txtEngine

v4.0

Generated by Doxygen 1.7.5.1

Fri Sep 30 2011 15:28:14

Contents

Chapter 1

Documentation for the txtEngine Project

1.1 Date Updated:

29-11-2011

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 Documentation:

To view the txtEngine Language Specification and Documentation click here: txt-Engine Documentation

1.4 Links:

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

1.5 Report Bugs:

Please report any bugs here: `https://github.com/smilefreak/txt-Engine/issues`

1.6 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, const char ∗name)
- ∼Area ()
- std::string get_area_name ()

Protected Attributes

- std::vector< Item ∗ > items

  A vector to hold the area's items.
- int num_descriptions

  The number of descriptions for the area.
- int num_commands

  The number of commands for the area.
- int num_items

  The number of items in the area.
- std::string status

  The status of the area.
- std::string id

  The area's id.
- std::string name

  The name of the area.
- std::string curr_desc_id

  The id of the area's current description.
- std::vector< StateDescriptor ∗ > description
- std::vector< AreaCommand ∗ > commands

  A vector of all the commands for the area.

4.1.1 Constructor & Destructor Documentation

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

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.
in	*name*	The name 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	*command_-name*	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 std::string Area::get_area_name ( )

Gets the name of the area.

Returns

The name of the area.

4.1.2.5 **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.

4.1.2.6 std::string Area::get_description ( )

Get the area description.

Returns

String description of the area.

4.1.2.7 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.

4.1.2.8 std::string Area::get_id ( )

Get the area id.

Returns

The id of the area.

4.1.2.9 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.

4.1.2.10 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`	*item*	The address of the item's index member variable.

Returns

A pointer to the item.

4.1.2.11 int Area::get_num_commands ( )

Get the number of commands for this area.

Returns

The number of AreaCommands for this area.

4.1.2.12 int Area::get_num_descriptions ( )

Get the number if descriptions for the area.

Returns

The number of descriptions for this area.

4.1.2.13 int Area::get_num_items ( )

Accessor for the number of items in the Area.

Returns

The number of items in the area.

4.1.2.14 std::string Area::get_status ( )

This accessor method returns the status of the area.

Returns

The status of the Area.

4.1.2.15 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.

4.1.2.16 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.

4.1.2.17 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.

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

Checks whether the area has an item.

Parameters

in	*item_to_find*	The name of the item to find.

Returns

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

4.1.2.19 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.

4.1.2.20 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.

4.1.2.21 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.

4.1.3 Member Data Documentation

4.1.3.1 std::vector<StateDescriptor*> Area::description [protected]

A vector of all the descriptions of the area

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

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

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

  Flag, whether this area command is locked.
- std::string name

  The name of this area command.
- std::string status

  The status of this area command.
- std::string message

  The message displayed when area command called.
- std::string depends

  What the area command depends on.
- std::string move_to_area

  New area when area command called.
- std::vector< std::string > ∗ synonyms

  Vector of synonyms for area command.

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	*areato-moveto*	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 the item that is a combination.

Returns

A pointer to the combined item.

4.3.2.2 std::string combine::get_description ( )

Gets the description of the combined item.

Returns

The description of the combined item.

4.3.2.3 std::string combine::get_first_id ( )

Get the id of the first item that made this combined item.

Returns

The id of the first item.

4.3.2.4 std::string combine::get_id ( )

Get the id of the combined item.

Returns

The id of the combined item.

4.3.2.5 std::string combine::get_second_id ( )

Get the id of the second item that made this combined item.

Returns

The id of the second item.

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

Sets the combination class member to a new item.

Parameters

in	*item*	A pointer to an item that is a combination of two items from inventory.

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

Sets the description of the combined item.

Parameters

in	*d*	The description for the item.

