COMP-345

v0.0.0

Generated by Doxygen 1.13.2

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Map::Cell Struct Reference

Represents a single cell in the map grid.

Public Attributes

- int x = 0
- int y = 0
- bool isWall = false
- bool isSpawner = false
- bool isTarget = false
- int flowDirectionX = 0
- int flowDirectionY = 0
- unsigned char flowDistance = flowDistanceMax

4.1.1 Detailed Description

Represents a single cell in the map grid.

Contains position, type, and flow field information for pathfinding

4.1.2 Member Data Documentation

4.1.2.1 flowDirectionX

```
int Map::Cell::flowDirectionX = 0
```

X component of flow direction

4.1.2.2 flowDirectionY

```
int Map::Cell::flowDirectionY = 0
```

Y component of flow direction

4.1.2.3 flowDistance

```
unsigned char Map::Cell::flowDistance = flowDistanceMax
```

Distance to target in flow field

4.1.2.4 isSpawner

```
bool Map::Cell::isSpawner = false
```

Whether this cell is an enemy spawner

4.1.2.5 isTarget

```
bool Map::Cell::isTarget = false
```

Whether this cell is a target

4.1.2.6 isWall

```
bool Map::Cell::isWall = false
```

Whether this cell is a wall

4.1.2.7 x

```
int Map::Cell::x = 0
```

X coordinate in the grid

4.1.2.8 y

```
int Map::Cell::y = 0
```

Y coordinate in the grid

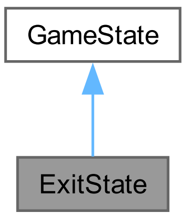
The documentation for this struct was generated from the following file:

- src/include/map/Map.h

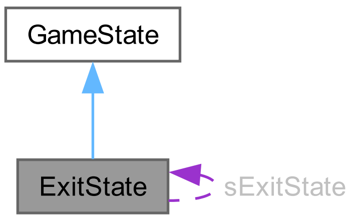
4.2 ExitState Class Reference

```
#include <ExitState.h>
```

Inheritance diagram for ExitState:



Collaboration diagram for ExitState:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static ExitState ∗ get ()

Private Member Functions

- ExitState ()

Static Private Attributes

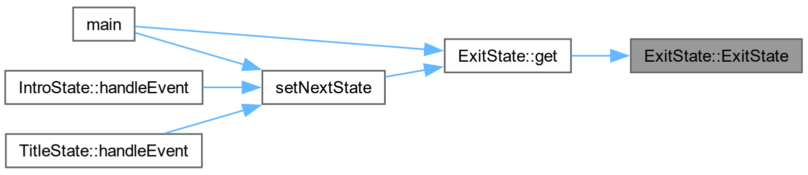
- static ExitState sExitState

4.2.1 Constructor & Destructor Documentation

4.2.1.1 ExitState()

```
ExitState::ExitState ()  [private]
```

Here is the caller graph for this function:



4.2.2 Member Function Documentation

4.2.2.1 enter()

```
bool ExitState::enter ()  [override], [virtual]
```

Implements GameState.

4.2.2.2 exit()

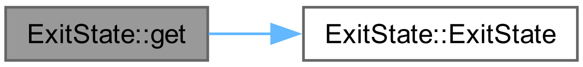
```
bool ExitState::exit ()  [override], [virtual]
```

Implements GameState.

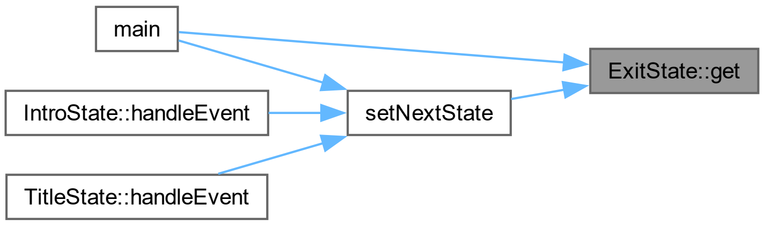
4.2.2.3 get()

```
ExitState * ExitState::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.2.4 handleEvent()

```
void ExitState::handleEvent (
             SDL_Event & e)  [override], [virtual]
```

Implements GameState.

4.2.2.5 render()

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

Implements GameState.

4.2.2.6 update()

```
void ExitState::update ()  [override], [virtual]
```

Implements GameState.

4.2.3 Member Data Documentation

4.2.3.1 sExitState

```
ExitState ExitState::sExitState  [static], [private]
```

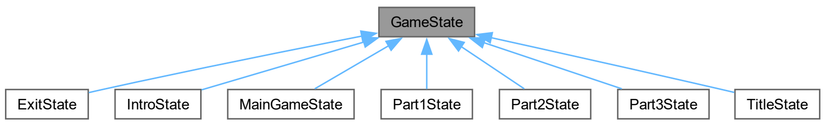
The documentation for this class was generated from the following files:

- src/include/states/ExitState.h
- src/states/ExitState.cpp

4.3 GameState Class Reference

```
#include <GameState.h>
```

Inheritance diagram for GameState:



Public Member Functions

- virtual bool enter ()=0
- virtual bool exit ()=0
- virtual void handleEvent (SDL_Event &e)=0
- virtual void update ()=0
- virtual void render ()=0
- virtual ∼GameState ()=default

4.3.1 Constructor & Destructor Documentation

4.3.1.1 ∼GameState()

```
virtual GameState::∼GameState ()  [virtual], [default]
```

4.3.2 Member Function Documentation

4.3.2.1 enter()

```
virtual bool GameState::enter ()  [pure virtual]
```

Implemented in ExitState, IntroState, MainGameState, Part1State, Part2State, Part3State, and TitleState.

4.3.2.2 exit()

```
virtual bool GameState::exit ()  [pure virtual]
```

Implemented in ExitState, IntroState, MainGameState, Part1State, Part2State, Part3State, and TitleState.

4.3.2.3 handleEvent()

```
virtual void GameState::handleEvent (
             SDL_Event & e)  [pure virtual]
```

Implemented in ExitState, IntroState, MainGameState, Part1State, Part2State, Part3State, and TitleState.

4.3.2.4 render()

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

Implemented in ExitState, IntroState, MainGameState, Part1State, Part2State, Part3State, and TitleState.

4.3.2.5 update()

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

Implemented in ExitState, IntroState, MainGameState, Part1State, Part2State, Part3State, and TitleState.

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

- src/include/states/GameState.h

4.4 Global Class Reference

```
#include <Global.h>
```

Static Public Attributes

- static const int kScreenWidth { Map::PIXELS_PER_CELL ∗ 15 }
- static const int kScreenHeight { Map::PIXELS_PER_CELL ∗ 11 }

4.4.1 Member Data Documentation

4.4.1.1 kScreenHeight

```
const int Global::kScreenHeight { Map::PIXELS_PER_CELL * 11 }  [static]
```

4.4.1.2 kScreenWidth

```
const int Global::kScreenWidth { Map::PIXELS_PER_CELL * 15 } [static]
```

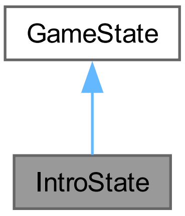
The documentation for this class was generated from the following file:

- src/include/Global.h

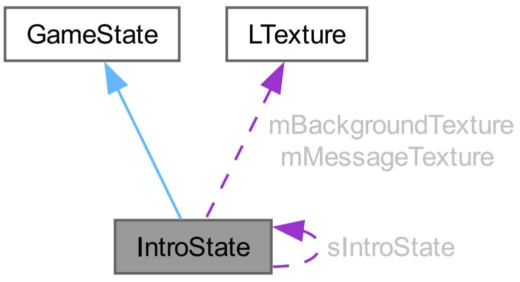
4.5 IntroState Class Reference

```
#include <IntroState.h>
```

Inheritance diagram for IntroState:



Collaboration diagram for IntroState:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static IntroState ∗ get ()

Private Member Functions

- IntroState ()

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture

Static Private Attributes

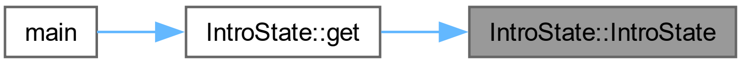
- static IntroState sIntroState

4.5.1 Constructor & Destructor Documentation

4.5.1.1 IntroState()

```
IntroState::IntroState () [private]
```

Here is the caller graph for this function:



4.5.2 Member Function Documentation

4.5.2.1 enter()

```
bool IntroState::enter () [override], [virtual]
```

Implements GameState.

4.5.2.2 exit()

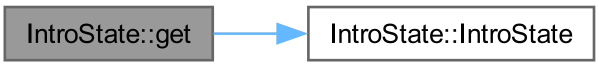
```
bool IntroState::exit ()  [override], [virtual]
```

Implements GameState.

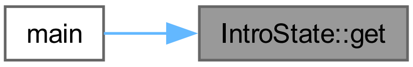
4.5.2.3 get()

```
IntroState * IntroState::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:

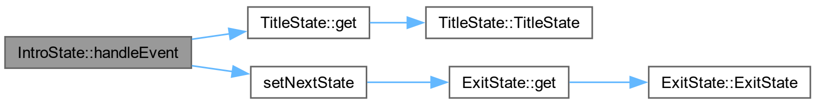


4.5.2.4 handleEvent()

```
void IntroState::handleEvent (
            SDL_Event & e)  [override], [virtual]
```

Implements GameState.

Here is the call graph for this function:



4.5.2.5 render()

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

Implements GameState.

4.5.2.6 update()

```
void IntroState::update ()  [override], [virtual]
```

Implements GameState.

4.5.3 Member Data Documentation

4.5.3.1 mBackgroundTexture

```
LTexture IntroState::mBackgroundTexture  [private]
```

4.5.3.2 mMessageTexture

```
LTexture IntroState::mMessageTexture  [private]
```

4.5.3.3 sIntroState

```
IntroState IntroState::sIntroState  [static], [private]
```

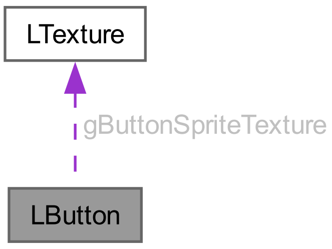
The documentation for this class was generated from the following files:

- src/include/states/IntroState.h
- src/states/IntroState.cpp

4.6 LButton Class Reference

```
#include <LButton.h>
```

Collaboration diagram for LButton:



Public Member Functions

- LButton ()
- void setPosition (float x, float y)
- void handleEvent (SDL_Event ∗e)
- bool setText (const std::string &text, SDL_Color textColor)
- void render ()
- bool isClicked () const

Static Public Attributes

- static constexpr int kButtonWidth = 300
- static constexpr int kButtonHeight = 50

Private Types

- enum eButtonSprite { eButtonSpriteMouseOut = 0 , eButtonSpriteMouseOverMotion = 1 , eButtonSpriteMouseDown = 2 , eButtonSpriteMouseUp = 3 }

Private Attributes

- SDL_FPoint mPosition
- eButtonSprite mCurrentSprite
- LTexture gButtonSpriteTexture

4.6.1 Member Enumeration Documentation

4.6.1.1 eButtonSprite

```
enum LButton::eButtonSprite  [private]
```

Enumerator

eButtonSpriteMouseOut	
eButtonSpriteMouseOverMotion	
eButtonSpriteMouseDown	
eButtonSpriteMouseUp	

4.6.2 Constructor & Destructor Documentation

4.6.2.1 LButton()

```
LButton::LButton ()
```

4.6.3 Member Function Documentation

4.6.3.1 handleEvent()

```
void LButton::handleEvent (
             SDL_Event * e)
```

4.6.3.2 isClicked()

```
bool LButton::isClicked () const
```

4.6.3.3 render()

```
void LButton::render ()
```

4.6.3.4 setPosition()

```
void LButton::setPosition (
             float x,
             float y)
```

4.6.3.5 setText()

```
bool LButton::setText (
             const std::string & text,
             SDL_Color textColor)
```

4.6.4 Member Data Documentation

4.6.4.1 gButtonSpriteTexture

```
LTexture LButton::gButtonSpriteTexture  [private]
```

4.6.4.2 kButtonHeight

```
int LButton::kButtonHeight = 50  [static], [constexpr]
```

4.6.4.3 kButtonWidth

```
int LButton::kButtonWidth = 300  [static], [constexpr]
```

4.6.4.4 mCurrentSprite

```
eButtonSprite LButton::mCurrentSprite  [private]
```

4.6.4.5 mPosition

```
SDL_FPoint LButton::mPosition  [private]
```

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

- src/include/ui/LButton.h
- src/ui/LButton.cpp

4.7 LTexture Class Reference

```
#include <LTexture.h>
```

Public Member Functions

- LTexture ()
- ∼LTexture ()
- bool loadFromFile (std::string path)
- bool loadFromRenderedText (std::string textureText, SDL_Color textColor)
- void destroy ()
- void setColor (Uint8 r, Uint8 g, Uint8 b)
- void setAlpha (Uint8 alpha)
- void setBlending (SDL_BlendMode blendMode)
- void render (float x, float y, SDL_FRect ∗clip=nullptr, float width=kOriginalSize, float height=kOriginalSize, double degrees=0.0, SDL_FPoint ∗center=nullptr, SDL_FlipMode flipMode=SDL_FLIP_NONE)
- int getWidth ()
- int getHeight ()

Static Public Attributes

- static constexpr float kOriginalSize = -1.f

Private Attributes

- SDL_Texture ∗ mTexture
- int mWidth
- int mHeight

4.7.1 Constructor & Destructor Documentation

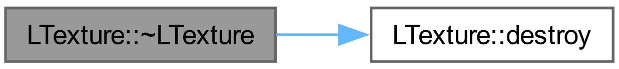
4.7.1.1 LTexture()

```
LTexture::LTexture ()
```

4.7.1.2 ∼LTexture()

```
LTexture::∼LTexture ()
```

Here is the call graph for this function:

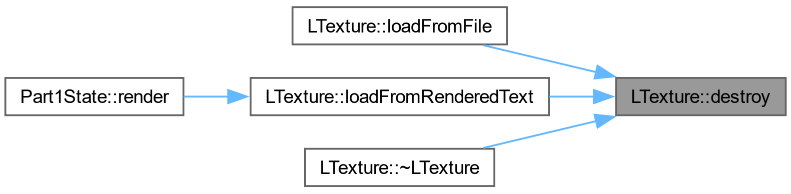


4.7.2 Member Function Documentation

4.7.2.1 destroy()

```
void LTexture::destroy ()
```

Here is the caller graph for this function:



4.7.2.2 getHeight()

```
int LTexture::getHeight ()
```

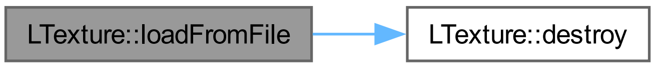
4.7.2.3 getWidth()

```
int LTexture::getWidth ()
```

4.7.2.4 loadFromFile()

```
bool LTexture::loadFromFile (
             std::string path)
```

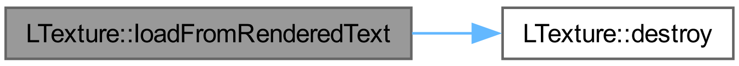
Here is the call graph for this function:



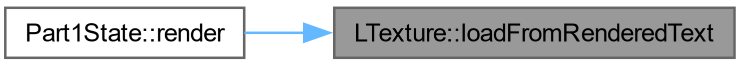
4.7.2.5 loadFromRenderedText()

```
bool LTexture::loadFromRenderedText (
             std::string textureText,
             SDL_Color textColor)
```

Here is the call graph for this function:



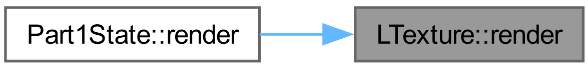
Here is the caller graph for this function:



4.7.2.6 render()

```
void LTexture::render (
             float x,
             float y,
             SDL_FRect * clip = nullptr,
             float width = kOriginalSize,
             float height = kOriginalSize,
             double degrees = 0.0,
             SDL_FPoint * center = nullptr,
             SDL_FlipMode flipMode = SDL_FLIP_NONE)
```

Here is the caller graph for this function:



4.7.2.7 setAlpha()

```
void LTexture::setAlpha (
             Uint8 alpha)
```

4.7.2.8 setBlending()

```
void LTexture::setBlending (
             SDL_BlendMode blendMode)
```

4.7.2.9 setColor()

```
void LTexture::setColor (
             Uint8 r,
             Uint8 g,
             Uint8 b)
```

4.7.3 Member Data Documentation

4.7.3.1 kOriginalSize

```
float LTexture::kOriginalSize = -1.f  [static], [constexpr]
```

4.7.3.2 mHeight

```
int LTexture::mHeight  [private]
```

4.7.3.3 mTexture

```
SDL_Texture* LTexture::mTexture  [private]
```

4.7.3.4 mWidth

```
int LTexture::mWidth  [private]
```

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

- src/include/ui/LTexture.h
- src/ui/LTexture.cpp

4.8 LTimer Class Reference

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

Public Member Functions

- LTimer ()
- void start ()
- void stop ()
- void pause ()
- void unpause ()
- Uint64 getTicksNS ()
- bool isStarted ()
- bool isPaused ()

Private Attributes

- Uint64 mStartTicks
- Uint64 mPausedTicks
- bool mPaused
- bool mStarted

4.8.1 Constructor & Destructor Documentation

4.8.1.1 LTimer()

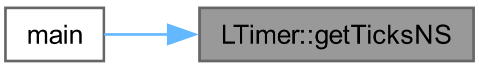
```
LTimer::LTimer ()
```

4.8.2 Member Function Documentation

4.8.2.1 getTicksNS()

```
Uint64 LTimer::getTicksNS ()
```

Here is the caller graph for this function:



4.8.2.2 isPaused()

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

4.8.2.3 isStarted()

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

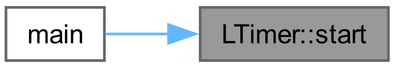
4.8.2.4 pause()

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

4.8.2.5 start()

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

Here is the caller graph for this function:



4.8.2.6 stop()

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

4.8.2.7 unpause()

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

4.8.3 Member Data Documentation

4.8.3.1 mPaused

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

4.8.3.2 mPausedTicks

```
Uint64 LTimer::mPausedTicks [private]
```

4.8.3.3 mStarted

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

4.8.3.4 mStartTicks

```
Uint64 LTimer::mStartTicks [private]
```

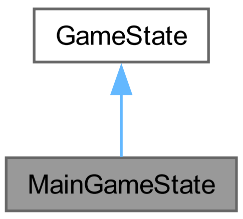
The documentation for this class was generated from the following files:

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

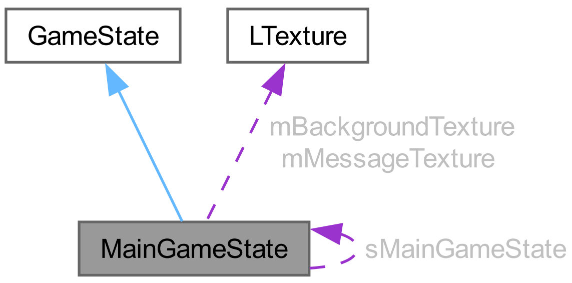
4.9 MainGameState Class Reference

```
#include <MainGameState.h>
```

Inheritance diagram for MainGameState:



Collaboration diagram for MainGameState:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static MainGameState ∗ get ()

Private Member Functions

- MainGameState ()

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture

Static Private Attributes

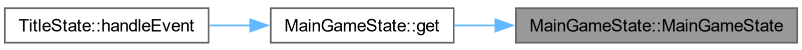
- static MainGameState sMainGameState

4.9.1 Constructor & Destructor Documentation

4.9.1.1 MainGameState()

```
MainGameState::MainGameState ()  [private]
```

Here is the caller graph for this function:



4.9.2 Member Function Documentation

4.9.2.1 enter()

```
bool MainGameState::enter ()  [override], [virtual]
```

Implements GameState.

