txtEngine

v4.0

Generated by Doxygen 1.7.5.1

Mon Sep 19 2011 22:22:27

Contents

Chapter 1

Documentation for the txtEngine Project

1.1 Date Updated:

14-08-2011

1.2 What is txtEngine?

txtEngine is an interpreter for text only adventure games. Games are written using the XML language making it easy for anyone to write and play their own games.

1.3 Links:

- txtEngine Project Site on Github: https://github.com/smilefreak/txtEngine
- TinyXML Documentation: http://www.grinninglizard.com/tinyxmldocs/index.html

1.4 Report Bugs:

Please report any bugs here: https://github.com/smilefreak/txtEngine/issues

1.5 Authors:

Toby Herbert, Michael Abrams, James Boocock, Tatai Nikora

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 documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Area Class Reference

Public Member Functions

- bool has_description (std::string desc_id)
- std::string get_status ()
- bool has_current_desc ()
- int get_num_items ()
- std::string get_description ()
- void remove_item (int index)
- void remove_item (std::string item_id)
- void add_item (Item *new_item)
- Item * get_item (int index)
- std::string get_id ()
- bool has_item (std::string item_to_find)
- Item * get_item (std::string item_id, unsigned int &item)
- void add_description (StateDescriptor *desc)
- void add_command (AreaCommand *command_name)
- int get_num_commands ()
- AreaCommand * get_command (int index)
- AreaCommand * has_command (std::string command_name)
- int get_num_descriptions ()
- StateDescriptor * get_descriptor (int index)
- void unlock (std::string area_command_id)
- Area (const char *id, const char *desc_id, const char *status)
- ~Area ()

Protected Attributes

- std::vector< Item * > **items**
- int **num_descriptions**
- int **num_commands**
- int **num_items**
- std::string **status**
- std::string **id**
- std::string **curr_desc_id**
- std::vector< StateDescriptor * > **description**
- std::vector< AreaCommand * > **commands**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Area::Area (const char * *id*, const char * *desc_id*, const char * *status*)

The constructor for an Area object.

Parameters

in	*id*	The id of the area.
in	*desc_id*	The id of the area description.
in	*status*	The status of the area.

4.1.1.2 Area::~Area ( )

Area Deconstructor.

4.1.2 Member Function Documentation

4.1.2.1 void Area::add_command (**AreaCommand** * *command_name*)

Adds an AreaCommand to an area.

Parameters

in	*A*	pointer to an AreaCommand.

4.1.2.2 void Area::add_description (**StateDescriptor** * *desc*)

Adds a StateDescriptor to an area.

Parameters

in	*desc*	A pointer to a StateDescriptor object.

4.1.2.3 void Area::add_item (Item * *new_item*)

Adds an item to the area.

Parameters

in	*new_item*	A pointer to the item to add to the items vector.

4.1.2.4 AreaCommand * Area::get_command (int *index*)

Get an AreaCommand for this area by index.

Parameters

in	*index*	The index of the AreaCommand in the commands vector.

Returns

A pointer to the AreaCommand or null if it doesn't exist.

4.1.2.5 std::string Area::get_description ( )

Get the area description.

Returns

String description of the area.

4.1.2.6 StateDescriptor * Area::get_descriptor (int *index*)

Gets a StateDescriptor for the area by index.

Parameters

in	*index*	The index of the StateDescriptor in the description vector.

Returns

A pointer to the StateDescriptor or null if it doesn't exist.

4.1.2.7 std::string Area::get_id ( )

Get the area id.

Returns

The id of the area.

4.1.2.8 Item * Area::get_item (int *index*)

Get the item in the area from items vector by index.

Parameters

in	*index*	The index of the item in the vector.

Returns

A pointer to the item.

4.1.2.9 Item * Area::get_item (std::string *item_id,* unsigned int & *item*)

Gets a pointer to the item by item id and sets index to the index of the item in the vector.

Parameters

in	*item_id*	The id of the item.
out	*index*	The address of the item's index member variable.

Returns

A pointer to the item.

4.1.2.10 int Area::get_num_commands ( )

Get the number of commands for this area.

Returns

The number of AreaCommands for this area.

4.1.2.11 int Area::get_num_descriptions ( )

Get the number if descriptions for the area.

