

# Praxisprojekt Code Dokumentation

Generated by Doxygen 1.9.1



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 badthings Namespace Reference	7
4.1.1 Detailed Description	7
4.1.2 Function Documentation	7
4.1.2.1 emptyBadThings()	7
4.1.2.2 looseClass()	8
4.1.2.3 looseHand()	8
4.1.2.4 looseLevel()	8
4.1.2.5 maleDeadFemaleLevelDown()	9
4.1.2.6 playerDies()	9
4.2 cardtypeaction Namespace Reference	9
4.2.1 Detailed Description	10
4.2.2 Function Documentation	10
4.2.2.1 curse()	10
4.2.2.2 item()	10
4.2.2.3 itemBuff()	11
4.2.2.4 joker()	11
4.2.2.5 lvlUp()	11
4.2.2.6 monster()	11
4.2.2.7 munchClass()	12
4.2.2.8 race()	12
<b>5 Class Documentation</b>	<b>13</b>
5.1 BadThingsRetVal Struct Reference	13
5.2 Button Class Reference	13
5.2.1 Detailed Description	14
5.2.2 Constructor & Destructor Documentation	14
5.2.2.1 Button()	14
5.2.3 Member Function Documentation	15
5.2.3.1 draw()	15
5.2.3.2 poll_click()	15
5.3 CardTypeRetVal Struct Reference	15
5.4 ExtrasRetVal Struct Reference	15
5.5 GameState Struct Reference	16
5.5.1 Detailed Description	16

5.6 InputEvent Struct Reference . . . . .	16
5.7 MouseParams Struct Reference . . . . .	17
5.7.1 Detailed Description . . . . .	17
5.8 MunchkinCard Class Reference . . . . .	17
5.8.1 Detailed Description . . . . .	18
5.8.2 Constructor & Destructor Documentation . . . . .	19
5.8.2.1 MunchkinCard() . . . . .	19
5.9 PlayerStats Struct Reference . . . . .	20
5.9.1 Detailed Description . . . . .	20
<b>6 File Documentation</b>	<b>21</b>
6.1 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/BadThings.h File Reference . . . . .	21
6.1.1 Detailed Description . . . . .	22
6.2 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/Button.h File Reference . . . . .	22
6.2.1 Detailed Description . . . . .	22
6.2.2 Enumeration Type Documentation . . . . .	22
6.2.2.1 ButtonOrigin . . . . .	22
6.3 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/CardTypeActions.h File Reference . . . . .	23
6.3.1 Detailed Description . . . . .	24
6.4 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h File Reference . . . . .	24
6.4.1 Detailed Description . . . . .	24
6.4.2 Enumeration Type Documentation . . . . .	24
6.4.2.1 MouseEvent . . . . .	24
6.5 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/InputEvent.h File Reference . . . . .	25
6.5.1 Detailed Description . . . . .	25
6.5.2 Enumeration Type Documentation . . . . .	25
6.5.2.1 EventType . . . . .	25
6.6 C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/MunchkinCards.h File Reference . . . . .	26
6.6.1 Detailed Description . . . . .	26
6.6.2 Enumeration Type Documentation . . . . .	26
6.6.2.1 CardType . . . . .	26
6.6.2.2 ItemType . . . . .	27
6.6.2.3 ParentCardType . . . . .	27
<b>Index</b>	<b>29</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">badthings</a>	Namespace containing all bad things functions. These can be assigned to a specific card . . .	<a href="#">7</a>
<a href="#">cardtypeaction</a>	Namespace containing all cardtype functions. These are mapped to the actual card type . . .	<a href="#">9</a>



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">BadThingsRetVal</a>	13
<a href="#">Button</a>	
Class for Buttons	13
<a href="#">CardTypeRetVal</a>	15
<a href="#">ExtrasRetVal</a>	15
<a href="#">GameState</a>	16
<a href="#">InputEvent</a>	16
<a href="#">MouseParams</a>	
Holds data of a mouse event	17
<a href="#">MunchkinCard</a>	
Class of munchkin cards with params that could be needed for a single card	17
<a href="#">PlayerStats</a>	
Holds data of player stats	20





