MonopolyVsAi

1.0.0

Generated by Doxygen 1.9.1

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 ann Namespace Reference

Classes

- class neuron
- class neuralnet

Enumerations

- enum type { RECURRENT , NON_RECURRENT }

5.1.1 Enumeration Type Documentation

5.1.1.1 type

```
enum ann::type
```

Enumerator

RECURRENT	
NON_RECURRENT	

5.2 neat Namespace Reference

Classes

- struct mutation_rate_container

- struct speciating_parameter_container
- struct network_info_container
- struct gene
- class genome
- struct specie
- class innovation_container
- class pool

Chapter 6

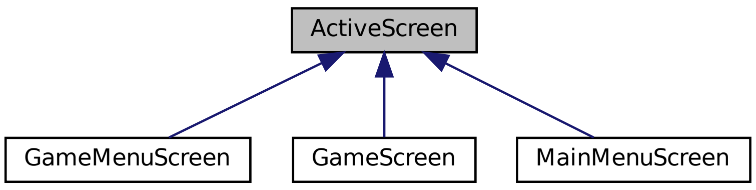
Class Documentation

6.1 ActiveScreen Class Reference

Represents the base class for handling displayed screens in the project.

```
#include <ActiveScreen.h>
```

Inheritance diagram for ActiveScreen:



Public Member Functions

- ActiveScreen ()

  Constructor for the ActiveScreen class.
- virtual ScreenEventType worker ()=0

  Pure virtual function representing the worker function for the screen.
- virtual void draw ()=0

  Pure virtual function to draw the screen.
- sf::Font & getFont ()

  Gets the SFML font used for text on the screen.
- void setFont (sf::Font font)

  Sets the SFML font used for text on the screen.
- void addButton (std::shared_ptr< Button > button_tmp)

  Adds a button to the screen.
- void addText (std::shared_ptr< sf::Text > text_tmp)

Adds a text object to the screen.

- std::vector< std::shared_ptr< Button > > & getButtons ()

  Gets the vector of buttons displayed on the screen.
- std::vector< std::shared_ptr< sf::Text > > & getTexts ()

  Gets the vector of SFML Text objects displayed on the screen.
- ContextWindow * getContextWindow ()

  Gets the pointer to the ContextWindow.
- void setContextWindow (ContextWindow *)

  Sets the pointer to the ContextWindow.
- ActiveScreenType getScreenType ()

  Gets the type of the active screen.
- void setScreenType (ActiveScreenType type)

  Sets the type of the active screen.
- void buttonSetColors (std::shared_ptr< Button > buttonPtr)

  Sets the colors of a button based on its state.
- virtual std::vector< std::shared_ptr< playerSettings > > getPlayersSettings () const

  Virtual function to get players' settings.
- virtual std::vector< std::shared_ptr< Player > > getPlayersResult ()

  Virtual function to get players' results.

6.1.1 Detailed Description

Represents the base class for handling displayed screens in the project.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ActiveScreen()

```
ActiveScreen::ActiveScreen ( )
```

Constructor for the ActiveScreen class.

6.1.3 Member Function Documentation

6.1.3.1 addButton()

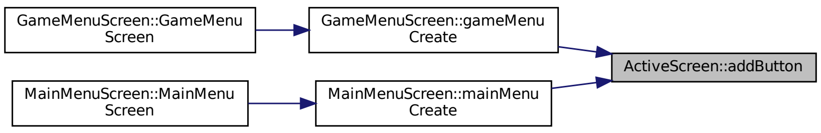
```
void ActiveScreen::addButton (
            std::shared_ptr< Button > button_tmp )
```

Adds a button to the screen.

Parameters

button_tmp	The button to add.

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create
ActiveScreen::addButton

6.1.3.2 addText()

```
void ActiveScreen::addText (
             std::shared_ptr< sf::Text > text_tmp )
```

Adds a text object to the screen.

Parameters

text_tmp	The SFML Text object to add.

6.1.3.3 buttonSetColors()

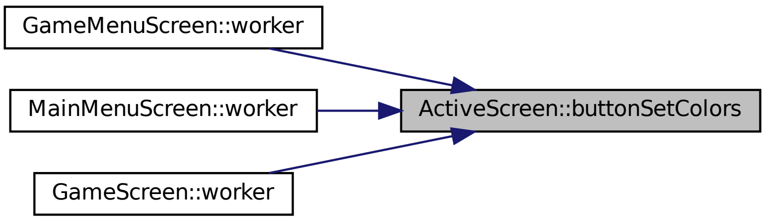
```
void ActiveScreen::buttonSetColors (
             std::shared_ptr< Button > buttonPtr )
```

Sets the colors of a button based on its state.

Parameters

buttonPtr	The shared pointer to the Button.

Here is the caller graph for this function:



6.1.3.4 draw()

```
virtual void ActiveScreen::draw ( )  [pure virtual]
```

Pure virtual function to draw the screen.

Implemented in GameScreen, MainMenuScreen, and GameMenuScreen.

6.1.3.5 getButtons()

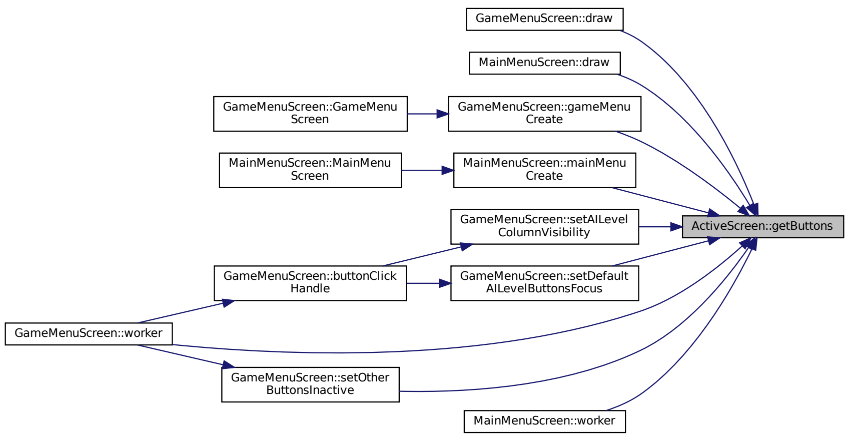
```
std::vector< std::shared_ptr< Button > > & ActiveScreen::getButtons ( )
```

Gets the vector of buttons displayed on the screen.

Returns

Vector of shared pointers to Button objects.

Here is the caller graph for this function:


GameMenuScreen::draw
MainMenuScreen::draw
GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create
GameMenuScreen::setAILevel ColumnVisibility
ActiveScreen::getButtons
GameMenuScreen::buttonClick Handle
GameMenuScreen::setDefault AILevelButtonsFocus
GameMenuScreen::worker
GameMenuScreen::setOther ButtonsInactive
MainMenuScreen::worker

6.1.3.6 getContextWindow()

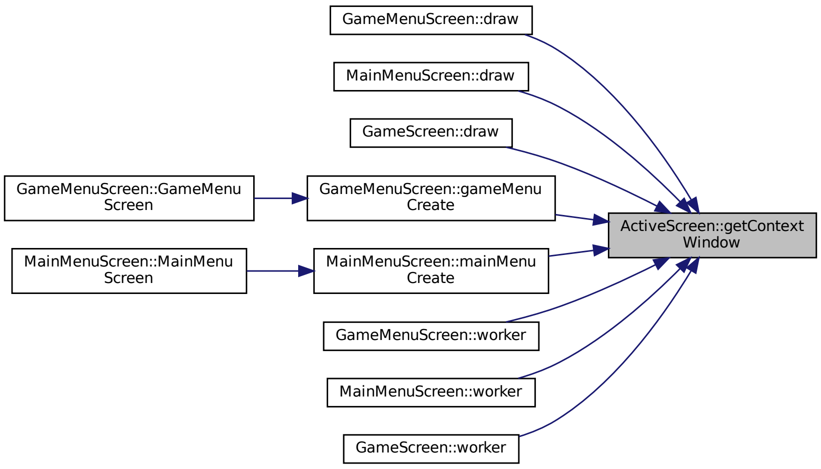
`ContextWindow * ActiveScreen::getContextWindow ( )`

Gets the pointer to the ContextWindow.

Returns

Pointer to the ContextWindow.

Here is the caller graph for this function:


GameMenuScreen::draw
MainMenuScreen::draw
GameScreen::draw
GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
ActiveScreen::getContext Window
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create
GameMenuScreen::worker
MainMenuScreen::worker
GameScreen::worker

6.1.3.7 getFont()

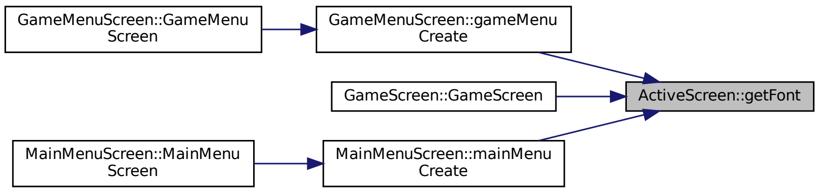
`sf::Font & ActiveScreen::getFont ( )`

Gets the SFML font used for text on the screen.

Returns

The SFML font.

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
GameScreen::GameScreen
ActiveScreen::getFont
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create

6.1.3.8 getPlayersResult()

`std::vector< std::shared_ptr< Player > > ActiveScreen::getPlayersResult ( ) [virtual]`

Virtual function to get players' results.

Returns

Vector of shared pointers to Player objects.

Reimplemented in GameScreen.

6.1.3.9 getPlayersSettings()

`std::vector< std::shared_ptr< playerSettings > > ActiveScreen::getPlayersSettings ( ) const [virtual]`

Virtual function to get players' settings.

Returns

Vector of shared pointers to playerSettings objects.

Reimplemented in GameMenuScreen.

6.1.3.10 getScreenType()

`ActiveScreenType ActiveScreen::getScreenType ( )`

Gets the type of the active screen.

Returns

The ActiveScreenType.

6.1.3.11 getTexts()

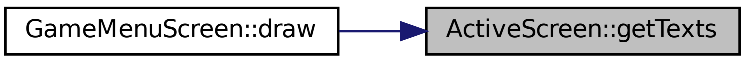
```
std::vector< std::shared_ptr< sf::Text > > & ActiveScreen::getTexts ( )
```

Gets the vector of SFML Text objects displayed on the screen.

Returns

Vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.1.3.12 setContextWindow()

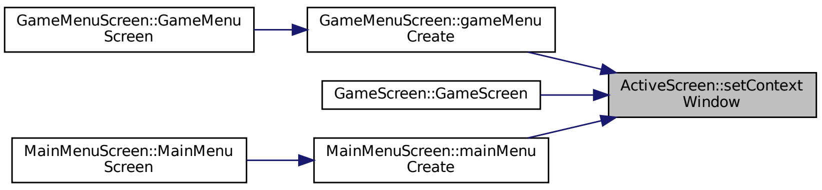
```
void ActiveScreen::setContextWindow (
            ContextWindow * cw )
```

Sets the pointer to the ContextWindow.

Parameters

context_window	Pointer to the ContextWindow to set.

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
GameScreen::GameScreen
ActiveScreen::setContext Window
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create

6.1.3.13 setFont()

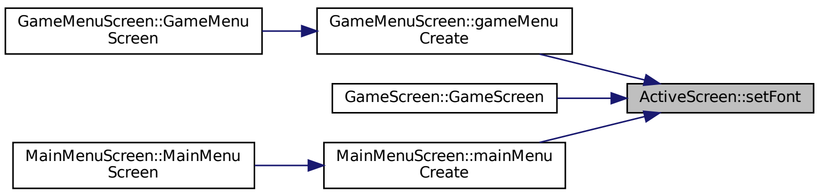
```
void ActiveScreen::setFont (
             sf::Font font )
```

Sets the SFML font used for text on the screen.

Parameters

font	The SFML font to set.

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
GameScreen::GameScreen
ActiveScreen::setFont
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create

6.1.3.14 setScreenType()

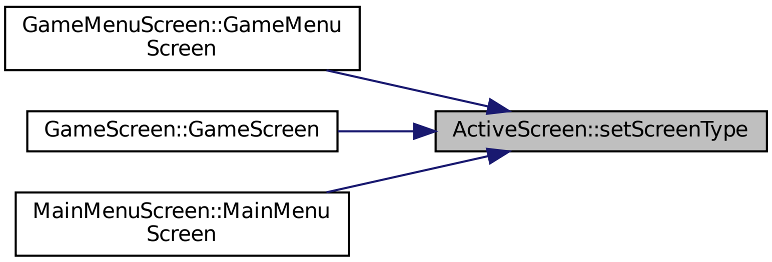
```
void ActiveScreen::setScreenType (
             ActiveScreenType type )
```

Sets the type of the active screen.

Parameters

type	The ActiveScreenType to set.

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameScreen::GameScreen
MainMenuScreen::MainMenu Screen
ActiveScreen::setScreenType

6.1.3.15 worker()

```
virtual ScreenEventType ActiveScreen::worker ( )  [pure virtual]
```

Pure virtual function representing the worker function for the screen.

Returns

The ScreenEventType associated with the user interaction.

Implemented in GameScreen, MainMenuScreen, and GameMenuScreen.

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

- /home/kamil/zpr/Monopoly/ActiveScreen.h
- /home/kamil/zpr/Monopoly/ActiveScreen.cc

6.2 AiAdapter Class Reference

Represents an adapter for AI input data.

```
#include <AiAdapter.h>
```

Public Member Functions

- AiAdapter ()

  Constructor for the AiAdapter class.
- std::vector< double > getInputs ()

  Gets the AI network inputs.
- double convertMoney (unsigned int money)

  Converts money into a format suitable for AI input.
- double convertMoneyValue (double value)

  Converts money value into a format suitable for AI input.
- double convertHouseValue (double value)

  Converts house value into a format suitable for AI input.
- double convertPosition (unsigned int position)

  Converts position into a format suitable for AI input.
- double convertCard (unsigned int cards)

  Converts card value into a format suitable for AI input.
- double convertHouse (bool isHotel, unsigned int houseNumber)

  Converts house state into a format suitable for AI input.
- void setTurn (unsigned int index)

  Sets the turn for a given index.
- void setSelection (unsigned int index)

  Sets the selection for a given index.
- void setSelectionState (unsigned int index, int state)

  Sets the selection state for a given index.
- void setMoneyContext (int state)

  Sets the money context state.
- void clearSelectionState ()

  Clears the selection state.
- void setPosition (unsigned int index, unsigned int position)

  Sets the position for a given index.
- void setMoney (unsigned int index, unsigned int money)

  Sets the money for a given index.
- void setCard (unsigned int index, unsigned int cards)

  Sets the card for a given index.
- void setJail (unsigned int index, unsigned int state)

  Sets the jail state for a given index.
- void setOwner (unsigned int property, unsigned int state)

  Sets the owner state for a given property.
- void setMortgage (unsigned int property, unsigned int state)

  Sets the mortgage state for a given property.
- void setHouse (bool isHotel, unsigned int houseNumber, unsigned int id)

  Sets the house state for a given house.

6.2.1 Detailed Description

Represents an adapter for AI input data.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AiAdapter()

```
AiAdapter::AiAdapter ( )
```

Constructor for the AiAdapter class.

6.2.3 Member Function Documentation

6.2.3.1 clearSelectionState()

```
void AiAdapter::clearSelectionState ( )
```

Clears the selection state.

6.2.3.2 convertCard()

```
double AiAdapter::convertCard (
             unsigned int cards )
```

Converts card value into a format suitable for AI input.

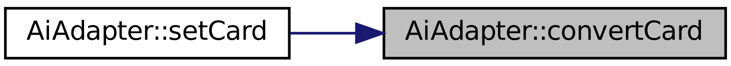
Parameters

cards	The card value to convert.

Returns

The converted card value.

Here is the caller graph for this function:



6.2.3.3 convertHouse()

```
double AiAdapter::convertHouse (
            bool isHotel,
            unsigned int houseNumber )
```

Converts house state into a format suitable for AI input.

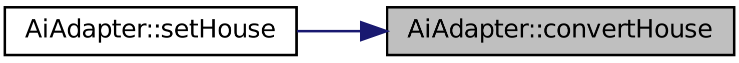
Parameters

isHotel	Flag indicating if it's a hotel.
houseNumber	The house number.

Returns

The converted house state.

Here is the caller graph for this function:



6.2.3.4 convertHouseValue()

```
double AiAdapter::convertHouseValue (
            double value )
```

Converts house value into a format suitable for AI input.

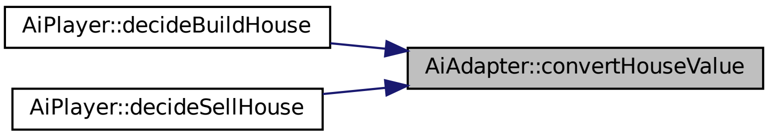
Parameters

value	The house value to convert.

Returns

The converted house value.

Here is the caller graph for this function:


AiPlayer::decideBuildHouse
AiPlayer::decideSellHouse
AiAdapter::convertHouseValue

6.2.3.5 convertMoney()

```
double AiAdapter::convertMoney (
             unsigned int money )
```

Converts money into a format suitable for AI input.

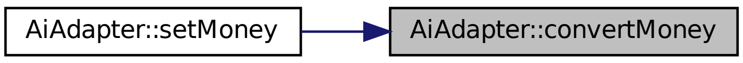
Parameters

money	The money value to convert.

Returns

The converted money value.

Here is the caller graph for this function:



6.2.3.6 convertMoneyValue()

```
double AiAdapter::convertMoneyValue (
             double value )
```

Converts money value into a format suitable for AI input.

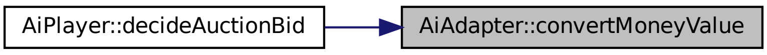
Parameters

value	The money value to convert.

Returns

The converted money value.

Here is the caller graph for this function:



6.2.3.7 convertPosition()

```
double AiAdapter::convertPosition (
             unsigned int position )
```

Converts position into a format suitable for AI input.

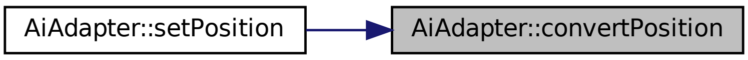
Parameters

position	The position to convert.

Returns

The converted position.

Here is the caller graph for this function:



6.2.3.8 getInputs()

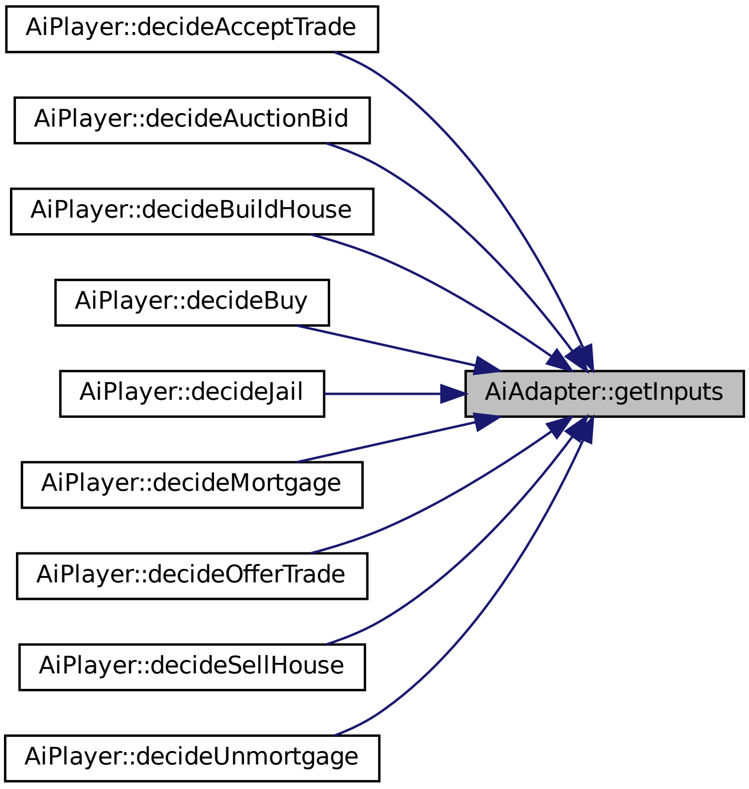
```
std::vector< double > AiAdapter::getInputs ( )
```

Gets the AI network inputs.

Returns

Vector of AI network inputs.

Here is the caller graph for this function:


AiPlayer::decideAcceptTrade
AiPlayer::decideAuctionBid
AiPlayer::decideBuildHouse
AiPlayer::decideBuy
AiPlayer::decideJail
AiPlayer::decideMortgage
AiPlayer::decideOfferTrade
AiPlayer::decideSellHouse
AiPlayer::decideUnmortgage
AiAdapter::getInputs

6.2.3.9 setCard()

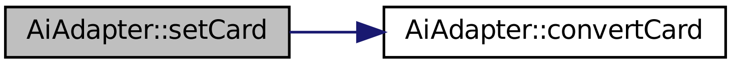
```
void AiAdapter::setCard (
            unsigned int index,
            unsigned int cards )
```

Sets the card for a given index.

Parameters

index	The index for which to set the card.
cards	The card to set.

Here is the call graph for this function:



6.2.3.10 setHouse()

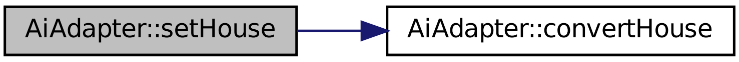
```
void AiAdapter::setHouse (
             bool isHotel,
             unsigned int houseNumber,
             unsigned int id )
```

Sets the house state for a given house.

Parameters

isHotel	Flag indicating if it's a hotel.
houseNumber	The house number.
id	The ID to set.

Here is the call graph for this function:



6.2.3.11 setJail()

```
void AiAdapter::setJail (
            unsigned int index,
            unsigned int state )
```

Sets the jail state for a given index.

Parameters

index	The index for which to set the jail state.
state	The state to set.

6.2.3.12 setMoney()

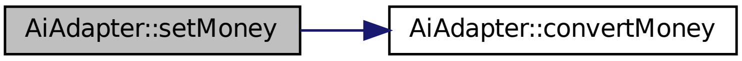
```
void AiAdapter::setMoney (
            unsigned int index,
            unsigned int money )
```

Sets the money for a given index.

Parameters

index	The index for which to set the money.
money	The money to set.

Here is the call graph for this function:



6.2.3.13 setMoneyContext()

```
void AiAdapter::setMoneyContext (
            int state )
```

Sets the money context state.

Parameters

state	The state to set.

6.2.3.14 setMortgage()

```
void AiAdapter::setMortgage (
             unsigned int property,
             unsigned int state )
```

Sets the mortgage state for a given property.

Parameters

property	The property for which to set the mortgage state.
state	The state to set.

6.2.3.15 setOwner()

```
void AiAdapter::setOwner (
             unsigned int property,
             unsigned int state )
```

Sets the owner state for a given property.

Parameters

property	The property for which to set the owner state.
state	The state to set.

6.2.3.16 setPosition()

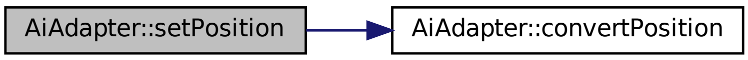
```
void AiAdapter::setPosition (
             unsigned int index,
             unsigned int position )
```

Sets the position for a given index.

Parameters

index	The index for which to set the position.
position	The position to set.

Here is the call graph for this function:



6.2.3.17 setSelection()

```
void AiAdapter::setSelection (
             unsigned int index )
```

Sets the selection for a given index.

Parameters

index	The index for which to set the selection.

6.2.3.18 setSelectionState()

```
void AiAdapter::setSelectionState (
             unsigned int index,
             int state )
```

Sets the selection state for a given index.

Parameters

index	The index for which to set the selection state.
state	The state to set.

6.2.3.19 setTurn()

```
void AiAdapter::setTurn (
             unsigned int index )
```

Sets the turn for a given index.

Parameters

index	The index for which to set the turn.

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

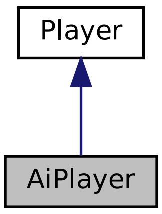
- /home/kamil/zpr/Monopoly/AiAdapter.h
- /home/kamil/zpr/Monopoly/AiAdapter.cc

6.3 AiPlayer Class Reference

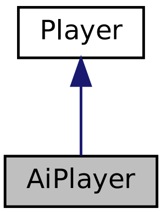
Represents an AI player in a Monopoly game, inheriting from Player.

```
#include <Player.h>
```

Inheritance diagram for AiPlayer:



Collaboration diagram for AiPlayer:



Public Member Functions

- AiPlayer ()
- AiPlayer (unsigned int money)
- AiPlayer (unsigned int money, ann::neuralnet nn)
- AiAdapter & getAdapter ()
- ann::neuralnet & getNeuralNetwork ()
- void setNeuralNetwork (ann::neuralnet new_nn)
- BuyDecision decideBuy (unsigned int index) override
- JailDecision decideJail () override
- Decision decideMortgage (unsigned int index) override
- Decision decideUnmortgage (unsigned int index) override
- unsigned int decideAuctionBid (unsigned int price) override
- unsigned int decideBuildHouse () override
- unsigned int decideSellHouse () override
- Decision decideOfferTrade () override
- Decision decideAcceptTrade () override

6.3.1 Detailed Description

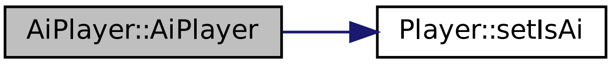
Represents an AI player in a Monopoly game, inheriting from Player.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 AiPlayer() `[1/3]`

```
AiPlayer::AiPlayer ( )
```

Default constructor for AiPlayer class. Here is the call graph for this function:



6.3.2.2 AiPlayer() `[2/3]`

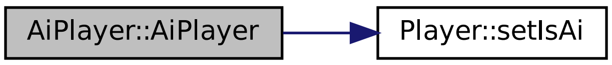
```
AiPlayer::AiPlayer (
             unsigned int money )
```

Constructor for AiPlayer class with initial money.

Parameters

money	Initial amount of money for the AI player.

Here is the call graph for this function:



6.3.2.3 AiPlayer() [3/3]

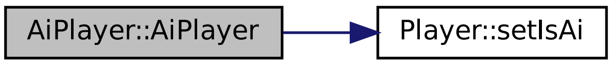
```
AiPlayer::AiPlayer (
            unsigned int money,
            ann::neuralnet nn )
```

Constructor for AiPlayer class with initial money and neural_network.

Parameters

money	Initial amount of money for the AI player.
n	Neural network used by a Player.

Here is the call graph for this function:



6.3.3 Member Function Documentation

6.3.3.1 decideAcceptTrade()

```
Decision AiPlayer::decideAcceptTrade ( )  [override], [virtual]
```

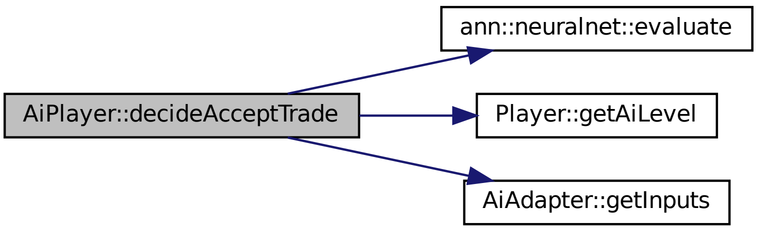
Make a decision for accepting a trade for the AI player (override from base class).

Returns

Decision object representing the acceptance of the trade.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.2 decideAuctionBid()

```
unsigned int AiPlayer::decideAuctionBid (
            unsigned int price )  [override], [virtual]
```

Make a decision for auction bidding for the AI player (override from base class).

Parameters

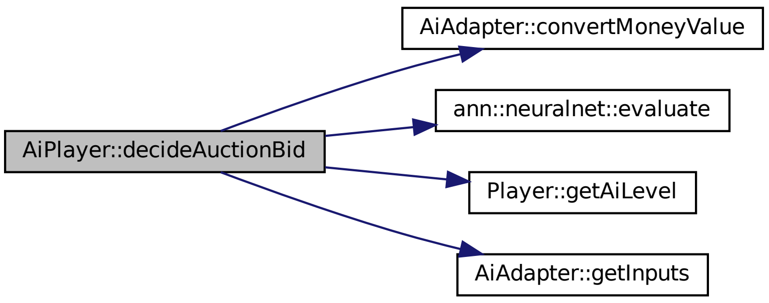
price	Current price in the auction.

Returns

The bid amount decided by the AI player.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.3 decideBuildHouse()

```
unsigned int AiPlayer::decideBuildHouse ( )  [override], [virtual]
```

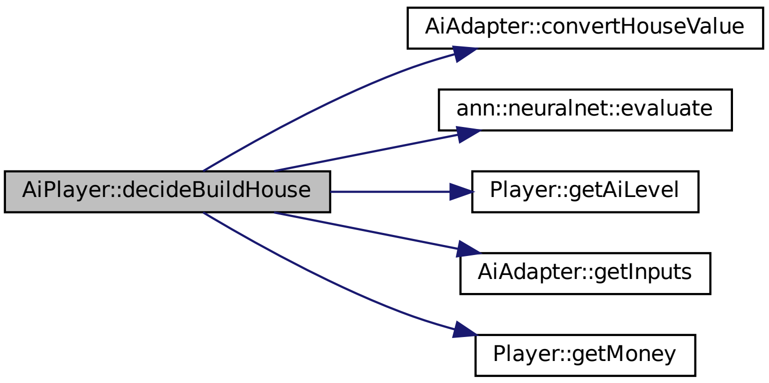
Make a decision for building a house for the AI player (override from base class).

Returns

The index of the property on which the AI player decides to build a house.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.4 decideBuy()

```
BuyDecision AiPlayer::decideBuy (
             unsigned int index )  [override], [virtual]
```

Make a buying decision for the AI player (override from base class).

Parameters

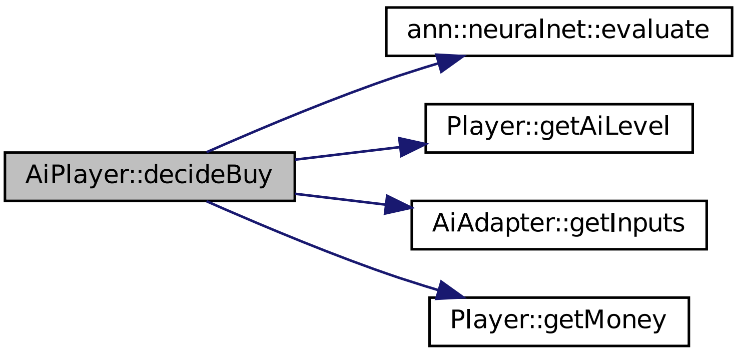
index	Index of the property to consider.

Returns

BuyDecision object representing the decision.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.5 decideJail()

```
JailDecision AiPlayer::decideJail ( )  [override], [virtual]
```

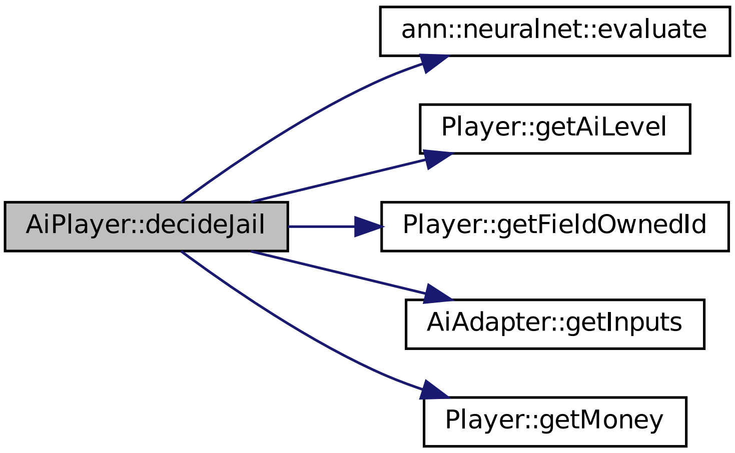
Make a jail decision for the AI player (override from base class).

Returns

JailDecision object representing the decision.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.6 decideMortgage()

```
Decision AiPlayer::decideMortgage (
             unsigned int index )  [override], [virtual]
```

Make a mortgage decision for the AI player (override from base class).

Parameters

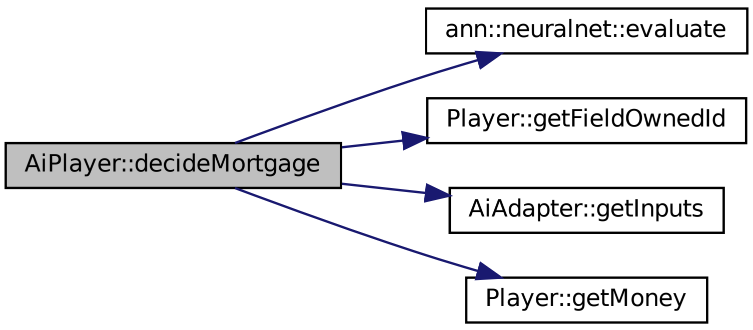
index	Index of the property to consider.

Returns

Decision object representing the mortgage decision.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.7 decideOfferTrade()

`Decision AiPlayer::decideOfferTrade ( ) [override], [virtual]`

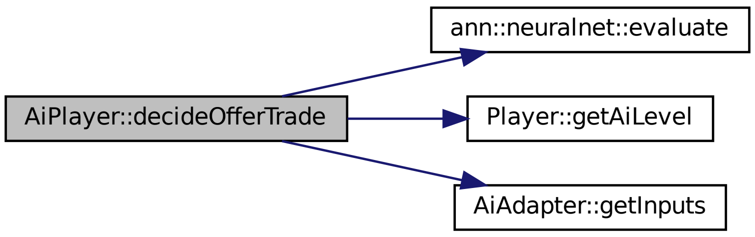
Make a decision for offering a trade for the AI player (override from base class).

Returns

Decision object representing the trade offer.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.8 decideSellHouse()

```
unsigned int AiPlayer::decideSellHouse ( )  [override], [virtual]
```

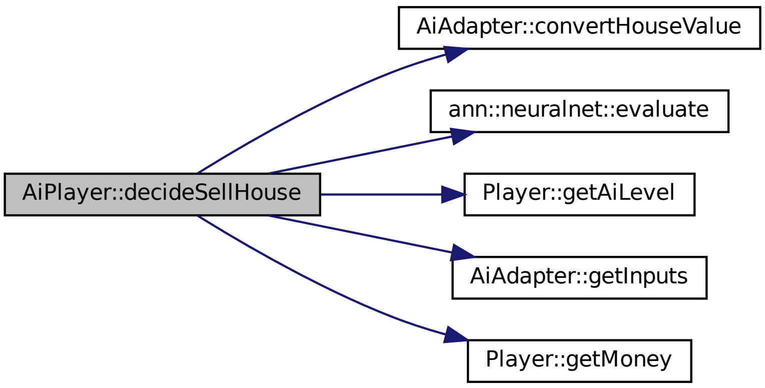
Make a decision for selling a house for the AI player (override from base class).

Returns

The index of the property from which the AI player decides to sell a house.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.9 decideUnmortgage()

```
Decision AiPlayer::decideUnmortgage (
             unsigned int index )  [override], [virtual]
```

Make an unmortgage decision for the AI player (override from base class).

Parameters

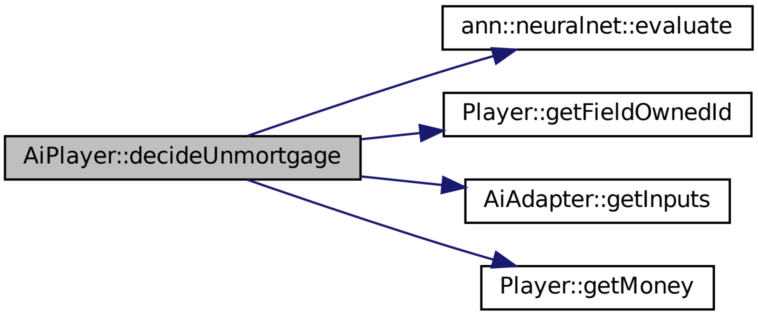
index	Index of the property to consider.

Returns

Decision object representing the unmortgage decision.

Reimplemented from Player.

Here is the call graph for this function:



6.3.3.10 getAdapter()

```
AiAdapter & AiPlayer::getAdapter ( ) [virtual]
```

Get the AI adapter associated with the AI player.

Reimplemented from Player.

6.3.3.11 getNeuralNetwork()

```
ann::neuralnet & AiPlayer::getNeuralNetwork ( ) [virtual]
```

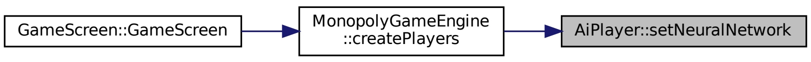
Get the neural network associated with the AI player.

Reimplemented from Player.

6.3.3.12 setNeuralNetwork()

```
void AiPlayer::setNeuralNetwork (
            ann::neuralnet new_nn )
```

Set the neura network used by the AI player. Here is the caller graph for this function:


GameScreen::GameScreen
MonopolyGameEngine ::createPlayers
AiPlayer::setNeuralNetwork

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

- /home/kamil/zpr/Monopoly/Player.h
- /home/kamil/zpr/Monopoly/Player.cc

6.4 Board Class Reference

Class representing the monopoly game board.

```
#include <Board.h>
```

Public Member Functions

- Board (const std::string file_path)

    Constructor for the Board class.
- const std::vector< PossibleFields > & getBoard ()

    Getter for the entire game board.
- unsigned int getFieldNumber ()

    Getter for the number of fields on the board.
- const sf::Vector2i getBoardPosition ()

    Getter for the position of the entire game board.
- void clearBoard ()

    Clears the entire game board.
- sf::Vector2i getFieldPositon (unsigned int id, sf::Vector2i prevPos, unsigned int x, unsigned int y)

    Getter for the position of a specific field on the board.
- float getFieldRotation (unsigned int id)

    Getter for the rotation angle of a specific field on the board.
- PossibleFields & getFieldById (unsigned int wanted_id)

    Getter for a specific field on the board by its unique identifier.

6.4.1 Detailed Description

Class representing the monopoly game board.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Board()

```
Board::Board (
             const std::string file_path )
```

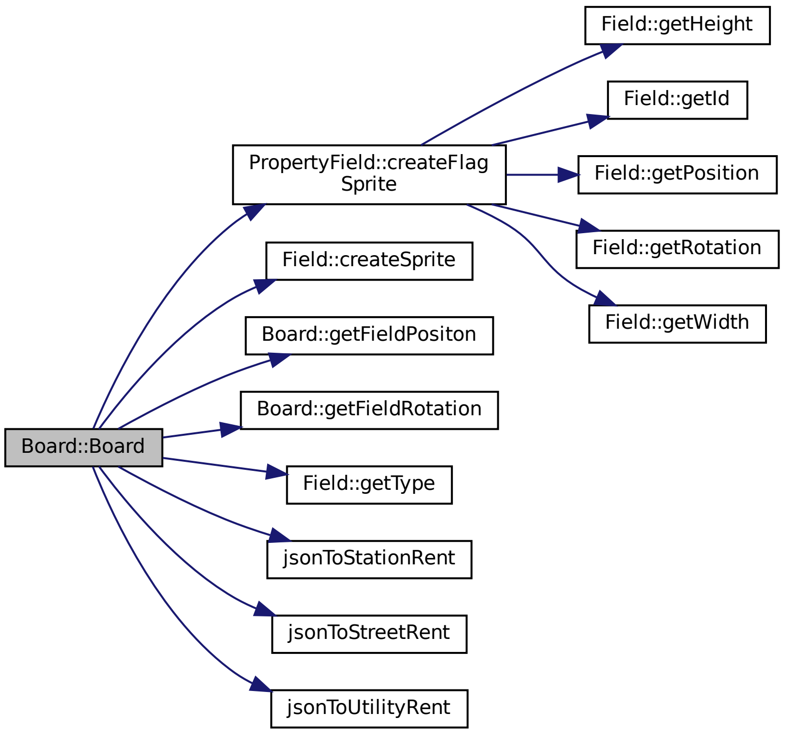
Constructor for the Board class.

Scale for displaying board stuff

Parameters

file_path	The file path to the 'board.json' file.

Here is the call graph for this function:



6.4.3 Member Function Documentation

6.4.3.1 clearBoard()

```
void Board::clearBoard ( )
```

Clears the entire game board.

6.4.3.2 getBoard()

```
const std::vector< PossibleFields > & Board::getBoard ( )
```

Getter for the entire game board.

Returns

A vector containing all types of fields on the board.

6.4.3.3 getBoardPosition()

```
const sf::Vector2i Board::getBoardPosition ( )
```

Getter for the position of the entire game board.

Returns

The position as an sf::Vector2i.

6.4.3.4 getFieldById()

```
PossibleFields & Board::getFieldById (
            unsigned int wanted_id )
```

Getter for a specific field on the board by its unique identifier.

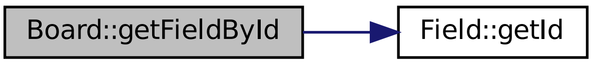
Parameters

wanted←_id	The unique identifier of the field.

Returns

A reference to the field specified by the identifier.

Here is the call graph for this function:



6.4.3.5 getFieldNumber()

```
unsigned int Board::getFieldNumber ( )
```

Getter for the number of fields on the board.

Returns

The number of fields.

6.4.3.6 getFieldPositon()

```
sf::Vector2i Board::getFieldPositon (
             unsigned int id,
             sf::Vector2i prevPos,
             unsigned int x,
             unsigned int y )
```

Getter for the position of a specific field on the board.

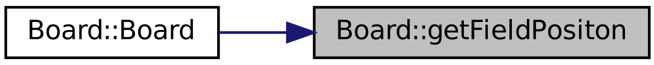
Parameters

id	The unique identifier of the field.
prevPos	The previous position of the field.
x	The horizontal position of the field.
y	The vertical position of the field.

Returns

The position of the field as an sf::Vector2i.

Here is the caller graph for this function:



6.4.3.7 getFieldRotation()

```
float Board::getFieldRotation (
             unsigned int id )
```

Getter for the rotation angle of a specific field on the board.

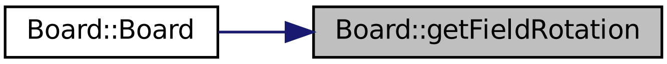
Parameters

id	The unique identifier of the field.

Returns

The rotation angle of the field.

Here is the caller graph for this function:



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

- /home/kamil/zpr/Monopoly/Board.h
- /home/kamil/zpr/Monopoly/Board.cc

6.5 Button Class Reference

Represents a button for handling user interactions.

```
#include <Button.h>
```

Public Member Functions

- Button (ScreenEventType type, sf::String btnText, sf::Vector2f buttonSize, int charSize)

    Constructor for the Button class.
- void draw (sf::RenderWindow &window)

    Draws the button on the specified window.
- bool isMouseOver (sf::RenderWindow &window)

    Checks if the mouse is over the button.
- virtual void mouseIsOver ()

    Virtual function called when the mouse is over the button.
- virtual void mouseIsNotOver ()

    Virtual function called when the mouse is not over the button.
- ScreenEventType getEventType ()

    Gets the type of screen event associated with the button.
- void setEventType (ScreenEventType event)

    Sets the type of screen event associated with the button.
- void setFont (sf::Font &fonts)

    Sets the font for the button text.
- void setPosition (sf::Vector2f point)

    Sets the position of the button.
- void setIsClicked (bool state)

    Sets the state of the button as clicked or not clicked.
- bool getIsClicked ()

    Gets the state of the button (clicked or not clicked).
- sf::Vector2f getSize ()

    Gets the size of the button.

- void setIsVisible (bool state)

  Sets the visibility state of the button.
- bool getIsVisible ()

  Gets the visibility state of the button.
- void setIsActive (bool state)

  Sets the active state of the button.
- bool getIsActive ()

  Gets the active state of the button.
- void setIsFocus (bool state)

  Sets the focus state of the button.
- bool getIsFocus ()

  Gets the focus state of the button.
- void setWasReleased (bool state)

  Sets the state of the button as released or not released.
- bool getWasReleased ()

  Gets the state of the button (released or not released).
- sf::Text & getText ()

  Gets the SFML Text object associated with the button.
- void setActiveBackColor (sf::Color color)

  Sets the background color and text color when the button is active.
- void setActiveTextColor (sf::Color color)

  Sets the text color when the button is active.
- void setInactiveBackColor (sf::Color color)

  Sets the background color and text color when the button is inactive.
- void setInactiveTextColor (sf::Color color)

  Sets the text color when the button is inactive.
- void setFocusBackColor (sf::Color color)

  Sets the background color and text color when the button is in focus.
- void setFocusTextColor (sf::Color color)

  Sets the text color when the button is in focus.
- void setButtonFocus ()

  Sets the button state to focused.
- void setButtonUnfocus ()

  Sets the button state to unfocused.
- void updateColors ()

  Updates the colors of the button based on its state.

6.5.1 Detailed Description

Represents a button for handling user interactions.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Button()

```
Button::Button (
            ScreenEventType type,
            sf::String btnText,
            sf::Vector2f buttonSize,
            int charSize )
```

Constructor for the Button class.

Parameters

type	The type of screen event associated with the button.
btnText	The text displayed on the button.
buttonSize	The size of the button.
charSize	The character size of the text.

6.5.3 Member Function Documentation

6.5.3.1 draw()

```
void Button::draw (
             sf::RenderWindow & window )
```

Draws the button on the specified window.

Parameters

window	The SFML window to draw the button on.

6.5.3.2 getEventType()

```
ScreenEventType Button::getEventType ( )
```

Gets the type of screen event associated with the button.

Returns

The screen event type.

6.5.3.3 getIsActive()

```
bool Button::getIsActive ( )
```

Gets the active state of the button.

Returns

True if the button is active, false otherwise.

6.5.3.4 getIsClicked()

```
bool Button::getIsClicked ( )
```

Gets the state of the button (clicked or not clicked).

Returns

True if the button is clicked, false otherwise.

6.5.3.5 getIsFocus()

```
bool Button::getIsFocus ( )
```

Gets the focus state of the button.

Returns

True if the button is in focus, false otherwise.

6.5.3.6 getIsVisible()

```
bool Button::getIsVisible ( )
```

Gets the visibility state of the button.

Returns

True if the button is visible, false otherwise.

6.5.3.7 getSize()

```
sf::Vector2f Button::getSize ( )
```

Gets the size of the button.

Returns

The size of the button.

6.5.3.8 getText()

```
sf::Text & Button::getText ( )
```

Gets the SFML Text object associated with the button.

Returns

The SFML Text object.

6.5.3.9 getWasReleased()

```
bool Button::getWasReleased ( )
```

Gets the state of the button (released or not released).

Returns

True if the button was released, false otherwise.

6.5.3.10 isMouseOver()

```
bool Button::isMouseOver (
             sf::RenderWindow & window )
```

Checks if the mouse is over the button.

Parameters

window	The SFML window.

Returns

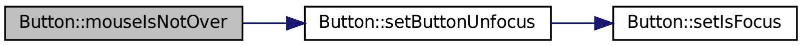
True if the mouse is over the button, false otherwise.

6.5.3.11 mouseIsNotOver()

```
void Button::mouseIsNotOver ( )  [virtual]
```

Virtual function called when the mouse is not over the button.

Here is the call graph for this function:

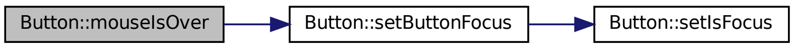


6.5.3.12 mouseIsOver()

```
void Button::mouseIsOver ( )  [virtual]
```

Virtual function called when the mouse is over the button.

Here is the call graph for this function:



6.5.3.13 setActiveBackColor()

```
void Button::setActiveBackColor (
             sf::Color color )
```

Sets the background color and text color when the button is active.

Parameters

color	The background color to set.

6.5.3.14 setActiveTextColor()

```
void Button::setActiveTextColor (
             sf::Color color )
```

Sets the text color when the button is active.

