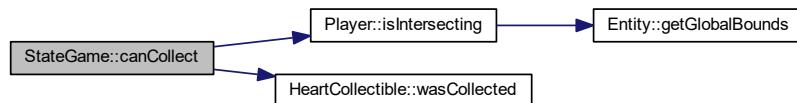
|              |                                      |
|--------------|--------------------------------------|
| <i>heart</i> | checked heart if it can be collected |
|--------------|--------------------------------------|

**Returns**

Can the heart be collected? A heart can be collected if all are satisfied:

- it was not collected before
- [player](#) is intersecting it
- [player](#) has missing life(s)

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.56.3.2 checkLoseCondition()

```
void StateGame::checkLoseCondition () [private]
```

Check if current frame was a losing frame. Lose condition: `player` lost all `lives`.

Here is the caller graph for this function:



### 10.56.3.3 checkWinCondition()

```
void StateGame::checkWinCondition () [private]
```

Check if current frame was a winning frame. Win condition: reach `objective`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.56.3.4 draw()

```
void StateGame::draw (
    Window & window ) [override], [virtual]
```

Draws the game onto the screen.

**Parameters**

|                     |                                    |
|---------------------|------------------------------------|
| <code>window</code> | The window the game is drawn onto. |
|---------------------|------------------------------------|

**Note**

The following entities are drawn:

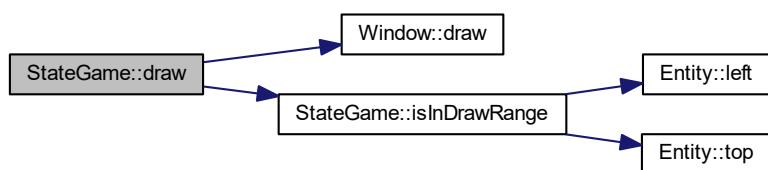
- 1. [background](#)
- 2. [blocks](#) Entities (collidables)
- 3. [spikes](#) Entities (killing entities)
- 4. [hearts](#) Entities (collectibles)
- 5. [objective](#), an Entity
- 6. [player](#), a controlling Entity
- 7. [lives](#), an overlay Entity HUD
- 8. [fps](#), an overlay Entity HUD

**Note**

Points **2**, **3**, **4**: only Entities in a drawable range are rendered. Point **4**: collected Entities are no longer displayed. Point **8**: may be disabled/enabled in [StateOptions](#) during run-time.

Implements [State](#).

Here is the call graph for this function:

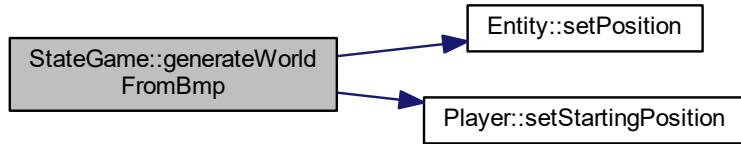


### 10.56.3.5 generateWorldFromBmp()

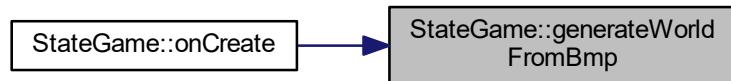
```
void StateGame::generateWorldFromBmp ( ) [private]
```

Generate world from a [Bmp](#). Allocate blocks and set them to proper position.

Here is the call graph for this function:



Here is the caller graph for this function:

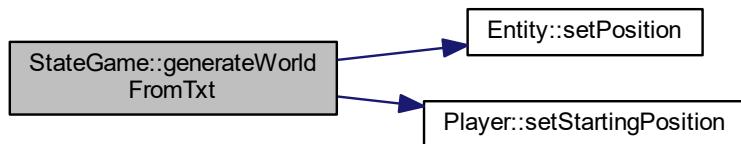


### 10.56.3.6 generateWorldFromTxt()

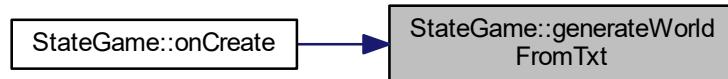
```
void StateGame::generateWorldFromTxt ( ) [private]
```

Generate world from a [Txt](#). Allocate blocks and set them to proper position.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.56.3.7 isInDrawRange()

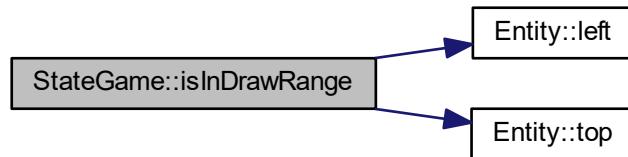
```
bool StateGame::isInDrawRange (
    const Entity & entity ) const [private]
```

Checks if [player](#) is in draw range of #entity. Draw range is a range that fits on screen.

Returns

Is entity within view range?

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.56.3.8 loadMusic()

```
void StateGame::loadMusic ( ) [private]
```

Load the [music](#).

### 10.56.3.9 onActivate()

```
void StateGame::onActivate ( ) [override], [virtual]
```

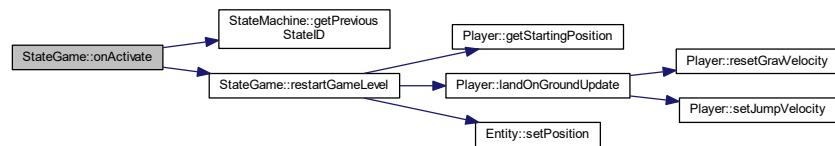
Called when [StateMenu](#) is activated. Updates activation [StateGame](#) logic.

[StateGame](#) activation logic:

- Because player can re-enter [StateGame](#) from PausedState, checks if [getPreviousStateID\(\)](#) is [state::restartID](#). Only in such a case, a [restartGameLevel\(\)](#) is performed.
- Plays the [music](#)
- Updates the sound volume.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.56.3.10 onCreate()

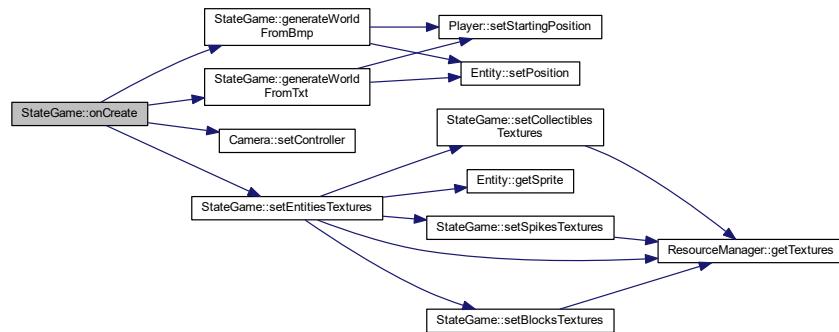
```
void StateGame::onCreate ( ) [override], [virtual]
```

Called when the [State](#) is created. Generates the World by visiting a variant.

- 1. Generate the World using static polymorphism.
- 2. Set the World textures.
- 3. Set [Camera](#) controller.
- 4. Create [Event](#) / [PhysicsEvent](#) classes.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.56.3.11 `onDeactivate()`

```
void StateGame::onDeactivate( ) [override], [virtual]
```

Called when [StateMenu](#) is left. Pause the music and store previous state as [state::gameID](#).

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.56.3.12 `onDestroy()`

```
void StateGame::onDestroy( ) [override], [virtual]
```

Called when the [StateGame](#) is destroyed, which is either upon [Window](#) close (end [Game](#)); or when new map is loaded.

Reimplemented from [State](#).

### 10.56.3.13 playerFellToDeath()

```
bool StateGame::playerFellToDeath () const [private]
```

Checks if [player](#) fell to death by falling below the map edge.

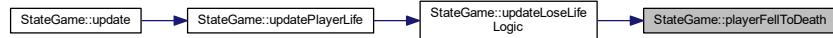
#### Returns

Did player fell to death?

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.56.3.14 processInput()

```
void StateGame::processInput () [override], [virtual]
```

Call [InputEvent->update\(\)](#) method, and process activation StatePause keypress input.

#### Note

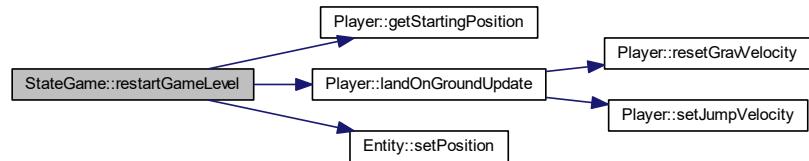
Also, process keypresses events trying to switch to a different [State](#).

Implements [State](#).

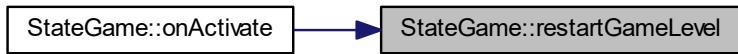
### 10.56.3.15 restartGameLevel()

```
void StateGame::restartGameLevel ( ) [private]
```

Refill [Player lives](#) and respawn [hearts](#). Set player to starting position. Respawn collectiblesHere is the call graph for this function:



Here is the caller graph for this function:



### 10.56.3.16 setBlocksTextures()

```
void StateGame::setBlocksTextures ( ) [private]
```

Set the textures of [Block](#) entities.

Here is the call graph for this function:



Here is the caller graph for this function:

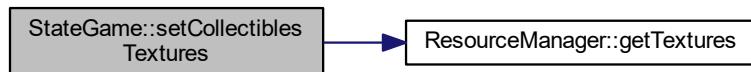


### 10.56.3.17 setCollectiblesTextures()

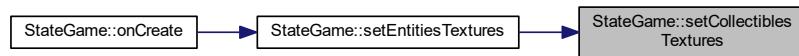
```
void StateGame::setCollectiblesTextures ( ) [private]
```

Set the collectible texture to all Collectible entities.

Here is the call graph for this function:



Here is the caller graph for this function:

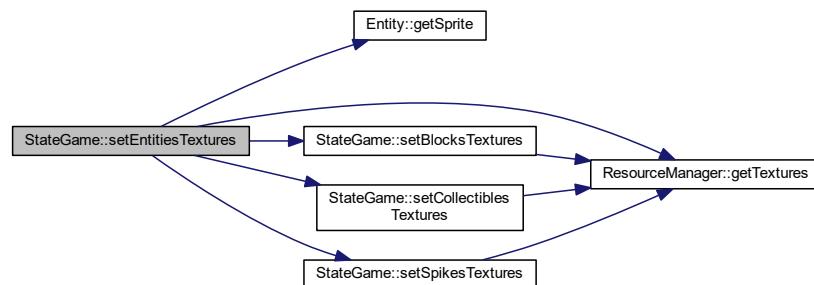


### 10.56.3.18 setEntitiesTextures()

```
void StateGame::setEntitiesTextures ( ) [private]
```

Sets the textures of all [Entity](#) objects.

todo: direct setterHere is the call graph for this function:



Here is the caller graph for this function:



#### 10.56.3.19 setSpikesTextures()

```
void StateGame::setSpikesTextures ( ) [private]
```

Set the spike texture facing correct direction to each [Spike](#) entity.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.56.3.20 update()

```
void StateGame::update (
    float dt ) [override], [virtual]
```

Performs a frame cycle of a main game loop.

##### Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>dt</i> | Time elapsed since previous frame. |
|-----------|------------------------------------|

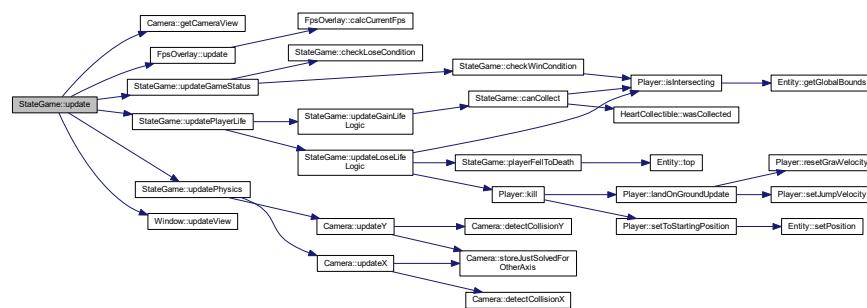
The following is called:

- updatePhysics(dt);
- window.updateView(camera.getCameraView());
- fps.update(dt);
- [updatePlayerLife\(\)](#);
- [updateGameStatus\(\)](#);

The physics is updated in an axis independent way: first the X axis, then the Y axis. This greatly simplified collision logic.

Reimplemented from [State](#).

Here is the call graph for this function:

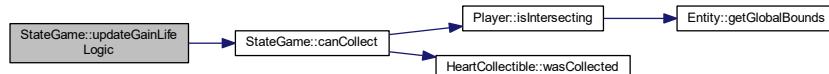


### 10.56.3.21 updateGainLifeLogic()

```
void StateGame::updateGainLifeLogic () [private]
```

[Player](#) collects heart if he touches a visible heart and [Player](#) is missing lives.

Here is the call graph for this function:



Here is the caller graph for this function:

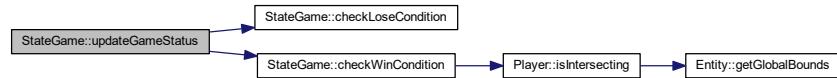


### 10.56.3.22 updateGameStatus()

```
void StateGame::updateGameStatus ( ) [private]
```

Update the game status: checks if game was won or lost in current frame.

Here is the call graph for this function:



Here is the caller graph for this function:

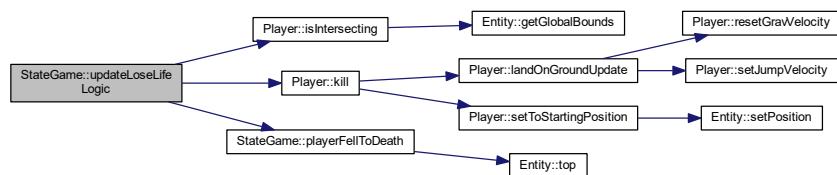


### 10.56.3.23 updateLoseLifeLogic()

```
void StateGame::updateLoseLifeLogic ( ) [private]
```

[Player](#) loses life if he touches the [Spike](#).

Here is the call graph for this function:



Here is the caller graph for this function:



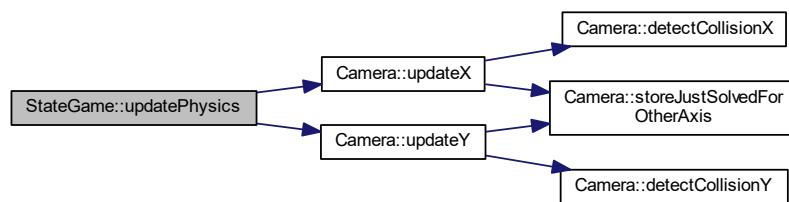
### 10.56.3.24 updatePhysics()

```
void StateGame::updatePhysics (
    float dt ) [private]
```

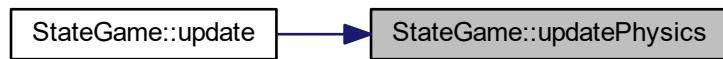
Update: [MovementEvent](#), [CollisionEvent](#) and [Camera](#) on axis X, then on axis Y.

Called every frame: movementsEvent->updateAxisY(dt); collisionEvent->updateAxisY(dt); camera.updateY();  
 movementsEvent->updateAxisY(dt); collisionEvent->updateAxisY(dt); camera.updateY();

\ Here is the call graph for this function:



Here is the caller graph for this function:

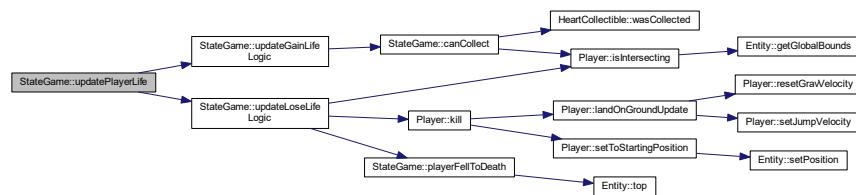


### 10.56.3.25 updatePlayerLife()

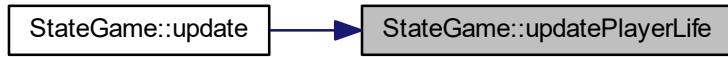
```
void StateGame::updatePlayerLife ( ) [private]
```

Update the status of player life (lose or gain life).

Here is the call graph for this function:



Here is the caller graph for this function:



## 10.56.4 Member Data Documentation

### 10.56.4.1 background

```
sf::Sprite StateGame::background [private]
```

### 10.56.4.2 blocks

```
std::vector<std::unique_ptr<Entity>> StateGame::blocks [private]
```

### 10.56.4.3 blocksQueue

```
std::queue<res::Texture> StateGame::blocksQueue [private]
```

### 10.56.4.4 bottomBorderHeight

```
const int StateGame::bottomBorderHeight = config::entityHeight * config::blocksCountHeight  
[private]
```

### 10.56.4.5 camera

```
Camera StateGame::camera [private]
```

**10.56.4.6 collectSound**

```
sf::Sound StateGame::collectSound [private]
```

todo: Simplifiable to e.g. SoundPlayer class

**10.56.4.7 collisionEvent**

```
std::unique_ptr<CollisionEvent> StateGame::collisionEvent [private]
```

**10.56.4.8 deathSound**

```
sf::Sound StateGame::deathSound [private]
```

**10.56.4.9 fps**

```
FpsOverlay StateGame::fps [private]
```

**10.56.4.10 hearts**

```
std::vector<std::unique_ptr<HeartCollectible>> StateGame::hearts [private]
```

**10.56.4.11 inputEvent**

```
std::unique_ptr<InputEvent> StateGame::inputEvent [private]
```

**10.56.4.12 lives**

```
LivesOverlay StateGame::lives [private]
```

todo: Simplifiable to Overlay class

**10.56.4.13 mapLoader**

```
std::variant<MapLoader<Bmp>, MapLoader<Txt>>& StateGame::mapLoader [private]
```

todo: Simplifiable to WorldLoader class

---

#### 10.56.4.14 movementsEvent

```
std::unique_ptr<MovementEvent> StateGame::movementsEvent [private]
```

todo: Simplifiable to e.g. EventsList class

#### 10.56.4.15 music

```
sf::Music StateGame::music [private]
```

#### 10.56.4.16 objective

```
Objective StateGame::objective [private]
```

#### 10.56.4.17 player

```
Player StateGame::player [private]
```

todo: Simplifiable to World class

#### 10.56.4.18 resources

```
ResourceManager& StateGame::resources [private]
```

#### 10.56.4.19 spikes

```
std::vector<std::unique_ptr<Spike>> StateGame::spikes [private]
```

#### 10.56.4.20 stateMachine

```
StateMachine& StateGame::stateMachine [private]
```

#### 10.56.4.21 window

```
Window& StateGame::window [private]
```

#### 10.56.4.22 winGameSound

```
sf::Sound StateGame::winGameSound [private]
```

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

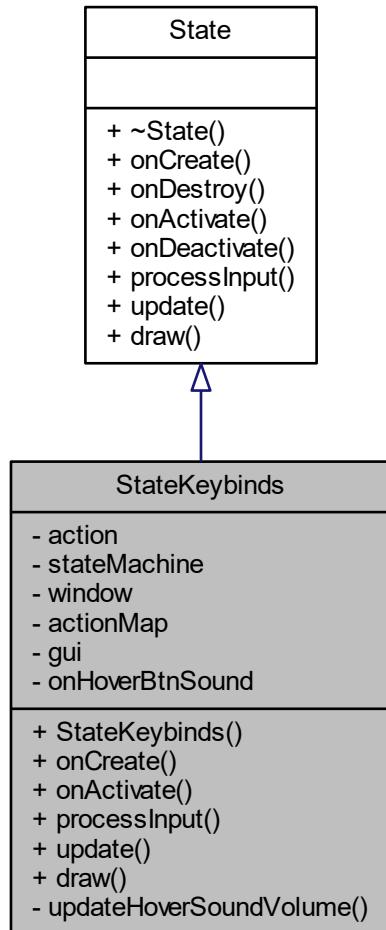
- [StateGame.h](#)
- [StateGame.cpp](#)

## 10.57 StateKeybinds Class Reference

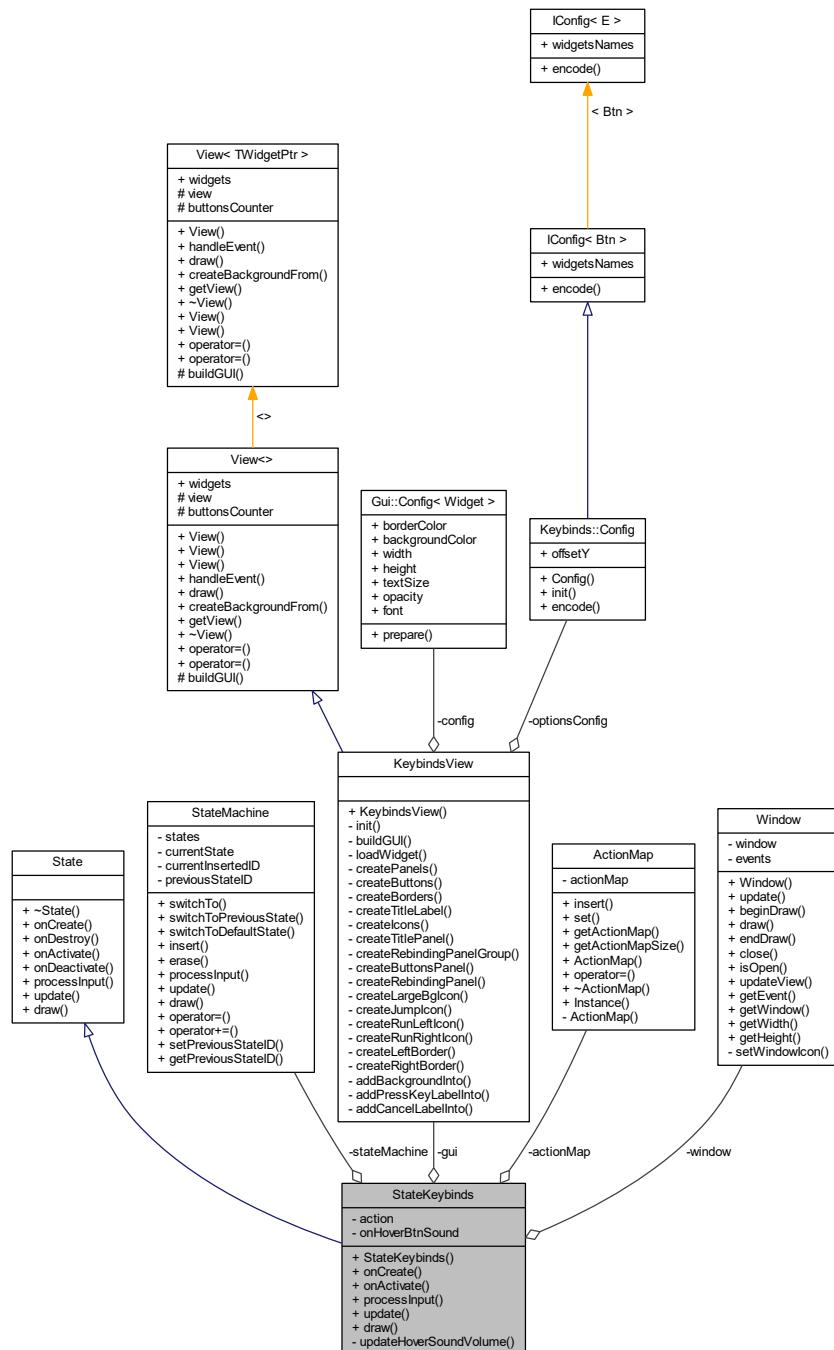
Encapsulates the logic of rebinding menu within one [StateKeybinds](#) class.

```
#include <StateKeybinds.h>
```

Inheritance diagram for StateKeybinds:



Collaboration diagram for StateKeybinds:



## Public Member Functions

- **StateKeybinds (StateMachine &stateMachine, Window &window, ResourceManager &resourceManager)**  
*Create the StateKeybinds.*
- **void onCreate () override**  
*Called when the StateKeybinds is created. Connects signals to gui widgets.*
- **void onActivate () override**

*Called when [StateKeybinds](#) is activated. Updates the sound volume.*

- void [processInput \(\)](#) override

*Processes all inputs received by [gui](#).*
- void [update \(float dt\)](#) override
- void [draw \(Window &\)](#) override

*Draws the [gui](#) onto the screen.*

## Private Member Functions

- void [updateHoverSoundVolume \(\)](#)

*Updates the buttons hover volume.*

## Private Attributes

- [Action action](#)
- [StateMachine & stateMachine](#)
- [Window & window](#)
- [ActionMap & actionMap = ActionMap::Instance\(\)](#)
- [KeybindsView gui](#)
- [sf::Sound onHoverBtnSound](#)

### 10.57.1 Detailed Description

Encapsulates the logic of rebinding menu within one [StateKeybinds](#) class.

#### Note

All possible transitions from [StateMenu](#) are:

- [StateOptions](#)

Inherits from [State](#) base. Created inside [Game](#) and Stored inside [StateMachine](#). The ID of [StateMenu](#) can be retrieved from [state::menuID](#).

### 10.57.2 Constructor & Destructor Documentation

#### 10.57.2.1 StateKeybinds()

```
StateKeybinds::StateKeybinds (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resourceManager )
```

Create the [StateKeybinds](#).

**Parameters**

|                        |  |
|------------------------|--|
| <i>stateMachine</i>    |  |
| <i>window</i>          |  |
| <i>resourceManager</i> |  |

Sets the default #onHoverSound. Constructor invoked from [Game](#).

### 10.57.3 Member Function Documentation

#### 10.57.3.1 draw()

```
void StateKeybinds::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



#### 10.57.3.2 onActivate()

```
void StateKeybinds::onActivate ( ) [override], [virtual]
```

Called when [StateKeybinds](#) is activated. Updates the sound volume.

Reimplemented from [State](#).

Here is the call graph for this function:



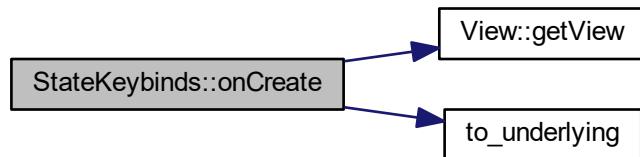
### 10.57.3.3 onCreate()

```
void StateKeybinds::onCreate ( ) [override], [virtual]
```

Called when the [StateKeybinds](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.57.3.4 processInput()

```
void StateKeybinds::processInput ( ) [override], [virtual]
```

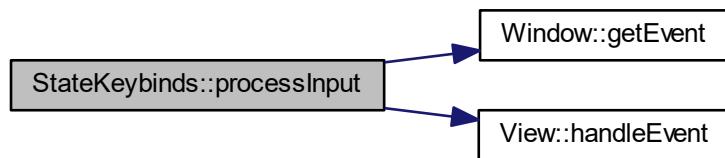
Processes all inputs received by [gui](#).

Note

Also, process keypresses events trying to go back to [StateOptions](#).

Implements [State](#).

Here is the call graph for this function:



### 10.57.3.5 update()

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

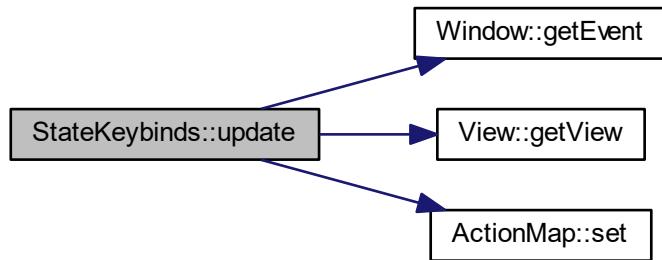
At the highest call level, invoked by [Game](#) class.

**Parameters**

|              |   |
|--------------|---|
| <i>float</i> | Likely a time elapsed since previous frame. |
|--------------|---|

Reimplemented from [State](#).

Here is the call graph for this function:



#### 10.57.3.6 updateHoverSoundVolume()

```
void StateKeybinds::updateHoverSoundVolume ( ) [private]
```

Updates the buttons hover volume.

Here is the caller graph for this function:



#### 10.57.4 Member Data Documentation

#### 10.57.4.1 action

```
Action StateKeybinds::action [private]
```

#### 10.57.4.2 actionMap

```
ActionMap& StateKeybinds::actionMap = ActionMap::Instance() [private]
```

#### 10.57.4.3 gui

```
KeybindsView StateKeybinds::gui [private]
```

#### 10.57.4.4 onHoverBtnSound

```
sf::Sound StateKeybinds::onHoverBtnSound [private]
```

#### 10.57.4.5 stateMachine

```
StateMachine& StateKeybinds::stateMachine [private]
```

#### 10.57.4.6 window

```
Window& StateKeybinds::window [private]
```

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

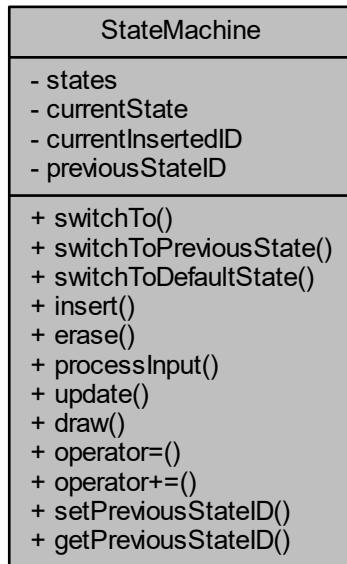
- [StateKeybinds.h](#)
- [StateKeybinds.cpp](#)

## 10.58 StateMachine Class Reference

Container for all game States and holds current state pointer. Updates [State](#) logic every frame.

```
#include <StateMachine.h>
```

Collaboration diagram for StateMachine:



### Public Member Functions

- void [switchTo](#) (int stateID)
- void [switchToPreviousState](#) ()
- void [switchToDefaultState](#) ()
- int [insert](#) (const std::shared\_ptr<[State](#)> &state)
- void [erase](#) (int stateID)
- void [processInput](#) () const
- void [update](#) (float dt) const
- void [draw](#) ([Window](#) &window) const
- [StateMachine](#) & [operator=](#) (int stateID)
- [StateMachine](#) & [operator+=](#) (const std::shared\_ptr<[State](#)> &s)
- void [setPreviousStateID](#) (int stateID)
- int [getPreviousStateID](#) () const

### Private Attributes

- std::unordered\_map< int, std::shared\_ptr<[State](#)> > [states](#) {}  
*Container for all states. Key: stateID, Value: a Concrete state.*
- std::shared\_ptr<[State](#)> [currentState](#) {nullptr}  
*Current state.*
- int [currentInsertedID](#) {1}
- int [previousStateID](#) {}

### 10.58.1 Detailed Description

Container for all game States and holds current state pointer. Updates [State](#) logic every frame.

Main [State](#) wrapper. All states are held in [states](#).

[currentState](#) calls [processInput\(\)](#), [update\(float dt\)](#) and [draw\(Window& window\)](#) every frame from within [Game](#) class.

Holds all [State](#) in a map, and holds a pointer to current [State](#).

### 10.58.2 Member Function Documentation

#### 10.58.2.1 draw()

```
void StateMachine::draw (
    Window & window ) const
```

Pass-through method, calls [draw\(Window& window\)](#) of whatever is [currentState](#).

Called by [Game](#). Works as a polymorphic indirection level between [Game](#) and Concrete [State](#). Here is the caller graph for this function:



#### 10.58.2.2 erase()

```
void StateMachine::erase (
    int stateID )
```

Erases the state of a given ID. The only explicitly erased state is [StateGame](#) when making new map.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>stateID</code> | the ID of state to be erased. |
|----------------------|-------------------------------|

- todo: investigate if its buggy \*/
- todo: throw EraseStateBadID \*/

Here is the caller graph for this function:



### 10.58.2.3 getPreviousStateID()

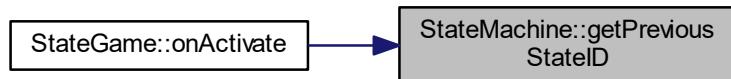
```
int StateMachine::getPreviousStateID ( ) const [inline]
```

Retrieve the [previousStateID](#).

**Returns**

int [previousStateID](#).

Here is the caller graph for this function:



### 10.58.2.4 insert()

```
int StateMachine::insert (
    const std::shared_ptr< State > & state )
```

Inserts new preallocated state into [states](#).

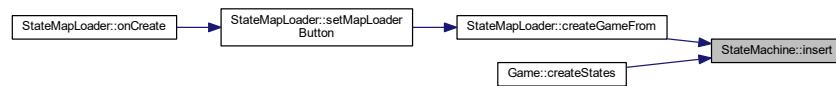
**Parameters**

|              |  |
|--------------|--|
| <i>state</i> | a shared_pointer to a newly allocated state. |
|--------------|--|

**Returns**

int the number of that state ID so that this state can be found later on.

Here is the caller graph for this function:

**10.58.2.5 operator+=( )**

```
StateMachine & StateMachine::operator+= (
    const std::shared_ptr< State > & s )
```

**10.58.2.6 operator=( )**

```
StateMachine & StateMachine::operator= (
    int stateID )
```

[State](#) can be switched to using [operator=](#) or [switchTo\(int\)](#).

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>stateID</code> | the ID of state to be switched to. |
|----------------------|------------------------------------|

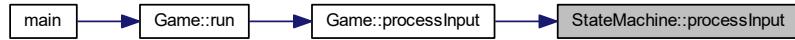
Here is the call graph for this function:

**10.58.2.7 processInput()**

```
void StateMachine::processInput ( ) const
```

Pass-through method, calls `processInput()` of whatever is `currentState`.

Called by `Game`. Works as a polymorphic indirection level between `Game` and Concrete `State`. Here is the caller graph for this function:



#### 10.58.2.8 setPreviousStateID()

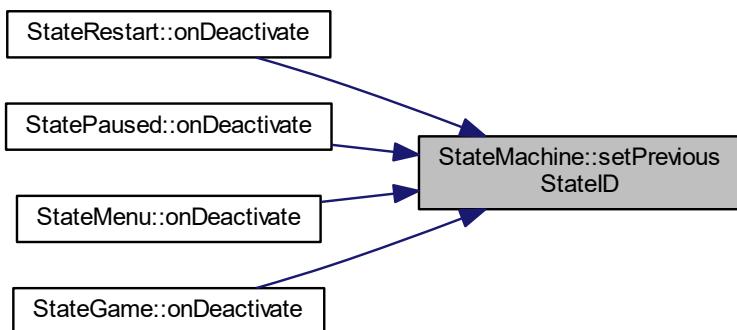
```
void StateMachine::setPreviousStateID (
    int stateID ) [inline]
```

Sets the `previousStateID` to a given ID.

##### Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>stateID</code> | new <code>previousStateID</code> . |
|----------------------|------------------------------------|

Here is the caller graph for this function:



#### 10.58.2.9 switchTo()

```
void StateMachine::switchTo (
    int stateID )
```

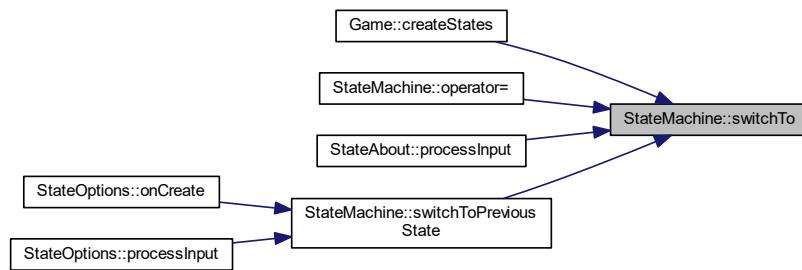
Switches `currentState` to a state found in the `states` by the given argument.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>stateID</code> | the ID of state to be switched to. |
|----------------------|------------------------------------|

- todo: throw BadSwitchStateID \*/

Here is the caller graph for this function:

**10.58.2.10 switchToDefaultState()**