## Chapter 3

# File Index

### 3.1 File List

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

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">BadThings.h</a>	
Bad Things Functions for Munchkin Cards . . . . .	21
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">Button.h</a>	
Class to define buttons which are used for user input . . . . .	22
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">CardTypeActions.h</a>	
Class to define the functions for munchkin card types . . . . .	23
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <b>common.h</b>	??
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <b>Extras.h</b>	??
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">GameState.h</a>	
<a href="#">GameState</a> class to store all needed parameters for tutorial . . . . .	24
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">InputEvent.h</a>	
Class for all input events of the player through mouseclicks . . . . .	25
C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/ <a href="#">MunchkinCards.h</a>	
MunchkinCards class in which all parameters for the munchkin cards are stored . . . . .	26



## 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 loses 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 loses his class.*
- [BadThingsRetVal playerDies](#) ([GameState](#) &gamestate, const [MunchkinCard](#) &card)  
*Player dies.*
- [BadThingsRetVal looseLevel](#) ([GameState](#) &gamestate, const [MunchkinCard](#) &card)  
*Player loses 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

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

**4.1.2.2 looseClass()**

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

Player loses his class.

## Parameters

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

**4.1.2.3 looseHand()**

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

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

## Parameters

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

**4.1.2.4 looseLevel()**

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

Player loses 1 level.

## Parameters

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

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

#### 4.1.2.6 playerDies()

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

Player dies.

##### Parameters

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

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

#### 4.2.2.6 monster()

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

```

    GameState & gamestate,
    const MunchkinCard & card )

```

processes card type monster

#### Parameters

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<i>gamestate</i>	The game state is modified according to the corresponding munchkin rule/card.
<i>card</i>	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

<a href="#">_id</a>	<a href="#">Button</a> id
<a href="#">_rect</a>	rectangle which defines the dimensions of the button
<a href="#">_color</a>	defines the color of the button in GBR
<a href="#">_visible</a>	used to hide button when not needed
<a href="#">_origin</a>	origin of the button as in enum class ButtonOrigin
<a href="#">_callback</a>	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

<i>canvas</i>	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

<i>point</i>	point of click
<i>canvas_size</i>	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 _parentCardType, 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 corresponsse with the card*
- int [strengthBoni](#)  
*how much strength the munchkin receives with this card*
- int [debuff](#)  
*how much strength the munchkin loses 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

<code>_markerID</code>	marker id that correspond with the card
<code>_cardName</code>	card name
<code>_effect</code>	card effect
<code>_badThings</code>	(deprecated) string of bad things
<code>_itemEffect</code>	effect that an item has e.g. attack with fire and flame
<code>_itemNeeds</code>	param that a munchkin needs to equip a item e.g. has to be a dwarf
<code>_bonis</code>	boni that the card can have
<code>_parentCardType</code>	param for parent card type
<code>_type</code>	param for "child" card type
<code>_itemType</code>	item type
<code>_strengthBoni</code>	how much strength the munchkin receives with this card
<code>_debuff</code>	how much strength the munchkin loses through this card
<code>_monsStrength</code>	param how much the monsters strength is
<code>_lvlUp</code>	how many level you get r.g. when defeating the monster
<code>_treasures</code>	how many treasure the munchkin gets when defeating the monster
<code>_itemValue</code>	how much gold the item sells for
<code>_handsNeeded</code>	how many hands a munchkin needs to equip the item
<code>_itemLarge</code>	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 obtained*
- vector< string > [munchRaces](#)  
*munchkin races the player obtained*
- 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 obtain*
- int [availableRaces](#)  
*how many races can the player obtain*
- 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/Bad Things.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 loses 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 loses his class.*
- [BadThingsRetVal](#) [badthings::playerDies](#) ([GameState](#) &gamestate, const [MunchkinCard](#) &card)  
*Player dies.*
- [BadThingsRetVal](#) [badthings::looseLevel](#) ([GameState](#) &gamestate, const [MunchkinCard](#) &card)  
*Player loses 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)  
*processes 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 mouseclicks.