Parameters

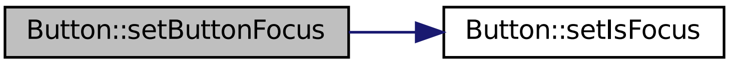
color	The text color to set.

6.5.3.15 setButtonFocus()

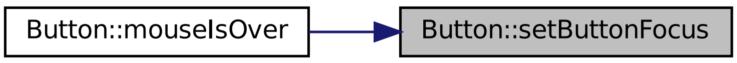
```
void Button::setButtonFocus ( )
```

Sets the button state to focused.

Here is the call graph for this function:



Here is the caller graph for this function:

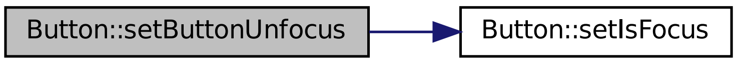


6.5.3.16 setButtonUnfocus()

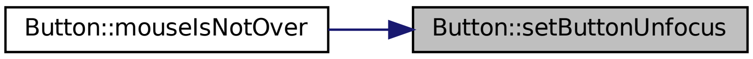
```
void Button::setButtonUnfocus ( )
```

Sets the button state to unfocused.

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3.17 setEventType()

```
void Button::setEventType (
             ScreenEventType event )
```

Sets the type of screen event associated with the button.

Parameters

event	The screen event type to set.

6.5.3.18 setFocusBackColor()

```
void Button::setFocusBackColor (
             sf::Color color )
```

Sets the background color and text color when the button is in focus.

Parameters

color	The background color to set.

6.5.3.19 setFocusTextColor()

```
void Button::setFocusTextColor (
             sf::Color color )
```

Sets the text color when the button is in focus.

Parameters

color	The text color to set.

6.5.3.20 setFont()

```
void Button::setFont (
            sf::Font & fonts )
```

Sets the font for the button text.

Parameters

fonts	The font to set.

6.5.3.21 setInactiveBackColor()

```
void Button::setInactiveBackColor (
            sf::Color color )
```

Sets the background color and text color when the button is inactive.

Parameters

color	The background color to set.

6.5.3.22 setInactiveTextColor()

```
void Button::setInactiveTextColor (
            sf::Color color )
```

Sets the text color when the button is inactive.

Parameters

color	The text color to set.

6.5.3.23 setIsActive()

```
void Button::setIsActive (
            bool state )
```

Sets the active state of the button.

Parameters

state	The active state to set.

6.5.3.24 setIsClicked()

```
void Button::setIsClicked (
             bool state )
```

Sets the state of the button as clicked or not clicked.

Parameters

state	The state to set.

6.5.3.25 setIsFocus()

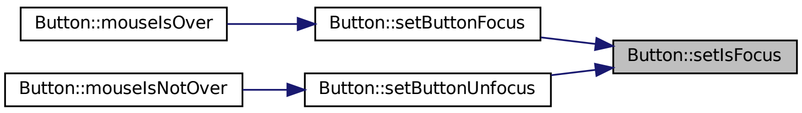
```
void Button::setIsFocus (
             bool state )
```

Sets the focus state of the button.

Parameters

state	The focus state to set.

Here is the caller graph for this function:



6.5.3.26 setIsVisible()

```
void Button::setIsVisible (
             bool state )
```

Sets the visibility state of the button.

Parameters

state	The visibility state to set.

6.5.3.27 setPosition()

```
void Button::setPosition (
             sf::Vector2f point )
```

Sets the position of the button.

Parameters

point	The position to set.

6.5.3.28 setWasReleased()

```
void Button::setWasReleased (
             bool state )
```

Sets the state of the button as released or not released.

Parameters

state	The state to set.

6.5.3.29 updateColors()

```
void Button::updateColors ( )
```

Updates the colors of the button based on its state.

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

- /home/kamil/zpr/Monopoly/Button.h
- /home/kamil/zpr/Monopoly/Button.cc

6.6 ChanceCard Class Reference

Represents a chance card in the monopoly game.

```
#include <Chance.h>
```

Public Member Functions

- ChanceCard (const unsigned int id, const ChanceType type, unsigned int value, const sf::String text, const std::string graphic_path, const unsigned int width, const unsigned int height, const sf::Vector2f position)

  Constructor for the ChanceCard class.
- unsigned int getId ()

  Getter for the unique identifier of the chance card.
- ChanceType getType ()

  Getter for the type of the chance card.
- int getValue ()

  Getter for the value associated with the chance card.
- const sf::String getText ()

  Getter for the text describing the action of the chance card.
- unsigned int getWidth ()

  Getter for the width of the graphic associated with the chance card.
- unsigned int getHeight ()

  Getter for the height of the graphic associated with the chance card.
- const std::string getGraphicPath ()

  Getter for the path to the graphic associated with the chance card.
- float getRotation ()

  Getter for the rotation of the chance card graphic.
- const sf::Sprite & getSprite ()

  Getter for the sprite associated with the chance card graphic.
- const sf::Texture & getTexture ()

  Getter for the texture associated with the chance card graphic.
- const sf::Vector2f & getPosition ()

  Getter for the position of the chance card on the screen.
- void setRotation (float new_roation)

  Setter for the rotation of the chance card graphic.
- void setPosition (sf::Vector2f pos)

  Setter for the position of the chance card on the screen.
- void createSprite ()

  Creates the sprite for the chance card using its graphic.

6.6.1 Detailed Description

Represents a chance card in the monopoly game.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 ChanceCard()

```
ChanceCard::ChanceCard (
             const unsigned int id,
             const ChanceType type,
             unsigned int value,
             const sf::String text,
             const std::string graphic_path,
             const unsigned int width,
             const unsigned int height,
             const sf::Vector2f position )
```

Constructor for the ChanceCard class.

Parameters

id	Unique identifier for the chance card.
type	Type of the chance card.
value	Value associated with the chance card.
text	Text describing the action of the chance card.
graphic_path	Path to the graphic associated with the chance card.
width	Width of the graphic.
height	Height of the graphic.
position	Position of the chance card on the screen.

6.6.3 Member Function Documentation

6.6.3.1 createSprite()

```
void ChanceCard::createSprite ( )
```

Creates the sprite for the chance card using its graphic.

6.6.3.2 getGraphicPath()

```
const std::string ChanceCard::getGraphicPath ( )
```

Getter for the path to the graphic associated with the chance card.

Returns

const std::string& Path to the graphic.

6.6.3.3 getHeight()

```
unsigned int ChanceCard::getHeight ( )
```

Getter for the height of the graphic associated with the chance card.

Returns

unsigned int Height of the graphic.

6.6.3.4 getId()

```
unsigned int ChanceCard::getId ( )
```

Getter for the unique identifier of the chance card.

Returns

unsigned int Unique identifier for the chance card.

6.6.3.5 getPosition()

```
const sf::Vector2f & ChanceCard::getPosition ( )
```

Getter for the position of the chance card on the screen.

Returns

const sf::Vector2f& Position of the chance card.

6.6.3.6 getRotation()

```
float ChanceCard::getRotation ( )
```

Getter for the rotation of the chance card graphic.

Returns

float Rotation of the chance card graphic.

6.6.3.7 getSprite()

```
const sf::Sprite & ChanceCard::getSprite ( )
```

Getter for the sprite associated with the chance card graphic.

Returns

const sf::Sprite& Sprite of the chance card graphic.

6.6.3.8 getText()

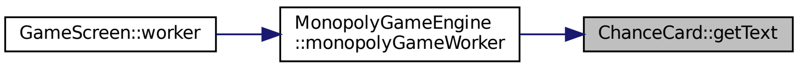
```
const sf::String ChanceCard::getText ( )
```

Getter for the text describing the action of the chance card.

Returns

const sf::String& Text describing the action of the chance card.

Here is the caller graph for this function:



6.6.3.9 getTexture()

```
const sf::Texture & ChanceCard::getTexture ( )
```

Getter for the texture associated with the chance card graphic.

Returns

const sf::Texture& Texture of the chance card graphic.

6.6.3.10 getType()

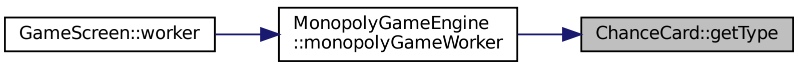
```
ChanceType ChanceCard::getType ( )
```

Getter for the type of the chance card.

Returns

ChanceType Type of the chance card.

Here is the caller graph for this function:



6.6.3.11 getValue()

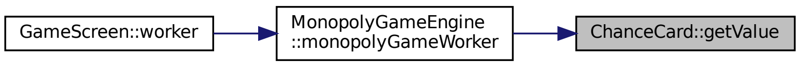
```
int ChanceCard::getValue ( )
```

Getter for the value associated with the chance card.

Returns

int Value associated with the chance card.

Here is the caller graph for this function:



6.6.3.12 getWidth()

```
unsigned int ChanceCard::getWidth ( )
```

Getter for the width of the graphic associated with the chance card.

Returns

unsigned int Width of the graphic.

6.6.3.13 setPosition()

```
void ChanceCard::setPosition (
             sf::Vector2f pos )
```

Setter for the position of the chance card on the screen.

Parameters

pos	New position value.

6.6.3.14 setRotation()

```
void ChanceCard::setRotation (
            float new_roation )
```

Setter for the rotation of the chance card graphic.

Parameters

new_roation	New rotation value.

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

- /home/kamil/zpr/Monopoly/Chance.h
- /home/kamil/zpr/Monopoly/Chance.cc

6.7 ContextWindow Class Reference

Represents a Singleton class for handling SFML window operations.

```
#include <ContextWindow.h>
```

Public Member Functions

- ContextWindow (ContextWindow &other)=delete
  Deleted constructor for proper Singleton class implementation.
- void operator= (const ContextWindow &)=delete
  Deleted = operator for proper Singleton class implementation.
- void display ()
  Displays the contents of the window.
- void clear ()
  Clears the contents of the window.
- bool isOpen ()
  Checks if the window is open.
- sf::RenderWindow & getWindow ()
  Gets the SFML RenderWindow object.
- sf::View & getView ()
  Gets the SFML View object.

Static Public Member Functions

- static ContextWindow $*$ GetInstance ()
  Gets the pointer to the ContextWindow instance.

Public Attributes

- sf::RenderWindow window_
- sf::View view_

6.7.1 Detailed Description

Represents a Singleton class for handling SFML window operations.

The ContextWindow class is a Singleton class used mainly for handling SFML window operations between other classes.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ContextWindow()

```
ContextWindow::ContextWindow (
             ContextWindow & other )  [delete]
```

Deleted constructor for proper Singleton class implementation.

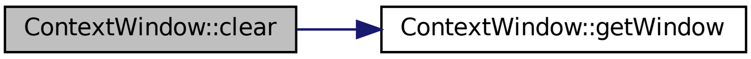
6.7.3 Member Function Documentation

6.7.3.1 clear()

```
void ContextWindow::clear ( )
```

Clears the contents of the window.

Here is the call graph for this function:

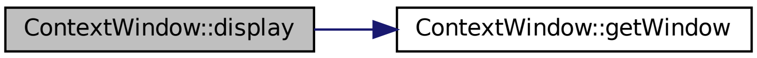


6.7.3.2 display()

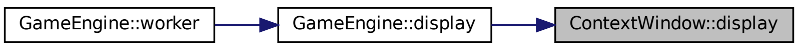
```
void ContextWindow::display ( )
```

Displays the contents of the window.

Here is the call graph for this function:



Here is the caller graph for this function:



6.7.3.3 GetInstance()

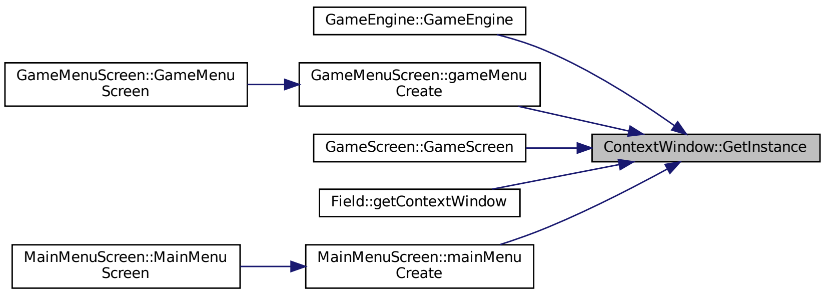
```
ContextWindow * ContextWindow::GetInstance ( )  [static]
```

Gets the pointer to the ContextWindow instance.

Returns

Pointer to the ContextWindow instance.

Here is the caller graph for this function:


GameEngine::GameEngine
GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create
GameScreen::GameScreen
ContextWindow::GetInstance
Field::getContextWindow
MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create

6.7.3.4 getView()

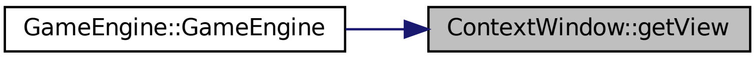
```
sf::View & ContextWindow::getView ( )
```

Gets the SFML View object.

Returns

Reference to the SFML View object.

Here is the caller graph for this function:



6.7.3.5 getWindow()

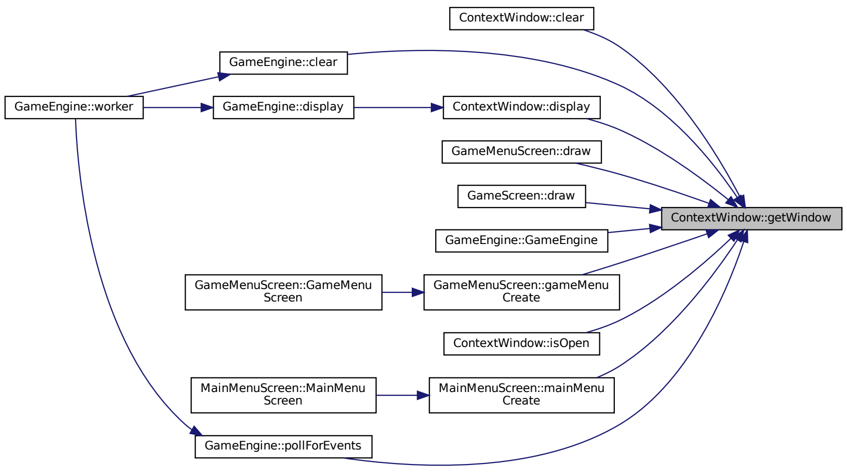
```
sf::RenderWindow & ContextWindow::getWindow ( )
```

Gets the SFML RenderWindow object.

Returns

Reference to the SFML RenderWindow object.

Here is the caller graph for this function:



6.7.3.6 isOpen()

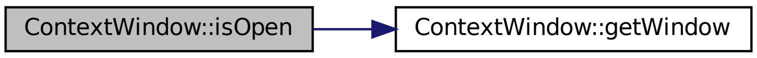
```
bool ContextWindow::isOpen ( )
```

Checks if the window is open.

Returns

True if the window is open, false otherwise.

Here is the call graph for this function:



6.7.3.7 operator=()

```
void ContextWindow::operator= (
             const ContextWindow & )  [delete]
```

Deleted = operator for proper Singleton class implementation.

6.7.4 Member Data Documentation

6.7.4.1 view_

```
sf::View ContextWindow::view_
```

SFML View object for defining a camera in the 2D scene.

6.7.4.2 window_

```
sf::RenderWindow ContextWindow::window_
```

SFML RenderWindow object for rendering graphics.

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

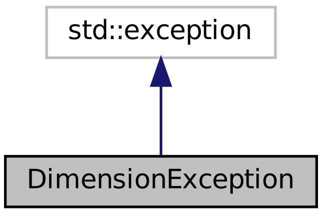
- /home/kamil/zpr/Monopoly/ContextWindow.h
- /home/kamil/zpr/Monopoly/ContextWindow.cc

6.8 DimensionException Class Reference

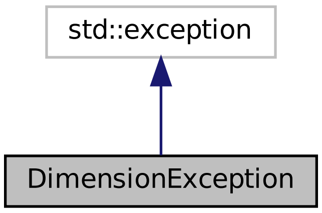
Exception for handling passing wrong dimensions to any displayed object.

`#include <main.h>`

Inheritance diagram for DimensionException:



Collaboration diagram for DimensionException:



Public Member Functions

- DimensionException (unsigned int dimension)
- DimensionException (const DimensionException &e) throw ()
- unsigned int getBadDimension ()

6.8.1 Detailed Description

Exception for handling passing wrong dimensions to any displayed object.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 DimensionException() `[1/2]`

```
DimensionException::DimensionException (
             unsigned int dimension )  [inline]
```

6.8.2.2 DimensionException() `[2/2]`

```
DimensionException::DimensionException (
             const DimensionException & e ) throw ( )   [inline]
```

6.8.3 Member Function Documentation

6.8.3.1 getBadDimension()

```
unsigned int DimensionException::getBadDimension ( )  [inline]
```

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

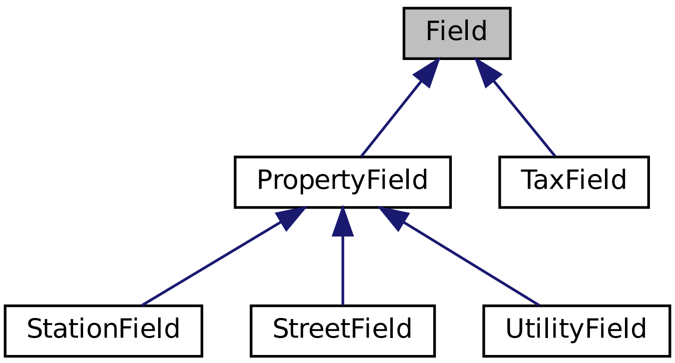
- /home/kamil/zpr/Monopoly/main.h

6.9 Field Class Reference

Base class representing a generic game field.

`#include <Field.h>`

Inheritance diagram for Field:



Public Member Functions

- Field (const unsigned int id, const FieldType type, const std::string name, const std::string graphic_path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position)

  Constructor for the Field class.
- ContextWindow ∗ getContextWindow ()

  Gets the pointer to the context window.
- unsigned int getId ()

  Gets the ID of the field.
- FieldType getType ()
- const std::string getName ()

  Gets the name of the field.
- const std::string getGraphicPath ()

  Gets the file path to the field's graphic.
- unsigned int getWidth ()

  Gets the width of the field.
- unsigned int getHeight ()

  Gets the height of the field.
- float getRotation ()

  Gets the rotation angle of the field.
- const sf::Sprite & getSprite ()

  Gets the sprite representing the field.
- const sf::Texture & getTexture ()

  Gets the texture of the field.
- const sf::Vector2i & getPosition ()

  Gets the position of the field on the board.
- const sf::Text & getNameText ()

  Gets the text representing the name of the field.
- void createSprite ()

  Creates the sprite for the field.
- void setHeight (unsigned int new_height)

  Sets the height of the field.
- void setWidth (unsigned int new_width)

  Sets the width of the field.
- void setRotation (float new_rotation)

  Sets the rotation angle of the field.
- void setPosition (sf::Vector2i pos)

  Sets the position of the field on the board.

6.9.1 Detailed Description

Base class representing a generic game field.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Field()

```
Field::Field (
             const unsigned int id,
             const FieldType type,
             const std::string name,
             const std::string graphic_path,
             const unsigned int width,
             const unsigned int height,
             const float rotation,
             const sf::Vector2i position )
```

Constructor for the Field class.

Parameters

id	The ID of the field.
type	The type of the field.
name	The name of the field.
graphic_path	The file path to the field's graphic.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the board.

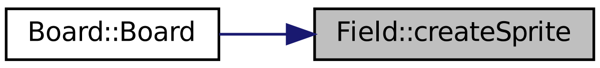
6.9.3 Member Function Documentation

6.9.3.1 createSprite()

```
void Field::createSprite ( )
```

Creates the sprite for the field.

Here is the caller graph for this function:



6.9.3.2 getContextWindow()

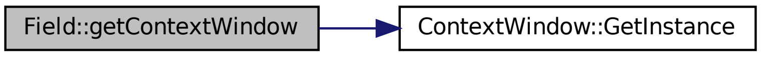
```
ContextWindow * Field::getContextWindow ( )
```

Gets the pointer to the context window.

Returns

A pointer to the context window.

Here is the call graph for this function:



6.9.3.3 getGraphicPath()

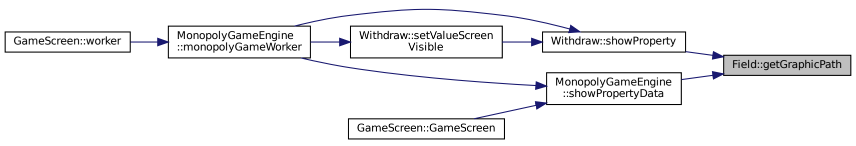
```
const std::string Field::getGraphicPath ( )
```

Gets the file path to the field's graphic.

Returns

The file path to the field's graphic.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setValueScreen Visible
Withdraw::showProperty
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen
Field::getGraphicPath

6.9.3.4 getHeight()

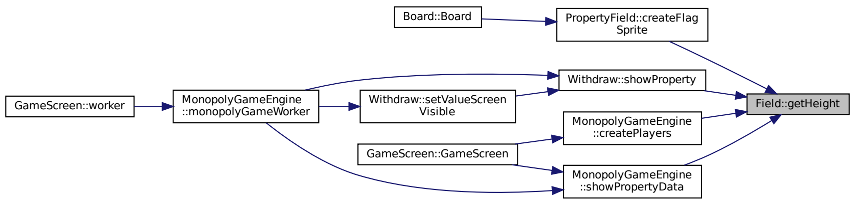
```
unsigned int Field::getHeight ( )
```

Gets the height of the field.

Returns

The height of the field.

Here is the caller graph for this function:


Board::Board
PropertyField::createFlag Sprite
Withdraw::showProperty
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setValueScreen Visible
Field::getHeight
MonopolyGameEngine ::createPlayers
GameScreen::GameScreen
MonopolyGameEngine ::showPropertyData

6.9.3.5 getId()

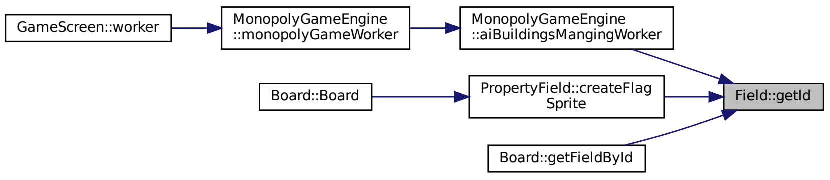
```
unsigned int Field::getId ( )
```

Gets the ID of the field.

Returns

The ID of the field.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
Board::Board
PropertyField::createFlag Sprite
Field::getId
Board::getFieldById

6.9.3.6 getName()

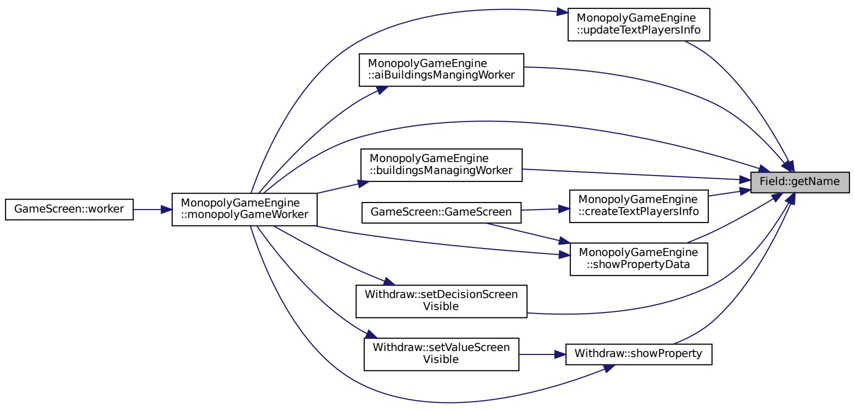
```
const std::string Field::getName ( )
```

Gets the name of the field.

Returns

The name of the field.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateTextPlayersInfo
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
GameScreen::GameScreen
MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::showPropertyData
Withdraw::setDecisionScreen Visible
Withdraw::setValueScreen Visible
Withdraw::showProperty
Field::getName

6.9.3.7 getNameText()

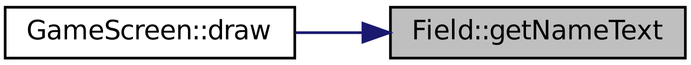
```
const sf::Text & Field::getNameText ( )
```

Gets the text representing the name of the field.

Returns

The text representing the name of the field.

Here is the caller graph for this function:



6.9.3.8 getPosition()

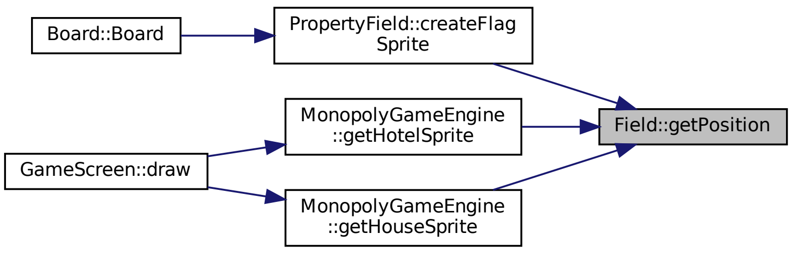
```
const sf::Vector2i & Field::getPosition ( )
```

Gets the position of the field on the board.

Returns

The position of the field on the board.

Here is the caller graph for this function:


Board::Board
PropertyField::createFlag Sprite
MonopolyGameEngine ::getHotelSprite
Field::getPosition
GameScreen::draw
MonopolyGameEngine ::getHouseSprite

6.9.3.9 getRotation()

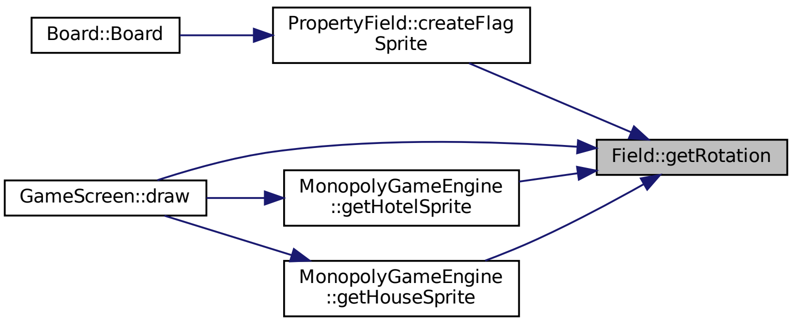
```
float Field::getRotation ( )
```

Gets the rotation angle of the field.

Returns

The rotation angle of the field.

Here is the caller graph for this function:


Board::Board
PropertyField::createFlag Sprite
Field::getRotation
GameScreen::draw
MonopolyGameEngine ::getHotelSprite
MonopolyGameEngine ::getHouseSprite

6.9.3.10 getSprite()

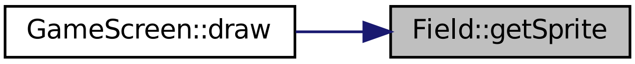
```
const sf::Sprite & Field::getSprite ( )
```

Gets the sprite representing the field.

Returns

The sprite representing the field.

Here is the caller graph for this function:



6.9.3.11 getTexture()

```
const sf::Texture & Field::getTexture ( )
```

Gets the texture of the field.

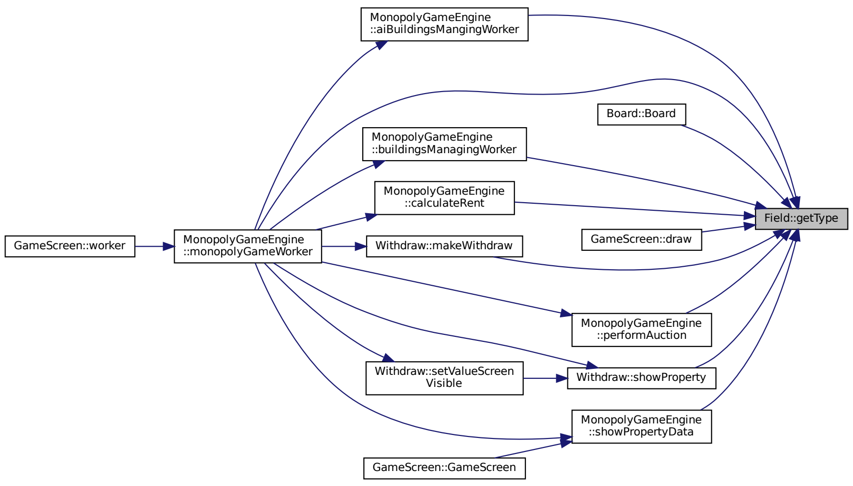
Returns

The texture of the field.

6.9.3.12 getType()

```
FieldType Field::getType ( )
```

Here is the caller graph for this function:


MonopolyGameEngine ::aiBuildingsMangingWorker
Board::Board
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::calculateRent
Field::getType
GameScreen::draw
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::makeWithdraw
MonopolyGameEngine ::performAuction
Withdraw::setValueScreen Visible
Withdraw::showProperty
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen

6.9.3.13 getWidth()

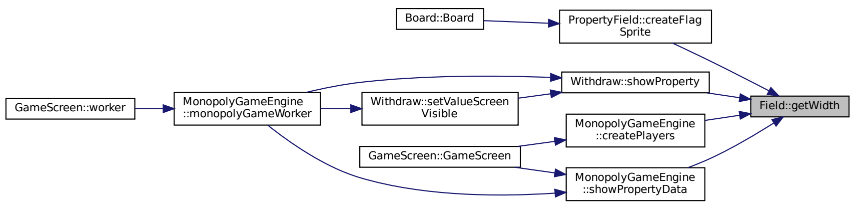
```
unsigned int Field::getWidth ( )
```

Gets the width of the field.

Returns

The width of the field.

Here is the caller graph for this function:


Board::Board
PropertyField::createFlagSprite
Withdraw::showProperty
GameScreen::worker
MonopolyGameEngine::monopolyGameWorker
Withdraw::setValueScreenVisible
Field::getWidth
MonopolyGameEngine::createPlayers
GameScreen::GameScreen
MonopolyGameEngine::showPropertyData

6.9.3.14 setHeight()

```
void Field::setHeight (
            unsigned int new_height )
```

Sets the height of the field.

Parameters

new_height	The new height of the field.

6.9.3.15 setPosition()

```
void Field::setPosition (
            sf::Vector2i pos )
```

Sets the position of the field on the board.

Parameters

pos	The new position of the field.

6.9.3.16 setRotation()

```
void Field::setRotation (
             float new_rotation )
```

Sets the rotation angle of the field.

Parameters

new_rotation	The new rotation angle of the field.

6.9.3.17 setWidth()

```
void Field::setWidth (
             unsigned int new_width )
```

Sets the width of the field.

Parameters

new_width	The new width of the field.

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

- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.10 GameEngine Class Reference

Handles low-level program operations, including input interactions and window display.

```
#include <GameEngine.h>
```

Public Member Functions

- GameEngine (double frame_rate_hz, uint window_width, uint window_height)

  Constructor for the GameEngine class.
- GameEngine (double frame_rate_hz)

  Additional constructor for the GameEngine class. Used only to train AI players.
- void clear ()

  Clears content on the displayed window.
- void display ()

Displays the content of the context window.
- void pollForEvents (sf::Event &event)

  Polls for events such as mouse and keyboard interactions.
- std::vector< std::shared_ptr< Player > > worker (std::vector< std::shared_ptr< Player >> &players_vec)

  Worker function for processing player-related tasks.
- unsigned int getwindow_width () const

  Gets the width of the game window.
- unsigned int getwindow_height () const

  Gets the height of the game window.
- ContextWindow ∗ getContextWindow ()

  Gets the pointer to the context window.

6.10.1 Detailed Description

Handles low-level program operations, including input interactions and window display.

The GameEngine class manages fundamental program operations, such as handling input interactions (mouse, keyboard) and displaying the window. It incorporates an ActiveScreen to manage various game screens. Also connects main monopolyGame engine with main loop in main.c to exchange data about results.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 GameEngine() [1/2]

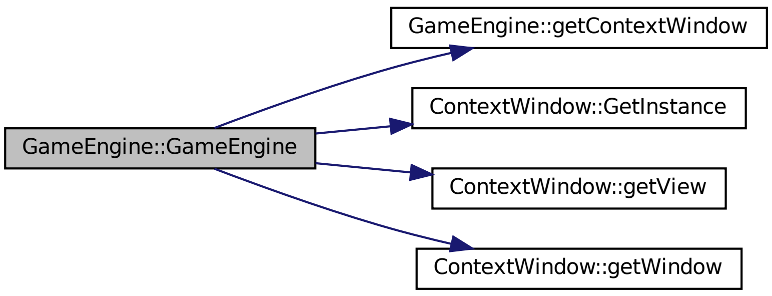
```
GameEngine::GameEngine (
            double frame_rate_hz,
            uint window_width,
            uint window_height )
```

Constructor for the GameEngine class.

Parameters

frame_rate_hz	Desired frame rate in Hertz (frames per second).
window_width	Width of the game window in pixels.
window_height	Height of the game window in pixels.

Here is the call graph for this function:



6.10.2.2 GameEngine() [2/2]

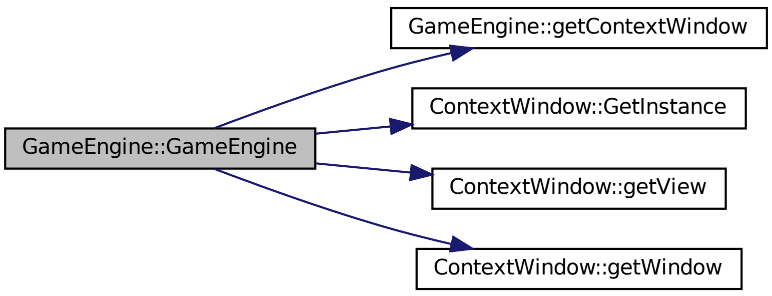
```
GameEngine::GameEngine (
             double frame_rate_hz )
```

Additional constructor for the GameEngine class. Used only to train AI players.

Parameters

frame_rate_hz	Desired frame rate in Hertz (frames per second).

Here is the call graph for this function:



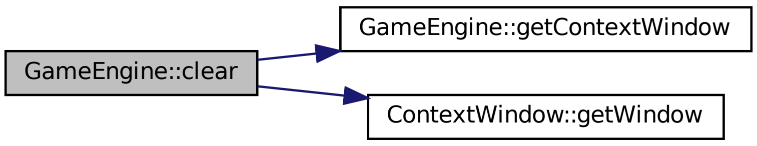
6.10.3 Member Function Documentation

6.10.3.1 clear()

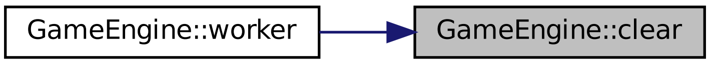
```
void GameEngine::clear ( )
```

Clears content on the displayed window.

Here is the call graph for this function:



Here is the caller graph for this function:

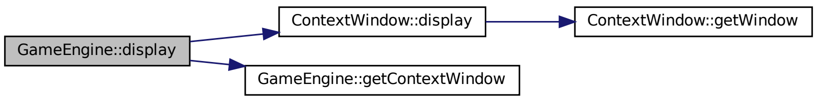


6.10.3.2 display()

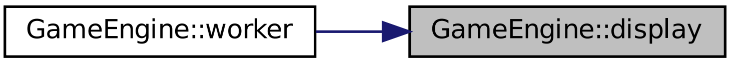
```
void GameEngine::display ( )
```

Displays the content of the context window.

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.3.3 getContextWindow()

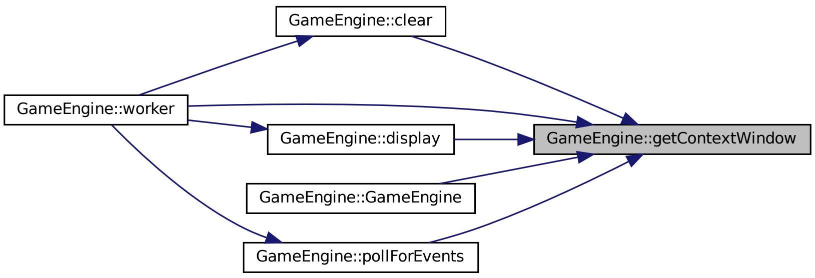
```
ContextWindow * GameEngine::getContextWindow ( )
```

Gets the pointer to the context window.

Returns

Pointer to the context window used for rendering graphics.

Here is the caller graph for this function:



6.10.3.4 getwindow_height()

```
unsigned int GameEngine::getwindow_height ( ) const
```

Gets the height of the game window.

Returns

Height of the game window in pixels.

6.10.3.5 getwindow_width()

```
unsigned int GameEngine::getwindow_width ( ) const
```

Gets the width of the game window.

Returns

Width of the game window in pixels.

6.10.3.6 pollForEvents()

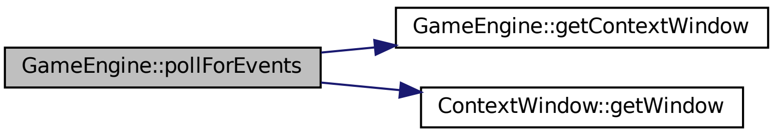
```
void GameEngine::pollForEvents (
            sf::Event & event )
```

Polls for events such as mouse and keyboard interactions.

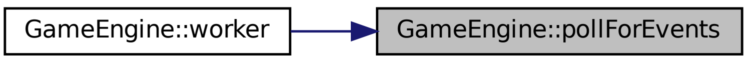
Parameters

event	Reference to an sf::Event object to store the polled event.

Here is the call graph for this function:



Here is the caller graph for this function:



6.10.3.7 worker()

```
std::vector< std::shared_ptr< Player > > GameEngine::worker (
            std::vector< std::shared_ptr< Player >> & players_vec )
```

Worker function for processing player-related tasks.

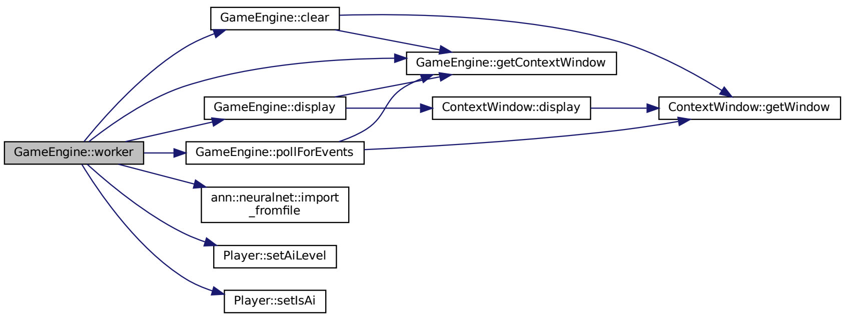
Parameters

players_vec	Vector of shared pointers to Player objects representing game players.

Returns

Vector of shared pointers to Player objects after processing.

Here is the call graph for this function:



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

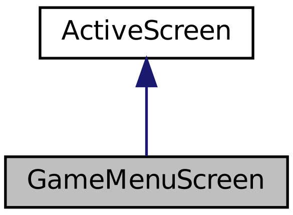
- /home/kamil/zpr/Monopoly/GameEngine.h
- /home/kamil/zpr/Monopoly/GameEngine.cc

6.11 GameMenuScreen Class Reference

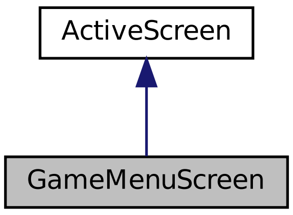
Represents the screen for the game menu.

```
#include <ActiveScreen.h>
```

Inheritance diagram for GameMenuScreen:



Collaboration diagram for GameMenuScreen:



Public Member Functions

- GameMenuScreen ()

    Constructor for the GameMenuScreen class.
- void gameMenuCreate ()

    Function to create the game menu.
- ScreenEventType worker ()

    Worker function for the game menu screen.
- void setPlayerSettings (unsigned int index, bool isNone, bool isHuman, int level)

    Sets the player settings.
- void buttonClickHandle (std::shared_ptr< Button > buttonPtr)

    Handles button clicks in the game menu.
- void setOtherButtonsInactive (std::shared_ptr< Button > buttonPtr)

    Sets other buttons inactive when a button is clicked.
- int getPlayerNumFromEventType (ScreenEventType event)

    Gets the player number from the ScreenEventType.
- void setAILevelColumnVisibility (int playerNum, bool visible)

    Sets the AI level column visibility.
- bool isEventTypeAILevel (int playerNum, ScreenEventType event)

Checks if the ScreenEventType is related to AI level.

- bool isEventTypeSetAI (int playerNum, ScreenEventType event)

  Checks if the ScreenEventType is related to setting AI.

- void setDefaultAILevelButtonsFocus (int playerNum)

  Sets the default AI level buttons' focus.

- std::vector< std::shared_ptr< playerSettings > > getPlayersSettings () const

  Gets the players' settings.

- void draw ()

  Draws the game menu screen.

6.11.1 Detailed Description

Represents the screen for the game menu.

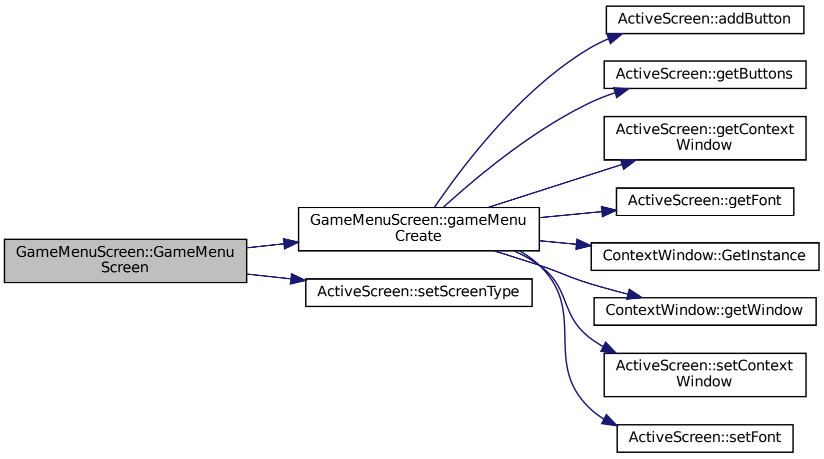
6.11.2 Constructor & Destructor Documentation

6.11.2.1 GameMenuScreen()

```
GameMenuScreen::GameMenuScreen ( )
```

Constructor for the GameMenuScreen class.

Here is the call graph for this function:


ActiveScreen::addButton
ActiveScreen::getButtons
ActiveScreen::getContext Window
ActiveScreen::getFont
GameMenuScreen::gameMenu Create
GameMenuScreen::GameMenu Screen
ContextWindow::GetInstance
ActiveScreen::setScreenType
ContextWindow::getWindow
ActiveScreen::setContext Window
ActiveScreen::setFont

6.11.3 Member Function Documentation

6.11.3.1 buttonClickHandle()

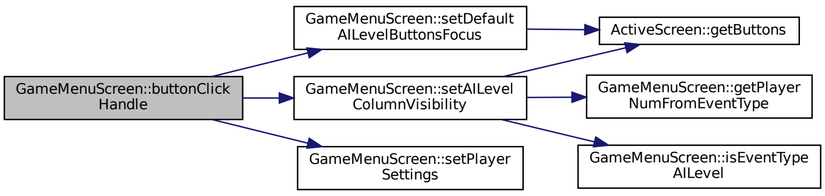
```
void GameMenuScreen::buttonClickHandle (
            std::shared_ptr< Button > buttonPtr )
```

Handles button clicks in the game menu.

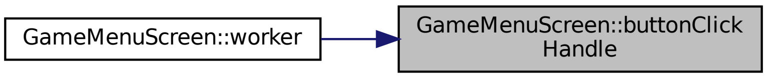
Parameters

buttonPtr	The shared pointer to the Button clicked.

Here is the call graph for this function:


GameMenuScreen::buttonClick Handle
GameMenuScreen::setDefault AILevelButtonsFocus
GameMenuScreen::setAILevel ColumnVisibility
GameMenuScreen::setPlayer Settings
ActiveScreen::getButtons
GameMenuScreen::getPlayer NumFromEventType
GameMenuScreen::isEventType AILevel

Here is the caller graph for this function:



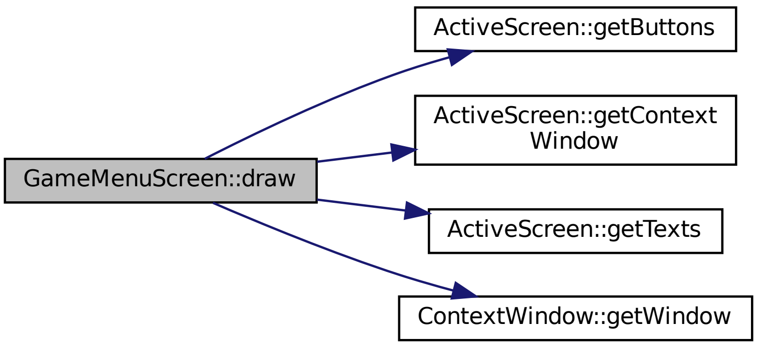
6.11.3.2 draw()

```
void GameMenuScreen::draw ( )  [virtual]
```

Draws the game menu screen.

Implements ActiveScreen.

Here is the call graph for this function:

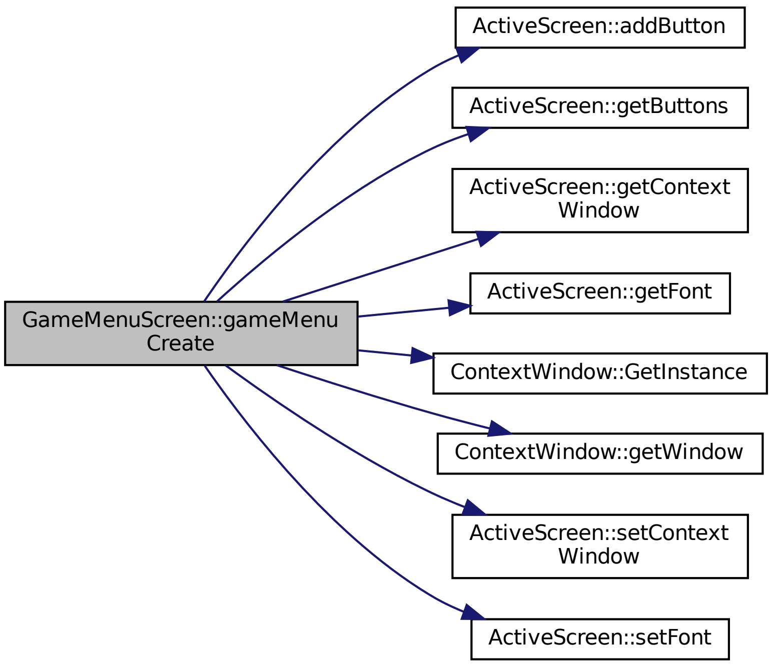


6.11.3.3 gameMenuCreate()

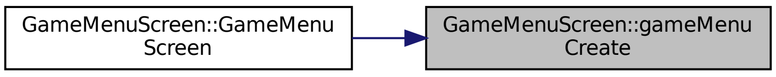
```
void GameMenuScreen::gameMenuCreate ( )
```

Function to create the game menu.

Here is the call graph for this function:


ActiveScreen::addButton
ActiveScreen::getButtons
ActiveScreen::getContext Window
ActiveScreen::getFont
GameMenuScreen::gameMenu Create
ContextWindow::GetInstance
ContextWindow::getWindow
ActiveScreen::setContext Window
ActiveScreen::setFont

Here is the caller graph for this function:


GameMenuScreen::GameMenu Screen
GameMenuScreen::gameMenu Create

6.11.3.4 getPlayerNumFromEventType()

```
int GameMenuScreen::getPlayerNumFromEventType (
            ScreenEventType event )
```

Gets the player number from the ScreenEventType.

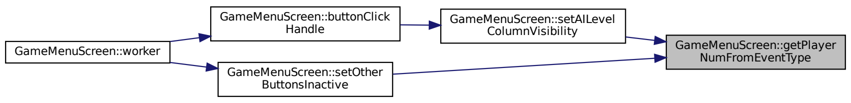
Parameters

event	The ScreenEventType to analyze.