Returns

The number of descriptions for this area.

4.1.2.12 int Area::get_num_items ( )

Accessor for the number of items in the Area.

Returns

The number of items in the area.

4.1.2.13 std::string Area::get_status ( )

This accessor method returns the status of the area.

Returns

The status of the Area.

4.1.2.14 AreaCommand * Area::has_command (std::string *command_name*)

Gets an AreaCommand for this area by name.

Parameters

in	*command_name*	The name of the command to get.

Returns

A pointer to the AreaCommand or null if it doesn't exist.

4.1.2.15 bool Area::has_current_desc ( )

Calls the has_description method.

The function arguments listed with "param" will be compared to the declaration and verified.

See also

has_description();

Returns

True if the area has the current description otherwise false.

4.1.2.16 bool Area::has_description (std::string *desc_id*)

Checks if an area has this description.

Parameters

in	*desc_id*	A description ID.

Returns

Returns true if 'description' holds desc_id, otherwise returns false.

4.1.2.17 bool Area::has_item (std::string *item_to_find*)

Checks whether the area has an item.

Parameters

in	*item_to_find*	

Returns

True if the area (items vector) contains the item.

4.1.2.18 void Area::remove_item (int *index*)

Remove an item from the area by index.

Parameters

in	*index*	The index of the item to remove.

Returns

Description of returned value.

4.1.2.19 void Area::remove_item (std::string *item_id*)

Removes an item from the area by the item id.

Parameters

in	*item_id*	The id of the item to be removed.

4.1.2.20 void Area::unlock (std::string *area_command_id*)

Unlocks an AreaCommand of this area.

Parameters

in	*area_command_id*	The id of an AreaCommand to unlock.

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

- /home/cshome/m/mabrams/345/txtEngine/Area.h
- /home/cshome/m/mabrams/345/txtEngine/Area.cpp

4.2 AreaCommand Class Reference

Public Member Functions

- AreaCommand (const char *callmeby, const char *areatomoveto, const char *status_command, const char *depends_command, std::vector< std::string > *synonyms, bool locked)
- ~AreaCommand ()
- std::string get_depends ()
- std::string get_status ()
- std::string get_name ()
- std::string get_area ()
- std::string get_message ()
- void set_message (const char *to_message)
- bool find (std::string to_find)
- void unlock ()
- bool is_locked ()
- bool has_synonym (std::string item)

Protected Attributes

- bool **locked**
- std::string **name**
- std::string **status**
- std::string **message**
- std::string **depends**
- std::string **move_to_area**
- std::vector< std::string > * **synonyms**

4.2.1 Constructor & Destructor Documentation

4.2.1.1 AreaCommand::AreaCommand (const char * *callmeby*, const char * *areatomoveto*, const char * *status_command*, const char * *depends_command*, std::vector< std::string > * *synonyms*, bool *locked*)

The constructor for an AreaCommand.

Parameters

in	*callmeby*	The name of this command.
in	*areamoveto*	The area to move to when this command is called.
in	*status_command*	The status id to change to.
in	*depends_command*	
in	*synonyms*	A vector containing synonyms of 'callmeby'.
in	*locked*	If true command cannot be called.

4.2.1.2 AreaCommand::~AreaCommand ( )

The AreaCommand Destructor

4.2.2 Member Function Documentation

4.2.2.1 bool AreaCommand::find (std::string *to_find*)

Compares the name of the command with a string.

Parameters

in	*to_find*	A string to compare with the command name.

Returns

True if the strings match otherwise false.

4.2.2.2 std::string AreaCommand::get_area ( )

Get the name of the area to move to when this command is used.

Returns

The name of the area to move to.

4.2.2.3 std::string AreaCommand::get_depends ( )

Returns what the AreaCommand depends on.

Returns

An id of an item the command depends on.

4.2.2.4 std::string AreaCommand::get_message ( )

Get the message to print when this command is used.

Returns

A message to print for this command.

4.2.2.5 std::string AreaCommand::get_name ( )

Get the name of the AreaCommand.

Returns

The name of the AreaCommand.

4.2.2.6 std::string AreaCommand::get_status ( )

Get the status of the AreaCommand

Returns

The status of the AreaCommand.

4.2.2.7 bool AreaCommand::has_synonym (std::string *item*)

Checks if the area command has a synonym matching a string.

