Praxisprojekt Code Dokumentation

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

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

Namespace Documentation

4.1 badthings Namespace Reference

Namespace containing all bad things functions. These can be assigned to a specific card.

Functions

- BadThingsRetVal emptyBadThings (GameState &gamestate, const MunchkinCard &card)
  Does nothing (placeholder)
- BadThingsRetVal looseHand (GameState &gamestate, const MunchkinCard &card)
  Player looses 1 hand and the function checks whether or not he can still carry all his items that need hands.
- BadThingsRetVal looseClass (GameState &gamestate, const MunchkinCard &card)
  Player looses his class.
- BadThingsRetVal playerDies (GameState &gamestate, const MunchkinCard &card)
  Player dies.
- BadThingsRetVal looseLevel (GameState &gamestate, const MunchkinCard &card)
  Player looses 1 level.
- BadThingsRetVal maleDeadFemaleLevelDown (GameState &gamestate, const MunchkinCard &card)
  Male Players die. Female players loose 1 level.

4.1.1 Detailed Description

Namespace containing all bad things functions. These can be assigned to a specific card.

4.1.2 Function Documentation

4.1.2.1 emptyBadThings()

```
BadThingsRetVal badthings::emptyBadThings (
            GameState & gamestate,
            const MunchkinCard & card )
```

Does nothing (placeholder)

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.1.2.2 looseClass()

```
BadThingsRetVal badthings::looseClass (
            GameState & gamestate,
            const MunchkinCard & card )
```

Player looses his class.

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.1.2.3 looseHand()

```
BadThingsRetVal badthings::looseHand (
            GameState & gamestate,
            const MunchkinCard & card )
```

Player looses 1 hand and the function checks whether or not he can still carry all his items that need hands.

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.1.2.4 looseLevel()

```
BadThingsRetVal badthings::looseLevel (
            GameState & gamestate,
            const MunchkinCard & card )
```

Player looses 1 level.

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.1.2.5 maleDeadFemaleLevelDown()

```
BadThingsRetVal badthings::maleDeadFemaleLevelDown (
            GameState & gamestate,
            const MunchkinCard & card )
```

Male Players die. Female players loose 1 level.

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.1.2.6 playerDies()

```
BadThingsRetVal badthings::playerDies (
            GameState & gamestate,
            const MunchkinCard & card )
```

Player dies.

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2 cardtypeaction Namespace Reference

Namespace containing all cardtype functions. These are mapped to the actual card type.

Functions

- CardTypeRetVal curse (GameState &gamestate, const MunchkinCard &card)
  processes card type curse
- CardTypeRetVal joker (GameState &gamestate, const MunchkinCard &card)
  processes card type joker
- CardTypeRetVal monster (GameState &gamestate, const MunchkinCard &card)
  proceses card type monster
- CardTypeRetVal munchClass (GameState &gamestate, const MunchkinCard &card)
  processes card Type munchkin class
- CardTypeRetVal race (GameState &gamestate, const MunchkinCard &card)

processes card type munchkin race

- CardTypeRetVal item (GameState &gamestate, const MunchkinCard &card)

  processes card type item
- CardTypeRetVal itemBuff (GameState &gamestate, const MunchkinCard &card)

  processes card type item buff
- CardTypeRetVal lvlUp (GameState &gamestate, const MunchkinCard &card)

  processes card type level up

4.2.1 Detailed Description

Namespace containing all cardtype functions. These are mapped to the actual card type.

4.2.2 Function Documentation

4.2.2.1 curse()

```
CardTypeRetVal cardtypeaction::curse (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type curse

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.2 item()

```
CardTypeRetVal cardtypeaction::item (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type item

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.3 itemBuff()

```
CardTypeRetVal cardtypeaction::itemBuff (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type item buff

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.4 joker()

```
CardTypeRetVal cardtypeaction::joker (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type joker

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.5 lvlUp()

```
CardTypeRetVal cardtypeaction::lvlUp (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type level up

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.6 monster()

```
CardTypeRetVal cardtypeaction::monster (
```

```
            GameState & gamestate,
            const MunchkinCard & card )
```

proceses card type monster

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.7 munchClass()

```
CardTypeRetVal cardtypeaction::munchClass (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card Type munchkin class

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

4.2.2.8 race()

```
CardTypeRetVal cardtypeaction::race (
            GameState & gamestate,
            const MunchkinCard & card )
```

processes card type munchkin race

Parameters

gamestate	The game state is modified according to the corresponding munchkin rule/card.
card	Used to supply additional information about the card.