```
void StateMachine::switchToDefaultState ( )
```

Switches to a default [State](#).

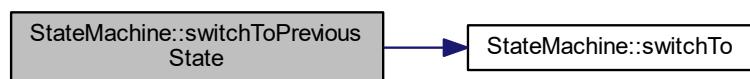
**Note**

Currently not implemented, but might be found useful in a future as an idea.

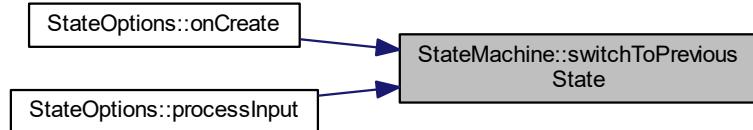
**10.58.2.11 switchToPreviousState()**

```
void StateMachine::switchToPreviousState ( )
```

Switches to whatever was the previous [State](#) held in `previousStateID`. Here is the call graph for this function:



Here is the caller graph for this function:



### 10.58.2.12 update()

```
void StateMachine::update (
    float dt ) const
```

Pass-through method, calls update(float dt) of whatever is [currentState](#).

Called by [Game](#). Works as a polymorphic indirection level between [Game](#) and Concrete [State](#). Here is the caller graph for this function:



## 10.58.3 Member Data Documentation

### 10.58.3.1 currentInsertedID

```
int StateMachine::currentInsertedID {1} [private]
```

### 10.58.3.2 currentState

```
std::shared_ptr<State> StateMachine::currentState {nullptr} [private]
```

Current state.

#### 10.58.3.3 previousStateID

```
int StateMachine::previousStateID {} [private]
```

#### 10.58.3.4 states

```
std::unordered_map<int, std::shared_ptr<State> > StateMachine::states {} [private]
```

Container for all states. Key: stateID, Value: a Concrete state.

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

- [StateMachine.h](#)
- [StateMachine.cpp](#)

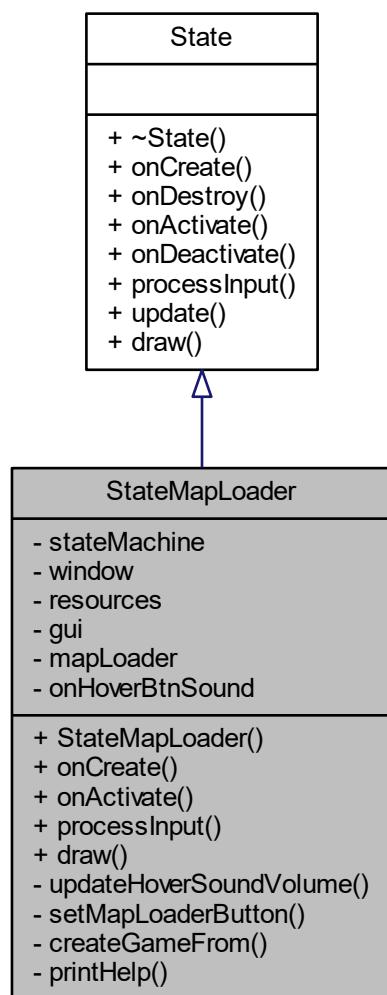
## 10.59 StateMapLoader Class Reference

Encapsulates the logic of map loader menu within one [StateMapLoader](#) class.

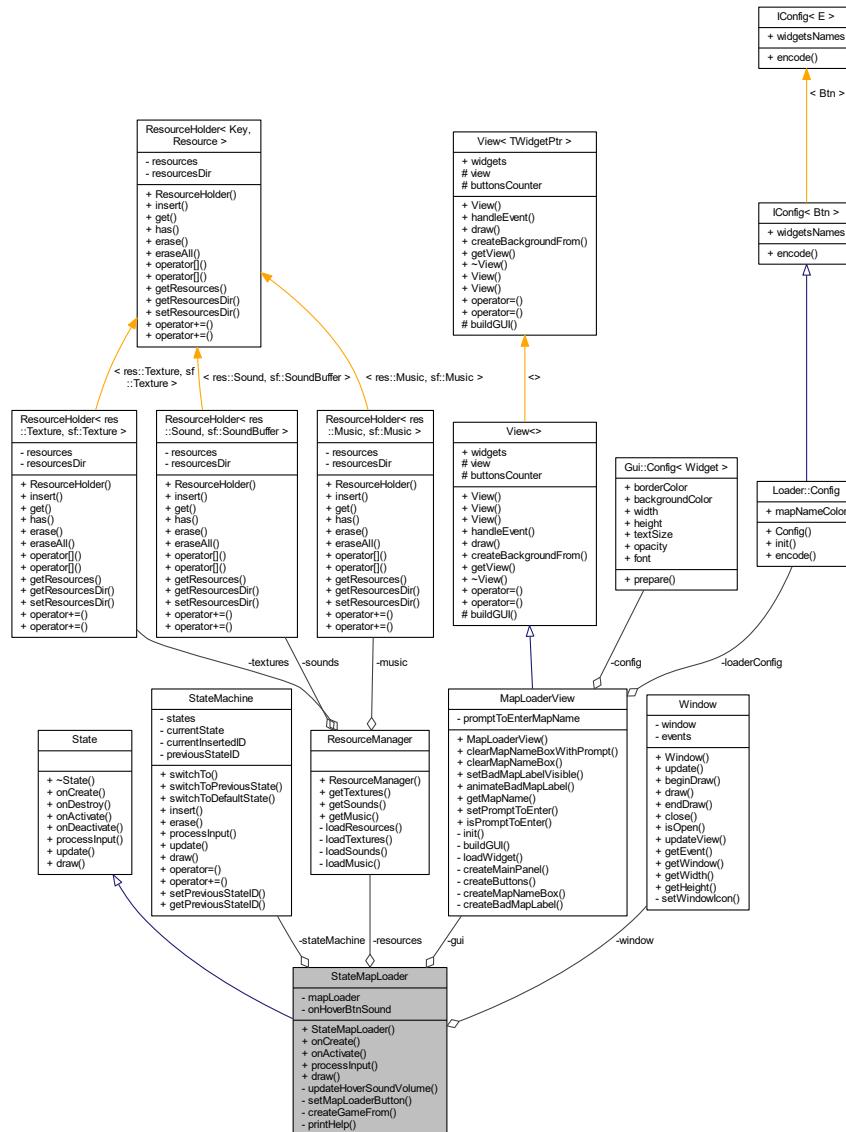
---

```
#include <StateMapLoader.h>
```

Inheritance diagram for StateMapLoader:



Collaboration diagram for StateMapLoader:



## Public Member Functions

- **StateMapLoader** (`StateMachine &stateMachine`, `Window &window`, `ResourceManager &resourceManager`)  
*Create the `StateMapLoader`.*
- void **onCreate ()** override  
*Called when the `StateMapLoader` is created. Connects signals to gui widgets.*
- void **onActivate ()** override  
*Called when `StateMapLoader` is activated. Hide badMapLabel and update the sound volume.*
- void **processInput ()** override  
*Processes all inputs received by `gui`.*
- void **draw (Window &)** override  
*Draws the `gui` onto the screen.*

## Private Member Functions

- void [updateHoverSoundVolume \(\)](#)  
*Updates the buttons hover volume.*
- void [setMapLoaderButton \(\)](#)
- void [createGameFrom \(std::string\\_view mapName\)](#)

## Static Private Member Functions

- static void [printHelp \(std::ostream &ost\)](#)

## Private Attributes

- [StateMachine & stateMachine](#)
- [Window & window](#)
- [ResourceManager & resources](#)
- [MapLoaderView gui](#)
- std::optional< std::variant< [MapLoader< Bmp >](#), [MapLoader< Txt >](#) > > [mapLoader](#)
- sf::Sound [onHoverBtnSound](#)

### 10.59.1 Detailed Description

Encapsulates the logic of map loader menu within one [StateMapLoader](#) class.

#### Note

All possible transitions from [StateMapLoader](#) are:

- [StateGame](#)
- [StateMenu](#)

Inherits from [State](#) base. Created inside [Game](#) and Stored inside [StateMachine](#). The ID of [StateMenu](#) can be retrieved from [state::menuID](#).

### 10.59.2 Constructor & Destructor Documentation

#### 10.59.2.1 StateMapLoader()

```
StateMapLoader::StateMapLoader (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resourceManager )
```

Create the [StateMapLoader](#).

**Parameters**

|                        |  |
|------------------------|--|
| <i>stateMachine</i>    |  |
| <i>window</i>          |  |
| <i>resourceManager</i> |  |

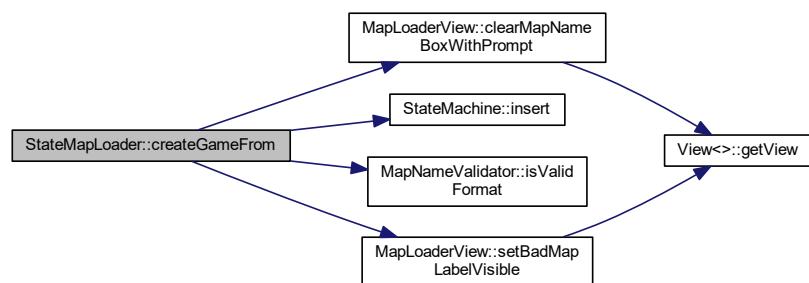
Sets the default #onHoverSound. Constructor invoked from [Game](#).

### 10.59.3 Member Function Documentation

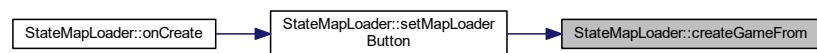
#### 10.59.3.1 `createGameFrom()`

```
void StateMapLoader::createGameFrom (
    std::string_view mapName ) [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:



### 10.59.3.2 draw()

```
void StateMapLoader::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



### 10.59.3.3 onActivate()

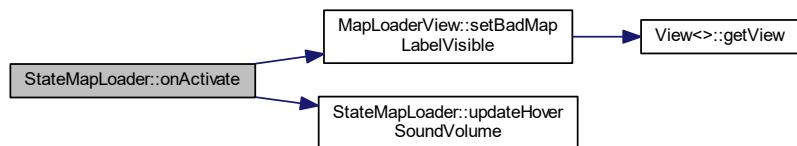
```
void StateMapLoader::onActivate () [override], [virtual]
```

Called when [StateMapLoader](#) is activated. Hide badMapLabel and update the sound volume.

Possibly the user was in [StateMapLoader](#) earlier on, entered wrong map name and went back to [StateMenu](#). Therefore: we need to hide badMapLabel message upon re-entering [StateMapLoader](#).

Reimplemented from [State](#).

Here is the call graph for this function:



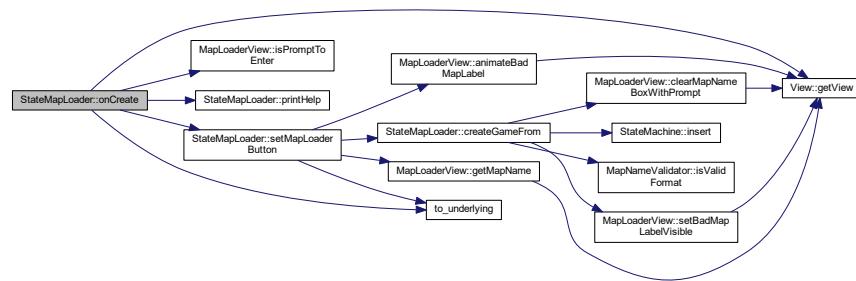
### 10.59.3.4 onCreate()

```
void StateMapLoader::onCreate ( ) [override], [virtual]
```

Called when the [StateMapLoader](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.59.3.5 printHelp()

```
void StateMapLoader::printHelp (
    std::ostream & ost ) [static], [private]
```

Here is the caller graph for this function:



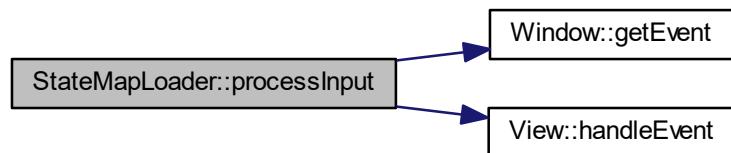
### 10.59.3.6 processInput()

```
void StateMapLoader::processInput ( ) [override], [virtual]
```

Processes all inputs received by [gui](#).

Implements [State](#).

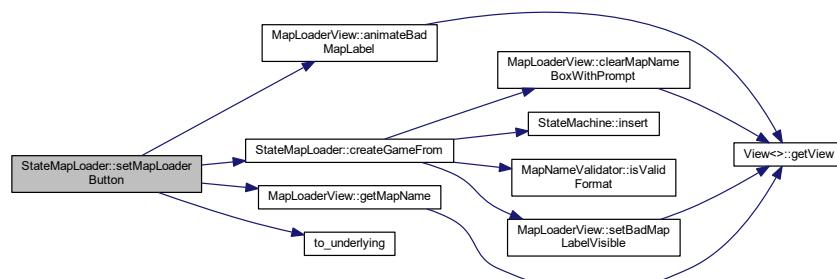
Here is the call graph for this function:



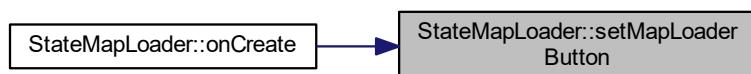
### 10.59.3.7 setMapLoaderButton()

```
void StateMapLoader::setMapLoaderButton ( ) [private]
```

Here is the call graph for this function:



Here is the caller graph for this function:

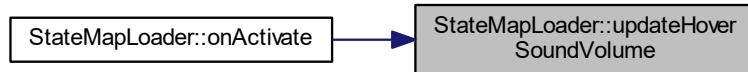


### 10.59.3.8 updateHoverSoundVolume()

```
void StateMapLoader::updateHoverSoundVolume ( ) [private]
```

Updates the buttons hover volume.

Here is the caller graph for this function:



## 10.59.4 Member Data Documentation

### 10.59.4.1 gui

```
MapLoaderView StateMapLoader::gui [private]
```

### 10.59.4.2 mapLoader

```
std::optional<std::variant<MapLoader<Bmp>, MapLoader<Txt>> > StateMapLoader::mapLoader  
[private]
```

### 10.59.4.3 onHoverBtnSound

```
sf::Sound StateMapLoader::onHoverBtnSound [private]
```

### 10.59.4.4 resources

```
ResourceManager& StateMapLoader::resources [private]
```

#### 10.59.4.5 stateMachine

```
StateMachine& StateMapLoader::stateMachine [private]
```

#### 10.59.4.6 window

```
Window& StateMapLoader::window [private]
```

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

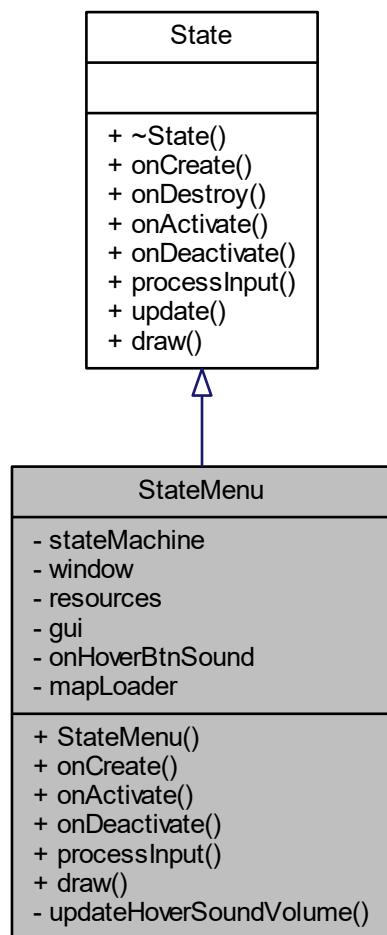
- [StateMapLoader.h](#)
- [StateMapLoader.cpp](#)

## 10.60 StateMenu Class Reference

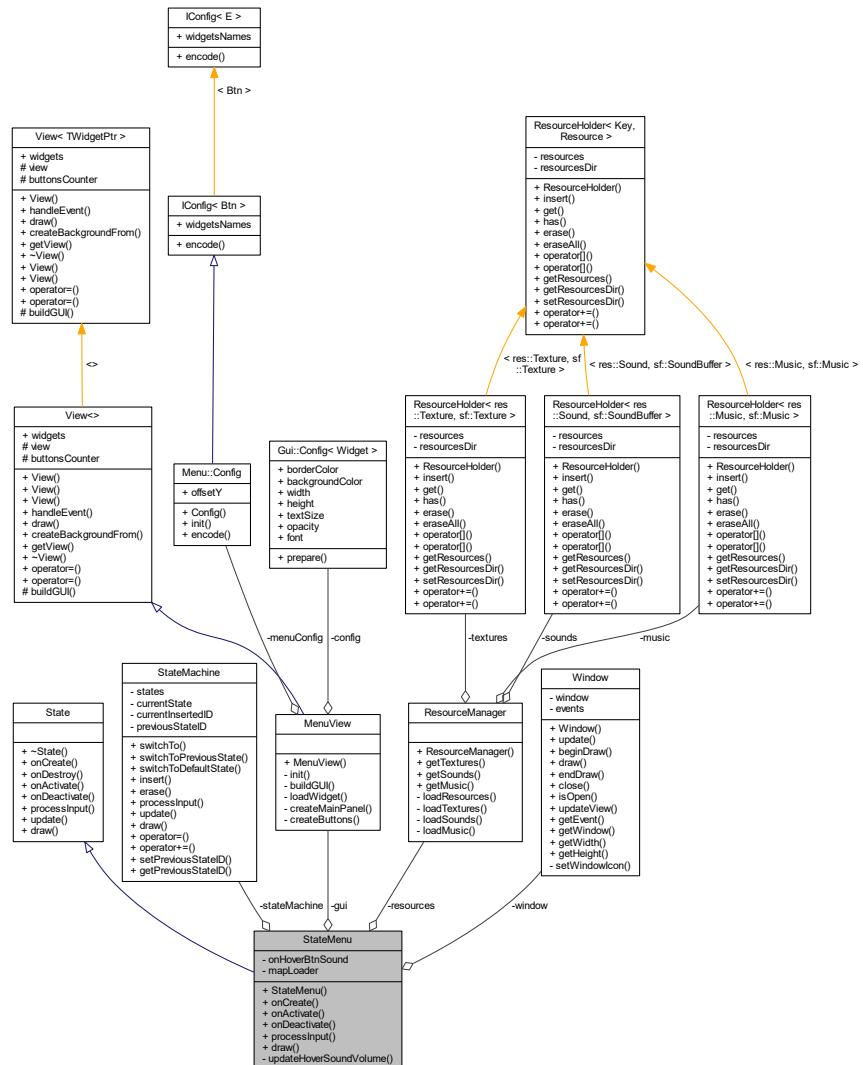
Encapsulates the logic of main menu within one [StateRestart](#) class.

```
#include <StateMenu.h>
```

Inheritance diagram for StateMenu:



## Collaboration diagram for StateMenu:



## Public Member Functions

- **StateMachine** (`StateMachine &stateMachine, Window &window, ResourceManager &resourceManager`)  
*Create the StateMenu.*
  - void **onCreate** () override  
*Called when the StateMenu is created. Connects signals to gui widgets.*
  - void **onActivate** () override  
*Called when StateMenu is activated. Updates the sound volume.*
  - void **onDeactivate** () override  
*Called when StateMenu is closed. Stores that previous state was state::menuID.*
  - void **processInput** () override  
*Processes all inputs received by gui.*
  - void **draw** (`Window &`) override  
*Draws the gui onto the screen.*

## Private Member Functions

- void [updateHoverSoundVolume \(\)](#)  
*Updates the buttons hover volume.*

## Private Attributes

- [StateMachine & stateMachine](#)
- [Window & window](#)
- [ResourceManager & resources](#)
- [MenuView gui](#)
- [sf::Sound onHoverBtnSound](#)
- [std::optional< std::variant< MapLoader< Bmp >, MapLoader< Txt > >> mapLoader](#)

### 10.60.1 Detailed Description

Encapsulates the logic of main menu within one [StateRestart](#) class.

#### Note

All possible transitions from [StateMenu](#) are:

- [StateGame](#)
- [StateMapLoader](#)
- [StateOptions](#)
- [StateAbout](#)

Inherits from [State](#) base. Created inside [Game](#) and Stored inside [StateMachine](#). The ID of [StateMenu](#) can be retrieved from [state::menuID](#).

### 10.60.2 Constructor & Destructor Documentation

#### 10.60.2.1 StateMenu()

```
StateMenu::StateMenu (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resourceManager )
```

Create the [StateMenu](#).

#### Parameters

|                              |  |
|------------------------------|--|
| <code>stateMachine</code>    |  |
| <code>window</code>          |  |
| <code>resourceManager</code> |  |

Sets the default #onHoverSound. Constructor invoked from [Game](#).

### 10.60.3 Member Function Documentation

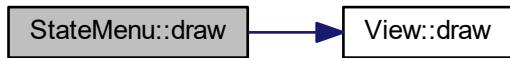
#### 10.60.3.1 draw()

```
void StateMenu::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



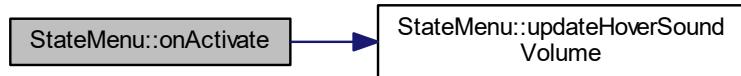
#### 10.60.3.2 onActivate()

```
void StateMenu::onActivate ( ) [override], [virtual]
```

Called when [StateMenu](#) is activated. Updates the sound volume.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.60.3.3 onCreate()

```
void StateMenu::onCreate ( ) [override], [virtual]
```

Called when the [StateMenu](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



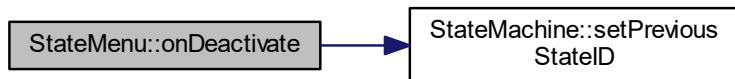
### 10.60.3.4 onDeactivate()

```
void StateMenu::onDeactivate ( ) [override], [virtual]
```

Called when [StateMenu](#) is closed. Stores that previous state was [state::menuID](#).

Reimplemented from [State](#).

Here is the call graph for this function:



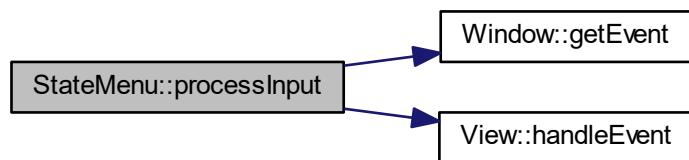
### 10.60.3.5 processInput()

```
void StateMenu::processInput ( ) [override], [virtual]
```

Processes all inputs received by [gui](#).

Implements [State](#).

Here is the call graph for this function:

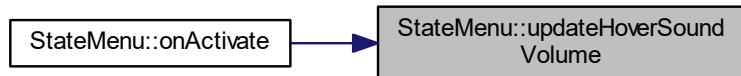


### 10.60.3.6 updateHoverSoundVolume()

```
void StateMenu::updateHoverSoundVolume ( ) [private]
```

Updates the buttons hover volume.

Here is the caller graph for this function:



## 10.60.4 Member Data Documentation

### 10.60.4.1 gui

```
MenuView StateMenu::gui [private]
```

**10.60.4.2 mapLoader**

```
std::optional<std::variant<MapLoader<Bmp>, MapLoader<Txt>>> StateMenu::mapLoader [private]
```

**10.60.4.3 onHoverBtnSound**

```
sf::Sound StateMenu::onHoverBtnSound [private]
```

**10.60.4.4 resources**

```
ResourceManager& StateMenu::resources [private]
```

**10.60.4.5 stateMachine**

```
StateMachine& StateMenu::stateMachine [private]
```

**10.60.4.6 window**

```
Window& StateMenu::window [private]
```

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

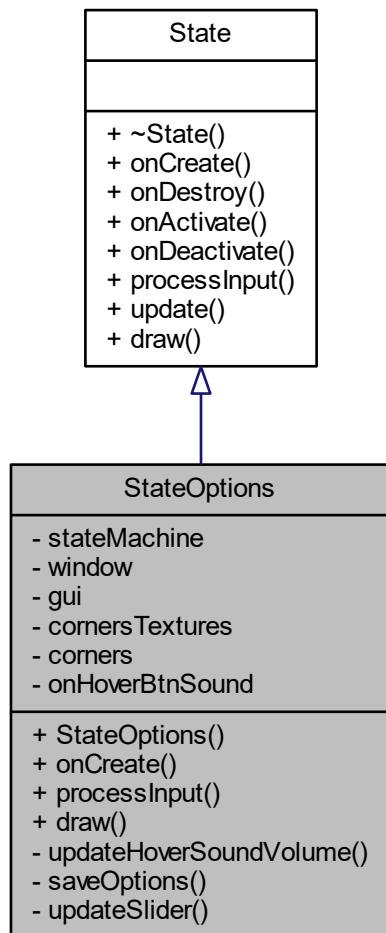
- [StateMenu.h](#)
- [StateMenu.cpp](#)

## 10.61 StateOptions Class Reference

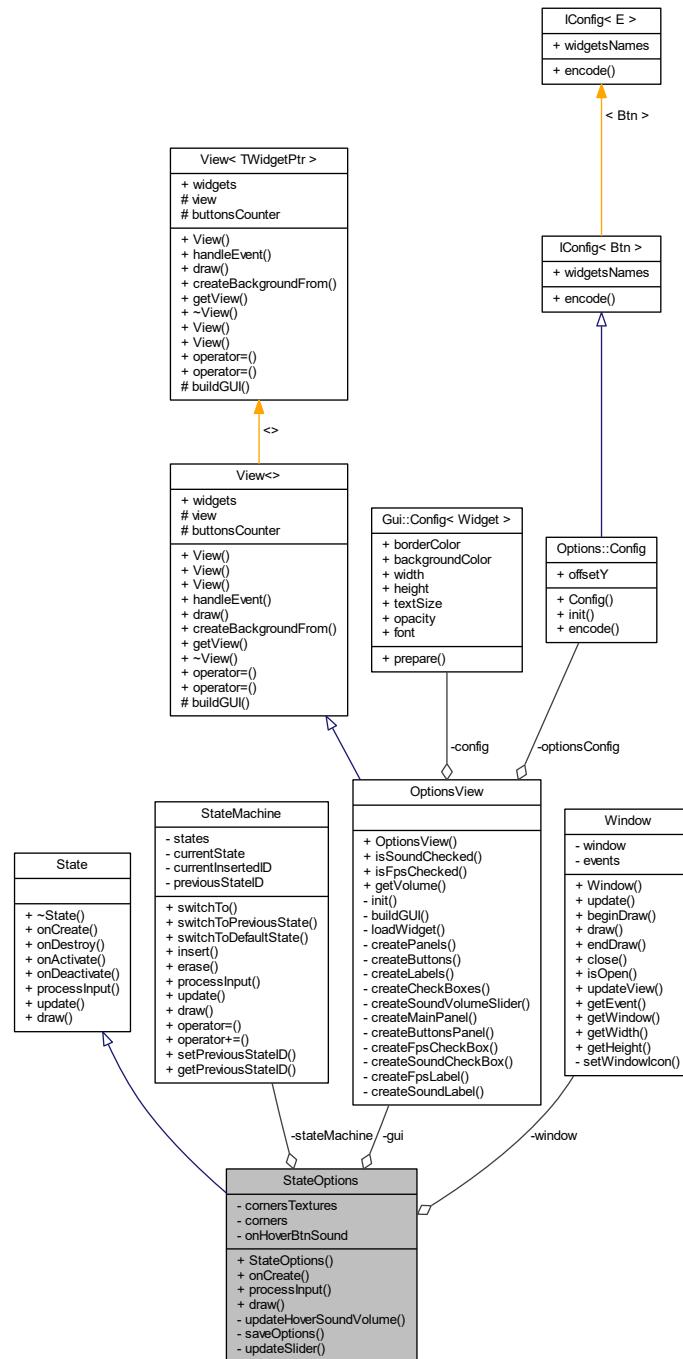
Encapsulates the logic of options menu within one [StateOptions](#) class.

```
#include <StateOptions.h>
```

Inheritance diagram for StateOptions:



Collaboration diagram for StateOptions:



## Public Member Functions

- **StateOptions (StateMachine &stateMachine, Window &window, ResourceManager &resources)**  
*Create the [StateOptions](#).*
- void **onCreate () override**  
*Called when the [StateOptions](#) is created. Connects signals to gui widgets.*
- void **processInput () override**

- Processes all inputs received by `gui`.*
- void `draw (Window &)` override  
*Draws the `gui` and `corners` onto the screen.*

## Private Member Functions

- void `updateHoverSoundVolume ()`  
*Updates the buttons hover volume.*
- void `saveOptions ()`
- void `updateSlider ()`

## Private Attributes

- `StateMachine & stateMachine`
- `Window & window`
- `OptionsView gui`
- `std::array< std::reference_wrapper< sf::Texture >, 4 > cornersTextures`
- `std::array< sf::Sprite, 4 > corners`
- `sf::Sound onHoverBtnSound`

### 10.61.1 Detailed Description

Encapsulates the logic of options menu within one `StateOptions` class.

#### Note

All possible transitions from `StateMenu` are:

- `StateGame`
- `StateMenu`
- `StateKeybinds`
- `StatePaused`

Inherits from `State` base. Created inside `Game` and Stored inside `StateMachine`. The ID of `StateMenu` can be retrieved from `state::menuID`.

### 10.61.2 Constructor & Destructor Documentation

#### 10.61.2.1 StateOptions()

```
StateOptions::StateOptions (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resources )
```

Create the `StateOptions`.

**Parameters**

|                        |  |
|------------------------|--|
| <i>stateMachine</i>    |  |
| <i>window</i>          |  |
| <i>resourceManager</i> |  |

Sets the default #onHoverSound and [corners](#) textures and positions. Constructor invoked from [Game](#).

### 10.61.3 Member Function Documentation

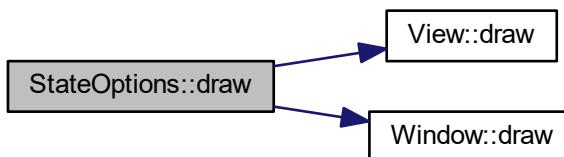
#### 10.61.3.1 draw()

```
void StateOptions::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) and [corners](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



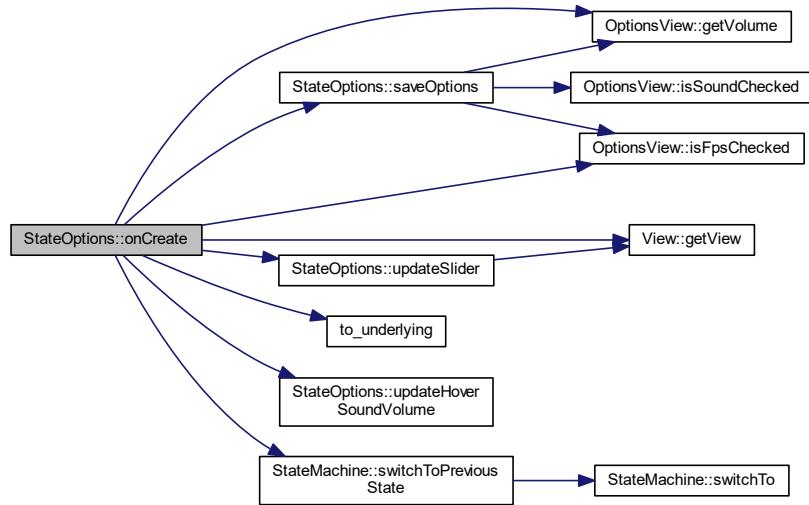
#### 10.61.3.2 onCreate()

```
void StateOptions::onCreate ( ) [override], [virtual]
```

Called when the [StateOptions](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.61.3.3 processInput()

```
void StateOptions::processInput( ) [override], [virtual]
```

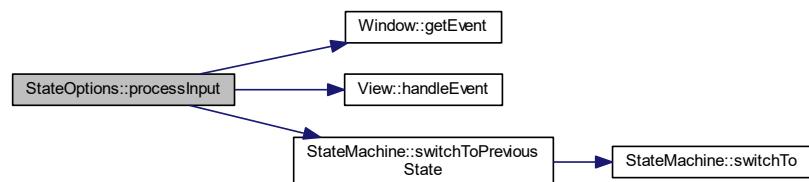
Processes all inputs received by [gui](#).

#### Note

Also, process keypresses events trying to switch to a [previous State](#).

Implements [State](#).

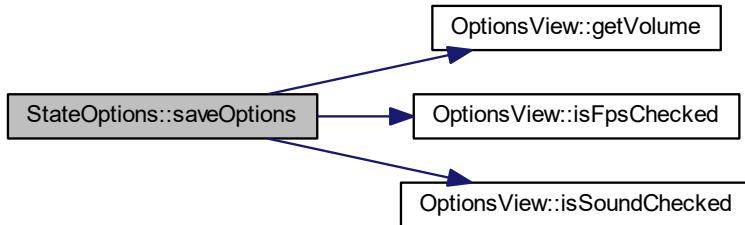
Here is the call graph for this function:



#### 10.61.3.4 saveOptions()

```
void StateOptions::saveOptions ( ) [private]
```

Saves configuration options into struct. Here is the call graph for this function:



Here is the caller graph for this function:



#### 10.61.3.5 updateHoverSoundVolume()

```
void StateOptions::updateHoverSoundVolume ( ) [private]
```

Updates the buttons hover volume.

Here is the caller graph for this function:



### 10.61.3.6 updateSlider()

```
void StateOptions::updateSlider ( ) [private]
```

Updates slider appearance view based on Sound Enable checkbox. Here is the call graph for this function:



Here is the caller graph for this function:



## 10.61.4 Member Data Documentation

### 10.61.4.1 corners

```
std::array<sf::Sprite, 4> StateOptions::corners [private]
```

### 10.61.4.2 cornersTextures

```
std::array<std::reference_wrapper<sf::Texture>, 4> StateOptions::cornersTextures [private]
```

### 10.61.4.3 gui

```
OptionsView StateOptions::gui [private]
```

#### 10.61.4.4 onHoverBtnSound

```
sf::Sound StateOptions::onHoverBtnSound [private]
```

#### 10.61.4.5 stateMachine

```
StateMachine& StateOptions::stateMachine [private]
```

#### 10.61.4.6 window

```
Window& StateOptions::window [private]
```

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

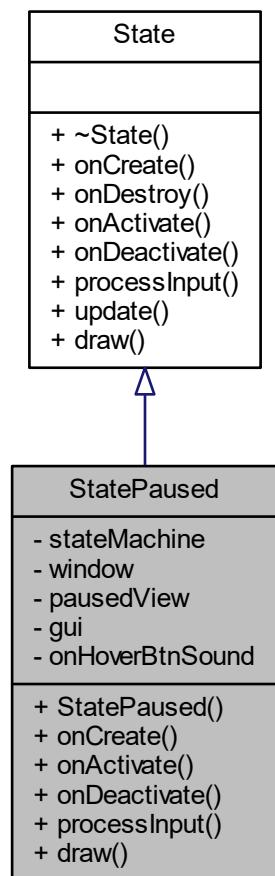
- [StateOptions.h](#)
- [StateOptions.cpp](#)

## 10.62 StatePaused Class Reference

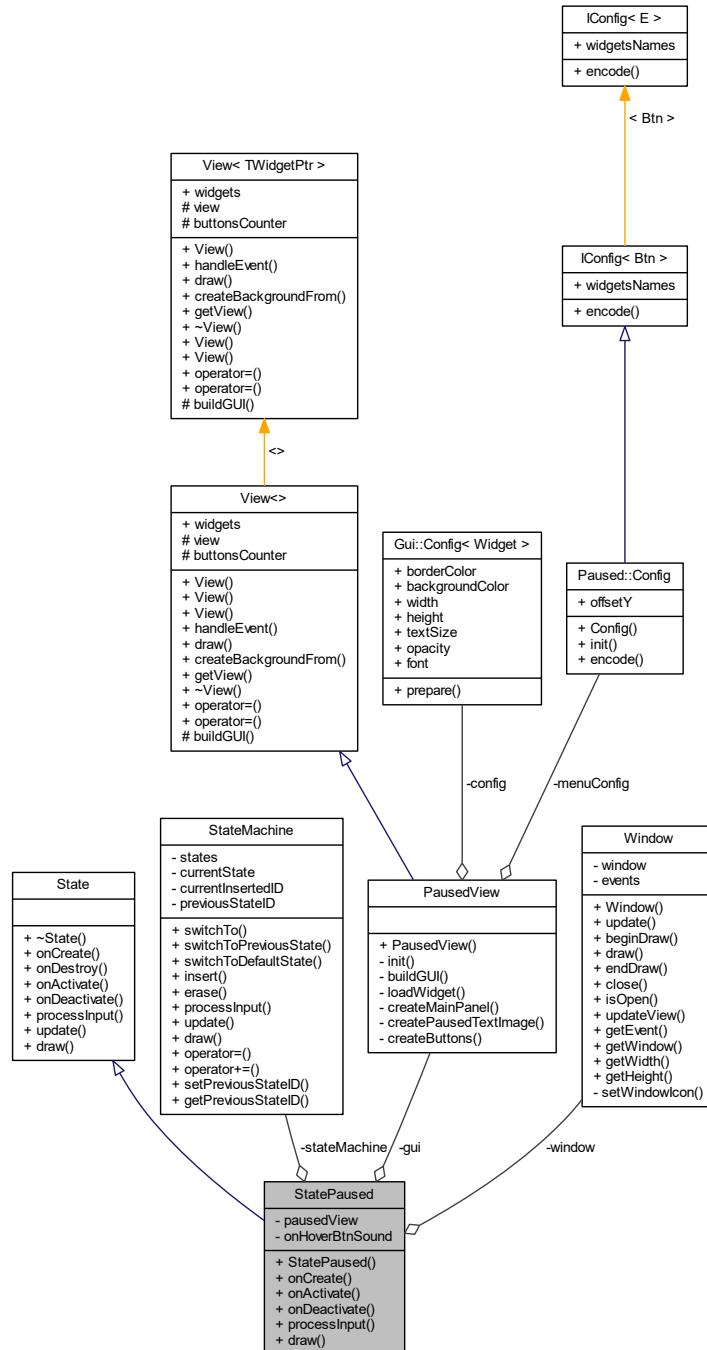
Encapsulates the logic of paused menu within one [StatePaused](#) class.