Parameters

in	*item*	The name to check

Returns

True if the synonym list has the string or false if not.

4.2.2.8 bool AreaCommand::is_locked ( )

Checks whether this area command is locked.

Returns

True if the command is locked or false if it is unlocked.

4.2.2.9 void AreaCommand::set_message (const char * *to_message*)

Change the message for this command.

Parameters

in	*to_message*	The new message for this command.

4.2.2.10 void AreaCommand::unlock ( )

Unlocks the command so it can be called.

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

- /home/cshome/m/mabrams/345/txtEngine/AreaCommand.h
- /home/cshome/m/mabrams/345/txtEngine/AreaCommand.cpp

4.3 combine Class Reference

Public Member Functions

- combine (std::string id, std::string first_id, std::string second_id)
- ~combine ()
- Item * get_combination ()
- std::string get_id ()
- std::string get_first_id ()
- std::string get_second_id ()
- void set_combination (Item *item)
- void set_description (StateDescriptor *d)
- std::string get_description ()

4.3.1 Constructor & Destructor Documentation

4.3.1.1 combine::combine (std::string *id,* std::string *first_id,* std::string *second_id*)

Constructor for a combine object.

Parameters

in	*id*	The id for the object.
in	*first_id*	The id of the first item to combine.
in	*second_id*	The id of the second object to combine.

4.3.1.2 combine::∼combine ( )

Destructor for combine object.

4.3.2 Member Function Documentation

4.3.2.1 Item ∗ combine::get_combination ( )

Get

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.2 std::string combine::get_description ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.3 std::string combine::get_first_id ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.4 std::string combine::get_id ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.5 std::string combine::get_second_id ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.6 void combine::set_combination (Item * *item*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.3.2.7 void combine::set_description (StateDescriptor * *d*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

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

- /home/cshome/m/mabrams/345/txtEngine/combine.h
- /home/cshome/m/mabrams/345/txtEngine/combine.cpp

4.4 Item Class Reference

Public Member Functions

- void remove_item (std::string item_id)
- void flip_locked ()

- bool is_locked ()
- bool has_container ()
- std::string print_contained_items ()
- Item * get_item (std::string item_id)
- void add_item (Item *)
- bool has_combine ()
- combine * get_combine ()
- void set_combine (combine *c)
- bool has_description (std::string desc_id)
- bool has_current_desc ()
- bool has_synonym (std::string item)
- std::string get_description ()
- void add_description (StateDescriptor *desc)
- void change_collectable (bool flip)
- bool is_collectable ()
- std::string get_id ()
- int get_num_commands ()
- void add_command (ItemCommand *command_name)
- ItemCommand * get_command (int index)
- ItemCommand * get_command (std::string command_name)
- int get_num_descriptions ()
- StateDescriptor * get_descriptor (int index)
- std::string get_depends ()
- void state_change (std::string to_change)
- Item (bool collect, const char *identifier, const char *initial_state, std::vector< std::string > *synonyms, const char *depends, bool container, bool locked)
- ~Item ()

Protected Attributes

- bool **collectable**
- int **num_descriptions**
- int **num_commands**
- int **num_items**
- std::string **id**
- bool **container**
- bool **locked**
- combine * **combine_var**
- std::vector< Item * > **contains**
- std::string **curr_desc_id**
- std::vector< StateDescriptor * > **description**
- std::vector< ItemCommand * > **commands**
- std::vector< std::string > * **synonyms**
- std::string **depends**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Item::Item (bool *collect*, const char * *identifier*, const char * *initial_state*, std::vector< std::string > * *synonyms*, const char * *depends*, bool *container*, bool *locked*)

The constructor for an Item.

Parameters

in	*collect*	Whether this item is collectable or not.
in	*identifier*	An identifier for this item.
in	*initial_state*	The initial state of the item.
in	*synonyms*	A vector of synonyms for the name of this item.
in	*depends*	An item this item depends on.
in	*container*	Whether this item is a container.
in	*locked*	Whether this item is locked.

4.4.1.2 Item::~Item ( )

The destructor for an Item.

4.4.2 Member Function Documentation

4.4.2.1 void Item::add_command (ItemCommand * *command_name*)

Add a command to this item.