Returns

The player number.

Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
GameMenuScreen::setOther ButtonsInactive
GameMenuScreen::setAILevel ColumnVisibility
GameMenuScreen::getPlayer NumFromEventType

6.11.3.5 getPlayersSettings()

```
std::vector< std::shared_ptr< playerSettings > > GameMenuScreen::getPlayersSettings ( ) const
[virtual]
```

Gets the players' settings.

Returns

Vector of shared pointers to playerSettings objects.

Reimplemented from ActiveScreen.

6.11.3.6 isEventTypeAILevel()

```
bool GameMenuScreen::isEventTypeAILevel (
            int playerNum,
            ScreenEventType event )
```

Checks if the ScreenEventType is related to AI level.

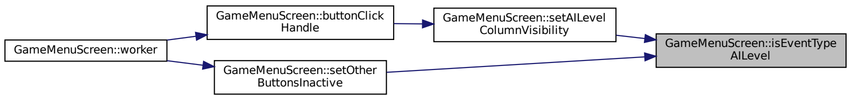
Parameters

playerNum	The player number.
event	The ScreenEventType to analyze.

Returns

True if the event is related to AI level, false otherwise.

Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
GameMenuScreen::setOther ButtonsInactive
GameMenuScreen::setAILevel ColumnVisibility
GameMenuScreen::isEventType AILevel

6.11.3.7 isEventTypeSetAI()

```
bool GameMenuScreen::isEventTypeSetAI (
             int playerNum,
             ScreenEventType event )
```

Checks if the ScreenEventType is related to setting AI.

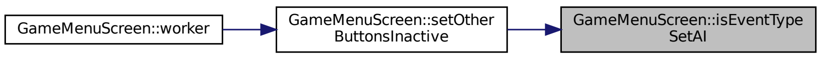
Parameters

playerNum	The player number.
event	The ScreenEventType to analyze.

Returns

True if the event is related to setting AI, false otherwise.

Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::setOther ButtonsInactive
GameMenuScreen::isEventType SetAI

6.11.3.8 setAILevelColumnVisibility()

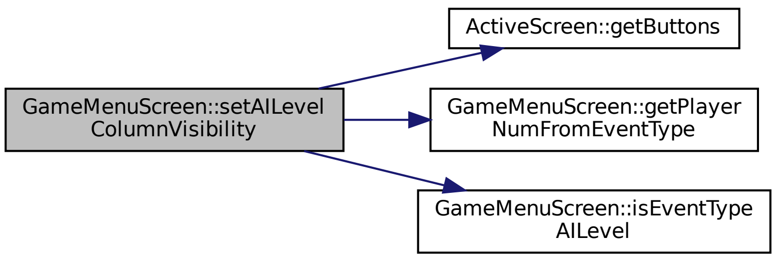
```
void GameMenuScreen::setAILevelColumnVisibility (
             int playerNum,
             bool visible )
```

Sets the AI level column visibility.

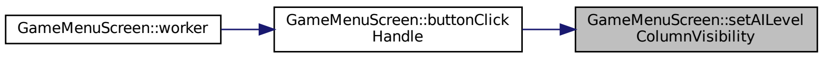
Parameters

playerNum	The player number.
visible	Flag indicating if the column should be visible.

Here is the call graph for this function:


ActiveScreen::getButtons
GameMenuScreen::setAILevel ColumnVisibility
GameMenuScreen::getPlayer NumFromEventType
GameMenuScreen::isEventType AILevel

Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
GameMenuScreen::setAILevel ColumnVisibility

6.11.3.9 setDefaultAILevelButtonsFocus()

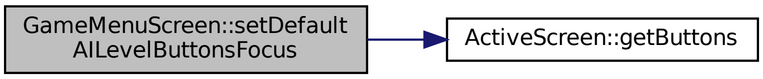
```
void GameMenuScreen::setDefaultAILevelButtonsFocus (
             int playerNum )
```

Sets the default AI level buttons' focus.

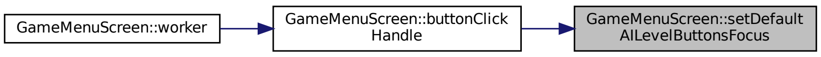
Parameters

playerNum	The player number.

Here is the call graph for this function:



Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
GameMenuScreen::setDefault AILevelButtonsFocus

6.11.3.10 setOtherButtonsInactive()

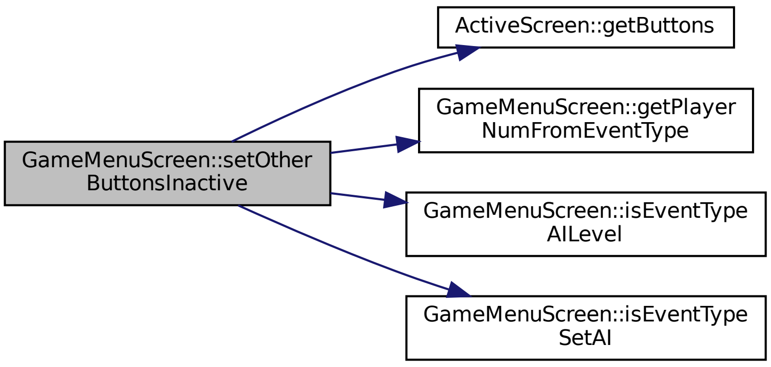
```
void GameMenuScreen::setOtherButtonsInactive (
            std::shared_ptr< Button > buttonPtr )
```

Sets other buttons inactive when a button is clicked.

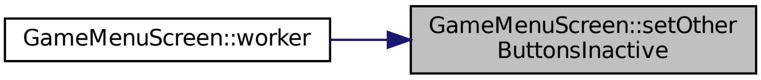
Parameters

buttonPtr	The shared pointer to the Button clicked.

Here is the call graph for this function:


GameMenuScreen::setOther ButtonsInactive
ActiveScreen::getButtons
GameMenuScreen::getPlayer NumFromEventType
GameMenuScreen::isEventType AILevel
GameMenuScreen::isEventType SetAI

Here is the caller graph for this function:



6.11.3.11 setPlayerSettings()

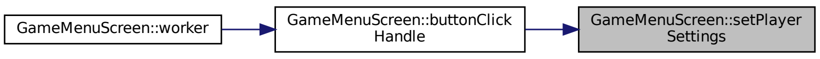
```
void GameMenuScreen::setPlayerSettings (
             unsigned int index,
             bool isNone,
             bool isHuman,
             int level )
```

Sets the player settings.

Parameters

index	The index of the player.
isNone	Flag indicating if the player is set to None.
isHuman	Flag indicating if the player is human.
level	The AI level of the player.

Here is the caller graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
GameMenuScreen::setPlayer Settings

6.11.3.12 worker()

```
ScreenEventType GameMenuScreen::worker ( ) [virtual]
```

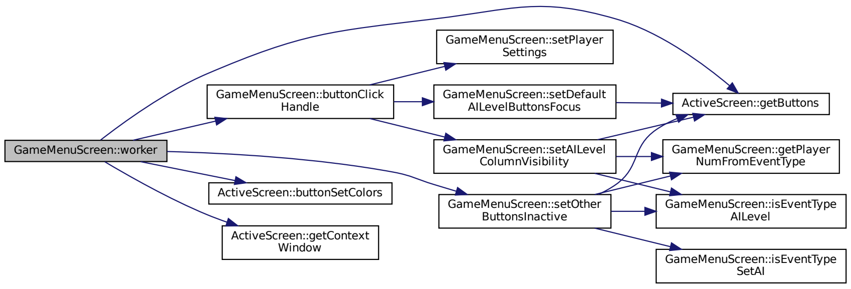
Worker function for the game menu screen.

Returns

The ScreenEventType associated with the user interaction.

Implements ActiveScreen.

Here is the call graph for this function:


GameMenuScreen::worker
GameMenuScreen::buttonClick Handle
ActiveScreen::buttonSetColors
ActiveScreen::getContext Window
GameMenuScreen::setPlayer Settings
GameMenuScreen::setDefault AILevelButtonsFocus
GameMenuScreen::setAILevel ColumnVisibility
GameMenuScreen::setOther ButtonsInactive
ActiveScreen::getButtons
GameMenuScreen::getPlayer NumFromEventType
GameMenuScreen::isEventType AILevel
GameMenuScreen::isEventType SetAI

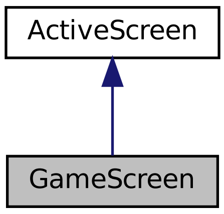
The documentation for this class was generated from the following files:

- /home/kamil/zpr/Monopoly/ActiveScreen.h
- /home/kamil/zpr/Monopoly/ActiveScreen.cc

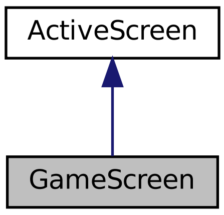
6.12 GameScreen Class Reference

```
#include <GameScreen.h>
```

Inheritance diagram for GameScreen:



Collaboration diagram for GameScreen:



Public Member Functions

- GameScreen (std::vector< std::shared_ptr< Player >> &players_)
- ScreenEventType worker ()

  Pure virtual function representing the worker function for the screen.
- void draw ()

  Pure virtual function to draw the screen.
- std::vector< std::shared_ptr< Player > > getPlayersResult ()

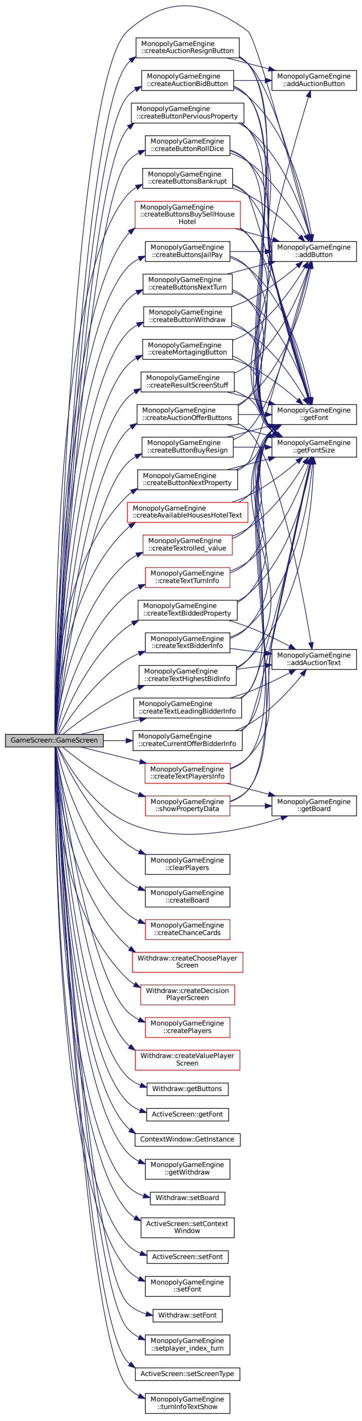
  Virtual function to get players' results.

6.12.1 Constructor & Destructor Documentation

6.12.1.1 GameScreen()

```
GameScreen::GameScreen (
            std::vector< std::shared_ptr< Player >> & players_ )
```

Here is the call graph for this function:



GameScreen::GameScreen
MonopolyGameEngine ::createAuctionResignButton
MonopolyGameEngine ::createAuctionBidButton
MonopolyGameEngine ::createButtonPerviousProperty
MonopolyGameEngine ::createButtonRollDice
MonopolyGameEngine ::createButtonsBankrupt
MonopolyGameEngine ::createButtonsBuySellHouse Hotel
MonopolyGameEngine ::createButtonsJailPay
MonopolyGameEngine ::createButtonsNextTurn
MonopolyGameEngine ::createButtonWithdraw
MonopolyGameEngine ::createMortagingButton
MonopolyGameEngine ::createResultScreenStuff
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::createButtonBuyResign
MonopolyGameEngine ::createButtonNextProperty
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::createTextrolled_value
MonopolyGameEngine ::createTextTurnInfo
MonopolyGameEngine ::createTextBiddedProperty
MonopolyGameEngine ::createTextBidderInfo
MonopolyGameEngine ::createTextHighestBidInfo
MonopolyGameEngine ::createTextLeadingBidderInfo
MonopolyGameEngine ::createCurrentOfferBidderInfo
MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::showPropertyData
MonopolyGameEngine ::addAuctionButton
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::clearPlayers
MonopolyGameEngine ::createBoard
MonopolyGameEngine ::createChanceCards
Withdraw::createChoosePlayer Screen
Withdraw::createDecision PlayerScreen
MonopolyGameEngine ::createPlayers
Withdraw::createValuePlayer Screen
Withdraw::getButtons
ActiveScreen::getFont
ContextWindow::GetInstance
MonopolyGameEngine ::getWithdraw
Withdraw::setBoard
ActiveScreen::setContext Window
ActiveScreen::setFont
MonopolyGameEngine ::setFont
Withdraw::setFont
MonopolyGameEngine ::setplayer_index_turn
ActiveScreen::setScreenType
MonopolyGameEngine ::turnInfoTextShow

6.12.2 Member Function Documentation

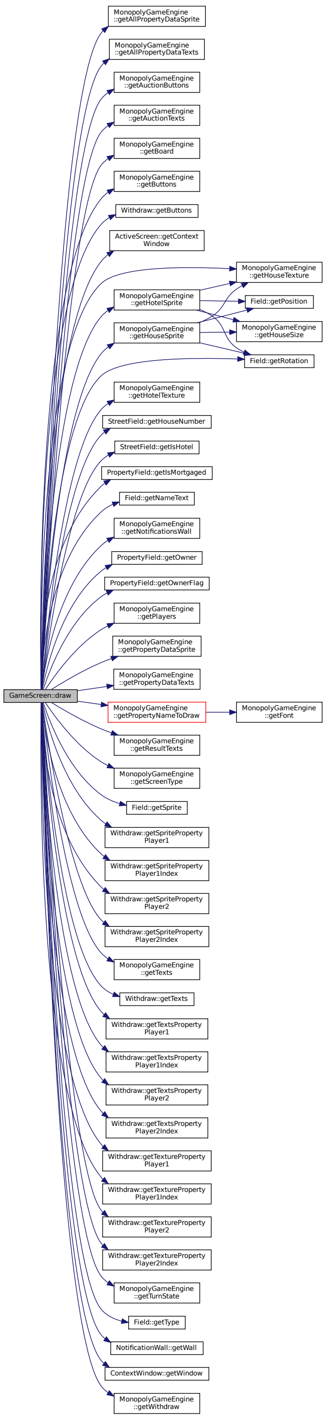
6.12.2.1 draw()

```
void GameScreen::draw ( )  [virtual]
```

Pure virtual function to draw the screen.

Implements ActiveScreen.

Here is the call graph for this function:


GameScreen::draw
MonopolyGameEngine ::getAllPropertyDataSprite
MonopolyGameEngine ::getAllPropertyDataTexts
MonopolyGameEngine ::getAuctionButtons
MonopolyGameEngine ::getAuctionTexts
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getButtons
Withdraw::getButtons
ActiveScreen::getContext Window
MonopolyGameEngine ::getHouseTexture
MonopolyGameEngine ::getHotelSprite
Field::getPosition
MonopolyGameEngine ::getHouseSprite
MonopolyGameEngine ::getHouseSize
Field::getRotation
MonopolyGameEngine ::getHotelTexture
StreetField::getHouseNumber
StreetField::getIsHotel
PropertyField::getIsMortgaged
Field::getNameText
MonopolyGameEngine ::getNotificationsWall
PropertyField::getOwner
PropertyField::getOwnerFlag
MonopolyGameEngine ::getPlayers
MonopolyGameEngine ::getPropertyDataSprite
MonopolyGameEngine ::getPropertyDataTexts
MonopolyGameEngine ::getPropertyNameToDraw
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getResultTexts
MonopolyGameEngine ::getScreenType
Field::getSprite
Withdraw::getSpriteProperty Player1
Withdraw::getSpriteProperty Player1Index
Withdraw::getSpriteProperty Player2
Withdraw::getSpriteProperty Player2Index
MonopolyGameEngine ::getTexts
Withdraw::getTexts
Withdraw::getTextsProperty Player1
Withdraw::getTextsProperty Player1Index
Withdraw::getTextsProperty Player2
Withdraw::getTextsProperty Player2Index
Withdraw::getTextureProperty Player1
Withdraw::getTextureProperty Player1Index
Withdraw::getTextureProperty Player2
Withdraw::getTextureProperty Player2Index
MonopolyGameEngine ::getTurnState
Field::getType
NotificationWall::getWall
ContextWindow::getWindow
MonopolyGameEngine ::getWithdraw

6.12.2.2 getPlayersResult()

```
std::vector< std::shared_ptr< Player > > GameScreen::getPlayersResult ( )  [virtual]
```

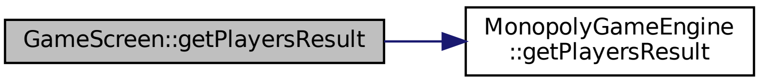
Virtual function to get players' results.

Returns

Vector of shared pointers to Player objects.

Reimplemented from ActiveScreen.

Here is the call graph for this function:


GameScreen::getPlayersResult
MonopolyGameEngine ::getPlayersResult

6.12.2.3 worker()

```
ScreenEventType GameScreen::worker ( )  [virtual]
```

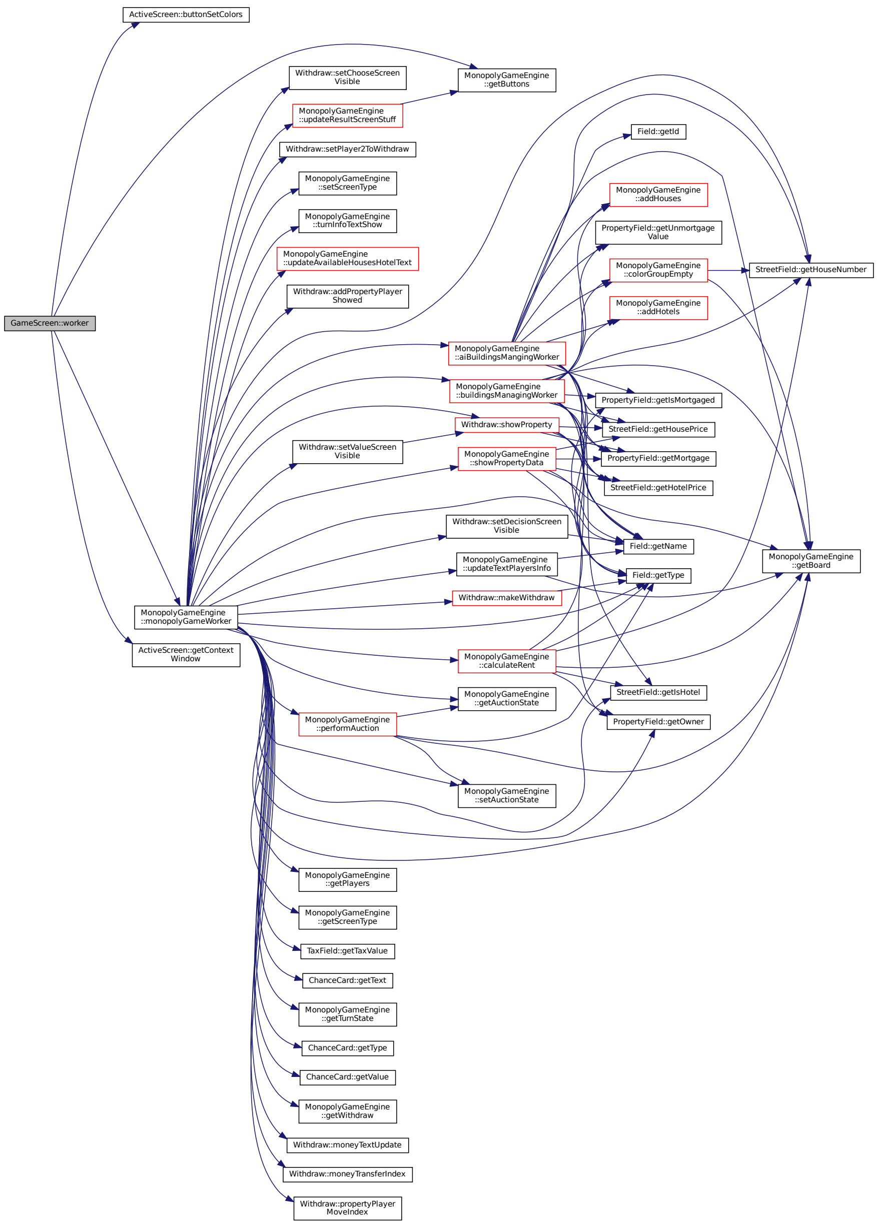
Pure virtual function representing the worker function for the screen.

Returns

The ScreenEventType associated with the user interaction.

Implements ActiveScreen.

Here is the call graph for this function:


ActiveScreen::buttonSetColors
Withdraw::setChooseScreen Visible
MonopolyGameEngine ::getButtons
MonopolyGameEngine ::updateResultScreenStuff
Withdraw::setPlayer2ToWithdraw
MonopolyGameEngine ::setScreenType
MonopolyGameEngine ::turnInfoTextShow
MonopolyGameEngine ::updateAvailableHousesHotelText
Withdraw::addPropertyPlayer Showed
GameScreen::worker
Field::getId
MonopolyGameEngine ::addHouses
PropertyField::getUnmortgage Value
MonopolyGameEngine ::colorGroupEmpty
StreetField::getHouseNumber
MonopolyGameEngine ::addHotels
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
PropertyField::getIsMortgaged
Withdraw::showProperty
StreetField::getHousePrice
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
PropertyField::getMortgage
StreetField::getHotelPrice
Withdraw::setDecisionScreen Visible
Field::getName
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::updateTextPlayersInfo
Field::getType
Withdraw::makeWithdraw
MonopolyGameEngine ::monopolyGameWorker
ActiveScreen::getContext Window
MonopolyGameEngine ::calculateRent
StreetField::getIsHotel
MonopolyGameEngine ::getAuctionState
MonopolyGameEngine ::performAuction
PropertyField::getOwner
MonopolyGameEngine ::setAuctionState
MonopolyGameEngine ::getPlayers
MonopolyGameEngine ::getScreenType
TaxField::getTaxValue
ChanceCard::getText
MonopolyGameEngine ::getTurnState
ChanceCard::getType
ChanceCard::getValue
MonopolyGameEngine ::getWithdraw
Withdraw::moneyTextUpdate
Withdraw::moneyTransferIndex
Withdraw::propertyPlayer MoveIndex

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

- /home/kamil/zpr/Monopoly/GameScreen.h
- /home/kamil/zpr/Monopoly/GameScreen.cc

6.13 neat::gene Struct Reference

```
#include <Tinyneat.h>
```

Public Attributes

- unsigned int innovation_num = -1
- unsigned int from_node = -1
- unsigned int to_node = -1
- double weight = 0.0
- bool enabled = true

6.13.1 Member Data Documentation

6.13.1.1 enabled

```
bool neat::gene::enabled = true
```

6.13.1.2 from_node

```
unsigned int neat::gene::from_node = -1
```

6.13.1.3 innovation_num

```
unsigned int neat::gene::innovation_num = -1
```

6.13.1.4 to_node

```
unsigned int neat::gene::to_node = -1
```

6.13.1.5 weight

```
double neat::gene::weight = 0.0
```

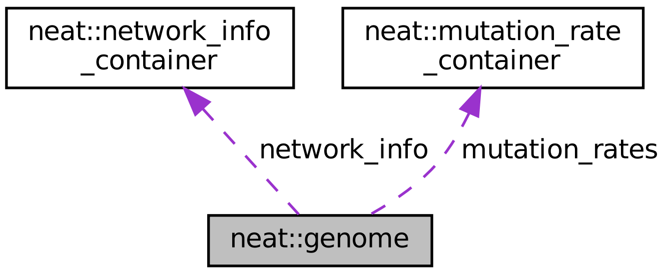
The documentation for this struct was generated from the following file:

- /home/kamil/zpr/Monopoly/Tinyneat.h

6.14 neat::genome Class Reference

```
#include <Tinyneat.h>
```

Collaboration diagram for neat::genome:



Public Member Functions

- genome (network_info_container &info, mutation_rate_container &rates)
- genome (const genome &)=default

Public Attributes

- unsigned int fitness = 0
- unsigned int adjusted_fitness = 0
- unsigned int global_rank = 0
- unsigned int max_neuron
- unsigned int can_be_recurrent = false
- mutation_rate_container mutation_rates
- network_info_container network_info
- std::map< unsigned int, gene > genes

6.14.1 Constructor & Destructor Documentation

6.14.1.1 genome() [1/2]

```
neat::genome::genome (
            network_info_container & info,
            mutation_rate_container & rates )
```

6.14.1.2 genome() [2/2]

```
neat::genome::genome (
            const genome & )  [default]
```

6.14.2 Member Data Documentation

6.14.2.1 adjusted_fitness

```
unsigned int neat::genome::adjusted_fitness = 0
```

6.14.2.2 can_be_recurrent

```
unsigned int neat::genome::can_be_recurrent = false
```

6.14.2.3 fitness

```
unsigned int neat::genome::fitness = 0
```

6.14.2.4 genes

```
std::map<unsigned int, gene> neat::genome::genes
```

6.14.2.5 global_rank

```
unsigned int neat::genome::global_rank = 0
```

6.14.2.6 max_neuron

```
unsigned int neat::genome::max_neuron
```

6.14.2.7 mutation_rates

```
mutation_rate_container neat::genome::mutation_rates
```

6.14.2.8 network_info

```
network_info_container neat::genome::network_info
```

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

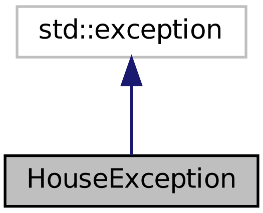
- /home/kamil/zpr/Monopoly/Tinyneat.h
- /home/kamil/zpr/Monopoly/Tinyneat.cc

6.15 HouseException Class Reference

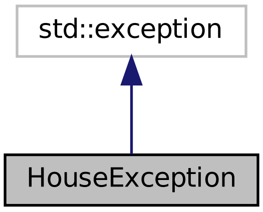
Custom exception class for handling invalid house numbers.

```
#include <Field.h>
```

Inheritance diagram for HouseException:



Collaboration diagram for HouseException:



Public Member Functions

- HouseException (unsigned int houses)
- HouseException (const HouseException &e) throw ()
- unsigned int getInvalidNumber ()

6.15.1 Detailed Description

Custom exception class for handling invalid house numbers.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 HouseException() `[1/2]`

```
HouseException::HouseException (
            unsigned int houses )   [inline]
```

6.15.2.2 HouseException() `[2/2]`

```
HouseException::HouseException (
            const HouseException & e ) throw ( )   [inline]
```

6.15.3 Member Function Documentation

6.15.3.1 getInvalidNumber()

```
unsigned int HouseException::getInvalidNumber ( )
```

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

- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.16 neat::innovation_container Class Reference

```
#include <Tinyneat.h>
```

Public Member Functions

- innovation_container ()
- void reset ()
- unsigned int add_gene (gene &g)
- unsigned int number ()

Friends

- class pool

6.16.1 Constructor & Destructor Documentation

6.16.1.1 innovation_container()

```
neat::innovation_container::innovation_container ( )  [inline]
```

6.16.2 Member Function Documentation

6.16.2.1 add_gene()

```
unsigned int neat::innovation_container::add_gene (
             gene & g )  [inline]
```

6.16.2.2 number()

```
unsigned int neat::innovation_container::number ( )  [inline]
```

6.16.2.3 reset()

```
void neat::innovation_container::reset ( )  [inline]
```

6.16.3 Friends And Related Function Documentation

6.16.3.1 pool

```
friend class pool  [friend]
```

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

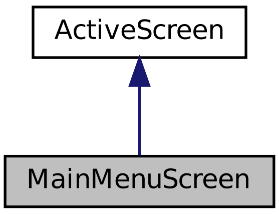
- /home/kamil/zpr/Monopoly/Tinyneat.h

6.17 MainMenuScreen Class Reference

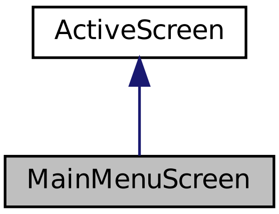
Represents the screen for the main menu.

```
#include <ActiveScreen.h>
```

Inheritance diagram for MainMenuScreen:



Collaboration diagram for MainMenuScreen:



Public Member Functions

- MainMenuScreen ()

  Constructor for the MainMenuScreen class.
- void mainMenuCreate ()

  Function to create the main menu.
- ScreenEventType worker ()

  Worker function for the main menu screen.
- void draw ()

  Draws the main menu screen.

6.17.1 Detailed Description

Represents the screen for the main menu.

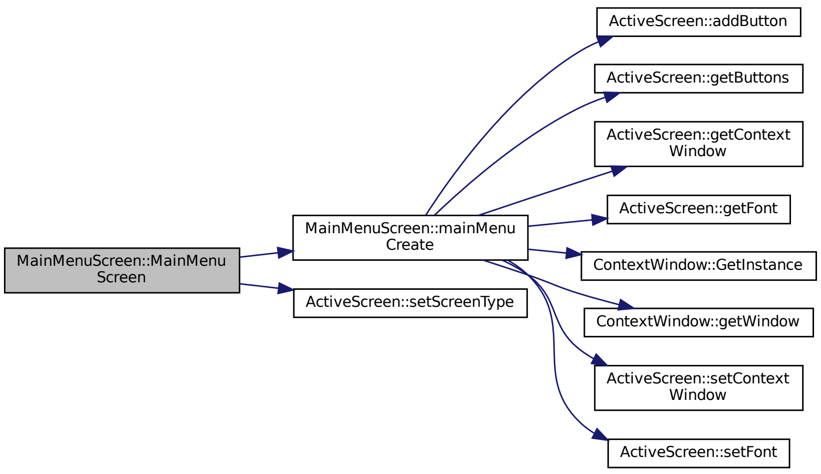
6.17.2 Constructor & Destructor Documentation

6.17.2.1 MainMenuScreen()

```
MainMenuScreen::MainMenuScreen ( )
```

Constructor for the MainMenuScreen class.

Here is the call graph for this function:


MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create
ActiveScreen::setScreenType
ActiveScreen::addButton
ActiveScreen::getButtons
ActiveScreen::getContext Window
ActiveScreen::getFont
ContextWindow::GetInstance
ContextWindow::getWindow
ActiveScreen::setContext Window
ActiveScreen::setFont

6.17.3 Member Function Documentation

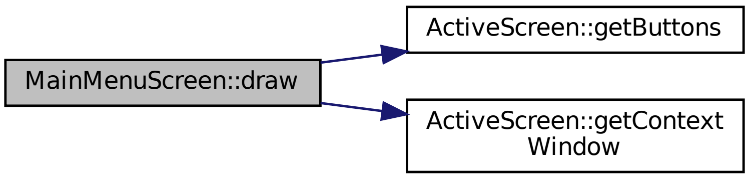
6.17.3.1 draw()

```
void MainMenuScreen::draw ( )  [virtual]
```

Draws the main menu screen.

Implements ActiveScreen.

Here is the call graph for this function:

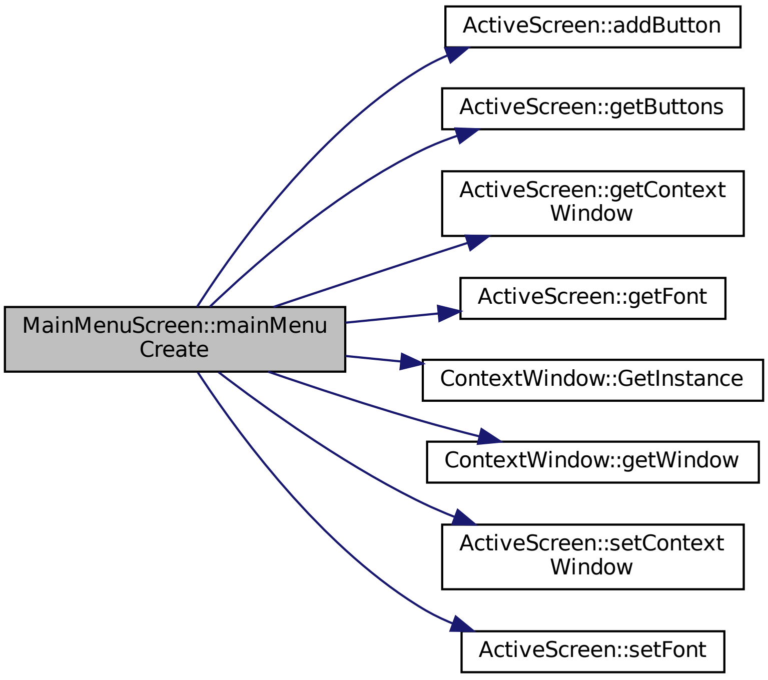


6.17.3.2 mainMenuCreate()

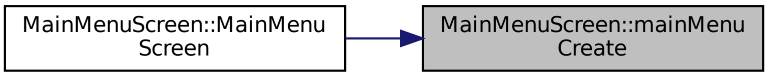
```
void MainMenuScreen::mainMenuCreate ( )
```

Function to create the main menu.

Here is the call graph for this function:


MainMenuScreen::mainMenu Create
ActiveScreen::addButton
ActiveScreen::getButtons
ActiveScreen::getContext Window
ActiveScreen::getFont
ContextWindow::GetInstance
ContextWindow::getWindow
ActiveScreen::setContext Window
ActiveScreen::setFont

Here is the caller graph for this function:


MainMenuScreen::MainMenu Screen
MainMenuScreen::mainMenu Create

6.17.3.3 worker()

```
ScreenEventType MainMenuScreen::worker ( )  [virtual]
```

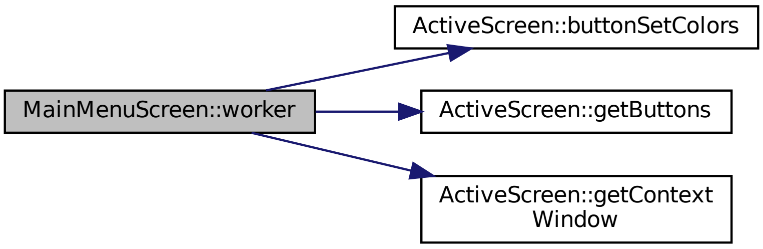
Worker function for the main menu screen.

Returns

The ScreenEventType associated with the user interaction.

Implements ActiveScreen.

Here is the call graph for this function:


ActiveScreen::buttonSetColors
MainMenuScreen::worker
ActiveScreen::getButtons
ActiveScreen::getContext Window

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

- /home/kamil/zpr/Monopoly/ActiveScreen.h
- /home/kamil/zpr/Monopoly/ActiveScreen.cc

6.18 MonopolyGameEngine Class Reference

Class representing the main game engine for the Monopoly game.

```
#include <MonopolyGameEngine.h>
```

Public Member Functions

- MonopolyGameEngine ()

    Constructor for the `monopolyGameEngine` class.
- void setScreenType (GameScreenType new_screen_type)

    Sets the screen type to the specified type.
- GameScreenType getScreenType () const

    Gets the current screen type.
- void createButtonRollDice ()

    Creates the button for rolling the dice.
- void createTextTurnInfo ()

    Creates text for displaying turn information.
- void createTextrolled_value ()

    Creates text for displaying the rolled dice value.
- void createTextPlayersInfo ()

    Creates text for displaying players' information.
- void updateTextPlayersInfo ()

    Updates the text displaying players' information.
- void createTextBiddedProperty ()

    Creates text for displaying information about the bidded property.
- void createTextBidderInfo ()

    Creates text for displaying information about the bidder.
- void createTextHighestBidInfo ()

    Creates text for displaying the highest bid in an auction.
- void createTextLeadingBidderInfo ()

    Creates text for displaying information about the leading bidder in an auction.
- void createCurrentOfferBidderInfo ()

    Creates text for displaying the current offer in an auction.
- void createButtonBuyResign ()

    Creates a button for buying or resigning from a property.
- void createButtonNextProperty ()

    Creates a button for moving to the next property.
- void createButtonPerviousProperty ()

    Creates a button for moving to the previous property.
- void createButtonsBuySellHouseHotel ()

    Creates buttons for buying, selling houses, and hotels.
- void createButtonsBankrupt ()

    Creates buttons for handling bankruptcy.
- void createButtonsNextTurn ()

    Creates buttons for moving to the next turn.
- void createButtonsJailPay ()

    Creates buttons for handling actions related to the jail (paying to get out).
- void createAuctionOfferButtons ()

    Creates buttons for handling auction offers.
- void createAuctionBidButton ()

    Creates a button for participating in an auction by bidding.
- void createAuctionResignButton ()

    Creates a button for resigning from an auction.
- void createButtonWithdraw ()

    Creates a button for withdrawing from the game.
- void createMortagingButton ()

Creates buttons for mortgaging properties.
- void createResultScreenStuff ()

  Creates various elements for the result screen.
- void updateResultScreenStuff ()

  Updates elements on the result screen.
- void showPropertyData (unsigned int pos, bool is_property_shown_to_buy)

  Shows property data on the GUI based on the position and whether it is shown to buy.
- void turnInfoTextShow ()

  Displays text related to turn information.
- sf::Font & getFont ()

  Returns the font used in the GUI.
- unsigned int getFontSize () const

  Returns the font size used in the GUI.
- void setFont (sf::Font font)

  Sets the font to be used in the GUI.
- void addButton (std::shared_ptr< Button > button_tmp)

  Adds a button to the GUI.
- void addText (std::shared_ptr< sf::Text > text_tmp)

  Adds text to the GUI.
- void addAuctionButton (std::shared_ptr< Button > button_tmp)

  Adds an auction button to the GUI.
- void addAuctionText (std::shared_ptr< sf::Text > text_tmp)

  Adds auction text to the GUI.
- std::vector< std::shared_ptr< Button > > & getButtons ()

  Returns a vector of pointers to buttons in the GUI.
- std::vector< std::shared_ptr< sf::Text > > & getTexts ()

  Returns a vector of pointers to text elements in the GUI.
- std::vector< std::shared_ptr< Button > > & getAuctionButtons ()

  Returns a vector of pointers to auction buttons in the GUI.
- std::vector< std::shared_ptr< sf::Text > > & getAuctionTexts ()

  Returns a vector of pointers to auction text elements in the GUI.
- std::vector< std::shared_ptr< sf::Text > > & getResultTexts ()

  Returns a vector of pointers to text elements used in the result screen.
- sf::Sprite & getPropertyDataSprite ()

  Returns the sprite representing property data in the GUI.
- std::vector< std::shared_ptr< sf::Text > > & getPropertyDataTexts ()

  Returns a vector of pointers to text elements representing property data in the GUI.
- sf::Sprite & getAllPropertyDataSprite ()

  Returns the sprite representing all property data in the GUI.
- std::vector< std::shared_ptr< sf::Text > > & getAllPropertyDataTexts ()

  Returns a vector of pointers to text elements representing all property data in the GUI.
- NotificationWall & getNotificationsWall ()

  Returns the notification wall used in the GUI.
- sf::Text getPropertyNameToDraw (sf::Text text, sf::Sprite &sprite, float rotation)

  Gets the text suitable for drawing a property name.
- sf::Texture & getHouseTexture ()

  Returns the texture of the house used in the GUI.
- sf::Texture & getHotelTexture ()

  Returns the texture of the hotel used in the GUI.
- sf::Vector2f & getHouseSize ()

  Returns the size of a house in the GUI.

- void createPlayers (std::vector< std::shared_ptr< Player >> &players_from_game_engine)

    Creates player objects based on input vector.
- void clearPlayers ()

    Clears player objects from the game.
- void createBoard ()

    Creates the game board.
- void clearBoard ()

    Clears the game board.
- std::shared_ptr< Board > getBoard ()

    Returns a pointer to the game board.
- std::vector< std::shared_ptr< Player > > & getPlayers ()

    Returns a vector of pointers to players in the game.
- std::vector< std::shared_ptr< Player > > getPlayersResult ()

    Returns a vector of pointers to players based on game result.
- void setplayer_index_turn (unsigned int indx)

    Sets the index of the player whose turn it is.
- TurnState getTurnState () const

    Gets the current turn state.
- void setAuctionState (AuctionState new_state)

    Sets the state of the auction.
- AuctionState getAuctionState ()

    Gets the current state of the auction.
- unsigned int getHouseCount ()

    Gets the count of houses available for purchase.
- unsigned int getHotelCount ()

    Gets the count of hotels available for purchase.
- void setHouseCount (unsigned int new_count)

    Sets the count of houses available for purchase.
- void setHotelCount (unsigned int new_count)

    Sets the count of hotels available for purchase.
- void addHouses (unsigned int added_amount)

    Adds houses to the available count.
- void substractHouses (unsigned int substracted_amount)

    Subtracts houses from the available count.
- void addHotels (unsigned int added_amount)

    Adds hotels to the available count.
- void substractHotels (unsigned int substracted_amount)

    Subtracts hotels from the available count.
- void performAuction ()

    Performs the auction.
- unsigned int calculateGroupFieldsOwned (std::vector< unsigned int > player_fields, PropertyField &field) const

    Calculates the number of group fields owned by a player.
- bool groupCompleted (std::vector< unsigned int > player_fields, PropertyField &field) const

    Checks if a group of properties is completed by a player.
- bool isBuildingLegal (std::shared_ptr< Player > builder, StreetField field)

    Checks if building houses on a street is legal for a player.
- bool isDestroyingLegal (std::shared_ptr< Player > builder, StreetField field)

    Checks if destroying houses on a street is legal for a player.
- bool isHotelBuildingLegal (std::shared_ptr< Player > builder, StreetField &field)

    Checks if building a hotel on a street is legal for a player.

- bool isHotelDestroyingLegal (std::shared_ptr< Player > builder, StreetField &field)
  Checks if destroying a hotel on a street is legal for a player.
- bool colorGroupEmpty (std::shared_ptr< Player > mortgaging, StreetField &field)
  Checks if a color group is empty (no properties owned) for a player.
- sf::Sprite getHouseSprite (StreetField &field, unsigned int houses_number)
  Gets the sprite for a house on a street.
- sf::Sprite getHotelSprite (StreetField &field)
  Gets the sprite for a hotel on a street.
- void createAvailableHousesHotelText ()
  Creates text displaying the available count of houses and hotels.
- void updateAvailableHousesHotelText ()
  Updates the text displaying the available count of houses and hotels.
- unsigned int calculateRent (unsigned int rolled_val, int pos)
  Calculates the rent to be paid based on the rolled dice value and property position.
- void buildingsManagingWorker ()
  Worker method for managing building and destroying houses/hotels.
- void aiBuildingsMangingWorker ()
  AI worker method for managing building and destroying houses/hotels.
- bool monopolyGameWorker ()
  Main Worker method for whole monopoly game engine.
- Withdraw & getWithdraw ()
  Returns a reference to the Withdraw object.
- void createChanceCards ()
  Creates the chance cards for the game.
- void shuffleChanceCards ()
  Shuffles the chance cards.

6.18.1 Detailed Description

Class representing the main game engine for the Monopoly game.

The MonopolyGameEngine class handles the overall game flow, including player turns, actions, and interactions with the game board. It also manages the graphical user interface (GUI) components.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 MonopolyGameEngine()

```
MonopolyGameEngine::MonopolyGameEngine ( )
```

Constructor for the `monopolyGameEngine` class.

6.18.3 Member Function Documentation

6.18.3.1 addAuctionButton()

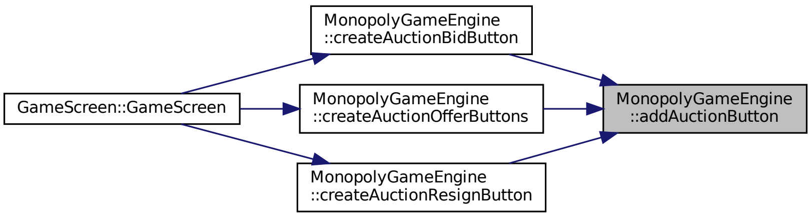
```
void MonopolyGameEngine::addAuctionButton (
            std::shared_ptr< Button > button_tmp )
```

Adds an auction button to the GUI.

Parameters

button_tmp	Pointer to the auction button to be added.

Here is the caller graph for this function:


MonopolyGameEngine ::createAuctionBidButton
GameScreen::GameScreen
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::addAuctionButton
MonopolyGameEngine ::createAuctionResignButton

6.18.3.2 addAuctionText()

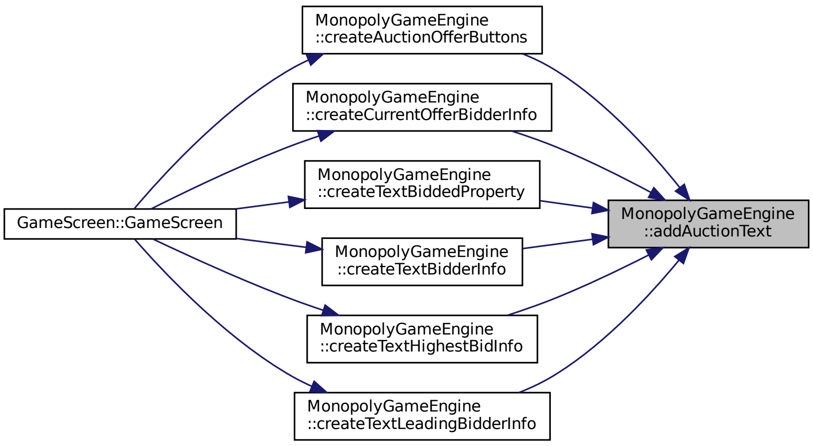
```
void MonopolyGameEngine::addAuctionText (
             std::shared_ptr< sf::Text > text_tmp )
```

Adds auction text to the GUI.

Parameters

text_tmp	Pointer to the auction text to be added.

Here is the caller graph for this function:


MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::createCurrentOfferBidderInfo
MonopolyGameEngine ::createTextBiddedProperty
GameScreen::GameScreen
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::createTextBidderInfo
MonopolyGameEngine ::createTextHighestBidInfo
MonopolyGameEngine ::createTextLeadingBidderInfo

6.18.3.3 addButton()

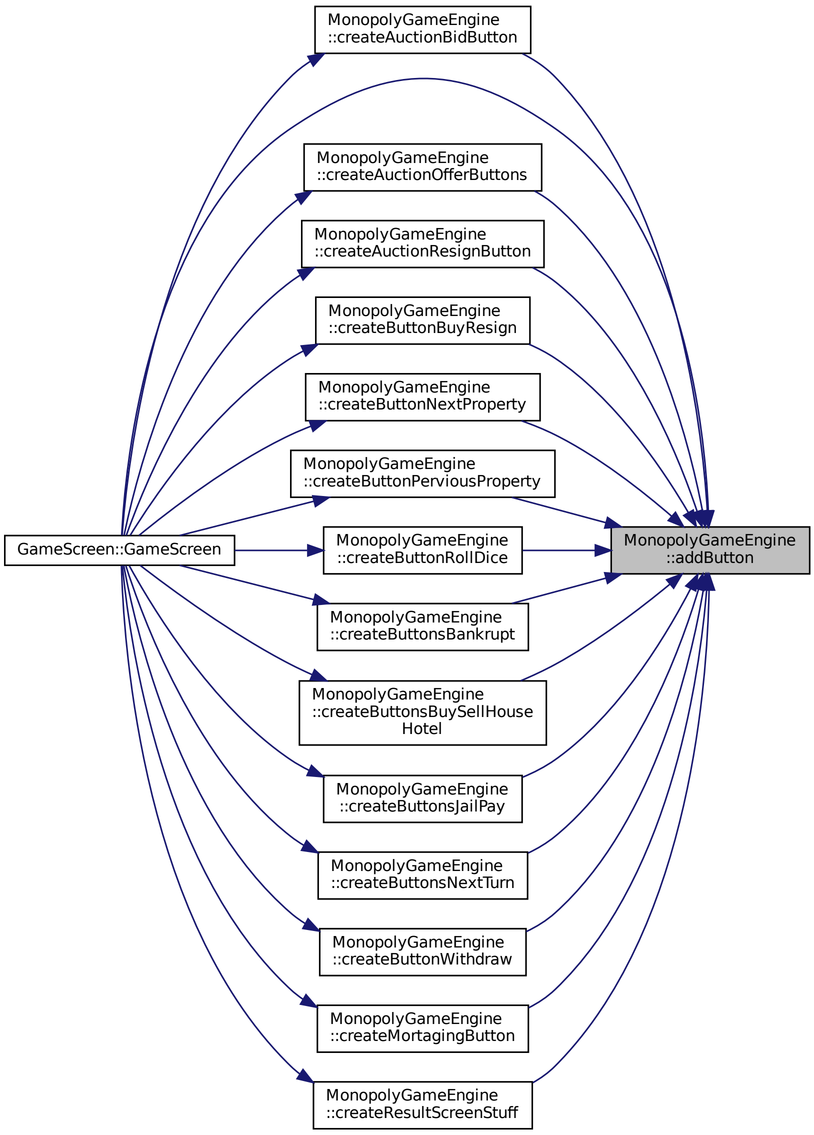
```
void MonopolyGameEngine::addButton (
             std::shared_ptr< Button > button_tmp )
```

Adds a button to the GUI.