```
#include <StatePaused.h>
```

Inheritance diagram for StatePaused:



Collaboration diagram for StatePaused:



## Public Member Functions

- **StatePaused** (`StateMachine &stateMachine, Window &window, ResourceManager &resources`)  
*Create the `StatePaused`.*
- void **onCreate** () override  
*Called when the `StatePaused` is created. Connects signals to gui widgets.*
- void **onActivate** () override

*Called when `StatePaused` is activated. Updates the view and the sound volume.*

- void `onDeactivate ()` override

*Called when `StatePaused` is closed. Stores that previous state was `state::pausedID`.*

- void `processInput ()` override

*Processes all inputs received by `gui`.*

- void `draw (Window &)` override

*Draws the `gui` onto the screen.*

## Private Attributes

- `StateMachine & stateMachine`
- `Window & window`
- `sf::View pausedView`
- `PausedView gui`
- `sf::Sound onHoverBtnSound`

### 10.62.1 Detailed Description

Encapsulates the logic of paused menu within one `StatePaused` class.

#### Note

All possible transitions from `StateMenu` are:

- `StateGame`
- `StateOptions`
- `StateMenu`

Inherits from `State` base. Created inside `Game` and Stored inside `StateMachine`. The ID of `StateMenu` can be retrieved from `state::menuID`.

### 10.62.2 Constructor & Destructor Documentation

#### 10.62.2.1 StatePaused()

```
StatePaused::StatePaused (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resources )
```

Create the `StatePaused`.

#### Parameters

|                              |  |
|------------------------------|--|
| <code>stateMachine</code>    |  |
| <code>window</code>          |  |
| <code>resourceManager</code> |  |

Sets the default #onHoverSound and #corners textures and positions. Constructor invoked from [Game](#).

### 10.62.3 Member Function Documentation

#### 10.62.3.1 draw()

```
void StatePaused::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



#### 10.62.3.2 onActivate()

```
void StatePaused::onActivate ( ) [override], [virtual]
```

Called when [StatePaused](#) is activated. Updates the view and the sound volume.

Reimplemented from [State](#).

Here is the call graph for this function:



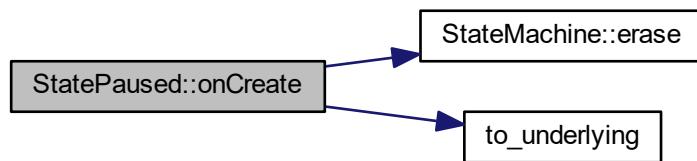
### 10.62.3.3 onCreate()

```
void StatePaused::onCreate( ) [override], [virtual]
```

Called when the [StatePaused](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



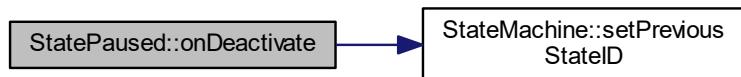
### 10.62.3.4 onDeactivate()

```
void StatePaused::onDeactivate( ) [override], [virtual]
```

Called when [StatePaused](#) is closed. Stores that previous state was [state::pausedID](#).

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.62.3.5 processInput()

```
void StatePaused::processInput( ) [override], [virtual]
```

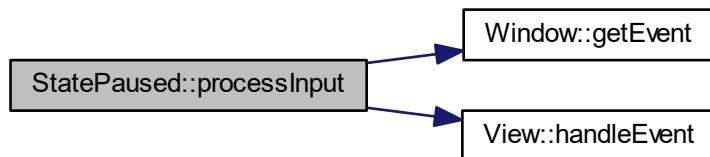
Processes all inputs received by [gui](#).

#### Note

Also, process keypresses events trying to switch to a different [State](#).

Implements [State](#).

Here is the call graph for this function:



## 10.62.4 Member Data Documentation

### 10.62.4.1 gui

```
PausedView StatePaused::gui [private]
```

### 10.62.4.2 onHoverBtnSound

```
sf::Sound StatePaused::onHoverBtnSound [private]
```

### 10.62.4.3 pausedView

```
sf::View StatePaused::pausedView [private]
```

#### 10.62.4.4 stateMachine

```
StateMachine& StatePaused::stateMachine [private]
```

#### 10.62.4.5 window

```
Window& StatePaused::window [private]
```

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

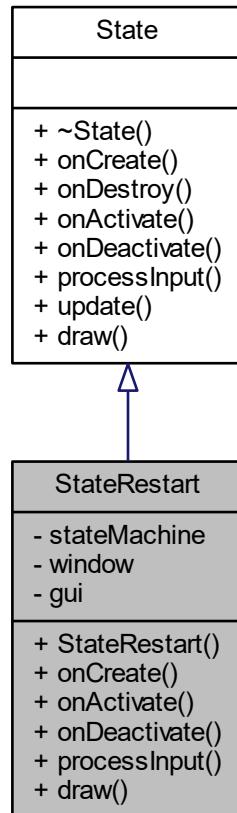
- [StatePaused.h](#)
- [StatePaused.cpp](#)

## 10.63 StateRestart Class Reference

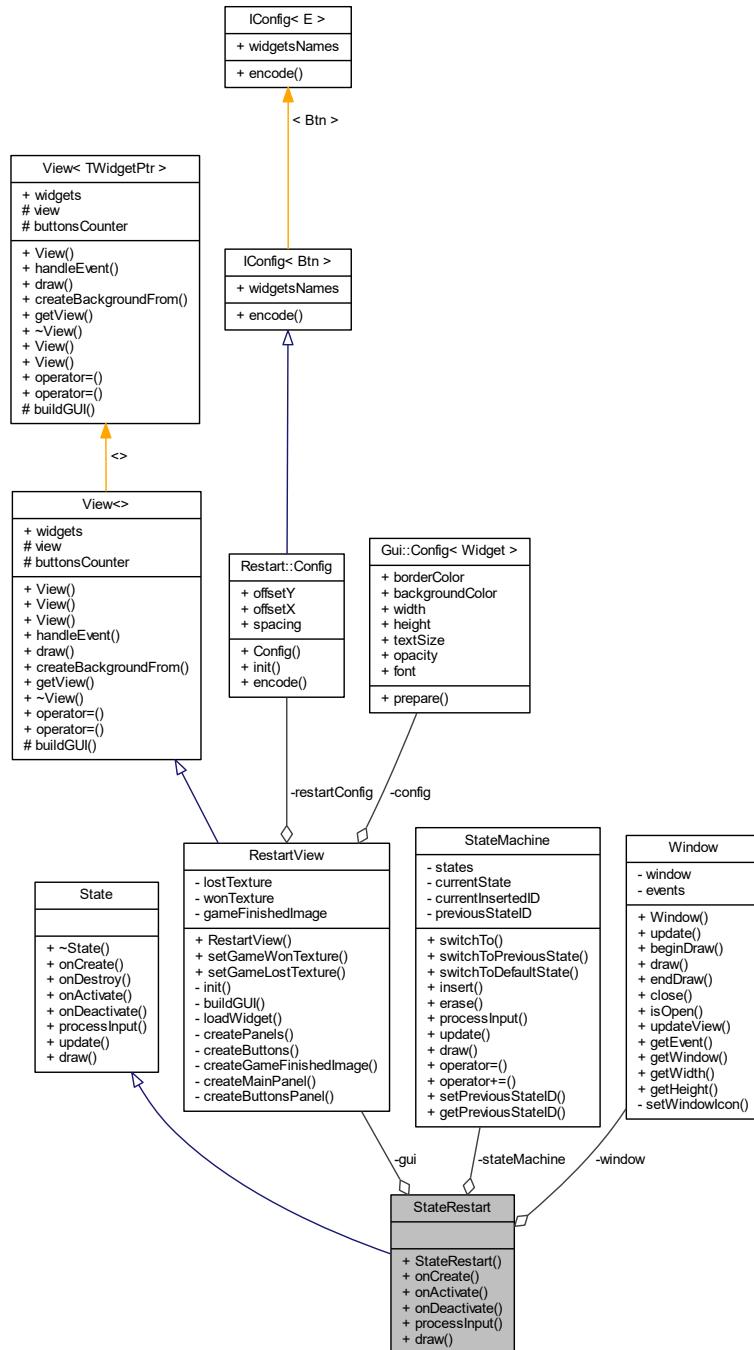
Encapsulates the logic of restart menu within one [StateRestart](#) class.

```
#include <StateRestart.h>
```

Inheritance diagram for StateRestart:



Collaboration diagram for StateRestart:



## Public Member Functions

- `StateRestart (StateMachine &stateMachine, Window &window, ResourceManager &resourceManager)`  
*Create the `StateRestart`.*
- `void onCreate () override`  
*Called when the `StateRestart` is created. Connects signals to gui widgets.*
- `void onActivate () override`

- Sets either `gameWonTexture()` or `gameLostTexture()`.
- void `onDeactivate ()` override
  - Stores the `state::restartID` as `#previousStateID`.*
- void `processInput ()` override
  - Processes all inputs received by `gui`.*
- void `draw (Window &)` override
  - Draws the `gui` onto the screen.*

## Private Attributes

- `StateMachine & stateMachine`
- `Window & window`
- `RestartView gui`

### 10.63.1 Detailed Description

Encapsulates the logic of restart menu within one `StateRestart` class.

#### Note

All possible transitions from `StateMenu` are:

- `StateGame`
- `StateMenu`

Inherits from `State` base. Created inside `Game` and Stored inside `StateMachine`. The ID of `StateMenu` can be retrieved from `state::menuID`.

### 10.63.2 Constructor & Destructor Documentation

#### 10.63.2.1 StateRestart()

```
StateRestart::StateRestart (
    StateMachine & stateMachine,
    Window & window,
    ResourceManager & resourceManager )
```

Create the `StateRestart`.

#### Parameters

|                              |  |
|------------------------------|--|
| <code>stateMachine</code>    |  |
| <code>window</code>          |  |
| <code>resourceManager</code> |  |

### 10.63.3 Member Function Documentation

#### 10.63.3.1 draw()

```
void StateRestart::draw (
    Window & ) [override], [virtual]
```

Draws the [gui](#) onto the screen.

Implements [State](#).

Here is the call graph for this function:



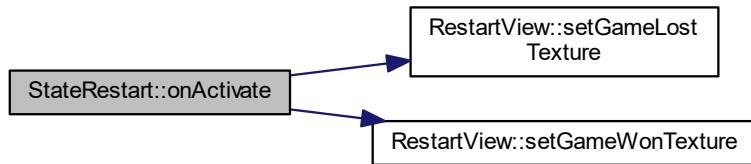
#### 10.63.3.2 onActivate()

```
void StateRestart::onActivate ( ) [override], [virtual]
```

Sets either gameWonTexture() or gameLostTexture().

Reimplemented from [State](#).

Here is the call graph for this function:



### 10.63.3.3 onCreate()

```
void StateRestart::onCreate ( ) [override], [virtual]
```

Called when the [StateRestart](#) is created. Connects signals to gui widgets.

Reimplemented from [State](#).

Here is the call graph for this function:



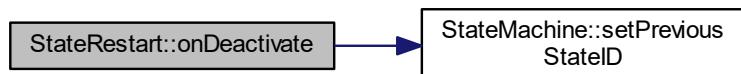
### 10.63.3.4 onDeactivate()

```
void StateRestart::onDeactivate ( ) [override], [virtual]
```

Stores the [state::restartID](#) as #previousStateID.

Reimplemented from [State](#).

Here is the call graph for this function:



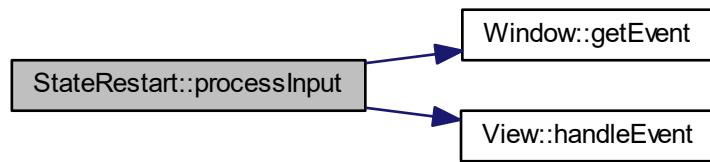
### 10.63.3.5 processInput()

```
void StateRestart::processInput ( ) [override], [virtual]
```

Processes all inputs received by [gui](#).

Implements [State](#).

Here is the call graph for this function:



## 10.63.4 Member Data Documentation

### 10.63.4.1 gui

```
RestartView StateRestart::gui [private]
```

### 10.63.4.2 stateMachine

```
StateMachine& StateRestart::stateMachine [private]
```

### 10.63.4.3 window

```
Window& StateRestart::window [private]
```

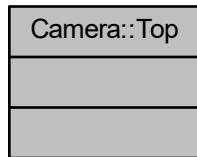
The documentation for this class was generated from the following files:

- [StateRestart.h](#)
- [StateRestart.cpp](#)

## 10.64 Camera::Top Struct Reference

Empty struct denoting previously solved on other axis was [Top](#) collision.

Collaboration diagram for Camera::Top:



### 10.64.1 Detailed Description

Empty struct denoting previously solved on other axis was [Top](#) collision.

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

- [Camera.h](#)

## 10.65 Txt Struct Reference

Empty struct used to determine file type in [MapLoader](#) variant.

```
#include <MapLoader.h>
```

Collaboration diagram for Txt:



### 10.65.1 Detailed Description

Empty struct used to determine file type in [MapLoader](#) variant.

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

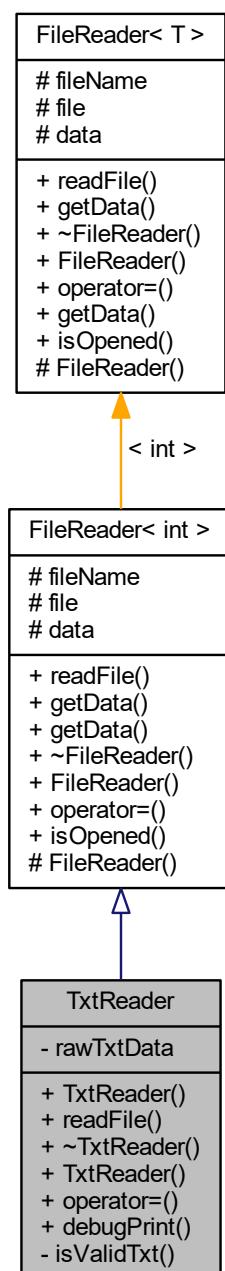
- [MapLoader.h](#)

## 10.66 TxtReader Class Reference

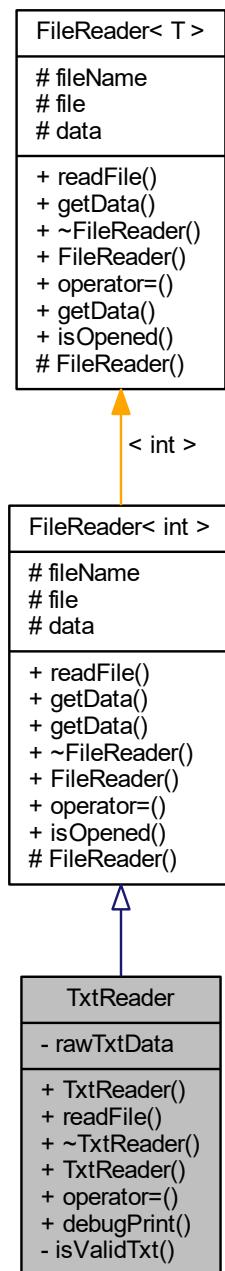
Reads the .txt file into the vector of int.

```
#include <TxtReader.h>
```

Inheritance diagram for TxtReader:



Collaboration diagram for TxtReader:



## Public Member Functions

- `TxtReader (const std::string &name, const std::ios_base::openmode &mode=std::ifstream::ate|std::ifstream::binary)`  
*Create a `TxtReader`.*
- `void readFile \(\) override`
- `~TxtReader \(\) override=default`

- `Default Virtual constructor.`
- `TxtReader (TxtReader &&) noexcept=default`  
*Enable move operations.*
- `TxtReader & operator= (TxtReader &&) noexcept=default`  
*Enable move operations.*
- `void debugPrint () const`  
*Print formatted file content debug info.*

## Private Member Functions

- `bool isValidTxt ()`  
*Analyzes if a .txt file is valid. It is valid only if it is well-formatted.*

## Private Attributes

- `std::string rawTxtData {}`  
*Used for analyzing if a file is a valid .txt.*

## Additional Inherited Members

### 10.66.1 Detailed Description

Reads the .txt file into the vector of int.

This would be a preferred way of loading maps, once the map editor is fully developed (*out of CP4 scope*).

### 10.66.2 Constructor & Destructor Documentation

#### 10.66.2.1 `TxtReader()` [1/2]

```
TxtReader::TxtReader (
    const std::string & name,
    const std::ios_base::openmode & mode = std::ifstream::ate | std::ifstream::binary
) [explicit]
```

Create a `TxtReader`.

#### Parameters

|                   |   |
|-------------------|---|
| <code>name</code> | Name of the file to be read.              |
| <code>mode</code> | Open mode introduced as a generalization. |

Reads the .txt file data.

### 10.66.2.2 ~TxtReader()

```
TxtReader::~TxtReader ( ) [override], [default]
```

Default Virtual constructor.

### 10.66.2.3 TxtReader() [2/2]

```
TxtReader::TxtReader (
    TxtReader && ) [default], [noexcept]
```

Enable move operations.

## 10.66.3 Member Function Documentation

### 10.66.3.1 debugPrint()

```
void TxtReader::debugPrint ( ) const
```

Print formatted file content debug info.

### 10.66.3.2 isValidTxt()

```
bool TxtReader::isValidTxt ( ) [private]
```

Analyzes if a .txt file is valid. It is valid only if it is well-formatted.

Uses regular expressions underneath.

#### Note

A file is valid if it matches  $R"((?:[:space:]*[:digit:]+[:space:]*)+)"$  pattern.

#### Returns

Was this a valid txt file?

Allocate the memory for raw txt data

Entire file is loaded into string

And checked if it indeed is a valid [Txt](#) file (white-space delimited numbers-only content)

### 10.66.3.3 operator=( )

```
TxtReader& TxtReader::operator= (
    TxtReader && ) [default], [noexcept]
```

Enable move operations.

### 10.66.3.4 readFile()

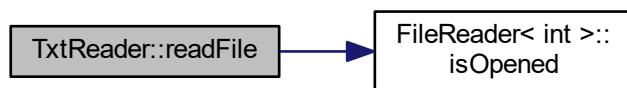
```
void TxtReader::readFile ( ) [override], [virtual]
```

**brief** Read the file data into the [PixelColor](#) vector.

Implements the pure virtual method of [FileReader< T >](#).

Implements [FileReader< int >](#).

Here is the call graph for this function:



## 10.66.4 Member Data Documentation

### 10.66.4.1 rawTxtData

```
std::string TxtReader::rawTxtData {} [private]
```

Used for analyzing if a file is a valid .txt.

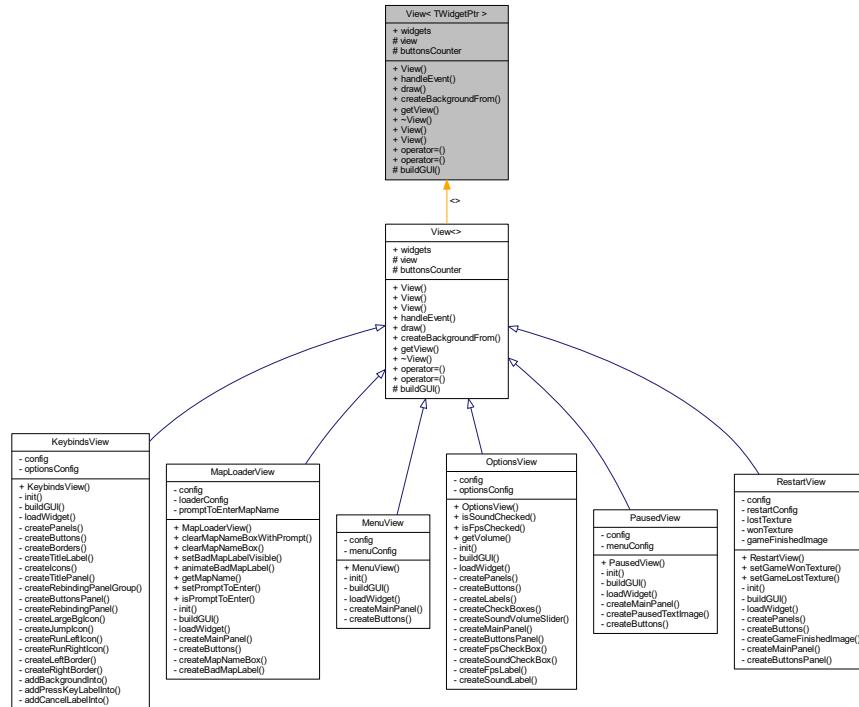
The documentation for this class was generated from the following files:

- [TxtReader.h](#)
- [TxtReader.cpp](#)

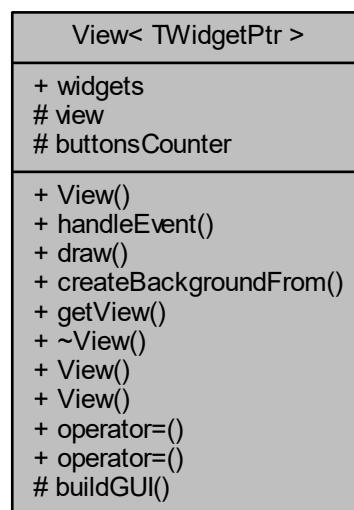
## 10.67 View< TWidgetPtr > Class Template Reference

```
#include <View.h>
```

Inheritance diagram for View< TWidgetPtr >:



Collaboration diagram for View< TWidgetPtr >:



## Public Member Functions

- `View (Window &window)`
- `virtual void handleEvent (std::queue< sf::Event > &events)`
- `virtual void draw ()`
- `void createBackgroundFrom (std::string_view backgroundName)`
- `auto & getView () const`
- `virtual ~View ()=default`
- `View (const View &)=default`
- `View (View &&) noexcept=default`
- `View & operator= (const View &)=default`
- `View & operator= (View &&) noexcept=default`

## Public Attributes

- `std::vector< TWidgetPtr > widgets`

## Protected Member Functions

- `virtual void buildGUI ()=0`

## Protected Attributes

- `tgui::Gui view`
- `int buttonsCounter {0}`

## 10.67.1 Constructor & Destructor Documentation

### 10.67.1.1 `View()` [1/3]

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
View< TWidgetPtr >::View (
    Window & window )  [inline], [explicit]
```

### 10.67.1.2 `~View()`

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
virtual View< TWidgetPtr >::~View ( )  [virtual], [default]
```

### 10.67.1.3 View() [2/3]

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
View< TWidgetPtr >::View (
    const View< TWidgetPtr > & ) [default]
```

### 10.67.1.4 View() [3/3]

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
View< TWidgetPtr >::View (
    View< TWidgetPtr > && ) [default], [noexcept]
```

## 10.67.2 Member Function Documentation

### 10.67.2.1 buildGUI()

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
virtual void View< TWidgetPtr >::buildGUI ( ) [protected], [pure virtual]
```

Implemented in [MapLoaderView](#), [RestartView](#), [OptionsView](#), [KeybindsView](#), [MenuView](#), and [PausedView](#).

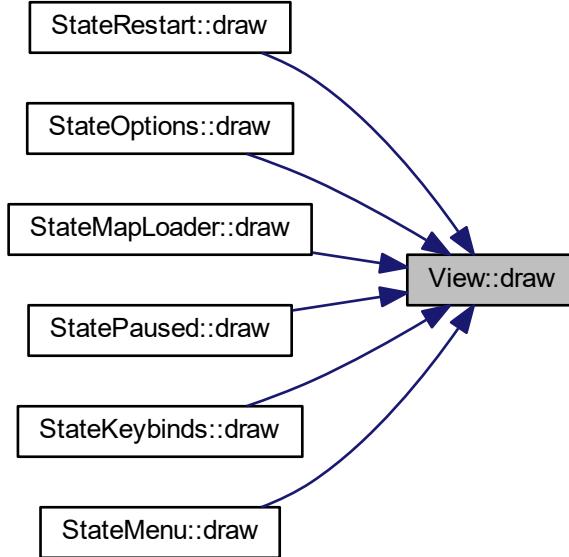
### 10.67.2.2 createBackgroundFrom()

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
void View< TWidgetPtr >::createBackgroundFrom (
    std::string_view backgroundName ) [inline]
```

### 10.67.2.3 draw()

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
virtual void View< TWidgetPtr >::draw ( ) [inline], [virtual]
```

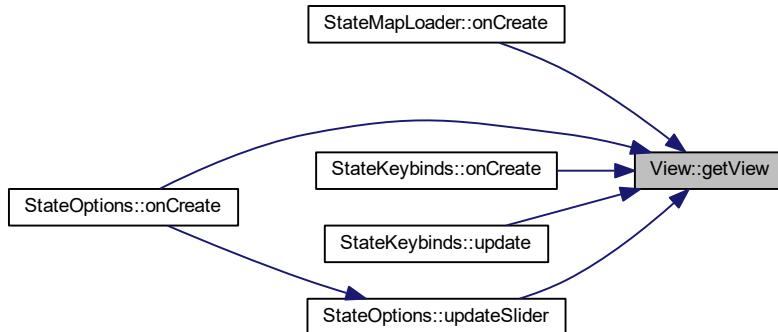
Here is the caller graph for this function:



### 10.67.2.4 getView()

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
auto& View< TWidgetPtr >::getView ( ) const [inline]
```

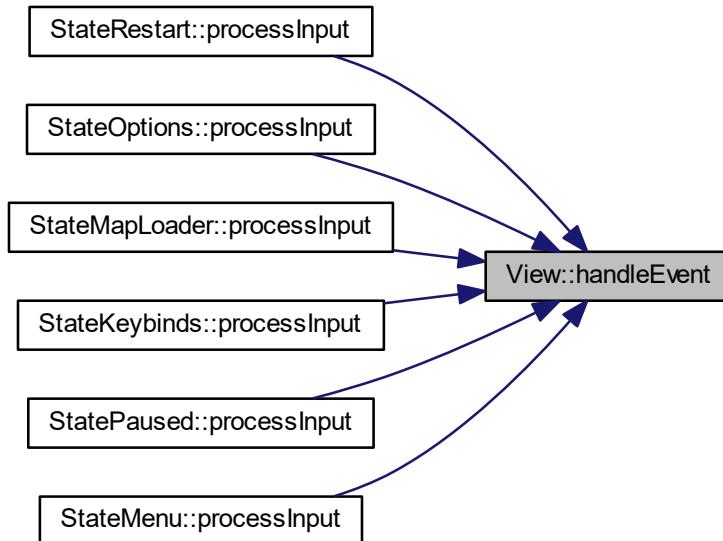
Here is the caller graph for this function:



### 10.67.2.5 handleEvent()

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
virtual void View< TWidgetPtr >::handleEvent (
    std::queue< sf::Event > & events ) [inline], [virtual]
```

Here is the caller graph for this function:



### 10.67.2.6 operator=() [1/2]

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
View& View< TWidgetPtr >::operator= (
    const View< TWidgetPtr > & ) [default]
```

### 10.67.2.7 operator=() [2/2]

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
View& View< TWidgetPtr >::operator= (
    View< TWidgetPtr > && ) [default], [noexcept]
```

### 10.67.3 Member Data Documentation

#### 10.67.3.1 buttonsCounter

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
int View< TWidgetPtr >::buttonsCounter {0} [protected]
```

#### 10.67.3.2 view

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
tgui::Gui View< TWidgetPtr >::view [protected]
```

#### 10.67.3.3 widgets

```
template<typename TWidgetPtr = tgui::Widget::Ptr>
std::vector<TWidgetPtr> View< TWidgetPtr >::widgets
```

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

- [View.h](#)

## 10.68 Window Class Reference

Responsible for drawing onto the screen.

```
#include <Window.h>
```

Collaboration diagram for Window:

| Window   |
|--|
| - window<br>- events   |
| + Window()<br>+ update()<br>+ beginDraw()<br>+ draw()<br>+ endDraw()<br>+ close()<br>+ isOpen()<br>+ updateView()<br>+ getEvent()<br>+ getWindow()<br>+ getWidth()<br>+ getHeight()<br>- setWindowIcon() |

## Public Member Functions

- `Window (std::string_view windowName)`  
*Constructs a `Window` with a given `windowName`.*
- `void update ()`  
*Updates the window events.*
- `void beginDraw ()`  
*Starts the drawing procedure.*
- `void draw (const sf::Drawable &drawable)`  
*Draws the drawable objects onto the screen.*
- `void endDraw ()`  
*Ends the drawing procedure.*
- `void close ()`  
*Closes the `window`.*
- `bool isOpen () const`
- `void updateView (const sf::View &view)`
- `std::queue< sf::Event > & getEvent ()`  
*Retrieve the `events` queue.*
- `sf::RenderWindow & getWindow ()`
- `unsigned int getWidth () const noexcept`
- `unsigned int getHeight () const noexcept`

## Private Member Functions

- `void setWindowIcon ()`

## Private Attributes

- `sf::RenderWindow window`  
*At a lowest level, everything is drawn to this instance.*
- `std::queue< sf::Event > events`  
*Queue of Events cleaned up game every frame.*

### 10.68.1 Detailed Description

Responsible for drawing onto the screen.

A fundamental class that encapsulates `window` and its utilities.

#### Note

`Game` execution ends once the `window` is closed by calling `close()` method.

### 10.68.2 Constructor & Destructor Documentation

### 10.68.2.1 Window()

```
Window::Window (
    std::string_view windowName ) [explicit]
```

Constructs a [Window](#) with a given windowName.

Sets framerate limit to `config::maxFps` and `config::maxFps` and `::maxFps` and `#maxFps` and `config::maxFps` and `config::maxFps` and `config::maxFps` and `setWindowIcon()` which may throw if no icon image is found.

When `setWindowIcon()` throws, the default operating system icon is loaded.

#### Note

The game works in a **frame-rate independent way** which means that the game works correctly in 10, 60 or 3000 FPS.

This game works in 3000 FPS on my slow computer when the maps are small. :-)

## 10.68.3 Member Function Documentation

### 10.68.3.1 beginDraw()

```
void Window::beginDraw ( )
```

Starts the drawing procedure.

Effectively: clears the screen with a color.

Called from [Game draw\(\)](#) method. Here is the caller graph for this function:



### 10.68.3.2 close()

```
void Window::close ( )
```

Closes the [window](#).

#### Warning

Calling this method effectively means ending the [Game loop](#), and hence finish the program.

### 10.68.3.3 draw()

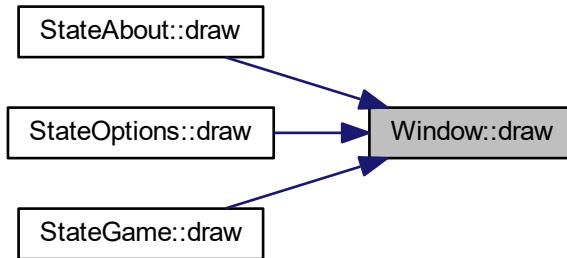
```
void Window::draw (
    const sf::Drawable & drawable )
```

Draws the drawable objects onto the screen.

**Parameters**

|                 |  |
|-----------------|--|
| <i>drawable</i> | A valid drawable entity Unlike <a href="#">beginDraw()</a> and <a href="#">endDraw()</a> , this method is called from <a href="#">StateMachine</a> |
|-----------------|--|

Here is the caller graph for this function:



#### 10.68.3.4 `endDraw()`

```
void Window::endDraw( )
```

Ends the drawing procedure.

Effectively: displays the rendered [window](#).

Called from [Game draw\(\)](#) method. Here is the caller graph for this function:



### 10.68.3.5 `getEvent()`

```
std::queue< sf::Event > & Window::getEvent ( )
```

Retrieve the `events` queue.

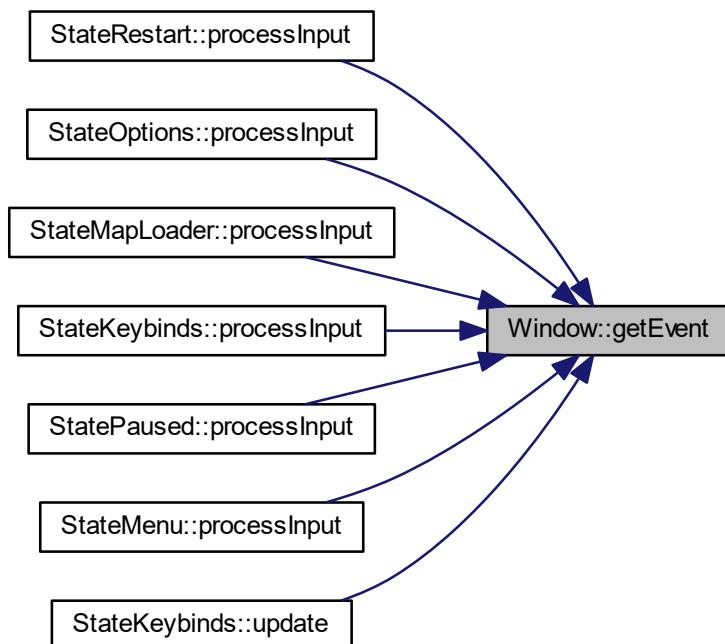
Called every frame by all classes having their own `View` gui instance.

Ensures correct drawing behaviour of `View` gui instances.

#### Note

`StateGame` and `StateAbout` do not use `View` gui instance.

Here is the caller graph for this function:



### 10.68.3.6 `getHeight()`

```
unsigned int Window::getHeight ( ) const [noexcept]
```

Get the width of the `window`.

#### Returns

The window height.

### 10.68.3.7 getWidth()