Chapter 5

Class Documentation

5.1 BadThingsRetVal Struct Reference

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/BadThings.h

5.2 Button Class Reference

class for Buttons

`#include <Button.h>`

Public Member Functions

- Button (int _id, const cv::Rect &_rect, const cv::Scalar &_color, bool _visible=true, ButtonOrigin _origin=ButtonOrigin::topleft, const button_callback &_callback={})

  constructor for a button
- Button (const Button &)=default

  Default copy constructor.
- Button (Button &&)=default

  Default move constructor.
- Button & operator= (const Button &)=default

  Default copy assignment.
- Button & operator= (Button &&)=default

  Default move assignment.
- bool poll_click (const cv::Point &point, const cv::Size &canvas_size) const

  function to check if click was inside the button
- void draw (cv::Mat &canvas) const

  draw function for buttons

Public Attributes

- int id

  Button id.
- cv::Rect rect

  rectangle that defines size of button
- button_callback callback

  callback function for button to apply action to button
- cv::Scalar color

  color of the button
- bool visible

  wether or not button should be visible
- ButtonOrigin origin

  position of the button

5.2.1 Detailed Description

class for Buttons

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Button()

```
Button::Button (
            int _id,
            const cv::Rect & _rect,
            const cv::Scalar & _color,
            bool _visible = true,
            ButtonOrigin _origin = ButtonOrigin::topleft,
            const button_callback & _callback = {} )
```

constructor for a button

Parameters

_id	Button id
_rect	rectangle which defines the dimensions of the button
_color	defines the color of the button in GBR
_visible	used to hide button when not needed
_origin	origin of the button as in enum class ButtonOrigin
_callback	callback function for button. Can be changed so that one button can serve mulitple purposes if necessary

5.2.3 Member Function Documentation

5.2.3.1 draw()

```
void Button::draw (
            cv::Mat & canvas ) const
```

draw function for buttons

Parameters

canvas	defines the button that should be drawn

5.2.3.2 poll_click()

```
bool Button::poll_click (
            const cv::Point & point,
            const cv::Size & canvas_size ) const
```

function to check if click was inside the button

Parameters

point	point of click
canvas_size	size of the canvas that defines the button size

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/Button.h
- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/Button.cpp

5.3 CardTypeRetVal Struct Reference

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/CardTypeActions.h

5.4 ExtrasRetVal Struct Reference

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/Extras.h

5.5 GameState Struct Reference

`#include <GameState.h>`

Public Attributes

- PlayerStats player01

  playerStats object for data of player stats that are needed
- MouseParams mouseparams

  mouseParams object for data of tutorial that are needed e.g. tutorial text
- std::vector< Button > buttons

  buttons that are needed for the tutorial
- bool should_exit

  param wether or not the game should exit after exit button was pressed
- bool should_continue

  param for when user input is awaited and the tutorial needs to jump to a different point in logic function
- bool end_turn

  param to signal the player that the end of the turn is reached
- bool run_away

  param for when the user needs to run away and the tutorial triggers an random number as dice roll
- bool remove_card

  param for when the player chooses or needs to get rid of a card he has equiped
- cv::Size canvas_size

  param of the canvas size used for the buttons

5.5.1 Detailed Description

data of all needed params for the tutorial

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h

5.6 InputEvent Struct Reference

Public Attributes

- EventType **type**

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/InputEvent.h

5.7 MouseParams Struct Reference

Holds data of a mouse event.

`#include <GameState.h>`

Public Attributes

- vector< cv::Point > poly
  Polygon of the clicked marker.
- int markerId
  Id of the marker.
- vector< cv::Point2f > markerCorner
  Four corners of the marker.
- vector< string > tutText
  Tutorial text to display for the marker.
- cv::Point clickP
  Mouse position.
- MouseEvent event
  Mouse event type.