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 ∗new_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, const char ∗name)
- ∼Item ()
- int get_num_items ()
- Item ∗ get_item (int index)
- std::string get_name ()

Protected Attributes

- bool collectable

  Flag, whether item can be collected.
- int num_descriptions

  Number of descriptions for the item.
- int num_commands

  Number of commands for this item.
- int num_items

  Number of items contained inside this item.
- std::string id

  The item's id.
- bool container

  Whether or not item is a container.
- bool locked

  Whether or not the item is locked.
- combine ∗ combine_var

  Pointer to a combine object.
- std::vector< Item ∗ > contains

  Vector of items contained in this item.
- std::string curr_desc_id

  The current description of this item.
- std::vector< StateDescriptor ∗ > description

  Vector of descriptions.
- std::vector< ItemCommand ∗ > commands

  Vector of commands for item.
- std::vector< std::string > ∗ synonyms

  Vector of synonyms for the item.
- std::string depends

What this item depends on (key).

- std::string name

  The name of the item.

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,* const char $*$ *name*)

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.
in	*name*	The name of the item.

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

`in`	*new_item*	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 Item ∗ Item::get_item (int *index*)

Get the item inside this item by index.

Parameters

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

Returns

Pointer to an item.

4.4.2.15 std::string Item::get_name ( )

Get the name of the item.

Returns

The name of the item.

4.4.2.16 int Item::get_num_commands ( )

Gets the number of commands for this item.

Returns

The number of commands this item has.

4.4.2.17 int Item::get_num_descriptions ( )

Get the number of descriptions for this item.

Returns

The number of descriptions for this item.

4.4.2.18 int Item::get_num_items ( )

Get the number of items inside this item.

Returns

The number of items inside this item.

4.4.2.19 bool Item::has_combine ( )

Checks whether this Item can combine with another.

Returns

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

4.4.2.20 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.

4.4.2.21 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.

4.4.2.22 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.

4.4.2.23 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.

4.4.2.24 bool Item::is_collectable ( )

Checks whether this item is collectable.

Returns

Description of returned value.

4.4.2.25 bool Item::is_locked ( )

Checks whether the the Item is locked.

Returns

True if the item is locked otherwise false.

4.4.2.26 std::string Item::print_contained_items ( )

Returns a string with all items the item contains.

Returns

A string of items this item contains.

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

Removes an item from inside this item by id.

Parameters

`in`	*item_id*	A string - the id of the item to remove.

4.4.2.28 void Item::set_combine (combine ∗ *c*)

A mutator for a combine object.

Parameters

in	*c*	A pointer to a combine object.

4.4.2.29 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

- 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)
- ∼ItemCommand ()
- 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 is_combine ()
- bool unlocks ()
- std::string unlock_areacommand_string ()
- std::string unlock_area_string ()
- std::string get_unlock_string ()

Protected Attributes

- std::string name

  The name of the item command.
- std::string state_change

  The state of the item after command called.
- std::string message

  A message to display when the command is called.
- std::string area_change

  The area of the item after using the command.
- std::string depends

  What this command depends on to be used.
- std::string status

  The commands status.
- std::string unlock

  Unlock string for command.
- bool call_combine

  Whether this command combines two items.
- std::vector< std::string > * synonyms

  Vector of synonyms for this command.
- bool change_collect

  If command makes item collectable.
- bool collect_dependent

  If command requires item to be in inventory.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 ItemCommand::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*)

The constructor for an ItemCommand object.

Parameters

`in`	*callmeby*	The command.
`in`	*state_-mutator*	State to change to.
`in`	*chng_collec*	True if command changes item to collectable.
`in`	*collec_dep*	Whether command depends on item being collected.
`in`	*area_chng*	New area for item.
`in`	*status_-command*	Status of item after command called.
`in`	*depends*	What item in inventory command depends on.
`in`	*synonyms*	Synonyms to the command.
`in`	*unlock*	What the command unlocks.
`in`	*combine*	A boolean - true if command combines items.

4.5.1.2 ItemCommand::∼ItemCommand ( )

The destructor for an ItemCommand object.

4.5.2 Member Function Documentation

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

Check if the item command has any synonyms.

Returns

True if a synonym for this command exists otherwise false.

4.5.2.2 std::string ItemCommand::get_area_change ( )

Get the id of the area the item changes to when this command is called.

Returns

The id of the area to change to.

4.5.2.3 bool ItemCommand::get_change_collect ( )

Check whether this item command makes the item collectable.

Returns

True if the item command changes whether the item is collectable otherwise false.

4.5.2.4 bool ItemCommand::get_collect_dependent ( )

Check whether this item command depends on having an item in inventory.