```
unsigned int Window::getWidth ( ) const [noexcept]
```

Get the width of the [window](#).

#### Returns

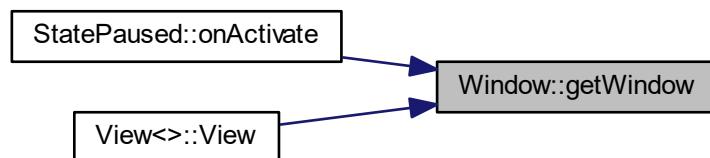
The window width.

### 10.68.3.8 getWindow()

```
sf::RenderWindow & Window::getWindow ( )
```

Retrieve the [window](#).

The retrieved [window](#) is an instance of [RenderTarget](#). Here is the caller graph for this function:



### 10.68.3.9 isOpen()

```
bool Window::isOpen ( ) const
```

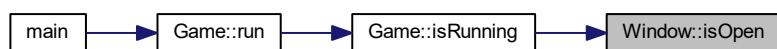
Check if [window](#) is still opened.

A condition for ending the main game loop.

#### Returns

Is window opened?

Here is the caller graph for this function:



### 10.68.3.10 setWindowIcon()

```
void Window::setWindowIcon ( ) [private]
```

Sets the application's window icon.

This method throws when no icon is found. In such a case default Operating System icon is set.

### 10.68.3.11 update()

```
void Window::update ( )
```

Updates the window events.

Inserts all gathered events since previous frame into [events](#) queue and checks if the user tried closing the window.

#### Note

A dedicated #queue is needed because there can be more than two windows within a frame, and handling them is done in a lower abstraction level classes.

Here is the caller graph for this function:



### 10.68.3.12 updateView()

```
void Window::updateView (
    const sf::View & view )
```

Updates the [window](#) view with a possibly new [Camera](#) view position.

Called every frame. Called only inside [StateGame](#).

#### Note

: Only a part of a World is displayed to the screen.

#### Parameters

|                   |   |
|-------------------|---|
| <code>view</code> | A new view of <a href="#">Camera</a> class. |
|-------------------|---|

Here is the caller graph for this function:



## 10.68.4 Member Data Documentation

### 10.68.4.1 events

```
std::queue<sf::Event> Window::events [private]
```

Queue of Events cleaned up game every frame.

### 10.68.4.2 window

```
sf::RenderWindow Window::window [private]
```

At a lowest level, everything is drawn to this instance.

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

- [Window.h](#)
- [Window.cpp](#)

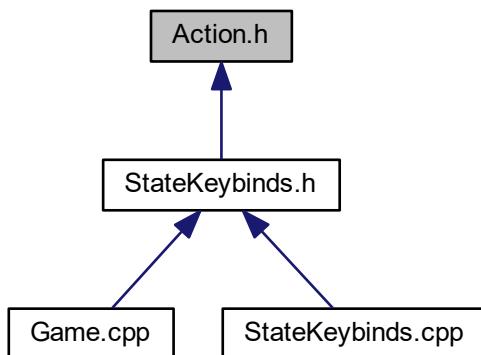


# Chapter 11

## File Documentation

### 11.1 Action.h File Reference

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



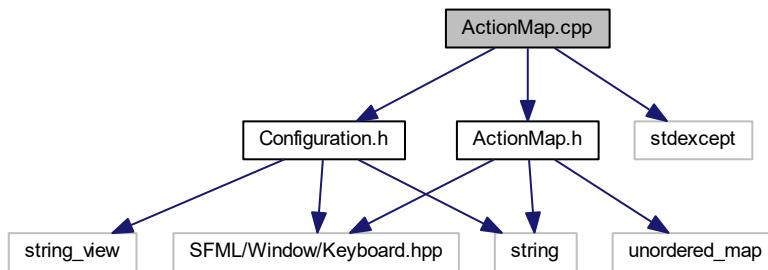
### Enumerations

- enum `Action` {  
  `Action::Jump,`  
  `Action::GoLeft,`  
  `Action::GoRight,`  
  `Action::NONE }`

*An enum containing available game actions taken by the player during the game.*

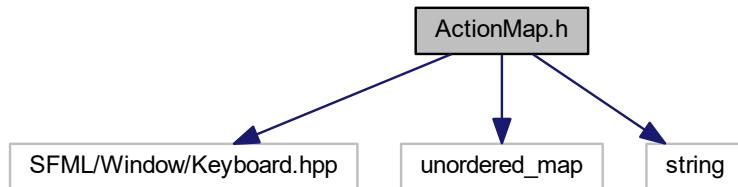
## 11.2 ActionMap.cpp File Reference

```
#include "ActionMap.h"
#include <stdexcept>
#include "Configuration.h"
Include dependency graph for ActionMap.cpp:
```

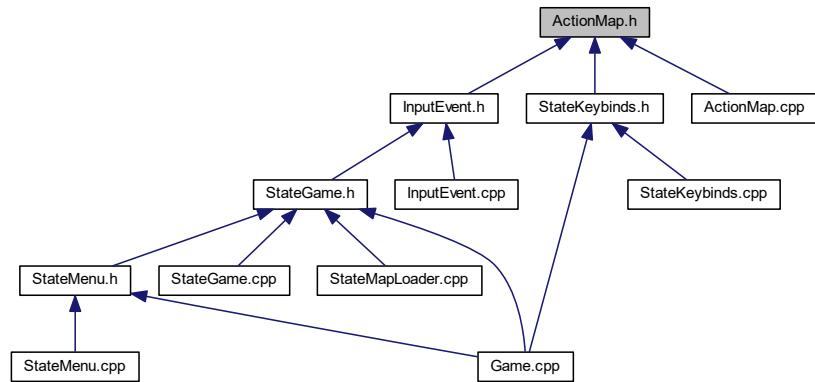


## 11.3 ActionMap.h File Reference

```
#include <SFML/Window/Keyboard.hpp>
#include <unordered_map>
#include <string>
Include dependency graph for ActionMap.h:
```



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

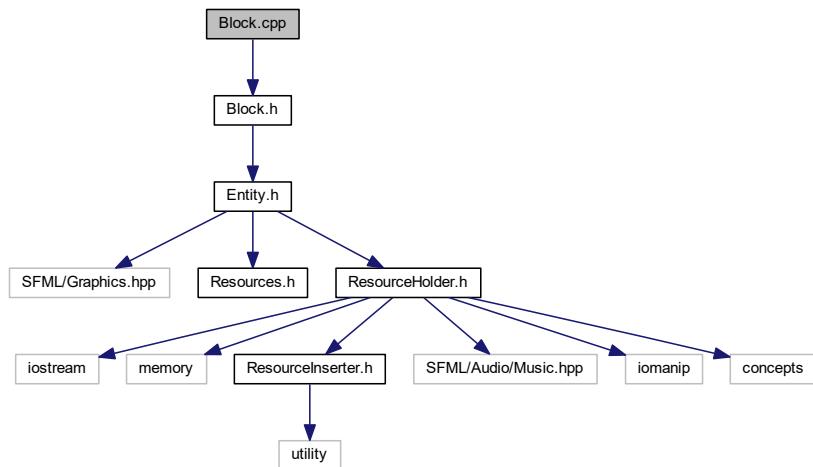


## Classes

- class [ActionMap](#)

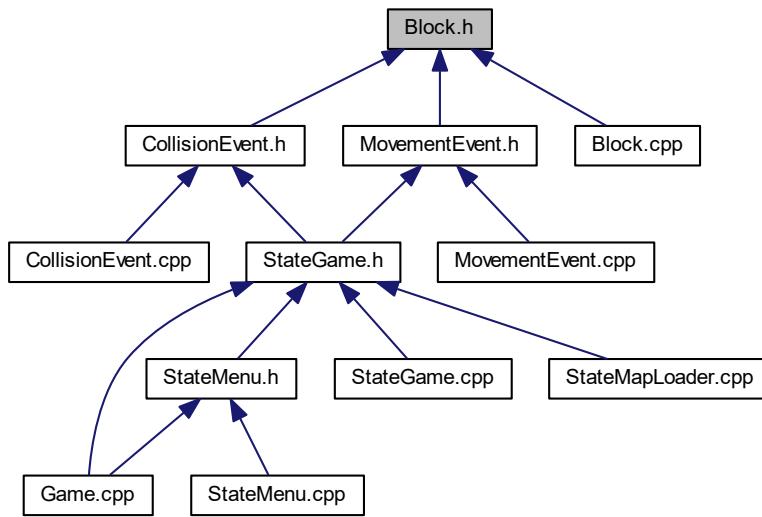
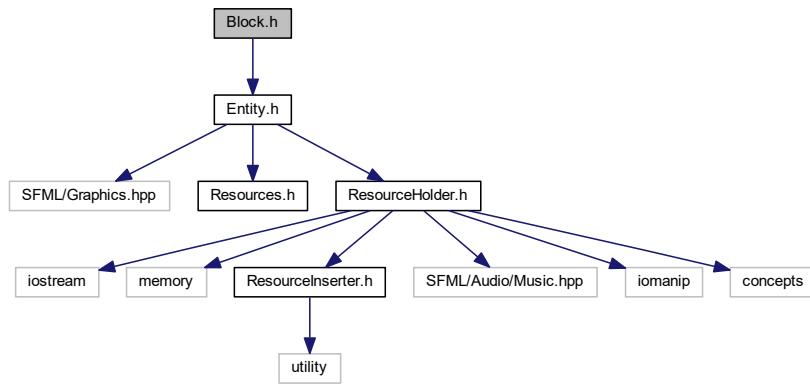
## 11.4 Block.cpp File Reference

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



## 11.5 Block.h File Reference

```
#include "Entity.h"
Include dependency graph for Block.h:
```



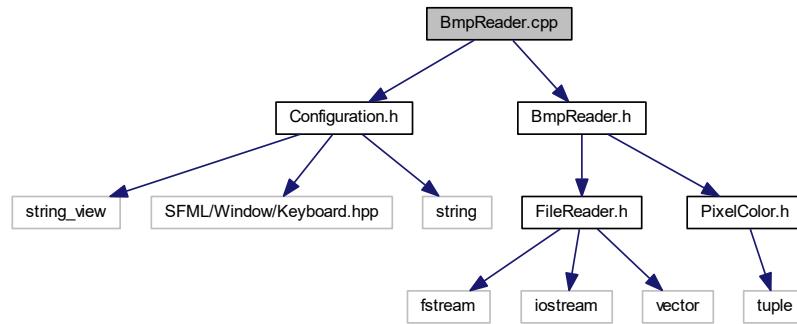
### Classes

- class [Block](#)

*Used to represent collidable game blocks.*

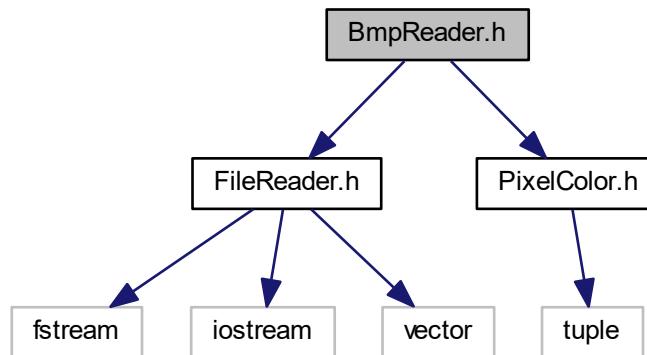
## 11.6 BmpReader.cpp File Reference

```
#include <Configuration.h>
#include "BmpReader.h"
Include dependency graph for BmpReader.cpp:
```

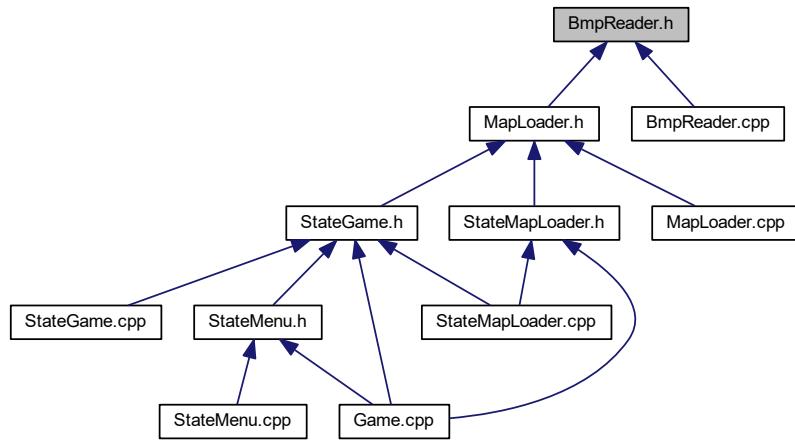


## 11.7 BmpReader.h File Reference

```
#include "FileReader.h"
#include "PixelColor.h"
Include dependency graph for BmpReader.h:
```



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



## Classes

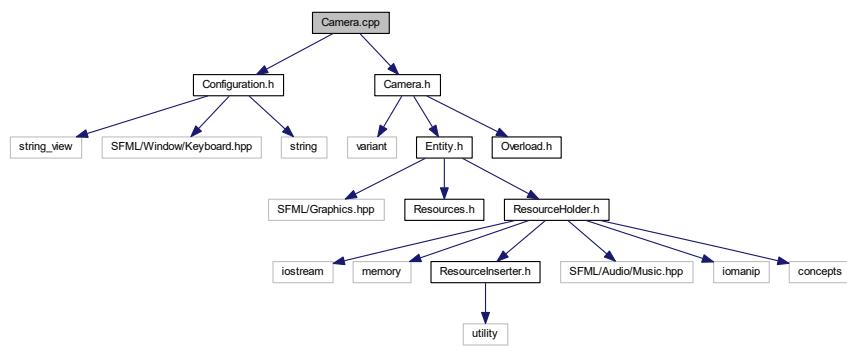
- class [BmpReader](#)

*Reads the .bmp file into the vector of PixelColor.*

## 11.8 Camera.cpp File Reference

```
#include <Configuration.h>
#include "Camera.h"
```

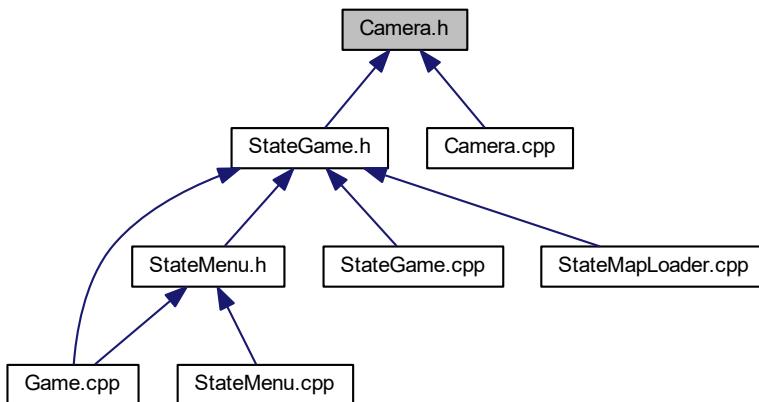
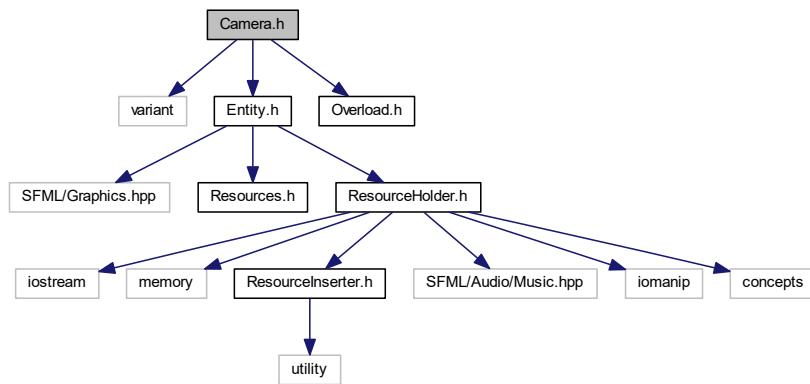
Include dependency graph for `Camera.cpp`:



## 11.9 Camera.h File Reference

```
#include <variant>
#include "Entity.h"
```

```
#include "Overload.h"
Include dependency graph for Camera.h:
```



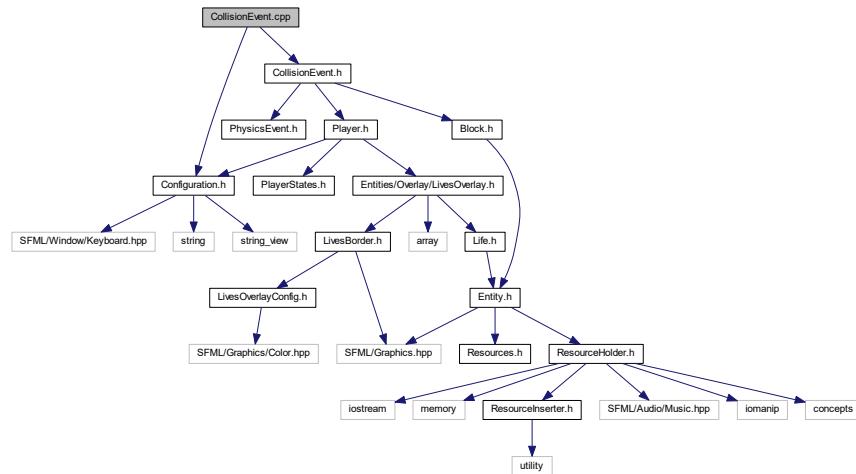
## Classes

- class [Camera](#)  
*Responsible for viewing only a sensible, displayable part of the World to the screen. Handles all [Camera](#) collisions.*
- struct [Camera::None](#)  
*Empty struct denoting there was previously [None](#) collision.*
- struct [Camera::Left](#)  
*Empty struct denoting previously solved on other axis was [Left](#) collision.*
- struct [Camera::Right](#)  
*Empty struct denoting previously solved on other axis was [Right](#) collision.*
- struct [Camera::Top](#)  
*Empty struct denoting previously solved on other axis was [Top](#) collision.*
- struct [Camera::Bot](#)  
*Empty struct denoting previously solved on other axis was [Bot](#) collision.*

## 11.10 CMakeLists.txt File Reference

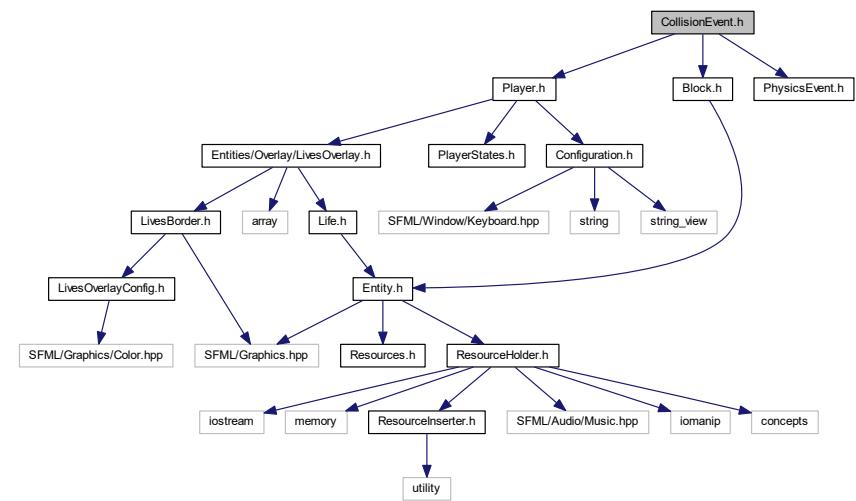
## 11.11 CollisionEvent.cpp File Reference

```
#include <Configuration.h>
#include "CollisionEvent.h"
Include dependency graph for CollisionEvent.cpp:
```

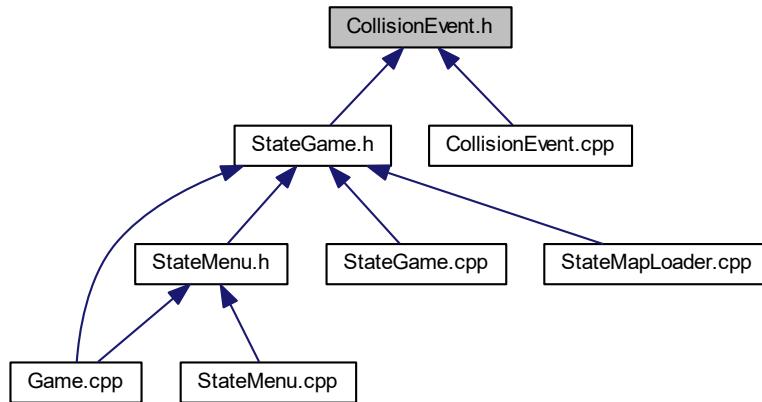


## 11.12 CollisionEvent.h File Reference

```
#include "Player.h"
#include "Block.h"
#include "PhysicsEvent.h"
Include dependency graph for CollisionEvent.h:
```



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



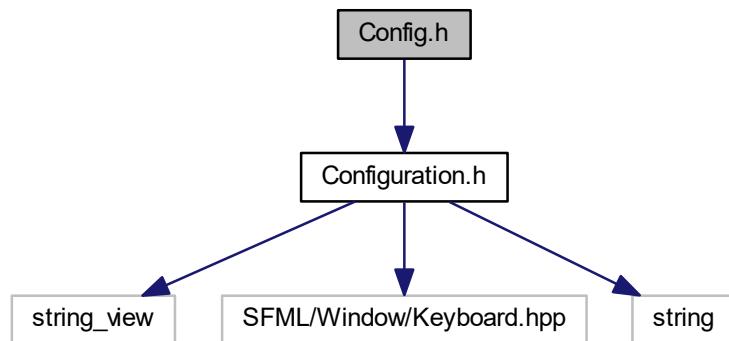
## Classes

- class [CollisionEvent](#)

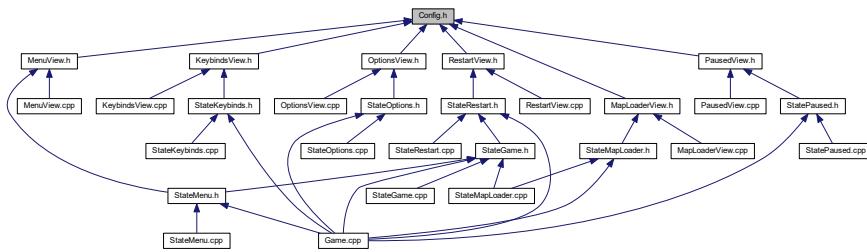
*CollisionEvent* class responsible for solves player<->collidable collisions in **axis-independent way**.

## 11.13 Config.h File Reference

```
#include "Configuration.h"
Include dependency graph for Config.h:
```



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



## Classes

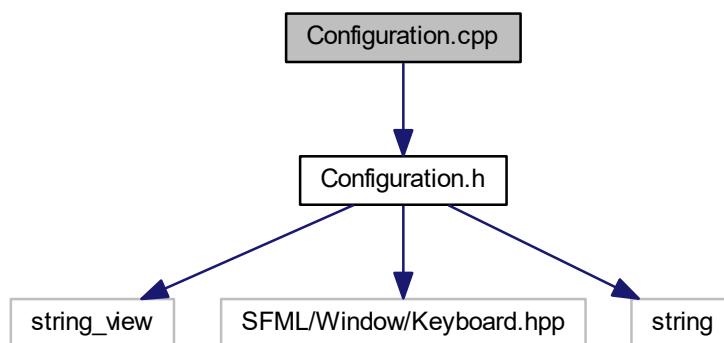
- struct [Gui::Config< Widget >](#)  
*A shared gui config. Contains common settings.*

## Namespaces

- [Gui](#)  
*Encapsulate common global [View Config](#) within an additional namespace.*

## 11.14 Configuration.cpp File Reference

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

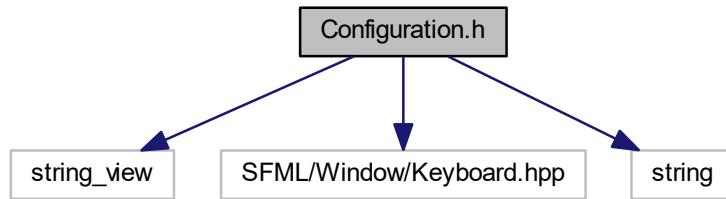


## Namespaces

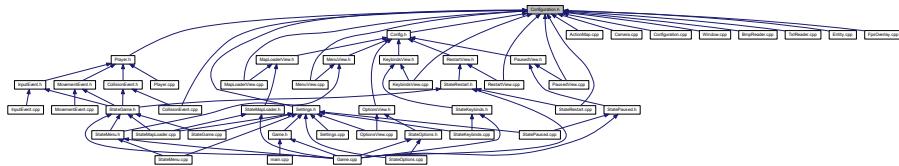
- [config](#)  
*Compile time customizable settings.*

## 11.15 Configuration.h File Reference

```
#include <string_view>
#include <SFML/Window/Keyboard.hpp>
#include <string>
Include dependency graph for Configuration.h:
```



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



## Namespaces

- `config`  
*Compile time customizable settings.*
- `config::bg`

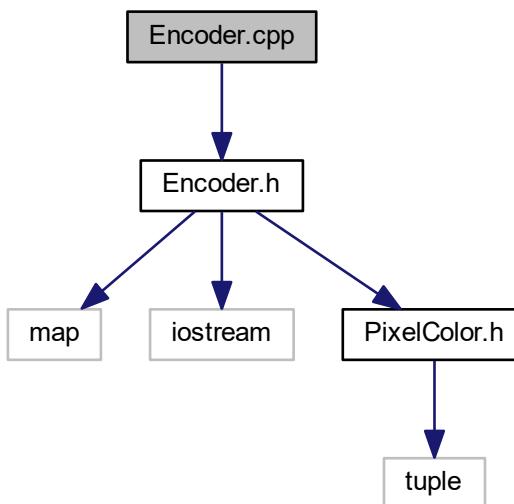
## Variables

- `constexpr static Keyboard::Key config::defaultJumpKey = Keyboard::W`
- `constexpr static Keyboard::Key config::defaultRunLeftKey = Keyboard::A`
- `constexpr static Keyboard::Key config::defaultRunRightKey = Keyboard::D`
- `constexpr static string_view config::mapEditorPath = "index.html"`
- `constexpr static string_view config::defaultMapName = "map.bmp"`
- `constexpr static string_view config::gameMusic = "../resources/music/Background.ogg"`
- `constexpr static auto config::defaultSoundVolume = 30.0f`
- `constexpr static string_view config::widgetsFontName = "../resources/Fonts/CascadiaCode.ttf"`
- `constexpr static string_view config::keybindsFontName = "../resources/Fonts/NeuropolX.ttf"`
- `constexpr static string_view config::bg::options = "../resources/Backgrounds/Options.png"`
- `constexpr static string_view config::bg::keybinds = "../resources/Backgrounds/Keybinds.png"`
- `constexpr static string_view config::bg::loader = "../resources/Backgrounds/Loader.jpg"`
- `constexpr static string_view config::bg::menu = "../resources/Backgrounds/Main_Menu.jpg"`

- `constexpr static string_view config::bg::paused = "../resources/Backgrounds/Paused.png"`
- `constexpr static string_view config::bg::restart = "../resources/Backgrounds/Restart_Game.png"`
- `constexpr static string_view config::programIcon = "../resources/Icons/Program_Icon.png"`
- `constexpr static string_view config::jumpIcon = "../resources/Icons/Jump.png"`
- `constexpr static string_view config::runLeftIcon = "../resources/Icons/Left_Arrow.png"`
- `constexpr static string_view config::runRightIcon = "../resources/Icons/Right_Arrow.png"`
- `constexpr static string_view config::largeBgIcon = "../resources/Icons/Keybinds_Large_Icon.png"`
- `constexpr static string_view config::sideBorder = "../resources/Misc/Options_Side_Border.png"`
- `constexpr static string_view config::pausedTextImage = "../resources/Misc/Game_Paused_Texture.png"`
- `constexpr static string_view config::rebindingBg = "../resources/Misc/Rebinding_Key.png"`
- `constexpr static auto config::horizontalVelocity = 300.0f`
- `constexpr static auto config::initialJumpVelocity = -680.0f`
- `constexpr static auto config::terminalVelocity = 1000.0f`
- `constexpr static auto config::gravity = 2000.0f`
- `constexpr static auto config::hitCeilingVelocity = 10.0f`
- `int config::blocksCountWidth = 0`
- `int config::blocksCountHeight = 0`
- `constexpr static auto config::entityWidth = 32`
- `constexpr static auto config::entityHeight = 32`
- `constexpr static auto config::windowWidth = entityWidth * 20`
- `constexpr static auto config::windowHeight = entityHeight * 18`
- `constexpr static auto config::maxFps = 3000`
- `bool config::playerWon = false`

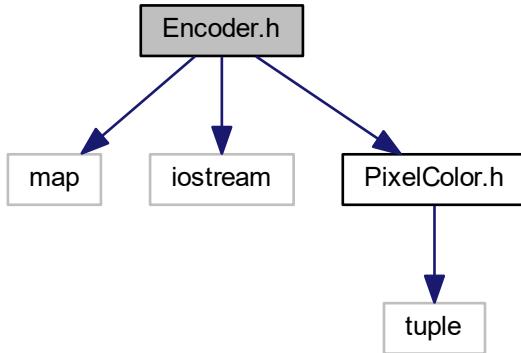
## 11.16 Encoder.cpp File Reference

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

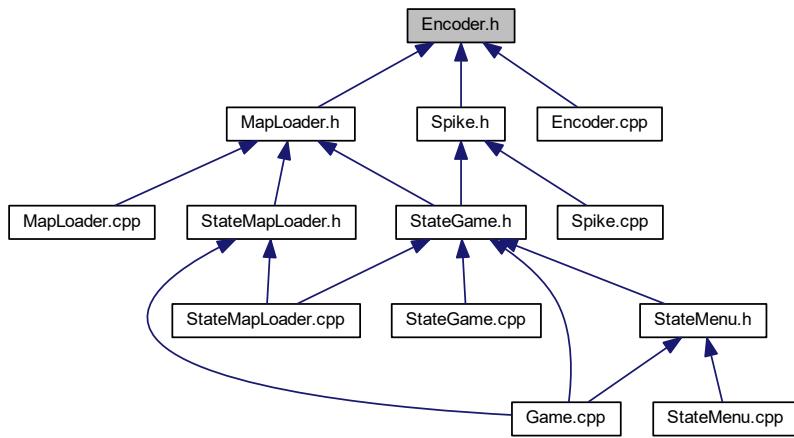


## 11.17 Encoder.h File Reference

```
#include <map>
#include <iostream>
#include "PixelColor.h"
Include dependency graph for Encoder.h:
```



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



## Classes

- class `Encoder< ReaderKey >`

*Encodes PixelColor (bmp) or int (txt) into Obj::Entity entities, depending on the variant type.*

## Namespaces

- [Obj](#)

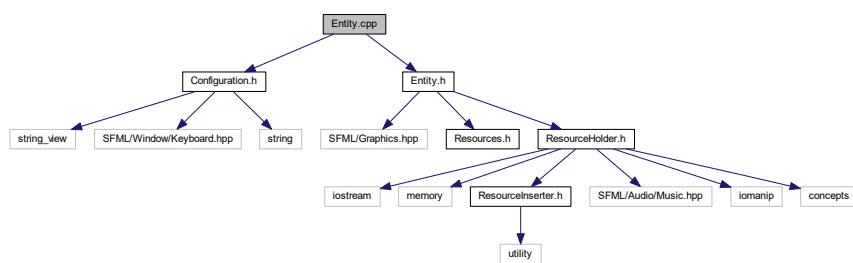
*Available game entity objects. Used only in [StateGame](#).*

## Enumerations

- enum [Obj::Entity](#) {
 [Obj::Entity::Empty](#),
 [Obj::Entity::Player](#),
 [Obj::Entity::Objective](#),
 [Obj::Entity::HeartCollectible](#),
 [Obj::Entity::BlockRed](#),
 [Obj::Entity::BlockBlue](#),
 [Obj::Entity::BlockBrown](#),
 [Obj::Entity::BlockGray](#),
 [Obj::Entity::BlockGreen](#),
 [Obj::Entity::BlockPurple](#),
 [Obj::Entity::BlockYellow](#),
 [Obj::Entity::BlockCrate](#),
 [Obj::Entity::Water1](#),
 [Obj::Entity::Water2](#),
 [Obj::Entity::Sign1](#),
 [Obj::Entity::Sign2](#),
 [Obj::Entity::Spike](#),
 [Obj::Entity::SpikeLeft](#),
 [Obj::Entity::SpikeRight](#),
 [Obj::Entity::SpikeTop](#)
}

## 11.18 Entity.cpp File Reference

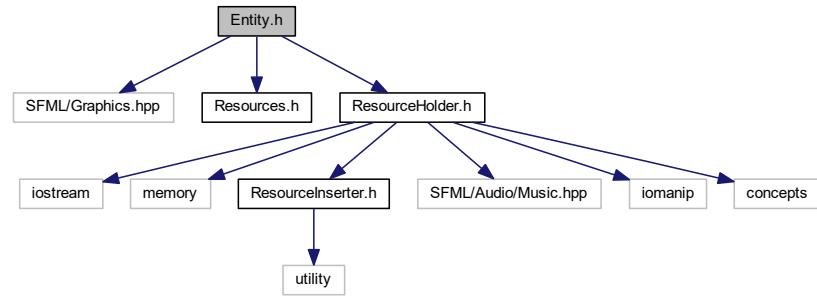
```
#include <Configuration.h>
#include "Entity.h"
Include dependency graph for Entity.cpp:
```



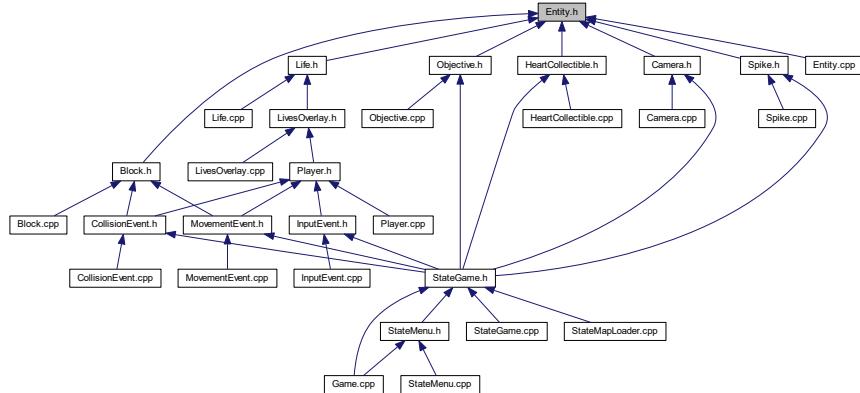
## 11.19 Entity.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Resources.h"
```

```
#include "ResourceHolder.h"
Include dependency graph for Entity.h:
```



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



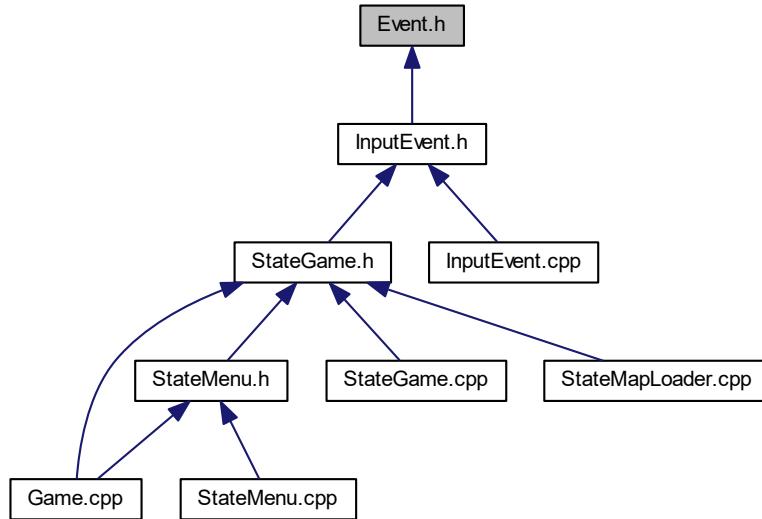
## Classes

- class [Entity](#)

A base drawable [Entity](#) class. All drawable entities (game objects) inherit from it.

