

SpaceGame

1.0.1

Generated by Doxygen 1.8.11

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	Cell Class Reference	5
3.1.1	Constructor & Destructor Documentation	6
3.1.1.1	Cell(int x, int y)	6
3.2	Character Class Reference	6
3.2.1	Detailed Description	7
3.2.2	Member Data Documentation	8
3.2.2.1	direction	8
3.3	CharacterState Class Reference	8
3.3.1	Detailed Description	8
3.3.2	Member Function Documentation	9
3.3.2.1	update(Character &character, const UInt8 *keyboardState)=0	9
3.4	DeadState Class Reference	9
3.4.1	Detailed Description	9
3.4.2	Member Function Documentation	10
3.4.2.1	update(Character &character, const UInt8 *keyboardState)	10
3.5	IdleState Class Reference	10
3.5.1	Detailed Description	11

3.5.2	Member Function Documentation	11
3.5.2.1	update(Character &character, const Uint8 *keyboardState)	11
3.6	InitialisationError Class Reference	11
3.7	Level Class Reference	12
3.7.1	Detailed Description	12
3.8	MainCharacter Class Reference	12
3.8.1	Detailed Description	13
3.9	Map Class Reference	13
3.9.1	Detailed Description	13
3.10	Oxygen Class Reference	14
3.10.1	Detailed Description	14
3.11	PlayerControlledState Class Reference	14
3.11.1	Detailed Description	15
3.11.2	Member Function Documentation	15
3.11.2.1	update(Character &character, const Uint8 *keyboardState)	15
3.12	ReachedGoalState Class Reference	15
3.12.1	Detailed Description	16
3.12.2	Member Function Documentation	16
3.12.2.1	update(Character &character, const Uint8 *keyboardState)	16
3.13	SpaceGame Class Reference	16
3.13.1	Detailed Description	17
3.14	SuffocatingState Class Reference	17
3.14.1	Detailed Description	18
3.14.2	Member Function Documentation	18
3.14.2.1	update(Character &character, const Uint8 *keyboardState)	18
3.15	Texture Class Reference	18
3.15.1	Detailed Description	19
3.15.2	Constructor & Destructor Documentation	19
3.15.2.1	Texture(const std::string &fileName)	19
3.16	WanderingState Class Reference	19
3.16.1	Detailed Description	19
3.16.2	Member Function Documentation	20
3.16.2.1	update(Character &character, const Uint8 *keyboardState)	20

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Cell	5
Character	6
MainCharacter	12
CharacterState	8
DeadState	9
IdleState	10
PlayerControlledState	14
ReachedGoalState	15
SuffocatingState	17
WanderingState	19
exception	
InitialisationError	11
Level	12
Map	13
Oxygen	14
SpaceGame	16
Texture	18

Chapter 2

Class Index

2.1 Class List

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

Cell	5
Character	
The abstract character class	6
CharacterState	
The character state class is used to store the character's current state	8
DeadState	
The DeadState is for when the character's health had reached 0	9
IdleState	
The IdleState is for when there is no danger and the player is not controlling the character	10
InitialisationError	11
Level	
This class generates the base of the level	12
MainCharacter	
The MainCharacter that will be controlled by the user	12
Map	
The Class that handles the creation of rooms	13
Oxygen	14
PlayerControlledState	
The PlayerControlledState is for when the user is pressing WASD to move the character	14
ReachedGoalState	
The character enters this class once they have reached the end goal	15
SpaceGame	
The main class	16
SuffocatingState	
The suffocating class is for when the character is on a cell with a low oxygen level	17
Texture	
Loads and renders images in the window	18
WanderingState	
WanderingState makes the character move around the room randomly	19

Chapter 3

Class Documentation

3.1 Cell Class Reference

Public Member Functions

- `Cell ()`
A constructor.
- `Cell (int x, int y)`
An alternate constructor.
- `~Cell ()`
A destructor.
- `int getX ()`
Gets the `Cell`'s `X` value.
- `int getY ()`
Gets the `Cell`'s `Y` value.
- `int getOxygenLevel ()`
Gets the `Cell`'s `oxygenLevel`.
- `int setX (int newX)`
Sets the `Cells` `X` value.
- `int setY (int newY)`
Sets the `Cells` `Y` value.
- `int setOxygenLevel (int newOxygenLevel)`
Sets the `Cell`'s `oxygenLevel`.