Parameters

button_tmp	Pointer to the button to be added.

Here is the caller graph for this function:


MonopolyGameEngine ::createAuctionBidButton
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::createAuctionResignButton
MonopolyGameEngine ::createButtonBuyResign
MonopolyGameEngine ::createButtonNextProperty
MonopolyGameEngine ::createButtonPerviousProperty
GameScreen::GameScreen
MonopolyGameEngine ::createButtonRollDice
MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonsBankrupt
MonopolyGameEngine ::createButtonsBuySellHouse Hotel
MonopolyGameEngine ::createButtonsJailPay
MonopolyGameEngine ::createButtonsNextTurn
MonopolyGameEngine ::createButtonWithdraw
MonopolyGameEngine ::createMortagingButton
MonopolyGameEngine ::createResultScreenStuff

6.18.3.4 addHotels()

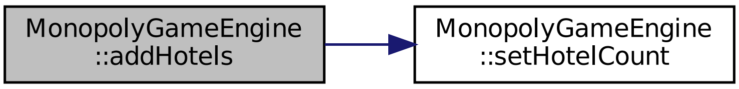
```
void MonopolyGameEngine::addHotels (
            unsigned int added_amount )
```

Adds hotels to the available count.

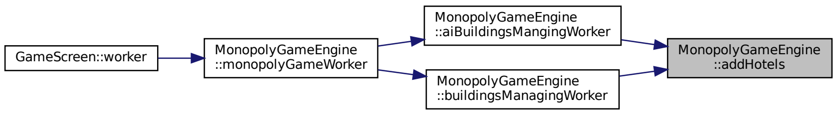
Parameters

added_amount	Amount to be added.

Here is the call graph for this function:


MonopolyGameEngine ::addHotels
MonopolyGameEngine ::setHotelCount

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHotels

6.18.3.5 addHouses()

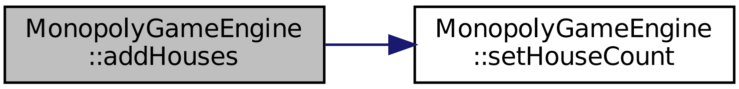
```
void MonopolyGameEngine::addHouses (
            unsigned int added_amount )
```

Adds houses to the available count.

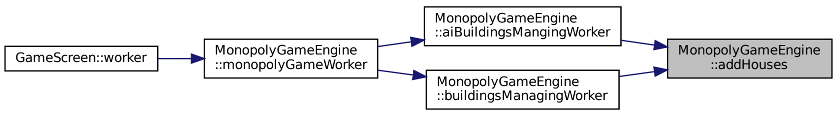
Parameters

added_amount	Amount to be added.

Here is the call graph for this function:


MonopolyGameEngine ::addHouses
MonopolyGameEngine ::setHouseCount

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHouses

6.18.3.6 addText()

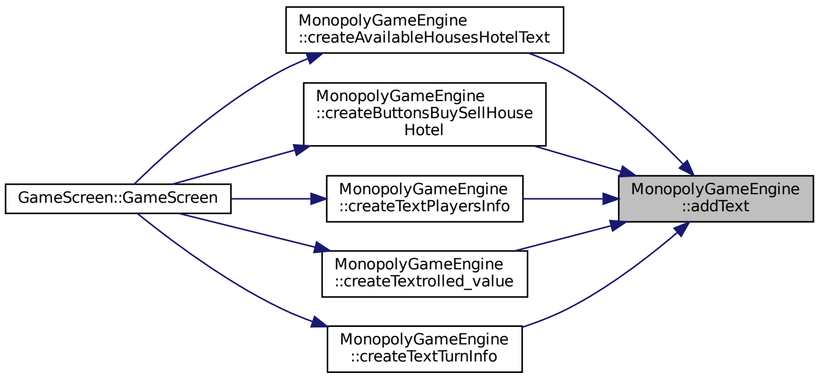
```
void MonopolyGameEngine::addText (
            std::shared_ptr< sf::Text > text_tmp )
```

Adds text to the GUI.

Parameters

text_tmp	Pointer to the text to be added.

Here is the caller graph for this function:

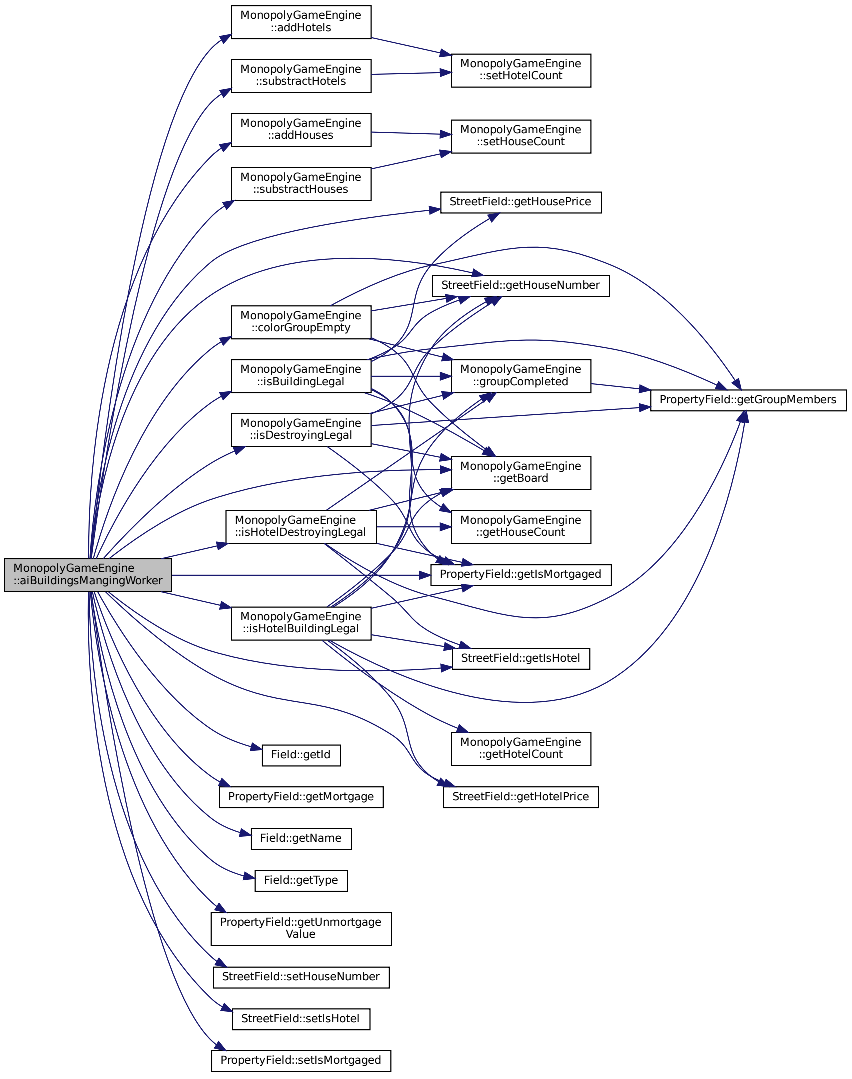

MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::createButtonsBuySellHouse Hotel
GameScreen::GameScreen
MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::addText
MonopolyGameEngine ::createTextrolled_value
MonopolyGameEngine ::createTextTurnInfo

6.18.3.7 aiBuildingsMangingWorker()

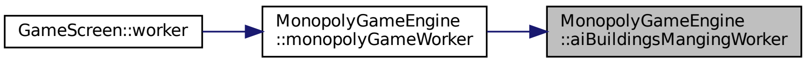
```
void MonopolyGameEngine::aiBuildingsMangingWorker ( )
```

AI worker method for managing building and destroying houses/hotels.

Here is the call graph for this function:


MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::addHotels
MonopolyGameEngine ::substractHotels
MonopolyGameEngine ::addHouses
MonopolyGameEngine ::substractHouses
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::isDestroyingLegal
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::setHotelCount
MonopolyGameEngine ::setHouseCount
StreetField::getHousePrice
StreetField::getHouseNumber
MonopolyGameEngine ::groupCompleted
PropertyField::getGroupMembers
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getHouseCount
PropertyField::getIsMortgaged
StreetField::getIsHotel
MonopolyGameEngine ::getHotelCount
StreetField::getHotelPrice
Field::getId
PropertyField::getMortgage
Field::getName
Field::getType
PropertyField::getUnmortgage Value
StreetField::setHouseNumber
StreetField::setIsHotel
PropertyField::setIsMortgaged

Here is the caller graph for this function:

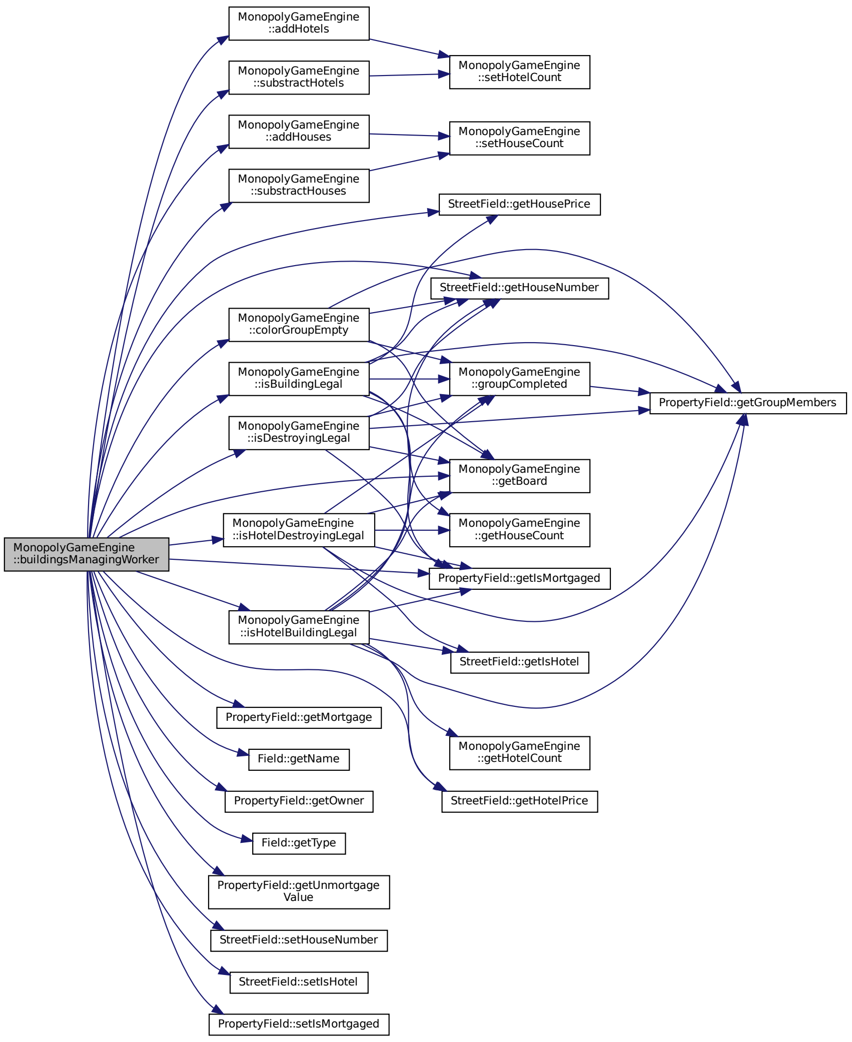

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker

6.18.3.8 buildingsManagingWorker()

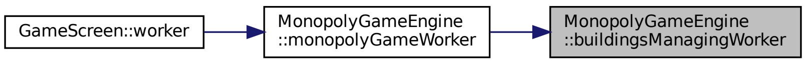
```
void MonopolyGameEngine::buildingsManagingWorker ( )
```

Worker method for managing building and destroying houses/hotels.

Here is the call graph for this function:


MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHotels
MonopolyGameEngine ::substractHotels
MonopolyGameEngine ::addHouses
MonopolyGameEngine ::substractHouses
MonopolyGameEngine ::setHotelCount
MonopolyGameEngine ::setHouseCount
StreetField::getHousePrice
StreetField::getHouseNumber
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::groupCompleted
PropertyField::getGroupMembers
MonopolyGameEngine ::isDestroyingLegal
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::getHouseCount
PropertyField::getIsMortgaged
MonopolyGameEngine ::isHotelBuildingLegal
StreetField::getIsHotel
PropertyField::getMortgage
Field::getName
PropertyField::getOwner
Field::getType
PropertyField::getUnmortgage Value
StreetField::setHouseNumber
StreetField::setIsHotel
PropertyField::setIsMortgaged
MonopolyGameEngine ::getHotelCount
StreetField::getHotelPrice

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker

6.18.3.9 calculateGroupFieldsOwned()

```
unsigned int MonopolyGameEngine::calculateGroupFieldsOwned (
            std::vector< unsigned int > player_fields,
            PropertyField & field ) const
```

Calculates the number of group fields owned by a player.

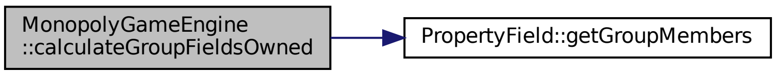
Parameters

player_fields	Vector of field positions owned by the player.
field	Property field to check for group completion.

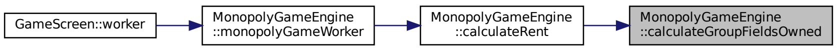
Returns

Number of group fields owned.

Here is the call graph for this function:


MonopolyGameEngine ::calculateGroupFieldsOwned
PropertyField::getGroupMembers

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::calculateGroupFieldsOwned

6.18.3.10 calculateRent()

```
unsigned int MonopolyGameEngine::calculateRent (
            unsigned int rolled_val,
            int pos )
```

Calculates the rent to be paid based on the rolled dice value and property position.

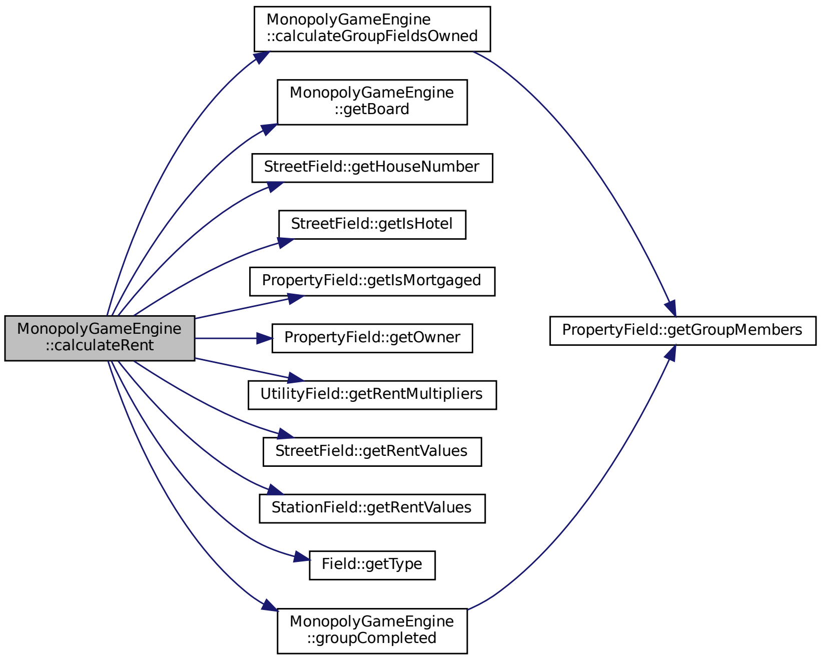
Parameters

rolled_val	Rolled dice value.
pos	Property position.

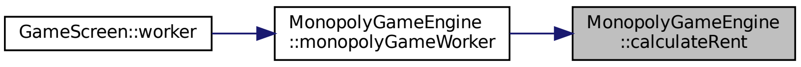
Returns

Calculated rent amount.

Here is the call graph for this function:


MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::calculateGroupFieldsOwned
MonopolyGameEngine ::getBoard
StreetField::getHouseNumber
StreetField::getIsHotel
PropertyField::getIsMortgaged
PropertyField::getOwner
UtilityField::getRentMultipliers
StreetField::getRentValues
StationField::getRentValues
Field::getType
MonopolyGameEngine ::groupCompleted
PropertyField::getGroupMembers

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::calculateRent

6.18.3.11 clearBoard()

```
void MonopolyGameEngine::clearBoard ( )
```

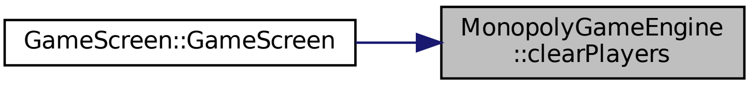
Clears the game board.

6.18.3.12 clearPlayers()

```
void MonopolyGameEngine::clearPlayers ( )
```

Clears player objects from the game.

Here is the caller graph for this function:



6.18.3.13 colorGroupEmpty()

```
bool MonopolyGameEngine::colorGroupEmpty (
            std::shared_ptr< Player > mortgaging,
            StreetField & field )
```

Checks if a color group is empty (no properties owned) for a player.

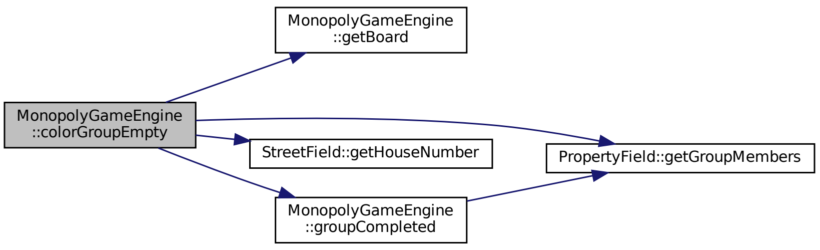
Parameters

mortgaging	Player attempting to mortgage properties.
field	Street field to check for a color group.

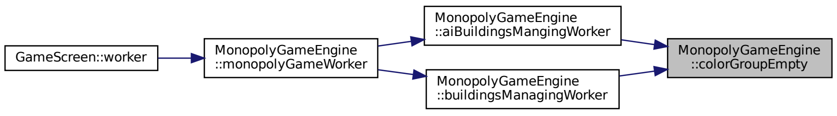
Returns

True if the color group is empty, false otherwise.

Here is the call graph for this function:


MonopolyGameEngine ::getBoard
MonopolyGameEngine ::colorGroupEmpty
StreetField::getHouseNumber
PropertyField::getGroupMembers
MonopolyGameEngine ::groupCompleted

Here is the caller graph for this function:

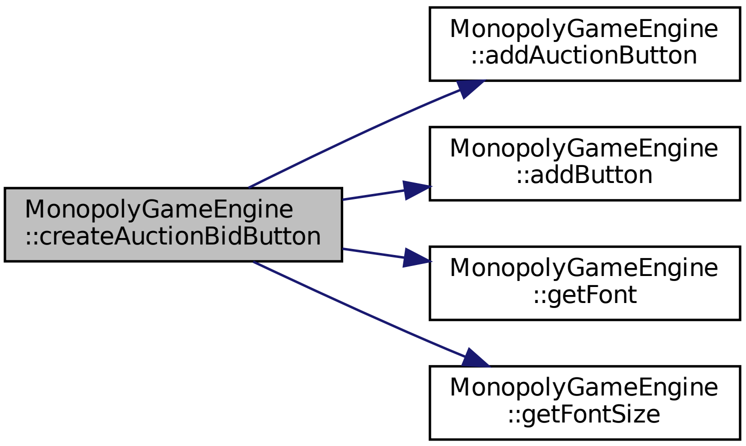

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::colorGroupEmpty

6.18.3.14 createAuctionBidButton()

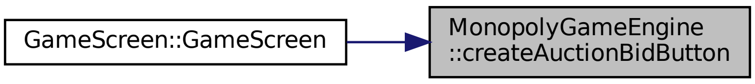
```
void MonopolyGameEngine::createAuctionBidButton ( )
```

Creates a button for participating in an auction by bidding.

Here is the call graph for this function:


MonopolyGameEngine ::createAuctionBidButton
MonopolyGameEngine ::addAuctionButton
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

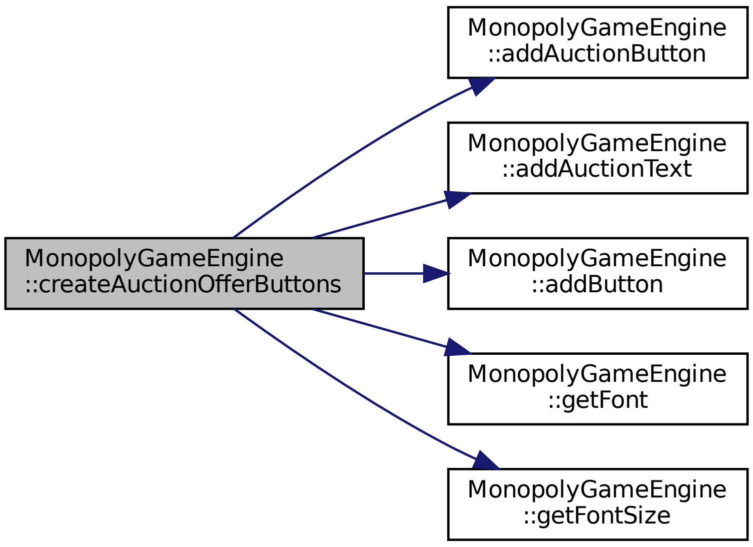


6.18.3.15 createAuctionOfferButtons()

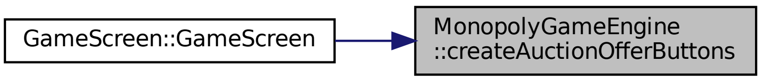
```
void MonopolyGameEngine::createAuctionOfferButtons ( )
```

Creates buttons for handling auction offers.

Here is the call graph for this function:


MonopolyGameEngine ::addAuctionButton
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

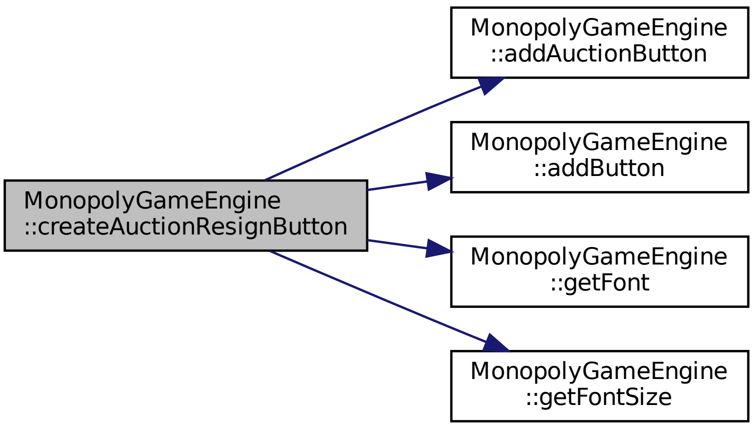


6.18.3.16 createAuctionResignButton()

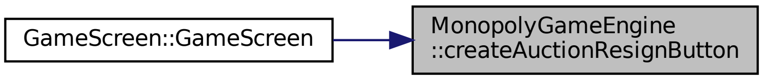
```
void MonopolyGameEngine::createAuctionResignButton ( )
```

Creates a button for resigning from an auction.

Here is the call graph for this function:


MonopolyGameEngine ::createAuctionResignButton
MonopolyGameEngine ::addAuctionButton
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

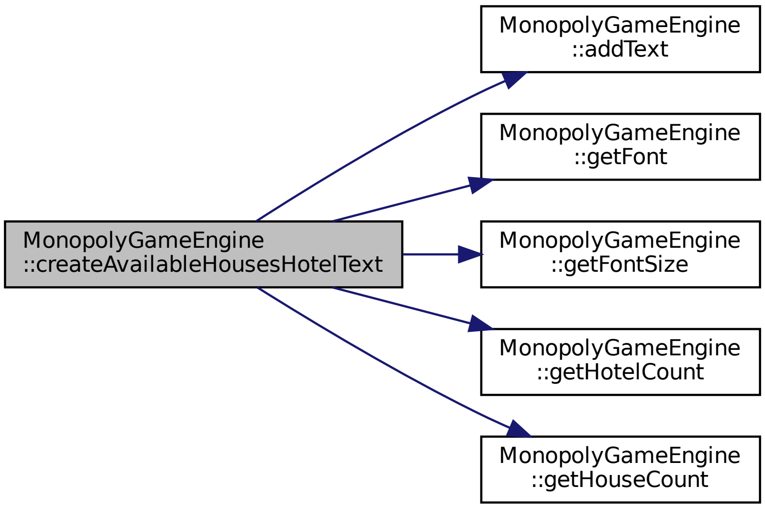


6.18.3.17 createAvailableHousesHotelText()

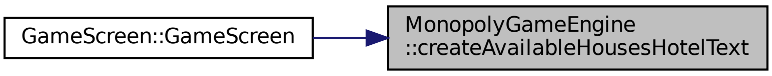
```
void MonopolyGameEngine::createAvailableHousesHotelText ( )
```

Creates text displaying the available count of houses and hotels.

Here is the call graph for this function:


MonopolyGameEngine ::addText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::getFontSize
MonopolyGameEngine ::getHotelCount
MonopolyGameEngine ::getHouseCount

Here is the caller graph for this function:

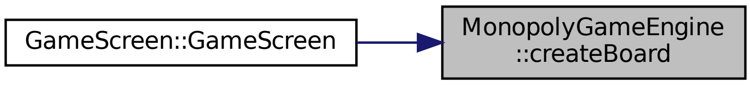


6.18.3.18 createBoard()

```
void MonopolyGameEngine::createBoard ( )
```

Creates the game board.

Here is the caller graph for this function:

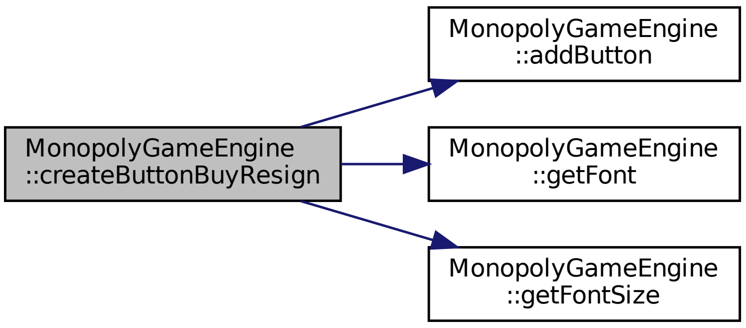


6.18.3.19 createButtonBuyResign()

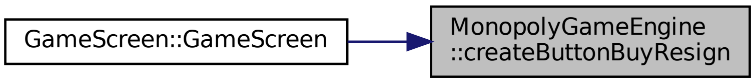
```
void MonopolyGameEngine::createButtonBuyResign ( )
```

Creates a button for buying or resigning from a property.

Here is the call graph for this function:


MonopolyGameEngine ::createButtonBuyResign
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

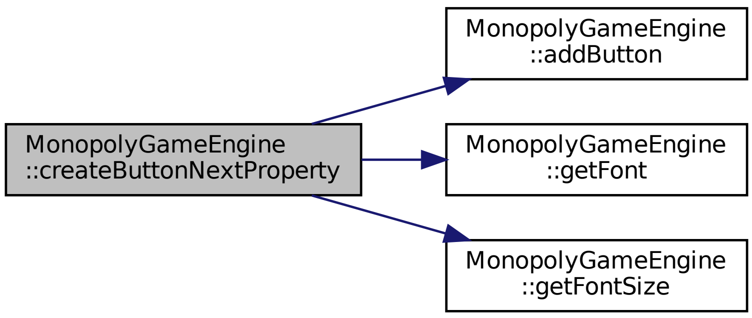


6.18.3.20 createButtonNextProperty()

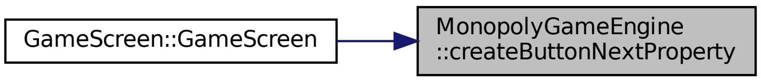
```
void MonopolyGameEngine::createButtonNextProperty ( )
```

Creates a button for moving to the next property.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonNextProperty
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

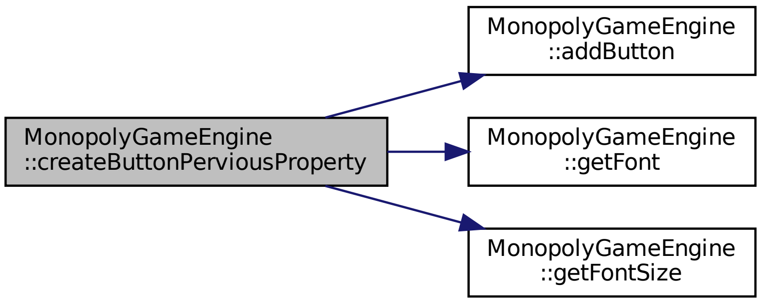


6.18.3.21 createButtonPerviousProperty()

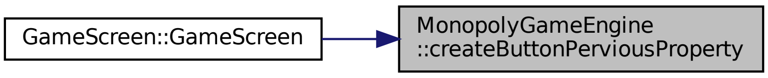
```
void MonopolyGameEngine::createButtonPerviousProperty ( )
```

Creates a button for moving to the previous property.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonPerviousProperty
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

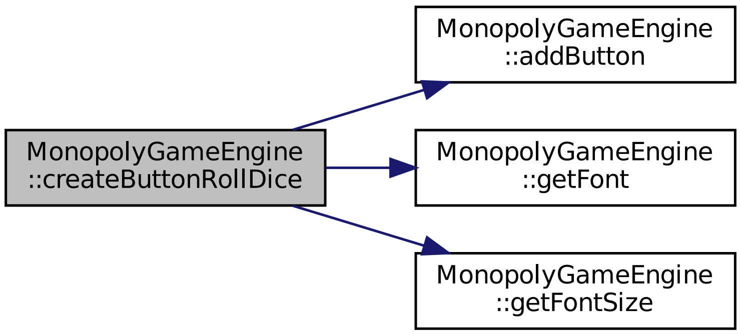


6.18.3.22 createButtonRollDice()

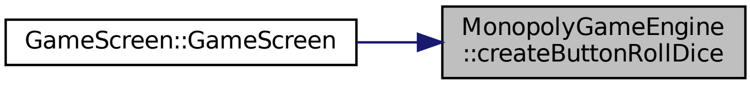
```
void MonopolyGameEngine::createButtonRollDice ( )
```

Creates the button for rolling the dice.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonRollDice
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

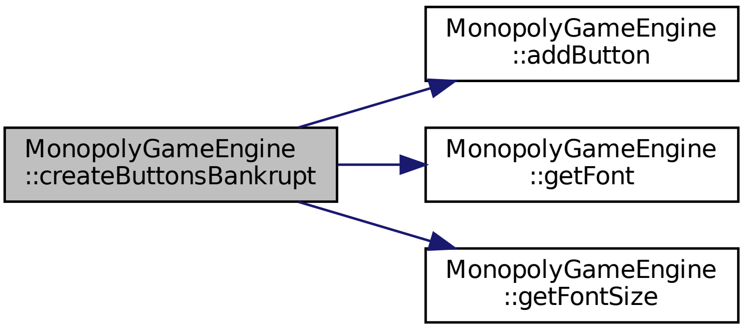


6.18.3.23 createButtonsBankrupt()

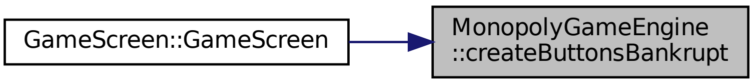
```
void MonopolyGameEngine::createButtonsBankrupt ( )
```

Creates buttons for handling bankruptcy.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonsBankrupt
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

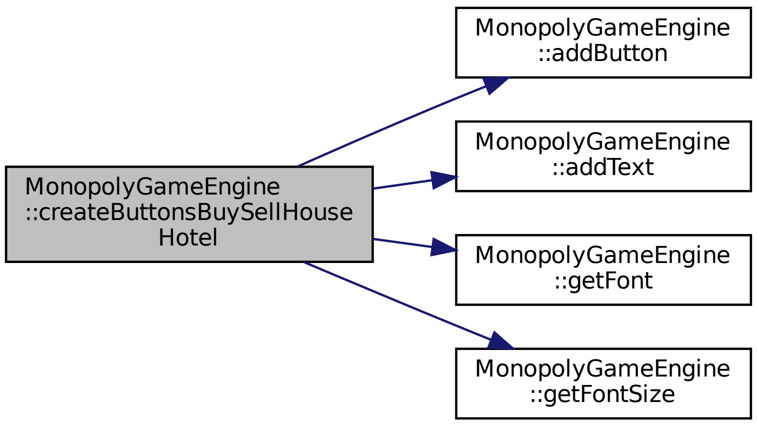


6.18.3.24 createButtonsBuySellHouseHotel()

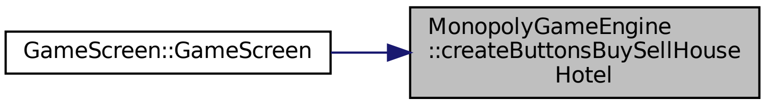
```
void MonopolyGameEngine::createButtonsBuySellHouseHotel ( )
```

Creates buttons for buying, selling houses, and hotels.

Here is the call graph for this function:


MonopolyGameEngine ::createButtonsBuySellHouse Hotel
MonopolyGameEngine ::addButton
MonopolyGameEngine ::addText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

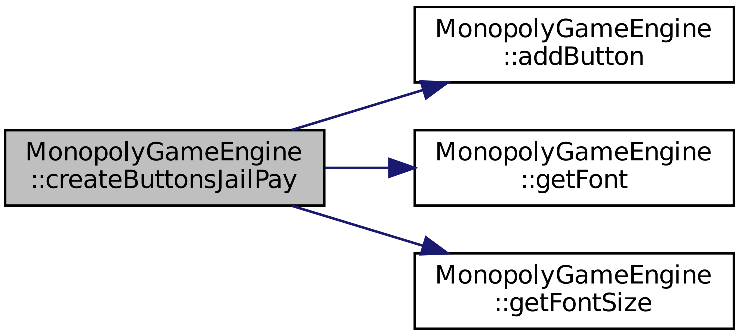


6.18.3.25 createButtonsJailPay()

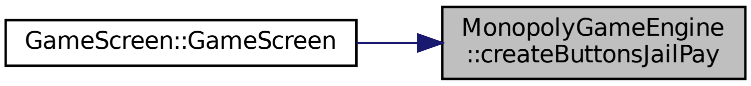
```
void MonopolyGameEngine::createButtonsJailPay ( )
```

Creates buttons for handling actions related to the jail (paying to get out).

Here is the call graph for this function:


MonopolyGameEngine ::createButtonsJailPay
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

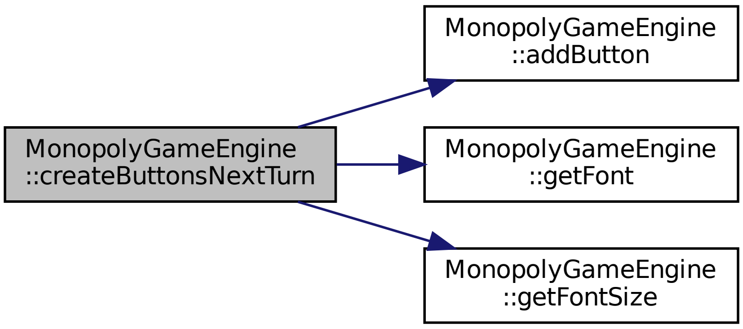


6.18.3.26 createButtonsNextTurn()

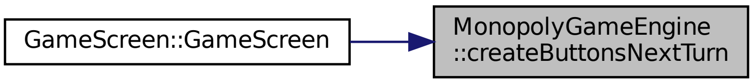
```
void MonopolyGameEngine::createButtonsNextTurn ( )
```

Creates buttons for moving to the next turn.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonsNextTurn
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

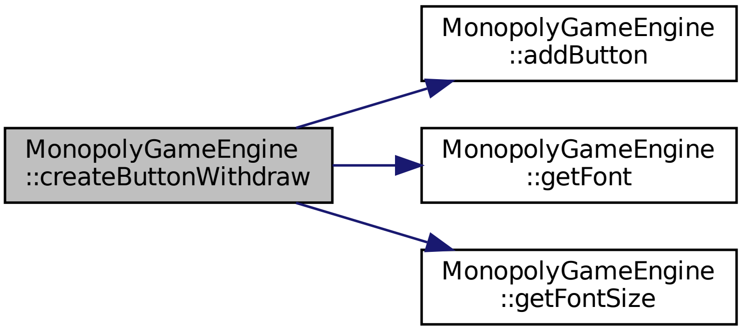


6.18.3.27 createButtonWithdraw()

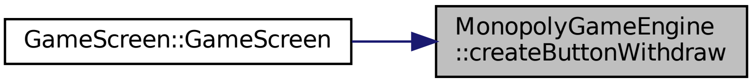
```
void MonopolyGameEngine::createButtonWithdraw ( )
```

Creates a button for withdrawing from the game.

Here is the call graph for this function:


MonopolyGameEngine ::addButton
MonopolyGameEngine ::createButtonWithdraw
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

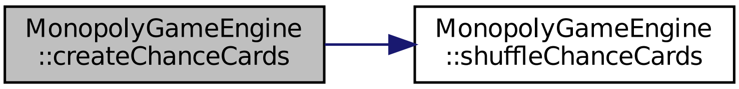


6.18.3.28 createChanceCards()

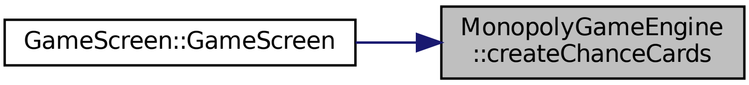
```
void MonopolyGameEngine::createChanceCards ( )
```

Creates the chance cards for the game.

This method initializes and populates the collection of ChanceCard objects used in the game. Here is the call graph for this function:


MonopolyGameEngine ::createChanceCards
MonopolyGameEngine ::shuffleChanceCards

Here is the caller graph for this function:

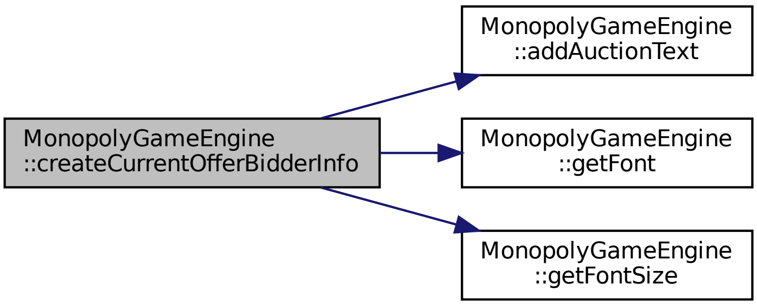


6.18.3.29 createCurrentOfferBidderInfo()

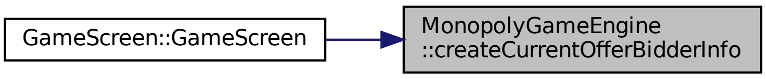
```
void MonopolyGameEngine::createCurrentOfferBidderInfo ( )
```

Creates text for displaying the current offer in an auction.

Here is the call graph for this function:


MonopolyGameEngine ::createCurrentOfferBidderInfo
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

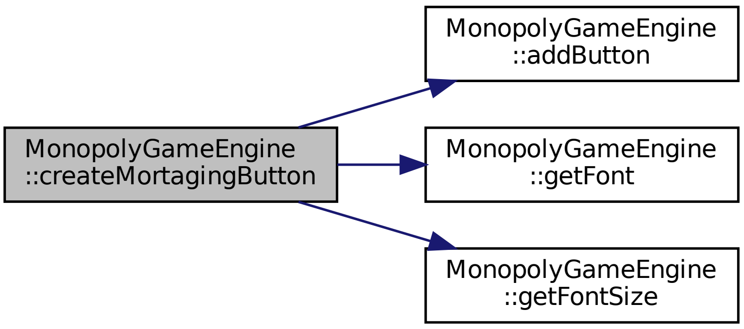


6.18.3.30 createMortagingButton()

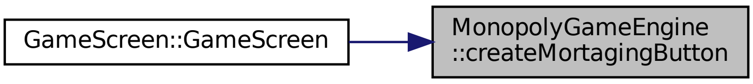
```
void MonopolyGameEngine::createMortagingButton ( )
```

Creates buttons for mortgaging properties.

Here is the call graph for this function:


MonopolyGameEngine ::createMortagingButton
MonopolyGameEngine ::addButton
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:



6.18.3.31 createPlayers()

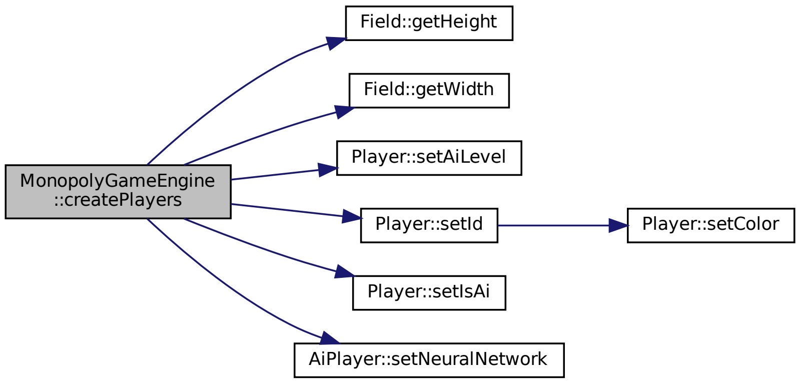
```
void MonopolyGameEngine::createPlayers (
            std::vector< std::shared_ptr< Player >> & players_from_game_engine )
```

Creates player objects based on input vector.

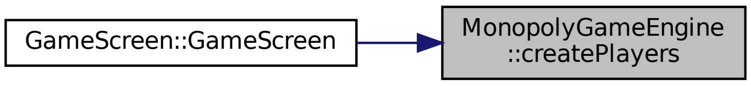
Parameters

players_from_game_engine	Vector of pointers to players.

Here is the call graph for this function:



Here is the caller graph for this function:

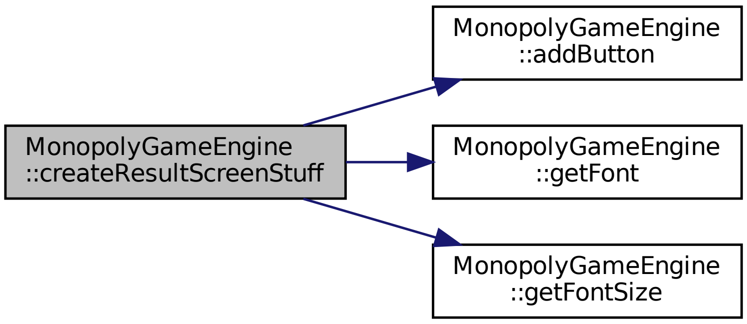


6.18.3.32 createResultScreenStuff()

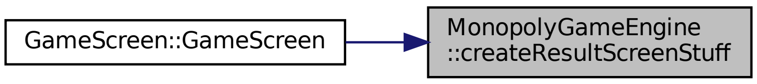
```
void MonopolyGameEngine::createResultScreenStuff ( )
```

Creates various elements for the result screen.

Here is the call graph for this function:



Here is the caller graph for this function:

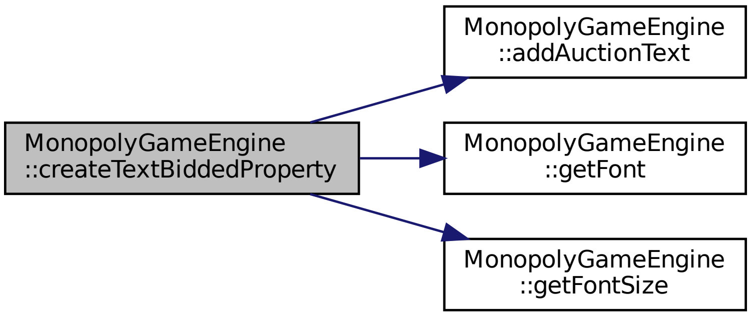


6.18.3.33 createTextBiddedProperty()

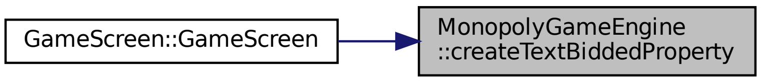
```
void MonopolyGameEngine::createTextBiddedProperty ( )
```

Creates text for displaying information about the bidded property.

Here is the call graph for this function:


MonopolyGameEngine ::createTextBiddedProperty
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

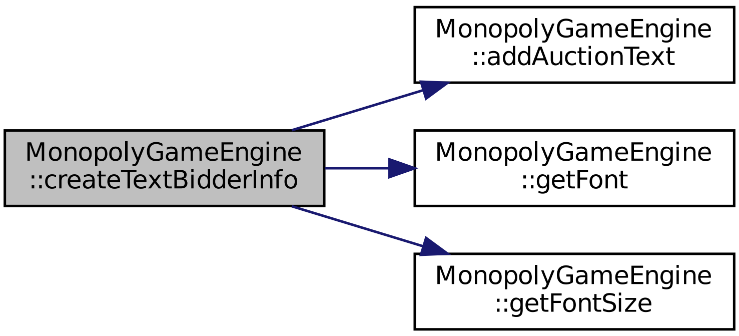


6.18.3.34 createTextBidderInfo()

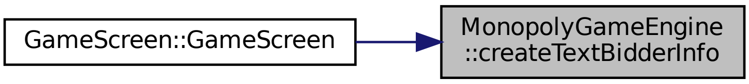
```
void MonopolyGameEngine::createTextBidderInfo ( )
```

Creates text for displaying information about the bidder.

Here is the call graph for this function:


MonopolyGameEngine ::createTextBidderInfo
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

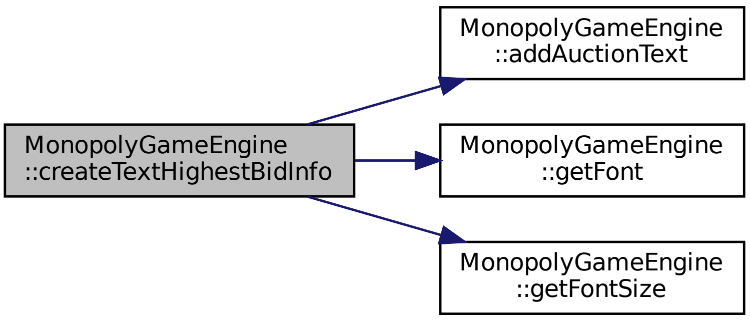


6.18.3.35 createTextHighestBidInfo()

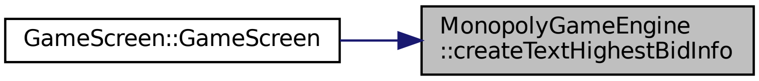
```
void MonopolyGameEngine::createTextHighestBidInfo ( )
```

Creates text for displaying the highest bid in an auction.