## 11.20 Event.h File Reference

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



### Classes

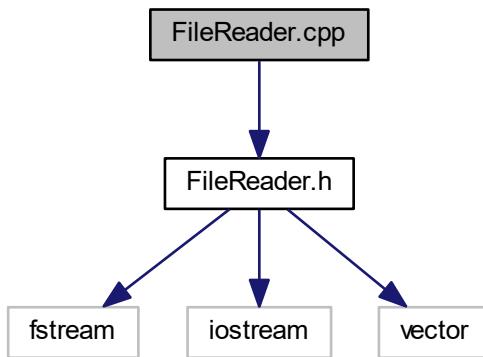
- class `Event`

`Event` class with `update()` pure virtual method handling trivial events.

## 11.21 FileReader.cpp File Reference

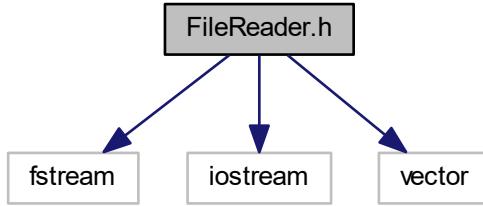
```
#include "FileReader.h"
```

Include dependency graph for `FileReader.cpp`:

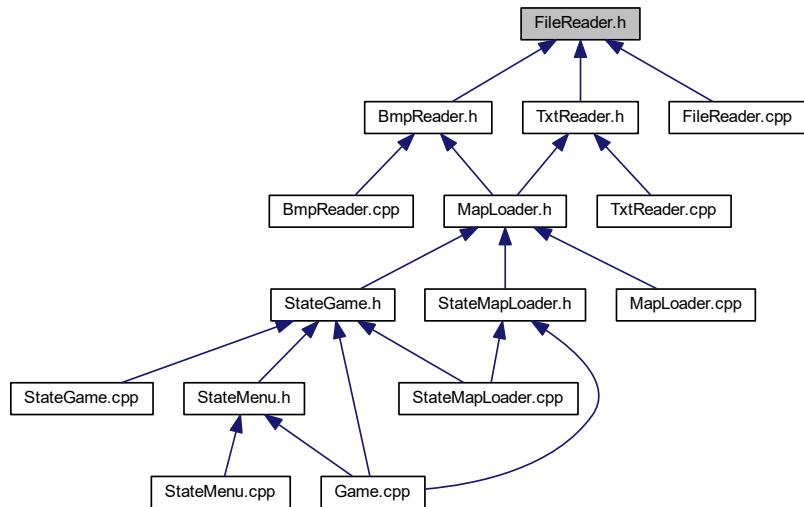


## 11.22 FileReader.h File Reference

```
#include <fstream>
#include <iostream>
#include <vector>
Include dependency graph for FileReader.h:
```



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



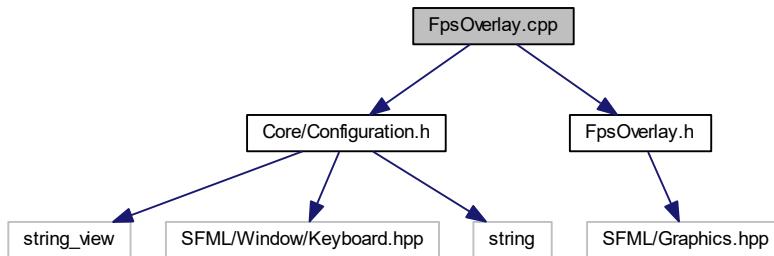
## Classes

- class [FileReader< T >](#)

*Abstract File Reader class template.*

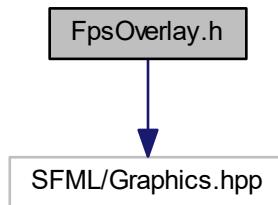
## 11.23 FpsOverlay.cpp File Reference

```
#include <Core/Configuration.h>
#include "FpsOverlay.h"
Include dependency graph for FpsOverlay.cpp:
```

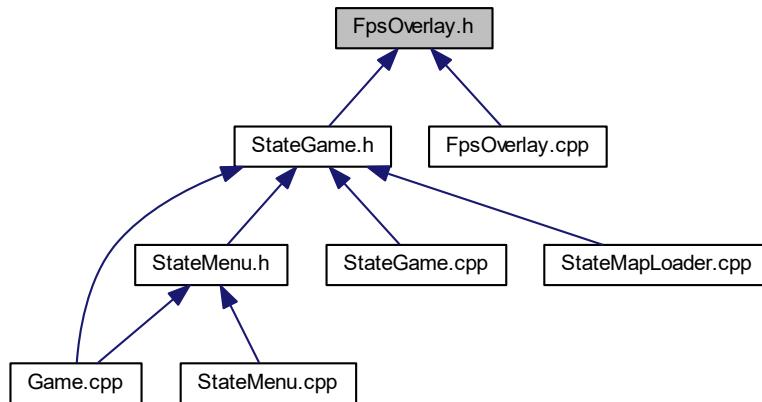


## 11.24 FpsOverlay.h File Reference

```
#include <SFML/Graphics.hpp>
Include dependency graph for FpsOverlay.h:
```



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



## Classes

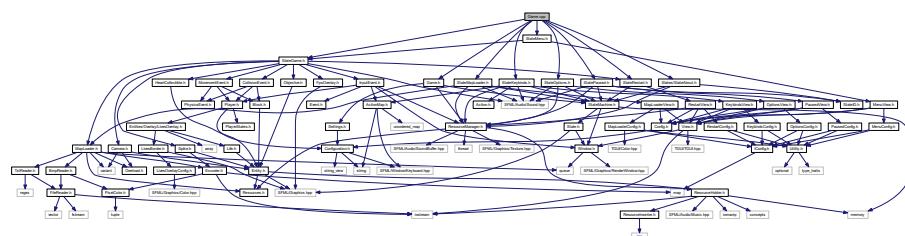
- class [MissingFont](#)  
*Custom exception class.*
- class [FpsOverlay](#)  
*Prints the Fps to the screen.*

## 11.25 Game.cpp File Reference

```

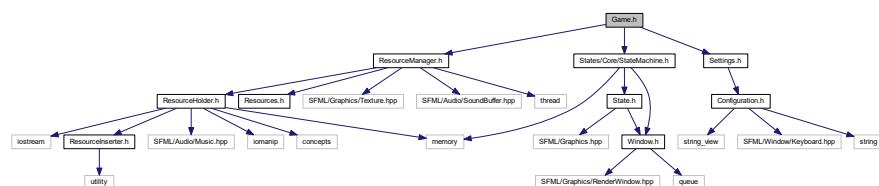
#include <StateGame.h>
#include <StateMenu.h>
#include <StateMapLoader.h>
#include <StateOptions.h>
#include <StatePaused.h>
#include <StateKeybinds.h>
#include <StateRestart.h>
#include <States/StateAbout.h>
#include "Game.h"
  
```

Include dependency graph for `Game.cpp`:

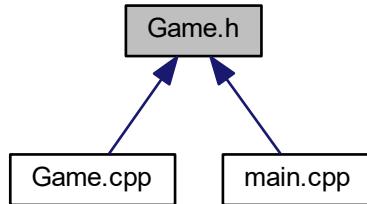


## 11.26 Game.h File Reference

```
#include "ResourceManager.h"
#include "States/Core/StateMachine.h"
#include "Settings.h"
Include dependency graph for Game.h:
```



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



### Classes

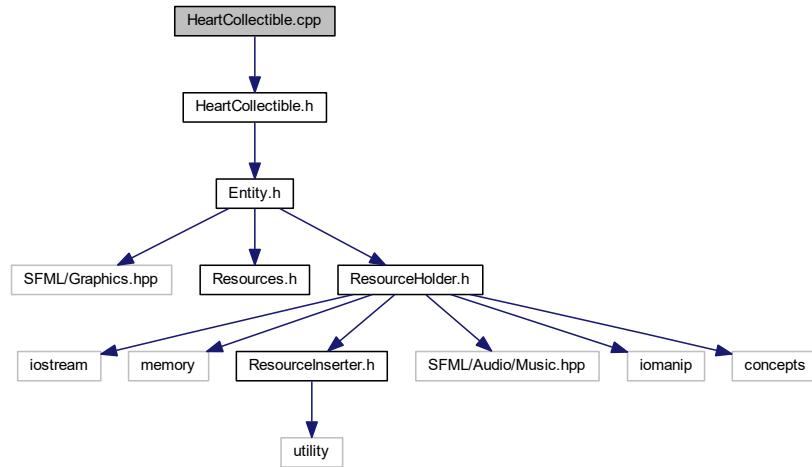
- class [Game](#)

*Container for player actions keybinds.*

## 11.27 HeartCollectible.cpp File Reference

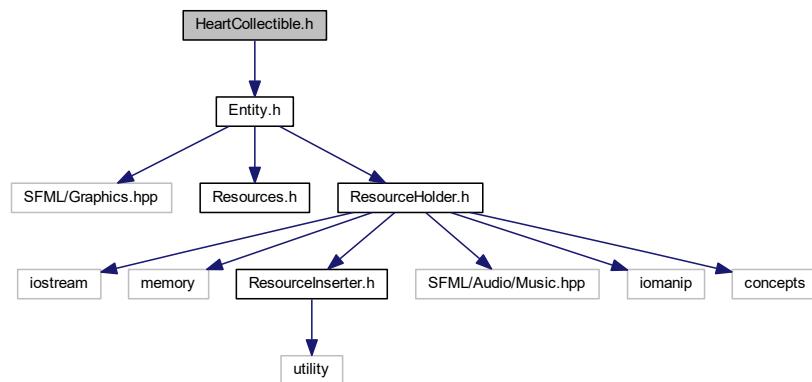
```
#include "HeartCollectible.h"
```

Include dependency graph for HeartCollectible.cpp:

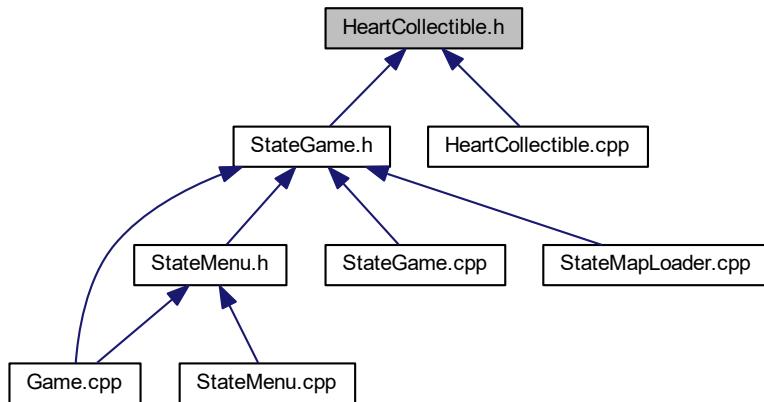


## 11.28 HeartCollectible.h File Reference

```
#include "Entity.h"
Include dependency graph for HeartCollectible.h:
```



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



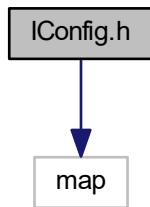
## Classes

- class [HeartCollectible](#)

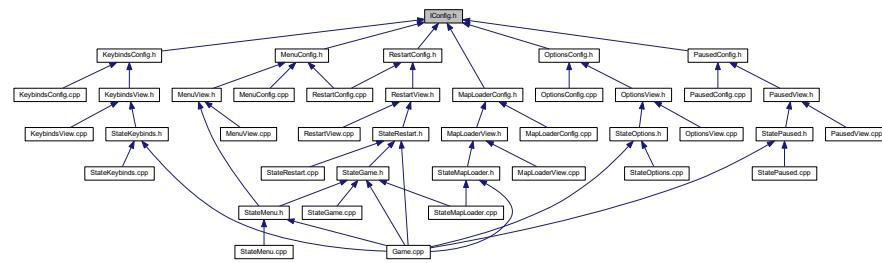
*Used to represent the collectible hearts (player lives).*

## 11.29 IConfig.h File Reference

```
#include <map>
Include dependency graph for IConfig.h:
```



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



## Classes

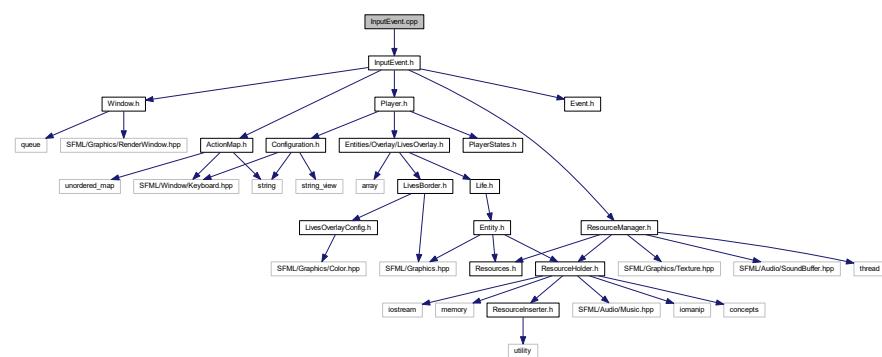
- struct `IConfig< E >`

*Every specific Config inherits from this struct.*

## 11.30 InputEvent.cpp File Reference

```
#include "InputEvent.h"
```

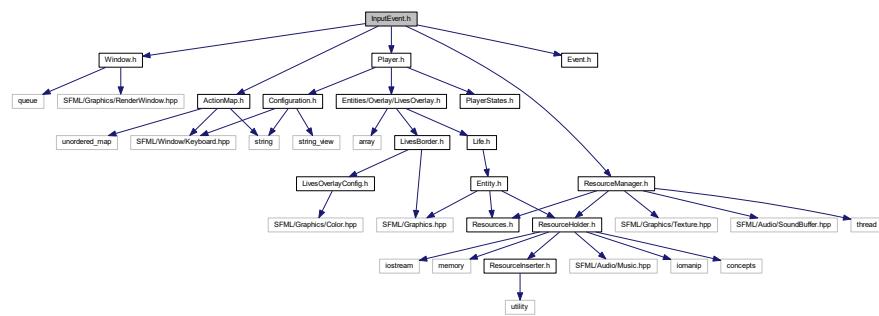
Include dependency graph for InputEvent.cpp:



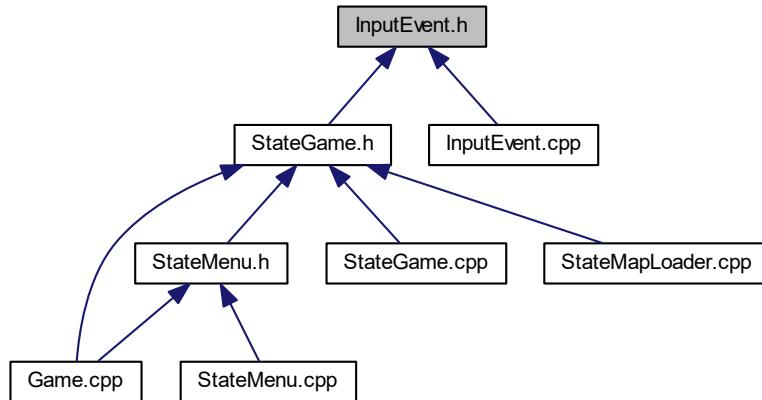
## 11.31 InputEvent.h File Reference

```
#include <Window.h>
#include <ActionMap.h>
#include "Player.h"
#include "ResourceManager.h"
```

```
#include "Event.h"
Include dependency graph for InputEvent.h:
```



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



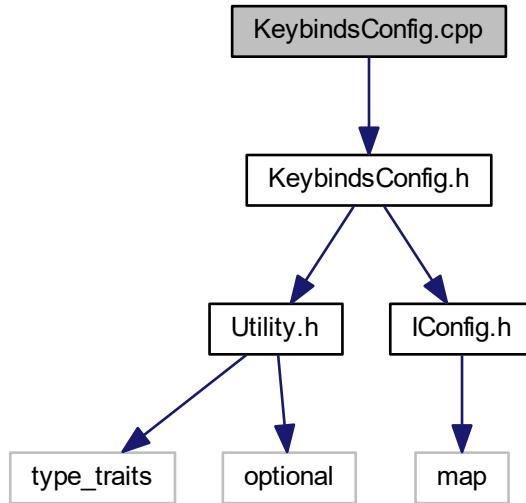
## Classes

- class [InputEvent](#)  
*Handles the user input within `StateGame` class. Uses `ActionMap`.*

## 11.32 KeybindsConfig.cpp File Reference

```
#include "KeybindsConfig.h"
```

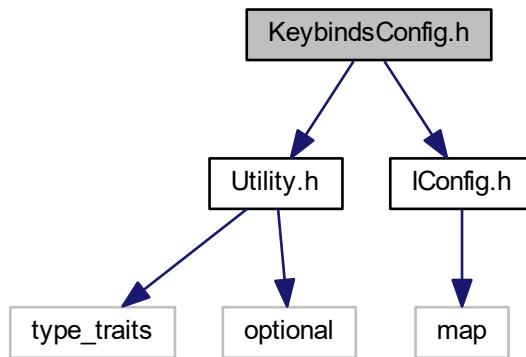
Include dependency graph for KeybindsConfig.cpp:



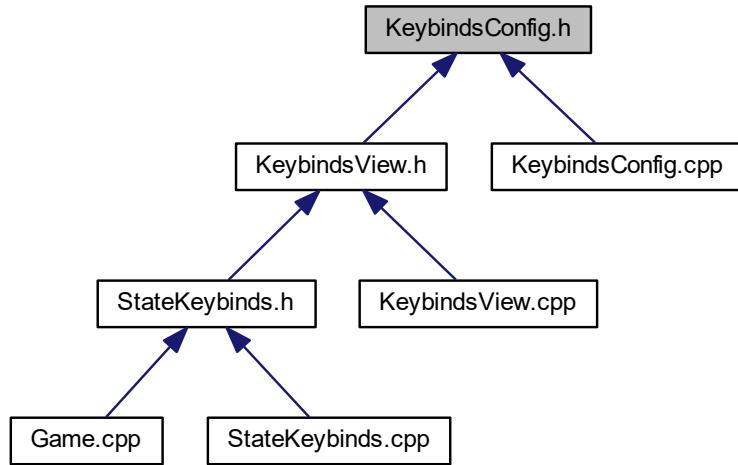
### 11.33 KeybindsConfig.h File Reference

```
#include "Utility.h"
#include "IConfig.h"
```

Include dependency graph for KeybindsConfig.h:



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



## Classes

- struct [Keybinds::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [KeybindsView](#).*

## Namespaces

- [Keybinds](#)

*Namespace designated for [Keybinds](#) view.*

## Enumerations

- enum [Keybinds::Btn](#) {
 [Keybinds::Btn::Jump](#),
 [Keybinds::Btn::RunLeft](#),
 [Keybinds::Btn::RunRight](#),
 [Keybinds::Btn::GoBack](#),
 [Keybinds::Btn::SIZE](#) }

*Keybinds buttons IDs.*

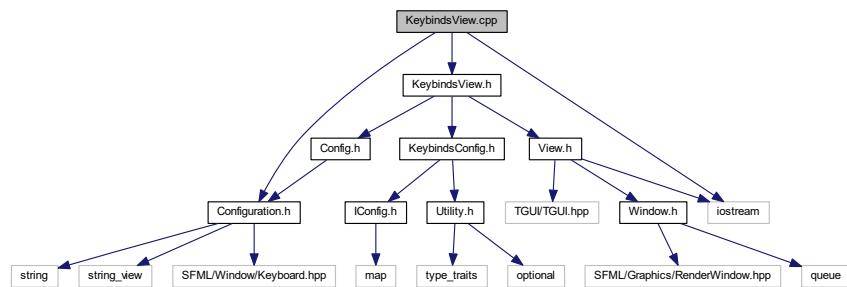
## Variables

- `constexpr std::initializer_list<Keybinds::Btn> Keybinds::Buttons`

*An additional initializer list to help mapping Btn names to Strings.*

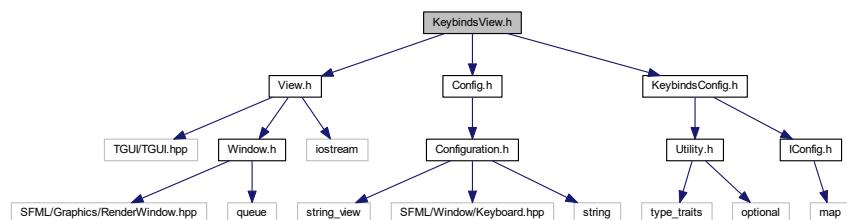
## 11.34 KeybindsView.cpp File Reference

```
#include "Configuration.h"
#include "KeybindsView.h"
#include <iostream>
Include dependency graph for KeybindsView.cpp:
```

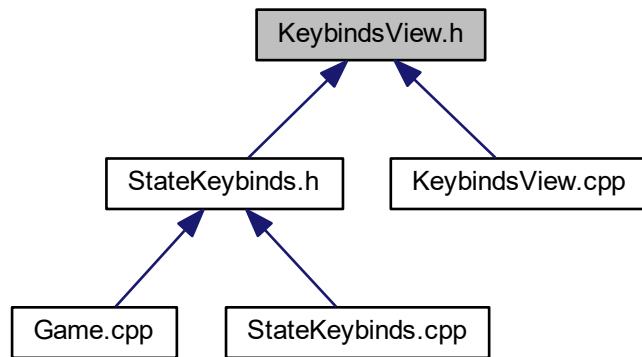


## 11.35 KeybindsView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "KeybindsConfig.h"
Include dependency graph for KeybindsView.h:
```



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



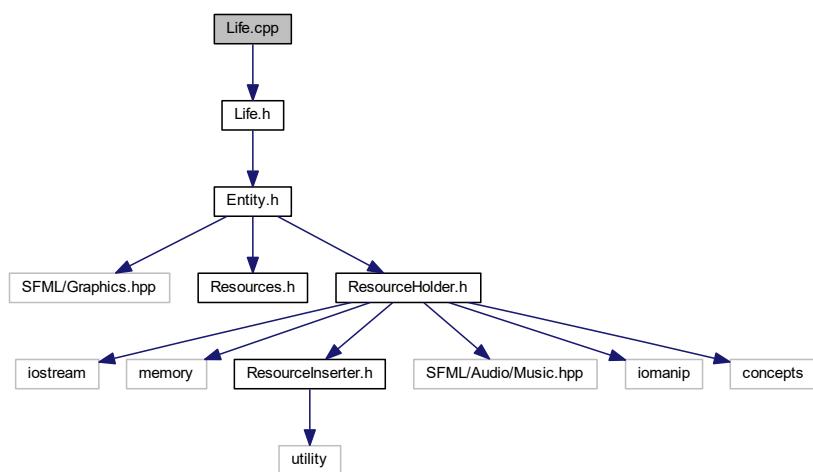
## Classes

- class [KeybindsView](#)  
*View class used to draw gui within StateKeybinds.*

## 11.36 Life.cpp File Reference

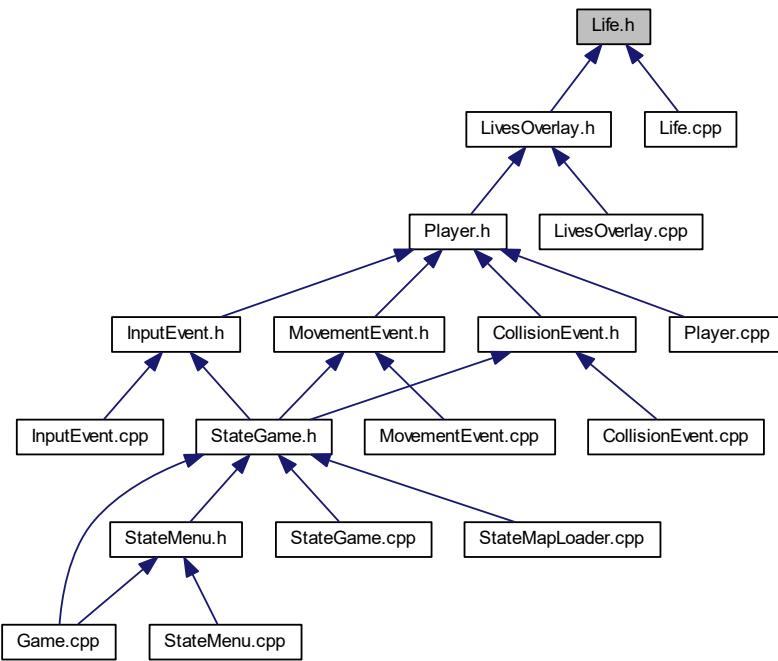
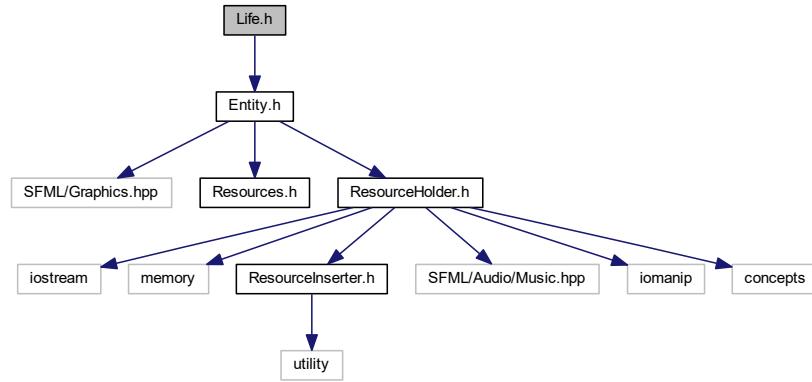
```
#include "Life.h"
```

Include dependency graph for `Life.cpp`:



## 11.37 Life.h File Reference

```
#include "Entity.h"
Include dependency graph for Life.h:
```



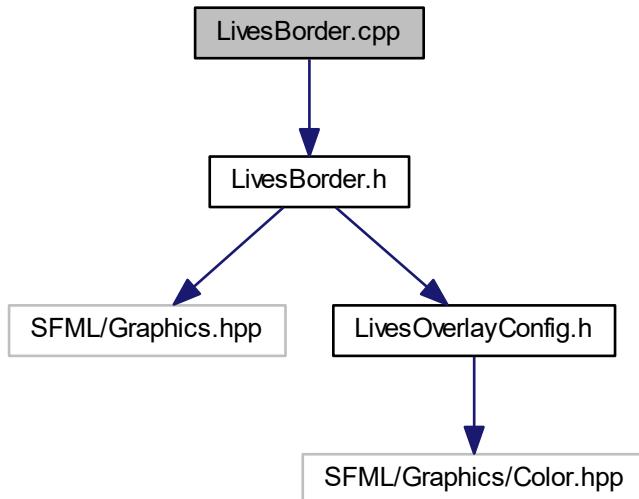
## Classes

- class [Life](#)

*Used to represent the player life overlay item as a hearts in corner of the screen(player lives).*

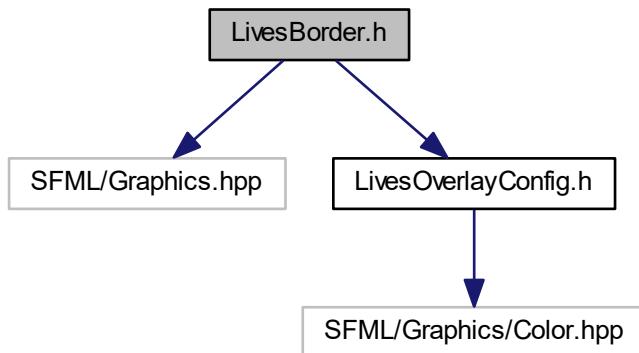
## 11.38 LivesBorder.cpp File Reference

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

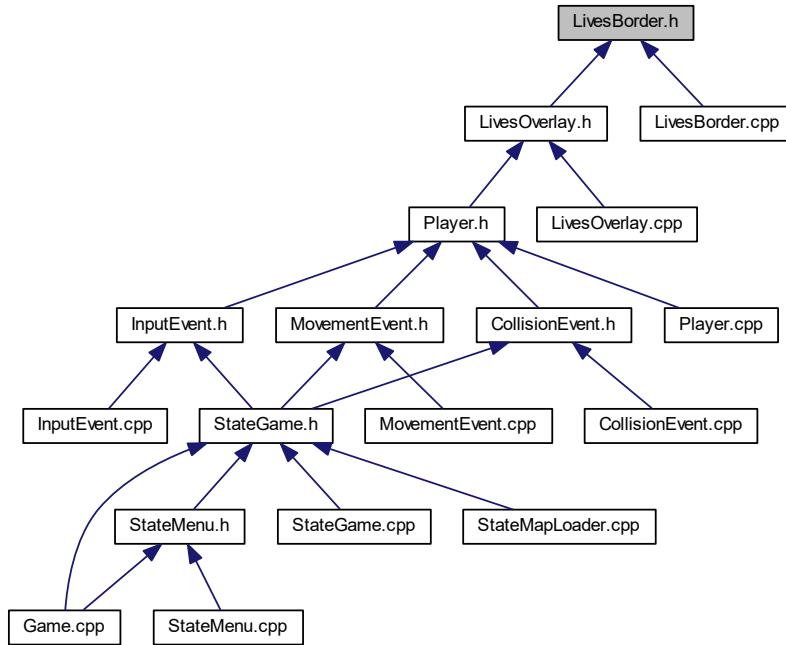


## 11.39 LivesBorder.h File Reference

```
#include <SFML/Graphics.hpp>  
#include "LivesOverlayConfig.h"  
Include dependency graph for LivesBorder.h:
```



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



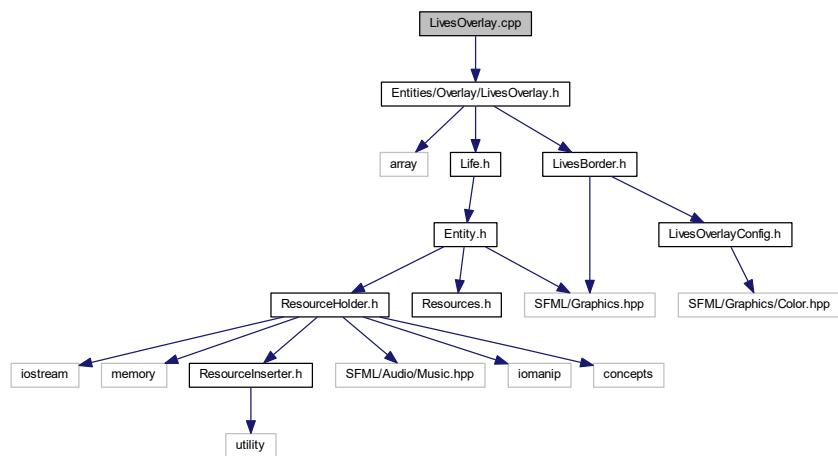
## Classes

- class [LivesBorder](#)

*A border around the [LivesOverlay](#).*

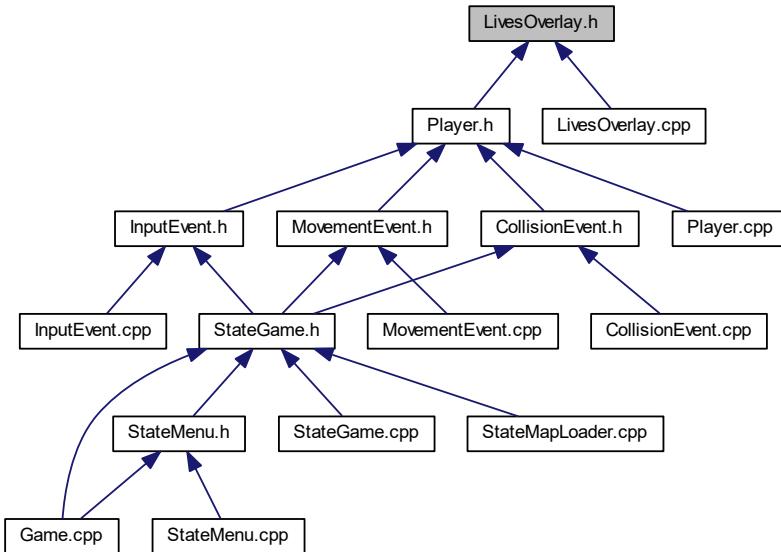
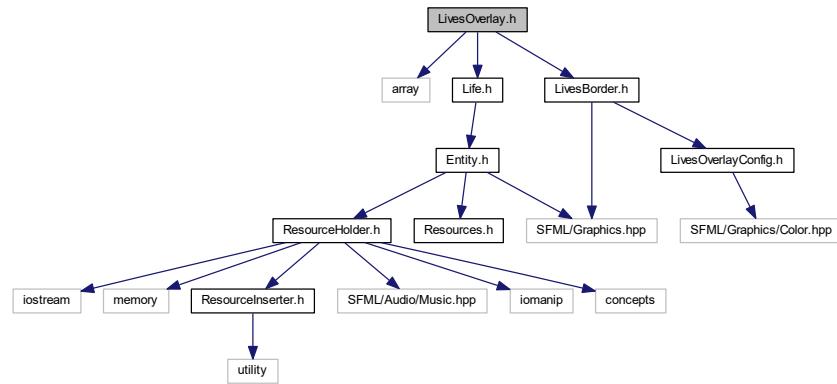
## 11.40 LivesOverlay.cpp File Reference

```
#include "Entities/Overlay/LivesOverlay.h"
Include dependency graph for LivesOverlay.cpp:
```



## 11.41 LivesOverlay.h File Reference

```
#include <array>
#include "Life.h"
#include "LivesBorder.h"
Include dependency graph for LivesOverlay.h:
```



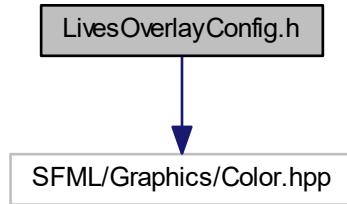
## Classes

- class [LivesOverlay](#)

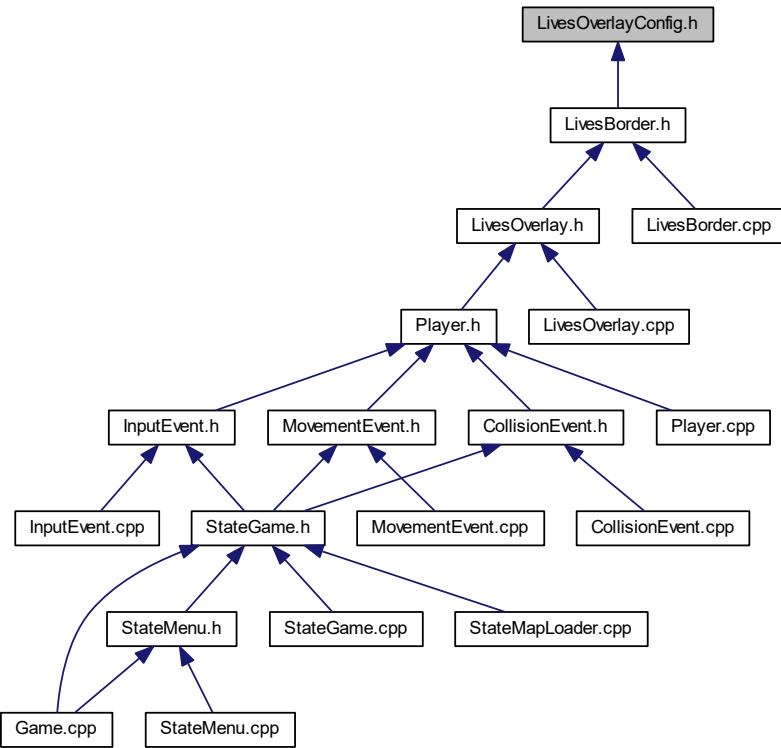
*Encapsulates player lives overlay.*

## 11.42 LivesOverlayConfig.h File Reference