Public Attributes

- `bool isRoom = false`
Whether the cell is part of a room.
- `bool isDoor = false`
Whether the cell is a door.
- `bool isGoal = false`
Represents the goal for the player.
- `int oxygenLevel = 100`
The `oxygenLevel` of the cell.

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Cell::Cell (int x, int y)

An alternate constructor.

This constructor requires an X and Y for the [Cell](#)

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

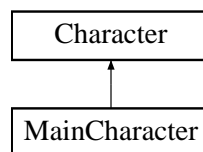
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Cell.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Cell.cpp

3.2 Character Class Reference

The abstract character class.

```
#include <Character.h>
```

Inheritance diagram for Character:



Public Member Functions

- [Character](#) ()
A constructor.
- [~Character](#) ()
A destructor.
- int [getX](#) ()
Gets the characters X value.
- int [getY](#) ()
Gets the characters Y value.
- int [getSize](#) ()
Gets the characters size.
- int [getSpeed](#) ()
Gets the characters speed.
- int [setX](#) (int newX)
Sets the characters X value.
- int [setY](#) (int newY)
Sets the characters Y value.
- int [setSpeed](#) (int newSpeed)
Sets the characters current speed.
- bool [isCellARoom](#) (int x, int y)
Checks whether a cell is a room.

- bool `isCellADoor` (int x, int y)
Checks whether a cell is a door.
- int `getOxygenLevel` (int x, int y)
Gets the oxygen level of a given room.
- bool `reachedGoal` (int x, int y)
Checks whether the player has won.
- void `chooseDirection` (const Uint8 *keyboardState)
- void `moveCharacter` (const Uint8 *keyboardState)
Changes the character's X and Y value depending on the player's input.

Public Attributes

- bool `isWandering` = false
Checks whether the character is wandering.
- std::shared_ptr< `CharacterState` > `state`
An shared pointer to the character's state.
- std::shared_ptr< `Level` > `currentRoom`
Shared pointer to the `Level` loaded in `SpaceGame`.
- double `health` = 100
A double for the character's health.
- bool `isAlive` = true
Boolean for whether character is alive.
- bool `hasWon` = false
- int `direction` = 1
Integer for the random direction.
- double `timer` = 0
Integer to store time spent in a state.
- int `suffocatingSpeed` = 1
Integers for the different movement speeds.
- int `wanderSpeed` = 2
- int `walkSpeed` = 3
- int `runSpeed` = 3
- int `lowOxygenThershold` = 40
The three oxygen levels that are used to change alter `Character`'s state.
- int `acceptableOxygenLevel` = 50
- int `dangeroursOxygenLevel` = 20
- int `windowWidth` = 800
Window size decided by `SpaceGame` window size.
- int `windowHeight` = 800

3.2.1 Detailed Description

The abstract character class.

This class is the base for the main character and the NPC. It contains all the functions needed to make the character move and react to different states.

3.2.2 Member Data Documentation

3.2.2.1 `int Character::direction = 1`

Integer for the random direction.

Direction is used to decide the direction the character will move in when in the wandering state

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

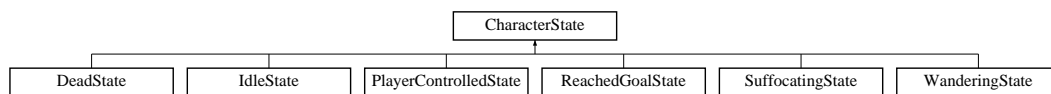
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Character.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Character.cpp

3.3 CharacterState Class Reference

The character state class is used to store the character's current state.

```
#include <CharacterState.h>
```

Inheritance diagram for CharacterState:



Public Member Functions

- [CharacterState](#) ()
A constructor.
- [~CharacterState](#) ()
A destructor.
- virtual void [update](#) ([Character](#) &character, const Uint8 *keyboardState)=0
The update function checks and updates the character state.

Public Attributes

- double [timer](#) = 0
Used to time how long the character has been in a state.

3.3.1 Detailed Description

The character state class is used to store the character's current state.