Parameters

in	*command_name*	A pointer to an ItemCommand object.

4.4.2.2 void Item::add_description (StateDescriptor * *desc*)

Add a StateDescriptor for this item.

Parameters

in	*desc*	A pointer to a StateDescriptor object to add.

4.4.2.3 void Item::add_item (Item * *new_item*)

Adds an item to the contains vector.

Parameters

The	pointer to an item.

4.4.2.4 void Item::change_collectable (bool *flip*)

Flip the value of collectable for this item.

Parameters

in	*flip*	True flips the value, false leaves it unchanged.

4.4.2.5 void Item::flip_locked ( )

Flips the locked variable for this item..

4.4.2.6 combine * Item::get_combine ( )

Accessor for a combine object.

Returns

A pointer to a combine object.

4.4.2.7 ItemCommand * Item::get_command (int *index*)

Gets a command from the commands vector for this item by index.

Parameters

in	*index*	The index of the item in the vector.

Returns

An ItemCommand object at the specified index.

4.4.2.8 ItemCommand * Item::get_command (std::string *command_name*)

Gets a command from the commands vector of this item by command_name.

Parameters

in	*command_name*	A string - the name of the command.

Returns

An ItemCommand object with the specified name.

4.4.2.9 std::string Item::get_depends ( )

Changes the state of the item.

Returns

Returns a string - what the item depends on.

4.4.2.10 std::string Item::get_description ( )

Gets the item description.

Returns

The description of the item.

4.4.2.11 StateDescriptor * Item::get_descriptor (int *index*)

Get a StateDescriptor from the descriptions vector by index.

Parameters

in	*index*	The index of the StateDescriptor in the vector.

Returns

A StateDescriptor at the specified index.

4.4.2.12 std::string Item::get_id ( )

Gets the id of the item.

Returns

A string - the id of the item.

4.4.2.13 Item * Item::get_item (std::string *item_id*)

Returns a pointer to an item by id or null if it does not exist.

Parameters

item_id	The id of the item to get.

Returns

A pointer to an item.

4.4.2.14 int Item::get_num_commands ( )

Gets the number of commands for this item.

Returns

The number of commands this item has.

4.4.2.15 int Item::get_num_descriptions ( )

Get the number of descriptions for this item.

Returns

The number of descriptions for this item.

4.4.2.16 bool Item::has_combine ( )

Checks whether this Item can combine with another.

Returns

True if this item can be combined with another otherwise false.

4.4.2.17 bool Item::has_container ( )

Checks whether the the Item is a container for other items.

Returns

True if the item is a container or false if not.

4.4.2.18 bool Item::has_current_desc ( )

Check whether this item has the current description. Calls has_description method passing the curr_desc_id.

Returns

True if the item has the current description otherwise false.

4.4.2.19 bool Item::has_description (std::string *desc_id*)

Check whether this item has a certain description id.

Parameters

in	*desc_id*	A string of an item description id.

Returns

True if this item contains the discription otherwise false.

4.4.2.20 bool Item::has_synonym (std::string *item*)

Checks whether this item has a particular synonym.

Parameters

in	*item*	A string that may be a synonym.

Returns

True if the item has the synonym otherwise false.

4.4.2.21 bool Item::is_collectable ( )

Checks whether this item is collectable.

Returns

Description of returned value.

4.4.2.22 bool Item::is_locked ( )

Checks whether the the Item is locked.

Returns

True if the item is locked otherwise false.

4.4.2.23 std::string Item::print_contained_items ( )

Returns a string with all items the item contains.

Returns

A string of items this item contains.

4.4.2.24 void Item::remove_item (std::string *item_id*)

Removes an item from inside this item by id.

Parameters

A	string - the id of the item to remove.

4.4.2.25 void Item::set_combine (combine * *c*)

A mutator for a combine object.

Parameters

in	*c*	A pointer to a combine object.

4.4.2.26 void Item::state_change (std::string *to_change*)

Changes the state of the item.

Parameters

in	*to_change*	A string - to change the state of the item to.