4.9.2.2 exit()

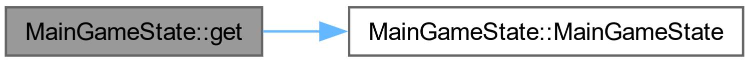
```
bool MainGameState::exit ()  [override], [virtual]
```

Implements GameState.

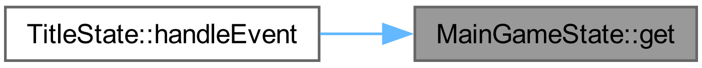
4.9.2.3 get()

```
MainGameState * MainGameState::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.9.2.4 handleEvent()

```
void MainGameState::handleEvent (
             SDL_Event & e)  [override], [virtual]
```

Implements GameState.

4.9.2.5 render()

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

Implements GameState.

4.9.2.6 update()

```
void MainGameState::update ()  [override], [virtual]
```

Implements GameState.

4.9.3 Member Data Documentation

4.9.3.1 mBackgroundTexture

```
LTexture MainGameState::mBackgroundTexture  [private]
```

4.9.3.2 mMessageTexture

```
LTexture MainGameState::mMessageTexture  [private]
```

4.9.3.3 sMainGameState

```
MainGameState MainGameState::sMainGameState  [static], [private]
```

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

- src/include/states/parts/MainGameState.h
- src/states/parts/MainGameState.cpp

4.10 Map Class Reference

```
#include <Map.h>
```

Classes

- struct Cell

  Represents a single cell in the map grid.

Public Member Functions

- Map (SDL_Renderer ∗renderer, int cellCountX, int cellCountY)

  Constructs a new Map object.
- void draw (SDL_Renderer ∗renderer)

  Draws all cells in the map.
- bool isCellWall (int x, int y)

  Checks if a cell is a wall.
- void setCellWall (int x, int y, bool setWall)

  Sets a cell's wall status.
- bool isTarget (int x, int y)

  Checks if a cell is a target.
- void setTarget (int x, int y)

  Sets a cell as the new target.
- bool isSpawner (int x, int y)

  Checks if a cell is a spawner.
- void setSpawner (int x, int y)

  Sets a cell as the new spawner.
- bool isValidPath ()

  Checks if a valid path exists from spawner to target.
- Vector2D getTargetPos ()

  Gets the position of the first target cell found.
- Vector2D getFlowNormal (int x, int y)

  Gets the normalized flow direction vector for a cell.

Static Public Attributes

- static constexpr int PIXELS_PER_CELL = 48

  Number of pixels each cell occupies on screen.

Private Member Functions

- bool isInbounds (int x, int y)

  Checks if a coordinate is within map boundaries.
- void drawCell (SDL_Renderer ∗renderer, const Cell &cell)

  Draws an individual cell.
- void calculateFlowField ()

  Recalculates the flow field for pathfinding.
- void calculateDistances ()

  Calculates distance values from target cells using BFS.
- void calculateFlowDirections ()

  Determines flow directions based on calculated distances.

Private Attributes

- std::vector< Cell > cells
- const int cellCountX
- const int cellCountY
- SDL_Texture ∗ textureCellWall
- SDL_Texture ∗ textureCellTarget
- SDL_Texture ∗ textureCellSpawner
- SDL_Texture ∗ textureCellEmpty
- SDL_Texture ∗ textureCellArrowUp
- SDL_Texture ∗ textureCellArrowRight
- SDL_Texture ∗ textureCellArrowDown
- SDL_Texture ∗ textureCellArrowLeft

Static Private Attributes

- static const unsigned char flowDistanceMax = 255

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Map()

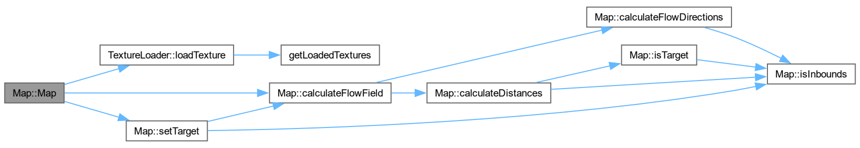
```
Map::Map (
             SDL_Renderer ∗ renderer,
             int setCellCountX,
             int setCellCountY)
```

Constructs a new Map object.

Parameters

renderer	The SDL renderer used for loading textures.
setCellCountX	The number of cells along the X-axis.
setCellCountY	The number of cells along the Y-axis.

Initializes cell textures, creates a grid of cells, sets the default target at the center, and calculates the initial flow field. Here is the call graph for this function:



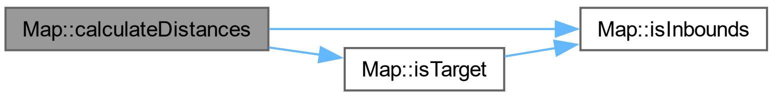
4.10.2 Member Function Documentation

4.10.2.1 calculateDistances()

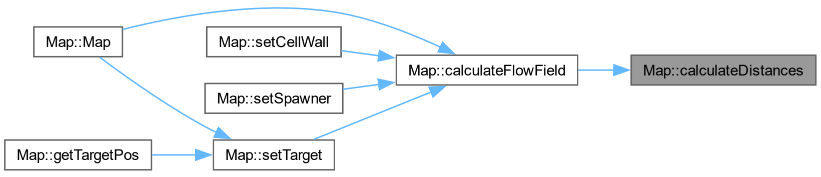
```
void Map::calculateDistances ()  [private]
```

Calculates distance values from target cells using BFS.

Initializes queue with target cell (distance 0), propagates distances to all reachable non-wall cells. Unreachable cells keep flowDistanceMax. Here is the call graph for this function:



Here is the caller graph for this function:

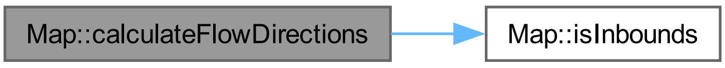


4.10.2.2 calculateFlowDirections()

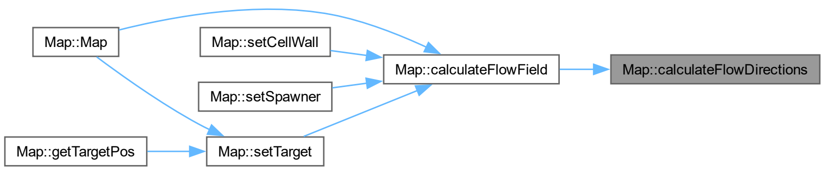
```
void Map::calculateFlowDirections ()  [private]
```

Determines flow directions based on calculated distances.

For each cell, sets flow direction towards the neighbor with the smallest distance value. Here is the call graph for this function:



Here is the caller graph for this function:

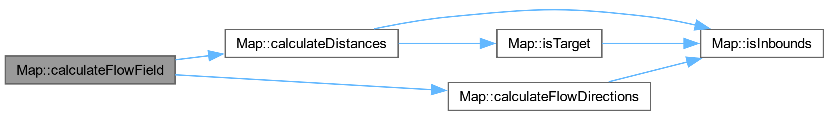


4.10.2.3 calculateFlowField()

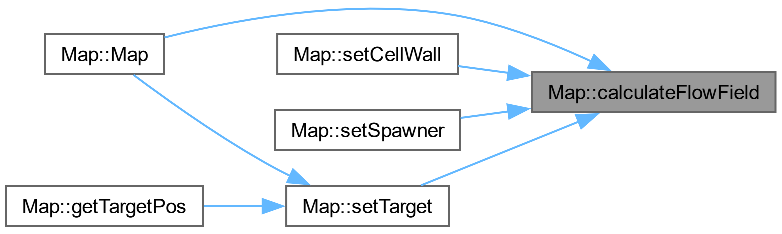
```
void Map::calculateFlowField ()  [private]
```

Recalculates the flow field for pathfinding.

Resets all flow data, calculates distances from targets using BFS, then determines optimal flow directions for each cell. Here is the call graph for this function:



Here is the caller graph for this function:



4.10.2.4 draw()

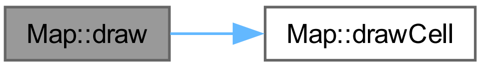
```
void Map::draw (
             SDL_Renderer * renderer)
```

Draws all cells in the map.

Parameters

renderer	The SDL renderer used for drawing.

Iterates through all cells and draws them using their respective textures based on type and flow direction. Here is the call graph for this function:



4.10.2.5 drawCell()

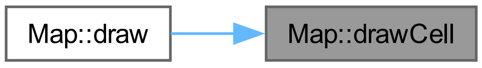
```
void Map::drawCell (
             SDL_Renderer * renderer,
             const Cell & cell)  [private]
```

Draws an individual cell.

Parameters

renderer	The SDL renderer used for drawing.
cell	The cell to be drawn.

Selects the appropriate texture based on cell properties. Here is the caller graph for this function:



4.10.2.6 getFlowNormal()

```
Vector2D Map::getFlowNormal (
             int x,
             int y)
```

Gets the normalized flow direction vector for a cell.

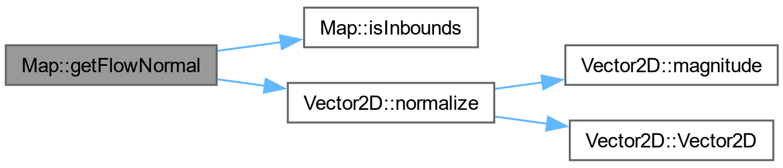
Parameters

x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Returns

Vector2D Normalized flow direction vector (zero vector if invalid).

Here is the call graph for this function:



4.10.2.7 getTargetPos()

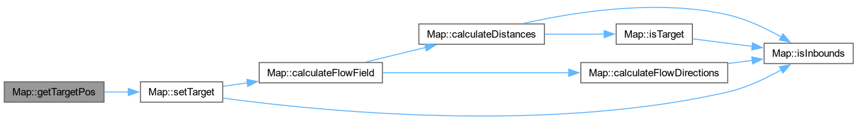
```
Vector2D Map::getTargetPos ()
```

Gets the position of the first target cell found.

Returns

Vector2D Position of the target in cell coordinates.

Falls back to center coordinates if no target found and sets one there. Here is the call graph for this function:



4.10.2.8 isCellWall()

```
bool Map::isCellWall (
             int x,
             int y)
```

Checks if a cell is a wall.

Parameters

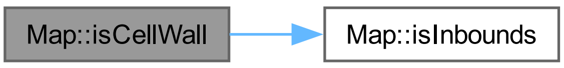
x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Returns

true If the cell is a wall.

false If the cell is not a wall or coordinates are invalid.

Here is the call graph for this function:



4.10.2.9 isInbounds()

```
bool Map::isInbounds (
             int x,
             int y) [private]
```

Checks if a coordinate is within map boundaries.

Parameters

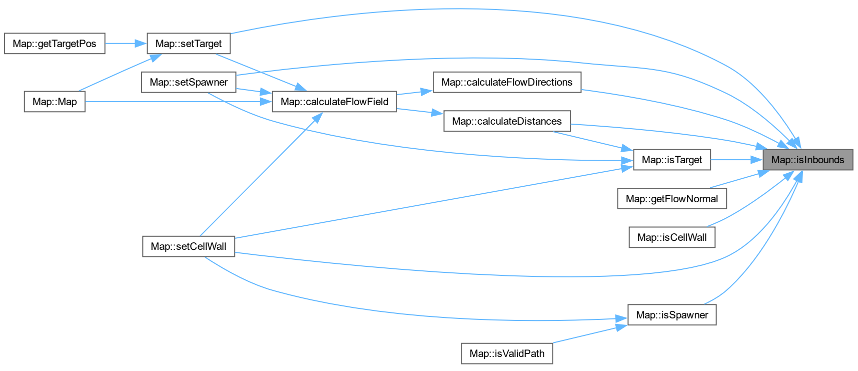
x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Returns

true If the coordinates are within valid bounds.

false If the coordinates are out of bounds.

Here is the caller graph for this function:



4.10.2.10 isSpawner()

```
bool Map::isSpawner (
            int x,
            int y)
```

Checks if a cell is a spawner.

Parameters

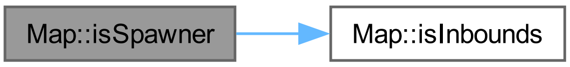
x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Returns

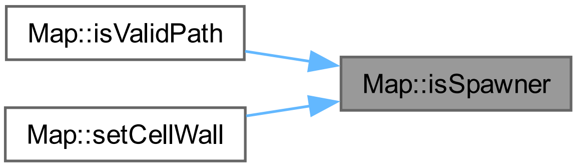
true If the cell is a spawner.

false If the cell is not a spawner or coordinates are invalid.

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.2.11 isTarget()

```
bool Map::isTarget (
            int x,
            int y)
```

Checks if a cell is a target.

Parameters

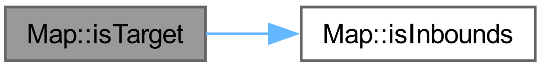
x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Returns

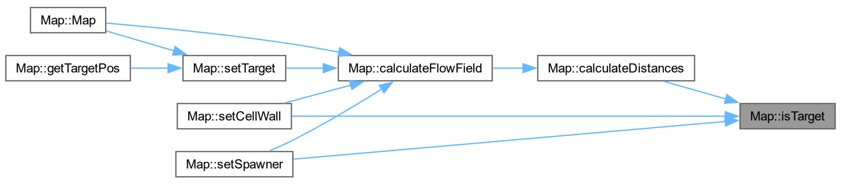
true If the cell is a target.

false If the cell is not a target or coordinates are invalid.

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.2.12 isValidPath()

```
bool Map::isValidPath ()
```

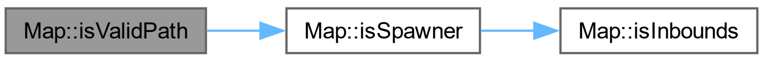
Checks if a valid path exists from spawner to target.

Returns

true If spawner exists and is reachable (valid flow path to target).

false If no spawner or spawner is unreachable.

Here is the call graph for this function:



4.10.2.13 setCellWall()

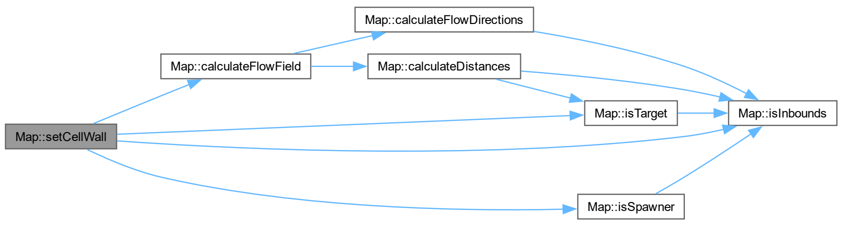
```
void Map::setCellWall (
             int x,
             int y,
             bool setWall)
```

Sets a cell's wall status.

Parameters

x	X-coordinate of the cell.
y	Y-coordinate of the cell.
setWall	true to make the cell a wall, false to clear.

Does nothing if coordinates are invalid or cell is target/spawner. Recalculates flow field after change. Here is the call graph for this function:



4.10.2.14 setSpawner()

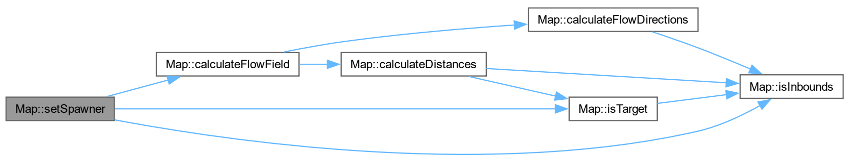
```
void Map::setSpawner (
             int x,
             int y)
```

Sets a cell as the new spawner.

Parameters

x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Clears existing spawners, sets new spawner, marks it non-wall, and recalculates flow field. Does nothing if coordinates are invalid or cell is a target. Here is the call graph for this function:



4.10.2.15 setTarget()

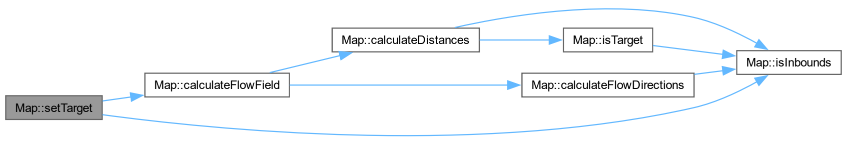
```
void Map::setTarget (
             int x,
             int y)
```

Sets a cell as the new target.

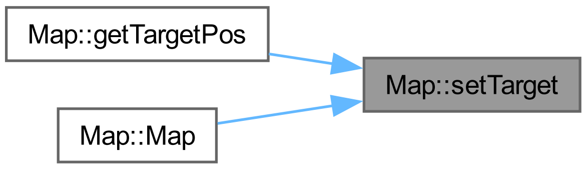
Parameters

x	X-coordinate of the cell.
y	Y-coordinate of the cell.

Clears existing targets, sets new target, marks it non-wall/non-spawner, and recalculates flow field. Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3 Member Data Documentation

4.10.3.1 cellCountX