Here is the call graph for this function:


MonopolyGameEngine ::createTextHighestBidInfo
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

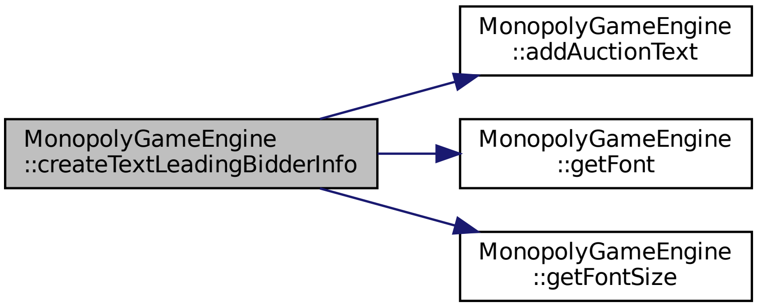


6.18.3.36 createTextLeadingBidderInfo()

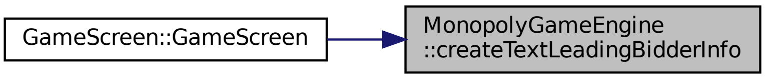
```
void MonopolyGameEngine::createTextLeadingBidderInfo ( )
```

Creates text for displaying information about the leading bidder in an auction.

Here is the call graph for this function:


MonopolyGameEngine ::createTextLeadingBidderInfo
MonopolyGameEngine ::addAuctionText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

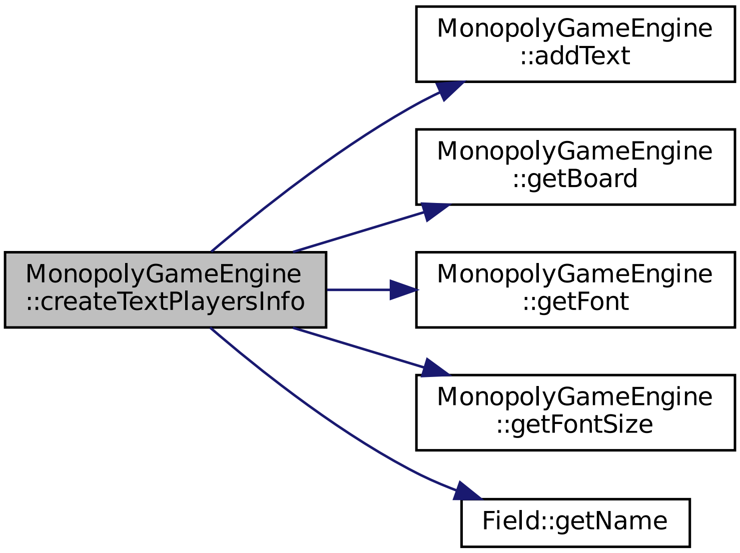


6.18.3.37 createTextPlayersInfo()

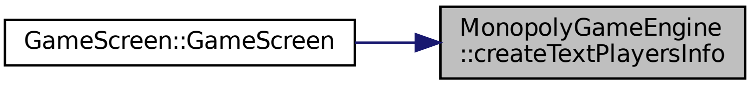
```
void MonopolyGameEngine::createTextPlayersInfo ( )
```

Creates text for displaying players' information.

Here is the call graph for this function:


MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::addText
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize
Field::getName

Here is the caller graph for this function:

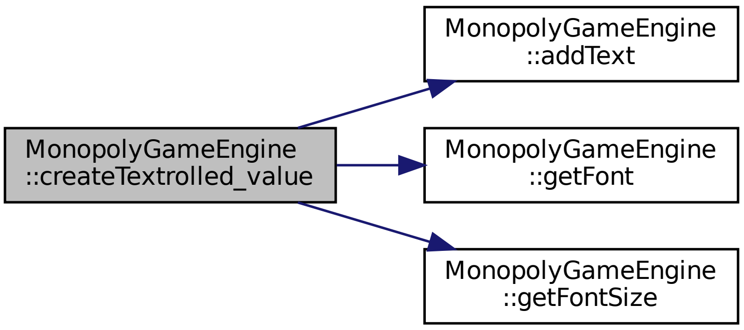


6.18.3.38 createTextrolled_value()

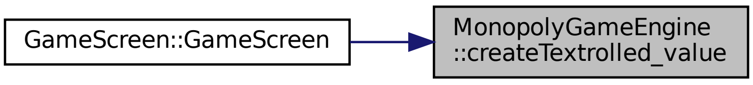
```
void MonopolyGameEngine::createTextrolled_value ( )
```

Creates text for displaying the rolled dice value.

Here is the call graph for this function:



Here is the caller graph for this function:

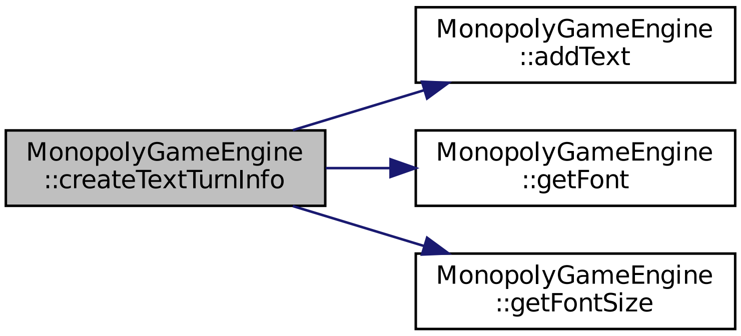


6.18.3.39 createTextTurnInfo()

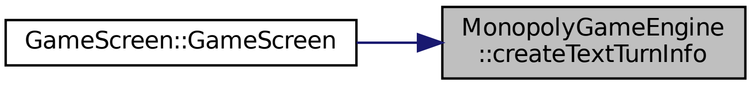
```
void MonopolyGameEngine::createTextTurnInfo ( )
```

Creates text for displaying turn information.

Here is the call graph for this function:


MonopolyGameEngine ::createTextTurnInfo
MonopolyGameEngine ::addText
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:



6.18.3.40 getAllPropertyDataSprite()

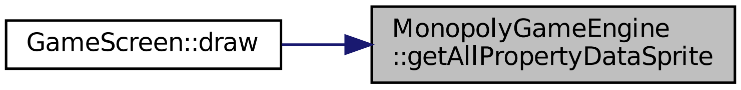
```
sf::Sprite & MonopolyGameEngine::getAllPropertyDataSprite ( )
```

Returns the sprite representing all property data in the GUI.

Returns

Reference to the all property data sprite.

Here is the caller graph for this function:



6.18.3.41 getAllPropertyDataTexts()

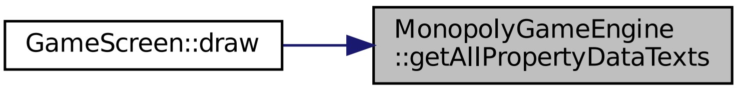
`std::vector< std::shared_ptr< sf::Text > > & MonopolyGameEngine::getAllPropertyDataTexts ( )`

Returns a vector of pointers to text elements representing all property data in the GUI.

Returns

Vector of pointers to all property data text elements.

Here is the caller graph for this function:



6.18.3.42 getAuctionButtons()

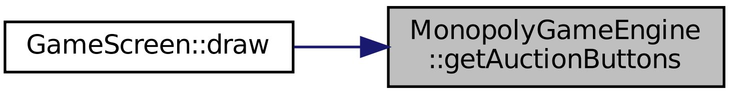
`std::vector< std::shared_ptr< Button > > & MonopolyGameEngine::getAuctionButtons ( )`

Returns a vector of pointers to auction buttons in the GUI.

Returns

Vector of pointers to auction buttons.

Here is the caller graph for this function:



6.18.3.43 getAuctionState()

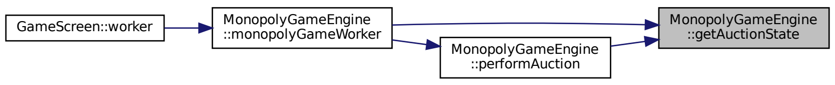
```
AuctionState MonopolyGameEngine::getAuctionState ( )
```

Gets the current state of the auction.

Returns

Current state of the auction.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::performAuction
MonopolyGameEngine ::getAuctionState

6.18.3.44 getAuctionTexts()

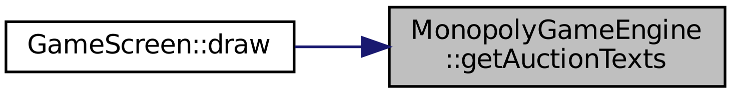
```
std::vector< std::shared_ptr< sf::Text > > & MonopolyGameEngine::getAuctionTexts ( )
```

Returns a vector of pointers to auction text elements in the GUI.

Returns

Vector of pointers to auction text elements.

Here is the caller graph for this function:



6.18.3.45 getBoard()

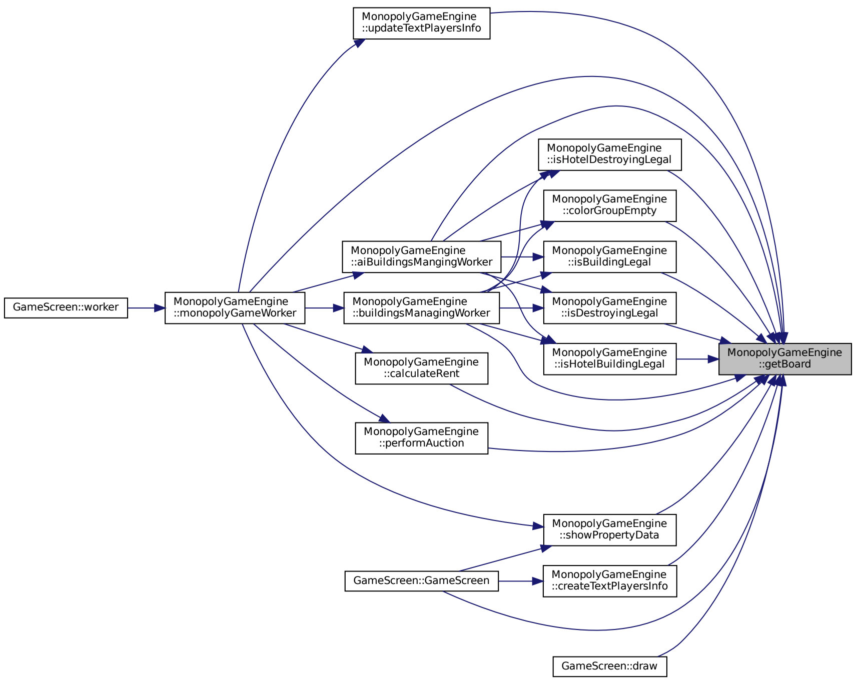
```
std::shared_ptr< Board > MonopolyGameEngine::getBoard ( )
```

Returns a pointer to the game board.

Returns

Pointer to the game board.

Here is the caller graph for this function:


MonopolyGameEngine ::updateTextPlayersInfo
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::isBuildingLegal
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isDestroyingLegal
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::performAuction
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen
MonopolyGameEngine ::createTextPlayersInfo
GameScreen::draw

6.18.3.46 getButtons()

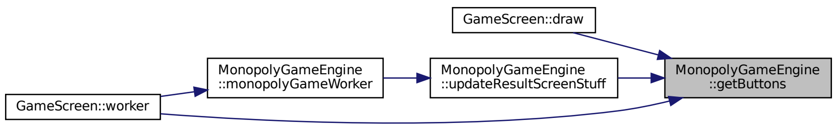
```
std::vector< std::shared_ptr< Button > > & MonopolyGameEngine::getButtons ( )
```

Returns a vector of pointers to buttons in the GUI.

Returns

Vector of pointers to buttons.

Here is the caller graph for this function:


GameScreen::draw
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateResultScreenStuff
MonopolyGameEngine ::getButtons
GameScreen::worker

6.18.3.47 getFont()

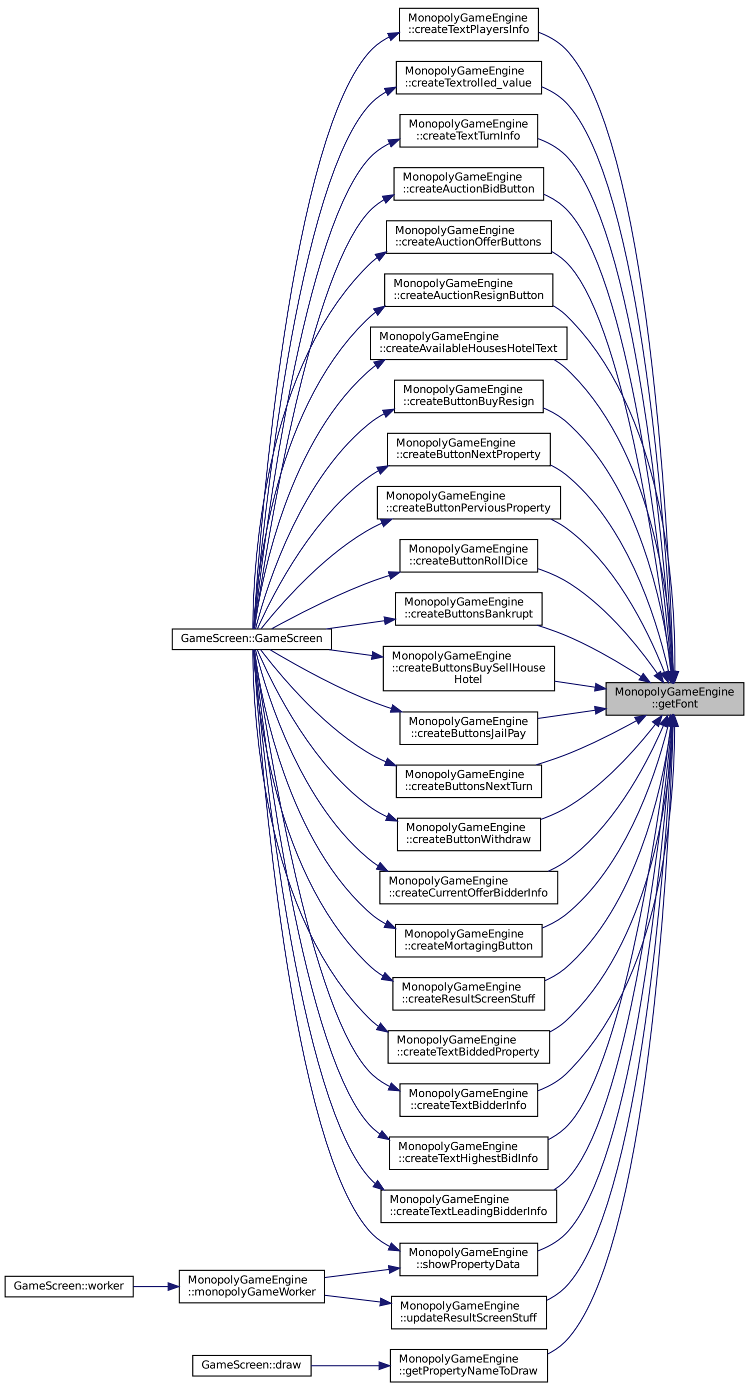
```
sf::Font & MonopolyGameEngine::getFont ( )
```

Returns the font used in the GUI.

Returns

Reference to the font.

Here is the caller graph for this function:



MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::createTextrolled_value
MonopolyGameEngine ::createTextTurnInfo
MonopolyGameEngine ::createAuctionBidButton
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::createAuctionResignButton
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::createButtonBuyResign
MonopolyGameEngine ::createButtonNextProperty
MonopolyGameEngine ::createButtonPerviousProperty
MonopolyGameEngine ::createButtonRollDice
MonopolyGameEngine ::createButtonsBankrupt
GameScreen::GameScreen
MonopolyGameEngine ::createButtonsBuySellHouse Hotel
MonopolyGameEngine ::getFont
MonopolyGameEngine ::createButtonsJailPay
MonopolyGameEngine ::createButtonsNextTurn
MonopolyGameEngine ::createButtonWithdraw
MonopolyGameEngine ::createCurrentOfferBidderInfo
MonopolyGameEngine ::createMortagingButton
MonopolyGameEngine ::createResultScreenStuff
MonopolyGameEngine ::createTextBiddedProperty
MonopolyGameEngine ::createTextBidderInfo
MonopolyGameEngine ::createTextHighestBidInfo
MonopolyGameEngine ::createTextLeadingBidderInfo
MonopolyGameEngine ::showPropertyData
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateResultScreenStuff
GameScreen::draw
MonopolyGameEngine ::getPropertyNameToDraw

6.18.3.48 getFontSize()

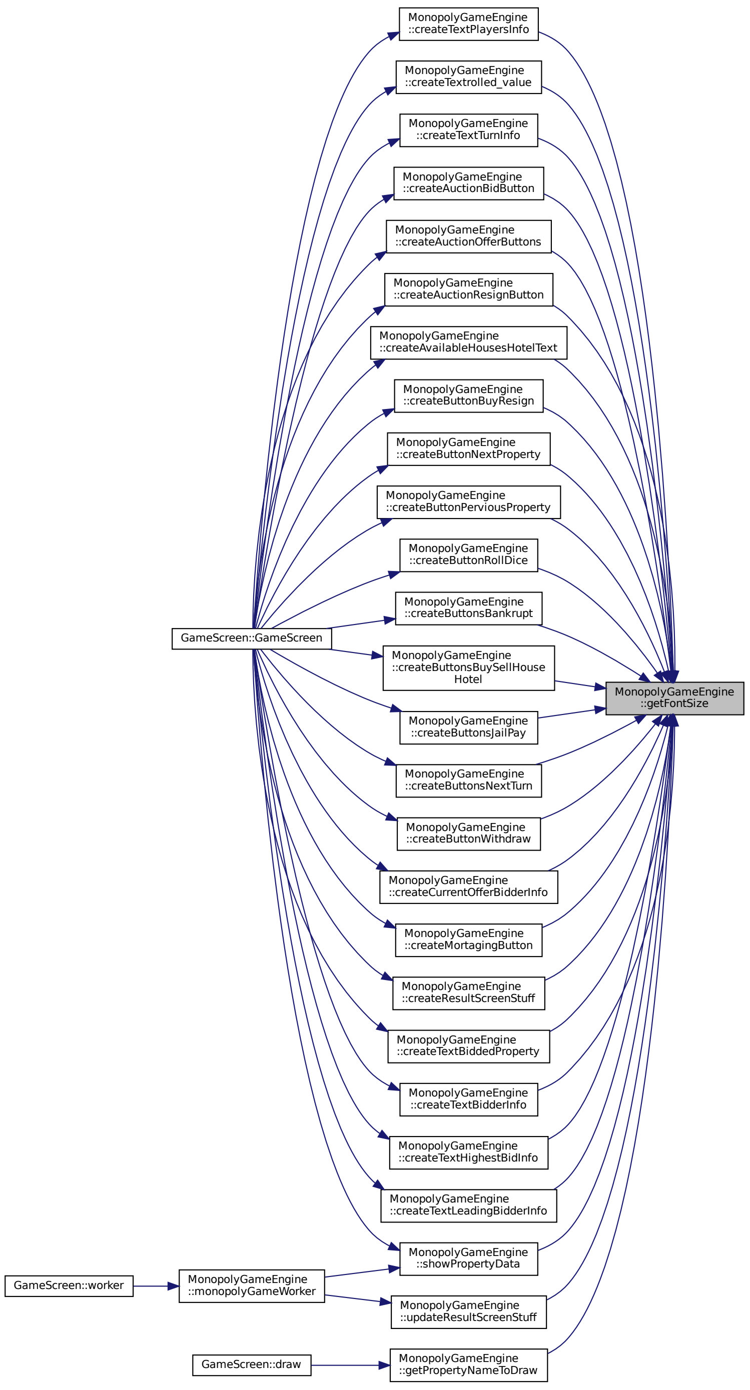
```
unsigned int MonopolyGameEngine::getFontSize ( ) const
```

Returns the font size used in the GUI.

Returns

Font size.

Here is the caller graph for this function:


MonopolyGameEngine ::createTextPlayersInfo
MonopolyGameEngine ::createTextrolled_value
MonopolyGameEngine ::createTextTurnInfo
MonopolyGameEngine ::createAuctionBidButton
MonopolyGameEngine ::createAuctionOfferButtons
MonopolyGameEngine ::createAuctionResignButton
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::createButtonBuyResign
MonopolyGameEngine ::createButtonNextProperty
MonopolyGameEngine ::createButtonPerviousProperty
MonopolyGameEngine ::createButtonRollDice
MonopolyGameEngine ::createButtonsBankrupt
GameScreen::GameScreen
MonopolyGameEngine ::createButtonsBuySellHouse Hotel
MonopolyGameEngine ::getFontSize
MonopolyGameEngine ::createButtonsJailPay
MonopolyGameEngine ::createButtonsNextTurn
MonopolyGameEngine ::createButtonWithdraw
MonopolyGameEngine ::createCurrentOfferBidderInfo
MonopolyGameEngine ::createMortagingButton
MonopolyGameEngine ::createResultScreenStuff
MonopolyGameEngine ::createTextBiddedProperty
MonopolyGameEngine ::createTextBidderInfo
MonopolyGameEngine ::createTextHighestBidInfo
MonopolyGameEngine ::createTextLeadingBidderInfo
MonopolyGameEngine ::showPropertyData
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateResultScreenStuff
GameScreen::draw
MonopolyGameEngine ::getPropertyNameToDraw

6.18.3.49 getHotelCount()

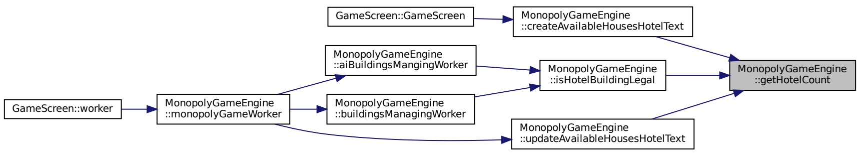
```
unsigned int MonopolyGameEngine::getHotelCount ( )
```

Gets the count of hotels available for purchase.

Returns

Count of hotels.

Here is the caller graph for this function:


GameScreen::GameScreen
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::getHotelCount
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::updateAvailableHousesHotelText

6.18.3.50 getHotelSprite()

```
sf::Sprite MonopolyGameEngine::getHotelSprite (
             StreetField & field )
```

Gets the sprite for a hotel on a street.

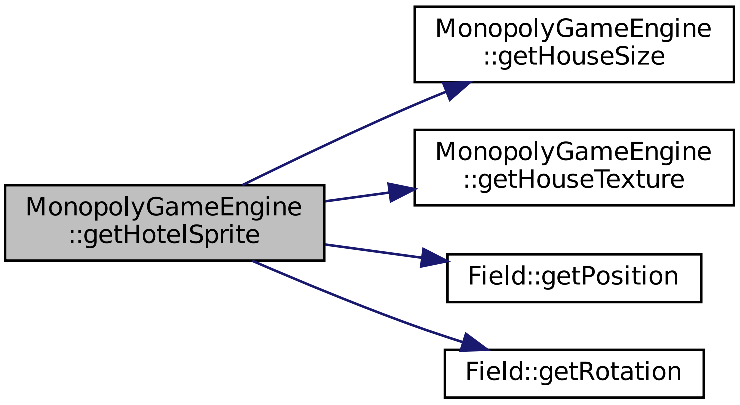
Parameters

field	Street field to get the hotel sprite for.

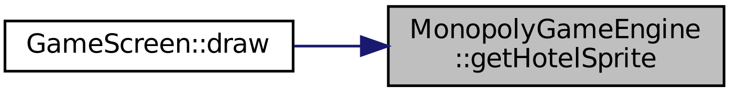
Returns

Sprite of the hotel.

Here is the call graph for this function:


MonopolyGameEngine ::getHotelSprite
MonopolyGameEngine ::getHouseSize
MonopolyGameEngine ::getHouseTexture
Field::getPosition
Field::getRotation

Here is the caller graph for this function:



6.18.3.51 getHotelTexture()

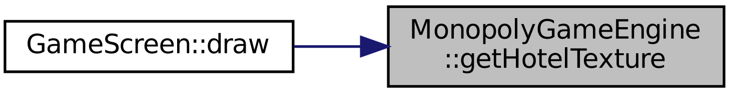
```
sf::Texture & MonopolyGameEngine::getHotelTexture ( )
```

Returns the texture of the hotel used in the GUI.

Returns

Reference to the hotel texture.

Here is the caller graph for this function:



6.18.3.52 getHouseCount()

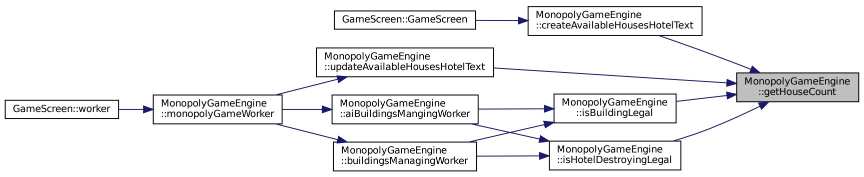
```
unsigned int MonopolyGameEngine::getHouseCount ( )
```

Gets the count of houses available for purchase.

Returns

Count of houses.

Here is the caller graph for this function:


GameScreen::GameScreen
MonopolyGameEngine ::createAvailableHousesHotelText
MonopolyGameEngine ::updateAvailableHousesHotelText
MonopolyGameEngine ::getHouseCount
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isHotelDestroyingLegal

6.18.3.53 getHouseSize()

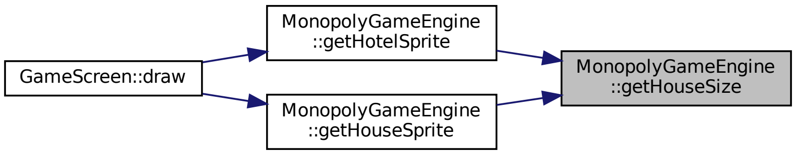
```
sf::Vector2f & MonopolyGameEngine::getHouseSize ( )
```

Returns the size of a house in the GUI.

Returns

Reference to the house size.

Here is the caller graph for this function:


GameScreen::draw
MonopolyGameEngine ::getHotelSprite
MonopolyGameEngine ::getHouseSprite
MonopolyGameEngine ::getHouseSize

6.18.3.54 getHouseSprite()

```
sf::Sprite MonopolyGameEngine::getHouseSprite (
             StreetField & field,
             unsigned int houses_number )
```

Gets the sprite for a house on a street.

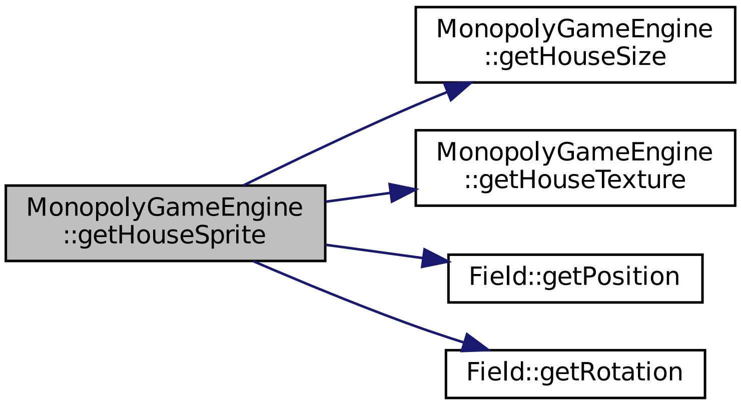
Parameters

field	Street field to get the house sprite for.
houses_number	Number of houses on the street.

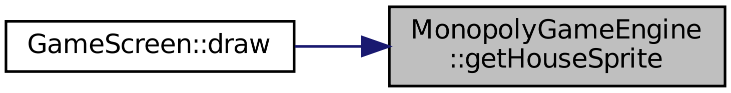
Returns

Sprite of the house.

Here is the call graph for this function:


MonopolyGameEngine ::getHouseSprite
MonopolyGameEngine ::getHouseSize
MonopolyGameEngine ::getHouseTexture
Field::getPosition
Field::getRotation

Here is the caller graph for this function:



6.18.3.55 getHouseTexture()

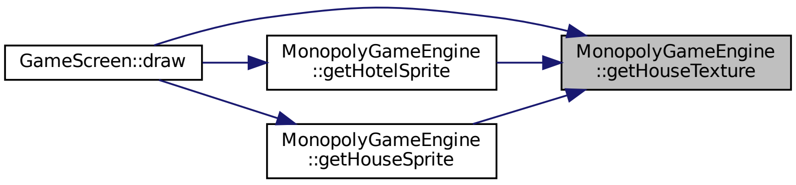
```
sf::Texture & MonopolyGameEngine::getHouseTexture ( )
```

Returns the texture of the house used in the GUI.

Returns

Reference to the house texture.

Here is the caller graph for this function:


GameScreen::draw
MonopolyGameEngine ::getHotelSprite
MonopolyGameEngine ::getHouseSprite
MonopolyGameEngine ::getHouseTexture

6.18.3.56 getNotificationsWall()

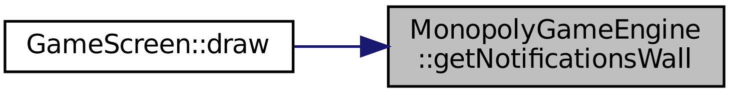
```
NotificationWall & MonopolyGameEngine::getNotificationsWall ( )
```

Returns the notification wall used in the GUI.

Returns

Reference to the notification wall.

Here is the caller graph for this function:



6.18.3.57 getPlayers()

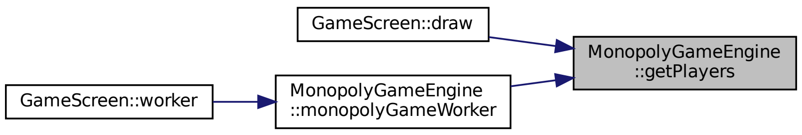
```
std::vector< std::shared_ptr< Player > > & MonopolyGameEngine::getPlayers ( )
```

Returns a vector of pointers to players in the game.

Returns

Vector of pointers to players.

Here is the caller graph for this function:


GameScreen::draw
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::getPlayers

6.18.3.58 getPlayersResult()

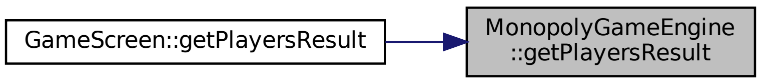
```
std::vector< std::shared_ptr< Player > > MonopolyGameEngine::getPlayersResult ( )
```

Returns a vector of pointers to players based on game result.

Returns

Vector of pointers to players.

Here is the caller graph for this function:


GameScreen::getPlayersResult
MonopolyGameEngine ::getPlayersResult

6.18.3.59 getPropertyDataSprite()

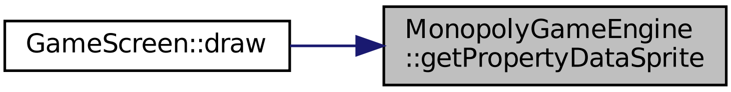
```
sf::Sprite & MonopolyGameEngine::getPropertyDataSprite ( )
```

Returns the sprite representing property data in the GUI.

Returns

Reference to the property data sprite.

Here is the caller graph for this function:



6.18.3.60 getPropertyDataTexts()

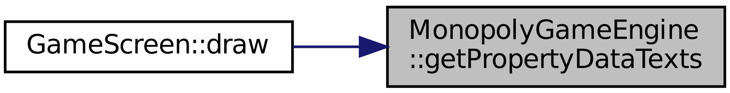
```
std::vector< std::shared_ptr< sf::Text > > & MonopolyGameEngine::getPropertyDataTexts ( )
```

Returns a vector of pointers to text elements representing property data in the GUI.

Returns

Vector of pointers to property data text elements.

Here is the caller graph for this function:



6.18.3.61 getPropertyNameToDraw()

```
sf::Text MonopolyGameEngine::getPropertyNameToDraw (
            sf::Text text,
            sf::Sprite & sprite,
            float rotation )
```

Gets the text suitable for drawing a property name.

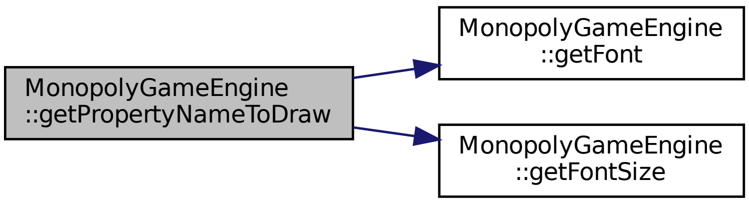
Parameters

text	Original text to be drawn.
sprite	Reference to the sprite associated with the property.
rotation	Rotation angle of the text.

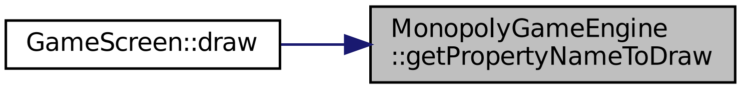
Returns

Modified text for drawing.

Here is the call graph for this function:


MonopolyGameEngine ::getPropertyNameToDraw
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:



6.18.3.62 getResultTexts()

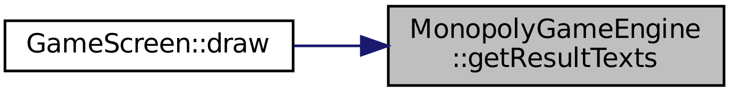
```
std::vector< std::shared_ptr< sf::Text > > & MonopolyGameEngine::getResultTexts ( )
```

Returns a vector of pointers to text elements used in the result screen.

Returns

Vector of pointers to result screen text elements.

Here is the caller graph for this function:



6.18.3.63 getScreenType()

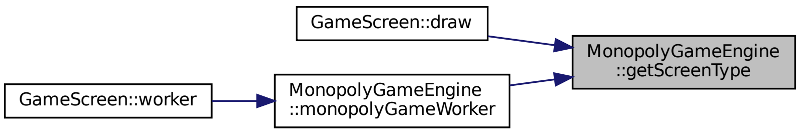
```
GameScreenType MonopolyGameEngine::getScreenType ( ) const
```

Gets the current screen type.

Returns

The current screen type.

Here is the caller graph for this function:


GameScreen::draw
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::getScreenType

6.18.3.64 getTexts()

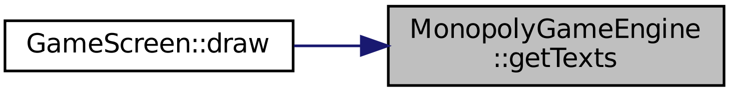
```
std::vector< std::shared_ptr< sf::Text > > & MonopolyGameEngine::getTexts ( )
```

Returns a vector of pointers to text elements in the GUI.

Returns

Vector of pointers to text elements.

Here is the caller graph for this function:



6.18.3.65 getTurnState()

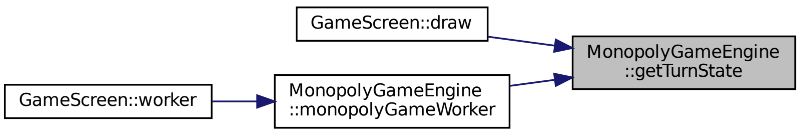
`TurnState MonopolyGameEngine::getTurnState ( ) const`

Gets the current turn state.

Returns

Current turn state.

Here is the caller graph for this function:


GameScreen::draw
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::getTurnState

6.18.3.66 getWithdraw()

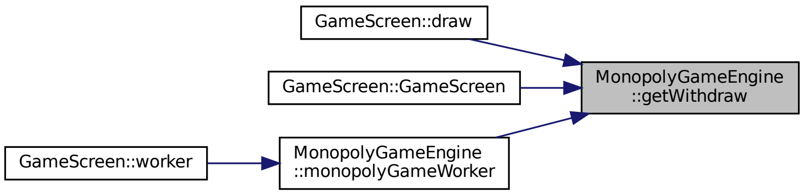
`Withdraw & MonopolyGameEngine::getWithdraw ( )`

Returns a reference to the Withdraw object.

Returns

Withdraw& Reference to the Withdraw object.

Here is the caller graph for this function:


GameScreen::draw
GameScreen::GameScreen
MonopolyGameEngine ::getWithdraw
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker

6.18.3.67 groupCompleted()

```
bool MonopolyGameEngine::groupCompleted (
            std::vector< unsigned int > player_fields,
            PropertyField & field ) const
```

Checks if a group of properties is completed by a player.

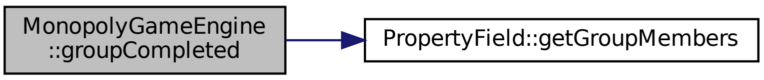
Parameters

player_fields	Vector of field positions owned by the player.
field	Property field to check for group completion.

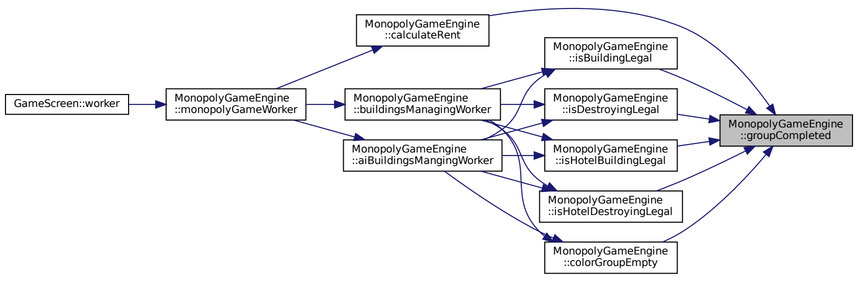
Returns

True if the group is completed, false otherwise.

Here is the call graph for this function:


MonopolyGameEngine ::groupCompleted
PropertyField::getGroupMembers

Here is the caller graph for this function:



MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::isBuildingLegal
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isDestroyingLegal
MonopolyGameEngine ::groupCompleted
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::colorGroupEmpty

6.18.3.68 isBuildingLegal()

```
bool MonopolyGameEngine::isBuildingLegal (
            std::shared_ptr< Player > builder,
            StreetField field )
```

Checks if building houses on a street is legal for a player.

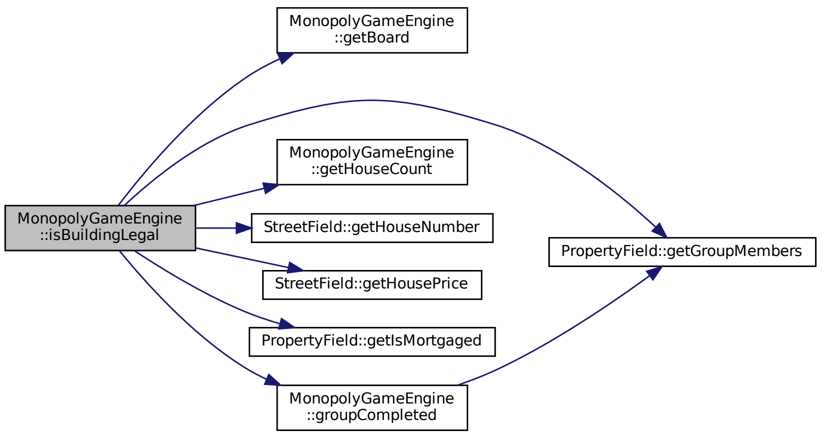
Parameters

builder	Player attempting to build houses.
field	Street field to build houses on.

Returns

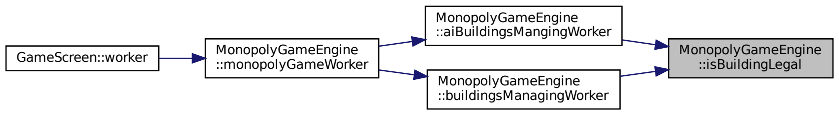
True if building is legal, false otherwise.

Here is the call graph for this function:



MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getHouseCount
MonopolyGameEngine ::isBuildingLegal
StreetField::getHouseNumber
PropertyField::getGroupMembers
StreetField::getHousePrice
PropertyField::getIsMortgaged
MonopolyGameEngine ::groupCompleted

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isBuildingLegal

6.18.3.69 isDestroyingLegal()

```
bool MonopolyGameEngine::isDestroyingLegal (
             std::shared_ptr< Player > builder,
             StreetField field )
```

Checks if destroying houses on a street is legal for a player.

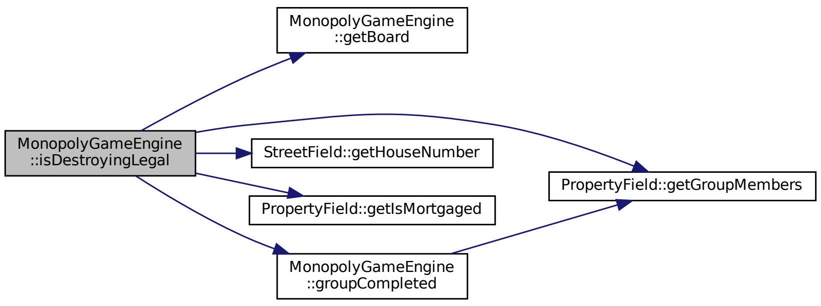
Parameters

builder	Player attempting to destroy houses.
field	Street field to destroy houses on.

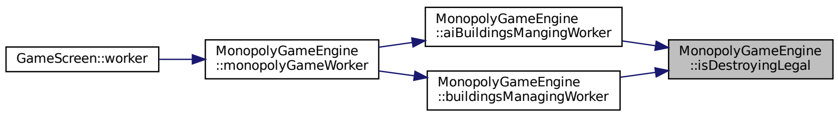
Returns

True if destroying is legal, false otherwise.

Here is the call graph for this function:


MonopolyGameEngine ::getBoard
MonopolyGameEngine ::isDestroyingLegal
StreetField::getHouseNumber
PropertyField::getGroupMembers
PropertyField::getIsMortgaged
MonopolyGameEngine ::groupCompleted

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isDestroyingLegal

6.18.3.70 isHotelBuildingLegal()

```
bool MonopolyGameEngine::isHotelBuildingLegal (
            std::shared_ptr< Player > builder,
            StreetField & field )
```

Checks if building a hotel on a street is legal for a player.

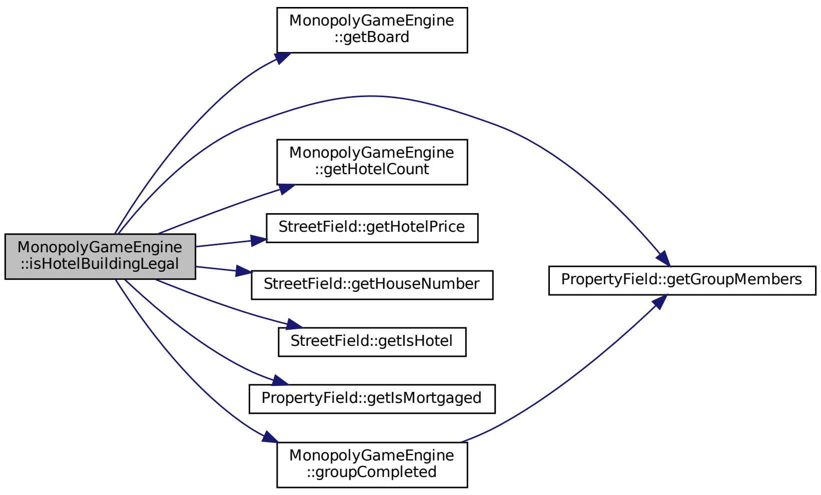
Parameters

builder	Player attempting to build a hotel.
field	Street field to build a hotel on.

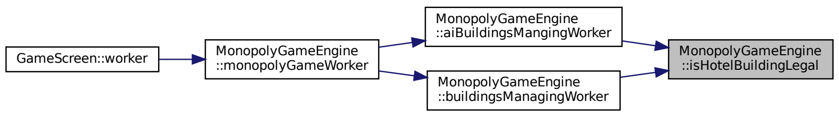
Returns

True if building a hotel is legal, false otherwise.

Here is the call graph for this function:


MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getHotelCount
StreetField::getHotelPrice
MonopolyGameEngine ::isHotelBuildingLegal
StreetField::getHouseNumber
PropertyField::getGroupMembers
StreetField::getIsHotel
PropertyField::getIsMortgaged
MonopolyGameEngine ::groupCompleted

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isHotelBuildingLegal

6.18.3.71 isHotelDestroyingLegal()

```
bool MonopolyGameEngine::isHotelDestroyingLegal (
            std::shared_ptr< Player > builder,
            StreetField & field )
```

Checks if destroying a hotel on a street is legal for a player.

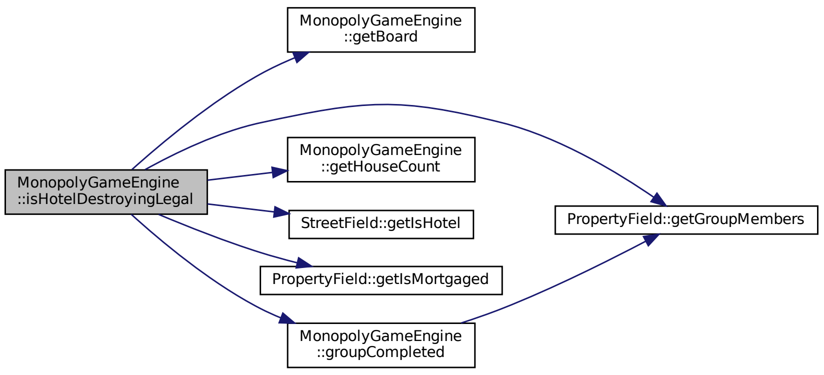
Parameters

builder	Player attempting to destroy a hotel.
field	Street field to destroy a hotel on.

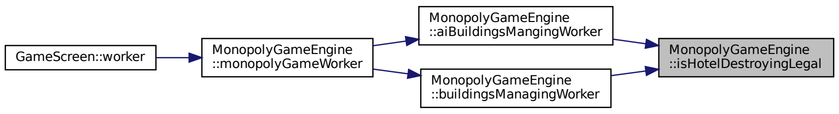
Returns

True if destroying a hotel is legal, false otherwise.

Here is the call graph for this function:


MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getHouseCount
MonopolyGameEngine ::isHotelDestroyingLegal
StreetField::getIsHotel
PropertyField::getGroupMembers
PropertyField::getIsMortgaged
MonopolyGameEngine ::groupCompleted

Here is the caller graph for this function:

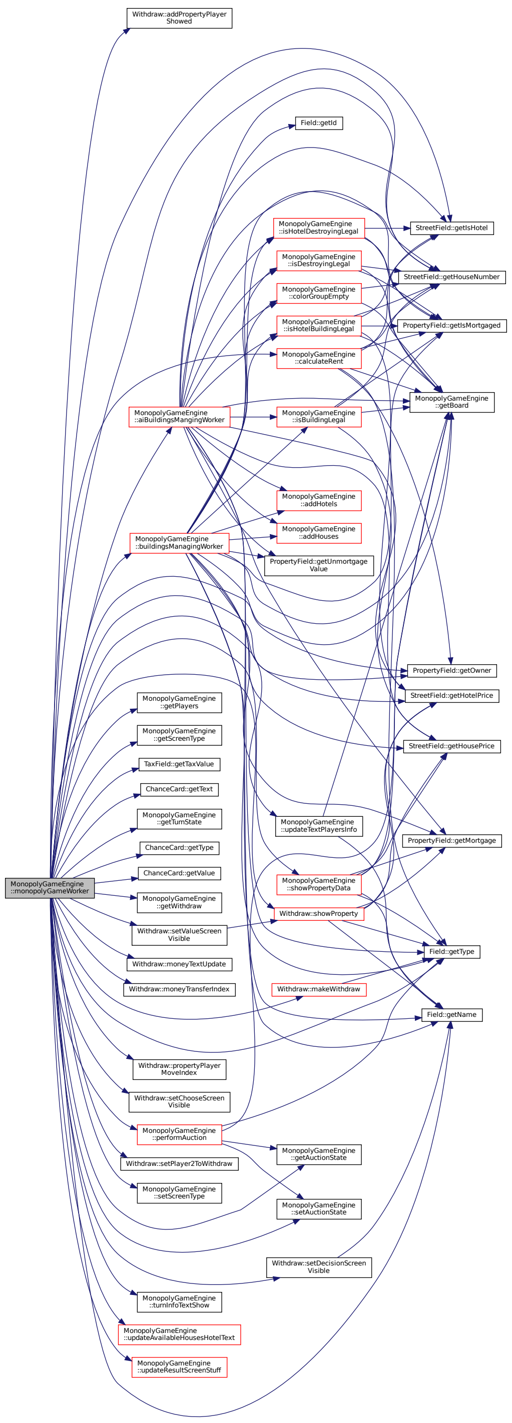

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isHotelDestroyingLegal

6.18.3.72 monopolyGameWorker()

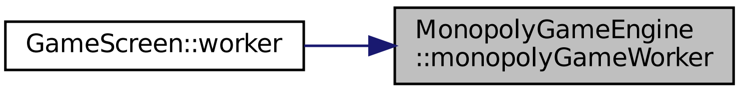
```
bool MonopolyGameEngine::monopolyGameWorker ( )
```

Main Worker method for whole monopoly game engine.