**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 mouseclicks.

**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

armor  
    MunchkinCards.h, 27

badthings, 7  
    emptyBadThings, 7  
    looseClass, 8  
    looseHand, 8  
    looseLevel, 8  
    maleDeadFemaleLevelDown, 9  
    playerDies, 9

BadThingsRetVal, 13

boni  
    MunchkinCards.h, 27

bottomleft  
    Button.h, 23

bottomright  
    Button.h, 23

Button, 13  
    Button, 14  
    draw, 15  
    poll\_click, 15

Button.h  
    bottomleft, 23  
    bottomright, 23  
    ButtonOrigin, 22  
    toleft, 23  
    topright, 23

ButtonOrigin  
    Button.h, 22

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/BadThings.h, 21

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/Button.h, 22

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/CardTypeActions.h, 23

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/GameState.h, 24

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/InputEvent.h, 25

C:/Users/Jlueb/source/repos/TTMunchkinTut/TTMunchkinTut/MunchkinCards.h, 26

CardType  
    MunchkinCards.h, 26

cardtypeaction, 9  
    curse, 10  
    item, 10  
    itemBuff, 10  
    joker, 11  
    lvlUp, 11

monster, 11

munchClass, 12

race, 12

CardTypeRetVal, 15

clothing  
    MunchkinCards.h, 27

curse  
    cardtypeaction, 10  
    MunchkinCards.h, 27

door  
    MunchkinCards.h, 27

draw  
    Button, 15

emptyBadThings  
    badthings, 7

EventType  
    InputEvent.h, 25

ExtrasRetVal, 15

GameState, 16

GameState.h  
    lclick, 25  
    MouseEvent, 24  
    move, 25  
    none, 25  
    rclick, 25

hat  
    MunchkinCards.h, 27

InputEvent, 16

InputEvent.h  
    EventType, 25  
    KeyPress, 25  
    KeyRelease, 25  
    LmPress, 25  
    LmRelease, 25  
    RmPress, 25  
    RmRelease, 25

item  
    cardtypeaction, 10  
    MunchkinCards.h, 27

itemBuff  
    cardtypeaction, 10  
    MunchkinCards.h, 27

ItemType  
    MunchkinCards.h, 27

joker

- cardtypeaction, [11](#)
  - MunchkinCards.h, [27](#)
- KeyPress
  - InputEvent.h, [25](#)
- KeyRelease
  - InputEvent.h, [25](#)
- lclick
  - GameState.h, [25](#)
- LmPress
  - InputEvent.h, [25](#)
- LmRelease
  - InputEvent.h, [25](#)
- looseClass
  - badthings, [8](#)
- looseHand
  - badthings, [8](#)
- looseLevel
  - badthings, [8](#)
- lvlUp
  - cardtypeaction, [11](#)
  - MunchkinCards.h, [27](#)
- maleDeadFemaleLevelDown
  - badthings, [9](#)
- monster
  - cardtypeaction, [11](#)
  - MunchkinCards.h, [27](#)
- MouseEvent
  - GameState.h, [24](#)
- MouseParams, [17](#)
- move
  - GameState.h, [25](#)
- munchClass
  - cardtypeaction, [12](#)
  - MunchkinCards.h, [27](#)
- MunchkinCard, [17](#)
  - MunchkinCard, [19](#)
- MunchkinCards.h
  - armor, [27](#)
  - boni, [27](#)
  - CardType, [26](#)
  - clothing, [27](#)
  - curse, [27](#)
  - door, [27](#)
  - hat, [27](#)
  - item, [27](#)
  - itemBuff, [27](#)
  - ItemType, [27](#)
  - joker, [27](#)
  - lvlUp, [27](#)
  - monster, [27](#)
  - munchClass, [27](#)
  - ParentCardType, [27](#)
  - race, [27](#)
  - removeCard, [27](#)
  - shoes, [27](#)
  - treasure, [27](#)
  - weapon, [27](#)
- none
  - GameState.h, [25](#)
- ParentCardType
  - MunchkinCards.h, [27](#)
- playerDies
  - badthings, [9](#)
- PlayerStats, [20](#)
- poll\_click
  - Button, [15](#)
- race
  - cardtypeaction, [12](#)
  - MunchkinCards.h, [27](#)
- rclick
  - GameState.h, [25](#)
- removeCard
  - MunchkinCards.h, [27](#)
- RmPress
  - InputEvent.h, [25](#)
- RmRelease
  - InputEvent.h, [25](#)
- shoes
  - MunchkinCards.h, [27](#)
- toleft
  - Button.h, [23](#)
- topright
  - Button.h, [23](#)
- treasure
  - MunchkinCards.h, [27](#)
- weapon
  - MunchkinCards.h, [27](#)