```
const int Map::cellCountX  [private]
```

4.10.3.2 cellCountY

```
const int Map::cellCountY  [private]
```

Grid dimensions

4.10.3.3 cells

```
std::vector<Cell> Map::cells  [private]
```

Grid cells storage

4.10.3.4 flowDistanceMax

```
const unsigned char Map::flowDistanceMax = 255  [static], [private]
```

4.10.3.5 PIXELS_PER_CELL

```
int Map::PIXELS_PER_CELL = 48  [static], [constexpr]
```

Number of pixels each cell occupies on screen.

4.10.3.6 textureCellArrowDown

```
SDL_Texture* Map::textureCellArrowDown  [private]
```

Texture for downward flow

4.10.3.7 textureCellArrowLeft

```
SDL_Texture* Map::textureCellArrowLeft  [private]
```

Texture for leftward flow

4.10.3.8 textureCellArrowRight

```
SDL_Texture* Map::textureCellArrowRight  [private]
```

Texture for rightward flow

4.10.3.9 textureCellArrowUp

```
SDL_Texture* Map::textureCellArrowUp  [private]
```

Texture for upward flow

4.10.3.10 textureCellEmpty

```
SDL_Texture* Map::textureCellEmpty  [private]
```

Texture for empty cells

4.10.3.11 textureCellSpawner

```
SDL_Texture* Map::textureCellSpawner  [private]
```

Texture for spawner cells

4.10.3.12 textureCellTarget

```
SDL_Texture* Map::textureCellTarget [private]
```

Texture for target cells

4.10.3.13 textureCellWall

```
SDL_Texture* Map::textureCellWall [private]
```

Texture for wall cells

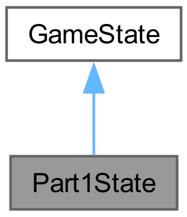
The documentation for this class was generated from the following files:

- src/include/map/Map.h
- src/map/Map.cpp

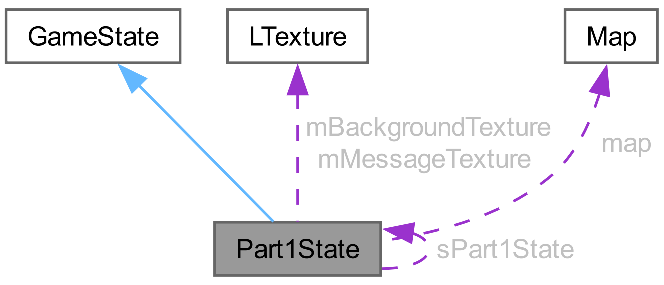
4.11 Part1State Class Reference

```
#include <Part1State.h>
```

Inheritance diagram for Part1State:



Collaboration diagram for Part1State:



Public Member Functions

- bool enter () override

    Initialize the state.
- bool exit () override

    Clean up the state.
- void handleEvent (SDL_Event &e) override

    Handle input events.
- void update () override

    Update the state.
- void render () override

    Render the state.

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static Part1State ∗ get ()

    Get the singleton instance of Part1State.

Private Member Functions

- Part1State ()

    Default constructor for Part1State.

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture
- int mouseDownStatus = 0
- int keyDownStatus = 0
- Map ∗ map = nullptr

    Pointer to the map being edited.

Static Private Attributes

- static Part1State sPart1State

    Singleton instance of the Part1State.

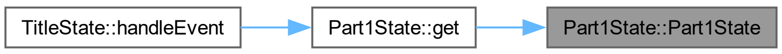
4.11.1 Constructor & Destructor Documentation

4.11.1.1 Part1State()

```
Part1State::Part1State ()  [private]
```

Default constructor for Part1State.

Here is the caller graph for this function:



4.11.2 Member Function Documentation

4.11.2.1 enter()

```
bool Part1State::enter ()  [override], [virtual]
```

Initialize the state.

Creates a new Map instance sized to fit the screen dimensions

Returns

true if initialization was successful

Implements GameState.

4.11.2.2 exit()

```
bool Part1State::exit ()  [override], [virtual]
```

Clean up the state.

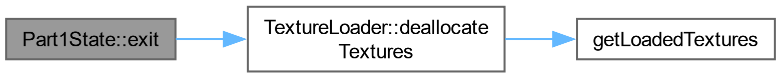
Deallocates textures and frees the map

Returns

true if cleanup was successful

Implements GameState.

Here is the call graph for this function:



4.11.2.3 get()

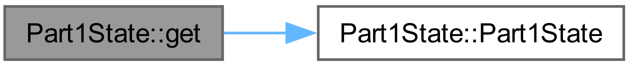
```
Part1State * Part1State::get ()  [static]
```

Get the singleton instance of Part1State.

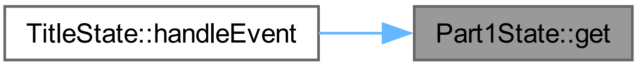
Returns

Pointer to the Part1State singleton

Here is the call graph for this function:



Here is the caller graph for this function:



4.11.2.4 handleEvent()

```
void Part1State::handleEvent (
            SDL_Event & e)  [override], [virtual]
```

Handle input events.

Processes mouse and keyboard input to:

- Place/remove walls with left/right click
- Set target/spawner with Shift + left/right click

Parameters

e	SDL event to process

Implements GameState.

4.11.2.5 render()

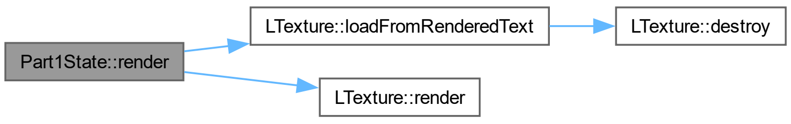
```
void Part1State::render ()  [override], [virtual]
```

Render the state.

Draws the map and displays path validity status in the top-left corner

Implements GameState.

Here is the call graph for this function:



4.11.2.6 update()

```
void Part1State::update ()  [override], [virtual]
```

Update the state.

Currently empty, reserved for future implementation

Implements GameState.

4.11.3 Member Data Documentation

4.11.3.1 keyDownStatus

```
int Part1State::keyDownStatus = 0  [private]
```

4.11.3.2 map

```
Map* Part1State::map = nullptr  [private]
```

Pointer to the map being edited.

nullptr if no map is currently loaded

4.11.3.3 mBackgroundTexture

```
LTexture Part1State::mBackgroundTexture  [private]
```

4.11.3.4 mMessageTexture

```
LTexture Part1State::mMessageTexture [private]
```

4.11.3.5 mouseDownStatus

```
int Part1State::mouseDownStatus = 0 [private]
```

4.11.3.6 sPart1State

```
Part1State Part1State::sPart1State [static], [private]
```

Singleton instance of the Part1State.

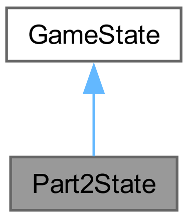
The documentation for this class was generated from the following files:

- src/include/states/parts/Part1State.h
- src/states/parts/Part1State.cpp

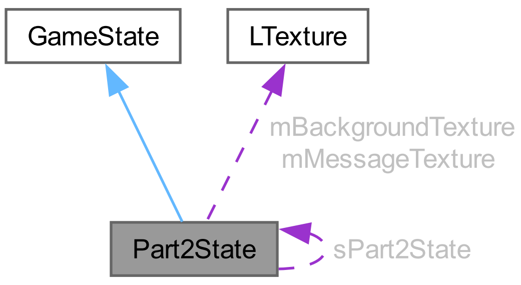
4.12 Part2State Class Reference

```
#include <Part2State.h>
```

Inheritance diagram for Part2State:



Collaboration diagram for Part2State:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static Part2State ∗ get ()

Private Member Functions

- Part2State ()

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture

Static Private Attributes

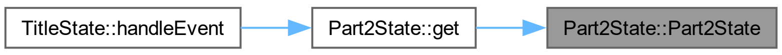
- static Part2State sPart2State

4.12.1 Constructor & Destructor Documentation

4.12.1.1 Part2State()

```
Part2State::Part2State ()  [private]
```

Here is the caller graph for this function:



4.12.2 Member Function Documentation

4.12.2.1 enter()

```
bool Part2State::enter ()  [override], [virtual]
```

Implements GameState.

4.12.2.2 exit()

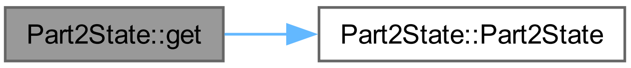
```
bool Part2State::exit ()  [override], [virtual]
```

Implements GameState.

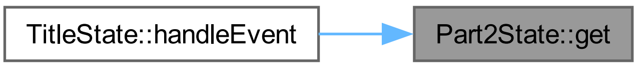
4.12.2.3 get()

```
Part2State * Part2State::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.12.2.4 handleEvent()

```
void Part2State::handleEvent (
            SDL_Event & e)  [override], [virtual]
```

Implements GameState.

4.12.2.5 render()

```
void Part2State::render ()  [override], [virtual]
```

Implements GameState.

4.12.2.6 update()

```
void Part2State::update ()  [override], [virtual]
```

Implements GameState.

4.12.3 Member Data Documentation

4.12.3.1 mBackgroundTexture

```
LTexture Part2State::mBackgroundTexture  [private]
```

4.12.3.2 mMessageTexture

```
LTexture Part2State::mMessageTexture  [private]
```

4.12.3.3 sPart2State

```
Part2State Part2State::sPart2State  [static], [private]
```

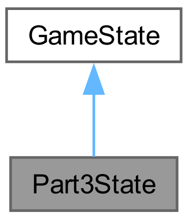
The documentation for this class was generated from the following files:

- src/include/states/parts/Part2State.h
- src/states/parts/Part2State.cpp

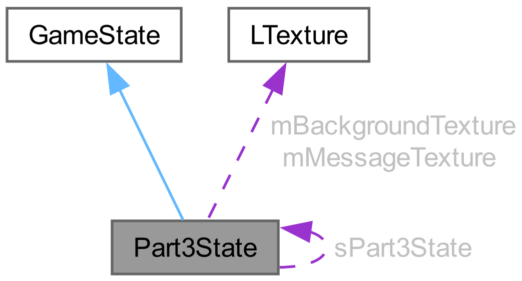
4.13 Part3State Class Reference

```
#include <Part3State.h>
```

Inheritance diagram for Part3State:



Collaboration diagram for Part3State:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static Part3State ∗ get ()

Private Member Functions

- Part3State ()

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture

Static Private Attributes

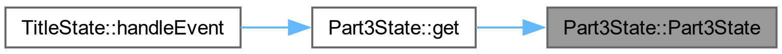
- static Part3State sPart3State

4.13.1 Constructor & Destructor Documentation

4.13.1.1 Part3State()

```
Part3State::Part3State ()  [private]
```

Here is the caller graph for this function:



4.13.2 Member Function Documentation

4.13.2.1 enter()

```
bool Part3State::enter ()  [override], [virtual]
```

Implements GameState.

4.13.2.2 exit()

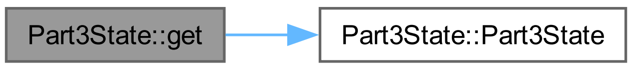
```
bool Part3State::exit ()  [override], [virtual]
```

Implements GameState.

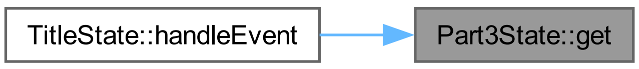
4.13.2.3 get()

```
Part3State * Part3State::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.13.2.4 handleEvent()

```
void Part3State::handleEvent (
            SDL_Event & e)  [override], [virtual]
```

Implements GameState.

4.13.2.5 render()

```
void Part3State::render ()  [override], [virtual]
```

Implements GameState.

4.13.2.6 update()

```
void Part3State::update ()  [override], [virtual]
```

Implements GameState.

4.13.3 Member Data Documentation

4.13.3.1 mBackgroundTexture

```
LTexture Part3State::mBackgroundTexture  [private]
```

4.13.3.2 mMessageTexture

```
LTexture Part3State::mMessageTexture  [private]
```

4.13.3.3 sPart3State

```
Part3State Part3State::sPart3State  [static], [private]
```

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

- src/include/states/parts/Part3State.h
- src/states/parts/Part3State.cpp

4.14 TextureLoader Class Reference

```
#include <TextureLoader.h>
```

Static Public Member Functions

- static SDL_Texture ∗ loadTexture (SDL_Renderer ∗renderer, std::string filename)

  Load a texture from a file and return it.
- static void deallocateTextures ()

  Destroy all loaded textures and clear the texture cache.

Static Private Attributes

- static std::unordered_map< std::string, SDL_Texture ∗ > loadedTextures

  Static map storing all loaded textures.
- static const std::string TEXTURE_PATH = "assets/"

4.14.1 Member Function Documentation

4.14.1.1 deallocateTextures()

```
void TextureLoader::deallocateTextures () [static]
```

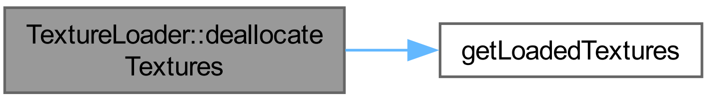
Destroy all loaded textures and clear the texture cache.

This function should be called during cleanup to prevent memory leaks. It iterates through all cached textures, properly destroys each one using SDL, and clears the texture map.

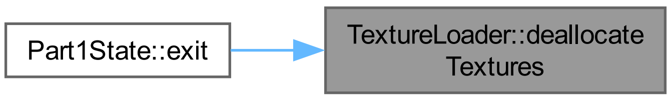
Warning

After calling this function, any existing pointers to previously loaded textures will be invalid

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.1.2 loadTexture()

```
SDL_Texture * TextureLoader::loadTexture (
            SDL_Renderer * renderer,
            std::string filename)  [static]
```

Load a texture from a file and return it.

This function implements texture caching. If the requested texture has already been loaded, it returns the cached version instead of loading it again. New textures are automatically added to the cache.

Parameters

renderer	The SDL renderer used to create the texture
filename	The path to the texture file, relative to the 'assets' folder

Returns

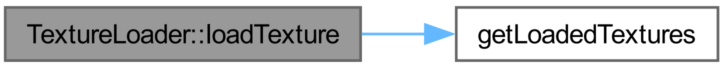
SDL_Texture∗ The loaded texture, or nullptr if loading failed

Note

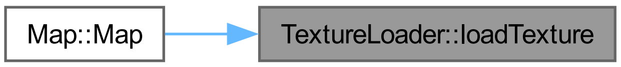
The returned texture has SDL_BLENDMODE_BLEND enabled by default

The function automatically manages the temporary SDL_Surface used during loading

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.2 Member Data Documentation

4.14.2.1 loadedTextures

```
std::unordered_map< std::string, SDL_Texture * > TextureLoader::loadedTextures  [static],
[private]
```

Static map storing all loaded textures.

4.14.2.2 TEXTURE_PATH

```
const std::string TextureLoader::TEXTURE_PATH = "assets/"  [inline], [static], [private]
```

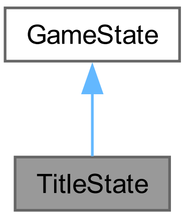
The documentation for this class was generated from the following files:

- src/include/util/TextureLoader.h
- src/util/TextureLoader.cpp

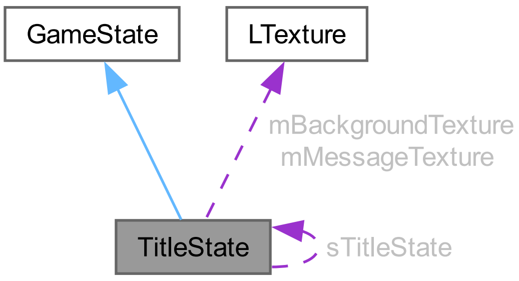
4.15 TitleState Class Reference

```
#include <TitleState.h>
```

Inheritance diagram for TitleState:



Collaboration diagram for TitleState:



Public Member Functions

- bool enter () override
- bool exit () override
- void handleEvent (SDL_Event &e) override
- void update () override
- void render () override

Public Member Functions inherited from GameState

- virtual ∼GameState ()=default

Static Public Member Functions

- static TitleState ∗ get ()

Private Member Functions

- TitleState ()

Private Attributes

- LTexture mBackgroundTexture
- LTexture mMessageTexture

Static Private Attributes

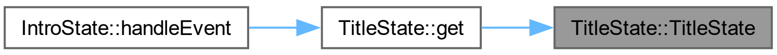
- static TitleState sTitleState

4.15.1 Constructor & Destructor Documentation

4.15.1.1 TitleState()

```
TitleState::TitleState ()  [private]
```

Here is the caller graph for this function:



4.15.2 Member Function Documentation

4.15.2.1 enter()

```
bool TitleState::enter ()  [override], [virtual]
```

Implements GameState.

4.15.2.2 exit()

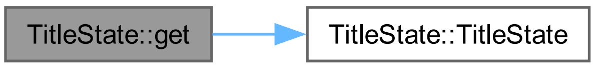
```
bool TitleState::exit ()  [override], [virtual]
```

Implements GameState.

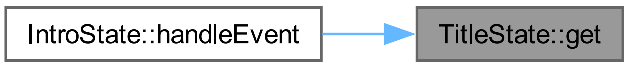
4.15.2.3 get()

```
TitleState * TitleState::get ()  [static]
```

Here is the call graph for this function:



Here is the caller graph for this function:

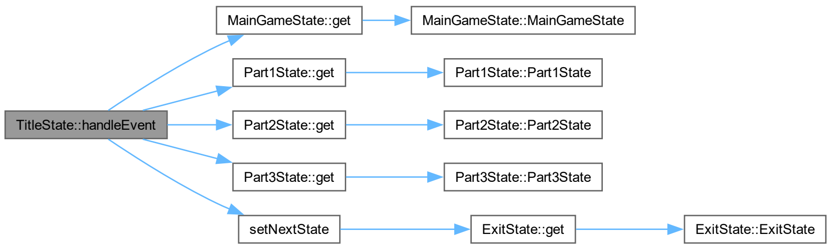


4.15.2.4 handleEvent()