Here is the call graph for this function:


Withdraw::addPropertyPlayer Showed
Field::getId
MonopolyGameEngine ::isHotelDestroyingLegal
StreetField::getIsHotel
MonopolyGameEngine ::isDestroyingLegal
StreetField::getHouseNumber
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::isHotelBuildingLegal
PropertyField::getIsMortgaged
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::addHotels
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHouses
PropertyField::getUnmortgage Value
PropertyField::getOwner
StreetField::getHotelPrice
MonopolyGameEngine ::getPlayers
MonopolyGameEngine ::getScreenType
StreetField::getHousePrice
TaxField::getTaxValue
ChanceCard::getText
MonopolyGameEngine ::getTurnState
MonopolyGameEngine ::updateTextPlayersInfo
PropertyField::getMortgage
ChanceCard::getType
ChanceCard::getValue
MonopolyGameEngine ::showPropertyData
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::getWithdraw
Withdraw::showProperty
Withdraw::setValueScreen Visible
Field::getType
Withdraw::moneyTextUpdate
Withdraw::moneyTransferIndex
Withdraw::makeWithdraw
Field::getName
Withdraw::propertyPlayer MoveIndex
Withdraw::setChooseScreen Visible
MonopolyGameEngine ::performAuction
MonopolyGameEngine ::getAuctionState
Withdraw::setPlayer2ToWithdraw
MonopolyGameEngine ::setScreenType
MonopolyGameEngine ::setAuctionState
Withdraw::setDecisionScreen Visible
MonopolyGameEngine ::turnInfoTextShow
MonopolyGameEngine ::updateAvailableHousesHotelText
MonopolyGameEngine ::updateResultScreenStuff

Here is the caller graph for this function:

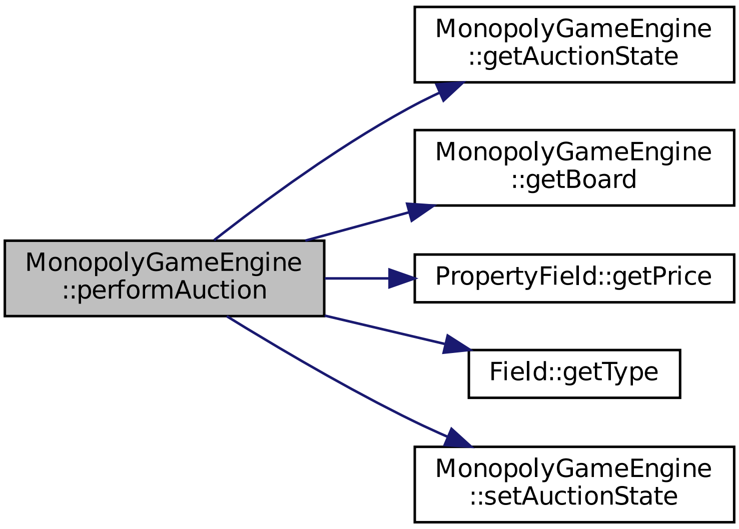


6.18.3.73 performAuction()

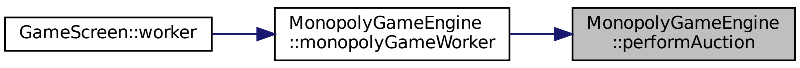
```
void MonopolyGameEngine::performAuction ( )
```

Performs the auction.

Here is the call graph for this function:


MonopolyGameEngine ::performAuction
MonopolyGameEngine ::getAuctionState
MonopolyGameEngine ::getBoard
PropertyField::getPrice
Field::getType
MonopolyGameEngine ::setAuctionState

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::performAuction

6.18.3.74 setAuctionState()

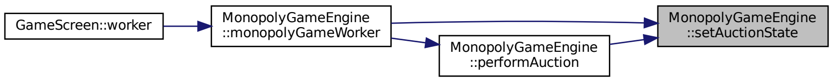
```
void MonopolyGameEngine::setAuctionState (
            AuctionState new_state )
```

Sets the state of the auction.

Parameters

new_state	New state of the auction.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::performAuction
MonopolyGameEngine ::setAuctionState

6.18.3.75 setFont()

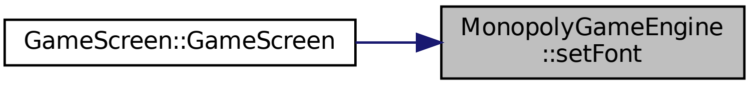
```
void MonopolyGameEngine::setFont (
            sf::Font font )
```

Sets the font to be used in the GUI.

Parameters

font	Font to be set.

Here is the caller graph for this function:



6.18.3.76 setHotelCount()

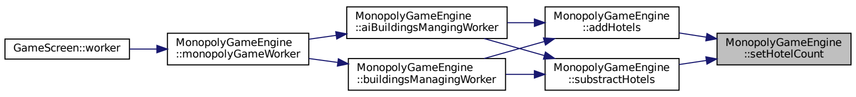
```
void MonopolyGameEngine::setHotelCount (
            unsigned int new_count )
```

Sets the count of hotels available for purchase.

Parameters

new_count	New count of hotels.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHotels
MonopolyGameEngine ::substractHotels
MonopolyGameEngine ::setHotelCount

6.18.3.77 setHouseCount()

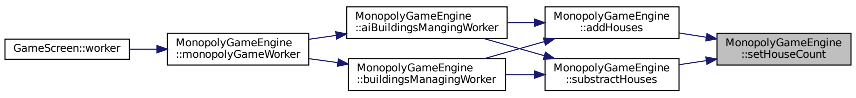
```
void MonopolyGameEngine::setHouseCount (
            unsigned int new_count )
```

Sets the count of houses available for purchase.

Parameters

new_count	New count of houses.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::addHouses
MonopolyGameEngine ::substractHouses
MonopolyGameEngine ::setHouseCount

6.18.3.78 setplayer_index_turn()

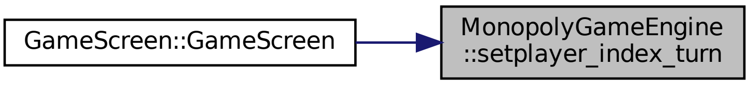
```
void MonopolyGameEngine::setplayer_index_turn (
            unsigned int indx )
```

Sets the index of the player whose turn it is.

Parameters

indx	Index of the player.

Here is the caller graph for this function:



6.18.3.79 setScreenType()

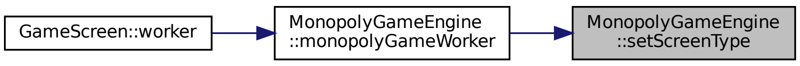
```
void MonopolyGameEngine::setScreenType (
            GameScreenType new_screen_type )
```

Sets the screen type to the specified type.

Parameters

new_screen_type	The new screen type.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::setScreenType

6.18.3.80 showPropertyData()

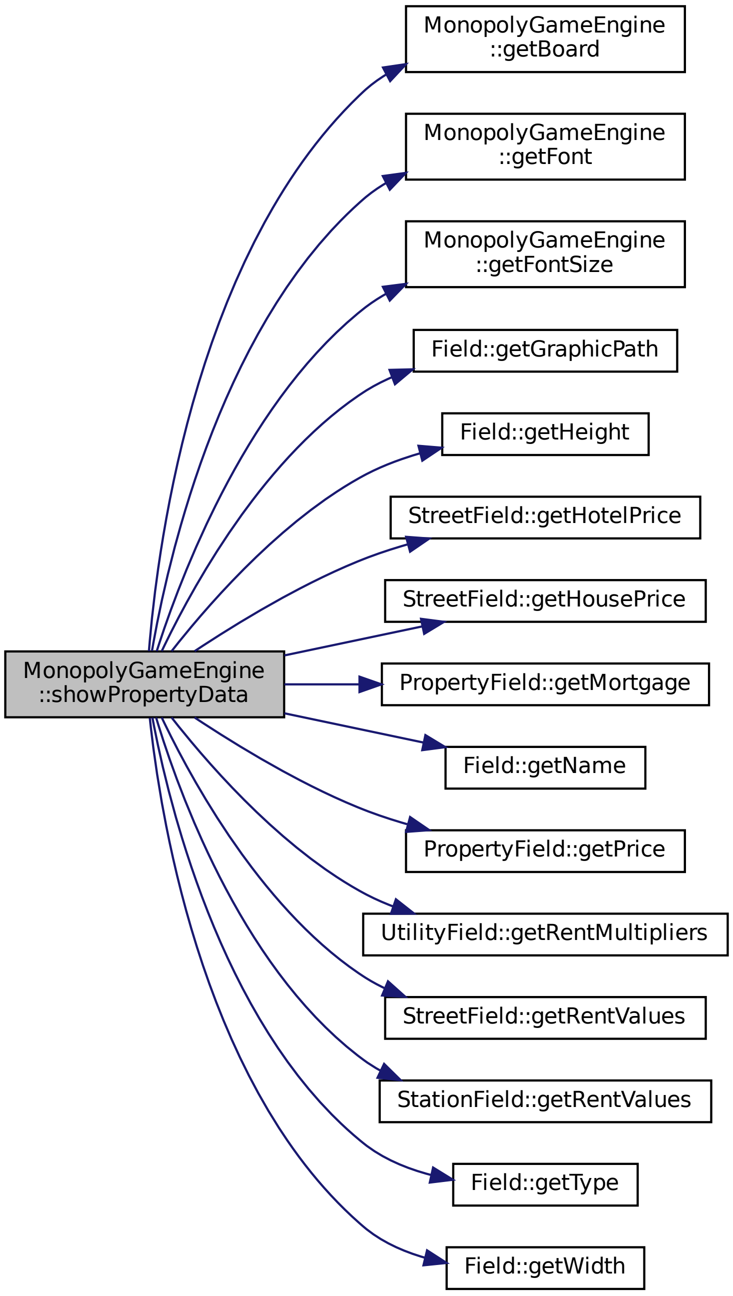
```
void MonopolyGameEngine::showPropertyData (
            unsigned int pos,
            bool is_property_shown_to_buy )
```

Shows property data on the GUI based on the position and whether it is shown to buy.

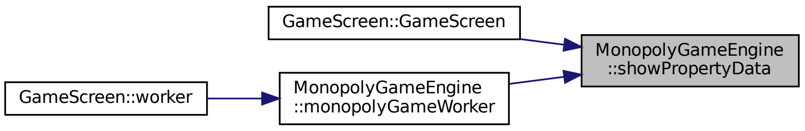
Parameters

pos	The position of the property.
is_property_shown_to_buy	Indicates whether the property information is shown for buying.

Here is the call graph for this function:


MonopolyGameEngine ::showPropertyData
MonopolyGameEngine ::getBoard
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize
Field::getGraphicPath
Field::getHeight
StreetField::getHotelPrice
StreetField::getHousePrice
PropertyField::getMortgage
Field::getName
PropertyField::getPrice
UtilityField::getRentMultipliers
StreetField::getRentValues
StationField::getRentValues
Field::getType
Field::getWidth

Here is the caller graph for this function:

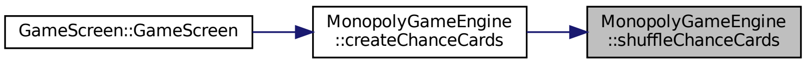

GameScreen::GameScreen
MonopolyGameEngine ::showPropertyData
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker

6.18.3.81 shuffleChanceCards()

```
void MonopolyGameEngine::shuffleChanceCards ( )
```

Shuffles the chance cards.

This method shuffles the collection of ChanceCard objects to randomize the order in which they will be drawn during the game. Here is the caller graph for this function:


GameScreen::GameScreen
MonopolyGameEngine ::createChanceCards
MonopolyGameEngine ::shuffleChanceCards

6.18.3.82 substractHotels()

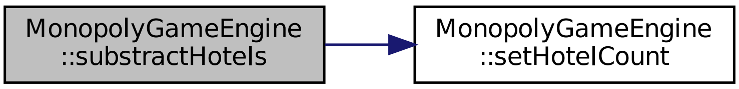
```
void MonopolyGameEngine::substractHotels (
             unsigned int substracted_amount )
```

Subtracts hotels from the available count.

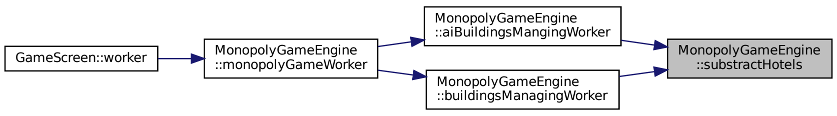
Parameters

substracted_amount	Amount to be subtracted.

Here is the call graph for this function:


MonopolyGameEngine ::substractHotels
MonopolyGameEngine ::setHotelCount

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::substractHotels

6.18.3.83 substractHouses()

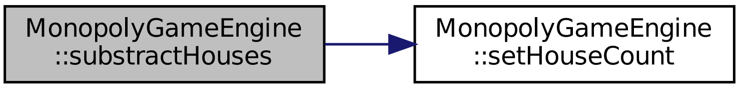
```
void MonopolyGameEngine::substractHouses (
             unsigned int substracted_amount )
```

Subtracts houses from the available count.

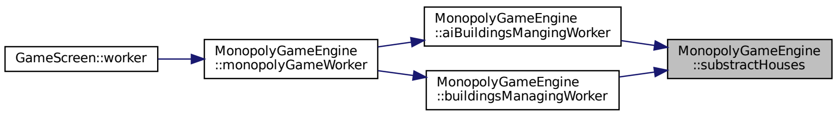
Parameters

substracted_amount	Amount to be subtracted.

Here is the call graph for this function:


MonopolyGameEngine ::substractHouses
MonopolyGameEngine ::setHouseCount

Here is the caller graph for this function:

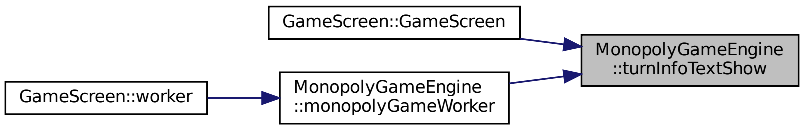

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::substractHouses

6.18.3.84 turnInfoTextShow()

```
void MonopolyGameEngine::turnInfoTextShow ( )
```

Displays text related to turn information.

Here is the caller graph for this function:

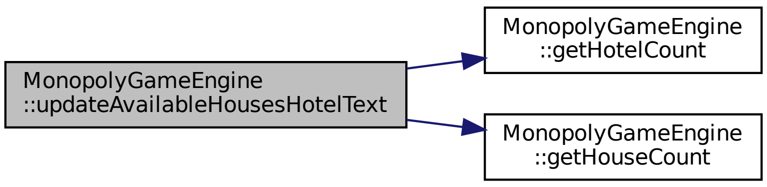

GameScreen::GameScreen
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::turnInfoTextShow

6.18.3.85 updateAvailableHousesHotelText()

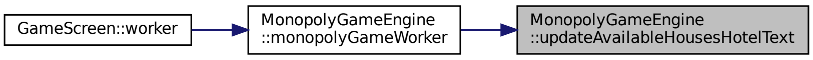
```
void MonopolyGameEngine::updateAvailableHousesHotelText ( )
```

Updates the text displaying the available count of houses and hotels.

Here is the call graph for this function:


MonopolyGameEngine ::updateAvailableHousesHotelText
MonopolyGameEngine ::getHotelCount
MonopolyGameEngine ::getHouseCount

Here is the caller graph for this function:

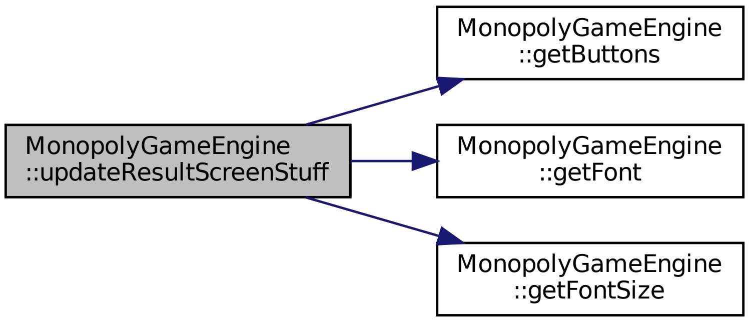

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateAvailableHousesHotelText

6.18.3.86 updateResultScreenStuff()

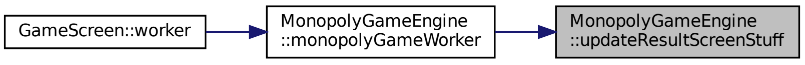
```
void MonopolyGameEngine::updateResultScreenStuff ( )
```

Updates elements on the result screen.

Here is the call graph for this function:


MonopolyGameEngine ::updateResultScreenStuff
MonopolyGameEngine ::getButtons
MonopolyGameEngine ::getFont
MonopolyGameEngine ::getFontSize

Here is the caller graph for this function:

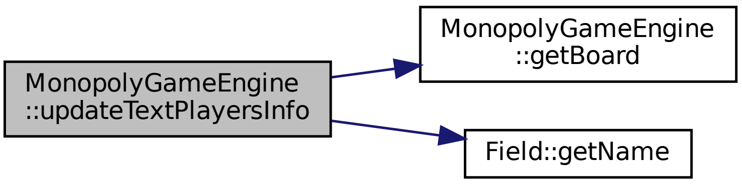

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateResultScreenStuff

6.18.3.87 updateTextPlayersInfo()

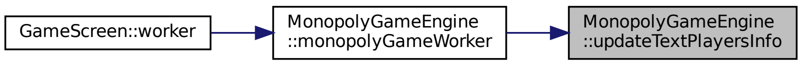
```
void MonopolyGameEngine::updateTextPlayersInfo ( )
```

Updates the text displaying players' information.

Here is the call graph for this function:


MonopolyGameEngine ::updateTextPlayersInfo
MonopolyGameEngine ::getBoard
Field::getName

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::updateTextPlayersInfo

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

- /home/kamil/zpr/Monopoly/MonopolyGameEngine.h
- /home/kamil/zpr/Monopoly/MonopolyGameEngine.cc

6.19 neat::mutation_rate_container Struct Reference

```
#include <Tinyneat.h>
```

Public Member Functions

- void read (std::ifstream &o)
- void write (std::ofstream &o, std::string prefix)

Public Attributes

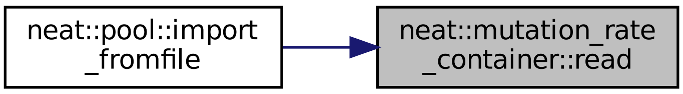
- double connection_mutate_chance = 0.25
- double perturb_chance = 0.90
- double crossover_chance = 0.75
- double link_mutation_chance = 2.0
- double node_mutation_chance = 0.50
- double bias_mutation_chance = 0.40
- double step_size = 0.1
- double disable_mutation_chance = 0.4
- double enable_mutation_chance = 0.2

6.19.1 Member Function Documentation

6.19.1.1 read()

```
void neat::mutation_rate_container::read (
             std::ifstream & o )
```

Here is the caller graph for this function:


neat::pool::import_fromfile
neat::mutation_rate_container::read

6.19.1.2 write()

```
void neat::mutation_rate_container::write (
             std::ofstream & o,
             std::string prefix )
```

6.19.2 Member Data Documentation

6.19.2.1 bias_mutation_chance

```
double neat::mutation_rate_container::bias_mutation_chance = 0.40
```

6.19.2.2 connection_mutate_chance

```
double neat::mutation_rate_container::connection_mutate_chance = 0.25
```

6.19.2.3 crossover_chance

```
double neat::mutation_rate_container::crossover_chance = 0.75
```

6.19.2.4 disable_mutation_chance

```
double neat::mutation_rate_container::disable_mutation_chance = 0.4
```

6.19.2.5 enable_mutation_chance

```
double neat::mutation_rate_container::enable_mutation_chance = 0.2
```

6.19.2.6 link_mutation_chance

```
double neat::mutation_rate_container::link_mutation_chance = 2.0
```

6.19.2.7 node_mutation_chance

```
double neat::mutation_rate_container::node_mutation_chance = 0.50
```

6.19.2.8 perturb_chance

```
double neat::mutation_rate_container::perturb_chance = 0.90
```

6.19.2.9 step_size

```
double neat::mutation_rate_container::step_size = 0.1
```

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

- /home/kamil/zpr/Monopoly/Tinyneat.h
- /home/kamil/zpr/Monopoly/Tinyneat.cc

6.20 neat::network_info_container Struct Reference

```
#include <Tinyneat.h>
```

Public Attributes

- unsigned int input_size
- unsigned int bias_size
- unsigned int output_size
- unsigned int functional_nodes
- bool recurrent

6.20.1 Member Data Documentation

6.20.1.1 bias_size

```
unsigned int neat::network_info_container::bias_size
```

6.20.1.2 functional_nodes

```
unsigned int neat::network_info_container::functional_nodes
```

6.20.1.3 input_size

```
unsigned int neat::network_info_container::input_size
```

6.20.1.4 output_size

```
unsigned int neat::network_info_container::output_size
```

6.20.1.5 recurrent

```
bool neat::network_info_container::recurrent
```

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

- /home/kamil/zpr/Monopoly/Tinyneat.h

6.21 ann::neuralnet Class Reference

```
#include <Tinyann.h>
```

Public Member Functions

- neuralnet ()
- void from_genome (const neat::genome &a)
- void evaluate (const std::vector< double > &input, std::vector< double > &output)
- void import_fromfile (std::string filename)
- void export_tofile (std::string filename)

6.21.1 Constructor & Destructor Documentation

6.21.1.1 neuralnet()

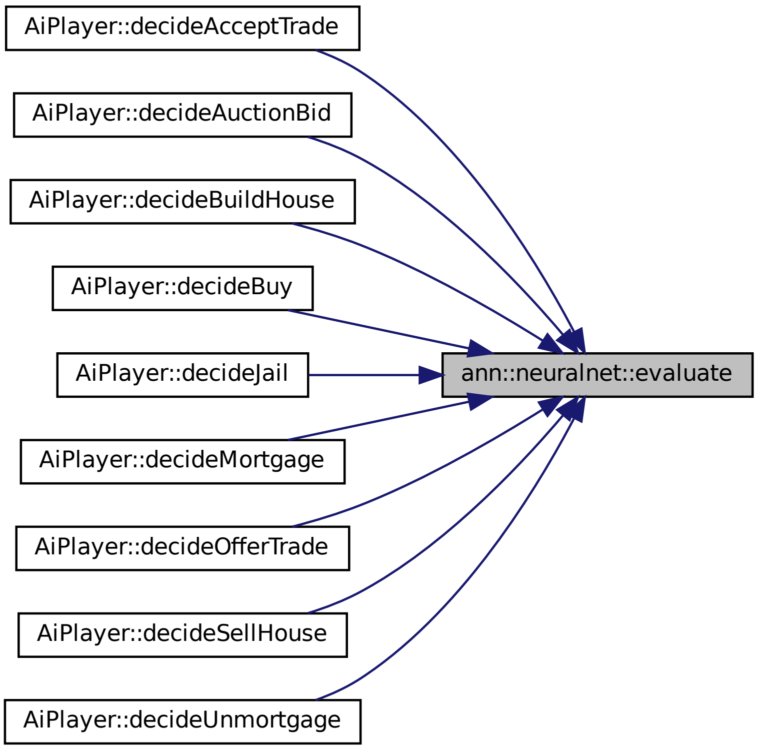
```
ann::neuralnet::neuralnet ( )
```

6.21.2 Member Function Documentation

6.21.2.1 evaluate()

```
void ann::neuralnet::evaluate (
             const std::vector< double > & input,
             std::vector< double > & output )
```

Here is the caller graph for this function:


AiPlayer::decideAcceptTrade
AiPlayer::decideAuctionBid
AiPlayer::decideBuildHouse
AiPlayer::decideBuy
AiPlayer::decideJail
AiPlayer::decideMortgage
AiPlayer::decideOfferTrade
AiPlayer::decideSellHouse
AiPlayer::decideUnmortgage
ann::neuralnet::evaluate

6.21.2.2 export_tofile()

```
void ann::neuralnet::export_tofile (
             std::string filename )
```

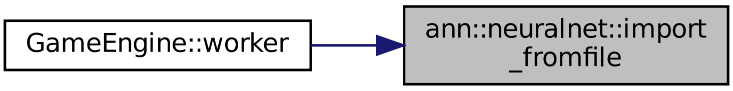
6.21.2.3 from_genome()

```
void ann::neuralnet::from_genome (
             const neat::genome & a )
```

6.21.2.4 import_fromfile()

```
void ann::neuralnet::import_fromfile (
            std::string filename )
```

Here is the caller graph for this function:



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

- /home/kamil/zpr/Monopoly/Tinyann.h
- /home/kamil/zpr/Monopoly/Tinyann.cc

6.22 ann::neuron Class Reference

```
#include <Tinyann.h>
```

Public Member Functions

- neuron ()
- ∼neuron ()

Public Attributes

- int type = 0
- double value = 0.0
- bool visited = false
- std::vector< std::pair< size_t, double > > in_nodes

6.22.1 Constructor & Destructor Documentation

6.22.1.1 neuron()

```
ann::neuron::neuron ( )  [inline]
```

6.22.1.2 ∼neuron()

```
ann::neuron::∼neuron ( ) [inline]
```

6.22.2 Member Data Documentation

6.22.2.1 in_nodes

```
std::vector<std::pair<size_t, double> > ann::neuron::in_nodes
```

6.22.2.2 type

```
int ann::neuron::type = 0
```

6.22.2.3 value

```
double ann::neuron::value = 0.0
```

6.22.2.4 visited

```
bool ann::neuron::visited = false
```

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

- /home/kamil/zpr/Monopoly/Tinyann.h

6.23 NotificationWall Class Reference

Represents a notification wall that displays messages.

```
#include <NotificationWall.h>
```

Public Member Functions

- NotificationWall ()
- void clearWall ()
- std::vector< std::shared_ptr< sf::Text > > & getWall ()
- void addToWall (std::string text)
- unsigned int getFontSize () const
- void setFont (sf::Font font)
- sf::Font & getFont ()

6.23.1 Detailed Description

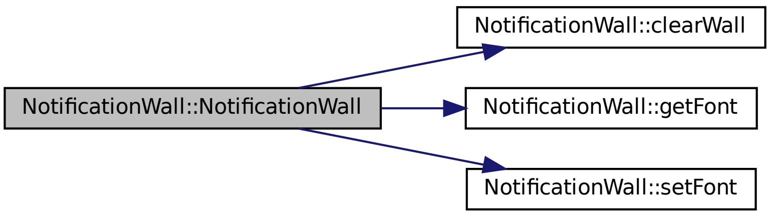
Represents a notification wall that displays messages.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 NotificationWall()

```
NotificationWall::NotificationWall ( )
```

Default constructor for the NotificationWall class. Here is the call graph for this function:


NotificationWall::NotificationWall
NotificationWall::clearWall
NotificationWall::getFont
NotificationWall::setFont

6.23.3 Member Function Documentation

6.23.3.1 addToWall()

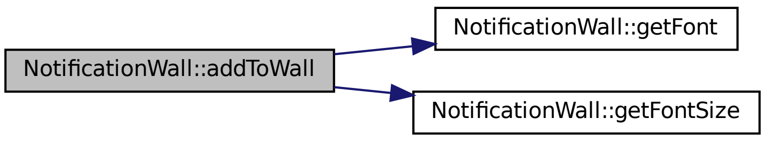
```
void NotificationWall::addToWall (
            std::string text )
```

Add a new message to the notification wall.

Parameters

text	The text of the message to be added.

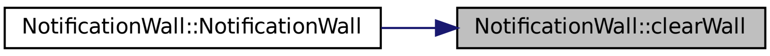
Here is the call graph for this function:


NotificationWall::addToWall
NotificationWall::getFont
NotificationWall::getFontSize

6.23.3.2 clearWall()

```
void NotificationWall::clearWall ( )
```

Clear all notifications from the wall. Here is the caller graph for this function:


NotificationWall::NotificationWall
NotificationWall::clearWall

6.23.3.3 getFont()

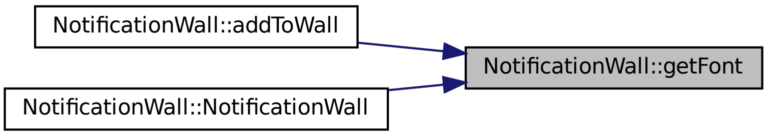
```
sf::Font & NotificationWall::getFont ( )
```

Get the font used for rendering notifications.

Returns

Reference to font of Player object.

Here is the caller graph for this function:


NotificationWall::addToWall
NotificationWall::NotificationWall
NotificationWall::getFont

6.23.3.4 getFontSize()

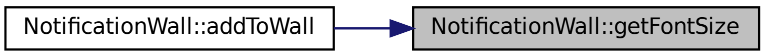
```
unsigned int NotificationWall::getFontSize ( ) const
```

Get the font size used for rendering notifications.

Returns

Used font size for Player object.

Here is the caller graph for this function:



6.23.3.5 getWall()

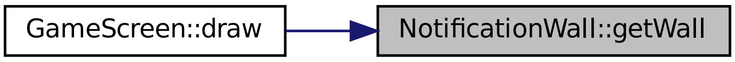
```
std::vector< std::shared_ptr< sf::Text > > & NotificationWall::getWall ( )
```

Get the vector of shared pointers to sf::Text for notifications.

Returns

Reference for vector of pointer to text messages in the wall.

Here is the caller graph for this function:



6.23.3.6 setFont()

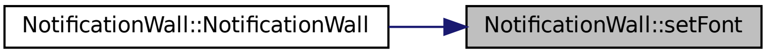
```
void NotificationWall::setFont (
             sf::Font font )
```

Set the font for rendering notifications.

Parameters

font	The font to be set.

Here is the caller graph for this function:



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

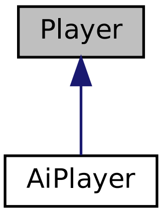
- /home/kamil/zpr/Monopoly/NotificationWall.h
- /home/kamil/zpr/Monopoly/NotificationWall.cc

6.24 Player Class Reference

Represents a player in a Monopoly game.

```
#include <Player.h>
```

Inheritance diagram for Player:



Public Member Functions

- Player ()
- Player (unsigned int money)
- unsigned int getPosition () const
- void setPosition (unsigned int new_position)
- bool hasFieldOwnedId (unsigned int id) const
- std::vector$<$ unsigned int $>$ getFieldOwnedId () const
- void addFieldOwnedId (unsigned int id)
- void removeFieldOwnedId (unsigned int id)
- void clearFieldOwnedId ()
- void setMoney (unsigned int value)
- unsigned int getMoney () const
- void addMoney (unsigned int value)
- bool substractMoney (unsigned int value)
- void setJailStatus (unsigned int new_jail_status)
- unsigned int getJailStatus () const
- void setJailCards (unsigned int new_jail_cards)
- unsigned int getJailCards () const
- void reduceJailStatus ()
- void setId (unsigned int new_id)
- unsigned int getId () const
- void setColor (sf::Color new_color)
- sf::Color getColor () const
- void setIsAi (bool new_is_ai)
- bool getIsAi () const
- void setAiLevel (unsigned int ai_level)
- unsigned int getAiLevel () const
- void setResultPlace (unsigned int place)
- unsigned int getResultPlace () const
- void createSprite ()
- sf::Texture & getTexture ()
- sf::Sprite & getSprite ()
- float getSpriteOffsetX () const
- float getSpriteOffsetY () const
- void setSpriteOffsetX (const float offset_x)
- void setSpriteOffsetY (const float offset_y)

- void setSpritePosition (sf::Vector2f new_pos)
- virtual AiAdapter & getAdapter ()
- virtual ann::neuralnet & getNeuralNetwork ()
- virtual BuyDecision decideBuy (unsigned int index)
- virtual JailDecision decideJail ()
- virtual Decision decideMortgage (unsigned int index)
- virtual Decision decideUnmortgage (unsigned int index)
- virtual unsigned int decideAuctionBid (unsigned int price)
- virtual unsigned int decideBuildHouse ()
- virtual unsigned int decideSellHouse ()
- virtual Decision decideOfferTrade ()
- virtual Decision decideAcceptTrade ()

6.24.1 Detailed Description

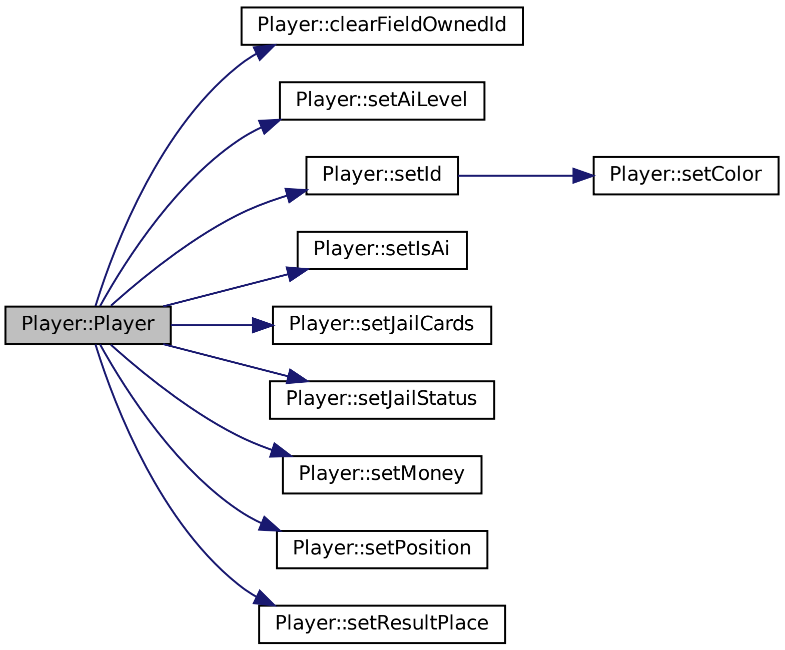
Represents a player in a Monopoly game.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 Player() [1/2]

```
Player::Player ( )
```

Default constructor for Player class. Here is the call graph for this function:



6.24.2.2 Player() [2/2]

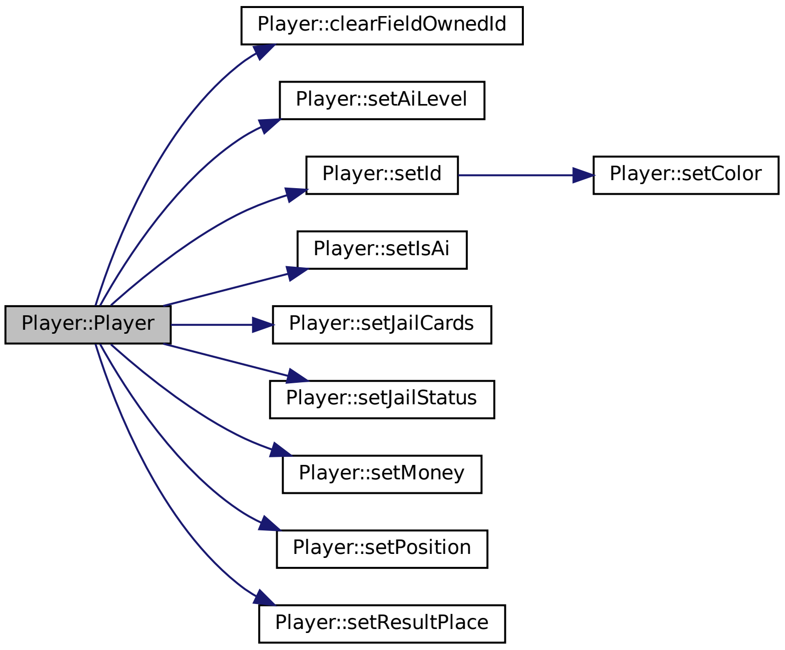
```
Player::Player (
            unsigned int money )
```

Constructor for Player class with initial money.

Parameters

money	Initial amount of money for the player.

Here is the call graph for this function:



6.24.3 Member Function Documentation

6.24.3.1 addFieldOwnedId()

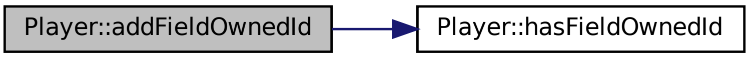
```
void Player::addFieldOwnedId (
            unsigned int id )
```

Add a property with the specified ID to the list of properties owned by the player.

Parameters

id	ID of the property to add.

Here is the call graph for this function:



6.24.3.2 addMoney()

```
void Player::addMoney (
             unsigned int value )
```

Add a specified amount of money to the player's balance.

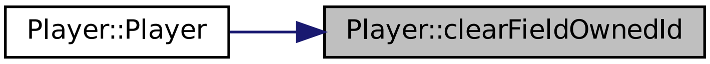
Parameters

value	Amount of money to add.

6.24.3.3 clearFieldOwnedId()

```
void Player::clearFieldOwnedId ( )
```

Clear the list of properties owned by the player. Here is the caller graph for this function:



6.24.3.4 createSprite()

```
void Player::createSprite ( )
```

Create the sprite for the player.

6.24.3.5 decideAcceptTrade()

```
Decision Player::decideAcceptTrade ( )  [virtual]
```

Make a decision for accepting a trade (virtual function, needs to be overridden by derived classes).

Returns

Decision object representing the acceptance of the trade.

Reimplemented in AiPlayer.

6.24.3.6 decideAuctionBid()

```
unsigned int Player::decideAuctionBid (
             unsigned int price )  [virtual]
```

Make a decision for auction bidding (virtual function, needs to be overridden by derived classes).

Parameters

price	Current price in the auction.

Returns

The bid amount decided by the player.

Reimplemented in AiPlayer.

6.24.3.7 decideBuildHouse()

```
unsigned int Player::decideBuildHouse ( )  [virtual]
```

Make a decision for building a house (virtual function, needs to be overridden by derived classes).

Returns

The index of the property on which to build a house.

Reimplemented in AiPlayer.

6.24.3.8 decideBuy()

```
BuyDecision Player::decideBuy (
             unsigned int index )  [virtual]
```

Make a buying decision (virtual function, needs to be overridden by derived classes).

Parameters

index	Index of the property to consider.

Returns

BuyDecision object representing the decision.

Reimplemented in AiPlayer.

6.24.3.9 decideJail()

```
JailDecision Player::decideJail ( )  [virtual]
```

Make a jail decision (virtual function, needs to be overridden by derived classes).

Returns

JailDecision object representing the decision.

Reimplemented in AiPlayer.

6.24.3.10 decideMortgage()

```
Decision Player::decideMortgage (
             unsigned int index )  [virtual]
```

Make a mortgage decision (virtual function, needs to be overridden by derived classes).

Parameters

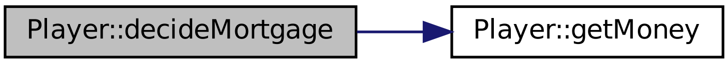
index	Index of the property to consider.

Returns

Decision object representing the mortgage decision.

Reimplemented in AiPlayer.

Here is the call graph for this function:



6.24.3.11 decideOfferTrade()

`Decision Player::decideOfferTrade ( ) [virtual]`

Make a decision for offering a trade (virtual function, needs to be overridden by derived classes).

Returns

Decision object representing the trade offer.

Reimplemented in AiPlayer.

6.24.3.12 decideSellHouse()

`unsigned int Player::decideSellHouse ( ) [virtual]`

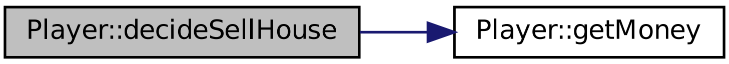
Make a decision for selling a house (virtual function, needs to be overridden by derived classes).

Returns

The index of the property from which to sell a house.

Reimplemented in AiPlayer.

Here is the call graph for this function:



6.24.3.13 decideUnmortgage()

```
Decision Player::decideUnmortgage (
            unsigned int index ) [virtual]
```

Make an unmortgage decision (virtual function, needs to be overridden by derived classes).

Parameters

index	Index of the property to consider.

Returns

Decision object representing the unmortgage decision.

Reimplemented in AiPlayer.

6.24.3.14 getAdapter()

```
AiAdapter & Player::getAdapter ( ) [virtual]
```

Get the AI adapter (virtual function, needs to be overridden by derived classes). DO NOT USE THIS

Returns

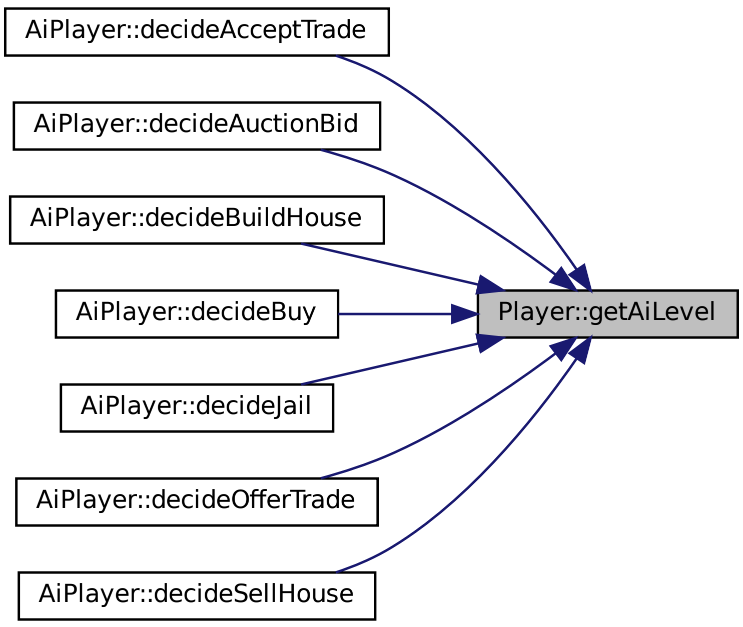
Reference to the AI adapter.

Reimplemented in AiPlayer.

6.24.3.15 getAiLevel()

```
unsigned int Player::getAiLevel ( ) const
```

Get the AI level of the player. Here is the caller graph for this function:


AiPlayer::decideAcceptTrade
AiPlayer::decideAuctionBid
AiPlayer::decideBuildHouse
AiPlayer::decideBuy
AiPlayer::decideJail
AiPlayer::decideOfferTrade
AiPlayer::decideSellHouse
Player::getAiLevel

6.24.3.16 getColor()

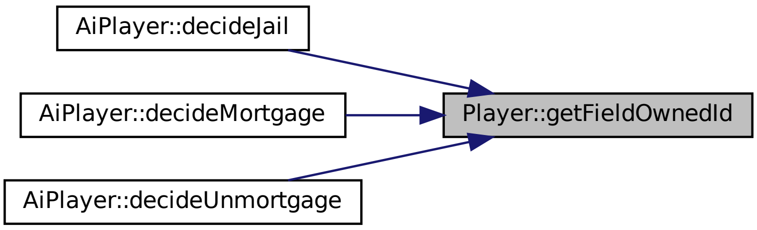
```
sf::Color Player::getColor ( ) const
```

Get the color associated with the player.

6.24.3.17 getFieldOwnedId()

```
std::vector< unsigned int > Player::getFieldOwnedId ( ) const
```

Get a vector of property IDs owned by the player. Here is the caller graph for this function:



6.24.3.18 getId()

```
unsigned int Player::getId ( ) const
```

Get the unique identifier of the player.

6.24.3.19 getIsAi()

```
bool Player::getIsAi ( ) const
```

Get the AI status of the player.

6.24.3.20 getJailCards()

```
unsigned int Player::getJailCards ( ) const
```

Get the current number of jail cards the player has.

6.24.3.21 getJailStatus()

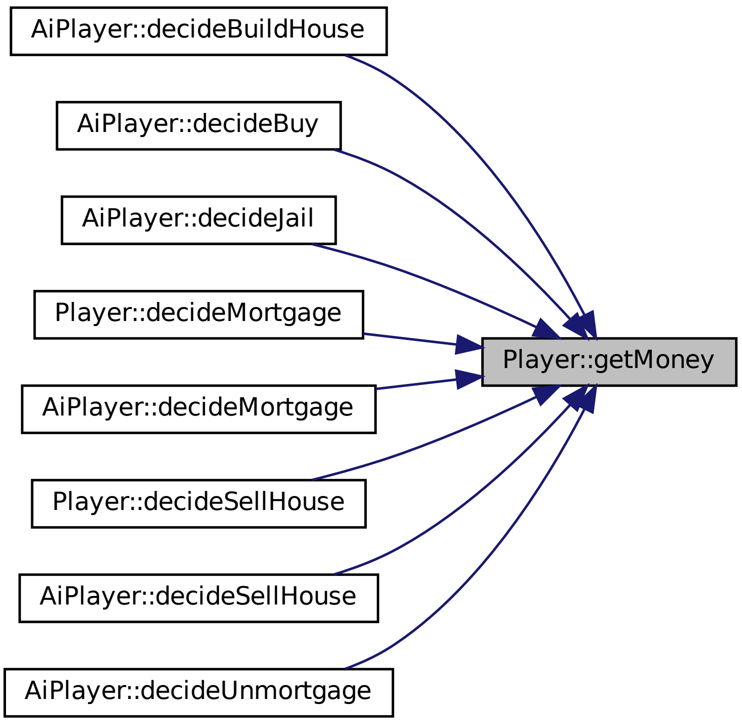
```
unsigned int Player::getJailStatus ( ) const
```

Get the current jail status of the player.

6.24.3.22 getMoney()

```
unsigned int Player::getMoney ( ) const
```

Get the current amount of money the player has. Here is the caller graph for this function:



6.24.3.23 getNeuralNetwork()

```
virtual ann::neuralnet& Player::getNeuralNetwork ( ) [inline], [virtual]
```

Get the Neural Network of a player (virtual function, needs to be overridden by derived classes).

Returns

Reference to the neuralnet class from a tinyai library.

Reimplemented in AiPlayer.

6.24.3.24 getPosition()

```
unsigned int Player::getPosition ( ) const
```

Get the current position of the player.

6.24.3.25 getResultPlace()

```
unsigned int Player::getResultPlace ( ) const
```

Get the final result place of the player in the game.

6.24.3.26 getSprite()

```
sf::Sprite & Player::getSprite ( )
```

Get the sprite representing the player on the game board.

6.24.3.27 getSpriteOffsetX()

```
float Player::getSpriteOffsetX ( ) const
```

Get the offset of the player sprite along the X-axis.

6.24.3.28 getSpriteOffsetY()

```
float Player::getSpriteOffsetY ( ) const
```

Get the offset of the player sprite along the Y-axis.

6.24.3.29 getTexture()

```
sf::Texture & Player::getTexture ( )
```

Get the texture of the player's sprite.

6.24.3.30 hasFieldOwnedId()

```
bool Player::hasFieldOwnedId (
             unsigned int id ) const
```

Check if the player owns a property with the specified ID.