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

- /home/cshome/m/mabrams/345/txtEngine/Item.h
- /home/cshome/m/mabrams/345/txtEngine/Item.cpp

4.5 ItemCommand Class Reference

Public Member Functions

- bool **is_combine** ()
- **ItemCommand** (const char *callmeby, const char *state_mutator, bool chng_collec, bool collec_dep, const char *area_chng, const char *status_command, const char *depends, std::vector< std::string > *synonyms, std::string unlock, bool combine)
- std::string get_depends ()
- bool get_change_collect ()
- bool get_collect_dependent ()
- std::string get_area_change ()
- std::string get_status ()
- std::string get_message ()
- std::string get_name ()

- std::string get_state_change ()
- bool check_synonyms (std::string command)
- void set_message (const char *to_message)
- bool **unlocks** ()
- std::string **unlock_areacommand_string** ()
- std::string **unlock_area_string** ()
- std::string **get_unlock_string** ()

Protected Attributes

- std::string **name**
- std::string **state_change**
- std::string **message**
- std::string **area_change**
- std::string **depends**
- std::string **status**
- std::string **unlock**
- bool **call_combine**
- std::vector< std::string > * **synonyms**
- bool **change_collect**
- bool **collect_dependent**

4.5.1 Member Function Documentation

4.5.1.1 bool ItemCommand::check_synonyms (std::string *command*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.2 std::string ItemCommand::get_area_change ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.3 bool ItemCommand::get_change_collect ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.4 bool ItemCommand::get_collect_dependent ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.5 std::string ItemCommand::get_depends ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.6 std::string ItemCommand::get_message ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.7 std::string ItemCommand::get_name ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.8 std::string ItemCommand::get_state_change ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.9 std::string ItemCommand::get_status ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.5.1.10 void ItemCommand::set_message (const char * *to_message*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

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

- /home/cshome/m/mabrams/345/txtEngine/ItemCommand.h
- /home/cshome/m/mabrams/345/txtEngine/ItemCommand.cpp

4.6 StateDescriptor Class Reference

Public Member Functions

- StateDescriptor (const char ∗identifier)
- void set_description (const char ∗desc)
- ∼StateDescriptor ()
- std::string get_id ()
- std::string get_description ()

Protected Attributes

- std::string **id**
- std::string **description**

4.6.1 Constructor & Destructor Documentation

4.6.1.1 StateDescriptor::StateDescriptor (const char ∗ *identifier*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.6.1.2 StateDescriptor::~StateDescriptor ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.6.2 Member Function Documentation

4.6.2.1 std::string StateDescriptor::get_description ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.6.2.2 std::string StateDescriptor::get_id ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.6.2.3 void StateDescriptor::set_description (const char * *desc*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

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

- /home/cshome/m/mabrams/345/txtEngine/StateDescriptor.h
- /home/cshome/m/mabrams/345/txtEngine/StateDescriptor.cpp

4.7 World Class Reference

Public Member Functions

- std::string get_author ()
- std::string get_language ()
- Area * get_active_area ()
- Area * **get_area** (int index)
- void **add_area** (Area *new_area)
- int **get_num_areas** ()
- Area * **get_area** (std::string area_id)
- bool **init_active_area** ()
- void **change_area** (std::string name)
- **World** (const char *lang, const char *auth, const char *init_area)
- ~World ()

Protected Attributes

- std::string **language**
- std::string **author**
- std::vector< Area * > **areas**
- std::string **initial_area**
- int **num_areas**
- Area * **active_area**

4.7.1 Constructor & Destructor Documentation

4.7.1.1 World::~World ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.7.2 Member Function Documentation

4.7.2.1 Area * World::get_active_area ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.7.2.2 std::string World::get_author ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

4.7.2.3 std::string World::get_language ( )

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

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

- /home/cshome/m/mabrams/345/txtEngine/World.h
- /home/cshome/m/mabrams/345/txtEngine/World.cpp

Chapter 5

File Documentation

5.1 /home/cshome/m/mabrams/345/txtEngine/Area.cpp File - Reference

Source file for Area functionality.

```
#include "Area.h"
```

5.1.1 Detailed Description

Source file for Area functionality. Provides the functionality for an area in a game.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.2 /home/cshome/m/mabrams/345/txtEngine/Area.h File Reference

Defines the Area class.

```
#include <string> #include <vector> #include <iostream> ×
#include "Item.h" #include "StateDescriptor.h" #include "-
AreaCommand.h"
```