```
#include <SFML/Graphics/Color.hpp>
Include dependency graph for LivesOverlayConfig.h:
```



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



## Namespaces

- `lives`

*Encapsulates lives overlay variables.*

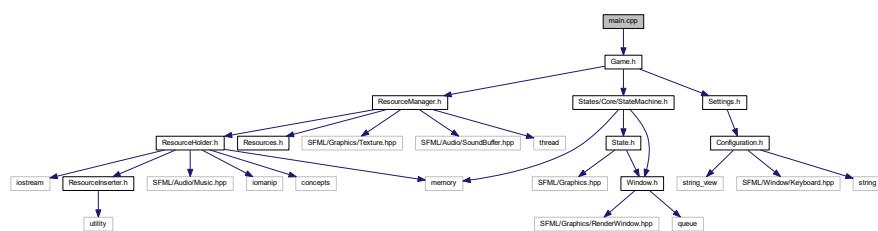
## Variables

- constexpr auto `lives::count` = 3
- constexpr auto `lives::boxSide` = 50.0f
- const sf::Color `lives::fillColor` = sf::Color{255, 255, 255, 140}
- const sf::Color `lives::borderColor` = sf::Color::Red
- constexpr auto `lives::borderThickness` = 2
- constexpr auto `lives::borderOffsetX` = 15
- constexpr auto `lives::borderOffsetY` = 15
- constexpr auto `lives::heartOffsetX` = 25
- constexpr auto `lives::heartOffsetY` = 25

## 11.43 main.cpp File Reference

```
#include "Game.h"
```

Include dependency graph for main.cpp:



## Functions

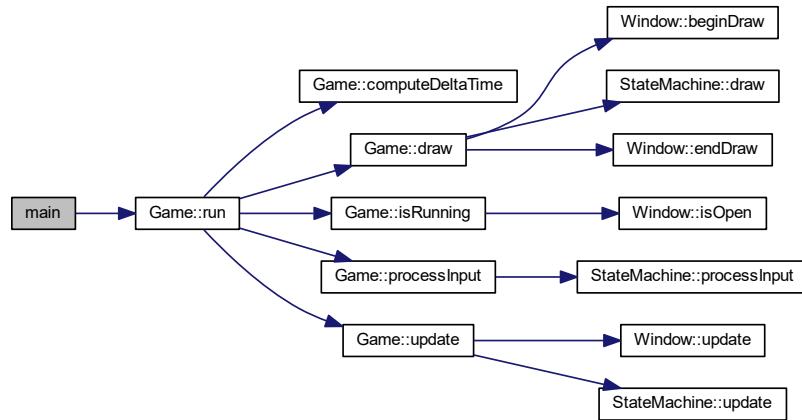
- int `main ([[maybe_unused]] int argc, [[maybe_unused]] char **argv)`

### 11.43.1 Function Documentation

#### 11.43.1.1 main()

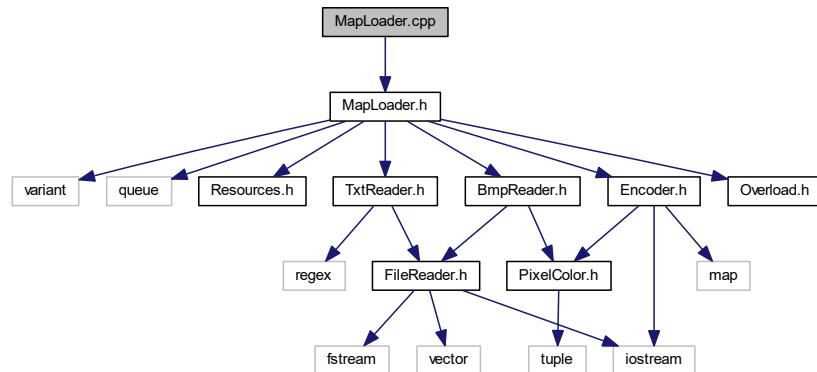
```
int main (
    [[maybe_unused]] int argc,
    [[maybe_unused]] char ** argv )
```

Here is the call graph for this function:



## 11.44 MapLoader.cpp File Reference

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

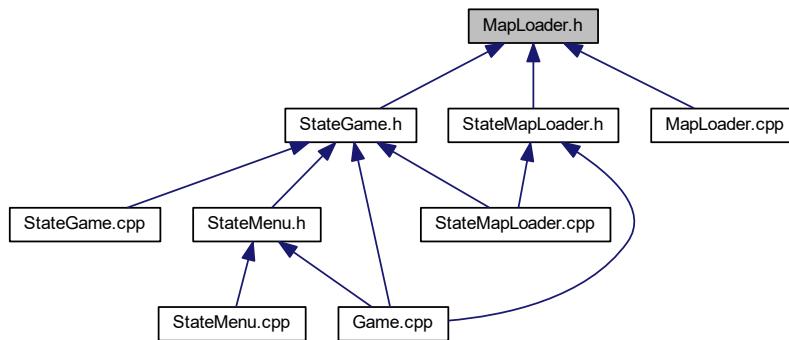
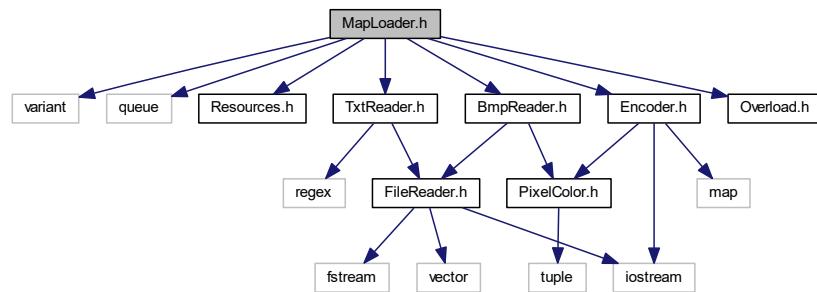


## 11.45 MapLoader.h File Reference

```
#include <variant>
#include <queue>
#include "Resources.h"
#include "BmpReader.h"
#include "TxtReader.h"
#include "Encoder.h"
```

```
#include "Overload.h"
```

Include dependency graph for MapLoader.h:



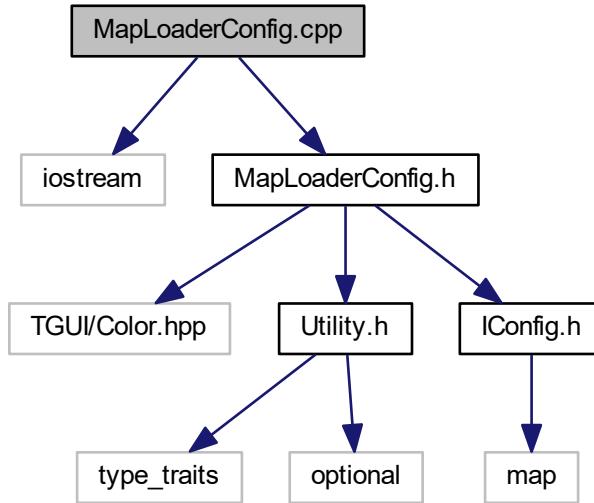
## Classes

- class [ILoader](#)  
*A base interface for a map loader.*
- struct [Bmp](#)  
*Empty struct used to determine file type in [MapLoader](#) variant.*
- struct [Txt](#)  
*Empty struct used to determine file type in [MapLoader](#) variant.*
- class [MapLoader< FileType >](#)  
*Load the map file.*

## 11.46 MapLoaderConfig.cpp File Reference

```
#include <iostream>
#include "MapLoaderConfig.h"
```

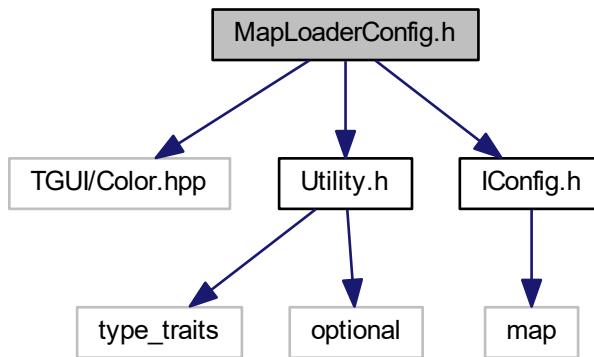
Include dependency graph for MapLoaderConfig.cpp:



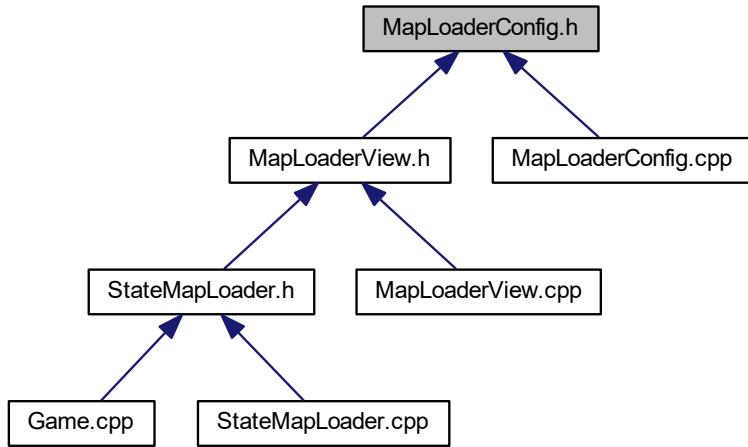
## 11.47 MapLoaderConfig.h File Reference

```
#include <TGUI/Color.hpp>
#include "Utility.h"
#include "IConfig.h"
```

Include dependency graph for MapLoaderConfig.h:



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



## Classes

- struct [Loader::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MapLoaderView](#).*

## Namespaces

- [Loader](#)

*Namespace designated for [MapLoader](#) view.*

## Enumerations

- enum [Loader::Btn](#) {
 [Loader::Btn::loadMap](#),
 [Loader::Btn::openEditor](#),
 [Loader::Btn::goBack](#),
 [Loader::Btn::SIZE](#) }

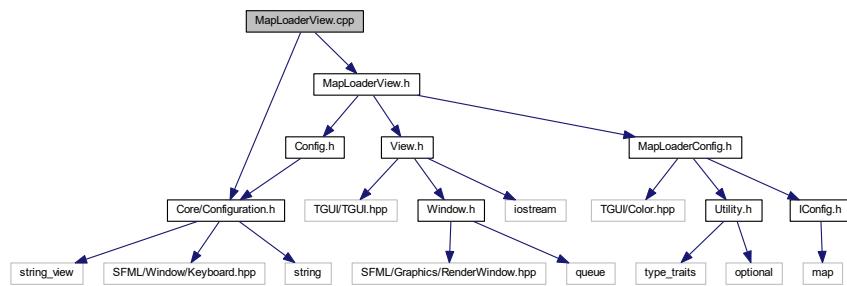
## Variables

- `constexpr std::initializer_list< Loader::Btn > Loader::Buttons`

*An additional initializer list to help mapping Btn names to Strings.*

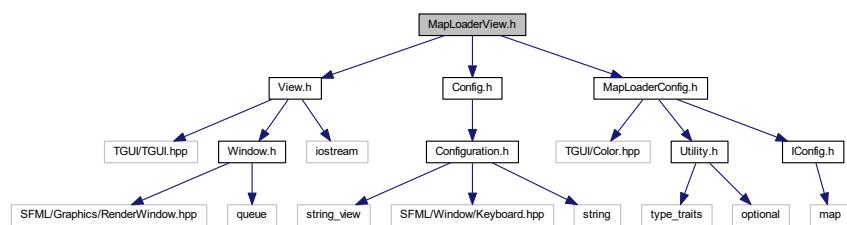
## 11.48 MapLoaderView.cpp File Reference

```
#include <Core/Configuration.h>
#include "MapLoaderView.h"
Include dependency graph for MapLoaderView.cpp:
```

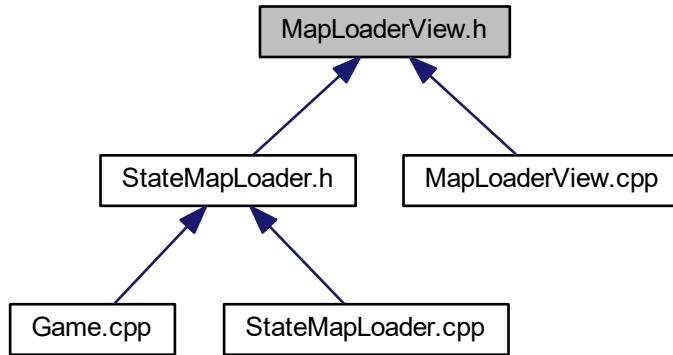


## 11.49 MapLoaderView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "MapLoaderConfig.h"
Include dependency graph for MapLoaderView.h:
```



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

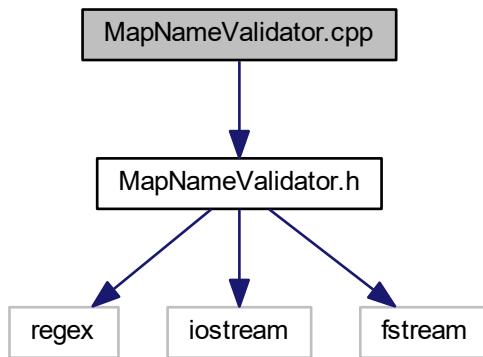


## Classes

- class [MapLoaderView](#)  
*View class used to draw gui within StateMapLoader.*

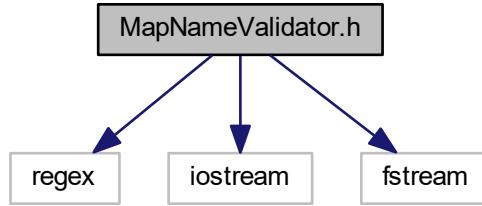
## 11.50 MapNameValidator.cpp File Reference

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

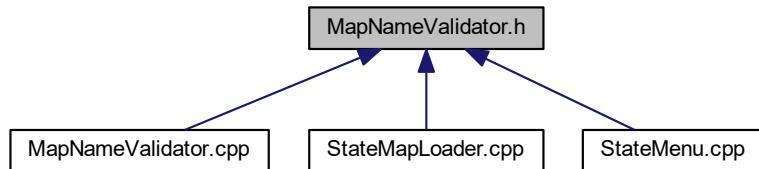


## 11.51 MapNameValidator.h File Reference

```
#include <regex>
#include <iostream>
#include <fstream>
Include dependency graph for MapNameValidator.h:
```



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



## Classes

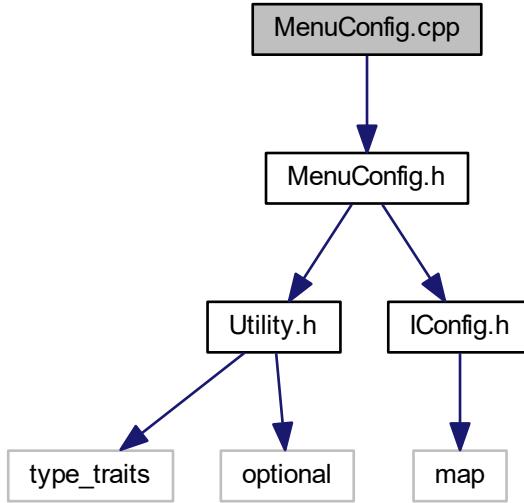
- class [MapNameValidator](#)

*A utility class validating if a given map file name is valid.*

## 11.52 MenuConfig.cpp File Reference

```
#include "MenuConfig.h"
```

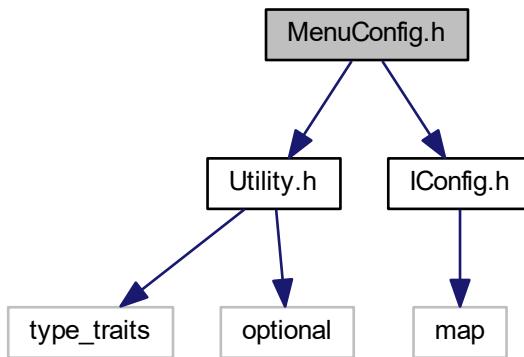
Include dependency graph for MenuConfig.cpp:



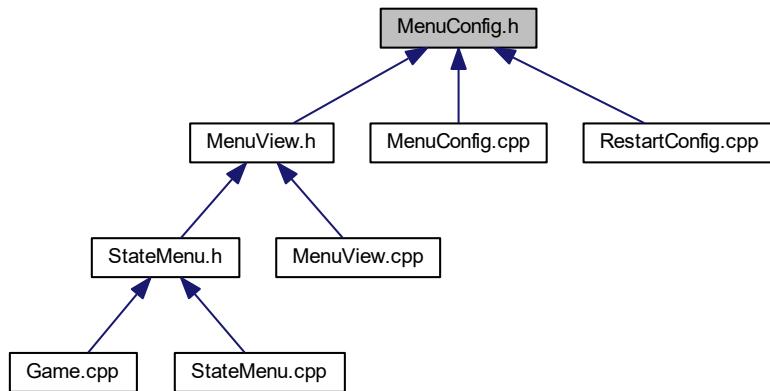
### 11.53 MenuConfig.h File Reference

```
#include "Utility.h"
#include "IConfig.h"
```

Include dependency graph for MenuConfig.h:



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



## Classes

- struct [Menu::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [MenuView](#).*

## Namespaces

- [Menu](#)

*Namespace designated for [Menu](#) view.*

## Enumerations

- enum [Menu::Btn](#) {
 [Menu::Btn::newGame](#),
 [Menu::Btn::loadGame](#),
 [Menu::Btn::options](#),
 [Menu::Btn::about](#),
 [Menu::Btn::exit](#),
 [Menu::Btn::SIZE](#) }

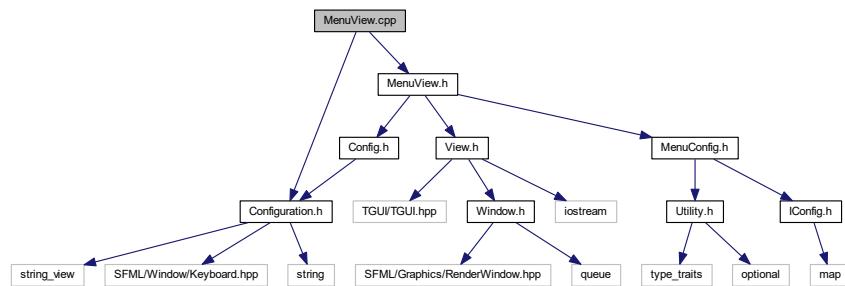
## Variables

- [constexpr std::initializer\\_list<Menu::Btn> Menu::Buttons](#)

*An additional initializer list to help mapping Btn names to Strings.*

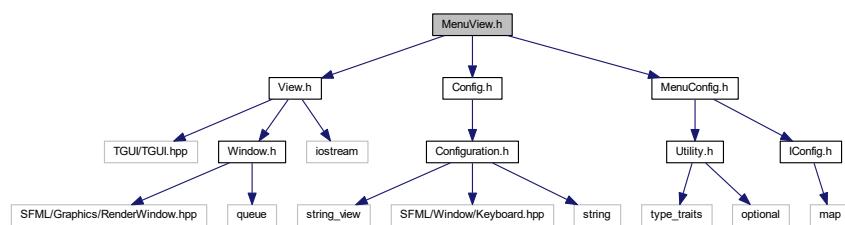
## 11.54 MenuView.cpp File Reference

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

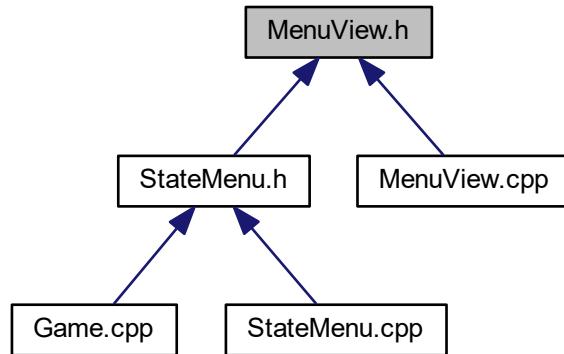


## 11.55 MenuView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "MenuConfig.h"
Include dependency graph for MenuView.h:
```



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



## Classes

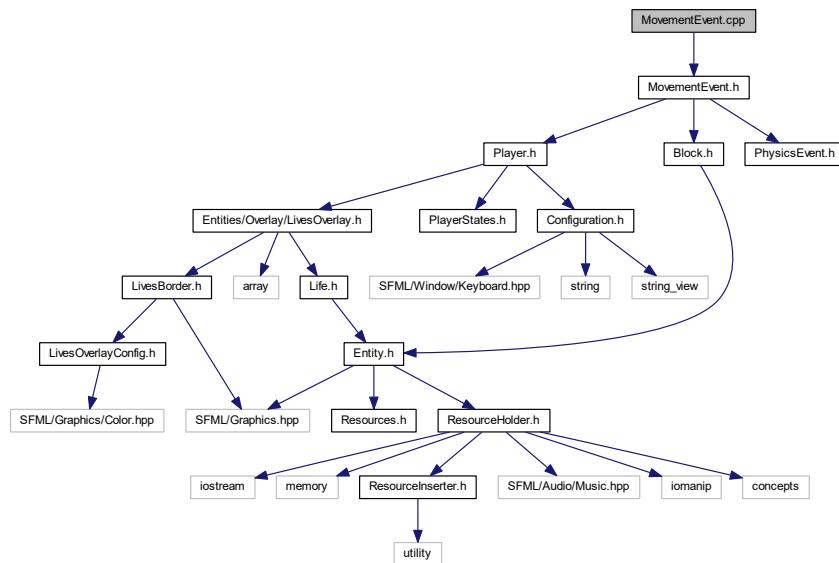
- class [MenuView](#)

*View* class used to draw gui within [StateMenu](#).

## 11.56 MovementEvent.cpp File Reference

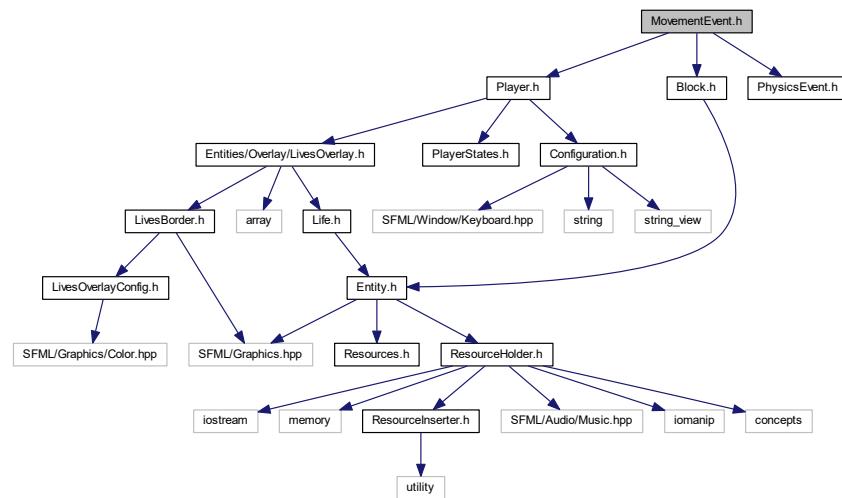
```
#include "MovementEvent.h"
```

Include dependency graph for MovementEvent.cpp:

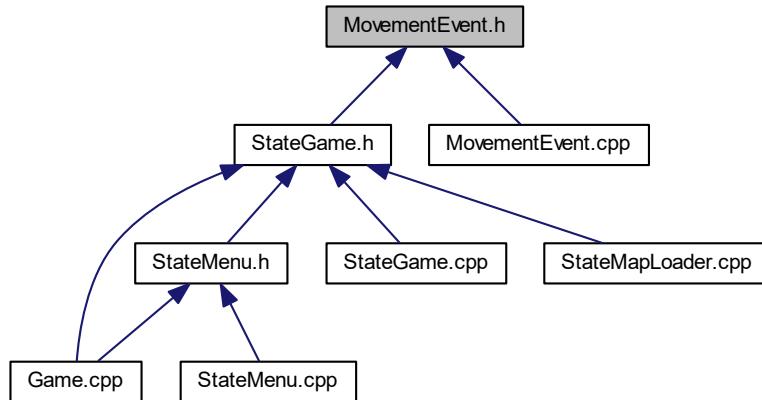


## 11.57 MovementEvent.h File Reference

```
#include "Player.h"
#include "Block.h"
#include "PhysicsEvent.h"
Include dependency graph for MovementEvent.h:
```



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



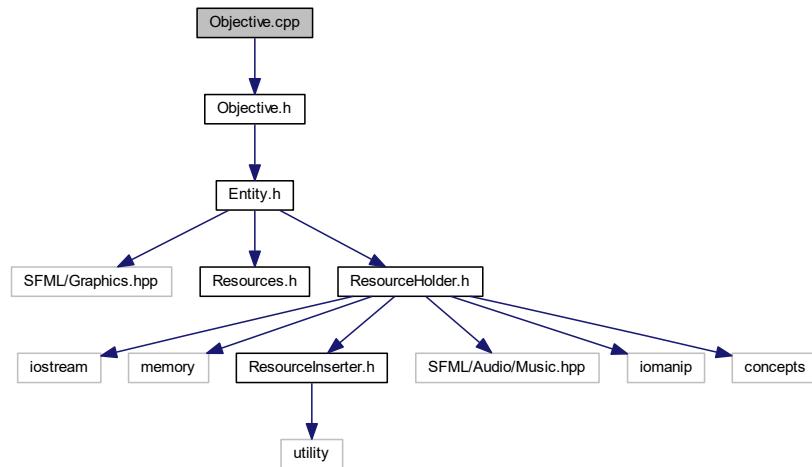
## Classes

- class [MovementEvent](#)

*MovementEvent* class responsible for updating the player movement in an **axis-independent way**.

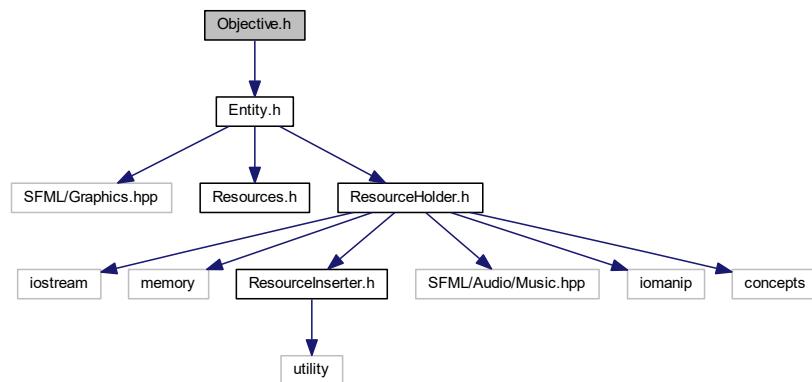
## 11.58 Objective.cpp File Reference

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

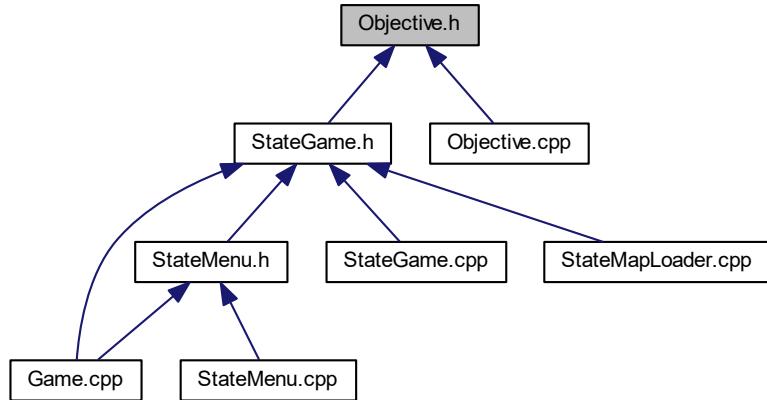


## 11.59 Objective.h File Reference

```
#include "Entity.h"
Include dependency graph for Objective.h:
```



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



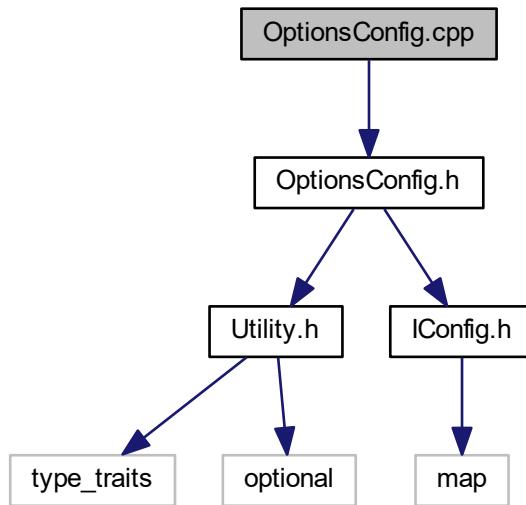
## Classes

- class [Objective](#)

*Used to represent the game Objective. Player intersecting Objective wins the game.*

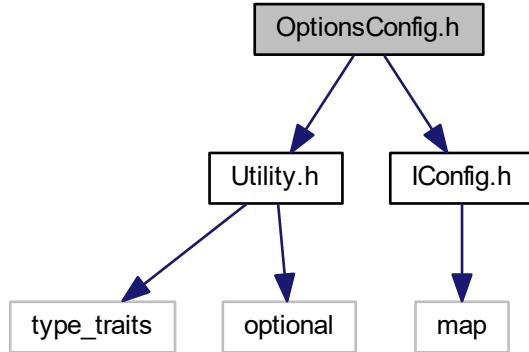
## 11.60 OptionsConfig.cpp File Reference

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

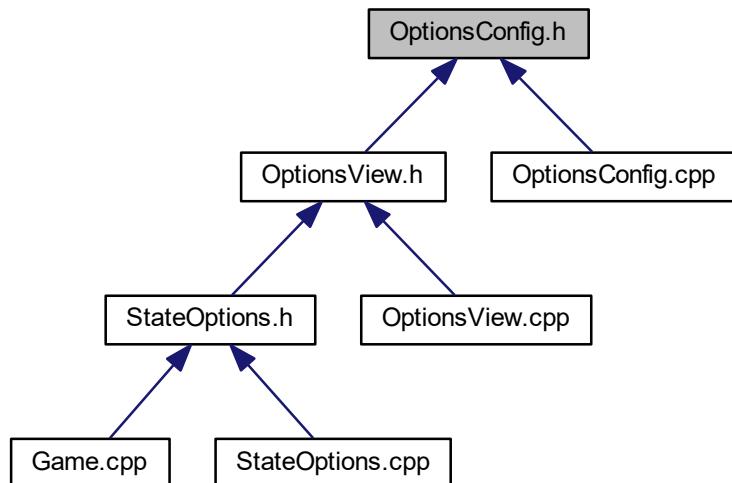


## 11.61 OptionsConfig.h File Reference

```
#include "Utility.h"
#include "IConfig.h"
Include dependency graph for OptionsConfig.h:
```



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



### Classes

- struct [Options::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [OptionsView](#).*

## Namespaces

- [Options](#)

*Namespace designated for [Options](#) view.*

## Enumerations

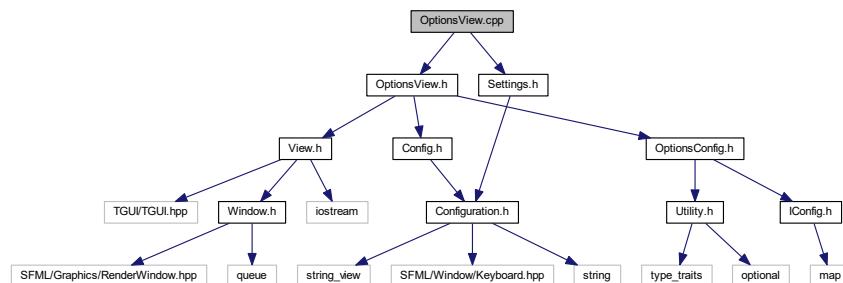
- enum [Options::Btn](#) {
 [Options::Btn::Keybinds](#),
 [Options::Btn::GoBack](#),
 [Options::Btn::SIZE](#) }

## Variables

- [constexpr std::initializer\\_list< Options::Btn > Options::Buttons](#)  
*An additional initializer list to help mapping Btn names to Strings.*

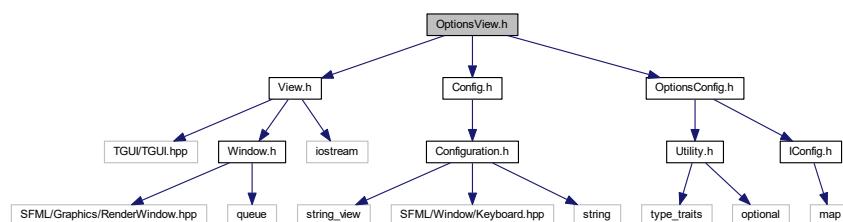
## 11.62 OptionsView.cpp File Reference

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

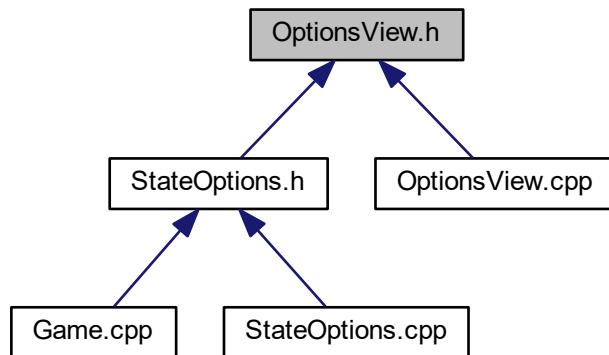


## 11.63 OptionsView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "OptionsConfig.h"
Include dependency graph for OptionsView.h:
```



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

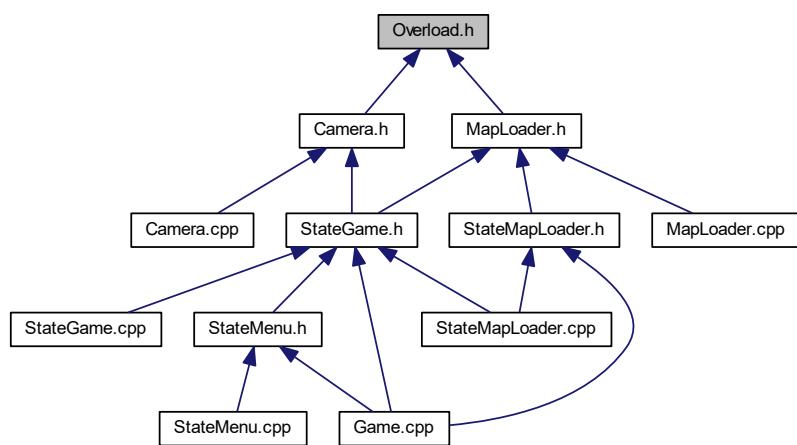


## Classes

- class `OptionsView`  
*View class used to draw gui within `StateOptions`.*

## 11.64 Overload.h File Reference

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

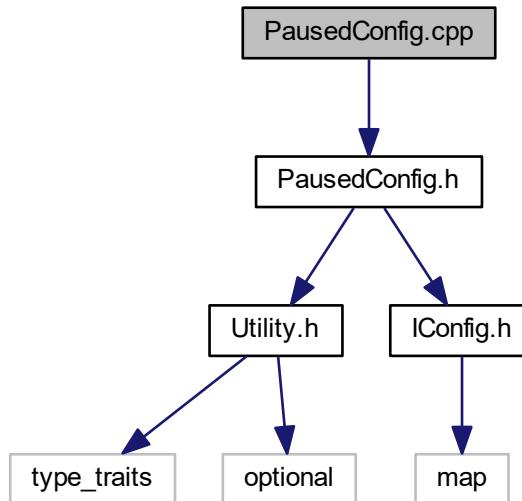


## Classes

- struct `overload< Ts >`

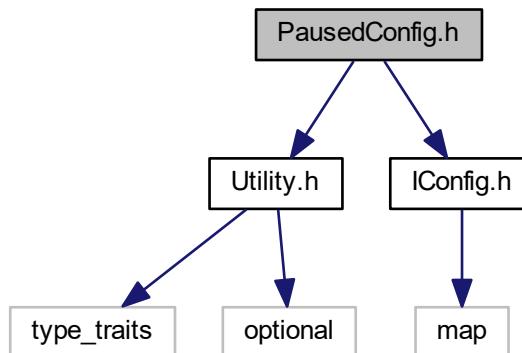
## 11.65 PausedConfig.cpp File Reference