Returns

True if the command depends on possessing an item.

4.5.2.5 std::string ItemCommand::get_depends ( )

Get the id of an item that this ItemCommand depends on.

Returns

The id of an item that this command depends on.

4.5.2.6 std::string ItemCommand::get_message ( )

Get the message - this is displayed automatically when the item command is called.

Returns

A message to display.

4.5.2.7 std::string ItemCommand::get_name ( )

Get the name of the item command.

Returns

The name of the item command.

4.5.2.8 std::string ItemCommand::get_state_change ( )

Get the state the item will change to when this command is called.

Returns

The state for item to change to.

4.5.2.9 std::string ItemCommand::get_status ( )

Get the status of the item command.

Returns

The status of the command.

4.5.2.10 std::string ItemCommand::get_unlock_string ( )

Get the unlock string.

Returns

The unlock string.

4.5.2.11 bool ItemCommand::is_combine ( )

Checks whether this item command will combine two items.

Returns

True if the command will combine items, otherwise false.

4.5.2.12 void ItemCommand::set_message (const char $*$ *to_message*)

Sets the message to be displayed when the command is called.

Parameters

in	*to_message*	A string - the message to be displayed.

4.5.2.13 std::string ItemCommand::unlock_area_string ( )

Gets the area this item command unlocks.

Returns

An area.

4.5.2.14 std::string ItemCommand::unlock_areacommand_string ( )

Gets the areacommand this item command unlocks.

Returns

An areacommand.

4.5.2.15 bool ItemCommand::unlocks ( )

Checks whether this item command will unlock an item.

Returns

True if the command unlocks an item.

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

  The id of the state descriptor.
- std::string description

  The description.

4.6.1 Constructor & Destructor Documentation

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

Constructor for a state descriptor.

Parameters

`in`	*identifier*	The identifier of the descriptor.

4.6.1.2 StateDescriptor::$\sim$StateDescriptor ( )

StateDescriptor Destructor.

4.6.2 Member Function Documentation

4.6.2.1 std::string StateDescriptor::get_description ( )

Gets the description of the object.

Returns

The description of the object.

4.6.2.2 std::string StateDescriptor::get_id ( )

Gets the id of the object.

Returns

The id of the object.

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

Sets the description variable of the object.

Parameters

`in`	*desc*	The description of the object.

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

  The language the game is written in.
- std::string author

  The author(s) of the game.
- std::vector< Area ∗ > areas

  All the areas in the game.
- std::string initial_area

  The starting area.
- int num_areas

  How many areas in the game.
- Area ∗ active_area

  The area that is currently active.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 World::World (const char ∗ *lang,* const char ∗ *auth,* const char ∗ *init_area*)

The constructor for a world object.

Parameters

in	*lang*	The name of the language for the game.
in	*auth*	The author(s) of the game.
in	*init_area*	The initial area for the game.

4.7.1.2 World::∼World ( )

The deconstructor for the World object.

4.7.2 Member Function Documentation

4.7.2.1 void World::add_area (Area ∗ *new_area*)

Adds an area to the world.

Parameters

in	*new_area*	A pointer to an area object

4.7.2.2 void World::change_area (std::string *name*)

Sets the active area to the specified area.

Parameters

in	*name*	An id of an area to set as active.

4.7.2.3 Area ∗ World::get_active_area ( )

Gets the active area.

Returns

A pointer to the active area in the game world.

4.7.2.4 Area ∗ World::get_area (int *index*)

Gets an area from the areas vector by index.

Parameters

in	*index*	The index of an area pointer in the vector.

Returns

A pointer to an area at the index given.

4.7.2.5 Area ∗ World::get_area (std::string *area_id*)

Gets an area from the areas vector by area id.

Parameters

in	*area_id*	The id of an area.

Returns

A pointer to an area with the given id.

4.7.2.6 std::string World::get_author ( )

Gets the author of the game specified in the world tag of the game.

Returns

A string - the author of the game.

4.7.2.7 std::string World::get_language ( )

Gets the language specified in the world tag of the game.

Returns

A string - the language the game is written in.

4.7.2.8 int World::get_num_areas ( )

Gets the number of areas in the world.

Returns

The number of areas in the world.