Classes

- class Area

5.2.1 Detailed Description

Defines the Area class. Area.h defines the methods for the Area.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.3 /home/cshome/m/mabrams/345/txtEngine/AreaCommand.cpp - File Reference

Source file for area command functionality.

```
#include "AreaCommand.h"
```

5.3.1 Detailed Description

Source file for area command functionality. Provides the functionality for an AreaCommand in the game.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.4 /home/cshome/m/mabrams/345/txtEngine/AreaCommand.h - File Reference

Defines the AreaCommand class.

```
#include <vector> #include <string>
```

Classes

- class AreaCommand

5.4.1 Detailed Description

Defines the AreaCommand class. AreaCommand.h defines the methods for the AreaCommand.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.5 /home/cshome/m/mabrams/345/txtEngine/combine.cpp File Reference

Source file for Combine functionality.

```
#include "combine.h" #include "Item.h"
```

5.5.1 Detailed Description

Source file for Combine functionality. Provides combine functionality in the game. An object consists of its id, the id of the first item that can be combined, and the id of the second object that can be combined.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.6 /home/cshome/m/mabrams/345/txtEngine/combine.h File - Reference

Defines the Combine class.

`#include <iostream>  #include <string>  #include "State-Descriptor.h"`

Classes

- class combine

5.6.1 Detailed Description

Defines the Combine class. Combine.h defines the methods for the Combine.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.7 /home/cshome/m/mabrams/345/txtEngine/Constants.h File - Reference

Defines the constants for the game.

Defines

- #define **DEFAULT_VALUE** "default_value"
- #define **MAX_CHARACTERS_PER_LINE** 80
- #define **WIN** "win"
- #define **DIE** "die"
- #define **NONE** "none"
- #define **LOOK** "look"
- #define **BAG** "bag"
- #define **GO** "go"
- #define **INVENTORY** "inventory"

- #define **QUIT** "quit"
- #define **NORTH** "north"
- #define **N** "n"
- #define **SOUTH** "south"
- #define **S** "s"
- #define **EAST** "east"
- #define **E** "e"
- #define **WEST** "west"
- #define **W** "w"
- #define **HELP** "help"
- #define **HELP_COMMAND** "Schrodinger says the cat is both dead and alive."
- #define **SAVE** "save"
- #define **LOAD** "load"
- #define **IGNORELIST** "input/ignorewords.txt"
- #define **IGNORELISTERROR** "\n\nERROR: Filter List not found!\n\n"
- #define **TOOMANYWORDS** "Please use fewer words for commands"
- #define **COMBINE** "combine"
- #define **PUT** "put"
- #define **STORE** "store"
- #define **MIX** "mix"
- #define **GARBAGE** "garbage"

5.7.1 Detailed Description

Defines the constants for the game.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.8 /home/cshome/m/mabrams/345/txtEngine/Item.cpp File Reference

Source file for Item functionality.

```
#include "Item.h" #include <iostream>
```

5.8.1 Detailed Description

Source file for Item functionality. Item.cpp provides the functionality for an Item in the game.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.9 /home/cshome/m/mabrams/345/txtEngine/Item.h File Reference

Defines the Item class.

```
#include <string>  #include <vector>  #include "State-
Descriptor.h" #include "ItemCommand.h" #include "combine.-
h"
```

Classes

- class Item

5.9.1 Detailed Description

Defines the Item class. Item.h defines the methods for the Item.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.10 /home/cshome/m/mabrams/345/txtEngine/ItemCommand.cpp File Reference

Source file for an ItemCommand.

`#include "ItemCommand.h"` `#include <iostream>`

5.10.1 Detailed Description

Source file for an ItemCommand. Provides the functionality for an ItemCommand.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.11 /home/cshome/m/mabrams/345/txtEngine/ItemCommand.h - File Reference

Defines the ItemCommand class.