```
void TitleState::handleEvent (
             SDL_Event & e)  [override], [virtual]
```

Implements GameState.

Here is the call graph for this function:



4.15.2.5 render()

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

Implements GameState.

4.15.2.6 update()

```
void TitleState::update ()  [override], [virtual]
```

Implements GameState.

4.15.3 Member Data Documentation

4.15.3.1 mBackgroundTexture

```
LTexture TitleState::mBackgroundTexture  [private]
```

4.15.3.2 mMessageTexture

```
LTexture TitleState::mMessageTexture  [private]
```

4.15.3.3 sTitleState

```
TitleState TitleState::sTitleState  [static], [private]
```

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

- src/include/states/TitleState.h
- src/states/TitleState.cpp

4.16 Vector2D Class Reference

```
#include <Vector2D.h>
```

Public Member Functions

- Vector2D (float setX, float setY)
- Vector2D (const Vector2D &other)
- Vector2D (float angleRad)
- Vector2D ()
- float angle ()
- float magnitude ()
- Vector2D normalize ()
- Vector2D getNegativeReciprocal ()
- float dot (const Vector2D &other)
- float cross (const Vector2D &other)
- float angleBetween (const Vector2D &other)
- Vector2D operator+ (const float amount)
- Vector2D operator- (const float amount)
- Vector2D operator∗ (const float amount)
- Vector2D operator/ (const float amount)
- Vector2D operator+ (const Vector2D &other)
- Vector2D operator- (const Vector2D &other)
- Vector2D operator∗ (const Vector2D &other)
- Vector2D operator/ (const Vector2D &other)
- Vector2D & operator+= (const float amount)
- Vector2D & operator-= (const float amount)
- Vector2D & operator∗= (const float amount)
- Vector2D & operator/= (const float amount)
- Vector2D & operator+= (const Vector2D &other)
- Vector2D & operator-= (const Vector2D &other)
- Vector2D & operator∗= (const Vector2D &other)
- Vector2D & operator/= (const Vector2D &other)

Public Attributes

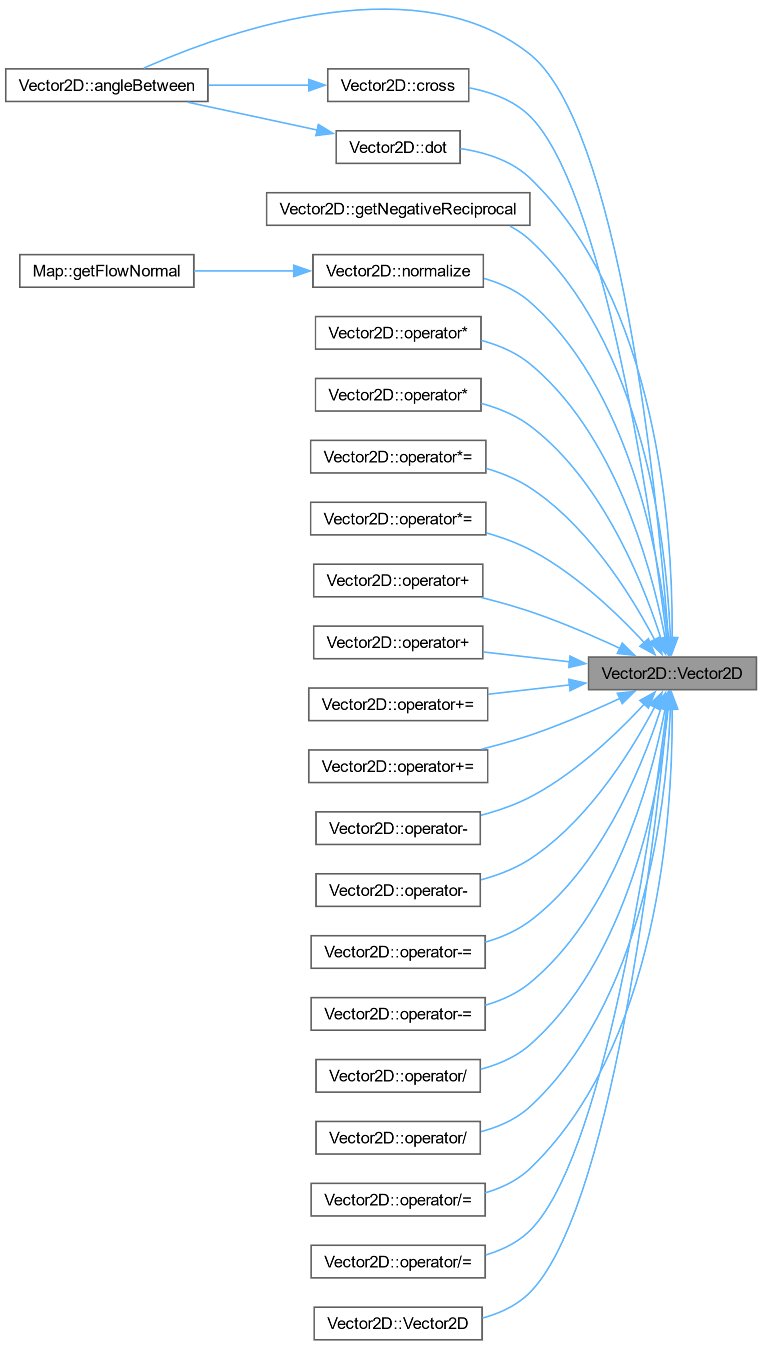
- float x
- float y

4.16.1 Constructor & Destructor Documentation

4.16.1.1 Vector2D() [1/4]

```
Vector2D::Vector2D (
             float setX,
             float setY)  [inline]
```

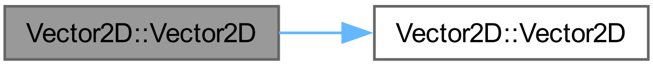
Here is the caller graph for this function:



4.16.1.2 Vector2D() [2/4]

```
Vector2D::Vector2D (
             const Vector2D & other)  [inline]
```

Here is the call graph for this function:



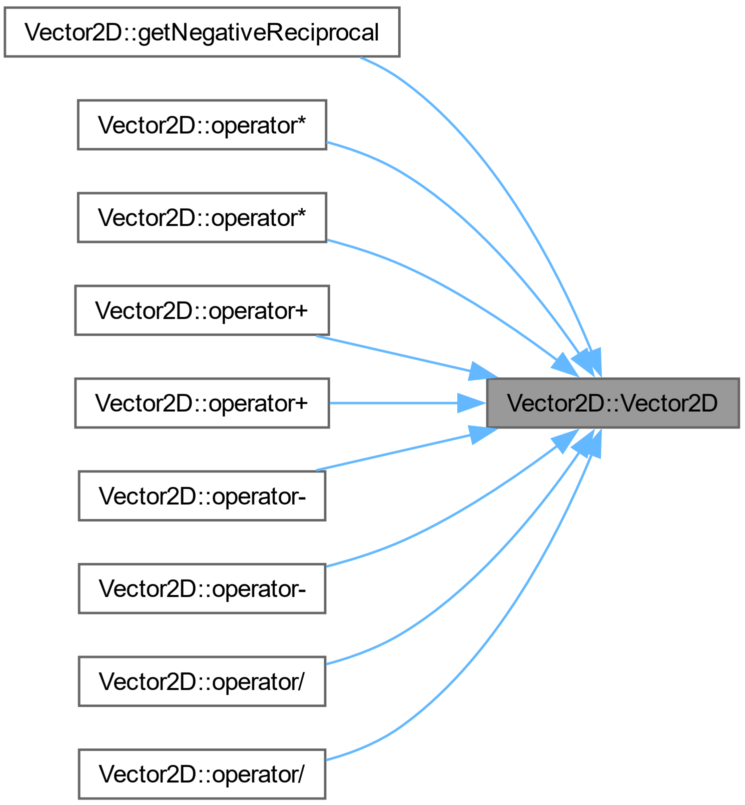
4.16.1.3 Vector2D() [3/4]

```
Vector2D::Vector2D (
             float angleRad)  [inline]
```

4.16.1.4 Vector2D() [4/4]

```
Vector2D::Vector2D ()  [inline]
```

Here is the caller graph for this function:



4.16.2 Member Function Documentation

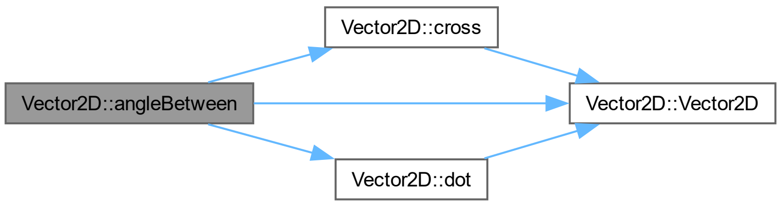
4.16.2.1 angle()

```
float Vector2D::angle ()  [inline]
```

4.16.2.2 angleBetween()

```
float Vector2D::angleBetween (
            const Vector2D & other)  [inline]
```

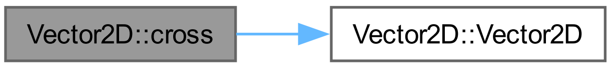
Here is the call graph for this function:



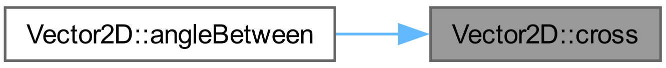
4.16.2.3 cross()

```
float Vector2D::cross (
            const Vector2D & other)  [inline]
```

Here is the call graph for this function:



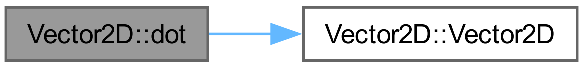
Here is the caller graph for this function:



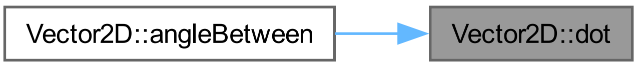
4.16.2.4 dot()

```
float Vector2D::dot (
            const Vector2D & other)  [inline]
```

Here is the call graph for this function:



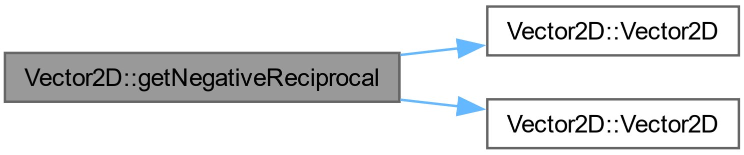
Here is the caller graph for this function:



4.16.2.5 getNegativeReciprocal()

```
Vector2D Vector2D::getNegativeReciprocal ()  [inline]
```

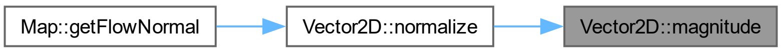
Here is the call graph for this function:



4.16.2.6 magnitude()

```
float Vector2D::magnitude ()  [inline]
```

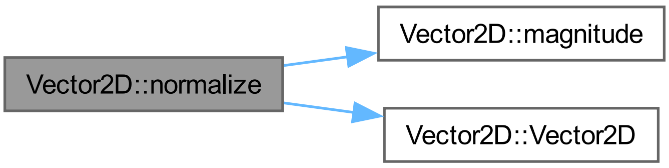
Here is the caller graph for this function:



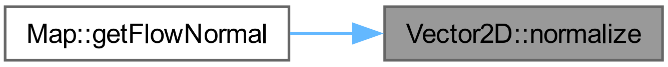
4.16.2.7 normalize()

```
Vector2D Vector2D::normalize ()
```

Here is the call graph for this function:



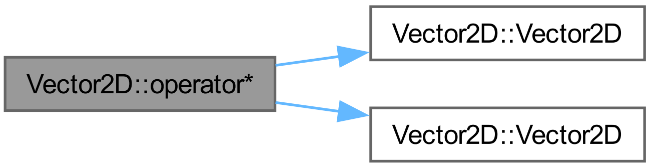
Here is the caller graph for this function:



4.16.2.8 operator∗() [1/2]

```
Vector2D Vector2D::operator* (
            const float amount)  [inline]
```

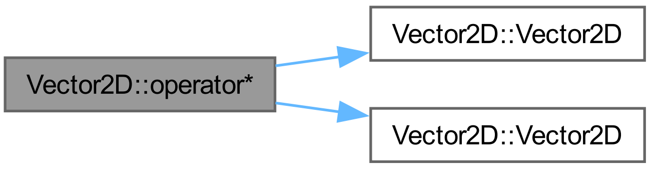
Here is the call graph for this function:



4.16.2.9 operator*() [2/2]

```
Vector2D Vector2D::operator* (
             const Vector2D & other)  [inline]
```

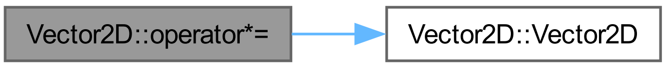
Here is the call graph for this function:



4.16.2.10 operator*=() [1/2]

```
Vector2D & Vector2D::operator*= (
             const float amount)  [inline]
```

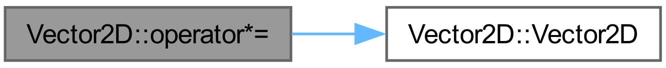
Here is the call graph for this function:



4.16.2.11 operator*=() [2/2]

```
Vector2D & Vector2D::operator*= (
             const Vector2D & other)  [inline]
```

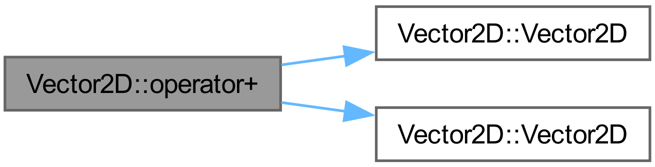
Here is the call graph for this function:



4.16.2.12 operator+() [1/2]

```
Vector2D Vector2D::operator+ (
             const float amount)  [inline]
```

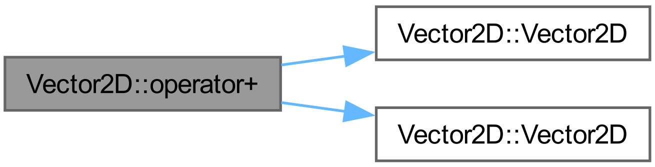
Here is the call graph for this function:



4.16.2.13 operator+() [2/2]

```
Vector2D Vector2D::operator+ (
             const Vector2D & other)  [inline]
```

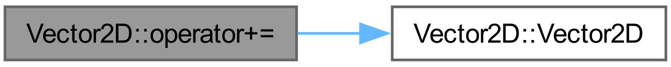
Here is the call graph for this function:



4.16.2.14 operator+=() [1/2]

```
Vector2D & Vector2D::operator+= (
             const float amount)  [inline]
```

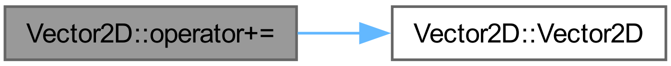
Here is the call graph for this function:



4.16.2.15 operator+=() [2/2]

```
Vector2D & Vector2D::operator+= (
            const Vector2D & other)  [inline]
```

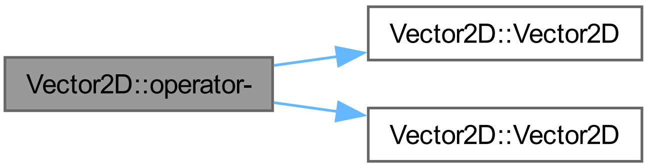
Here is the call graph for this function:



4.16.2.16 operator-() [1/2]

```
Vector2D Vector2D::operator- (
            const float amount)  [inline]
```

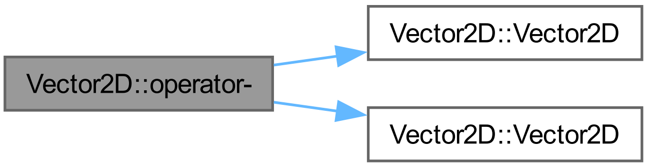
Here is the call graph for this function:



4.16.2.17 operator-() [2/2]

```
Vector2D Vector2D::operator- (
            const Vector2D & other)  [inline]
```

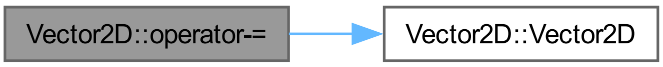
Here is the call graph for this function:



4.16.2.18 operator-=() [1/2]

```
Vector2D & Vector2D::operator-= (
             const float amount)   [inline]
```

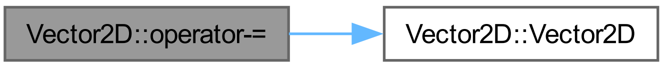
Here is the call graph for this function:



4.16.2.19 operator-=() [2/2]

```
Vector2D & Vector2D::operator-= (
             const Vector2D & other)   [inline]
```

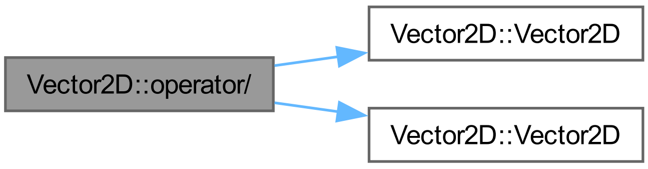
Here is the call graph for this function:



4.16.2.20 operator/() [1/2]

```
Vector2D Vector2D::operator/ (
             const float amount)   [inline]
```

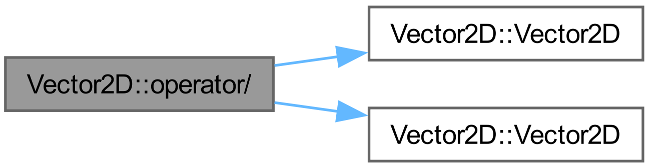
Here is the call graph for this function:



4.16.2.21 operator/() [2/2]

```
Vector2D Vector2D::operator/ (
            const Vector2D & other)  [inline]
```

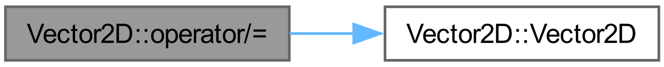
Here is the call graph for this function:



4.16.2.22 operator/=() [1/2]