5.7.1 Detailed Description

Holds data of a mouse event.

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h

5.8 MunchkinCard Class Reference

class of munchkin cards with params that could be needed for a single card

`#include <MunchkinCards.h>`

Public Member Functions

- MunchkinCard ()
  Default constructor.
- MunchkinCard (int _markerID, const string &_cardName, const string &_effect, const string &_badThings, const string &_itemEffect, const string &_itemNeeds, vector< string > _bonis, ParentCardType _parent↩CardType, CardType _type, ItemType _itemType, int _strengthBoni, int _debuff, int _monsStrength, int _lvlUp, int _treasures, int _itemValue, int _handsNeeded, bool _itemLarge)
  custom constructor
- MunchkinCard (const MunchkinCard &other)=default
  default copy constructor
- MunchkinCard (MunchkinCard &&other)=default
  default move constructor
- MunchkinCard & operator= (const MunchkinCard &other)=default
  default copy assignment
- MunchkinCard & operator= (MunchkinCard &&other)=default
  default move assignment

Static Public Member Functions

- static vector< MunchkinCard > cardsConstr ()

  function to create all cards from MunchkinCards.cpp

Public Attributes

- string cardName

  card name
- string effect

  card effect
- vector< string > bonis

  boni that the card can have
- string badThings

  (deprecated) string of bad things
- string itemEffect

  effect that an item has e.g. attack with fire and flame
- string itemNeeds

  param that a munchkin needs to equip a item e.g. has to be a dwarf
- ParentCardType parentCardType

  param for parent card type
- CardType type

  param for "child" card type
- ItemType itemType

  item type
- int markerID

  marker id that corresponse with the card
- int strengthBoni

  how much strength the munchkin receives with this card
- int debuff

  how much strength the munchkin looses through this card
- int monsStrength

  param how much the monsters strength is
- int lvlUp

  how many level you get r.g. when defeating the monster
- int treasures

  how many treasure the munchkin gets when defeating the monster
- int itemValue

  how much gold the item sells for
- int handsNeeded

  how many hands a munchkin needs to equip the item
- bool itemLarge

  wether or not the item is large
- BadThingsFunc badThingsFunc

  bad things function that mapps onto a bad things function in badThingsFunc.h

5.8.1 Detailed Description

class of munchkin cards with params that could be needed for a single card

5.8.2 Constructor & Destructor Documentation

5.8.2.1 MunchkinCard()

```
MunchkinCard::MunchkinCard (
            int _markerID,
            const string & _cardName,
            const string & _effect,
            const string & _badThings,
            const string & _itemEffect,
            const string & _itemNeeds,
            vector< string > _bonis,
            ParentCardType _parentCardType,
            CardType _type,
            ItemType _itemType,
            int _strengthBoni,
            int _debuff,
            int _monsStrength,
            int _lvlUp,
            int _treasures,
            int _itemValue,
            int _handsNeeded,
            bool _itemLarge )
```

custom constructor

Parameters

_markerID	marker id that corresponse with the card
_cardName	card name
_effect	card effect
_badThings	(deprecated) string of bad things
_itemEffect	effect that an item has e.g. attack with fire and flame
_itemNeeds	param that a munchkin needs to equip a item e.g. has to be a dwarf
_bonis	boni that the card can have
_parentCardType	param for parent card type
_type	param for "child" card type
_itemType	item type
_strengthBoni	how much strength the munchkin receives with this card
_debuff	how much strength the munchkin looses through this card
_monsStrength	param how much the monsters strength is
_lvlUp	how many level you get r.g. when defeating the monster
_treasures	how many treasure the munchkin gets when defeating the monster
_itemValue	how much gold the item sells for
_handsNeeded	how many hands a munchkin needs to equip the item
_itemLarge	wether or not the item is large

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/MunchkinCards.h
- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/MunchkinCards.cpp

5.9 PlayerStats Struct Reference

Holds data of player stats.