`#include "Constants.h"` `#include <vector>` `#include <string>` ×

Classes

- class ItemCommand

5.11.1 Detailed Description

Defines the ItemCommand class. ItemCommand.h defines the methods for the ItemCommand.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.12 /home/cshome/m/mabrams/345/txtEngine/main.cpp File - Reference

The main file for txtEngine.

```
#include <iostream> #include <sstream> #include <fstream> ×
#include <algorithm> #include <string> #include "parser.-
h" #include "Constants.h"
```

Functions

- void gameloop ()

  The main gameloop.
- std::string one_word_command (std::string command)

  A method to handle one word commands.
- std::string two_word_command (std::string command1, std::string command2)

  A method to handle two word commands.
- std::string three_word_command (std::string command)
- void print_inventory ()
- std::string word_wrap (std::string input_string)
- void print_world_tree ()
- void load_game ()
- void save_game ()
- std::string input_filter (std::string input_string)
- void read_filter_list (std::string str)
- void **process_input** (std::string to_process, bool load)
- void **load** (char ∗const file)
- void read_filter_list (const char ∗file)
- int **main** (int argc, char ∗∗argv)

Variables

- World ∗ **world**
- bool **game_over** = false
- std::vector< std::string > **commandList**
- std::vector< std::string > **filterList**
- std::string **save**

5.12.1 Detailed Description

The main file for txtEngine. Main file for the game.

Open-source

Date

14/08/2011

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

Remarks

Parser code is freely distributed TinyXML library

5.12.2 Function Documentation

5.12.2.1 std::string input_filter (std::string *input_string*)

Checks the input string for words that are in the filterList vector. If they are in the list they are removed from the string.

Parameters

in	*input_string*	A string to be filtered

Returns

A string with words from filterList removed.

5.12.2.2 void load_game ( )

Loads a game from a .sav file.

5.12.2.3 std::string one_word_command (std::string *command*)

A method to handle one word commands.

Parameters

in	*command*	A single word command in the form of a string.

Returns

Output of the command.

5.12.2.4 void print_inventory ( )

Prints out the contents of the inventory vector.

5.12.2.5 void print_world_tree ( )

This method is used for debug purposes only: Prints out the parsed XML file in a tree structure.

5.12.2.6 void read_filter_list (std::string *str*)

Reads words from a specified file into the filterList vector.

Parameters

in	*str*	A string of a file path to a list of words to ignore.

5.12.2.7 void read_filter_list (const char * *file*)

Reads in words from file to filterList vector.

5.12.2.8 void save_game ( )

Saves a game to a .sav file by dumping the command list vector to a file.

5.12.2.9 std::string three_word_command (std::string *command*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.12.2.10 std::string two_word_command (std::string *command1*, std::string *command2*)

A method to handle two word commands.

Parameters

in	*command*	A two word command in the form of a string.

Returns

Output of the command.

5.12.2.11 std::string word_wrap (std::string *input_string*)

Wraps the output to a specified size.

Parameters

in	*input_string*	The output of the game to be wrapped

Returns

A wrapped string, properly formatted.

5.13 /home/cshome/m/mabrams/345/txtEngine/parser.cpp File - Reference

The source file for parser functionality.

```
#include "parser.h" #include "tinyxml.h"
```

Defines

- #define **WORLD_ATTRIBUTES** 3
- #define **AREA_ATTRIBUTES** 2
- #define **STATE_DESCRIPTION_ATTRIBUTES** 1
- #define **ITEM_ATTRIBUTES** 3
- #define **COMBINE_ATTRIBUTES** 3
- #define **PARSING_ERROR** 2
- #define **AREA_COMMAND_ATTRIBUTES** 2
- #define **ITEM_COMMAND_ATTRIBUTES** 5

- #define **INVALID** "invalid"
- #define **NONE** "none"
- #define **MISSING_TAGS** "missing tags"
- #define **UNDER_PARENT** "under tag with id: "
- #define **SEPERATOR** ","
- #define **INSIDE_INDEX** -1

Functions

- World * read_file (const char *pFilename, World *world)
- void **string_explode** (std::string str, std::string seperator, std::vector< std::string > *&result)
- combine * **make_combine** (TiXmlNode *pCommand, const char *parent_id, World *world)
- ItemCommand * make_item_command (TiXmlNode *pCommand, const char *parent_id, World *world)
- AreaCommand * make_area_command (TiXmlNode *pCommand, const char *parent_id, World *world)
- StateDescriptor * make_state_descriptor (TiXmlNode *pDescription, const char *parent_id, World *world)
- Item * **make_item** (TiXmlNode *pItem, const char *parent_id, World *world)
- Area * make_area (TiXmlNode *pArea, int area_index, World *world)
- World * make_world (TiXmlNode *pParent, World *world)
- void error_parsing (std::string message, World *world)
- World * make_objects (TiXmlNode *pParent, World *world)