```
Vector2D & Vector2D::operator/= (
            const float amount)  [inline]
```

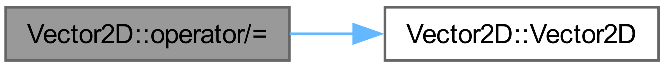
Here is the call graph for this function:



4.16.2.23 operator/=() [2/2]

```
Vector2D & Vector2D::operator/= (
            const Vector2D & other)  [inline]
```

Here is the call graph for this function:



4.16.3 Member Data Documentation

4.16.3.1 x

```
float Vector2D::x
```

4.16.3.2 y

```
float Vector2D::y
```

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

- src/include/util/Vector2D.h
- src/util/Vector2D.cpp

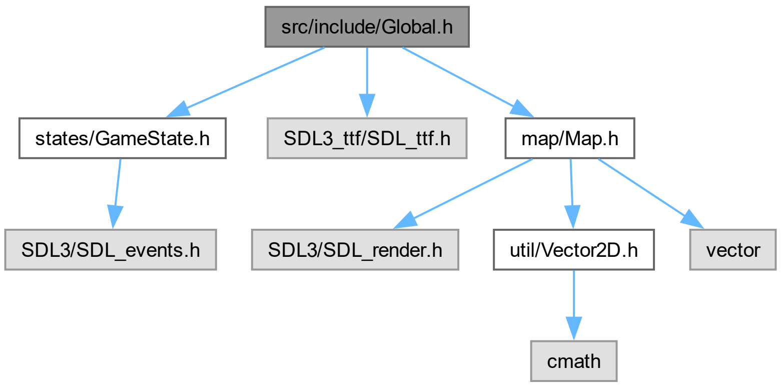
Chapter 5

File Documentation

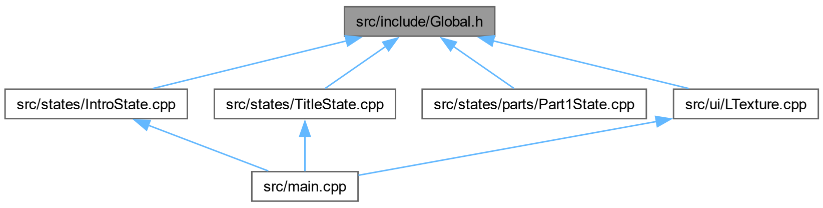
5.1 src/include/Global.h File Reference

```
#include <states/GameState.h>
#include <SDL3_ttf/SDL_ttf.h>
#include <map/Map.h>
```

Include dependency graph for Global.h:



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


src/include/Global.h
src/states/IntroState.cpp
src/states/TitleState.cpp
src/states/parts/Part1State.cpp
src/ui/LTexture.cpp
src/main.cpp

Classes

- class Global

Functions

- bool init ()
- bool loadMedia ()
- void close ()
- void setNextState (GameState ∗nextState)
- void changeState ()

Variables

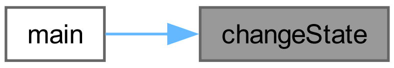
- SDL_Renderer ∗ gRenderer
- TTF_Font ∗ gFont

5.1.1 Function Documentation

5.1.1.1 changeState()

```
void changeState ()
```

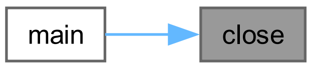
Here is the caller graph for this function:



5.1.1.2 close()

```
void close ()
```

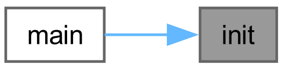
Here is the caller graph for this function:



5.1.1.3 init()

```
bool init ()
```

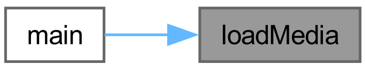
Here is the caller graph for this function:



5.1.1.4 loadMedia()

```
bool loadMedia ()
```

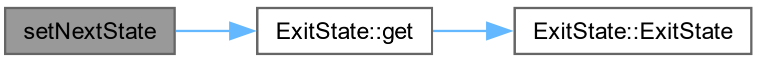
Here is the caller graph for this function:



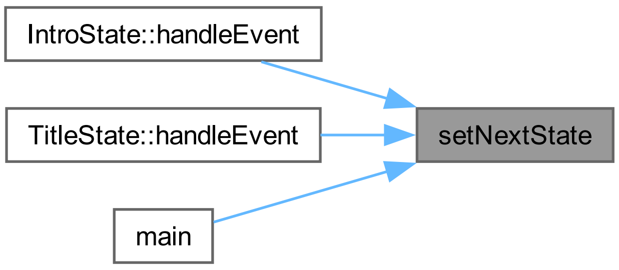
5.1.1.5 setNextState()

```
void setNextState (
            GameState * nextState)
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2 Variable Documentation

5.1.2.1 gFont

```
TTF_Font* gFont  [extern]
```

5.1.2.2 gRenderer

```
SDL_Renderer* gRenderer  [extern]
```

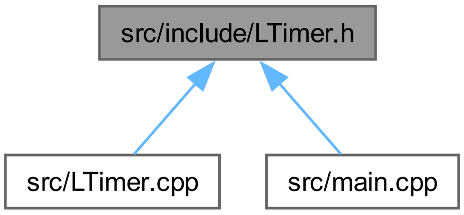
5.2 Global.h

Go to the documentation of this file.

```
#ifndef GLOBAL_H
#define GLOBAL_H

#include <states/GameState.h>
#include <SDL3_ttf/SDL_ttf.h>
#include <map/Map.h>

extern SDL_Renderer* gRenderer;
extern TTF_Font* gFont;

class Global {
public:
    //Screen dimension constants
    static const int kScreenWidth{ Map::PIXELS_PER_CELL * 15 };
    static const int kScreenHeight{ Map::PIXELS_PER_CELL * 11 };
};

/* Function Prototypes */
//Starts up SDL and creates window
bool init();

//Loads media
bool loadMedia();

//Frees media and shuts down SDL
void close();

//State managers
void setNextState(GameState* nextState);
void changeState();
#endif
```

5.3 src/include/LTimer.h File Reference

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



Classes

- class LTimer

5.4 LTimer.h

Go to the documentation of this file.

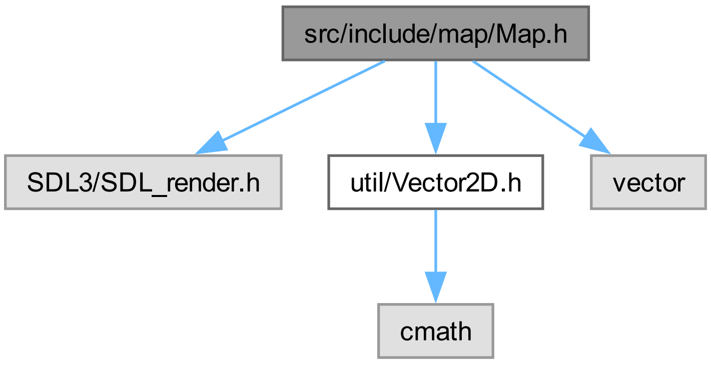
```
class LTimer {
public:
    //Initializes variables
    LTimer();

    //The various clock actions
    void start();
    void stop();
    void pause();
    void unpause();

    //Gets the timer’s time
    Uint64 getTicksNS();

    //Checks the status of the timer
    bool isStarted();
    bool isPaused();

private:
    //The clock time when the timer started
    Uint64 mStartTicks;

    //The ticks stored when the timer was paused
    Uint64 mPausedTicks;

    //The timer status
    bool mPaused;
    bool mStarted;
};
```

5.5 src/include/map/Map.h File Reference

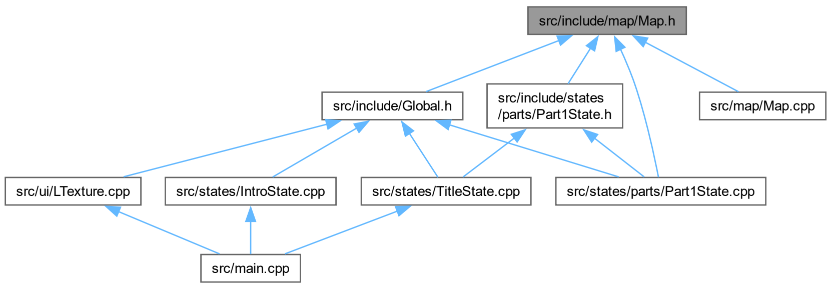
Responsible for generating the tower defense map.

```
#include <SDL3/SDL_render.h>
#include <util/Vector2D.h>
#include <vector>
```

Include dependency graph for Map.h:



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


src/include/map/Map.h
src/include/Global.h
src/include/states /parts/Part1State.h
src/map/Map.cpp
src/ui/LTexture.cpp
src/states/IntroState.cpp
src/states/TitleState.cpp
src/states/parts/Part1State.cpp
src/main.cpp

Classes

- class Map
- struct Map::Cell

    Represents a single cell in the map grid.

5.5.1 Detailed Description

Responsible for generating the tower defense map.

Author

Nathan Grenier

Date

2025-02-15

A class that manages a grid-based map for a tower defense game. The map consists of cells that can be walls, spawners, or targets. It also implements a flow field pathfinding system to guide enemies from spawners to targets.

5.6 Map.h

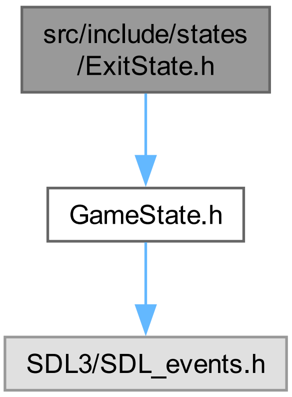
Go to the documentation of this file.

```

#pragma once

#include <SDL3/SDL_render.h>
#include <util/Vector2D.h>
#include <vector>

class Map {
private:
    static const unsigned char flowDistanceMax = 255;

    struct Cell {
        int x = 0;
        int y = 0;
        bool isWall = false;
        bool isSpawner = false;
        bool isTarget = false;
        int flowDirectionX = 0;
        int flowDirectionY = 0;
        unsigned char flowDistance = flowDistanceMax;
    };

public:
    static constexpr int PIXELS_PER_CELL = 48;

public:
    Map(SDL_Renderer* renderer, int cellCountX, int cellCountY);
    void draw(SDL_Renderer* renderer);

    bool isCellWall(int x, int y);
    void setCellWall(int x, int y, bool setWall);
    bool isTarget(int x, int y);
    void setTarget(int x, int y);
    bool isSpawner(int x, int y);
    void setSpawner(int x, int y);
    bool isValidPath();
    Vector2D getTargetPos();
    Vector2D getFlowNormal(int x, int y);

private:
    bool isInbounds(int x, int y);
    void drawCell(SDL_Renderer* renderer, const Cell& cell);
    void calculateFlowField();
    void calculateDistances();
    void calculateFlowDirections();

    std::vector<Cell> cells;
    const int cellCountX, cellCountY;

    SDL_Texture* textureCellWall;
    SDL_Texture* textureCellTarget;
    SDL_Texture* textureCellSpawner;
    SDL_Texture* textureCellEmpty;
    SDL_Texture* textureCellArrowUp;
    SDL_Texture* textureCellArrowRight;
    SDL_Texture* textureCellArrowDown;
    SDL_Texture* textureCellArrowLeft;
};
```

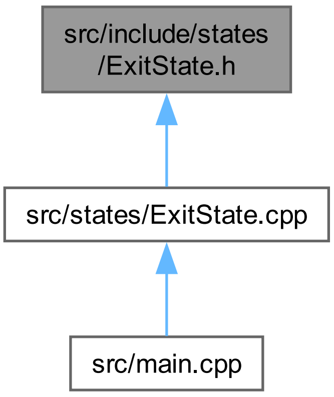
5.7 src/include/states/ExitState.h File Reference

```
#include "GameState.h"
```

Include dependency graph for ExitState.h:



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



Classes

- class ExitState

5.8 ExitState.h

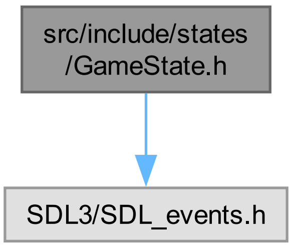
Go to the documentation of this file.

```
#include "GameState.h"

#ifndef EXITSTATE_H
#define EXITSTATE_H
class ExitState : public GameState {
public:
    //Static accessor
    static ExitState* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static ExitState sExitState;

    //Private constructor
    ExitState();
};
#endif
```

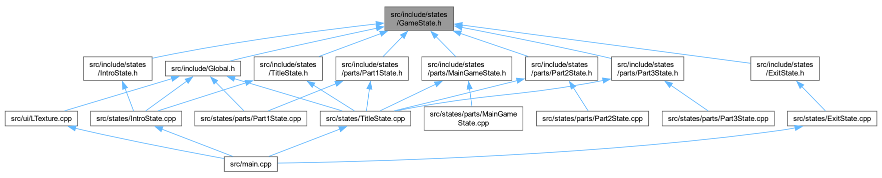
5.9 src/include/states/GameState.h File Reference

`#include <SDL3/SDL_events.h>`

Include dependency graph for GameState.h:



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



Classes

- class GameState

5.10 GameState.h

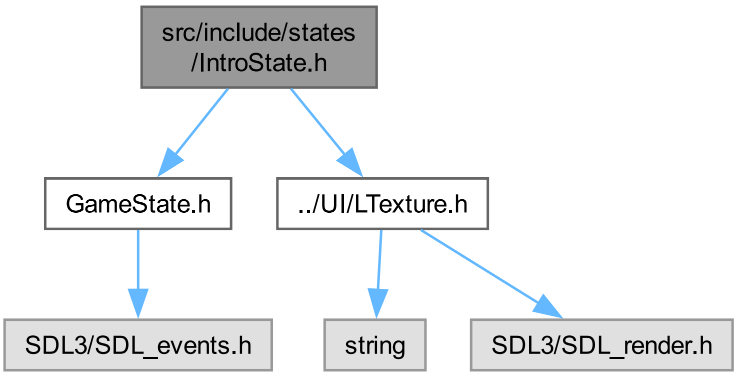
Go to the documentation of this file.

```
#pragma once
#include <SDL3/SDL_events.h>
class GameState {
public:
    //State transitions
    virtual bool enter() = 0;
    virtual bool exit() = 0;

    //Main loop functions
    virtual void handleEvent(SDL_Event& e) = 0;
    virtual void update() = 0;
    virtual void render() = 0;

    //Make sure to call child destructor
    virtual ~GameState() = default;
};
```

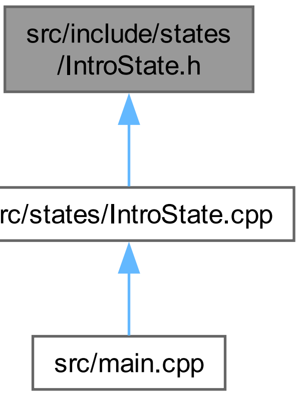
5.11 src/include/states/IntroState.h File Reference

```
#include "GameState.h"
#include "../UI/LTexture.h"
```

Include dependency graph for IntroState.h:



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



Classes

- class IntroState

Macros

- #define INTROSTATE_H

5.11.1 Macro Definition Documentation

5.11.1.1 INTROSTATE_H

```
#define INTROSTATE_H
```

5.12 IntroState.h

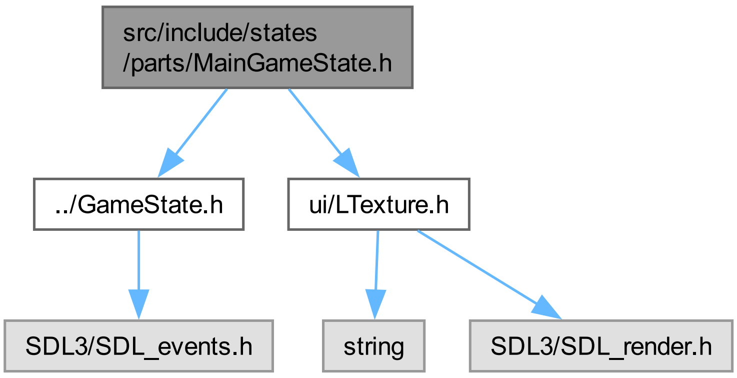
Go to the documentation of this file.

```
#include "GameState.h"
#include "../UI/LTexture.h"

#ifndef INTROSTATE_H
#define INTROSTATE_H
class IntroState : public GameState {
public:
    //Static accessor
    static IntroState* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static IntroState sIntroState;

    //Private constructor
    IntroState();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;
};
#endif
```

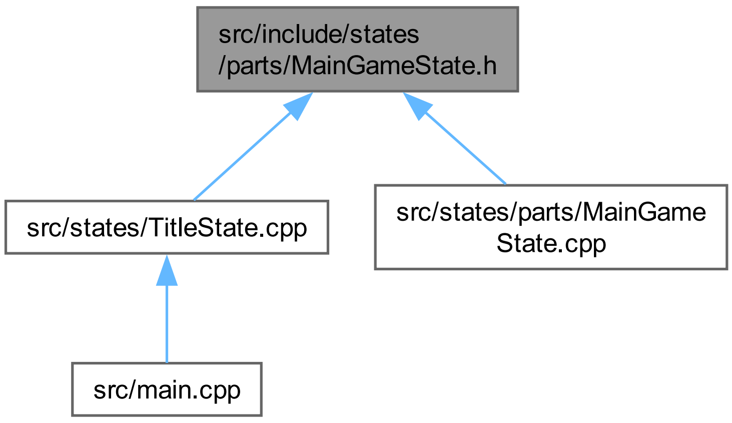
5.13 src/include/states/parts/MainGameState.h File Reference

```
#include "../GameState.h"
#include <ui/LTexture.h>
```

Include dependency graph for MainGameState.h:



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



Classes

- class MainGameState

5.14 MainGameState.h

Go to the documentation of this file.