This class is the class all the other states inherit from. It is used in the [Character](#) to store the current state

3.3.2 Member Function Documentation

3.3.2.1 void CharacterState::update (Character &character, const Uint8 *keyboardState) [pure virtual]

The update function checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implemented in [IdleState](#), [PlayerControlledState](#), [DeadState](#), [ReachedGoalState](#), [SuffocatingState](#), and [WanderingState](#).

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

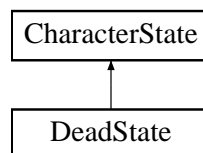
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/CharacterState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/CharacterState.cpp

3.4 DeadState Class Reference

The [DeadState](#) is for when the character's health had reached 0.

```
#include <DeadState.h>
```

Inheritance diagram for DeadState:



Public Member Functions

- [DeadState](#) ()
A constructor.
- [~DeadState](#) ()
A destructor.
- void [update](#) ([Character](#) &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Additional Inherited Members

3.4.1 Detailed Description

The [DeadState](#) is for when the character's health had reached 0.

This class inherits from the [CharacterState](#) class. When the character enters the dead state they can't leave it.

3.4.2 Member Function Documentation

3.4.2.1 void DeadState::update (Character & character, const Uint8 * keyboardState) [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implements [CharacterState](#).

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

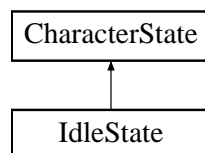
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/DeadState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/DeadState.cpp

3.5 IdleState Class Reference

The [IdleState](#) is for when there is no danger and the player is not controlling the character.

```
#include <IdleState.h>
```

Inheritance diagram for IdleState:



Public Member Functions

- [IdleState](#) ()
A constructor.
- [~IdleState](#) ()
A destructor.
- void [update](#) ([Character](#) &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Public Attributes

- double [FRAME_RATE](#) = 60
Used to add to the timer.

3.5.1 Detailed Description

The [IdleState](#) is for when there is no danger and the player is not controlling the character.

This class inherits from the [CharacterState](#) class. The character does not do anything in the [IdleState](#) it just checks for when it should update to another state

3.5.2 Member Function Documentation

3.5.2.1 void IdleState::update ([Character](#) & *character*, const Uint8 * *keyboardState*) [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the [Character](#) which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

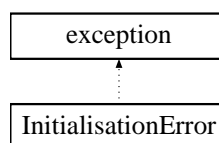
Implements [CharacterState](#).

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/IdleState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/IdleState.cpp

3.6 InitialisationError Class Reference

Inheritance diagram for InitialisationError:



Public Member Functions

- **InitialisationError** (const std::string &msg)
- const char * **what** ()

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/InitialisationError.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/InitialisationError.cpp

3.7 Level Class Reference

This class generates the base of the level.

```
#include <Level.h>
```

Public Member Functions

- [Level](#) ()
A constructor.
- [~Level](#) ()
A destructor.
- int [getCellSize](#) ()
Return the cellSize.
- void [makeGrid](#) (int Window_Width, int Window_Height)
Fills grid with vectors of shared pointers to cells.

Public Attributes

- std::vector< std::vector< std::shared_ptr< [Cell](#) > > > [grid](#)
The base grid that contains the cells.

Protected Attributes

- int [cellSize](#) = 50
The size that the cell will be rendered at.

3.7.1 Detailed Description

This class generates the base of the level.

This class creates a vector of vector of shared pointers to cells

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

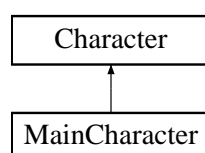
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Level.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Level.cpp

3.8 MainCharacter Class Reference

The [MainCharacter](#) that will be controlled by the user.

```
#include <MainCharacter.h>
```

Inheritance diagram for MainCharacter:



Public Member Functions

- [MainCharacter](#) ()
A constructor.
- [~MainCharacter](#) ()
A destructor.

Additional Inherited Members

3.8.1 Detailed Description

The [MainCharacter](#) that will be controlled by the user.

This class is for the games main character that the user will control. It inherits from the character class.

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/MainCharacter.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/MainCharacter.cpp

3.9 Map Class Reference

The Class that handles the creation of rooms.