5.13.1 Detailed Description

The source file for parser functionality. Turns XML game files into C++ objects.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.13.2 Function Documentation

5.13.2.1 void error_parsing (std::string *error_string,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.2 Area* make_area (TiXmlNode * *pArea*, int *area_index*, World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.3 AreaCommand* make_area_command (TiXmlNode * *pCommand*, const char * *parent_id*, World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.4 ItemCommand* make_item_command (TiXmlNode * *pCommand,* const char * *parent_id,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.5 World* make_objects (TiXmlNode * *pParent,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.6 StateDescriptor* make_state_descriptor (TiXmlNode * *pDescription,* const char * *parent_id,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.7 World* make_world (TiXmlNode * *pParent,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.13.2.8 World* read_file (const char * *pFilename,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14 /home/cshome/m/mabrams/345/txtEngine/parser.h File Reference

Defines the Area class.

```
#include <iostream> #include <sstream> #include <string> x
#include "tinyxml.h" #include "World.h"
```

Functions

- void error_parsing (std::string error_string, World *world)
- ItemCommand * make_item_command (TiXmlNode *pCommand, const char *parent_id, World *world)
- AreaCommand * make_area_command (TiXmlNode *pCommand, const char *parent_id, World *world)
- StateDescriptor * make_state_descriptor (TiXmlNode *pDescription, const char *parent_id, World *world)
- Area * make_area (TiXmlNode *pArea, int area_index, World *world)
- World * make_world (TiXmlNode *pParent, World *world)
- World * make_objects (TiXmlNode *pParent, World *world)
- World * read_file (const char *pFilename, World *world)
- combine * **make_combine** (TiXmlNode *pCommand, const char *parent_id, - World *world)
- Item * **make_item** (TiXmlNode *pItem, const char *parent_id, World *world)

5.14.1 Detailed Description

Defines the Area class. Area.h defines the methods for the Area.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.14.2 Function Documentation

5.14.2.1 void error_parsing (std::string *error_string,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.2 Area* make_area (TiXmlNode * *pArea,* int *area_index,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.3 AreaCommand* make_area_command (TiXmlNode * *pCommand,* const char * *parent_id,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.4 ItemCommand* make_item_command (TiXmlNode * *pCommand,* const char * *parent_id,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.5 World* make_objects (TiXmlNode * *pParent,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.6 StateDescriptor* make_state_descriptor (TiXmlNode * *pDescription,* const char * *parent_id,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.7 World* make_world (TiXmlNode * *pParent,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.14.2.8 World* read_file (const char * *pFilename,* World * *world*)

Write description of function here. The function should follow these comments. Use of "brief" tag is optional. (no point to it)

The function arguments listed with "param" will be compared to the declaration and verified.

Parameters

in	*_inArg1*	Description of first function argument.
out	*_outArg2*	Description of second function argument.
in,out	*_inoutArg3*	Description of third function argument.

Returns

Description of returned value.

5.15 /home/cshome/m/mabrams/345/txtEngine/StateDescriptor.cpp File Reference

Source file for a StateDescriptor.

```
#include "StateDescriptor.h"
```

5.15.1 Detailed Description

Source file for a StateDescriptor. Provides functionality for a StateDescriptor object.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.16 /home/cshome/m/mabrams/345/txtEngine/StateDescriptor.h File Reference

Dscribes the StateDescriptor class.

```
#include <string>
```

Classes

- class StateDescriptor

5.16.1 Detailed Description

Dscribes the StateDescriptor class. Area.h defines the methods for StateDescriptor.cpp

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.17 /home/cshome/m/mabrams/345/txtEngine/World.cpp File Reference

Source file for a World.

```
#include "World.h"
```

5.17.1 Detailed Description

Source file for a World. World.cpp provides the functionality for the game world.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.18 /home/cshome/m/mabrams/345/txtEngine/World.h File Reference

Defines the World class.

```
#include "Area.h" #include <string> #include <vector>
```

Classes

- class World

5.18.1 Detailed Description

Defines the World class. World.h defines the methods for the World.cpp source file.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3