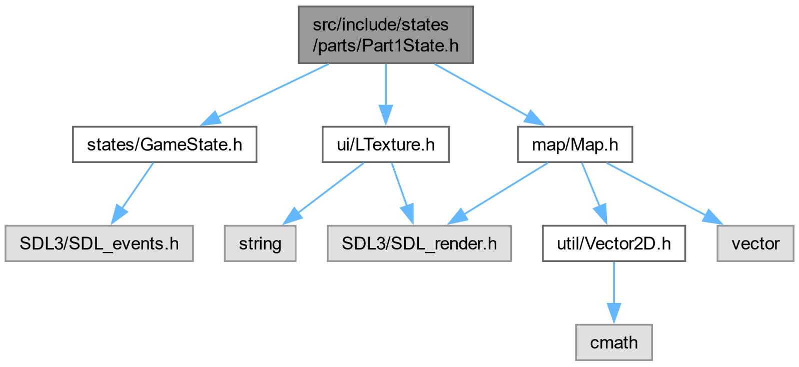
```
#ifndef MAINGAMESTATE_H
#define MAINGAMESTATE_H

#include "../GameState.h"
#include <ui/LTexture.h>

class MainGameState : public GameState {
public:
    //Static accessor
    static MainGameState* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static MainGameState sMainGameState;

    //Private constructor
    MainGameState();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;
};
#endif
```

5.15 src/include/states/parts/Part1State.h File Reference

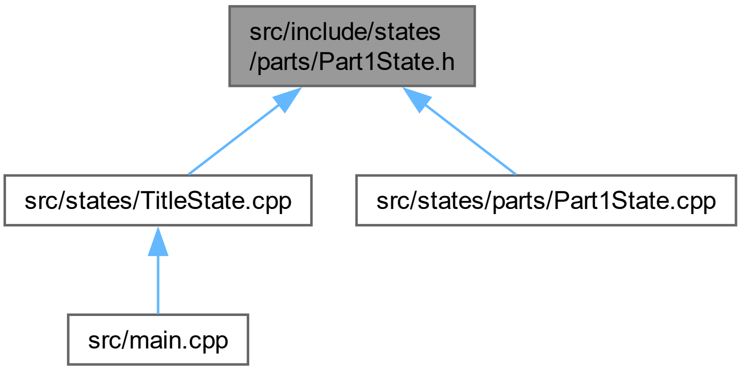
The header file of part 1's (Map) driver.

```
#include <states/GameState.h>
#include <ui/LTexture.h>
#include <map/Map.h>
```

Include dependency graph for Part1State.h:



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



Classes

- class Part1State

5.15.1 Detailed Description

The header file of part 1's (Map) driver.

Author

Nathan Grenier

Date

2025-02-15

5.16 Part1State.h

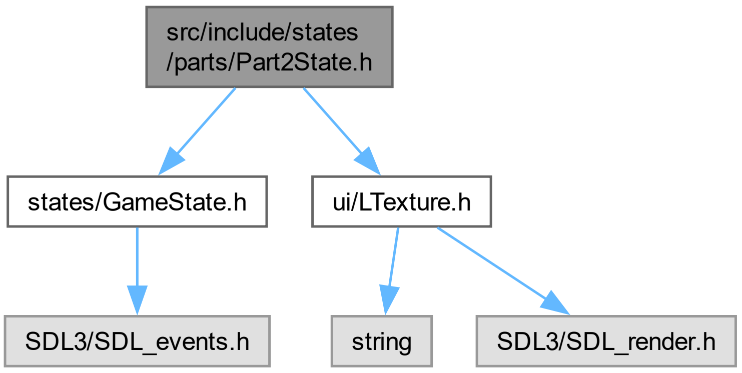
Go to the documentation of this file.

```
#pragma once

#include <states/GameState.h>
#include <ui/LTexture.h>
#include <map/Map.h>

class Part1State : public GameState {
public:
    //Static accessor
    static Part1State* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static Part1State sPart1State;

    //Private constructor
    Part1State();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;

    int mouseDownStatus = 0;
    int keyDownStatus = 0;

    Map* map = nullptr;
};
```

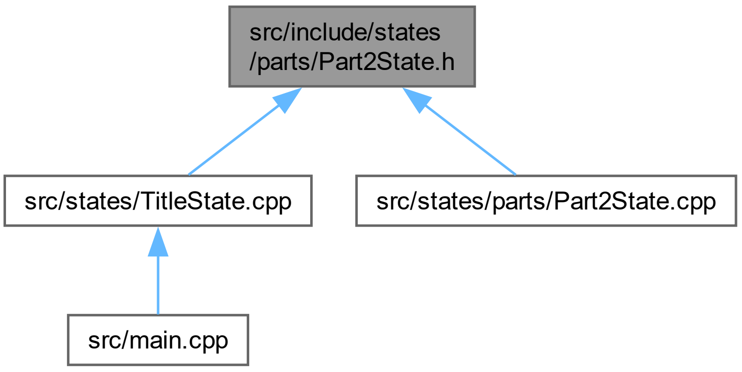
5.17 src/include/states/parts/Part2State.h File Reference

```
#include <states/GameState.h>
#include <ui/LTexture.h>
```

Include dependency graph for Part2State.h:



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


src/include/states /parts/Part2State.h
src/states/TitleState.cpp
src/states/parts/Part2State.cpp
src/main.cpp

Classes

- class Part2State

5.18 Part2State.h

Go to the documentation of this file.

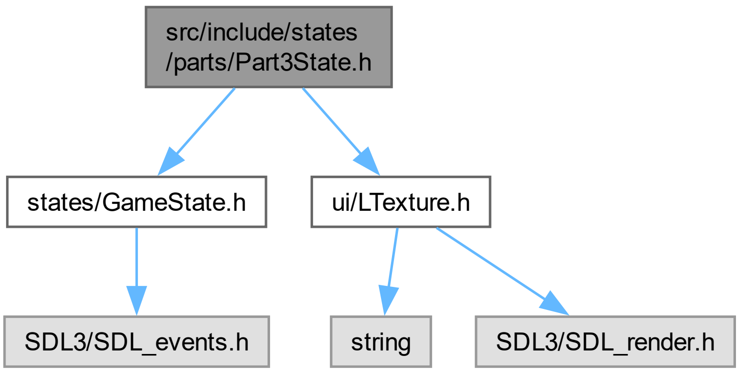
```
#ifndef PART2STATE_H
#define PART2STATE_H

#include <states/GameState.h>
#include <ui/LTexture.h>
```

```

class Part2State : public GameState {
public:
    //Static accessor
    static Part2State* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static Part2State sPart2State;

    //Private constructor
    Part2State();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;
};
#endif
```

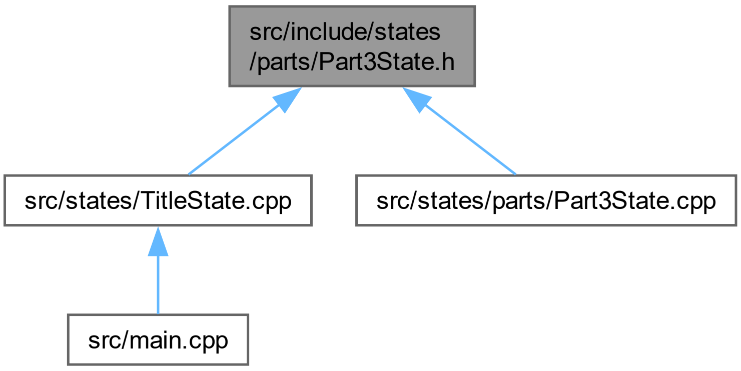
5.19 src/include/states/parts/Part3State.h File Reference

```
#include <states/GameState.h>
#include <ui/LTexture.h>
```

Include dependency graph for Part3State.h:



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



Classes

- class Part3State

5.20 Part3State.h

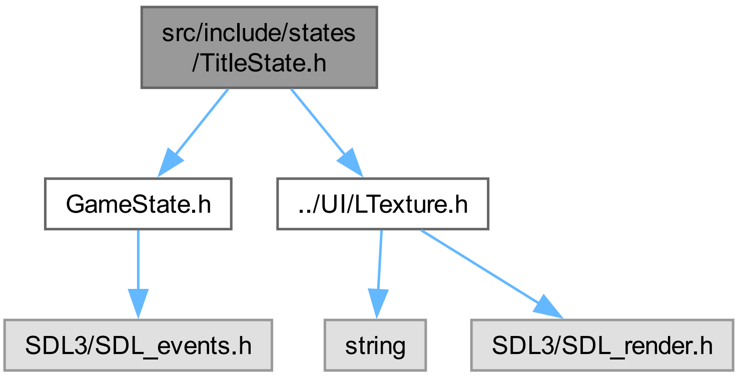
Go to the documentation of this file.

```
#ifndef PART3STATE_H
#define PART3STATE_H

#include <states/GameState.h>
#include <ui/LTexture.h>

class Part3State : public GameState {
public:
    //Static accessor
    static Part3State* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static Part3State sPart3State;

    //Private constructor
    Part3State();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;
};
#endif
```

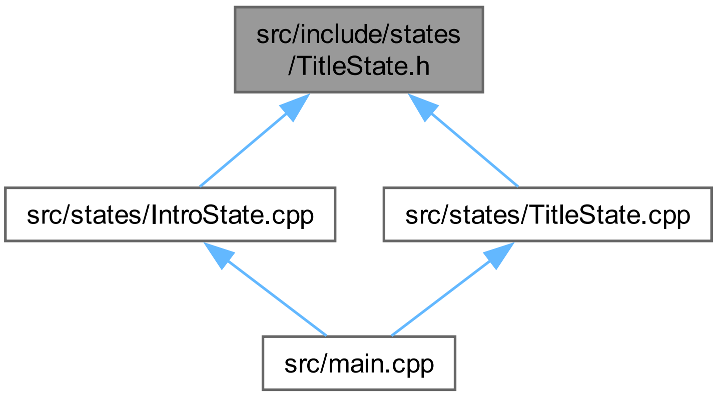
5.21 src/include/states/TitleState.h File Reference

```
#include "GameState.h"
#include "../UI/LTexture.h"
```

Include dependency graph for TitleState.h:



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



Classes

- class TitleState

5.22 TitleState.h

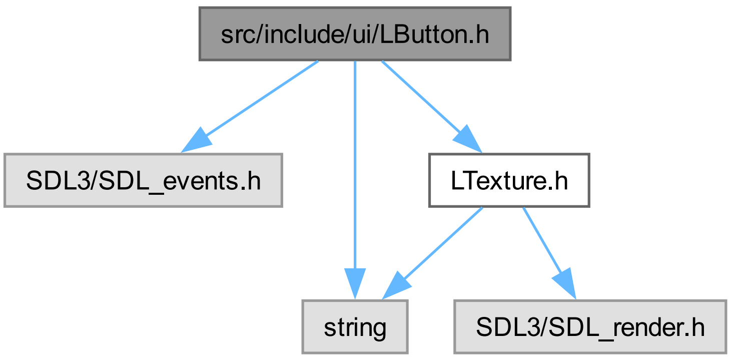
Go to the documentation of this file.

```
#include "GameState.h"
#include "../UI/LTexture.h"

#ifndef TITLESTATE_H
#define TITLESTATE_H
class TitleState : public GameState {
public:
    //Static accessor
    static TitleState* get();

    //Transitions
    bool enter() override;
    bool exit() override;

    //Main loop functions
    void handleEvent(SDL_Event& e) override;
    void update() override;
    void render() override;

private:
    //Static instance
    static TitleState sTitleState;

    //Private constructor
    TitleState();

    //Intro background
    LTexture mBackgroundTexture;

    //Intro message
    LTexture mMessageTexture;
};
#endif
```

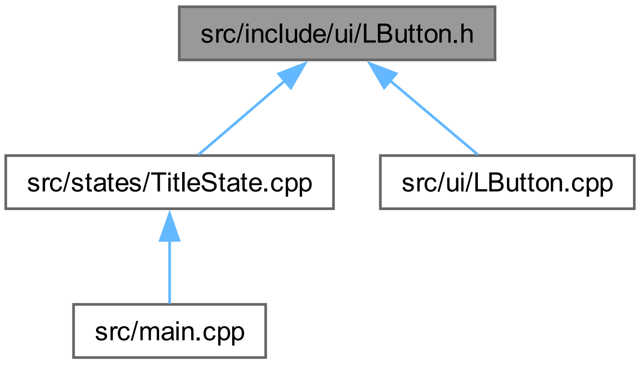
5.23 src/include/ui/LButton.h File Reference

```
#include <SDL3/SDL_events.h>
#include <string>
#include "LTexture.h"
```

Include dependency graph for LButton.h:



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



Classes

- class LButton

5.24 LButton.h

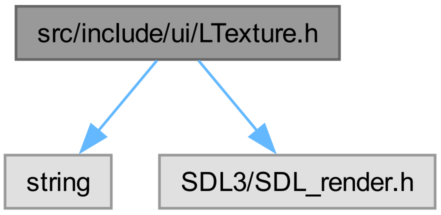
Go to the documentation of this file.

```
#pragma once
#include <SDL3/SDL_events.h>
#include <string>
#include "LTexture.h"
class LButton {
public:
    //Button dimensions
    static constexpr int kButtonWidth = 300;
    static constexpr int kButtonHeight = 50;

    //Initializes internal variables
    LButton();

    //Sets top left position
    void setPosition(float x, float y);

    //Handles mouse event
    void handleEvent(SDL_Event* e);

    bool setText(const std::string& text, SDL_Color textColor);

    //Shows button sprite
    void render();

    bool isClicked() const;

private:
    enum eButtonSprite {
        eButtonSpriteMouseOut = 0,
        eButtonSpriteMouseOverMotion = 1,
        eButtonSpriteMouseDown = 2,
        eButtonSpriteMouseUp = 3
    };

    //Top left position
    SDL_FPoint mPosition;

    //Currently used global sprite
    eButtonSprite mCurrentSprite;

    LTexture gButtonSpriteTexture;
};
```

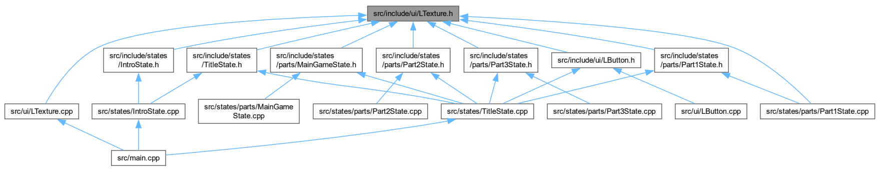
5.25 src/include/ui/LTexture.h File Reference

```
#include <string>
#include <SDL3/SDL_render.h>
```

Include dependency graph for LTexture.h:



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


src/include/ui/LTexture.h
src/include/states /IntroState.h
src/include/states /TitleState.h
src/include/states /parts/MainGameState.h
src/include/states /parts/Part2State.h
src/include/states /parts/Part3State.h
src/include/ui/LButton.h
src/include/states /parts/Part1State.h
src/ui/LTexture.cpp
src/states/IntroState.cpp
src/states/parts/MainGame State.cpp
src/states/parts/Part2State.cpp
src/states/TitleState.cpp
src/states/parts/Part3State.cpp
src/ui/LButton.cpp
src/states/parts/Part1State.cpp
src/main.cpp

Classes

- class LTexture

5.26 LTexture.h

Go to the documentation of this file.

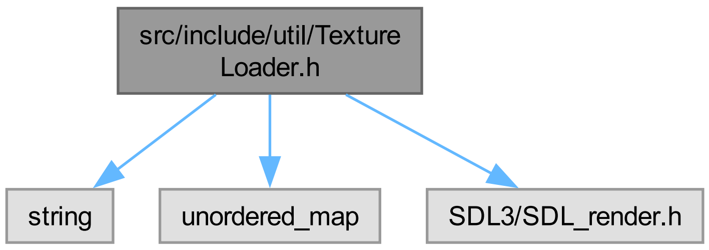
```
#pragma once

#include <string>
#include <SDL3/SDL_render.h>
class LTexture {
public:
    //Symbolic constant
    static constexpr float kOriginalSize = -1.f;

    //Initializes texture variables
    LTexture();

    //Cleans up texture variables
    ~LTexture();

    //Loads texture from disk
    bool loadFromFile(std::string path);

    bool loadFromRenderedText(std::string textureText, SDL_Color textColor);

    //Cleans up texture
    void destroy();

    //Sets color modulation
    void setColor(Uint8 r, Uint8 g, Uint8 b);

    //Sets opacity
    void setAlpha(Uint8 alpha);

    //Sets blend mode
    void setBlending(SDL_BlendMode blendMode);

    //Draws texture
    void render(float x, float y, SDL_FRect* clip = nullptr, float width = kOriginalSize, float height
 = kOriginalSize, double degrees = 0.0, SDL_FPoint* center = nullptr, SDL_FlipMode flipMode =
 SDL_FLIP_NONE);

    //Gets texture dimensions
    int getWidth();
    int getHeight();

private:
    //Contains texture data
    SDL_Texture* mTexture;

    //Texture dimensions
    int mWidth;
    int mHeight;
};
```

5.27 src/include/util/TextureLoader.h File Reference

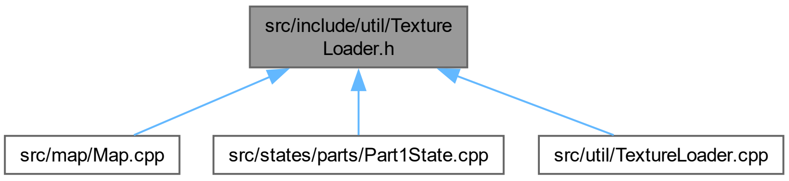
Load and store textures.

```
#include <string>
#include <unordered_map>
#include <SDL3/SDL_render.h>
```

Include dependency graph for TextureLoader.h:



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


src/include/util/Texture Loader.h
src/map/Map.cpp
src/states/parts/Part1State.cpp
src/util/TextureLoader.cpp

Classes

- class TextureLoader

5.27.1 Detailed Description

Load and store textures.