4.7.2.9 bool World::init_active_area ( )

Sets the initial area in the areas vector to the active area.

Returns

True if an initial area is found in the areas vector otherwise false.

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.h File Reference

Defines the Area class.

```
#include <string> #include <vector> #include <iostream>
#include <cstring>  #include "Item.h"  #include "State-
Descriptor.h" #include "AreaCommand.h"
```

Classes

- class Area

5.1.1 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

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

Source file for area command functionality.

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

5.2.1 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

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

Defines the AreaCommand class.

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

Classes

- class AreaCommand

5.3.1 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

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

Source file for Combine functionality.

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

5.4.1 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

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

Defines the Combine class.

`#include <iostream>` `#include <string>` `#include "StateDescriptor.h"`

Classes

- class combine

5.5.1 Detailed Description

Defines the Combine class. 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/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"

5.6.1 Detailed Description

Defines the constants for the game.

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.6.2 Define Documentation

5.6.2.1 #define BAG "bag"

Define bag command value.

5.6.2.2 #define COMBINE "combine"

Define combine command value.

5.6.2.3 #define DEFAULT_VALUE "default_value"

The default value for a tag.

5.6.2.4 #define DIE "die"

Define die tag value.

5.6.2.5 #define E "e"

Define e command value.

5.6.2.6 #define EAST "east"

Define east command value.

5.6.2.7 #define GARBAGE "garbage"

Define garbage tag value.

5.6.2.8 #define GO "go"

Define go command value.

5.6.2.9 #define HELP "help"

Define help command value.

5.6.2.10 #define HELP_COMMAND "Schrodinger says the cat is both dead and alive."

Defines the help command message.

5.6.2.11 #define IGNORELIST "input/ignorewords.txt"

Define path to the ignore list.

5.6.2.12 #define IGNORELISTERROR "\n\nERROR: Filter List not found!\n\n"

Define error message when the ignore list is not found.

5.6.2.13 #define INVENTORY "inventory"

Define inventory command value.

5.6.2.14 #define LOAD "load"

Define load command value.

5.6.2.15 #define LOOK "look"

Define look command value.

5.6.2.16 #define MAX_CHARACTERS_PER_LINE 80

The maximum characters per line for terminal output.

5.6.2.17 #define MIX "mix"

Define mix command value.

5.6.2.18 #define N "n"

Define n command value.

5.6.2.19 #define NONE "none"

Define none tag value.

5.6.2.20 #define NORTH "north"

Define north command value.

5.6.2.21 #define PUT "put"

Define put command value.

5.6.2.22 #define QUIT "quit"

Define quit command value.

5.6.2.23 #define S "s"

Define s command value.

5.6.2.24 #define SAVE "save"

Define save command value.

5.6.2.25 #define SOUTH "south"

Define south command value.

5.6.2.26 #define STORE "store"

Define store command value.

5.6.2.27 #define TOOMANYWORDS "Please use fewer words for commands"

Define error message when commands have too many words.

5.6.2.28 #define W "w"

Define w command value.

5.6.2.29 #define WEST "west"

Define west command value.

5.6.2.30 #define WIN "win"

Define win tag value.

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

Source file for Item functionality.

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

5.7.1 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

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

Defines the Item class.

```
#include <string> #include <vector> #include <cstring>
#include "StateDescriptor.h"  #include "ItemCommand.h" ×
#include "combine.h"
```

Classes

- class Item

5.8.1 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

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

Source file for an ItemCommand.

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

5.9.1 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

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

Defines the ItemCommand class.

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

Classes

- class ItemCommand

5.10.1 Detailed Description

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

Author

Michael Abrams
James Boocock
Toby Herbert
Tatai Nikora

Version

0.3

5.11 /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 (char ∗const file)
- void save (char ∗const file)
- 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 read_filter_list (const char ∗file)
- std::string get_all_area_commands ()
- std::string valid_item_command_inv (Item ∗temp_item, int item)
- std::string valid_item_command_area (Item ∗temp_item, int items)
- std::string get_all_item_commands ()
- void external_output (std::string command)
- int main (int argc, char ∗∗argv)

Variables

- World ∗ world

  The world object.
- bool game_over = false

  Flag to end a game.
- std::vector< std::string > commandList

  List of commands used.
- std::vector< std::string > filterList

  List of words to filter from input.