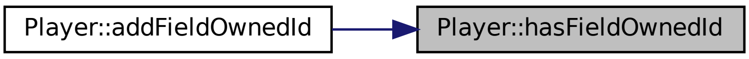
Parameters

id	ID of the property to check.

Returns

True if the player owns the property, false otherwise.

Here is the caller graph for this function:



6.24.3.31 reduceJailStatus()

```
void Player::reduceJailStatus ( )
```

Reduce the jail status of the player by one.

6.24.3.32 removeFieldOwnedId()

```
void Player::removeFieldOwnedId (
             unsigned int id )
```

Remove a property with the specified ID from the list of properties owned by the player.

Parameters

id	ID of the property to remove.

6.24.3.33 setAiLevel()

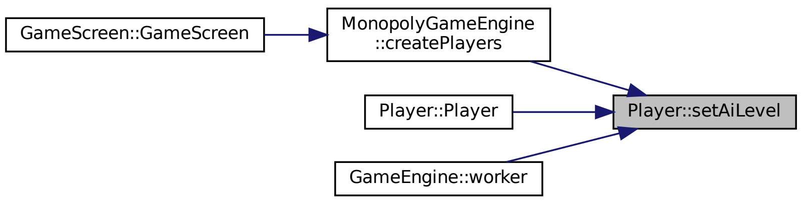
```
void Player::setAiLevel (
             unsigned int ai_level )
```

Set the AI level of the player.

Parameters

ai_level	New AI level for the player.

Here is the caller graph for this function:



6.24.3.34 setColor()

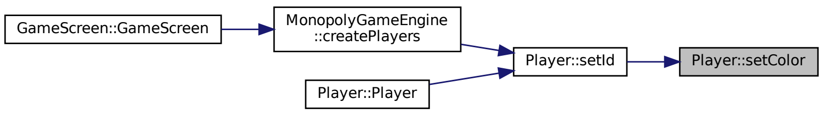
```
void Player::setColor (
             sf::Color new_color )
```

Set the color associated with the player.

Parameters

new_color	New color for the player.

Here is the caller graph for this function:



6.24.3.35 setId()

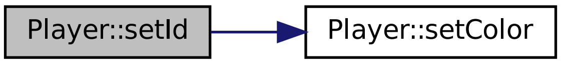
```
void Player::setId (
             unsigned int new_id )
```

Set the unique identifier for the player.

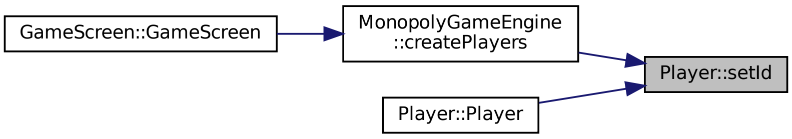
Parameters

new↩ _id	New unique identifier for the player.

Here is the call graph for this function:



Here is the caller graph for this function:



6.24.3.36 setIsAi()

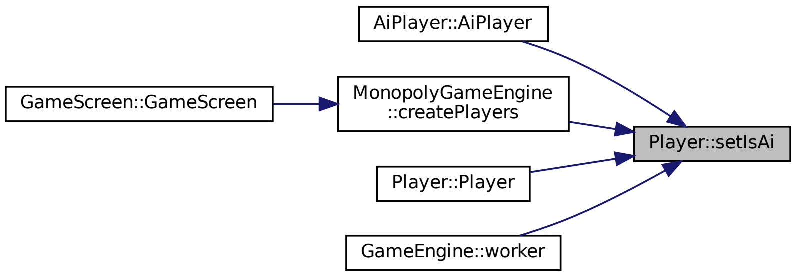
```
void Player::setIsAi (
             bool new_is_ai )
```

Set the AI status of the player.

Parameters

new_is↵ _ai	New AI status for the player.

Here is the caller graph for this function:



6.24.3.37 setJailCards()

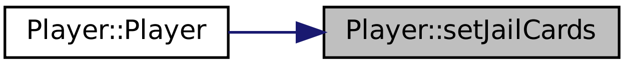
```
void Player::setJailCards (
             unsigned int new_jail_cards )
```

Set the number of jail cards the player has.

Parameters

new_jail_cards	New number of jail cards for the player.

Here is the caller graph for this function:



6.24.3.38 setJailStatus()

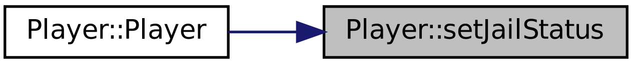
```
void Player::setJailStatus (
             unsigned int new_jail_status )
```

Set the jail status of the player.

Parameters

new_jail_status	New jail status for the player.

Here is the caller graph for this function:



6.24.3.39 setMoney()

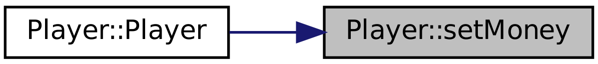
```
void Player::setMoney (
             unsigned int value )
```

Set the amount of money the player has.

Parameters

value	New amount of money for the player.

Here is the caller graph for this function:



6.24.3.40 setPosition()

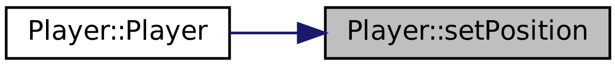
```
void Player::setPosition (
             unsigned int new_position )
```

Set the current position of the player.

Parameters

new_position	New position for the player.

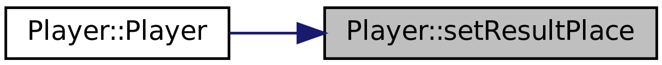
Here is the caller graph for this function:



6.24.3.41 setResultPlace()

```
void Player::setResultPlace (
             unsigned int place )
```

Set the final result place of the player in the game. Here is the caller graph for this function:



6.24.3.42 setSpriteOffsetX()

```
void Player::setSpriteOffsetX (
             const float offset_x )
```

Set the offset of the player sprite along the X-axis.

Parameters

offset← _x	New offset value.

6.24.3.43 setSpriteOffsetY()

```
void Player::setSpriteOffsetY (
             const float offset_y )
```

Set the offset of the player sprite along the Y-axis.

Parameters

offset←↩ _y	New offset value.

6.24.3.44 setSpritePosition()

```
void Player::setSpritePosition (
             sf::Vector2f new_pos )
```

Set the position of the player sprite.

Parameters

new_pos	New position for the player sprite.

6.24.3.45 substractMoney()

```
bool Player::substractMoney (
             unsigned int value )
```

Subtract a specified amount of money from the player's balance.

Parameters

value	Amount of money to subtract.

Returns

True if the player had enough money and the subtraction was successful, false otherwise.

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

- /home/kamil/zpr/Monopoly/Player.h
- /home/kamil/zpr/Monopoly/Player.cc

6.25 playerSettings Struct Reference

Struct describing player settings manipulted from game menu.

```
#include <main.h>
```

Public Attributes

- bool isNone
- bool isHuman
- int level

6.25.1 Detailed Description

Struct describing player settings manipulted from game menu.

6.25.2 Member Data Documentation

6.25.2.1 isHuman

```
bool playerSettings::isHuman
```

6.25.2.2 isNone

```
bool playerSettings::isNone
```

6.25.2.3 level

```
int playerSettings::level
```

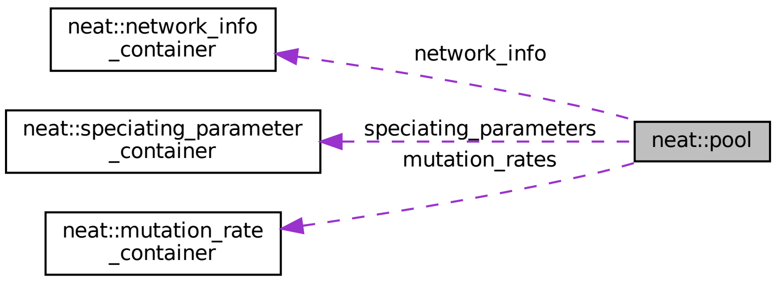
The documentation for this struct was generated from the following file:

- /home/kamil/zpr/Monopoly/main.h

6.26 neat::pool Class Reference

```
#include <Tinyneat.h>
```

Collaboration diagram for neat::pool:


neat::network_info _container
neat::speciating_parameter _container
neat::mutation_rate _container
network_info
speciating_parameters
mutation_rates
neat::pool

Public Member Functions

- pool (unsigned int input, unsigned int output, unsigned int bias=1, bool rec=false)
- void new_generation ()
- unsigned int generation ()
- std::vector< std::pair< specie ∗, genome ∗ > > get_genomes ()
- void import_fromfile (std::string filename)
- void export_tofile (std::string filename)

Public Attributes

- unsigned int max_fitness = 0
- mutation_rate_container mutation_rates
- speciating_parameter_container speciating_parameters
- network_info_container network_info
- std::random_device rd
- std::mt19937 generator
- std::list< specie > species

6.26.1 Constructor & Destructor Documentation

6.26.1.1 pool()

```
neat::pool::pool (
            unsigned int input,
            unsigned int output,
            unsigned int bias = 1,
            bool rec = false )  [inline]
```

6.26.2 Member Function Documentation

6.26.2.1 export_tofile()

```
void neat::pool::export_tofile (
            std::string filename )
```

6.26.2.2 generation()

```
unsigned int neat::pool::generation ( ) [inline]
```

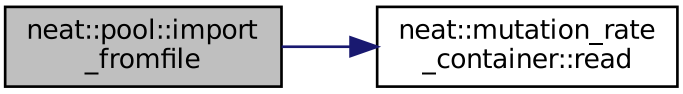
6.26.2.3 get_genomes()

```
std::vector<std::pair<specie*, genome*> > neat::pool::get_genomes ( ) [inline]
```

6.26.2.4 import_fromfile()

```
void neat::pool::import_fromfile (
            std::string filename )
```

Here is the call graph for this function:


neat::pool::import_fromfile
neat::mutation_rate_container::read

6.26.2.5 new_generation()

```
void neat::pool::new_generation ( )
```

6.26.3 Member Data Documentation

6.26.3.1 generator

```
std::mt19937 neat::pool::generator
```

6.26.3.2 max_fitness

```
unsigned int neat::pool::max_fitness = 0
```

6.26.3.3 mutation_rates

```
mutation_rate_container neat::pool::mutation_rates
```

6.26.3.4 network_info

```
network_info_container neat::pool::network_info
```

6.26.3.5 rd

```
std::random_device neat::pool::rd
```

6.26.3.6 speciating_parameters

```
speciating_parameter_container neat::pool::speciating_parameters
```

6.26.3.7 species

```
std::list<specie> neat::pool::species
```

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

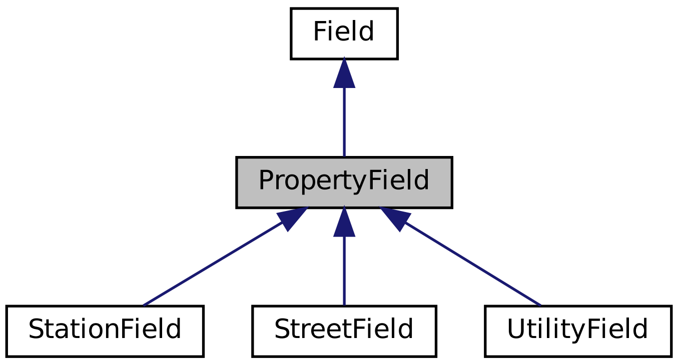
- /home/kamil/zpr/Monopoly/Tinyneat.h
- /home/kamil/zpr/Monopoly/Tinyneat.cc

6.27 PropertyField Class Reference

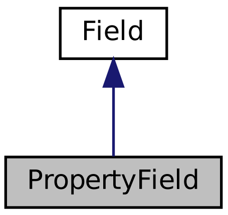
Derived class representing a property field on the game board.

```
#include <Field.h>
```

Inheritance diagram for PropertyField:



Collaboration diagram for PropertyField:



Public Member Functions

- PropertyField (const unsigned int id, const FieldType type, const std::string name, const std::string graphic←_path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position, const unsigned int price, const std::vector< unsigned int > group_members, const unsigned int mortgage)

  Constructor for the PropertyField class.
- unsigned int getPrice ()

  Gets the price of the property.
- const std::vector< unsigned int > getGroupMembers ()

  Gets the IDs of the group members (properties in the same group).

- unsigned int getMortgage ()

  Gets the mortgage value of the property.
- bool getIsMortgaged ()

  Checks if the property is mortgaged.
- unsigned int getUnmortgageValue ()

  Gets the value needed to unmortgage the property.
- std::shared_ptr< Player > getOwner ()

  Gets the owner of the property.
- sf::RectangleShape & getOwnerFlag ()

  Gets the owner flag sprite.
- void setIsMortgaged (bool new_state)

  Sets the mortgaged state of the property.
- void setOwner (std::shared_ptr< Player > new_owner)

  Sets the owner of the property.
- void resetOwner ()

  Resets the owner of the property.
- void resetDefault ()

  Resets the property to its default state.
- void createFlagSprite ()

  Creates the owner flag sprite.

6.27.1 Detailed Description

Derived class representing a property field on the game board.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 PropertyField()

```
PropertyField::PropertyField (
            const unsigned int id,
            const FieldType type,
            const std::string name,
            const std::string graphic_path,
            const unsigned int width,
            const unsigned int height,
            const float rotation,
            const sf::Vector2i position,
            const unsigned int price,
            const std::vector< unsigned int > group_members,
            const unsigned int mortgage )  [inline]
```

Constructor for the PropertyField class.

Parameters

id	The ID of the field.
type	The type of the field.

Parameters

name	The name of the field.
graphic_path	The file path to the field's graphic.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the board.
price	The price of the property.
group_members	The IDs of the group members (properties in the same group).
mortgage	The mortgage value of the property.

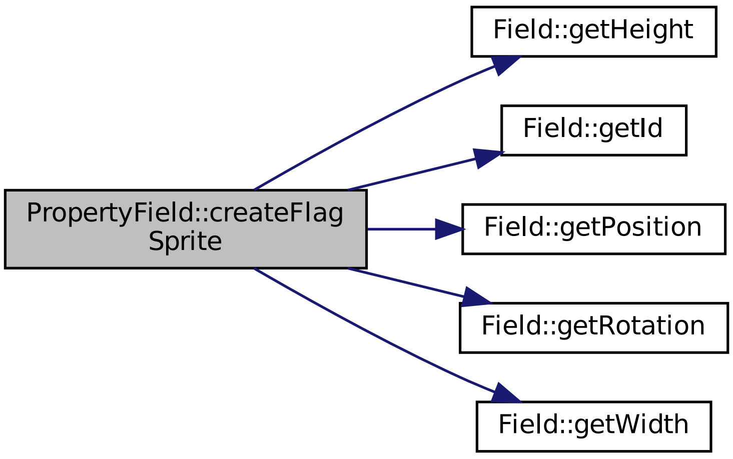
6.27.3 Member Function Documentation

6.27.3.1 createFlagSprite()

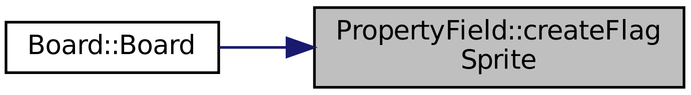
```
void PropertyField::createFlagSprite ( )
```

Creates the owner flag sprite.

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.3.2 getGroupMembers()

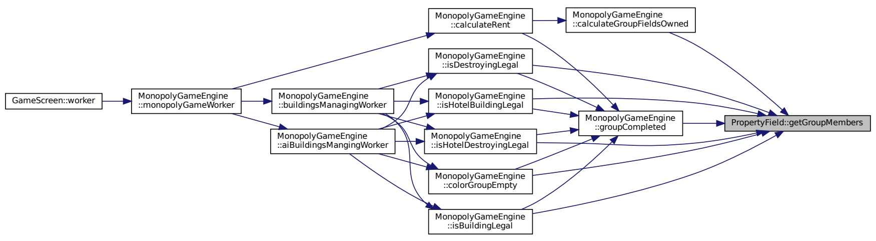
```
const std::vector< unsigned int > PropertyField::getGroupMembers ( )
```

Gets the IDs of the group members (properties in the same group).

Returns

The IDs of the group members.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::isDestroyingLegal
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::calculateGroupFieldsOwned
MonopolyGameEngine ::groupCompleted
PropertyField::getGroupMembers

6.27.3.3 getIsMortgaged()

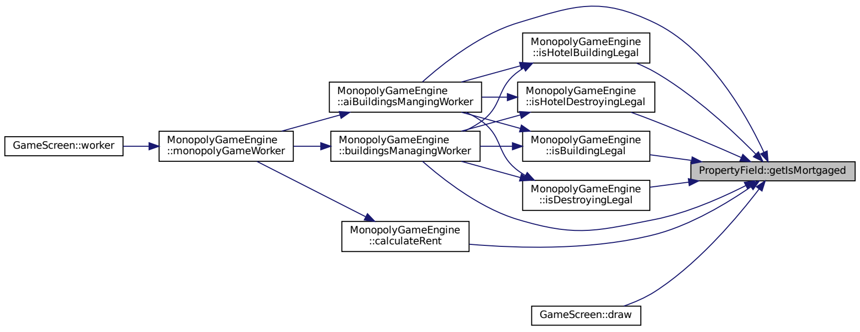
```
bool PropertyField::getIsMortgaged ( )
```

Checks if the property is mortgaged.

Returns

True if the property is mortgaged, false otherwise.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::isHotelDestroyingLegal
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::isDestroyingLegal
GameScreen::draw
PropertyField::getIsMortgaged

6.27.3.4 getMortgage()

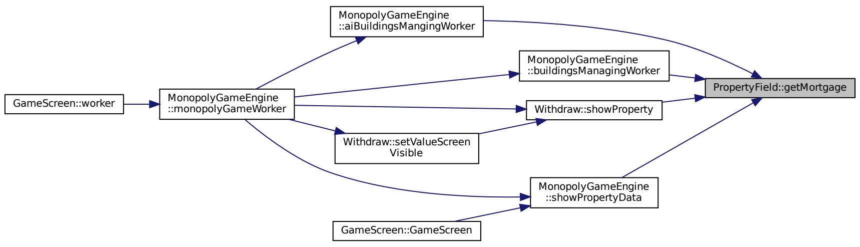
```
unsigned int PropertyField::getMortgage ( )
```

Gets the mortgage value of the property.

Returns

The mortgage value of the property.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
Withdraw::showProperty
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen
PropertyField::getMortgage

6.27.3.5 getOwner()

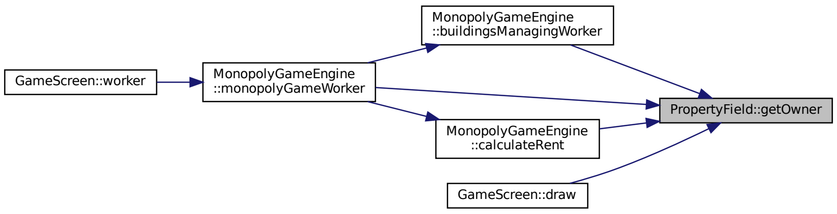
```
std::shared_ptr< Player > PropertyField::getOwner ( )
```

Gets the owner of the property.

Returns

A shared pointer to the owner of the property.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::calculateRent
GameScreen::draw
PropertyField::getOwner

6.27.3.6 getOwnerFlag()

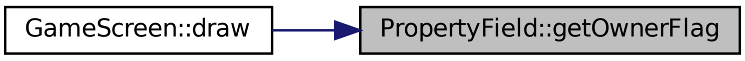
```
sf::RectangleShape & PropertyField::getOwnerFlag ( )
```

Gets the owner flag sprite.

Returns

The owner flag sprite.

Here is the caller graph for this function:



6.27.3.7 getPrice()

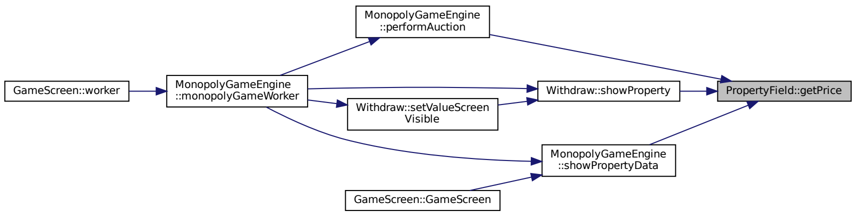
```
unsigned int PropertyField::getPrice ( )
```

Gets the price of the property.

Returns

The price of the property.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::performAuction
Withdraw::setValueScreen Visible
Withdraw::showProperty
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen
PropertyField::getPrice

6.27.3.8 getUnmortgageValue()

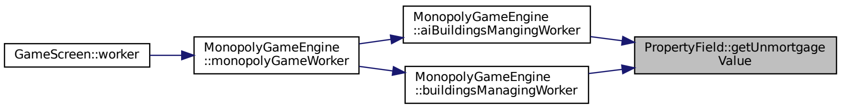
```
unsigned int PropertyField::getUnmortgageValue ( )
```

Gets the value needed to unmortgage the property.

Returns

The value needed to unmortgage the property.

Here is the caller graph for this function:

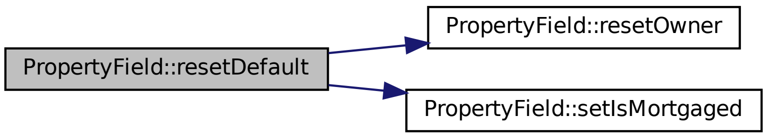

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
PropertyField::getUnmortgage Value

6.27.3.9 resetDefault()

```
void PropertyField::resetDefault ( )
```

Resets the property to its default state.

Here is the call graph for this function:

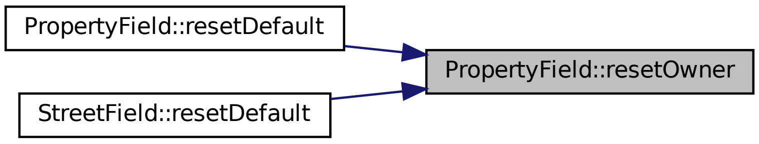

PropertyField::resetDefault
PropertyField::resetOwner
PropertyField::setIsMortgaged

6.27.3.10 resetOwner()

```
void PropertyField::resetOwner ( )
```

Resets the owner of the property.

Here is the caller graph for this function:


PropertyField::resetDefault
StreetField::resetDefault
PropertyField::resetOwner

6.27.3.11 setIsMortgaged()

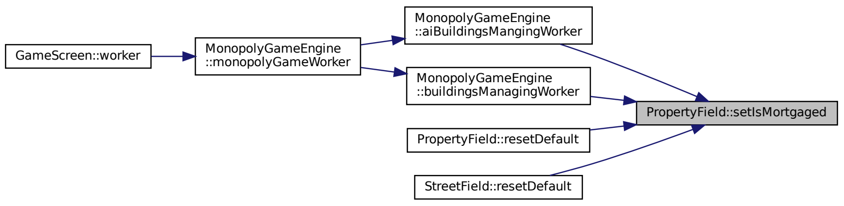
```
void PropertyField::setIsMortgaged (
             bool new_state )
```

Sets the mortgaged state of the property.

Parameters

new_state	The new mortgaged state.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
PropertyField::resetDefault
StreetField::resetDefault
PropertyField::setIsMortgaged

6.27.3.12 setOwner()

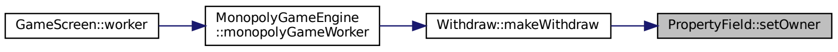
```
void PropertyField::setOwner (
             std::shared_ptr< Player > new_owner )
```

Sets the owner of the property.

Parameters

new_owner	A shared pointer to the new owner.

Here is the caller graph for this function:



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

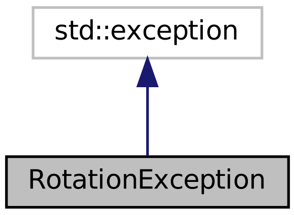
- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.28 RotationException Class Reference

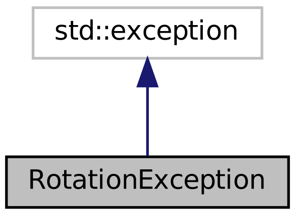
Exception for handling passing wrong rotation to any displayed object.

```
#include <main.h>
```

Inheritance diagram for RotationException:



Collaboration diagram for RotationException:



Public Member Functions

- RotationException (float rotation)
- RotationException (const RotationException &e) throw ()
- float getBadRotation ()

6.28.1 Detailed Description

Exception for handling passing wrong rotation to any displayed object.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 RotationException() [1/2]

```
RotationException::RotationException (
            float rotation )  [inline]
```

6.28.2.2 RotationException() [2/2]

```
RotationException::RotationException (
            const RotationException & e ) throw ( )   [inline]
```

6.28.3 Member Function Documentation

6.28.3.1 getBadRotation()

```
float RotationException::getBadRotation ( )  [inline]
```

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

- /home/kamil/zpr/Monopoly/main.h

6.29 neat::speciating_parameter_container Struct Reference

```
#include <Tinyneat.h>
```

Public Member Functions

- void read (std::ifstream &o)
- void write (std::ofstream &o, std::string prefix)

Public Attributes

- unsigned int population = 240
- double delta_disjoint = 2.0
- double delta_weights = 0.4
- double delta_threshold = 1.3
- unsigned int stale_species = 15

6.29.1 Member Function Documentation

6.29.1.1 read()

```
void neat::speciating_parameter_container::read (
            std::ifstream & o )
```

6.29.1.2 write()

```
void neat::speciating_parameter_container::write (
             std::ofstream & o,
             std::string prefix )
```

6.29.2 Member Data Documentation

6.29.2.1 delta_disjoint

```
double neat::speciating_parameter_container::delta_disjoint = 2.0
```

6.29.2.2 delta_threshold

```
double neat::speciating_parameter_container::delta_threshold = 1.3
```

6.29.2.3 delta_weights

```
double neat::speciating_parameter_container::delta_weights = 0.4
```

6.29.2.4 population

```
unsigned int neat::speciating_parameter_container::population = 240
```

6.29.2.5 stale_species

```
unsigned int neat::speciating_parameter_container::stale_species = 15
```

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

- /home/kamil/zpr/Monopoly/Tinyneat.h
- /home/kamil/zpr/Monopoly/Tinyneat.cc

6.30 neat::specie Struct Reference

```
#include <Tinyneat.h>
```

Public Attributes

- unsigned int top_fitness = 0
- unsigned int average_fitness = 0
- unsigned int staleness = 0
- std::vector< genome > genomes

6.30.1 Member Data Documentation

6.30.1.1 average_fitness

```
unsigned int neat::specie::average_fitness = 0
```

6.30.1.2 genomes

```
std::vector<genome> neat::specie::genomes
```

6.30.1.3 staleness

```
unsigned int neat::specie::staleness = 0
```

6.30.1.4 top_fitness

```
unsigned int neat::specie::top_fitness = 0
```

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

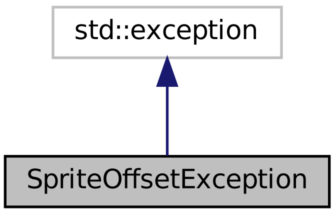
- /home/kamil/zpr/Monopoly/Tinyneat.h

6.31 SpriteOffsetException Class Reference

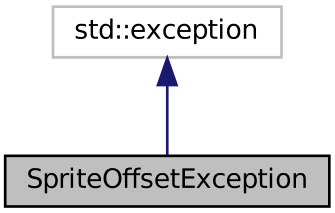
Exception for handling wrong offset to any displayed object.

`#include <main.h>`

Inheritance diagram for SpriteOffsetException:



Collaboration diagram for SpriteOffsetException:



Public Member Functions

- SpriteOffsetException (float offset)
- SpriteOffsetException (const SpriteOffsetException &e) throw ()
- float getBadOffset ()

6.31.1 Detailed Description

Exception for handling wrong offset to any displayed object.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 SpriteOffsetException() [1/2]

```
SpriteOffsetException::SpriteOffsetException (
            float offset )   [inline]
```

6.31.2.2 SpriteOffsetException() [2/2]

```
SpriteOffsetException::SpriteOffsetException (
            const SpriteOffsetException & e ) throw ( )    [inline]
```

6.31.3 Member Function Documentation

6.31.3.1 getBadOffset()

```
float SpriteOffsetException::getBadOffset ( )  [inline]
```

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

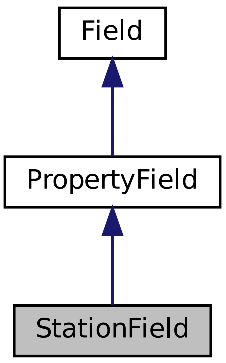
- /home/kamil/zpr/Monopoly/main.h

6.32 StationField Class Reference

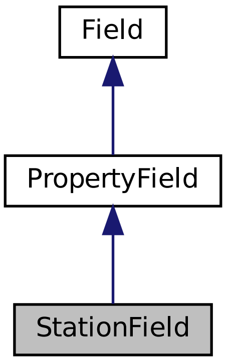
Derived class representing a station field on the game board.

`#include <Field.h>`

Inheritance diagram for StationField:



Collaboration diagram for StationField:



Public Member Functions

- StationField (const unsigned int id, const FieldType type, const std::string name, const std::string graphic_↩ path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position, const unsigned int price, const std::map< StationTiers, unsigned int > rent_values, const std::vector< unsigned int > group_members, const unsigned int Mortgage)

  Constructor for the StationField class.
- const std::map< StationTiers, unsigned int > getRentValues ()

  Gets the rent values for different tiers.
- unsigned int calculateRent ()

  Calculates the rent for the property.

6.32.1 Detailed Description

Derived class representing a station field on the game board.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 StationField()

```
StationField::StationField (
            const unsigned int id,
            const FieldType type,
            const std::string name,
            const std::string graphic_path,
            const unsigned int width,
            const unsigned int height,
            const float rotation,
            const sf::Vector2i position,
            const unsigned int price,
            const std::map< StationTiers, unsigned int > rent_values,
            const std::vector< unsigned int > group_members,
            const unsigned int Mortgage )  [inline]
```

Constructor for the StationField class.

Parameters

id	The ID of the field.
type	The type of the field.
name	The name of the field.
graphic_path	The file path to the field's graphic.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the board.
price	The price of the property.
rent_values	The rent values for different tiers.
group_members	The IDs of the group members (properties in the same group).
mortgage	The mortgage value of the property.

6.32.3 Member Function Documentation

6.32.3.1 calculateRent()

```
unsigned int StationField::calculateRent ( )
```

Calculates the rent for the property.

Returns

The calculated rent value.

6.32.3.2 getRentValues()

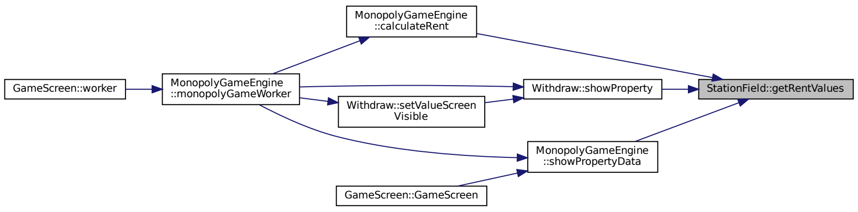
```
const std::map< StationTiers, unsigned int > StationField::getRentValues ( )
```

Gets the rent values for different tiers.

Returns

The rent values for different tiers.

Here is the caller graph for this function:


MonopolyGameEngine ::calculateRent
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::showProperty
StationField::getRentValues
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen

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

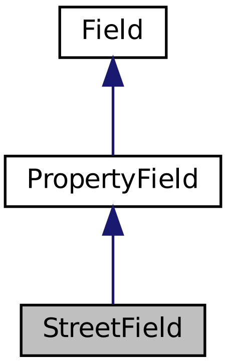
- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.33 StreetField Class Reference

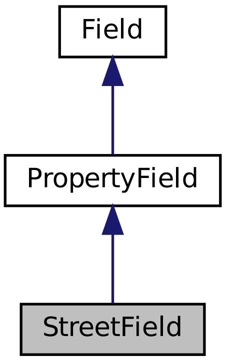
Derived class representing a street field on the game board.

```
#include <Field.h>
```

Inheritance diagram for StreetField:



Collaboration diagram for StreetField:



Public Member Functions

- StreetField (const unsigned int id, const FieldType type, const std::string name, const std::string graphic↩ _path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position, const unsigned int price, const unsigned int house_price, const unsigned int hotel_price, const std::map< StreetTiers, unsigned int > rent_values, const std::vector< unsigned int > group_members, const unsigned int Mortgage)

    Constructor for the StreetField class.
- const std::map< StreetTiers, unsigned int > getRentValues ()

    Gets the rent values for different tiers.
- unsigned int getHousePrice ()

    Gets the price of a house.
- unsigned int getHotelPrice ()

    Gets the price of a hotel.
- unsigned int getHouseNumber ()

    Gets the number of houses on the property.
- bool getIsHotel ()

    Checks if there is a hotel on the property.
- void setHouseNumber (unsigned int new_house_number)

    Sets the number of houses on the property.
- void setIsHotel (bool new_state)

    Sets the hotel state on the property.
- void resetDefault ()

    Resets the property to its default state.
- unsigned int calculateRent ()

    Calculates the rent for the property.

6.33.1 Detailed Description

Derived class representing a street field on the game board.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 StreetField()

```
StreetField::StreetField (
             const unsigned int id,
             const FieldType type,
             const std::string name,
             const std::string graphic_path,
             const unsigned int width,
             const unsigned int height,
             const float rotation,
             const sf::Vector2i position,
             const unsigned int price,
             const unsigned int house_price,
             const unsigned int hotel_price,
             const std::map< StreetTiers, unsigned int > rent_values,
             const std::vector< unsigned int > group_members,
             const unsigned int Mortgage )  [inline]
```

Constructor for the StreetField class.

Parameters

id	The ID of the field.
type	The type of the field.
name	The name of the field.
graphic_path	The file path to the field's graphic.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the board.
price	The price of the property.
house_price	The price of a house.
hotel_price	The price of a hotel.
rent_values	The rent values for different tiers.
group_members	The IDs of the group members (properties in the same group).
mortgage	The mortgage value of the property.

6.33.3 Member Function Documentation

6.33.3.1 calculateRent()

```
unsigned int StreetField::calculateRent ( )
```

Calculates the rent for the property.

Returns

The calculated rent value.

6.33.3.2 getHotelPrice()

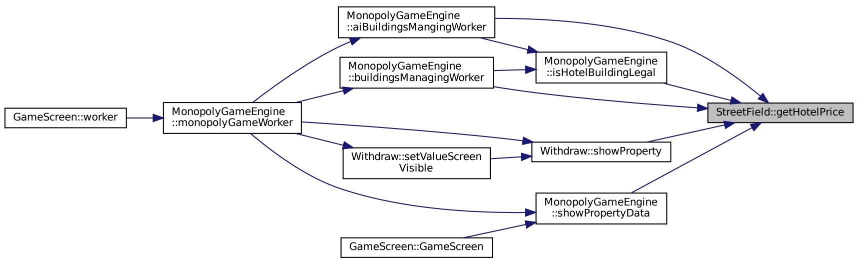
`unsigned int StreetField::getHotelPrice ( )`

Gets the price of a hotel.

Returns

The price of a hotel.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
Withdraw::setValueScreen Visible
GameScreen::GameScreen
MonopolyGameEngine ::isHotelBuildingLegal
Withdraw::showProperty
MonopolyGameEngine ::showPropertyData
StreetField::getHotelPrice

6.33.3.3 getHouseNumber()

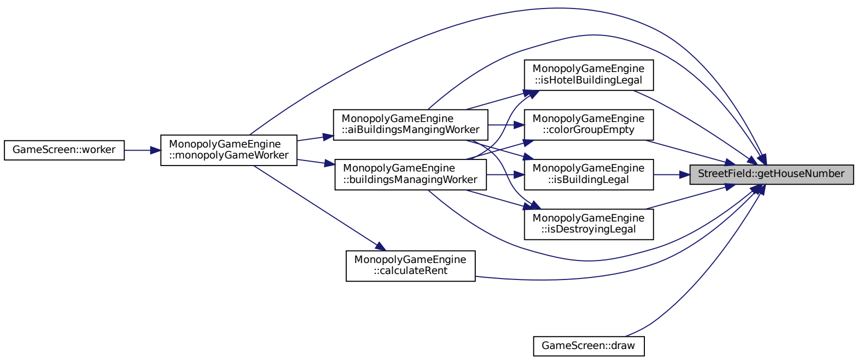
`unsigned int StreetField::getHouseNumber ( )`

Gets the number of houses on the property.

Returns

The number of houses on the property.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::calculateRent
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::colorGroupEmpty
MonopolyGameEngine ::isBuildingLegal
MonopolyGameEngine ::isDestroyingLegal
GameScreen::draw
StreetField::getHouseNumber

6.33.3.4 getHousePrice()

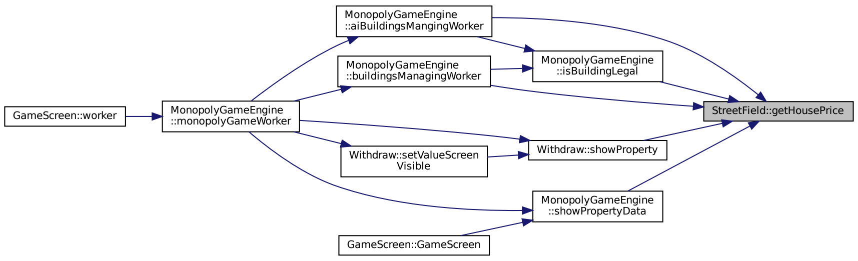
```
unsigned int StreetField::getHousePrice ( )
```

Gets the price of a house.

Returns

The price of a house.

Here is the caller graph for this function:


MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isBuildingLegal
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
StreetField::getHousePrice
Withdraw::showProperty
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen

6.33.3.5 getIsHotel()

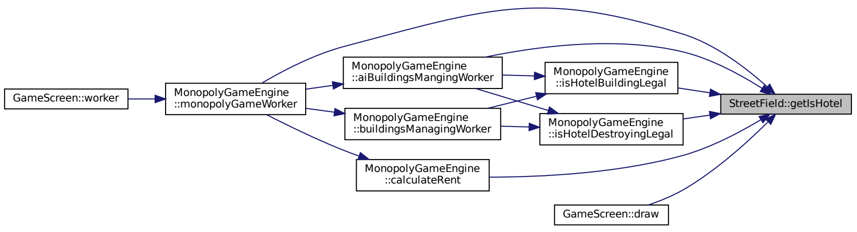
```
bool StreetField::getIsHotel ( )
```

Checks if there is a hotel on the property.

Returns

True if there is a hotel, false otherwise.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
MonopolyGameEngine ::isHotelBuildingLegal
MonopolyGameEngine ::isHotelDestroyingLegal
StreetField::getIsHotel
MonopolyGameEngine ::calculateRent
GameScreen::draw

6.33.3.6 getRentValues()

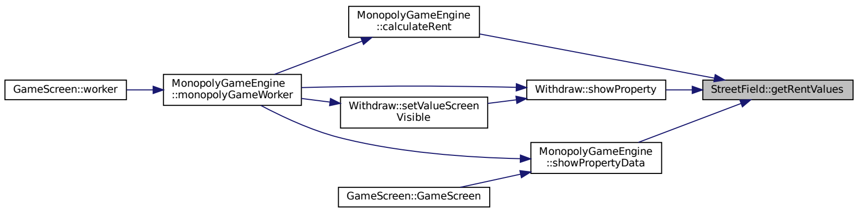
```
const std::map< StreetTiers, unsigned int > StreetField::getRentValues ( )
```

Gets the rent values for different tiers.

Returns

The rent values for different tiers.

Here is the caller graph for this function:

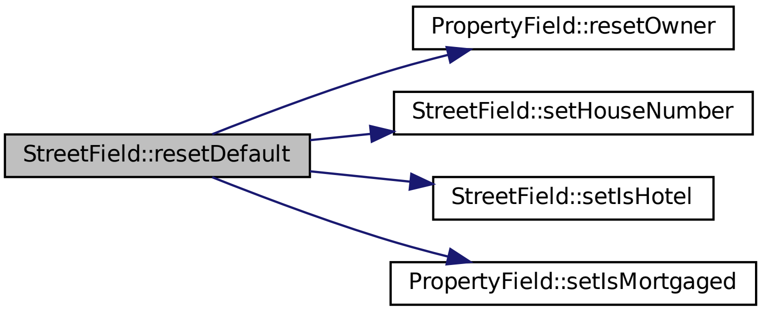

MonopolyGameEngine ::calculateRent
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::showProperty
StreetField::getRentValues
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen

6.33.3.7 resetDefault()

```
void StreetField::resetDefault ( )
```

Resets the property to its default state.

Here is the call graph for this function:


PropertyField::resetOwner
StreetField::setHouseNumber
StreetField::resetDefault
StreetField::setIsHotel
PropertyField::setIsMortgaged

6.33.3.8 setHouseNumber()

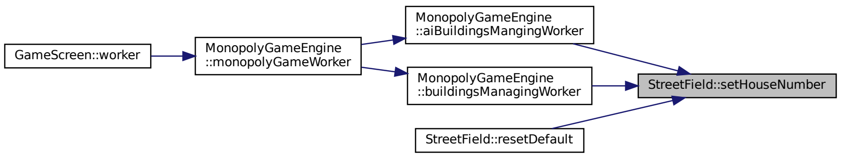
```
void StreetField::setHouseNumber (
            unsigned int new_house_number )
```

Sets the number of houses on the property.

Parameters

new_house_number	The new number of houses on the property.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
StreetField::resetDefault
StreetField::setHouseNumber

6.33.3.9 setIsHotel()

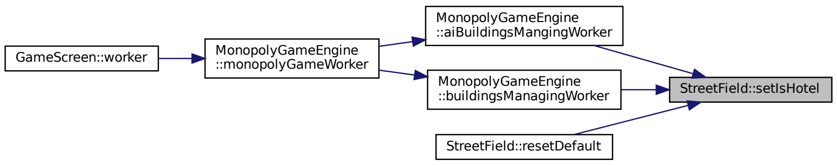
```
void StreetField::setIsHotel (
             bool new_state )
```

Sets the hotel state on the property.

Parameters

new_state	The new hotel state.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
MonopolyGameEngine ::aiBuildingsMangingWorker
MonopolyGameEngine ::buildingsManagingWorker
StreetField::resetDefault
StreetField::setIsHotel

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

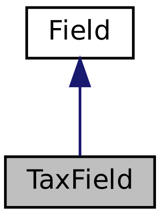
- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.34 TaxField Class Reference

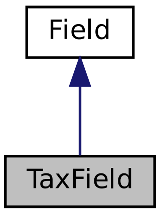
Derived class representing a tax field on the monopoly board.

```
#include <Field.h>
```

Inheritance diagram for TaxField:



Collaboration diagram for TaxField:



Public Member Functions

- TaxField (const unsigned int id, const FieldType type, const std::string name, const std::string graphic_path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position, const unsigned int tax_value)

  Constructor for the TaxField class.
- unsigned int getTaxValue ()

  Calculates the tax for Tax property.

6.34.1 Detailed Description

Derived class representing a tax field on the monopoly board.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 TaxField()

```
TaxField::TaxField (
             const unsigned int id,
             const FieldType type,
             const std::string name,
             const std::string graphic_path,
             const unsigned int width,
             const unsigned int height,
             const float rotation,
             const sf::Vector2i position,
             const unsigned int tax_value )  [inline]
```

Constructor for the TaxField class.

Parameters

id	The unique identifier of the field.
type	The type of the field.
name	The name of the field.
graphic_path	The file path to the graphic representation of the field.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the game board.
tax_value	The value of the tax.

6.34.3 Member Function Documentation

6.34.3.1 getTaxValue()

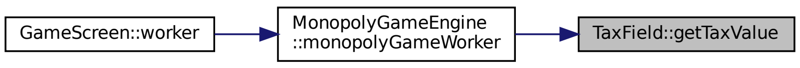
```
unsigned int TaxField::getTaxValue ( )
```

Calculates the tax for Tax property.

Returns

The calculated tax value.

Here is the caller graph for this function:



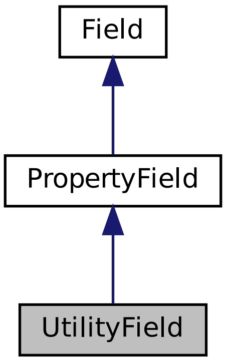
The documentation for this class was generated from the following files:

- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

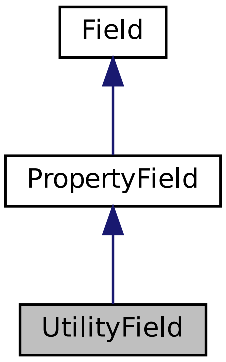
6.35 UtilityField Class Reference

`#include <Field.h>`

Inheritance diagram for UtilityField:



Collaboration diagram for UtilityField:



Public Member Functions

- UtilityField (const unsigned int id, const FieldType type, const std::string name, const std::string graphic_↩ path, const unsigned int width, const unsigned int height, const float rotation, const sf::Vector2i position, const unsigned int price, const std::map< UtilityTiers, unsigned int > rent_multipliers, const std::vector< unsigned int > group_members, const unsigned int Mortgage)

    Constructor for the UtilityField class.
- const std::map< UtilityTiers, unsigned int > getRentMultipliers ()

    Calculates the rent for the utility property.
- unsigned int calculateRent (unsigned int dice_roll)

    Calculates the rent for the utility property.

6.35.1 Constructor & Destructor Documentation

6.35.1.1 UtilityField()

```
UtilityField::UtilityField (
            const unsigned int id,
            const FieldType type,
            const std::string name,
            const std::string graphic_path,
            const unsigned int width,
            const unsigned int height,
            const float rotation,
            const sf::Vector2i position,
            const unsigned int price,
            const std::map< UtilityTiers, unsigned int > rent_multipliers,
            const std::vector< unsigned int > group_members,
            const unsigned int Mortgage )  [inline]
```

Constructor for the UtilityField class.

Parameters

id	The ID of the field.
type	The type of the field.
name	The name of the field.
graphic_path	The file path to the field's graphic.
width	The width of the field.
height	The height of the field.
rotation	The rotation angle of the field.
position	The position of the field on the board.
price	The price of the property.
rent_multipliers	The rent multipliers for different tiers.
group_members	The IDs of the group members (properties in the same group).
mortgage	The mortgage value of the property.

6.35.2 Member Function Documentation

6.35.2.1 calculateRent()

```
unsigned int UtilityField::calculateRent (
            unsigned int dice_roll )
```

Calculates the rent for the utility property.

Parameters

dice_roll	The roll of the dice.

Returns

The calculated rent value.

6.35.2.2 getRentMultipliers()

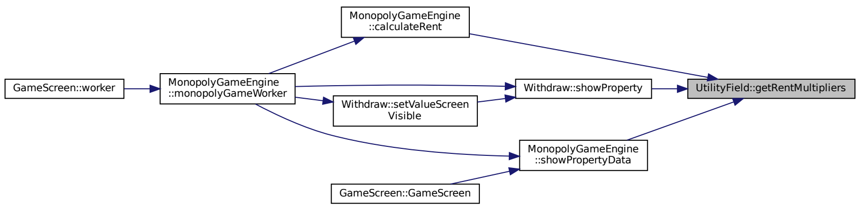
```
const std::map< UtilityTiers, unsigned int > UtilityField::getRentMultipliers ( )
```

Calculates the rent for the utility property.