Author

Nathan Grenier

Date

2025-02-15

5.28 TextureLoader.h

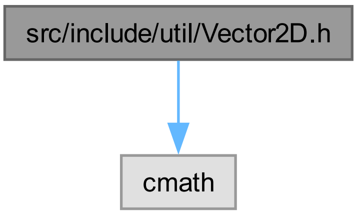
Go to the documentation of this file.

```
#pragma once

#include <string>
#include <unordered_map>
#include <SDL3/SDL_render.h>

class TextureLoader {
public:
    static SDL_Texture* loadTexture(SDL_Renderer* renderer, std::string filename);
    static void deallocateTextures();

private:
    static std::unordered_map<std::string, SDL_Texture*> loadedTextures;
    static inline const std::string TEXTURE_PATH = "assets/";
};
```

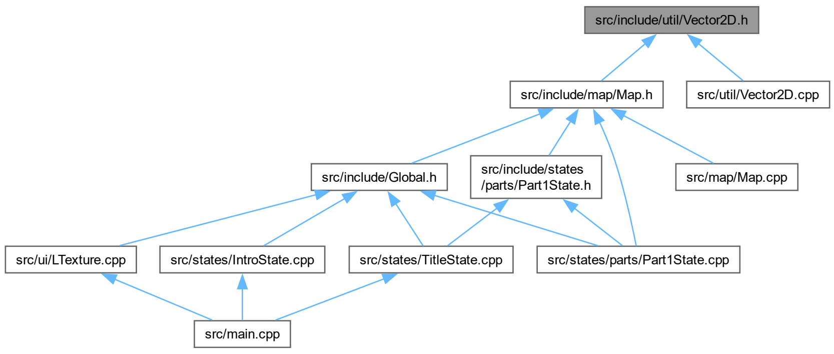
5.29 src/include/util/Vector2D.h File Reference

Representation of a 2D vector.

```
#include <cmath>
```
Include dependency graph for Vector2D.h:



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


src/include/util/Vector2D.h
src/include/map/Map.h
src/util/Vector2D.cpp
src/include/Global.h
src/include/states /parts/Part1State.h
src/map/Map.cpp
src/ui/LTexture.cpp
src/states/IntroState.cpp
src/states/TitleState.cpp
src/states/parts/Part1State.cpp
src/main.cpp

Classes

- class Vector2D

5.29.1 Detailed Description

Representation of a 2D vector.

5.30 Vector2D.h

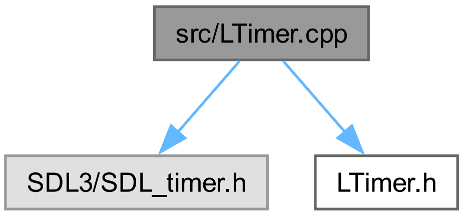
Go to the documentation of this file.

```

#pragma once

#include <cmath>

class Vector2D {
public:
    Vector2D(float setX, float setY) : x(setX), y(setY) {}
    Vector2D(const Vector2D& other) : x(other.x), y(other.y) {}
    Vector2D(float angleRad) : x(cos(angleRad)), y(sin(angleRad)) {}
    Vector2D() : x(0.0f), y(0.0f) {}

    float angle() { return atan2(y, x); }

    float magnitude() { return sqrt(x * x + y * y); }
    Vector2D normalize();
    Vector2D getNegativeReciprocal() { return Vector2D(-y, x); }

    float dot(const Vector2D& other) { return x * other.x + y * other.y; }
    float cross(const Vector2D& other) { return x * other.y - y * other.x; }
    float angleBetween(const Vector2D& other) { return atan2(cross(other), dot(other)); }

    Vector2D operator+(const float amount) { return Vector2D(x + amount, y + amount); }
    Vector2D operator-(const float amount) { return Vector2D(x - amount, y - amount); }
    Vector2D operator*(const float amount) { return Vector2D(x * amount, y * amount); }
    Vector2D operator/(const float amount) { return Vector2D(x / amount, y / amount); }

    Vector2D operator+(const Vector2D& other) { return Vector2D(x + other.x, y + other.y); }
    Vector2D operator-(const Vector2D& other) { return Vector2D(x - other.x, y - other.y); }
    Vector2D operator*(const Vector2D& other) { return Vector2D(x * other.x, y * other.y); }
    Vector2D operator/(const Vector2D& other) { return Vector2D(x / other.x, y / other.y); }

    Vector2D& operator+=(const float amount) { x += amount; y += amount; return *this; }
    Vector2D& operator-=(const float amount) { x -= amount; y -= amount; return *this; }
    Vector2D& operator*=(const float amount) { x *= amount; y *= amount; return *this; }
    Vector2D& operator/=(const float amount) { x /= amount; y /= amount; return *this; }

    Vector2D& operator+=(const Vector2D& other) { x += other.x; y += other.y; return *this; }
    Vector2D& operator-=(const Vector2D& other) { x -= other.x; y -= other.y; return *this; }
    Vector2D& operator*=(const Vector2D& other) { x *= other.x; y *= other.y; return *this; }
    Vector2D& operator/=(const Vector2D& other) { x /= other.x; y /= other.y; return *this; }

    float x, y;
};
```

5.31 src/LTimer.cpp File Reference

```
#include <SDL3/SDL_timer.h>
#include <LTimer.h>
```

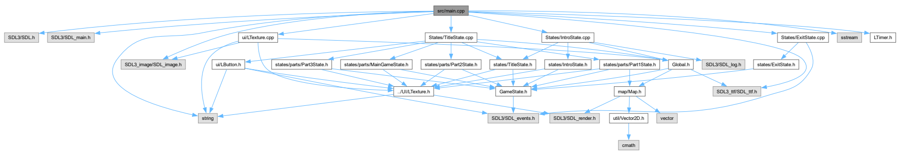
Include dependency graph for LTimer.cpp:



5.32 src/main.cpp File Reference

```
#include <SDL3/SDL.h>
#include <SDL3/SDL_main.h>
#include <SDL3_image/SDL_image.h>
#include <SDL3_ttf/SDL_ttf.h>
#include <string>
#include <sstream>
#include "States/TitleState.cpp"
#include "States/IntroState.cpp"
#include "States/ExitState.cpp"
#include "ui/LTexture.cpp"
#include <LTimer.h>
```

Include dependency graph for main.cpp:



Functions

- void setNextState (GameState ∗newState)
- void changeState ()
- bool init ()
- bool loadMedia ()
- void close ()
- int main (int argc, char ∗args[ ])

Variables

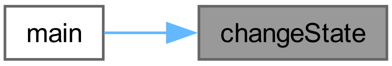
- constexpr int kScreenFps { 60 }
- SDL_Window ∗ gWindow { nullptr }
- SDL_Renderer ∗ gRenderer = nullptr
- TTF_Font ∗ gFont = nullptr
- LTexture gFpsTexture
- GameState ∗ gCurrentState { nullptr }
- GameState ∗ gNextState { nullptr }

5.32.1 Function Documentation

5.32.1.1 changeState()

```
void changeState ()
```

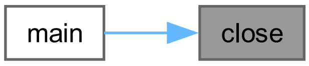
Here is the caller graph for this function:



5.32.1.2 close()

```
void close ()
```

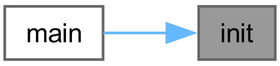
Here is the caller graph for this function:



5.32.1.3 init()

```
bool init ()
```

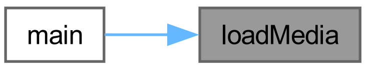
Here is the caller graph for this function:



5.32.1.4 loadMedia()

```
bool loadMedia ()
```

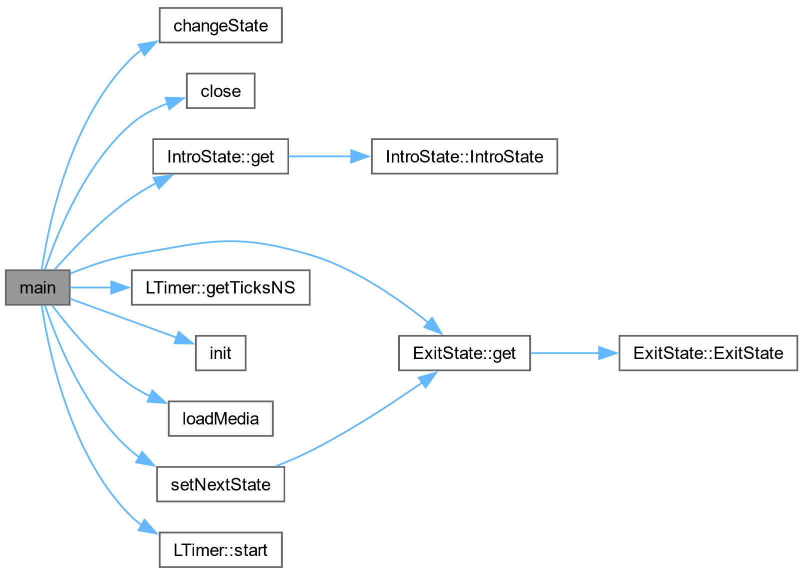
Here is the caller graph for this function:



5.32.1.5 main()

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

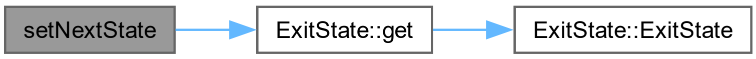
Here is the call graph for this function:



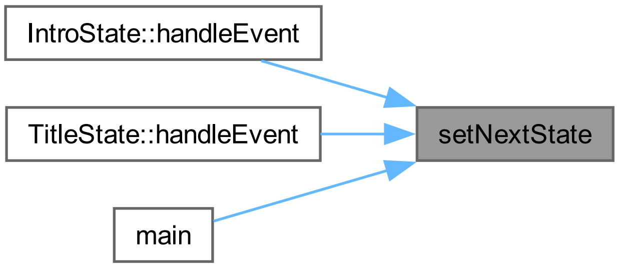
5.32.1.6 setNextState()

```
void setNextState (
            GameState * newState)
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.2 Variable Documentation

5.32.2.1 gCurrentState

```
GameState* gCurrentState { nullptr }
```

5.32.2.2 gFont

```
TTF_Font* gFont = nullptr
```

5.32.2.3 gFpsTexture

```
LTexture gFpsTexture
```

5.32.2.4 gNextState

```
GameState* gNextState { nullptr }
```

5.32.2.5 gRenderer

```
SDL_Renderer* gRenderer = nullptr
```

5.32.2.6 gWindow

```
SDL_Window* gWindow { nullptr }
```

5.32.2.7 kScreenFps

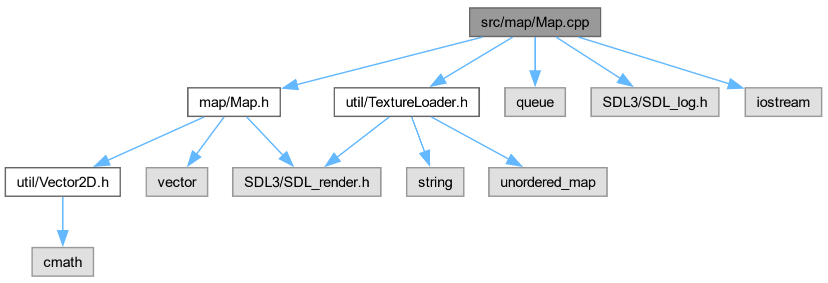
```
int kScreenFps { 60 } [constexpr]
```

5.33 src/map/Map.cpp File Reference

Implementation of the Map class.

```
#include <map/Map.h>
#include <util/TextureLoader.h>
#include <queue>
#include <SDL3/SDL_log.h>
#include <iostream>
```

Include dependency graph for Map.cpp:



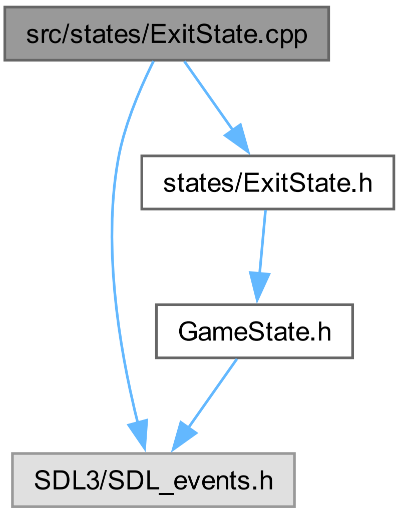
5.33.1 Detailed Description

Implementation of the Map class.

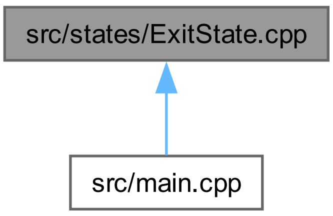
5.34 src/states/ExitState.cpp File Reference

```
#include <SDL3/SDL_events.h>
#include <states/ExitState.h>
```

Include dependency graph for ExitState.cpp:



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



5.35 ExitState.cpp

Go to the documentation of this file.

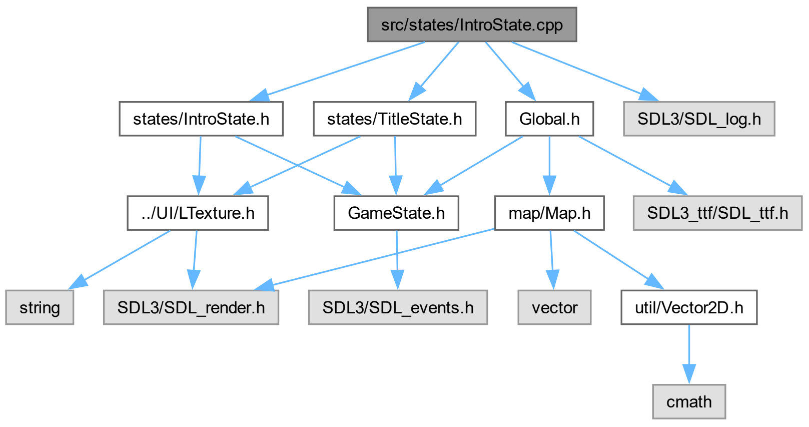
```
#include <SDL3/SDL_events.h>
#include <states/ExitState.h>

ExitState ExitState::sExitState;

```

```
//Hollow exit state
ExitState* ExitState::get() {
    return &sExitState;
}

bool ExitState::enter() {
    return true;
}

bool ExitState::exit() {
    return true;
}

void ExitState::handleEvent(SDL_Event& e) {

}

void ExitState::update() {

}

void ExitState::render() {

}

ExitState::ExitState() {

}
```

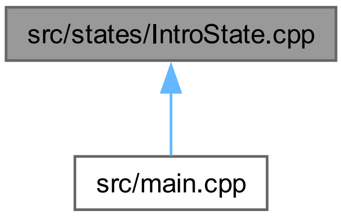
5.36 src/states/IntroState.cpp File Reference

```
#include <states/IntroState.h>
#include <SDL3/SDL_log.h>
#include <states/TitleState.h>
#include <Global.h>
```

Include dependency graph for IntroState.cpp:



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



5.37 IntroState.cpp

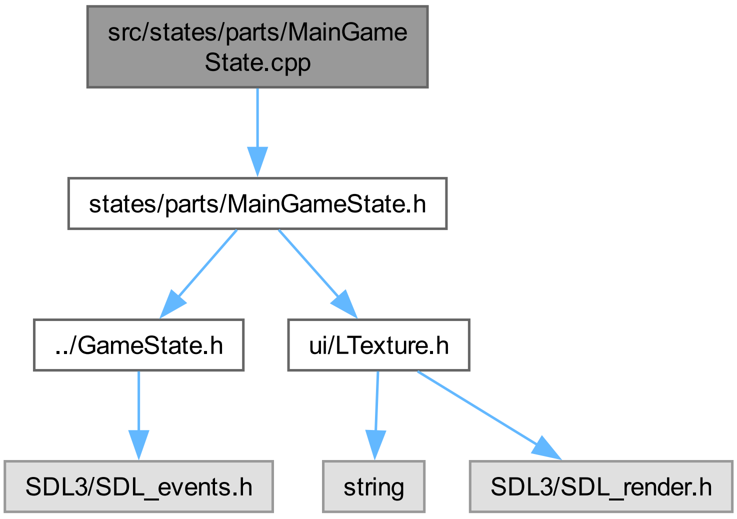
Go to the documentation of this file.