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

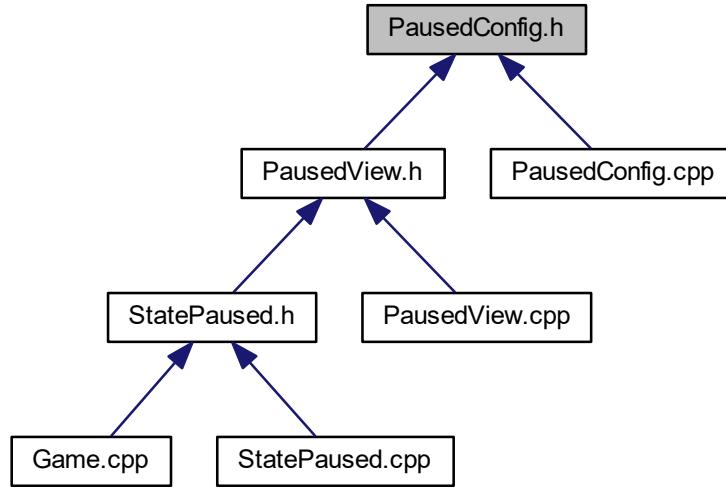


## 11.66 PausedConfig.h File Reference

```
#include "Utility.h"  
#include "IConfig.h"  
Include dependency graph for PausedConfig.h:
```



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



## Classes

- struct [Paused::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [PausedView](#).*

## Namespaces

- [Paused](#)

*Namespace designated for [Paused](#) view.*

## Enumerations

- enum [Paused::Btn](#) {
 [Paused::Btn::resume](#),
 [Paused::Btn::options](#),
 [Paused::Btn::menu](#),
 [Paused::Btn::SIZE](#) }

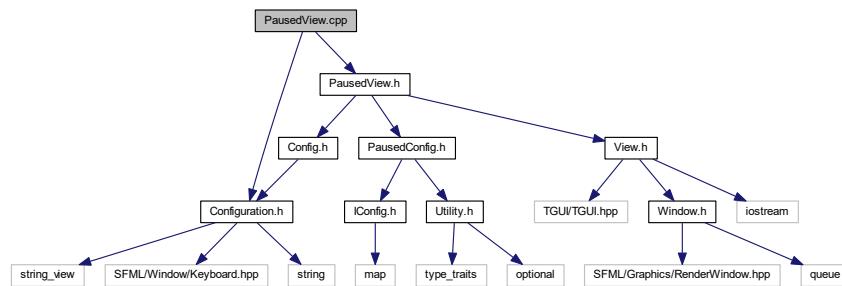
## Variables

- `constexpr std::initializer_list<Paused::Btn> Paused::Buttons`

*An additional initializer list to help mapping Btn names to Strings.*

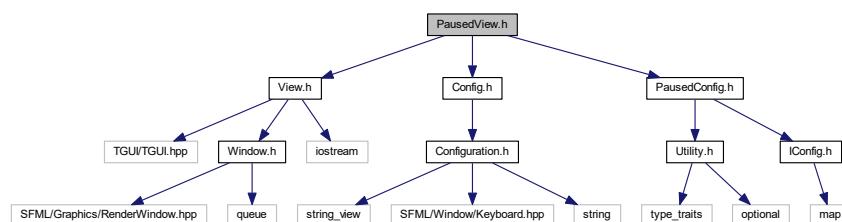
## 11.67 PausedView.cpp File Reference

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

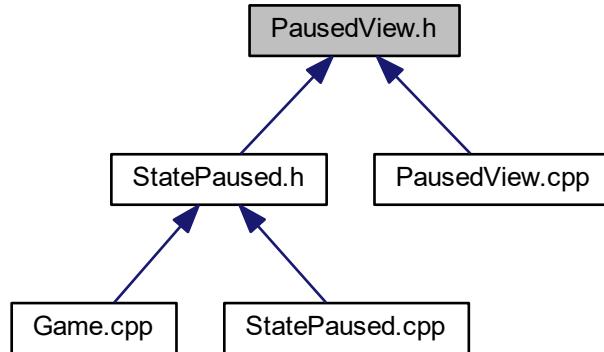


## 11.68 PausedView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "PausedConfig.h"
Include dependency graph for PausedView.h:
```



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

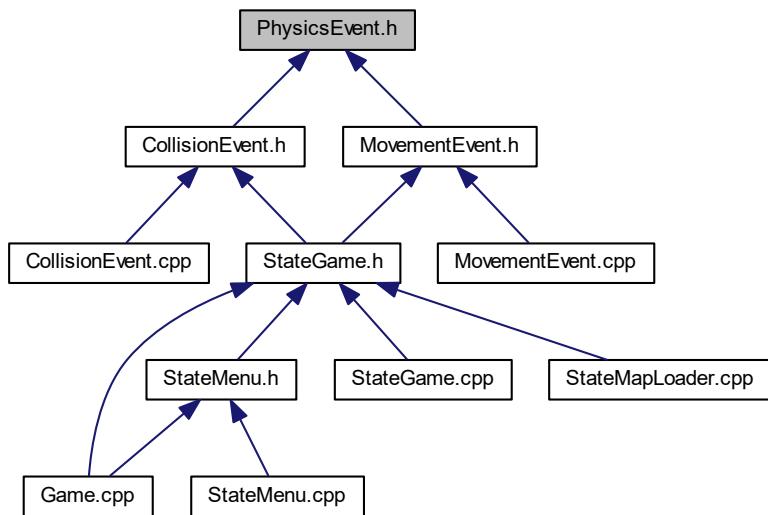


## Classes

- class [PausedView](#)  
*View class used to draw gui within `StatePaused`.*

## 11.69 PhysicsEvent.h File Reference

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



## Classes

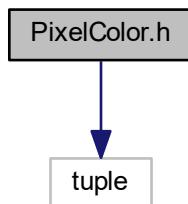
- class [PhysicsEvent](#)

*Specified Base [PhysicsEvent](#) class handling physics-related events on two separate axises.*

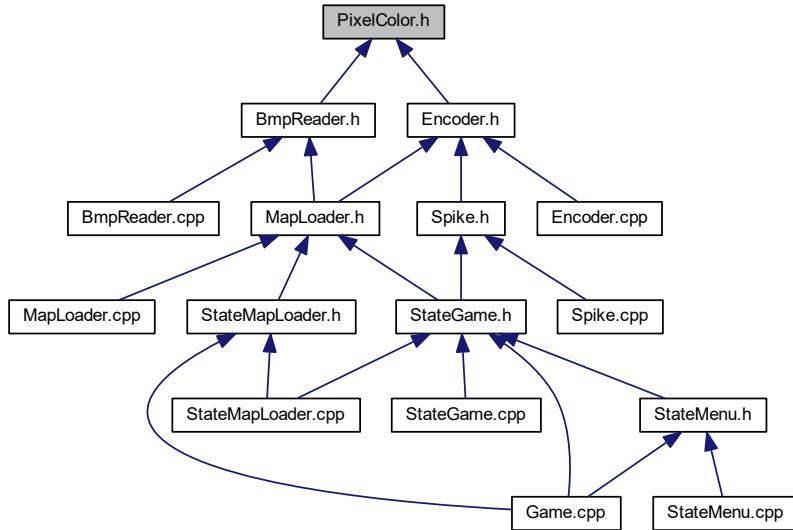
## 11.70 PixelColor.h File Reference

```
#include <tuple>
```

Include dependency graph for PixelColor.h:



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



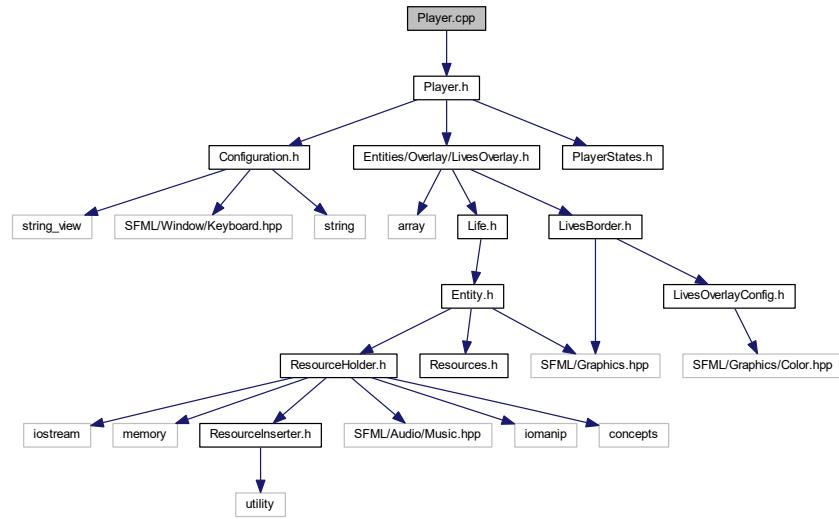
## Classes

- struct [PixelColor](#)

*Container for one .bmp BGR pixel.*

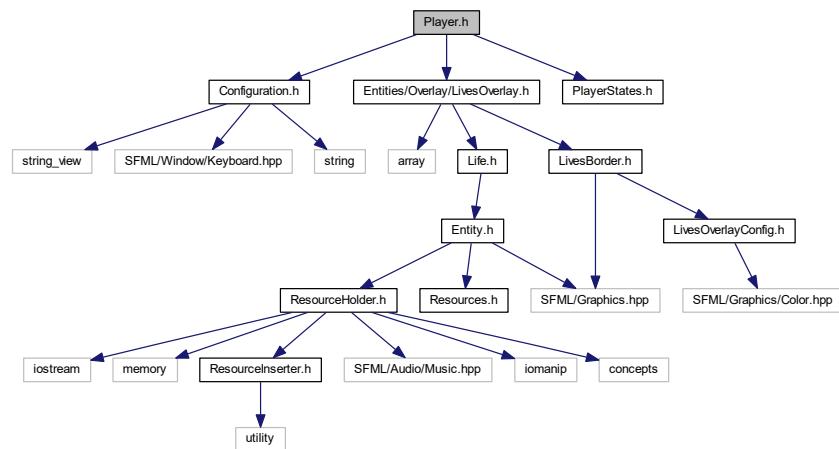
## 11.71 Player.cpp File Reference

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

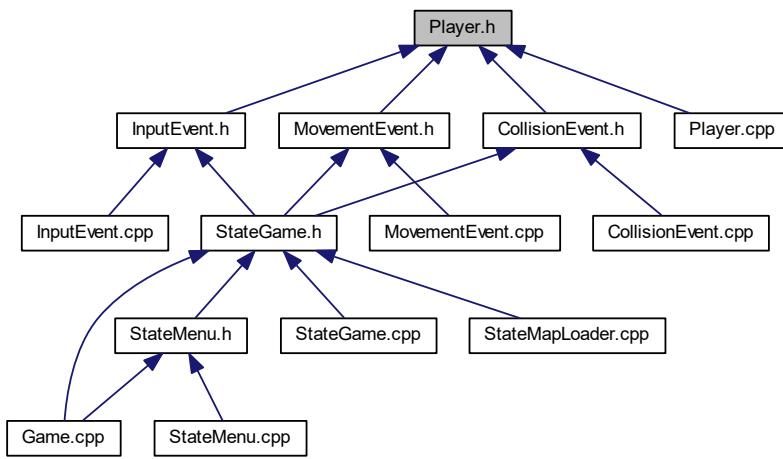


## 11.72 Player.h File Reference

```
#include <Configuration.h>
#include <Entities/Overlay/LivesOverlay.h>
#include "PlayerStates.h"
Include dependency graph for Player.h:
```



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



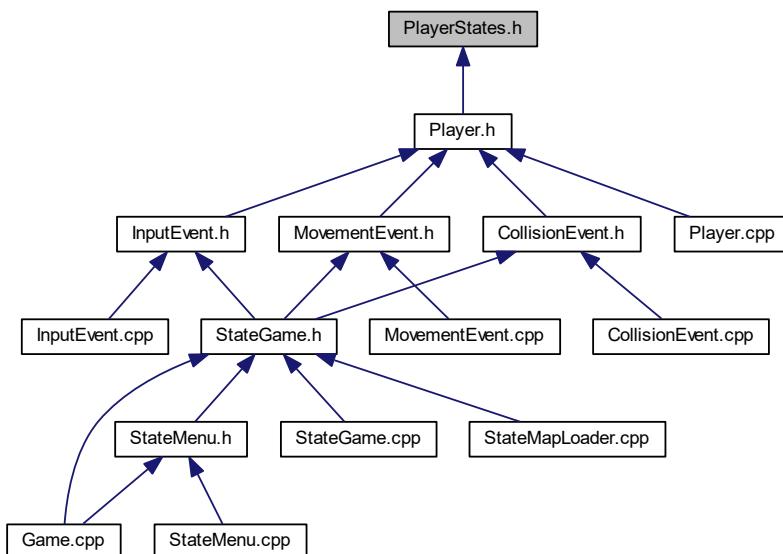
## Classes

- class [Player](#)

*Controls the player behaviour.*

## 11.73 PlayerStates.h File Reference

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



## Enumerations

- enum MovingState {  
    MovingState::movingLeft,  
    MovingState::movingRight,  
    MovingState::standing }

*Represents the horizontal MovingState of the player. Player is in one state at a time.*

  - enum JumpingState {  
    JumpingState::onGround,  
    JumpingState::jumping,  
    JumpingState::gravity }

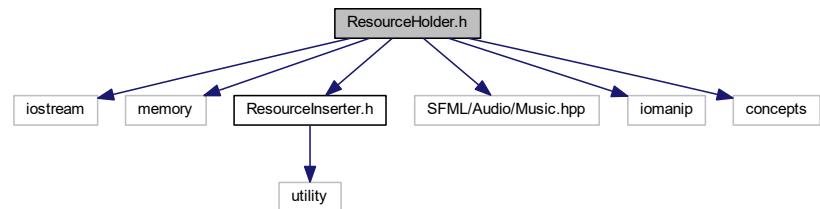
*Represents the JumpingState of the player. Player is in one state at a time.*

## 11.74 README.md File Reference

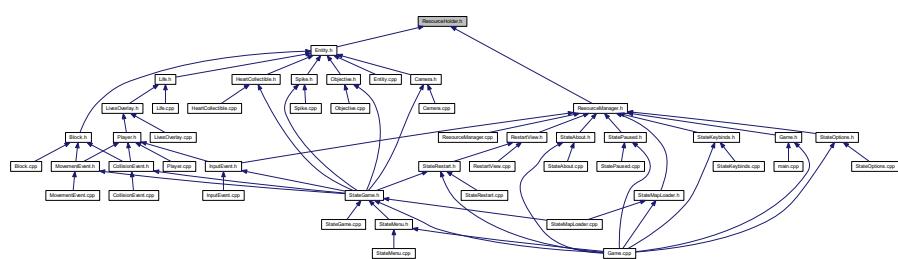
## 11.75 ResourceHolder.cpp File Reference

## 11.76 ResourceHolder.h File Reference

```
#include <iostream>
#include <memory>
#include "ResourceInserter.h"
#include <SFML/Audio/Music.hpp>
#include <iomanip>
#include <concepts>
```



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

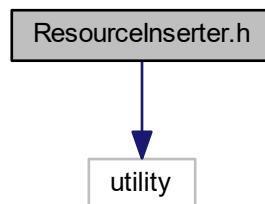


## Classes

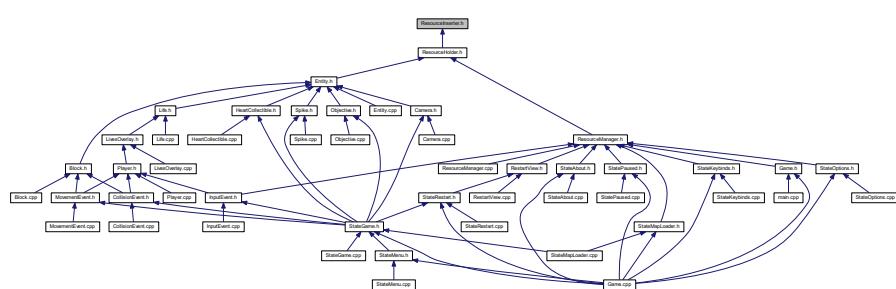
- class [MissingResource< Resource >](#)  
*Meaningful response error when resource of given type was missing.*
- class [ResourceHolder< Key, Resource >](#)  
*A generic [ResourceHolder](#) storing resources of type [Resource](#), accessed by [Key](#).*

## 11.77 Resourcelnserter.h File Reference

```
#include <utility>
Include dependency graph for Resourcelnserter.h:
```



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



## Classes

- class [Resourcelnserter< Key, Args >](#)  
*A proxy class used to handle 2+ arguments in operator+=, which takes only 1 argument.*

## Functions

- template<typename T, typename... Args>  
`Resourcelnserter(T, std::string_view, Args... args) -> Resourcelnserter< T, Args... >`  
*Deduction guide.*

### **11.77.1 Function Documentation**

### 11.77.1.1 ResourceInserter()

```
template<typename T , typename... Args>
ResourceInserter (
    T ,
    std::string_view ,
    Args... args ) -> ResourceInserter< T, Args... >
```

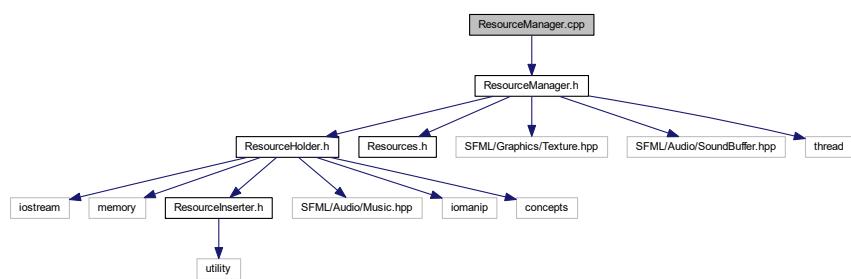
## Deduction guide.

Here is the caller graph for this function:



## 11.78 ResourceManager.cpp File Reference

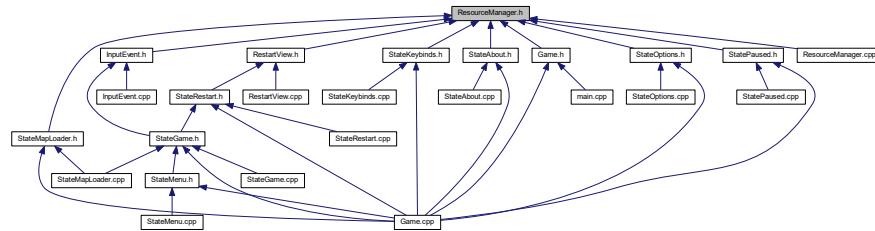
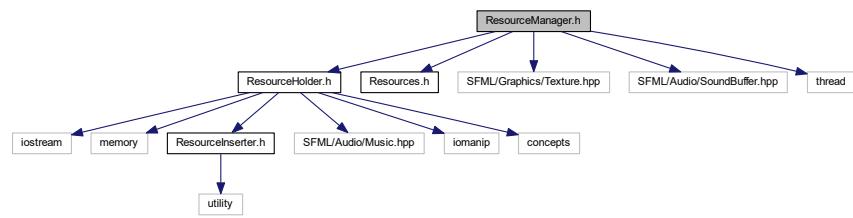
```
#include "ResourceManager.h"
Include dependency graph for ResourceManager.cpp:
```



## 11.79 ResourceManager.h File Reference

```
#include "ResourceHolder.h"
#include "Resources.h"
#include <SFML/Graphics/Texture.hpp>
#include <SFML/Audio/SoundBuffer.hpp>
```

```
#include <thread>
Include dependency graph for ResourceManager.h:
```



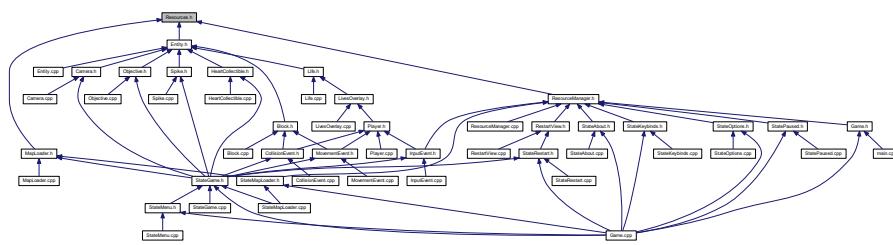
## Classes

- class [ResourceManager](#)

*ResourceManager* is a class responsible for managing all kinds of ResourceHolders.

## 11.80 Resources.h File Reference

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



## Namespaces

- [res](#)

Namespace wrapper over Resources enum IDs.

## Enumerations

- enum `res::Texture {`  
`res::Texture::BlockRed,`  
`res::Texture::BlockBlue,`  
`res::Texture::BlockBrown,`  
`res::Texture::BlockGray,`  
`res::Texture::BlockGreen,`  
`res::Texture::BlockPurple,`  
`res::Texture::BlockYellow,`  
`res::Texture::BlockCrate,`  
`res::Texture::Water1,`  
`res::Texture::Water2,`  
`res::Texture::Sign1,`  
`res::Texture::Sign2,`  
`res::Texture::Player,`  
`res::Texture::PlayerLeft,`  
`res::Texture::PlayerRight,`  
`res::Texture::Objective,`  
`res::Texture::Heart,`  
`res::Texture::HeartEmpty,`  
`res::Texture::BgGame,`  
`res::Texture::BgAbout,`  
`res::Texture::Spike,`  
`res::Texture::OptionsLeftTopCorner,`  
`res::Texture::OptionsLeftBotCorner,`  
`res::Texture::OptionsRightBotCorner,`  
`res::Texture::OptionsRightTopCorner,`  
`res::Texture::GameLost,`  
`res::Texture::GameWon }`

*Available Texture IDs.*

- enum `res::Sound {`  
`res::Sound::Collect,`  
`res::Sound::BtnHover,`  
`res::Sound::Death,`  
`res::Sound::WinGame }`

*Available Sound IDs.*

- enum `res::Music { res::Music::Background }`

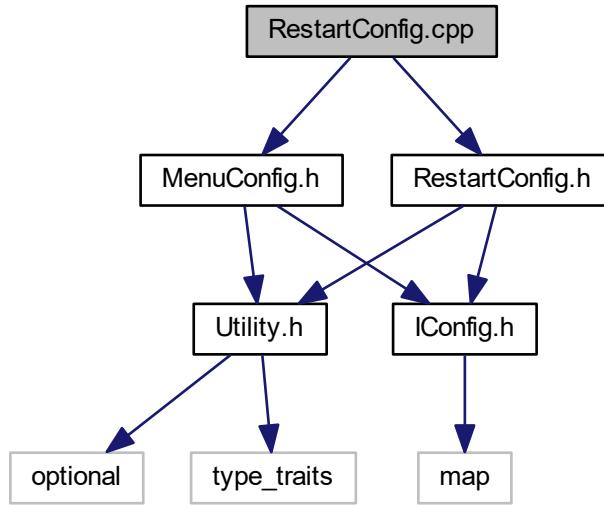
*Available Music IDs.*

## 11.81 RestartConfig.cpp File Reference

---

```
#include "MenuConfig.h"
#include "RestartConfig.h"
```

Include dependency graph for RestartConfig.cpp:

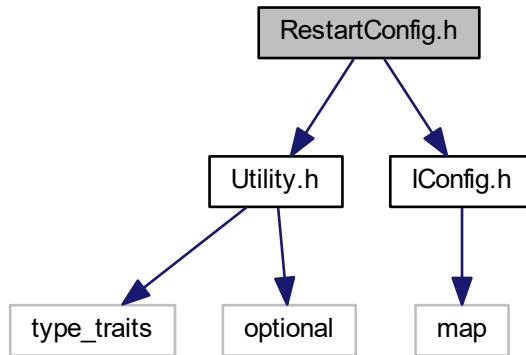


## 11.82 RestartConfig.h File Reference

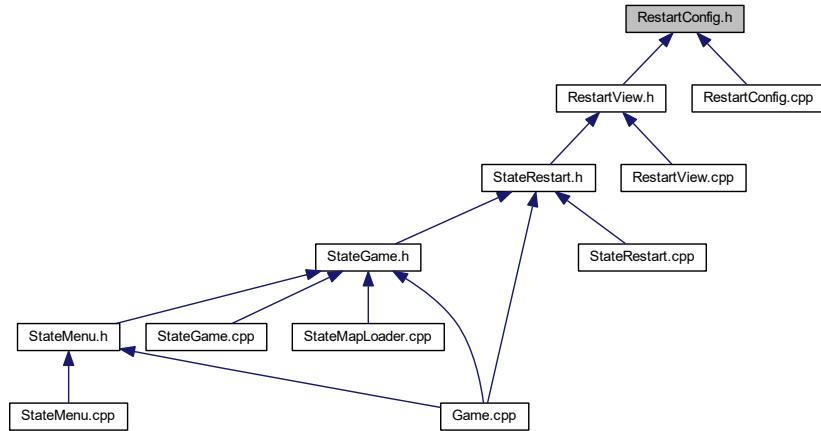
```
#include "Utility.h"
```

```
#include "IConfig.h"
```

Include dependency graph for `RestartConfig.h`:



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



## Classes

- struct [Restart::Config](#)

*Deriving from [IConfig<Btn>](#), holds meaningful view configurations specific to [RestartView](#).*

## Namespaces

- [Restart](#)

*Namespace designated for [Restart](#) view.*

## Enumerations

- enum [Restart::Btn](#) {
 [Restart::Btn::PlayAgain](#),
 [Restart::Btn::Menu](#),
 [Restart::Btn::SIZE](#) }

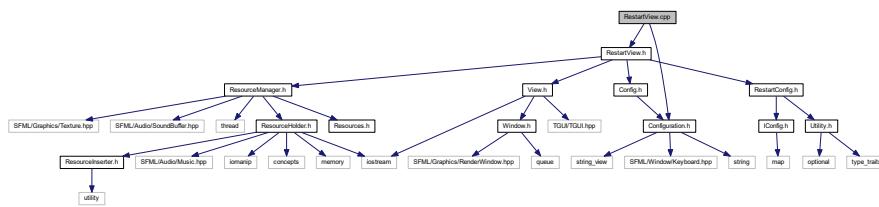
## Variables

- constexpr std::initializer\_list<[Restart::Btn](#)> [Restart::Buttons](#)

*An additional initializer list to help mapping Btn names to Strings.*

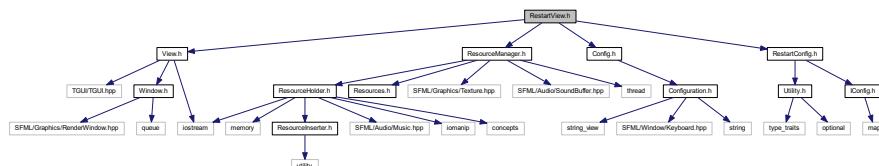
## 11.83 RestartView.cpp File Reference

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

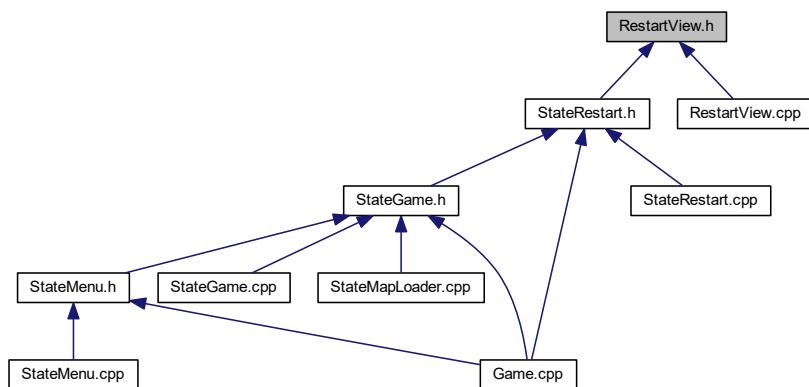


## 11.84 RestartView.h File Reference

```
#include "View.h"
#include "Config.h"
#include "RestartConfig.h"
#include "ResourceManager.h"
Include dependency graph for RestartView.h:
```



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



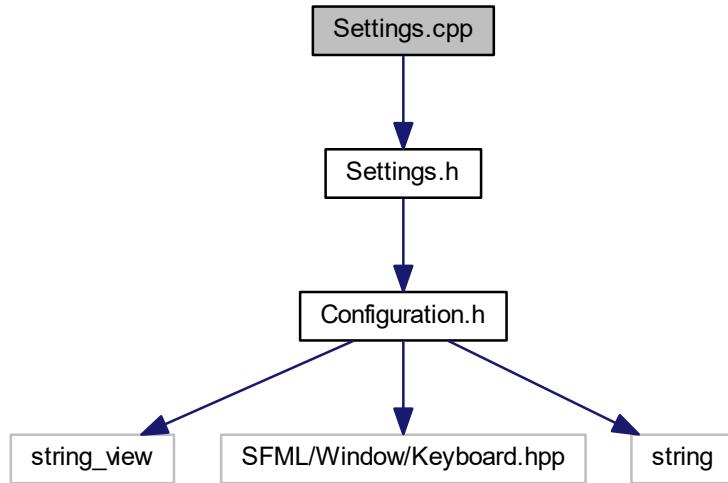
## Classes

- class [RestartView](#)

*View* class used to draw gui within [StateRestart](#).

## 11.85 Settings.cpp File Reference

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



## Variables

- [AudioCfg audioConfig {}](#)

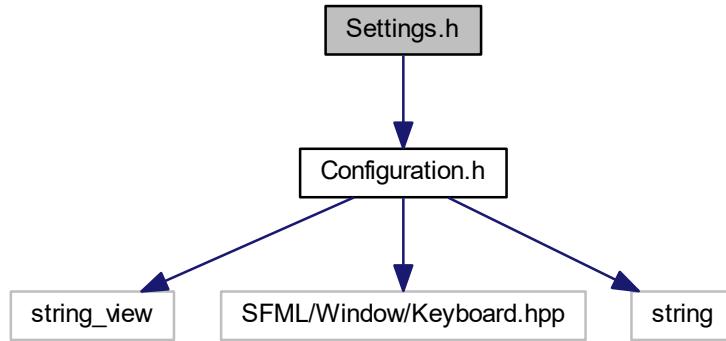
### 11.85.1 Variable Documentation

#### 11.85.1.1 [audioConfig](#)

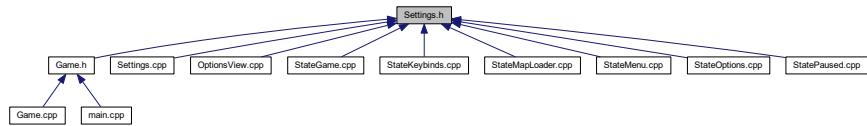
```
AudioCfg audioConfig {}
```

## 11.86 Settings.h File Reference

```
#include "Configuration.h"
Include dependency graph for Settings.h:
```



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



## Classes

- class [AudioCfg](#)  
*Responsible for storing audio config enable.*

## Variables

- [AudioCfg audioConfig](#)

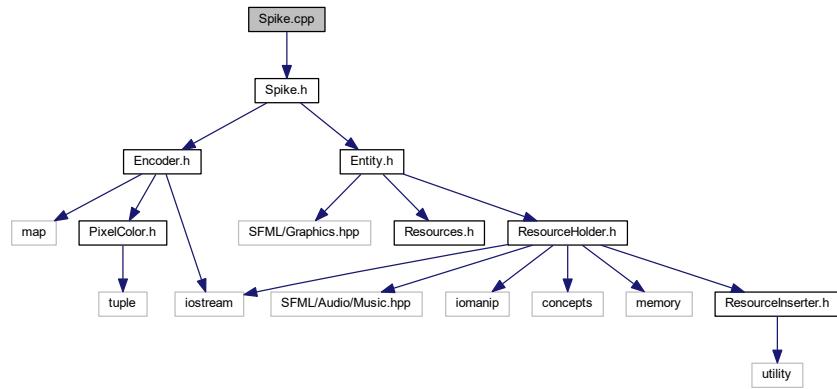
### 11.86.1 Variable Documentation

#### 11.86.1.1 audioConfig

[AudioCfg](#) `audioConfig`

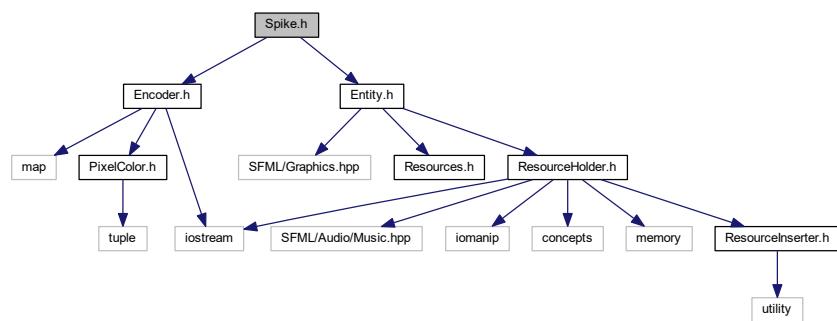
## 11.87 Spike.cpp File Reference

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

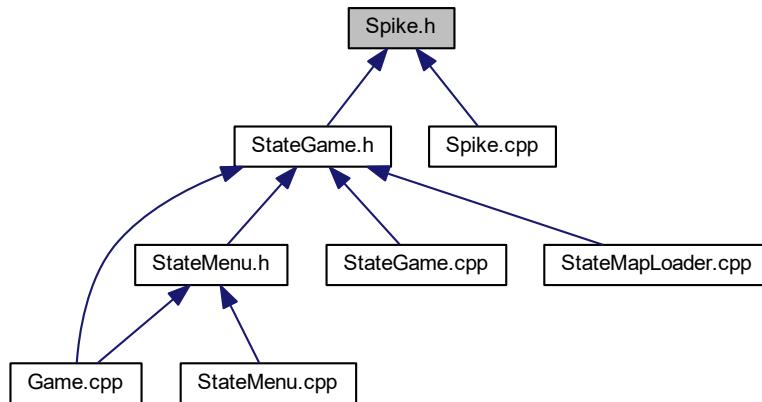


## 11.88 Spike.h File Reference

```
#include "Encoder.h"
#include "Entity.h"
Include dependency graph for Spike.h:
```



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

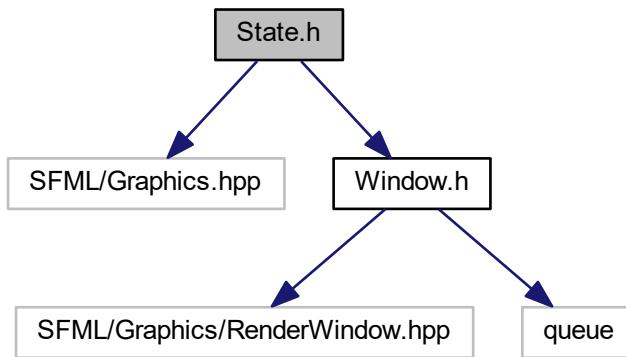


## Classes

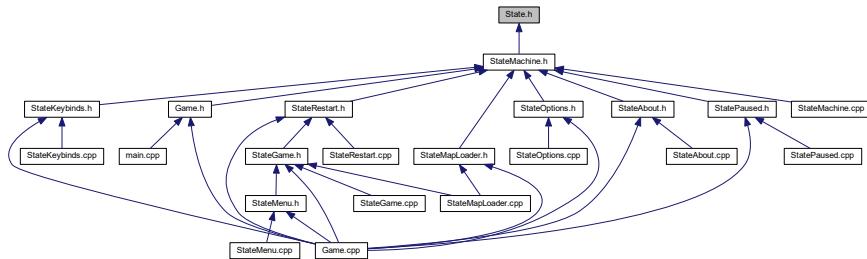
- class [Spike](#)  
*Used to represent the spikes. Intersecting spike kills the [Player](#).*

## 11.89 State.h File Reference