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

Public Member Functions

- [Map](#) ()
A Constructor.
- [~Map](#) ()
A Deconstructor.
- int [random](#) (int smallestValue, int largestValue)
Generates a random integer.
- void [LoadMap](#) (std::string filename, [Level](#) room)
Loads in a map from a txt file.
- void [generateMap](#) ([Level](#) level)
Randomly generates a map and modifies the level.

3.9.1 Detailed Description

The Class that handles the creation of rooms.

This class modifies the level class to make patterns out of the cells and turn them into rooms using all of it's various functions.

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Map.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Map.cpp

3.10 Oxygen Class Reference

```
#include <Oxygen.h>
```

Public Member Functions

- [Oxygen](#) ()
A constructor.
- [~Oxygen](#) ()
A destructor.
- void [update](#) (int cellSize, [Level](#) grid)
Update method updates the oxygen level each frame.
- void [addOxygen](#) (int mouseX, int mouseY, int cellSize, [Level](#) grid)
Adds oxygen based on where the mouse was clicked.
- void [removeOxygen](#) (int mouseX, int mouseY, int cellSize, [Level](#) grid)
Removes oxygen based on where the mouse was clicked.
- int [getOxygenReserves](#) ()
Getter for getting the oxygen reserve level.
- int [setOxygenReserves](#) (int newOxygenReserveLevel)
Setter for setting the oxygen reserve level.

Public Attributes

- int [cellX](#)
Initialising cellX and cellY for cell loop.
- int [cellY](#)

3.10.1 Detailed Description

This class manages how the oxygen spreads through the room cells and how much oxygen is able to be placed.

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

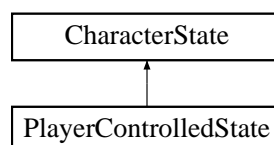
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Oxygen.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Oxygen.cpp

3.11 PlayerControlledState Class Reference

The [PlayerControlledState](#) is for when the user is pressing WASD to move the character.

```
#include <PlayerControlledState.h>
```

Inheritance diagram for PlayerControlledState:



Public Member Functions

- [PlayerControlledState](#) ()
A constructor.
- [~PlayerControlledState](#) ()
A destructor.
- void [update](#) ([Character](#) &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Additional Inherited Members

3.11.1 Detailed Description

The [PlayerControlledState](#) is for when the user is pressing WASD to move the character.

This class checks for keyboard input and updates the character's X and Y depending on what key was pressed

3.11.2 Member Function Documentation

3.11.2.1 void [PlayerControlledState::update](#) ([Character](#) & *character*, const Uint8 * *keyboardState*) [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implements [CharacterState](#).

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

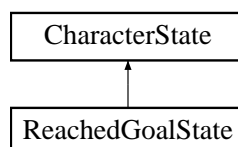
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/PlayerControlledState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/PlayerControlledState.cpp

3.12 ReachedGoalState Class Reference

The character enters this class once they have reached the end goal.

```
#include <ReachedGoalState.h>
```

Inheritance diagram for ReachedGoalState:



Public Member Functions

- [ReachedGoalState](#) ()
A constructor.
- [~ReachedGoalState](#) ()
A destructor.
- void [update](#) ([Character](#) &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Additional Inherited Members

3.12.1 Detailed Description

The character enters this class once they have reached the end goal.

This class triggers the You Won screen and doesn't let the character move. Once the character has entered this state it can't leave it.

3.12.2 Member Function Documentation

3.12.2.1 void [ReachedGoalState::update](#) ([Character](#) & *character*, const Uint8 * *keyboardState*) [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implements [CharacterState](#).

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/ReachedGoalState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/ReachedGoalState.cpp

3.13 SpaceGame Class Reference

The main class.

```
#include <SpaceGame.h>
```

Public Member Functions

- [SpaceGame](#) ()
A constructor.
- [~SpaceGame](#) ()
A deconstructor.
- void [run](#) ()

Public Attributes

- int `mouse_X`
Coordinates of the mouse.
- int `mouse_Y`

Static Public Attributes

- static const int `WINDOW_WIDTH` = 800
The window width.
- static const int `WINDOW_HEIGHT` = 800
The window height.