5.11.1 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

5.11.2 Function Documentation

5.11.2.1 void external_output (std::string *command*)

External output method used by web interface.

Parameters

`in`	*command*	Input commands to be processed.

5.11.2.2 std::string get_all_area_commands ( )

Get all of the area commands for the current area.

Returns

A string containing all of the area commands.

5.11.2.3 std::string get_all_item_commands ( )

Reads words from a specified file into the filterList vector.

Returns

A string of a file path to a list of words to ignore.

5.11.2.4 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.

5.11.2.5 void load (char *const *file*)

Loads a game from a .sav file.

5.11.2.6 int main (int *argc*, char ∗∗ *argv*)

The main method of the program.

Parameters

`in`	*argc*	The number of command line args.
`in`	*argv*	Char∗ array of command line args.

Returns

An integer.

5.11.2.7 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.

5.11.2.8 void print_inventory ( )

Prints out the contents of the inventory vector.

5.11.2.9 void print_world_tree ( )

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

5.11.2.10 void process_input (std::string *to_process*, bool *load*)

Processes input and calls the appropriate method for the number of words in the command. Commands are stored in the commandList and if too many words are in the string a feedback message is output. If the game is in load mode then no output is displayed.

Parameters

`in`	*to_process*	An input string to process.
`in`	*load*	A flag for whether the game is in load mode.

5.11.2.11 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.

5.11.2.12 void read_filter_list (const char ∗ *file*)

Reads words from a specified file into the filterList vector.

Parameters

`in`	*file*	A char array - the file path to a list of words to ignore.

5.11.2.13 void save (char ∗const *file*)

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

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

A method for handling three word commands. For example: combine pen paper.

Parameters

`in`	*command*	The three word command.

Returns

A string to output.

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

A method to handle two word commands.

Parameters

`in`	*command1*	First command in the form of a string.
`in`	*command2*	Second command in the form of a string.

Returns

Output of the command.

5.11.2.16 std::string valid_item_command_area (Item * *temp_item,* int *items*)

Gets the valid area commands for items in the current area.

Parameters

in	*temp_item*	A pointer to the item.
in	*items*	Number of items.

Returns

The valid item commands for items in the area.

5.11.2.17 std::string valid_item_command_inv (Item * *temp_item,* int *item*)

Gets all valid item commands for an inventory item.

Parameters

in	*temp_item*	A pointer to an item.
in	*item*	An integer.

Returns

The valid item commands.

5.11.2.18 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.

5.12 /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)

5.12.1 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

5.12.2 Function Documentation

5.12.2.1 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	*error_string*	An error message to be displayed.
in	*world*	A pointer to THE world object.

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

Creates an Area object from XML.

Parameters

in	*pArea*	Pointer to a TinyXML Node.
in	*area_index*	The index of the area inside world's vector of areas.
in	*world*	A pointer to THE world object.

Returns

An Area object.

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

Creates an AreaCommand object from XML.

Parameters

in	*pCommand*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

An AreaCommand object.

5.12.2.4 combine∗ make_combine (TiXmlNode ∗ *pCommand*, const char ∗ *parent_id*, World ∗ *world*)

Creates a combine object for the game.

Parameters

in	*pCommand*	Pointer to a TinyXML node.
in	*parent_id*	The id of the parent node.
in	*world*	Pointer to the world object

Returns

A combine object.

5.12.2.5 Item∗ make_item (TiXmlNode ∗ *pItem*, const char ∗ *parent_id*, World ∗ *world*)

Creates an item ovject for the game.

Parameters

in	*pItem*	Pointer to a TinyXML node.
in	*parent_id*	The id of the parent node.
in	*world*	Pointer to the world object.

Returns

An Item object.

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

Creates an item command object from XML.

Parameters

in	*pCommand*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

An ItemCommand object.

5.12.2.7 World∗ make_objects (TiXmlNode ∗ *pParent,* World ∗ *world*)

Starts making the objects for the game.

Parameters

in	*pParent*	Pointert to a TinyXML Node.
in	*world*	Pointer to THE world object.

Returns

A World object.