```
#include <SFML/Graphics.hpp>
#include "Window.h"
Include dependency graph for State.h:
```



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



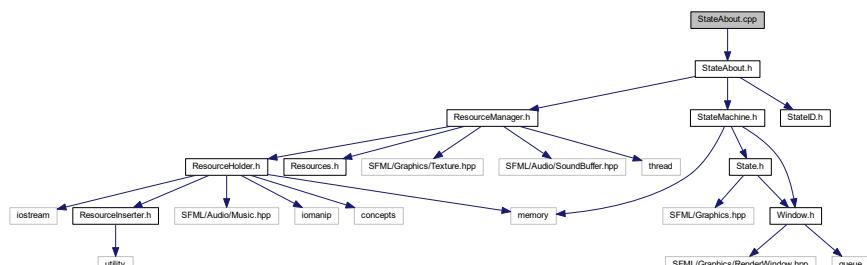
## Classes

- class State
    - < Pass-through

## 11.90 StateAbout.cpp File Reference

```
#include "StateAbout.h"
```

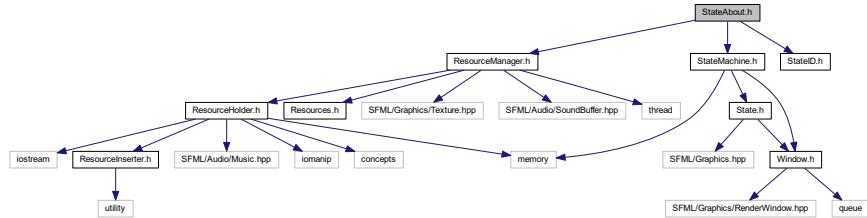
Include dependency graph for StateAbout.cpp:



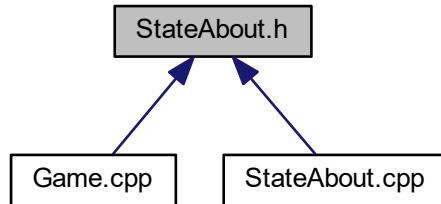
## 11.91 StateAbout.h File Reference

```
#include "ResourceManager.h"  
#include "StateMachine.h"  
#include "StateID.h"
```

Include dependency graph for StateAbout.h:



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



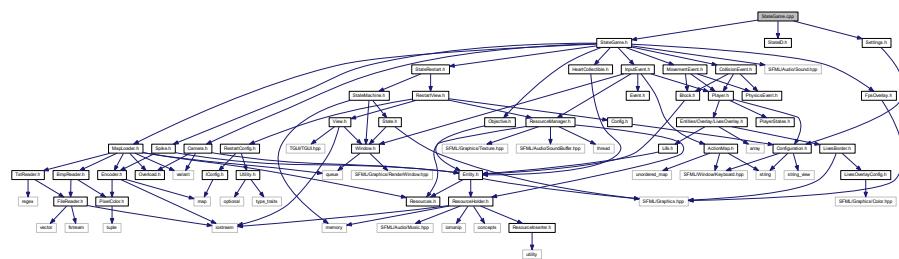
## Classes

- class [StateAbout](#)

*Encapsulates the logic of about menu within one [StateAbout](#) class. Displays the about texture.*

## 11.92 StateGame.cpp File Reference

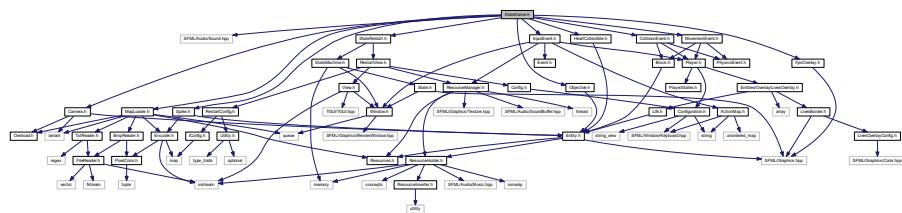
```
#include "StateGame.h"
#include "StateID.h"
#include "Settings.h"
Include dependency graph for StateGame.cpp:
```



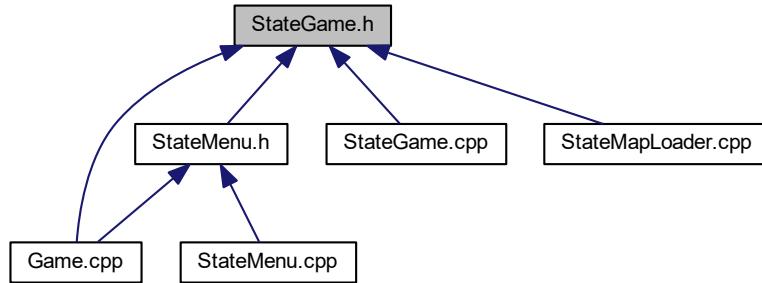
## 11.93 StateGame.h File Reference

```
#include <SFML/Audio/Sound.hpp>
#include "MapLoader.h"
#include "StateRestart.h"
#include "Camera.h"
#include "CollisionEvent.h"
#include "MovementEvent.h"
#include "InputEvent.h"
#include "FpsOverlay.h"
#include "Objective.h"
```

```
#include "Spike.h"
#include "HeartCollectible.h"
Include dependency graph for StateGame.h:
```



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

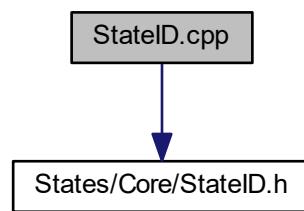


## Classes

- class [StateGame](#)  
*Encapsulates the main game logic within one [StateGame](#) class.*

## 11.94 StateID.cpp File Reference

```
#include "States/Core/StateID.h"
Include dependency graph for StateID.cpp:
```



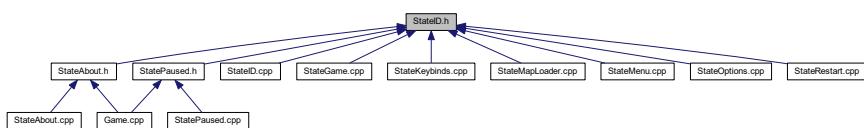
## Namespaces

- [state](#)

*The IDs of the stored states, so the user can access an arbitrary state.*

## 11.95 StateID.h File Reference

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



## Namespaces

- [state](#)

*The IDs of the stored states, so the user can access an arbitrary state.*

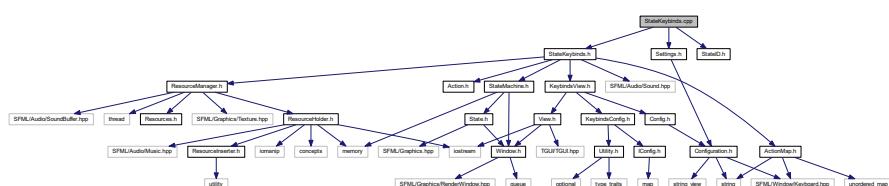
## Variables

- int [state::gameID](#)
- int [state::menuID](#)
- int [state::loaderID](#)
- int [state::optionsID](#)
- int [state::keybindsID](#)
- int [state::pausedID](#)
- int [state::restartID](#)
- int [state::aboutID](#)

## 11.96 StateKeybinds.cpp File Reference

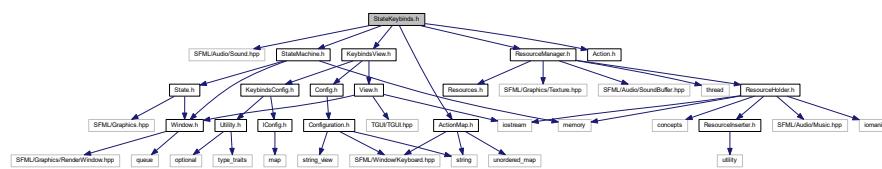
```
#include "StateKeybinds.h"
#include "StateID.h"
#include "Settings.h"

Include dependency graph for StateKeybinds.cpp:
```

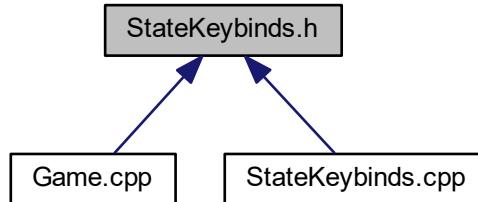


## 11.97 StateKeybinds.h File Reference

```
#include <SFML/Audio/Sound.hpp>
#include "ResourceManager.h"
#include "KeybindsView.h"
#include "Action.h"
#include "ActionMap.h"
#include "StateMachine.h"
Include dependency graph for StateKeybinds.h:
```



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



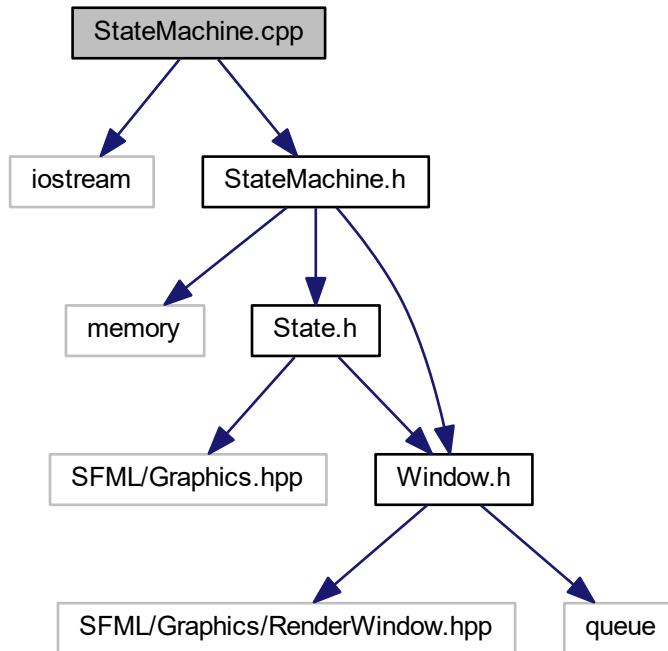
## Classes

- class [StateKeybinds](#)  
*Encapsulates the logic of rebinding menu within one [StateKeybinds](#) class.*

## 11.98 StateMachine.cpp File Reference

```
#include <iostream>
#include "StateMachine.h"
```

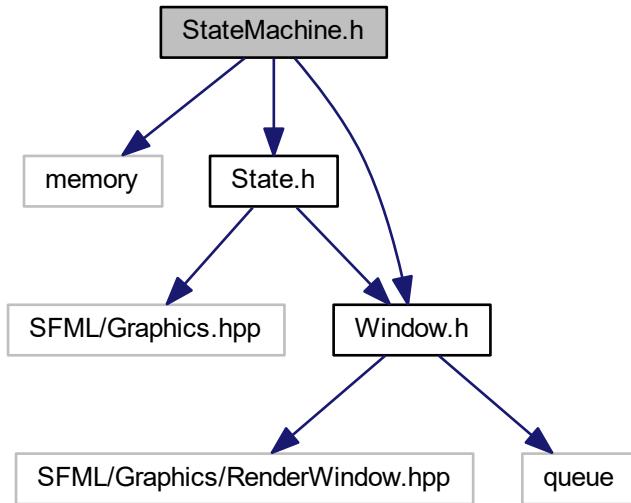
Include dependency graph for StateMachine.cpp:



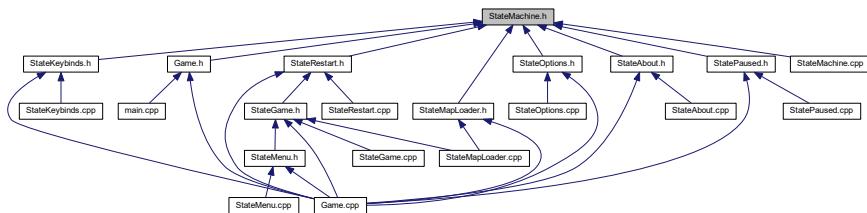
## 11.99 StateMachine.h File Reference

```
#include <memory>
#include "State.h"
#include "Window.h"
```

Include dependency graph for StateMachine.h:



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



## Classes

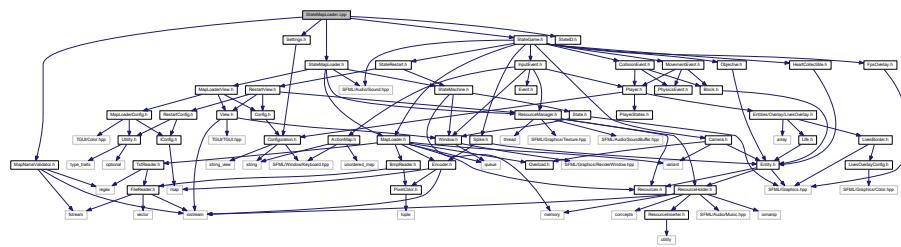
- class [StateMachine](#)

*Container for all game States and holds current state pointer. Updates [State](#) logic every frame.*

## 11.100 StateMapLoader.cpp File Reference

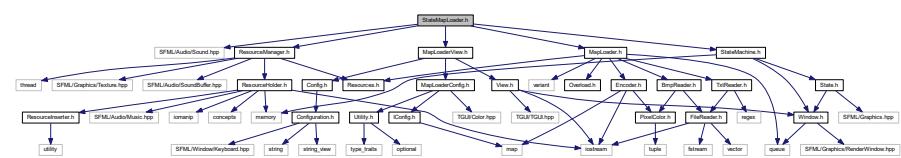
```
#include "StateMapLoader.h"
#include "StateID.h"
#include "StateGame.h"
#include "MapNameValidator.h"
```

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

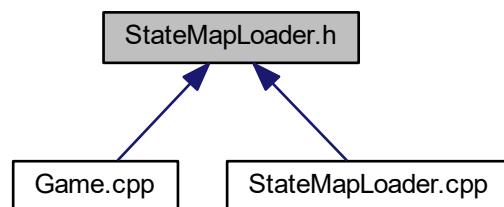


## 11.101 StateMapLoader.h File Reference

```
#include <SFML/Audio/Sound.hpp>
#include "ResourceManager.h"
#include "MapLoaderView.h"
#include "MapLoader.h"
#include "StateMachine.h"
Include dependency graph for StateMapLoader.h:
```



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



## Classes

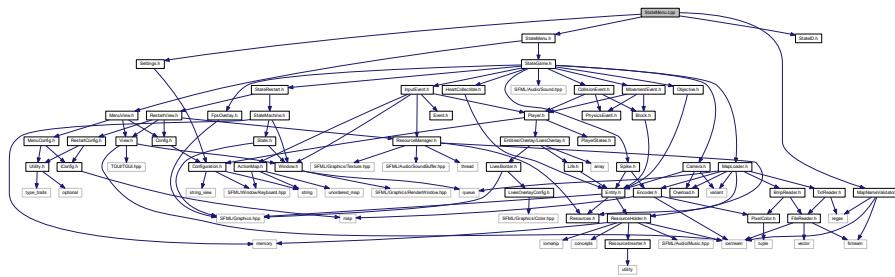
- class [StateMapLoader](#)

*Encapsulates the logic of map loader menu within one [StateMapLoader](#) class.*

## 11.102 StateMenu.cpp File Reference

```
#include "Settings.h"
#include "MapNameValidator.h"
#include "StateID.h"
#include "StateMenu.h"

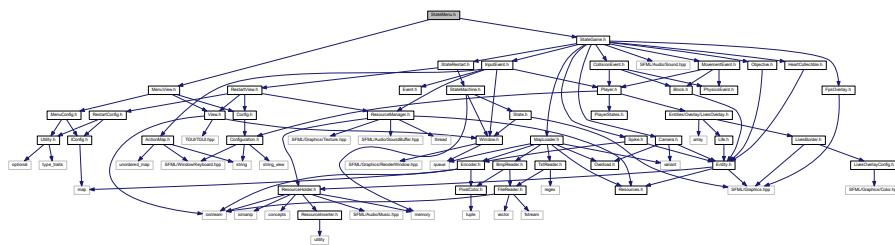
Include dependency graph for StateMenu.cpp:
```



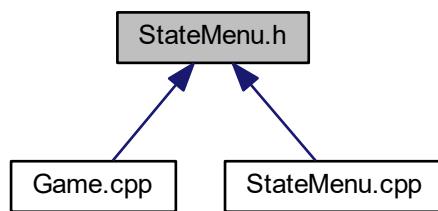
## 11.103 StateMenu.h File Reference

```
#include "StateGame.h"
#include "MenuView.h"

Include dependency graph for StateMenu.h:
```



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



## Classes

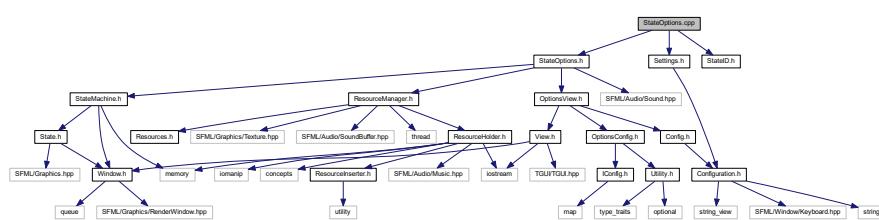
- class [StateMenu](#)

*Encapsulates the logic of main menu within one [StateRestart](#) class.*

## 11.104 StateOptions.cpp File Reference

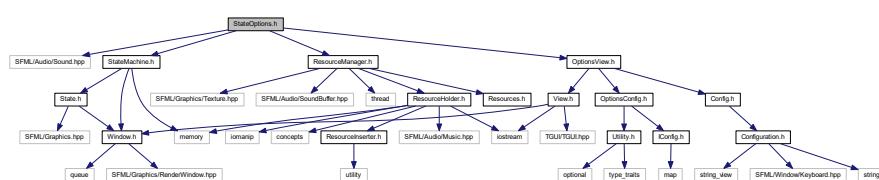
```
#include "StateOptions.h"
#include "StateID.h"
#include "Settings.h"
```

Include dependency graph for StateOptions.cpp:

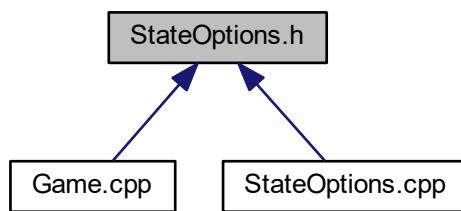


## 11.105 StateOptions.h File Reference

```
#include <SFML/Audio/Sound.hpp>
#include "OptionsView.h"
#include "ResourceManager.h"
#include "StateMachine.h"
Include dependency graph for StateOptions.h:
```



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



## Classes

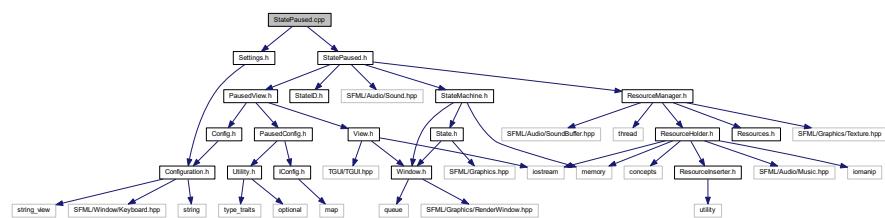
- class [StateOptions](#)

*Encapsulates the logic of options menu within one [StateOptions](#) class.*

## 11.106 StatePaused.cpp File Reference

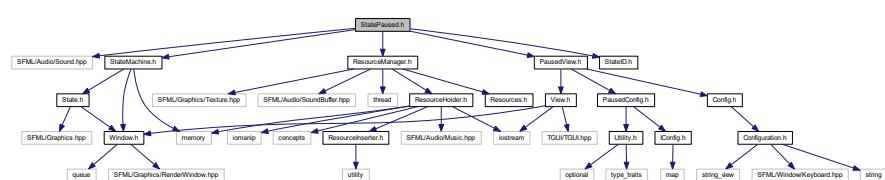
```
#include "Settings.h"
#include "StatePaused.h"
```

Include dependency graph for StatePaused.cpp:

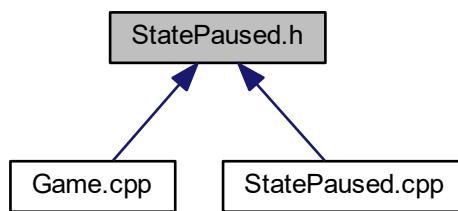


## 11.107 StatePaused.h File Reference

```
#include <SFML/Audio/Sound.hpp>
#include "PausedView.h"
#include "ResourceManager.h"
#include "StateMachine.h"
#include "StateID.h"
Include dependency graph for StatePaused.h:
```



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



## Classes

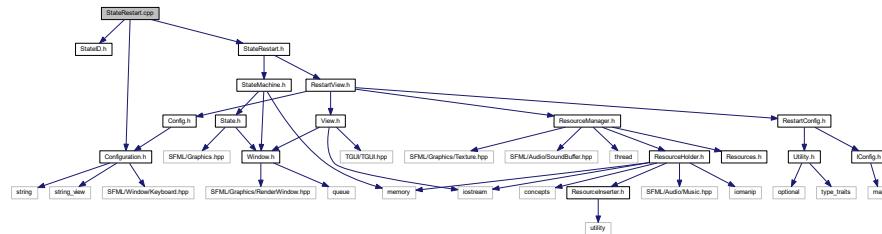
- class [StatePaused](#)

*Encapsulates the logic of paused menu within one [StatePaused](#) class.*

## 11.108 StateRestart.cpp File Reference

```
#include "StateID.h"
#include "Configuration.h"
#include "StateRestart.h"
```

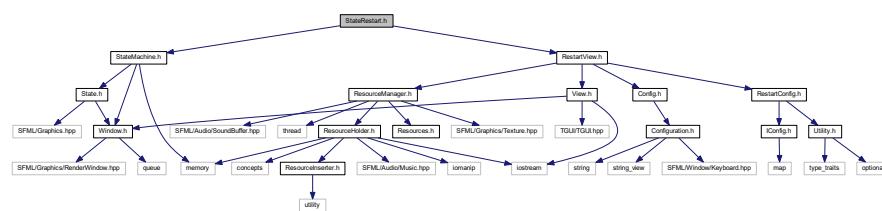
Include dependency graph for StateRestart.cpp:



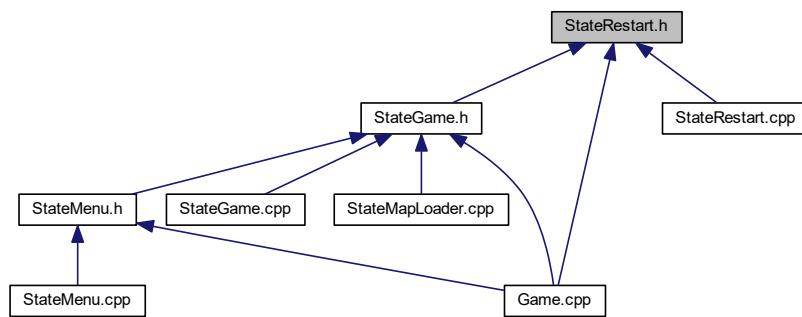
## 11.109 StateRestart.h File Reference

```
#include "StateMachine.h"
#include "RestartView.h"
```

Include dependency graph for StateRestart.h:



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



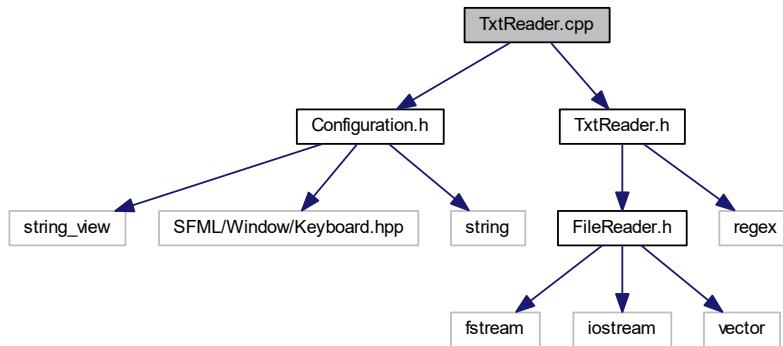
## Classes

- class [StateRestart](#)

*Encapsulates the logic of restart menu within one [StateRestart](#) class.*

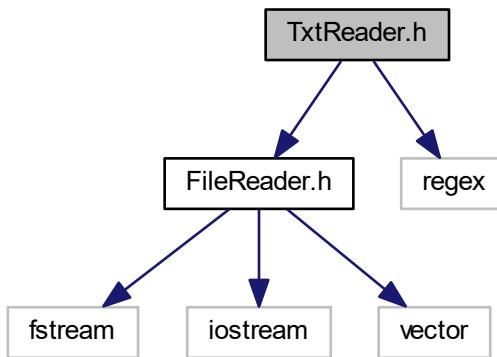
## 11.110 TxtReader.cpp File Reference

```
#include <Configuration.h>
#include "TxtReader.h"
Include dependency graph for TxtReader.cpp:
```

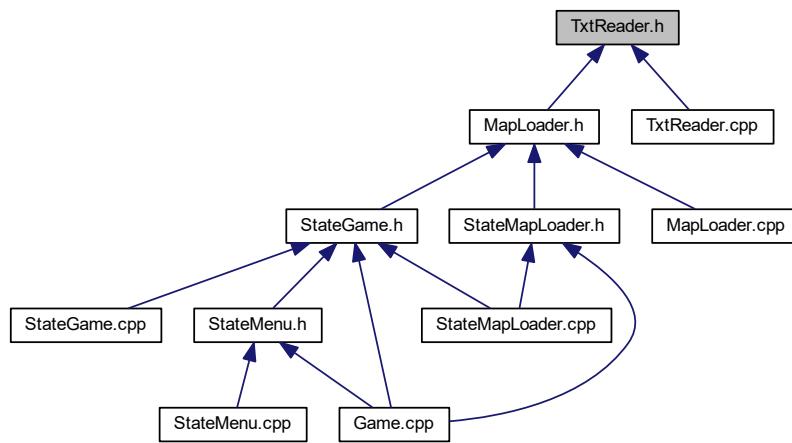


## 11.111 TxtReader.h File Reference

```
#include "FileReader.h"
#include <regex>
Include dependency graph for TxtReader.h:
```



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



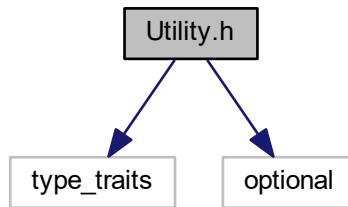
## Classes

- class [TxtReader](#)

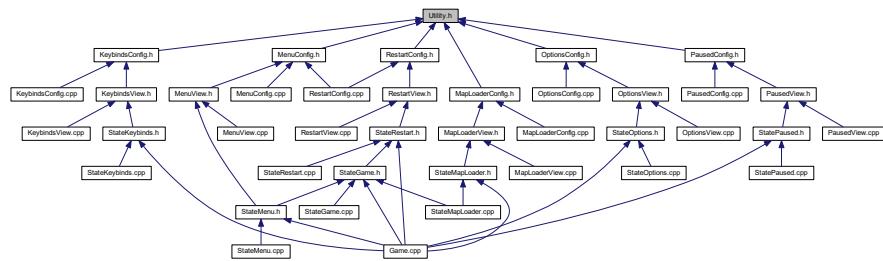
*Reads the .txt file into the vector of int.*

## 11.112 Utility.h File Reference

```
#include <type_traits>
#include <optional>
Include dependency graph for Utility.h:
```



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



## Classes

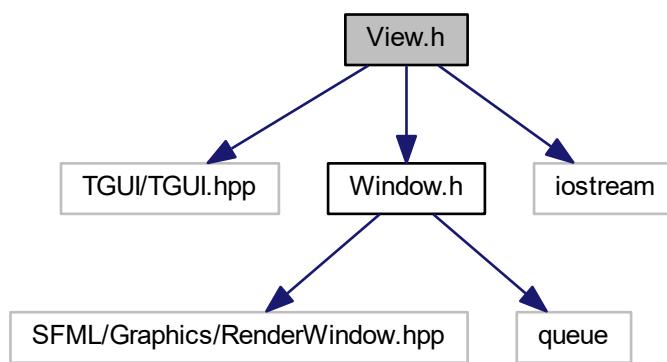
- struct `mapListOfHelper< T >`  
*A helper for mapListOf converter.*

# Functions

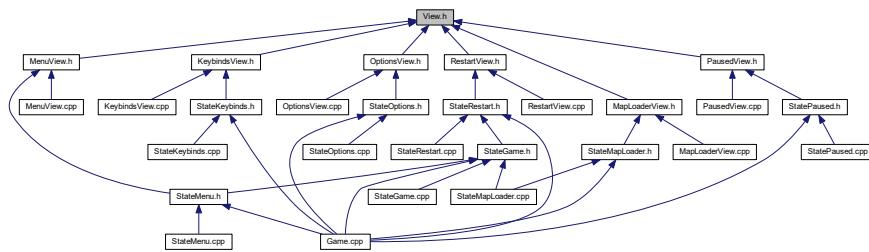
- template<typename E >  
constexpr auto **to\_underlying** (E e) noexcept  
*Converts an argument (Enum) to its underlying type.*
  - template<typename E >  
constexpr std::optional< int > **toInt** (E e) noexcept  
*Converts an argument (Enum) to integer.*
  - template<typename T >  
constexpr **mapListOfHelper**< T > **mapListOf** (T &item)

## 11.113 View.h File Reference

```
#include <TGUI/TGUI.hpp>
#include "Window.h"
#include <iostream>
Include dependency graph for View.h
```



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

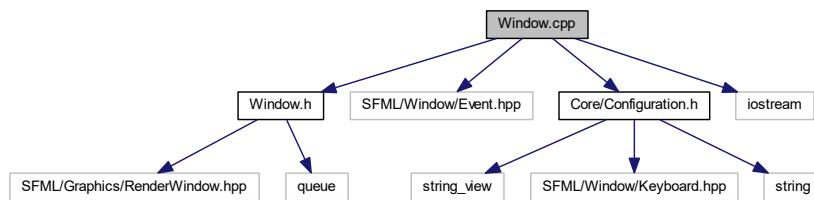


## Classes

- class `View< TWidgetPtr >`

## 11.114 Window.cpp File Reference

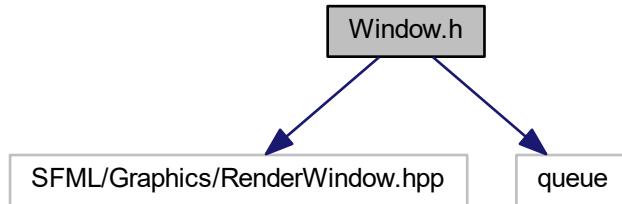
```
#include "Window.h"
#include <SFML/Window/Event.hpp>
#include <Core/Configuration.h>
#include <iostream>
Include dependency graph for Window.cpp:
```



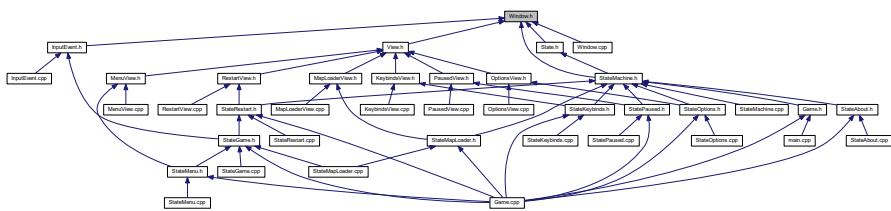
## 11.115 Window.h File Reference

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

Include dependency graph for Window.h:



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



## Classes

- class `Window`

*Responsible for drawing onto the screen.*



# Chapter 12

## Example Documentation

### 12.1 C:/Users/wenox/Desktop/cpp/platformer-2d/src/Core/Camera.cpp

Check Y position after sprite was moved on Y axis.

```
#include <Configuration.h>
#include "Camera.h"
Camera::Camera(const sf::View& view) noexcept
    : cameraView{view}
{
    this->setCameraConstants();
}
void Camera::updateX() noexcept {
    float solvedX = detectCollisionX();
    std::visit(overload{
        [&] (Bot&) { cameraView.setCenter(solvedX, bottomBorderBoundary); },
        [&] (Top&) { cameraView.setCenter(solvedX, topBorderBoundary); },
        [&] (None&) { cameraView.setCenter(solvedX, controller->getPosition().y + halvedSpriteHeight);
    },
        [&] (auto) noexcept { }
    }, previouslySolvedOtherAxis);
    storeJustSolvedForOtherAxis(solvedNow);
}
void Camera::updateY() noexcept {
    float solvedY = detectCollisionY();
    std::visit(overload{
        [&] (Left&) { cameraView.setCenter(leftBorderBoundary, solvedY); },
        [&] (Right&) { cameraView.setCenter(rightBorderBoundary, solvedY); },
        [&] (auto) noexcept { }
    }, previouslySolvedOtherAxis);
    storeJustSolvedForOtherAxis(solvedNow);
}
float Camera::detectCollisionX() noexcept {
    const auto cameraX = controller->getPosition().x + halvedSpriteWidth;
    float xCoord{};
    if (cameraX > rightBorderBoundary) {
        xCoord = rightBorderBoundary;
        solvedNow = CurrentCollision::Right;
    } else if (cameraX < leftBorderBoundary) {
        xCoord = leftBorderBoundary;
        solvedNow = CurrentCollision::Left;
    } else {
        xCoord = cameraX;
        solvedNow = CurrentCollision::None;
    }
    return xCoord;
}
float Camera::detectCollisionY() noexcept {
    const auto cameraY = controller->getPosition().y + halvedSpriteHeight;
    float yCoord{};
    if (cameraY > bottomBorderBoundary) {
        yCoord = bottomBorderBoundary;
        solvedNow = CurrentCollision::Bot;
    } else if (cameraY < topBorderBoundary) {
        yCoord = topBorderBoundary;
        solvedNow = CurrentCollision::Top;
    } else {
        yCoord = cameraY;
        solvedNow = CurrentCollision::None;
    }
}
```

```
    return yCoord;
}
sf::View& Camera::getCameraView() noexcept {
    return cameraView;
}
void Camera::setController(Entity& controllingEntity) noexcept {
    this->controller = &controllingEntity.getSprite();
    this->setControllerConstants();
}
void Camera::storeJustSolvedForOtherAxis(CurrentCollision wasCollision) noexcept {
    switch (wasCollision) {
        case CurrentCollision::Left:
            previouslySolvedOtherAxis = Left{};
            break;
        case CurrentCollision::Right:
            previouslySolvedOtherAxis = Right{};
            break;
        case CurrentCollision::None:
            previouslySolvedOtherAxis = None{};
            break;
        case CurrentCollision::Top:
            previouslySolvedOtherAxis = Top{};
            break;
        case CurrentCollision::Bot:
            previouslySolvedOtherAxis = Bot{};
            break;
    }
}
void Camera::setCameraConstants() noexcept {
    cameraWidth = cameraView.getSize().x;
    cameraHeight = cameraView.getSize().y;
    halvedCameraWidth = cameraWidth / 2.0;
    halvedCameraHeight = cameraHeight / 2.0;
    mapWidth = config::blocksCountWidth * 32;
    mapHeight = config::blocksCountHeight * 32;
    leftBorderBoundary = halvedCameraWidth;
    topBorderBoundary = halvedCameraHeight;
    rightBorderBoundary = mapWidth - halvedCameraWidth;
    bottomBorderBoundary = mapHeight - halvedCameraHeight;
}
void Camera::setControllerConstants() noexcept {
    halvedSpriteWidth = controller->getTexture()->getSize().x / 2.0;
    halvedSpriteHeight = controller->getTexture()->getSize().y / 2.0;
}
```

## 12.2 Calculate

correct resolved non-colliding Y position.