```
#include <states/IntroState.h>
#include <SDL3/SDL_log.h>
#include <states/TitleState.h>
#include <Global.h>

IntroState IntroState::sIntroState;

//InrtoState Implementation
IntroState* IntroState::get() {
    //Get static instance
    return &sIntroState;
}

bool IntroState::enter() {
    //Loading success flag
    bool success = true;

    //Load text
    SDL_Color textColor{ 0x00, 0x00, 0x00, 0xFF };
    if (success &= mMessageTexture.loadFromRenderedText("The NullTerminators Present...", textColor);
!success)
    {
        SDL_Log("Failed to render intro text!\n");
        success = false;
    }

    return success;
}

bool IntroState::exit() {
    //Free background and text
    mBackgroundTexture.destroy();
    mMessageTexture.destroy();

    return true;
}

void IntroState::handleEvent(SDL_Event& e) {
    //If the user pressed enter
    if ((e.type == SDL_EVENT_MOUSE_BUTTON_DOWN))
    {
        //Move onto title state
        setNextState(TitleState::get());
    }
}

void IntroState::update() {

}

void IntroState::render() {
    //Show the background
    mBackgroundTexture.render(0, 0);
00053
00054     //Show the message
00055     mMessageTexture.render((Global::kScreenWidth - mMessageTexture.getWidth()) / 2.f,
      (Global::kScreenHeight - mMessageTexture.getHeight()) / 2.f);
00056 }
00057
00058 IntroState::IntroState() {}
```

5.38 src/states/parts/MainGameState.cpp File Reference

`#include <states/parts/MainGameState.h>`

Include dependency graph for MainGameState.cpp:

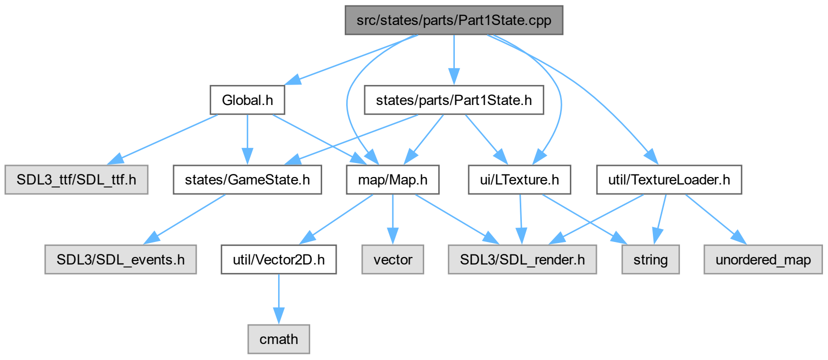


5.39 src/states/parts/Part1State.cpp File Reference

The drive file for part 1 of assignment 1 (Map)

```
#include <states/parts/Part1State.h>
#include <Global.h>
#include <util/TextureLoader.h>
#include <map/Map.h>
#include <ui/LTexture.h>
```

Include dependency graph for Part1State.cpp:



5.39.1 Detailed Description

The drive file for part 1 of assignment 1 (Map)

Author

Nathan Grenier

Date

2025-02-15

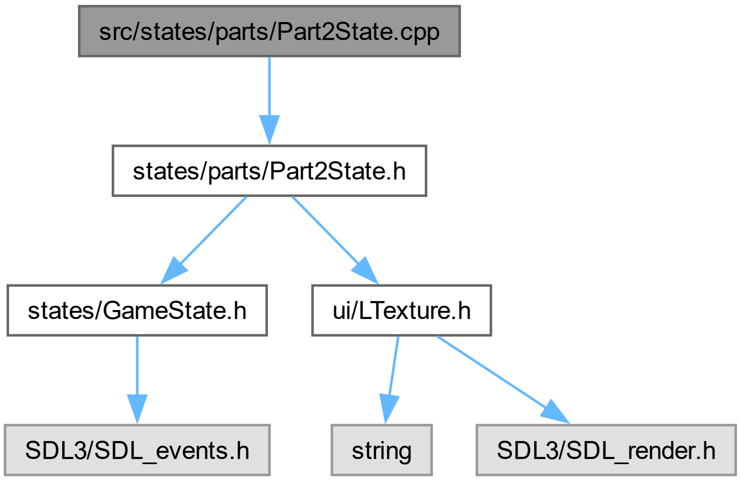
Controls:

- Left Click: Place a wall.
- Right Click: Remove a wall.
- Shift + Left Click: Set the target cell.
- Shift + Right Click: Set the spawner cell.

5.40 src/states/parts/Part2State.cpp File Reference

```
#include <states/parts/Part2State.h>
```

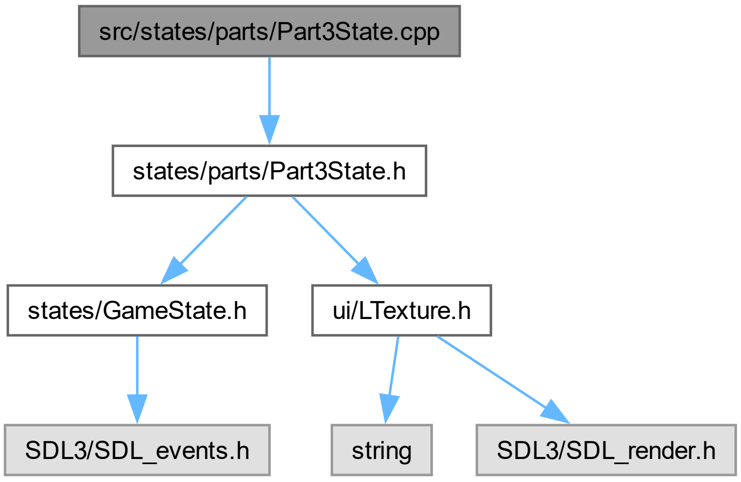
Include dependency graph for Part2State.cpp:



5.41 src/states/parts/Part3State.cpp File Reference

```
#include <states/parts/Part3State.h>
```

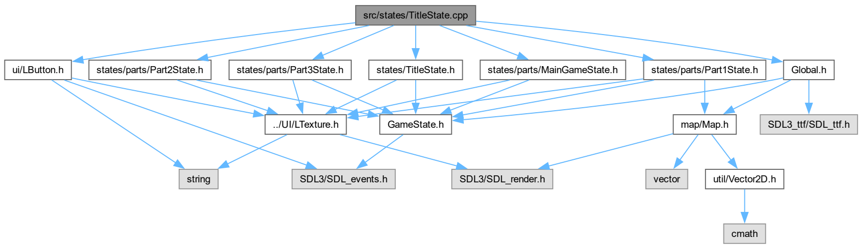
Include dependency graph for Part3State.cpp:



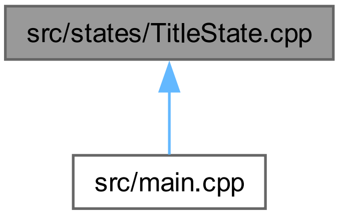
5.42 src/states/TitleState.cpp File Reference

```
#include <states/TitleState.h>
#include <ui/LButton.h>
#include <states/parts/MainGameState.h>
#include <states/parts/Part1State.h>
#include <states/parts/Part2State.h>
#include <states/parts/Part3State.h>
#include <Global.h>
```

Include dependency graph for TitleState.cpp:



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



Variables

- constexpr int kButtonCount = 4
- LButton buttons [kButtonCount]

5.42.1 Variable Documentation

5.42.1.1 buttons

```
LButton buttons[kButtonCount]
```

5.42.1.2 kButtonCount

```
int kButtonCount = 4  [constexpr]
```

5.43 TitleState.cpp

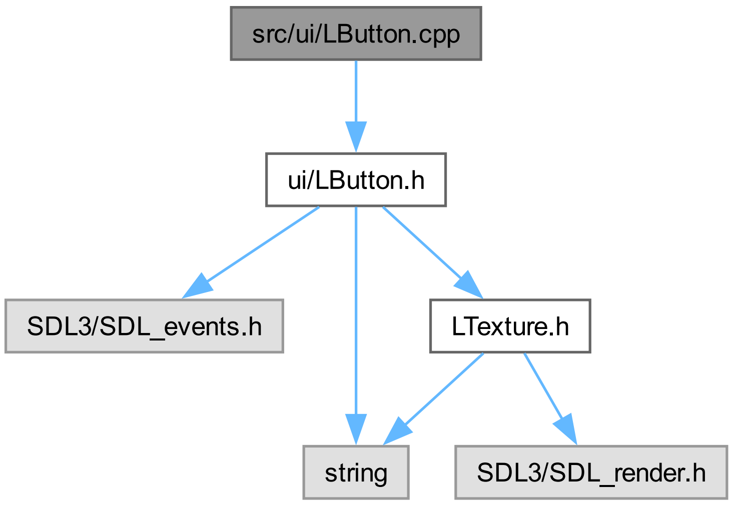
Go to the documentation of this file.

```
#include <states/TitleState.h>
#include <ui/LButton.h>
#include <states/parts/MainGameState.h>
#include <states/parts/Part1State.h>
#include <states/parts/Part2State.h>
#include <states/parts/Part3State.h>
#include <Global.h>

TitleState TitleState::sTitleState;

//Place buttons
constexpr int kButtonCount = 4;
LButton buttons[kButtonCount];

//TitleState Implementation
TitleState* TitleState::get() {
    //Get static instance
    return &sTitleState;
}

bool TitleState::enter() {
    bool success = true;

    SDL_Color textColor{ 0x00, 0x00, 0x00, 0xFF };

    if (!(success &= mMessageTexture.loadFromRenderedText("Tower Defense - The Game", textColor)))
    {
        printf("Failed to render title text!\n");
    }

    // Assign text to buttons
    const char* buttonLabels[kButtonCount] = {
        "Load Main Game",
        "Load Part 1",
        "Load Part 2",
        "Load Part 3"
    };

    for (int i = 0; i < kButtonCount; ++i)
    {
        if (!buttons[i].setText(buttonLabels[i], textColor))
        {
            printf("Failed to set button text: %s\n", buttonLabels[i]);
            success = false;
        }
    }

    return success;
}


bool TitleState::exit() {
    //Free background and text
    mBackgroundTexture.destroy();
    mMessageTexture.destroy();

    return true;
}

void TitleState::handleEvent(SDL_Event& e) {
    //Handle button events
    for (int i = 0; i < kButtonCount; ++i)
    {
        buttons[i].handleEvent(&e);

        // Check if any of the buttons were clicked and set the next state accordingly
        if (e.type == SDL_EVENT_MOUSE_BUTTON_DOWN && e.button.button == SDL_BUTTON_LEFT)
        {
            if (buttons[i].isClicked())
            {
                // Transition to the corresponding part state
                switch (i)
                {
                case 0:  // "Load Main Game" - Can stay in TitleState or go to main game
                    setNextState(MainGameState::get()); // Assuming you have a MainGameState
                    break;

                case 1:  // "Load Part 1"
                    setNextState(Part1State::get());
                    break;

                case 2:  // "Load Part 2"
                    setNextState(Part2State::get());
                    break;

                case 3:  // "Load Part 3"
                    setNextState(Part3State::get());
                    break;
                }
            }
        }
    }
}

void TitleState::update() {
    //Fill the background
    SDL_SetRenderDrawColor(gRenderer, 0xFF, 0xFF, 0xFF, 0xFF);
    SDL_RenderClear(gRenderer);
}

void TitleState::render() {
    // Define vertical spacing
    constexpr int buttonSpacing = 20;  // Space between buttons
    constexpr int startY = (Global::kScreenHeight - ((LButton::kButtonHeight * kButtonCount) +
(buttonSpacing * (kButtonCount - 1)))) / 2;

    // Set button positions
    for (int i = 0; i < kButtonCount; ++i)
    {
        buttons[i].setPosition((Global::kScreenWidth - LButton::kButtonWidth) / 2, startY + i *
(LButton::kButtonHeight + buttonSpacing));
    }

    // Show the background
    mBackgroundTexture.render(0, 0);

    // Show the message
    mMessageTexture.render((Global::kScreenWidth - mMessageTexture.getWidth()) / 2.f, 20);

    // Render buttons
    for (int i = 0; i < kButtonCount; i++)
    {
        buttons[i].render();
    }
}

TitleState::TitleState() {
    //No public instantiation
}
```

5.44 src/ui/LButton.cpp File Reference

```
#include <ui/LButton.h>
```

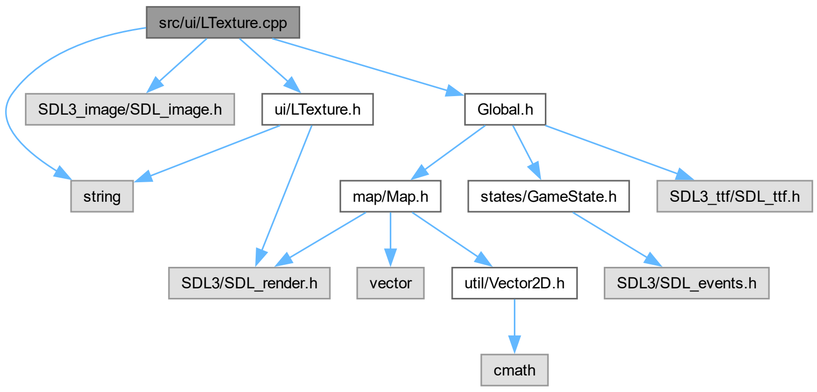
Include dependency graph for LButton.cpp:



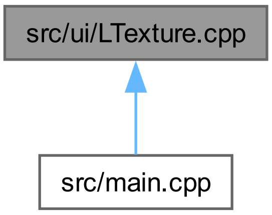
5.45 src/ui/LTexture.cpp File Reference

```
#include <string>
#include <SDL3_image/SDL_image.h>
#include <ui/LTexture.h>
#include <Global.h>
```

Include dependency graph for LTexture.cpp:



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



5.46 LTexture.cpp

Go to the documentation of this file.

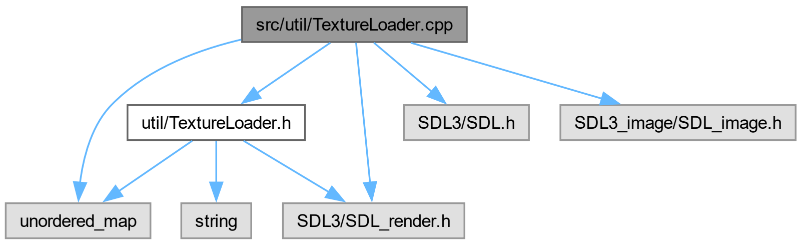
```
#include <string>
#include <SDL3_image/SDL_image.h>
#include <ui/LTexture.h>
#include <Global.h>

//LTexture Implementation
LTexture::LTexture() :
    //Initialize texture variables
    mTexture{ nullptr },
    mWidth{ 0 },
    mHeight{ 0 } {

}

LTexture::~LTexture() {
    //Clean up texture
    destroy();
}

bool LTexture::loadFromFile(std::string path) {
    //Clean up texture if it already exists
    destroy();

    //Load surface
    if (SDL_Surface* loadedSurface = IMG_Load(path.c_str()); loadedSurface == nullptr)
    {
        SDL_Log("Unable to load image %s! SDL_image error: %s\n", path.c_str(), SDL_GetError());
    } else
    {
        //Color key image
        if (!SDL_SetSurfaceColorKey(loadedSurface, true, SDL_MapSurfaceRGB(loadedSurface, 0x00, 0xFF,
0xFF)))
        {
            SDL_Log("Unable to color key! SDL error: %s", SDL_GetError());
        } else
        {
            //Create texture from surface
            if (mTexture = SDL_CreateTextureFromSurface(gRenderer, loadedSurface); mTexture ==
nullptr)
            {
                SDL_Log("Unable to create texture from loaded pixels! SDL error: %s\n",
SDL_GetError());
            } else
            {
                //Get image dimensions
                mWidth = loadedSurface->w;
                mHeight = loadedSurface->h;
            }
        }

        //Clean up loaded surface
        SDL_DestroySurface(loadedSurface);
    }

    //Return success if texture loaded
    return mTexture != nullptr;
}

bool LTexture::loadFromRenderedText(std::string textureText, SDL_Color textColor) {
    //Clean up existing texture
    destroy();

    //Load text surface
    if (SDL_Surface* textSurface = TTF_RenderText_Blended(gFont, textureText.c_str(), 0, textColor);
textSurface == nullptr)
    {
        SDL_Log("Unable to render text surface! SDL_ttf Error: %s\n", SDL_GetError());
    } else
    {
        //Create texture from surface
        if (mTexture = SDL_CreateTextureFromSurface(gRenderer, textSurface); mTexture == nullptr)
        {
            SDL_Log("Unable to create texture from rendered text! SDL Error: %s\n", SDL_GetError());
        } else
        {
            mWidth = textSurface->w;
            mHeight = textSurface->h;
        }

        //Free temp surface
        SDL_DestroySurface(textSurface);
    }

    //Return success if texture loaded
    return mTexture != nullptr;
}

void LTexture::destroy() {
    //Clean up texture
    SDL_DestroyTexture(mTexture);
    mTexture = nullptr;
    mWidth = 0;
    mHeight = 0;
}

void LTexture::setColor(Uint8 r, Uint8 g, Uint8 b) {
    SDL_SetTextureColorMod(mTexture, r, g, b);
}

void LTexture::setAlpha(Uint8 alpha) {
    SDL_SetTextureAlphaMod(mTexture, alpha);
}

void LTexture::setBlending(SDL_BlendMode blendMode) {
    SDL_SetTextureBlendMode(mTexture, blendMode);
}

void LTexture::render(float x, float y, SDL_FRect* clip, float width, float height, double degrees,
SDL_FPoint* center, SDL_FlipMode flipMode) {
    //Set texture position
    SDL_FRect dstRect = { x, y, static_cast<float>(mWidth), static_cast<float>(mHeight) };

    //Default to clip dimensions if clip is given
    if (clip != nullptr)
    {
        dstRect.w = clip->w;
        dstRect.h = clip->h;
    }

    //Resize if new dimensions are given
    if (width > 0)
    {
        dstRect.w = width;
    }
    if (height > 0)
    {
        dstRect.h = height;
    }

    //Render texture
    SDL_RenderTextureRotated(gRenderer, mTexture, clip, &dstRect, degrees, center, flipMode);
}

int LTexture::getWidth() {
    return mWidth;
}

int LTexture::getHeight() {
    return mHeight;
}
```

5.47 src/util/TextureLoader.cpp File Reference

Implementation of the Texture Loader Class.

```
#include <util/TextureLoader.h>
#include <SDL3/SDL_render.h>
#include <unordered_map>
#include <SDL3/SDL.h>
#include <SDL3_image/SDL_image.h>
```

Include dependency graph for TextureLoader.cpp:



Functions

- static std::unordered_map< std::string, SDL_Texture * > & getLoadedTextures ()

  Gets the static map of loaded textures.

5.47.1 Detailed Description

Implementation of the Texture Loader Class.

This file implements a texture loading system that manages SDL textures, including loading from files and memory management. It implements texture caching to prevent loading the same texture multiple times.

5.47.2 Function Documentation

5.47.2.1 getLoadedTextures()

```
static std::unordered_map< std::string, SDL_Texture * > & getLoadedTextures ()  [static]
```

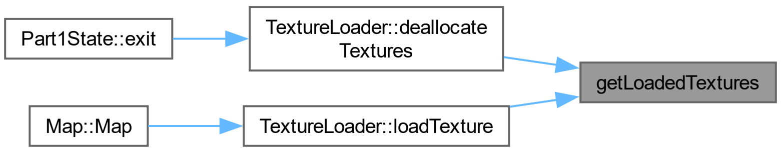
Gets the static map of loaded textures.

This function implements the Singleton pattern for the texture cache, ensuring there's only one instance of the texture map throughout the program's lifetime.

Returns

Reference to the static map of loaded textures

Here is the caller graph for this function:

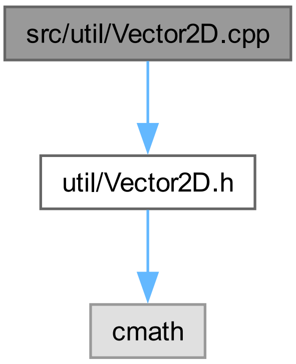

Part1State::exit
TextureLoader::deallocate Textures
getLoadedTextures
Map::Map
TextureLoader::loadTexture

5.48 src/util/Vector2D.cpp File Reference

Implementation of a the Vector2D class.

```
#include <util/Vector2D.h>
```

Include dependency graph for Vector2D.cpp:



5.48.1 Detailed Description

Implementation of a the Vector2D class.

Most of the implementation can be found in Vector2D.h. Most of the implementation is simply (standard math).

Index