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

Creates a StateDescriptor object from XML.

Parameters

in	*pDescription*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

A StateDescriptor object.

5.12.2.9 World∗ make_world (TiXmlNode ∗ *pParent,* World ∗ *world*)

Creates a World object from XML.

Parameters

in	*pParent*	Pointer to a TinyXML Node.
in	*world*	A pointer to THE world object.

Returns

A World object.

5.12.2.10 World∗ read_file (const char ∗ *pFilename,* World ∗ *world*)

Method to handle reading in the XML game file.

Parameters

in	*pFilename*	Path and file name of the game file to read.
in	*world*	A pointer to a world object.

Returns

A world object.

5.12.2.11 void string_explode (std::string *str*, std::string *seperator*, std::vector< std::string > ∗& *result*)

Formats the output to wrap correctly.

Parameters

in	*str*	The string to be formatted.
in	*seperator*	Separator to break the string by.
out	*result*	Pointer to a vector of strings.

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

Defines the parser class.

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

Functions

- void string_explode (std::string str, std::string seperator, std::vector< std::string > ∗&result)
- 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)

5.13.1 Detailed Description

Defines the parser class. Parser.h defines the methods for the Parser.cpp source file.

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`	*error_string*	An error message to be displayed.
`in`	*world*	A pointer to THE world object.

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

Creates an Area object from XML.

Parameters

`in`	*pArea*	Pointer to a TinyXML Node.
`in`	*area_index*	The index of the area inside world's vector of areas.
`in`	*world*	A pointer to THE world object.

Returns

An Area object.

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

Creates an AreaCommand object from XML.

Parameters

in	*pCommand*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

An AreaCommand object.

5.13.2.4 combine∗ make_combine (TiXmlNode ∗ *pCommand,* const char ∗ *parent_id,* World ∗ *world*)

Creates a combine object for the game.

Parameters

in	*pCommand*	Pointer to a TinyXML node.
in	*parent_id*	The id of the parent node.
in	*world*	Pointer to the world object

Returns

A combine object.

5.13.2.5 Item∗ make_item (TiXmlNode ∗ *pItem,* const char ∗ *parent_id,* World ∗ *world*)

Creates an item ovject for the game.

Parameters

in	*pItem*	Pointer to a TinyXML node.
in	*parent_id*	The id of the parent node.
in	*world*	Pointer to the world object.

Returns

An Item object.

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

Creates an item command object from XML.

Parameters

in	*pCommand*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

An ItemCommand object.

5.13.2.7 World∗ make_objects (TiXmlNode ∗ *pParent*, World ∗ *world*)

Starts making the objects for the game.

Parameters

in	*pParent*	Pointert to a TinyXML Node.
in	*world*	Pointer to THE world object.

Returns

A World object.

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

Creates a StateDescriptor object from XML.

Parameters

in	*pDescription*	Pointer to a TinyXML Node.
in	*parent_id*	A pointer to the parent id of the parent node.
in	*world*	A pointer to THE world object.

Returns

A StateDescriptor object.

5.13.2.9 World∗ make_world (TiXmlNode ∗ *pParent*, World ∗ *world*)

Creates a World object from XML.

Parameters

in	*pParent*	Pointer to a TinyXML Node.
in	*world*	A pointer to THE world object.

Returns

A World object.

5.13.2.10 World∗ read_file (const char ∗ *pFilename*, World ∗ *world*)

Method to handle reading in the XML game file.

Parameters

`in`	*pFilename*	Path and file name of the game file to read.
`in`	*world*	A pointer to a world object.

Returns

A world object.

5.13.2.11 void string_explode (std::string *str*, std::string *seperator*, std::vector< std::string > ∗& *result*)

Formats the output to wrap correctly.

Parameters

`in`	*str*	The string to be formatted.
`in`	*seperator*	Separator to break the string by.
`out`	*result*	Pointer to a vector of strings.

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

Source file for a StateDescriptor.

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

5.14.1 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

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

Dscribes the StateDescriptor class.

```
#include <string>
```

Classes

- class StateDescriptor

5.15.1 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

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

Source file for a World.

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

5.16.1 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

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

Defines the World class.

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

Classes

- class World

5.17.1 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