`#include <GameState.h>`

Public Attributes

- string sex

  sex of player (male/female)
- vector< string > bonis

  bonis for player that are gained through different cards e.g. munchkin class
- vector< string > curses

  permanent curses that are a burden for the player
- vector< string > itemEffects

  effects of different items that are supporting the player e.g. attack with fire
- vector< string > munchClasses

  munchkin classes the player optained
- vector< string > munchRaces

  munchkin races the player optained
- int lvl

  level of the player
- int strength

  strength of a player through his equipment
- int hands

  how many hands player has left that are not holding an object
- int runStrength

  dice roll hat to be larger than x for the player to run away
- int availableClasses

  how many classes can the player optain
- int availableRaces

  how many races can the player optain
- bool carriesLargeItems

  wether or not the player carries a large item
- bool hasArmor

  wether or not the player carries an armor
- bool hasHat

  wether or not the player carries a hat
- bool hasShoes

  wether or not the player carries shoes

5.9.1 Detailed Description

Holds data of player stats.

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

- C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h

Chapter 6

File Documentation

6.1 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/BadThings.h File Reference

Bad Things Functions for Munchkin Cards.

```
#include <functional>
```

Classes

- struct BadThingsRetVal

Namespaces

- badthings

  Namespace containing all bad things functions. These can be assigned to a specific card.

Typedefs

- using BadThingsFunc = std::function< BadThingsRetVal(GameState &, const MunchkinCard &)>

  Callback type for bad things behaviour.

Functions

- BadThingsRetVal badthings::emptyBadThings (GameState &gamestate, const MunchkinCard &card)

  Does nothing (placeholder)
- BadThingsRetVal badthings::looseHand (GameState &gamestate, const MunchkinCard &card)

  Player looses 1 hand and the function checks whether or not he can still carry all his items that need hands.
- BadThingsRetVal badthings::looseClass (GameState &gamestate, const MunchkinCard &card)

  Player looses his class.
- BadThingsRetVal badthings::playerDies (GameState &gamestate, const MunchkinCard &card)

  Player dies.
- BadThingsRetVal badthings::looseLevel (GameState &gamestate, const MunchkinCard &card)

  Player looses 1 level.
- BadThingsRetVal badthings::maleDeadFemaleLevelDown (GameState &gamestate, const MunchkinCard &card)

  Male Players die. Female players loose 1 level.

6.1.1 Detailed Description

Bad Things Functions for Munchkin Cards.

Author

Benjamin Lueben

In this class all the BadThings functions for every Munchkin Card are defined.

6.2 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/↩ Button.h File Reference

Class to define buttons which are used for user input.

```
#include "opencv2/opencv.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include <functional>
```

Classes

- class Button
  class for Buttons

Typedefs

- using **button_callback** = std::function< void(const Button &)>

Enumerations

- enum class ButtonOrigin { topleft , topright , bottomleft , bottomright }
  specifies in what corner the button should be shown

6.2.1 Detailed Description

Class to define buttons which are used for user input.

Author

Benjamin Lueben

With this class different buttons can be defined which can be used for user input. Buttons are used when the user needs to confirm an action or to exit the tutorial.

6.2.2 Enumeration Type Documentation

6.2.2.1 ButtonOrigin

```
enum ButtonOrigin  [strong]
```

specifies in what corner the button should be shown

Enumerator

topleft	top left corner
topright	top right corner
bottomleft	bottom left corner
bottomright	bottom right corner

6.3 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/CardTypeActions.h File Reference

Class to define the functions for munchkin card types.

```
#include <functional>
```

Classes

- struct CardTypeRetVal

Namespaces

- cardtypeaction

  Namespace containing all cardtype functions. These are mapped to the actual card type.

Typedefs

- using **CardTypeFunc** = std::function< CardTypeRetVal(GameState &, const MunchkinCard &)>

Functions

- CardTypeRetVal cardtypeaction::curse (GameState &gamestate, const MunchkinCard &card)

  processes card type curse
- CardTypeRetVal cardtypeaction::joker (GameState &gamestate, const MunchkinCard &card)

  processes card type joker
- CardTypeRetVal cardtypeaction::monster (GameState &gamestate, const MunchkinCard &card)

  proceses card type monster
- CardTypeRetVal cardtypeaction::munchClass (GameState &gamestate, const MunchkinCard &card)

  processes card Type munchkin class
- CardTypeRetVal cardtypeaction::race (GameState &gamestate, const MunchkinCard &card)

  processes card type munchkin race
- CardTypeRetVal cardtypeaction::item (GameState &gamestate, const MunchkinCard &card)

  processes card type item
- CardTypeRetVal cardtypeaction::itemBuff (GameState &gamestate, const MunchkinCard &card)

  processes card type item buff
- CardTypeRetVal cardtypeaction::lvlUp (GameState &gamestate, const MunchkinCard &card)

  processes card type level up

6.3.1 Detailed Description

Class to define the functions for munchkin card types.