3.13.1 Detailed Description

The main class.

This is the main class where the game is loaded and run.

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

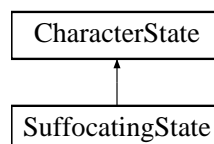
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/SpaceGame.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/SpaceGame.cpp

3.14 SuffocatingState Class Reference

The suffocating class is for when the character is on a cell with a low oxygen level.

```
#include <SuffocatingState.h>
```

Inheritance diagram for SuffocatingState:



Public Member Functions

- `SuffocatingState` ()
A constructor.
- `~SuffocatingState` ()
A destructor.
- void `update` (`Character` &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Additional Inherited Members

3.14.1 Detailed Description

The suffocating class is for when the character is on a cell with a low oxygen level.

This class alters the characters speed and health depending on the oxygen level of the cell it's currently on

3.14.2 Member Function Documentation

3.14.2.1 void SuffocatingState::update (Character & character, const Uint8 * keyboardState) [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implements [CharacterState](#).

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/SuffocatingState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/SuffocatingState.cpp

3.15 Texture Class Reference

Loads and renders images in the window.

```
#include <Texture.h>
```

Public Member Functions

- [Texture](#) (const std::string &fileName)
A constructor.
- [~Texture](#) ()
A destructor.
- SDL_Texture * [getTexture](#) ()
Loads the texture.
- void [render](#) (SDL_Renderer *renderer, int x, int y, int width, int height)
Renders the image in the window.
- void [alterTransparency](#) (int transparencyLevel)
Alters the alpha value of the image to make it appear transparent in the window.

3.15.1 Detailed Description

Loads and renders images in the window.

This class is used for all the images in the game. It loads textures from a given file location and can alter the image transparency if the image is a PNG.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Texture::Texture (const std::string & fileName)

A constructor.

Requires a file path to load the image from

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

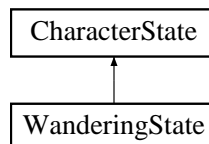
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Texture.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/Texture.cpp

3.16 WanderingState Class Reference

[WanderingState](#) makes the character move around the room randomly.

```
#include <WanderingState.h>
```

Inheritance diagram for WanderingState:



Public Member Functions

- [WanderingState](#) ()
A constructor.
- [~WanderingState](#) ()
A destructor.
- void [update](#) ([Character](#) &character, const Uint8 *keyboardState)
The update function that checks and updates the character state.

Additional Inherited Members

3.16.1 Detailed Description

[WanderingState](#) makes the character move around the room randomly.

If there is no danger or player input the character will be moved around the room in a random direction

3.16.2 Member Function Documentation

3.16.2.1 `void WanderingState::update (Character & character, const Uint8 * keyboardState)` [virtual]

The update function that checks and updates the character state.

Runs on every frame of the game to check the level and update the character state. It takes in the character which needs it's state updating and keyboardState to check whether the user has pressed the keyboard since the last update

Implements [CharacterState](#).

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

- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/WanderingState.h
- C:/Users/Warwick/Documents/GitHub/Desktop_game/our code/SDL_project/WanderingState.cpp

Index

- Cell, [5](#)
 - Cell, [6](#)
- Character, [6](#)
 - direction, [8](#)
- CharacterState, [8](#)
 - update, [9](#)
- DeadState, [9](#)
 - update, [10](#)
- direction
 - Character, [8](#)
- IdleState, [10](#)
 - update, [11](#)
- InitialisationError, [11](#)
- Level, [12](#)
- MainCharacter, [12](#)
- Map, [13](#)
- Oxygen, [14](#)
- PlayerControlledState, [14](#)
 - update, [15](#)
- ReachedGoalState, [15](#)
 - update, [16](#)
- SpaceGame, [16](#)
- SuffocatingState, [17](#)
 - update, [18](#)
- Texture, [18](#)
 - Texture, [19](#)
- update
 - CharacterState, [9](#)
 - DeadState, [10](#)
 - IdleState, [11](#)
 - PlayerControlledState, [15](#)
 - ReachedGoalState, [16](#)
 - SuffocatingState, [18](#)
 - WanderingState, [20](#)
- WanderingState, [19](#)
 - update, [20](#)