Author

Benjamin Lueben

In this class all the functions for the different munchkin card types are defined which are used to implement the game logic of the munchkin cardgame.

6.4 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h File Reference

GameState class to store all needed parameters for tutorial.

```
#include "opencv2/opencv.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include <string>
#include <vector>
#include "Button.h"
```

Classes

- struct MouseParams

  Holds data of a mouse event.
- struct PlayerStats

  Holds data of player stats.
- struct GameState

Enumerations

- enum class MouseEvent { lclick , rclick , move , none }

  Specifies the mouse event.

6.4.1 Detailed Description

GameState class to store all needed parameters for tutorial.

Author

Benjamin Lueben

In this class all the parameters that are needed for the tutorial are stored.

6.4.2 Enumeration Type Documentation

6.4.2.1 MouseEvent

```
enum MouseEvent  [strong]
```

Specifies the mouse event.

Enumerator

lclick	left click
rclick	right click
move	mouse move
none	empty event

6.5 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/InputEvent.h File Reference

Class for all input events of the player through mouseklicks.

Classes

- struct InputEvent

Enumerations

- enum class EventType {
  LmPress , LmRelease , RmPress , RmRelease ,
  KeyPress , KeyRelease }

  enum of possible event types

6.5.1 Detailed Description

Class for all input events of the player through mouseklicks.

Author

Benjamin Lueben

6.5.2 Enumeration Type Documentation

6.5.2.1 EventType

```
enum EventType  [strong]
```

enum of possible event types

Enumerator

LmPress	left mouse press
LmRelease	left mouse release
RmPress	right mouse press
RmRelease	right mouse release
KeyPress	keyboard key press
KeyRelease	keyboard key release

6.6 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/MunchkinCards.h File Reference

MunchkinCards class in which all parameters for the munchkin cards are stored.

```
#include <string>
#include <vector>
#include "opencv2/opencv.hpp"
#include "opencv2/imgcodecs.hpp"
#include "opencv2/imgproc.hpp"
#include <functional>
#include "GameState.h"
#include "BadThings.h"
```

Classes

- class MunchkinCard

  class of munchkin cards with params that could be needed for a single card

Enumerations

- enum class ParentCardType { door , treasure }

  enum class for parent card types
- enum class CardType {
  curse , munchClass , joker , lvlUp ,
  itemBuff , monster , race , item ,
  removeCard }

  enum class for "child" card Types
- enum class ItemType {
  armor , shoes , hat , boni ,
  weapon , joker , clothing }

  enum class of item types

6.6.1 Detailed Description

MunchkinCards class in which all parameters for the munchkin cards are stored.

Author

Benjamin Lueben

In this class the munchkin Card object is defined with every parameter the card could have and with custom constructors which could be used to define the cards more easily

6.6.2 Enumeration Type Documentation

6.6.2.1 CardType

```
enum CardType  [strong]
```

enum class for "child" card Types

Enumerator

curse	card type curse
munchClass	card type munchkin class
joker	card type joker
lvlUp	card type level up
itemBuff	card type item buff
monster	card type monster
race	card type munchkin race
item	card type item
removeCard	param to signal removing a card

6.6.2.2 ItemType

```
enum ItemType  [strong]
```

enum class of item types

Enumerator

armor	item type armor
shoes	item class shoes
hat	item class hat
boni	item class boni
weapon	item class weapon
joker	item class joker
clothing	item class clothing

6.6.2.3 ParentCardType

```
enum ParentCardType  [strong]
```

enum class for parent card types

Enumerator

door	card type door
treasure	card type treasure

Index