Returns

Map for Utility tiers mapped to mulitpliers values

Here is the caller graph for this function:


MonopolyGameEngine ::calculateRent
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::showProperty
UtilityField::getRentMultipliers
Withdraw::setValueScreen Visible
MonopolyGameEngine ::showPropertyData
GameScreen::GameScreen

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

- /home/kamil/zpr/Monopoly/Field.h
- /home/kamil/zpr/Monopoly/Field.cc

6.36 Withdraw Class Reference

Represents the trade and withdraw mechanism in a monopoly game.

```
#include <Withdraw.h>
```

Public Member Functions

- Withdraw ()

  Constructor for the Withdraw class.
- std::vector< unsigned int > getPlayer1IndexProperties ()

  Getter for the properties of Player 1 offer.
- std::vector< unsigned int > getPlayer2IndexProperties ()

  Getter for the properties of Player 2 offer.
- void setPlayer1IndexProperties (std::vector< unsigned int > new_index_properties)

  Setter for the properties of Player 1 offer.
- void setPlayer2IndexProperties (std::vector< unsigned int > new_index_properties)

  Setter for the properties of Player 2 offer.
- void setBoard (std::shared_ptr< Board > board_ptr)

  Setter for the gameboard.
- void setTurnState (TurnState state)

  Setter for the turn state of monopolygame when withdraw was started.
- TurnState getTurnState ()

  Getter for the turn state.
- sf::Font & getFont ()

  Getter for the font used in rendering text.
- unsigned int getFontSize () const

  Getter for the font size used in rendering text.
- void setFont (sf::Font font)

  Setter for the font used in rendering text.
- std::vector< std::shared_ptr< Button > > & getButtons ()

  Getter for the vector of buttons used in the withdrawal process.
- std::vector< std::shared_ptr< sf::Text > > & getTexts ()

  Getter for the vector of text objects used in the withdrawal process.
- std::shared_ptr< Button > getPlayer1Button ()

  Getter for button for choosing Player 1 as passive in withdraw.
- std::shared_ptr< Button > getPlayer2Button ()

  Getter for button for choosing Player 2 as passive in withdraw.
- std::shared_ptr< Button > getPlayer3Button ()

  Getter for button for choosing Player 3 as passive in withdraw.
- std::shared_ptr< Button > getPlayer4Button ()

  Getter for button for choosing Player 4 as passive in withdraw.
- std::shared_ptr< Button > getResignButton ()

  Getter for the resign button.
- void createChoosePlayerScreen ()

  Creates the screen for choosing players during withdrawal.
- void createValuePlayerScreen ()

  Creates the screen for specifying values during withdrawal.
- void createDecisionPlayerScreen ()

  Creates the screen for making decisions during withdrawal.
- void setChooseScreenVisible (bool is_visible)

  Sets the visibility of the choose player screen.
- void setValueScreenVisible (bool is_visible)

  Sets the visibility of the value player screen.
- void setDecisionScreenVisible (bool is_visible)

  Sets the visibility of the decision player screen.
- void setPlayer1ToWithdraw (std::shared_ptr< Player > player_ptr)

Sets Active Player for withdrawal.
- void setPlayer2ToWithdraw (std::shared_ptr< Player > player_ptr)

  Sets Passive Player for withdrawal.
- std::shared_ptr< Player > getPlayer1ToWithdraw ()

  Getter for Active Player object.
- std::shared_ptr< Player > getPlayer2ToWithdraw ()

  Getter for Passive Player object.
- std::shared_ptr< Button > getResignValueButton ()

  Getter for the resign button from value screen.
- std::shared_ptr< Button > getSubmitValueButton ()

  Getter for the submit value button from value screen.
- std::shared_ptr< Button > getPlayer1minus1 ()

  Getter for Active Player minus 1 money button.
- std::shared_ptr< Button > getPlayer1minus10 ()

  Getter for Active Player minus 10 money button.
- std::shared_ptr< Button > getPlayer1minus100 ()

  Getter for Active Player minus 100 money button.
- std::shared_ptr< Button > getPlayer1plus1 ()

  Getter for Active Player plus 1 money button.
- std::shared_ptr< Button > getPlayer1plus10 ()

  Getter for Active Player plus 10 money button.
- std::shared_ptr< Button > getPlayer1plus100 ()

  Getter for Active Player plus 100 money button.
- std::shared_ptr< Button > getPlayer2minus1 ()

  Getter for Passive Player minus 1 money button.
- std::shared_ptr< Button > getPlayer2minus10 ()

  Getter for Passive Player minus 10 money button.
- std::shared_ptr< Button > getPlayer2minus100 ()

  Getter for Passive Player minus 100 money button.
- std::shared_ptr< Button > getPlayer2plus1 ()

  Getter for Passive Player plus 1 money button.
- std::shared_ptr< Button > getPlayer2plus10 ()

  Getter for Passive Player plus 10 money button.
- std::shared_ptr< Button > getPlayer2plus100 ()

  Getter for Passive Player plus 100 money button.
- std::shared_ptr< Button > getPlayer1NextButton ()

  Getter for Player 1 next property button.
- std::shared_ptr< Button > getPlayer1PreviousButton ()

  Getter for Player 1 previous property button.
- std::shared_ptr< Button > getPlayer2NextButton ()

  Getter for Player 2 next property button.
- std::shared_ptr< Button > getPlayer2PreviousButton ()

  Getter for Player 2 previous property button.
- std::shared_ptr< Button > getPlayer1IndexNextButton ()

  Getter for Player 1 offer next property button.
- std::shared_ptr< Button > getPlayer1IndexPreviousButton ()

  Getter for Player 1 index previous property button.
- std::shared_ptr< Button > getPlayer2IndexNextButton ()

  Getter for Player 2 index next property button.
- std::shared_ptr< Button > getPlayer2IndexPreviousButton ()

  Getter for Player 2 index previous property button.

- std::shared_ptr< Button > getPlayer1AddButton ()

  Getter for Player 1 add to offer property button.
- std::shared_ptr< Button > getPlayer1RemoveButton ()

  Getter for Player 1 remove from offer property button.
- std::shared_ptr< Button > getPlayer2AddButton ()

  Getter for Player 2 add to offer property button.
- std::shared_ptr< Button > getPlayer2RemoveButton ()

  Getter for Player 2 remove from offer property button.
- std::shared_ptr< Button > getResignDecisionButton ()

  Getter for the resign decision button.
- std::shared_ptr< Button > getAcceptDecisionButton ()

  Getter for the accept decision button.
- void makeWithdraw ()

  Initiates the withdrawal process (exchange of money and properties).
- void moneyTransferIndex (unsigned int player_num, int money)

  Transfers money during the withdrawal process.
- void moneyTextUpdate ()

  Updates the displayed money text during the withdrawal process.
- void showProperty (int column)

  Shows the property details for the specified column during the withdrawal process.
- sf::Sprite & getSpritePropertyPlayer1 ()

  Getter for Player 1's property sprite.
- std::vector< std::shared_ptr< sf::Text > > & getTextsPropertyPlayer1 ()

  Getter for the text objects representing Player 1's property details.
- sf::Sprite & getSpritePropertyPlayer1Index ()

  Getter for Player 1's offered property sprite.
- std::vector< std::shared_ptr< sf::Text > > & getTextsPropertyPlayer1Index ()

  Getter for the text objects representing Player 1's offered property details.
- sf::Sprite & getSpritePropertyPlayer2 ()

  Getter for Player 2's property sprite.
- std::vector< std::shared_ptr< sf::Text > > & getTextsPropertyPlayer2 ()

  Getter for the text objects representing Player 2's property details.
- sf::Sprite & getSpritePropertyPlayer2Index ()

  Getter for Player 2's offered property sprite.
- std::vector< std::shared_ptr< sf::Text > > & getTextsPropertyPlayer2Index ()

  Getter for the text objects representing Player 2's offered property details.
- std::shared_ptr< sf::Texture > getTexturePropertyPlayer1 ()

  Getter for Player 1's property texture.
- std::shared_ptr< sf::Texture > getTexturePropertyPlayer1Index ()

  Getter for Player 1's offer property texture.
- std::shared_ptr< sf::Texture > getTexturePropertyPlayer2 ()

  Getter for Player 2's property texture.
- std::shared_ptr< sf::Texture > getTexturePropertyPlayer2Index ()

  Getter for Player 2's offer property texture.
- void addPropertyPlayerShowed (int i, unsigned int col)

  Changes showed property in certain column.
- void propertyPlayerMoveIndex (int dir, unsigned int plr_num)

  Moves properties between players ownership and withdraw offer.
- bool isNonZeroValue ()

  Check whenever withdraw is legal to be done.

6.36.1 Detailed Description

Represents the trade and withdraw mechanism in a monopoly game.

The Withdraw class handles the trade and withdraw functionality between players in a monopoly game. It includes features such as choosing players, specifying values, and making decisions during the withdrawal process.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 Withdraw()

```
Withdraw::Withdraw ( )
```

Constructor for the Withdraw class.

6.36.3 Member Function Documentation

6.36.3.1 addPropertyPlayerShowed()

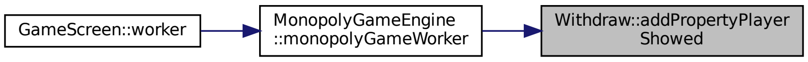
```
void Withdraw::addPropertyPlayerShowed (
             int i,
             unsigned int col )
```

Changes showed property in certain column.

Parameters

i	if i $>$ 0 moves righ in properties vector, if i $<$ 0 moves left
col	(1-4) defines properties column

Here is the caller graph for this function:

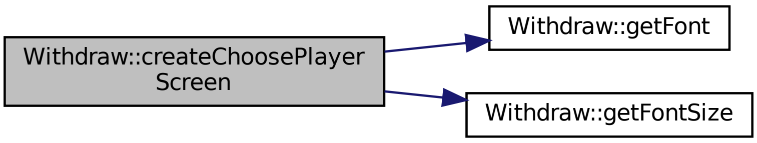

GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::addPropertyPlayer Showed

6.36.3.2 createChoosePlayerScreen()

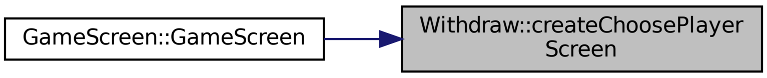
```
void Withdraw::createChoosePlayerScreen ( )
```

Creates the screen for choosing players during withdrawal.

Here is the call graph for this function:



Here is the caller graph for this function:

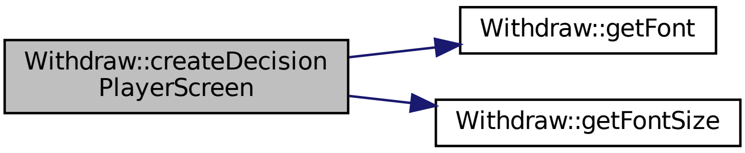


6.36.3.3 createDecisionPlayerScreen()

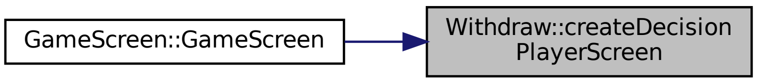
```
void Withdraw::createDecisionPlayerScreen ( )
```

Creates the screen for making decisions during withdrawal.

Here is the call graph for this function:



Here is the caller graph for this function:

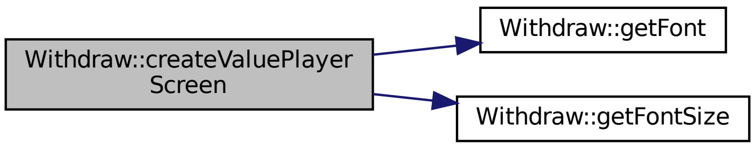


6.36.3.4 createValuePlayerScreen()

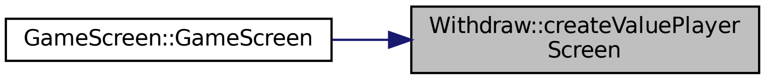
```
void Withdraw::createValuePlayerScreen ( )
```

Creates the screen for specifying values during withdrawal.

Here is the call graph for this function:



Here is the caller graph for this function:



6.36.3.5 getAcceptDecisionButton()

```
std::shared_ptr< Button > Withdraw::getAcceptDecisionButton ( )
```

Getter for the accept decision button.

Returns

A shared pointer to the accept decision Button object.

6.36.3.6 getButtons()

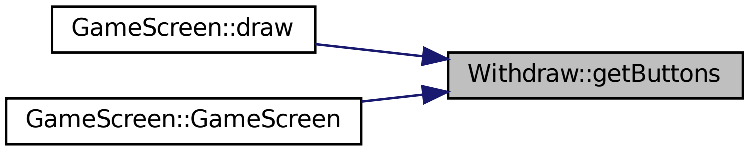
```
std::vector< std::shared_ptr< Button > > & Withdraw::getButtons ( )
```

Getter for the vector of buttons used in the withdrawal process.

Returns

A vector of shared pointers to Button objects.

Here is the caller graph for this function:



6.36.3.7 getFont()

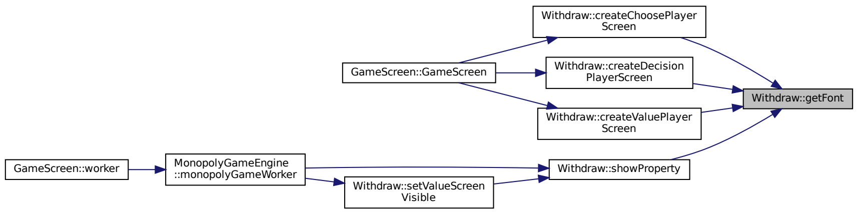
```
sf::Font & Withdraw::getFont ( )
```

Getter for the font used in rendering text.

Returns

A reference to the SFML Font object.

Here is the caller graph for this function:


Withdraw::createChoosePlayer Screen
GameScreen::GameScreen
Withdraw::createDecision PlayerScreen
Withdraw::createValuePlayer Screen
Withdraw::getFont
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::showProperty
Withdraw::setValueScreen Visible

6.36.3.8 getFontSize()

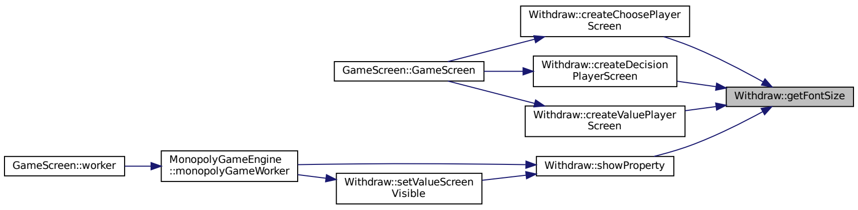
`unsigned int Withdraw::getFontSize ( ) const`

Getter for the font size used in rendering text.

Returns

An unsigned integer representing the font size.

Here is the caller graph for this function:


Withdraw::createChoosePlayer Screen
GameScreen::GameScreen
Withdraw::createDecision PlayerScreen
Withdraw::getFontSize
Withdraw::createValuePlayer Screen
GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::showProperty
Withdraw::setValueScreen Visible

6.36.3.9 getPlayer1AddButton()

`std::shared_ptr< Button > Withdraw::getPlayer1AddButton ( )`

Getter for Player 1 add to offer property button.

Returns

A shared pointer to the add property Button object for Player 1 offer.

6.36.3.10 getPlayer1Button()

`std::shared_ptr< Button > Withdraw::getPlayer1Button ( )`

Getter for button for choosing Player 1 as passive in withdraw.

Returns

A shared pointer to choose Player 1 Button object.

6.36.3.11 getPlayer1IndexNextButton()

```
std::shared_ptr< Button > Withdraw::getPlayer1IndexNextButton ( )
```

Getter for Player 1 offer next property button.

Returns

A shared pointer to the next property Button object for Player 1 offer.

6.36.3.12 getPlayer1IndexPreviousButton()

```
std::shared_ptr< Button > Withdraw::getPlayer1IndexPreviousButton ( )
```

Getter for Player 1 index previous property button.

Returns

A shared pointer to the previous property Button object for Player 1 offer.

6.36.3.13 getPlayer1IndexProperties()

```
std::vector< unsigned int > Withdraw::getPlayer1IndexProperties ( )
```

Getter for the properties of Player 1 offer.

Returns

A vector of unsigned integers representing the indexes of Player 1 offer properties.

6.36.3.14 getPlayer1minus1()

```
std::shared_ptr< Button > Withdraw::getPlayer1minus1 ( )
```

Getter for Active Player minus 1 money button.

Returns

A shared pointer to the minus 1 Button object for Active Player.

6.36.3.15 getPlayer1minus10()

```
std::shared_ptr< Button > Withdraw::getPlayer1minus10 ( )
```

Getter for Active Player minus 10 money button.

Returns

A shared pointer to the minus 10 Button object for Active Player.

6.36.3.16 getPlayer1minus100()

```
std::shared_ptr< Button > Withdraw::getPlayer1minus100 ( )
```

Getter for Active Player minus 100 money button.

Returns

A shared pointer to the minus 100 Button object for Active Player.

6.36.3.17 getPlayer1NextButton()

```
std::shared_ptr< Button > Withdraw::getPlayer1NextButton ( )
```

Getter for Player 1 next property button.

Returns

A shared pointer to the next property Button object for Player 1.

6.36.3.18 getPlayer1plus1()

```
std::shared_ptr< Button > Withdraw::getPlayer1plus1 ( )
```

Getter for Active Player plus 1 money button.

Returns

A shared pointer to the minus 1 Button object for Active Player.

6.36.3.19 getPlayer1plus10()

```
std::shared_ptr< Button > Withdraw::getPlayer1plus10 ( )
```

Getter for Active Player plus 10 money button.

Returns

A shared pointer to the minus 10 Button object for Active Player.

6.36.3.20 getPlayer1plus100()

```
std::shared_ptr< Button > Withdraw::getPlayer1plus100 ( )
```

Getter for Active Player plus 100 money button.

Returns

A shared pointer to the minus 100 Button object for Active Player.

6.36.3.21 getPlayer1PreviousButton()

```
std::shared_ptr< Button > Withdraw::getPlayer1PreviousButton ( )
```

Getter for Player 1 previous property button.

Returns

A shared pointer to the previous property Button object for Player 1.

6.36.3.22 getPlayer1RemoveButton()

```
std::shared_ptr< Button > Withdraw::getPlayer1RemoveButton ( )
```

Getter for Player 1 remove from offer property button.

Returns

A shared pointer to the remove property Button object from Player 1 offer.

6.36.3.23 getPlayer1ToWithdraw()

```
std::shared_ptr< Player > Withdraw::getPlayer1ToWithdraw ( )
```

Getter for Active Player object.

Returns

A shared pointer to the Player object.

6.36.3.24 getPlayer2AddButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2AddButton ( )
```

Getter for Player 2 add to offer property button.

Returns

A shared pointer to the add property Button object for Player 2 offer.

6.36.3.25 getPlayer2Button()

```
std::shared_ptr< Button > Withdraw::getPlayer2Button ( )
```

Getter for button for choosing Player 2 as passive in withdraw.

Returns

A shared pointer to choose Player 2 Button object.

6.36.3.26 getPlayer2IndexNextButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2IndexNextButton ( )
```

Getter for Player 2 index next property button.

Returns

A shared pointer to the next property Button object for Player 2 offer.

6.36.3.27 getPlayer2IndexPreviousButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2IndexPreviousButton ( )
```

Getter for Player 2 index previous property button.

Returns

A shared pointer to the previous property Button object for Player 2 offer.

6.36.3.28 getPlayer2IndexProperties()

```
std::vector< unsigned int > Withdraw::getPlayer2IndexProperties ( )
```

Getter for the properties of Player 2 offer.

Returns

A vector of unsigned integers representing the indexes of Player 2 offer properties.

6.36.3.29 getPlayer2minus1()

```
std::shared_ptr< Button > Withdraw::getPlayer2minus1 ( )
```

Getter for Passive Player minus 1 money button.

Returns

A shared pointer to the minus 1 Button object for Passive Player.

6.36.3.30 getPlayer2minus10()

```
std::shared_ptr< Button > Withdraw::getPlayer2minus10 ( )
```

Getter for Passive Player minus 10 money button.

Returns

A shared pointer to the minus 10 Button object for Passive Player.

6.36.3.31 getPlayer2minus100()

```
std::shared_ptr< Button > Withdraw::getPlayer2minus100 ( )
```

Getter for Passive Player minus 100 money button.

Returns

A shared pointer to the minus 100 Button object for Passive Player.

6.36.3.32 getPlayer2NextButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2NextButton ( )
```

Getter for Player 2 next property button.

Returns

A shared pointer to the next property Button object for Player 2.

6.36.3.33 getPlayer2plus1()

```
std::shared_ptr< Button > Withdraw::getPlayer2plus1 ( )
```

Getter for Passive Player plus 1 money button.

Returns

A shared pointer to the plus 1 Button object for Passive Player.

6.36.3.34 getPlayer2plus10()

```
std::shared_ptr< Button > Withdraw::getPlayer2plus10 ( )
```

Getter for Passive Player plus 10 money button.

Returns

A shared pointer to the plus 10 Button object for Passive Player.

6.36.3.35 getPlayer2plus100()

```
std::shared_ptr< Button > Withdraw::getPlayer2plus100 ( )
```

Getter for Passive Player plus 100 money button.

Returns

A shared pointer to the plus 100 Button object for Passive Player.

6.36.3.36 getPlayer2PreviousButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2PreviousButton ( )
```

Getter for Player 2 previous property button.

Returns

A shared pointer to the previous property Button object for Player 2 .

6.36.3.37 getPlayer2RemoveButton()

```
std::shared_ptr< Button > Withdraw::getPlayer2RemoveButton ( )
```

Getter for Player 2 remove from offer property button.

Returns

A shared pointer to the remove property Button object from Player 2 offer.

6.36.3.38 getPlayer2ToWithdraw()

```
std::shared_ptr< Player > Withdraw::getPlayer2ToWithdraw ( )
```

Getter for Passive Player object.

Returns

A shared pointer to the Player object.

6.36.3.39 getPlayer3Button()

```
std::shared_ptr< Button > Withdraw::getPlayer3Button ( )
```

Getter for button for choosing Player 3 as passive in withdraw.

Returns

A shared pointer to choose Player 3 Button object.

6.36.3.40 getPlayer4Button()

```
std::shared_ptr< Button > Withdraw::getPlayer4Button ( )
```

Getter for button for choosing Player 4 as passive in withdraw.

Returns

A shared pointer to choose Player 4 Button object.

6.36.3.41 getResignButton()

```
std::shared_ptr< Button > Withdraw::getResignButton ( )
```

Getter for the resign button.

Returns

A shared pointer to the resign Button object.

6.36.3.42 getResignDecisionButton()

```
std::shared_ptr< Button > Withdraw::getResignDecisionButton ( )
```

Getter for the resign decision button.

Returns

A shared pointer to the resign decision Button object.

6.36.3.43 getResignValueButton()

`std::shared_ptr< Button > Withdraw::getResignValueButton ( )`

Getter for the resign button from value screen.

Returns

A shared pointer to the resign value Button object.

6.36.3.44 getSpritePropertyPlayer1()

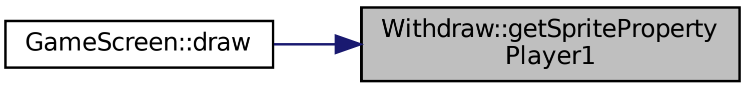
`sf::Sprite & Withdraw::getSpritePropertyPlayer1 ( )`

Getter for Player 1's property sprite.

Returns

A reference to the SFML Sprite object representing Player 1's properties.

Here is the caller graph for this function:



6.36.3.45 getSpritePropertyPlayer1Index()

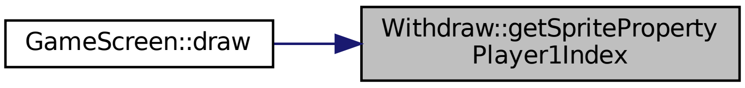
`sf::Sprite & Withdraw::getSpritePropertyPlayer1Index ( )`

Getter for Player 1's offered property sprite.

Returns

A reference to the SFML Sprite object representing Player 1's offered properties.

Here is the caller graph for this function:



6.36.3.46 getSpritePropertyPlayer2()

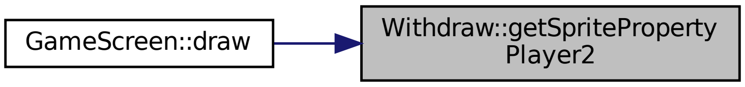
```
sf::Sprite & Withdraw::getSpritePropertyPlayer2 ( )
```

Getter for Player 2's property sprite.

Returns

A reference to the SFML Sprite object representing Player 2's properties.

Here is the caller graph for this function:



6.36.3.47 getSpritePropertyPlayer2Index()

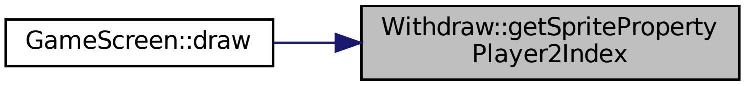
```
sf::Sprite & Withdraw::getSpritePropertyPlayer2Index ( )
```

Getter for Player 2's offered property sprite.

Returns

A reference to the SFML Sprite object representing Player 2's offered properties.

Here is the caller graph for this function:



6.36.3.48 getSubmitValueButton()

```
std::shared_ptr< Button > Withdraw::getSubmitValueButton ( )
```

Getter for the submit value button from value screen.

Returns

A shared pointer to the submit value Button object.

6.36.3.49 getTexts()

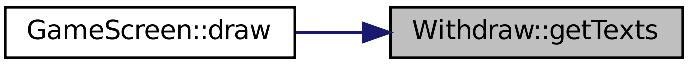
```
std::vector< std::shared_ptr< sf::Text > > & Withdraw::getTexts ( )
```

Getter for the vector of text objects used in the withdrawal process.

Returns

A vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.36.3.50 getTextsPropertyPlayer1()

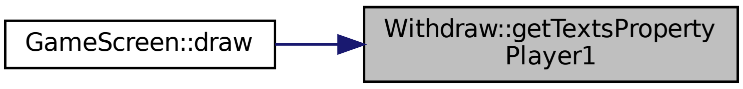
```
std::vector< std::shared_ptr< sf::Text > > & Withdraw::getTextsPropertyPlayer1 ( )
```

Getter for the text objects representing Player 1's property details.

Returns

A vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.36.3.51 getTextsPropertyPlayer1Index()

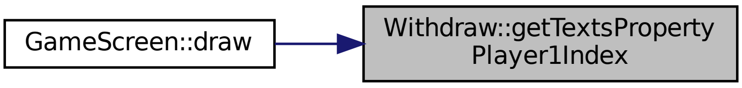
```
std::vector< std::shared_ptr< sf::Text > > & Withdraw::getTextsPropertyPlayer1Index ( )
```

Getter for the text objects representing Player 1's offered property details.

Returns

A vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.36.3.52 getTextsPropertyPlayer2()

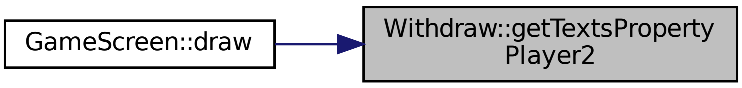
```
std::vector< std::shared_ptr< sf::Text > > & Withdraw::getTextsPropertyPlayer2 ( )
```

Getter for the text objects representing Player 2's property details.

Returns

A vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.36.3.53 getTextsPropertyPlayer2Index()

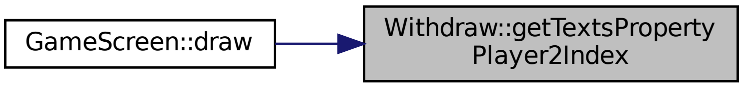
```
std::vector< std::shared_ptr< sf::Text > > & Withdraw::getTextsPropertyPlayer2Index ( )
```

Getter for the text objects representing Player 2's offered property details.

Returns

A vector of shared pointers to SFML Text objects.

Here is the caller graph for this function:



6.36.3.54 getTexturePropertyPlayer1()

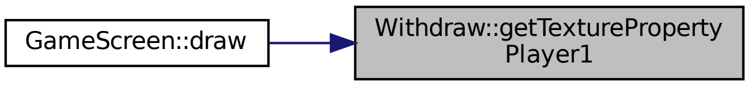
```
std::shared_ptr< sf::Texture > Withdraw::getTexturePropertyPlayer1 ( )
```

Getter for Player 1's property texture.

Returns

SFML Texrute object representing Player 1's properties.

Here is the caller graph for this function:



6.36.3.55 getTexturePropertyPlayer1Index()

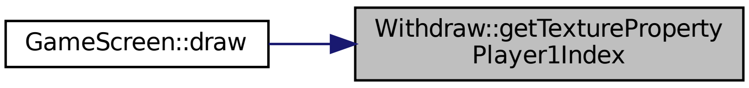
```
std::shared_ptr< sf::Texture > Withdraw::getTexturePropertyPlayer1Index ( )
```

Getter for Player 1's offer property texture.

Returns

SFML Texrute object representing Player 1's properties offer.

Here is the caller graph for this function:



6.36.3.56 getTexturePropertyPlayer2()

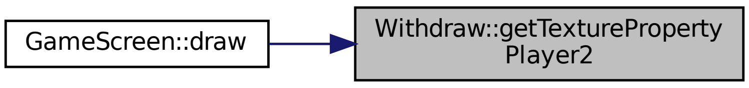
```
std::shared_ptr< sf::Texture > Withdraw::getTexturePropertyPlayer2 ( )
```

Getter for Player 2's property texture.

Returns

SFML Texrute object representing Player 2's properties.

Here is the caller graph for this function:



6.36.3.57 getTexturePropertyPlayer2Index()

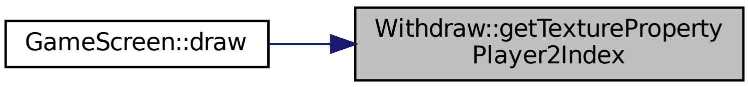
```
std::shared_ptr< sf::Texture > Withdraw::getTexturePropertyPlayer2Index ( )
```

Getter for Player 2's offer property texture.

Returns

SFML Texrute object representing Player 2's properties offer.

Here is the caller graph for this function:



6.36.3.58 getTurnState()

```
TurnState Withdraw::getTurnState ( )
```

Getter for the turn state.

Returns

The current TurnState enumeration representing the turn state of monopolygame when withdraw was started.

6.36.3.59 isNonZeroValue()

```
bool Withdraw::isNonZeroValue ( )
```

Check whenever withdraw is legal to be done.

Returns

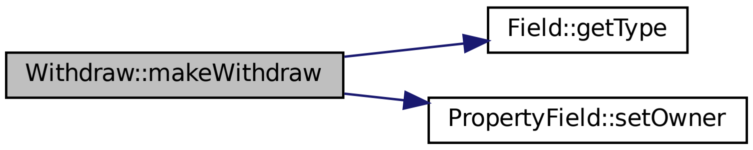
true if withdraw is legal, otherwise false

6.36.3.60 makeWithdraw()

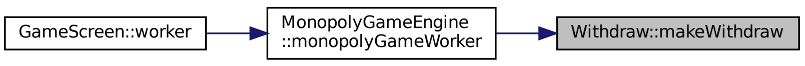
```
void Withdraw::makeWithdraw ( )
```

Initiates the withdrawal process (exchange of money and properties).

Here is the call graph for this function:



Here is the caller graph for this function:

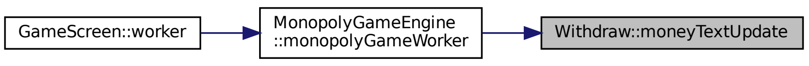


6.36.3.61 moneyTextUpdate()

```
void Withdraw::moneyTextUpdate ( )
```

Updates the displayed money text during the withdrawal process.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::moneyTextUpdate

6.36.3.62 moneyTransferIndex()

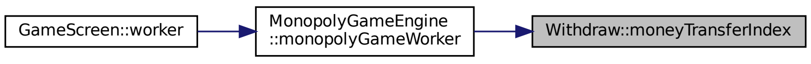
```
void Withdraw::moneyTransferIndex (
              unsigned int player_num,
              int money )
```

Transfers money during the withdrawal process.

Parameters

player_num	An unsigned integer representing the player number (1-2).
money	An integer representing the amount of money to transfer (can be negative).

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::moneyTransferIndex

6.36.3.63 propertyPlayerMoveIndex()

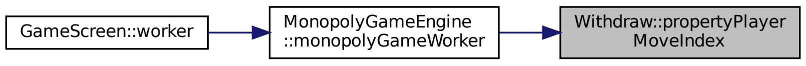
```
void Withdraw::propertyPlayerMoveIndex (
            int dir,
            unsigned int plr_num )
```

Moves properties between players ownership and withdraw offer.

Parameters

dir	if $>$ 0 moves to offer vector, if $<$ 0 moves from offer vectpr
plr_num	if 1 - active player, if 2 - passive player, in withdraw

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::propertyPlayer MoveIndex

6.36.3.64 setBoard()

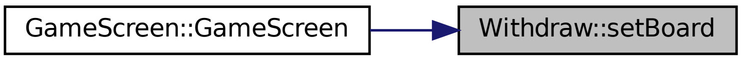
```
void Withdraw::setBoard (
            std::shared_ptr< Board > board_ptr )
```

Setter for the gameboard.

Parameters

board_ptr	A shared pointer to the Board object.

Here is the caller graph for this function:



6.36.3.65 setChooseScreenVisible()

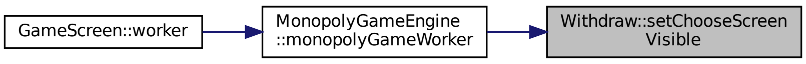
```
void Withdraw::setChooseScreenVisible (
             bool is_visible )
```

Sets the visibility of the choose player screen.

Parameters

is_visible	A boolean indicating whether the screen should be visible.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setChooseScreen Visible

6.36.3.66 setDecisionScreenVisible()

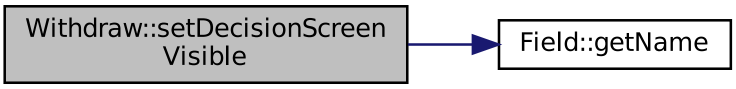
```
void Withdraw::setDecisionScreenVisible (
             bool is_visible )
```

Sets the visibility of the decision player screen.

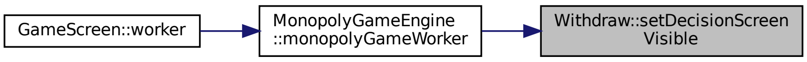
Parameters

is_visible	A boolean indicating whether the screen should be visible.

Here is the call graph for this function:



Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setDecisionScreen Visible

6.36.3.67 setFont()

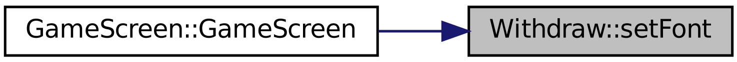
```
void Withdraw::setFont (
             sf::Font font )
```

Setter for the font used in rendering text.

Parameters

font	A reference to the SFML Font object.

Here is the caller graph for this function:



6.36.3.68 setPlayer1IndexProperties()

```
void Withdraw::setPlayer1IndexProperties (
             std::vector< unsigned int > new_index_properties )
```

Setter for the properties of Player 1 offer.

Parameters

new_index_properties	A vector of unsigned integers representing the new index for Player 1 offer properties.

6.36.3.69 setPlayer1ToWithdraw()

```
void Withdraw::setPlayer1ToWithdraw (
             std::shared_ptr< Player > player_ptr )
```

Sets Active Player for withdrawal.

Parameters

player_ptr	A shared pointer to the Player object.

6.36.3.70 setPlayer2IndexProperties()

```
void Withdraw::setPlayer2IndexProperties (
             std::vector< unsigned int > new_index_properties )
```

Setter for the properties of Player 2 offer.

Parameters

new_index_properties	A vector of unsigned integers representing the new index for Player 2 offer properties.

6.36.3.71 setPlayer2ToWithdraw()

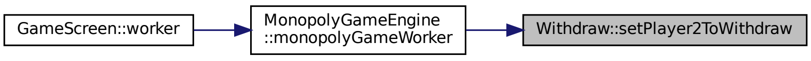
```
void Withdraw::setPlayer2ToWithdraw (
             std::shared_ptr< Player > player_ptr )
```

Sets Passive Player for withdrawal.

Parameters

player_ptr	A shared pointer to the Player object.

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setPlayer2ToWithdraw

6.36.3.72 setTurnState()

```
void Withdraw::setTurnState (
             TurnState state )
```

Setter for the turn state of monopolygame when withdraw was started.

Parameters

state	A TurnState enumeration representing the turn state.

6.36.3.73 setValueScreenVisible()

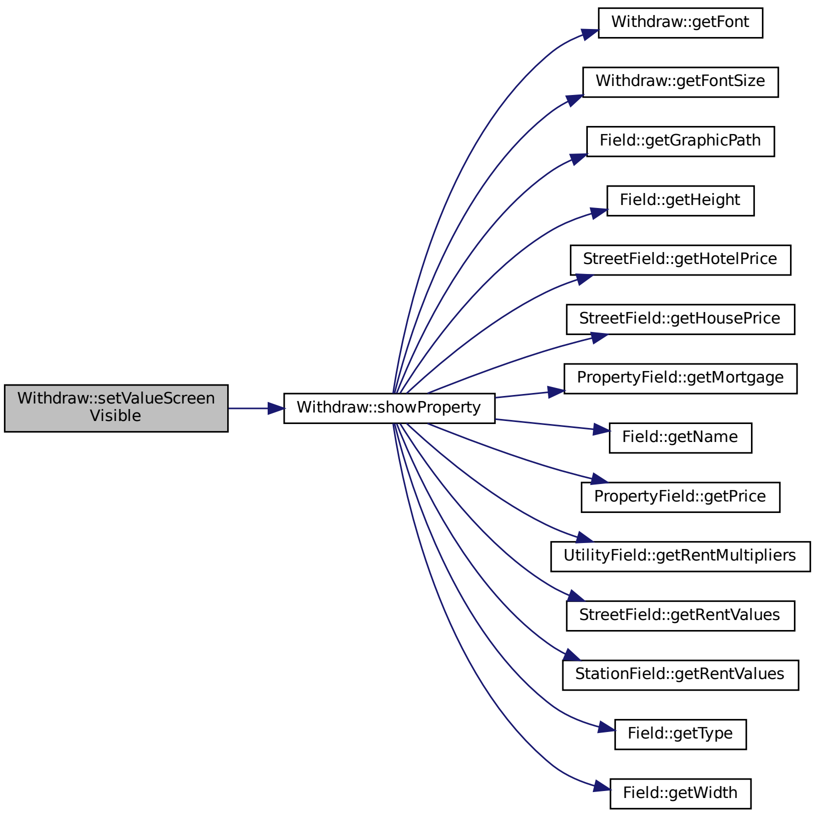
```
void Withdraw::setValueScreenVisible (
             bool is_visible )
```

Sets the visibility of the value player screen.

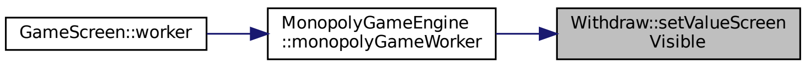
Parameters

is_visible	A boolean indicating whether the screen should be visible.

Here is the call graph for this function:


Withdraw::setValueScreen Visible
Withdraw::showProperty
Withdraw::getFont
Withdraw::getFontSize
Field::getGraphicPath
Field::getHeight
StreetField::getHotelPrice
StreetField::getHousePrice
PropertyField::getMortgage
Field::getName
PropertyField::getPrice
UtilityField::getRentMultipliers
StreetField::getRentValues
StationField::getRentValues
Field::getType
Field::getWidth

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setValueScreen Visible

6.36.3.74 showProperty()

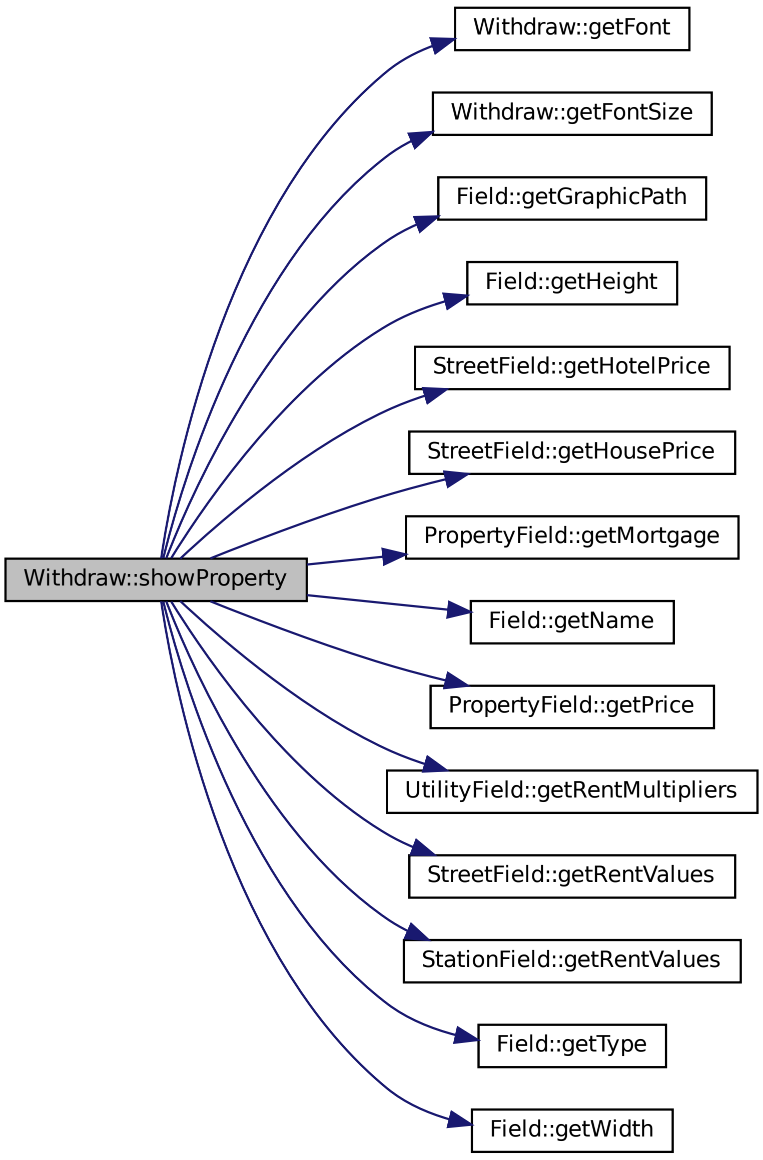
```
void Withdraw::showProperty (
             int column )
```

Shows the property details for the specified column during the withdrawal process.

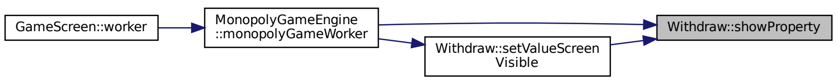
Parameters

column	An integer representing the column index (1-4).

Here is the call graph for this function:


Withdraw::showProperty
Withdraw::getFont
Withdraw::getFontSize
Field::getGraphicPath
Field::getHeight
StreetField::getHotelPrice
StreetField::getHousePrice
PropertyField::getMortgage
Field::getName
PropertyField::getPrice
UtilityField::getRentMultipliers
StreetField::getRentValues
StationField::getRentValues
Field::getType
Field::getWidth

Here is the caller graph for this function:


GameScreen::worker
MonopolyGameEngine ::monopolyGameWorker
Withdraw::setValueScreen Visible
Withdraw::showProperty

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

- /home/kamil/zpr/Monopoly/Withdraw.h
- /home/kamil/zpr/Monopoly/Withdraw.cc

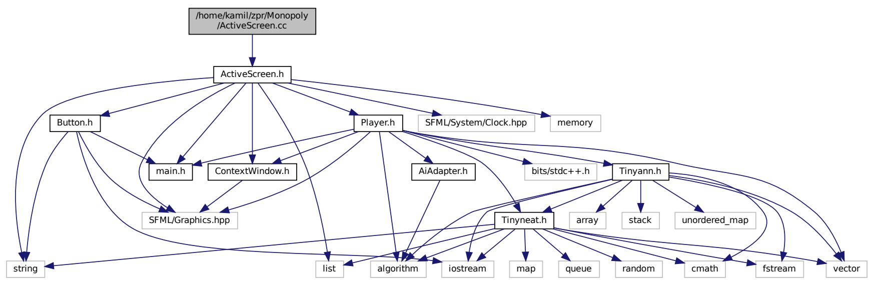
Chapter 7

File Documentation

7.1 /home/kamil/zpr/Monopoly/ActiveScreen.cc File Reference

Source file handling displayed screens of project Base claass is ActiveScreen, then derived class are used to work with specific screen shown.

`#include "ActiveScreen.h"`
Include dependency graph for ActiveScreen.cc:



7.1.1 Detailed Description

Source file handling displayed screens of project Base claass is ActiveScreen, then derived class are used to work with specific screen shown.

Author

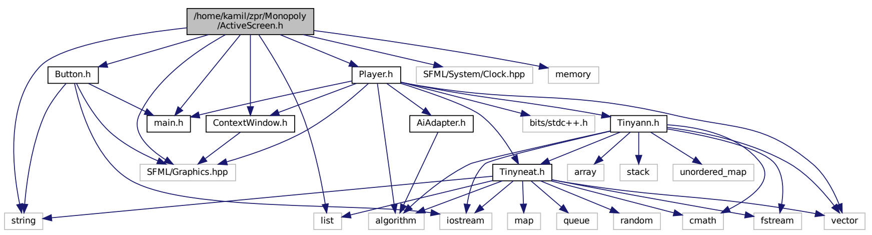
Kamil Kosnik, Kacper Radzikowski

7.2 /home/kamil/zpr/Monopoly/ActiveScreen.h File Reference

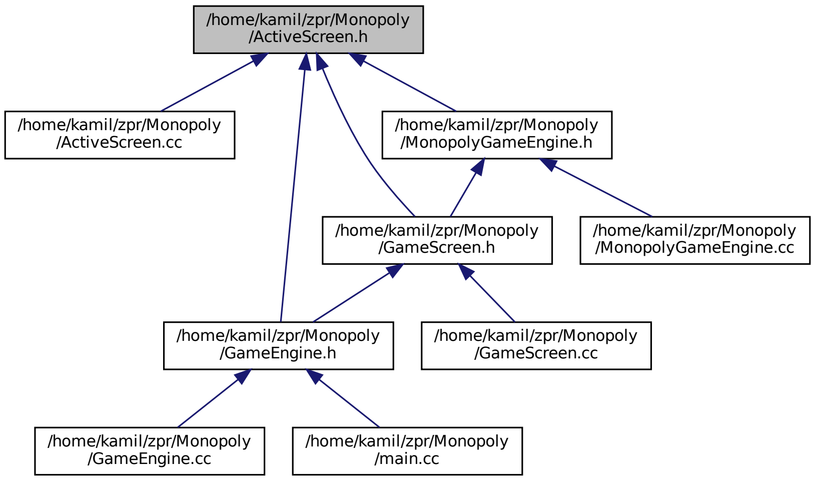
Header file handling displayed screens of project Base claass is ActiveScreen, then derived class are used to work with specific screen shown.

```
#include <SFML/Graphics.hpp>
#include <SFML/System/Clock.hpp>
#include <list>
#include <memory>
#include <string>
#include "Button.h"
#include "ContextWindow.h"
#include "Player.h"
#include "main.h"
```

Include dependency graph for ActiveScreen.h:



This graph shows which files directly or indirectly include this file:



/home/kamil/zpr/Monopoly /ActiveScreen.h
/home/kamil/zpr/Monopoly /ActiveScreen.cc
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class ActiveScreen

  Represents the base class for handling displayed screens in the project.
- class GameMenuScreen

  Represents the screen for the game menu.
- class MainMenuScreen

  Represents the screen for the main menu.

7.2.1 Detailed Description

Header file handling displayed screens of project Base claass is ActiveScreen, then derived class are used to work with specific screen shown.

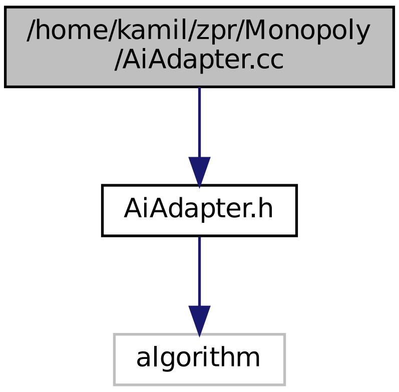
Author

Kamil Kosnik, Kacper Radzikowski

7.3 /home/kamil/zpr/Monopoly/AiAdapter.cc File Reference

TODO.

```
#include "AiAdapter.h"
```
Include dependency graph for AiAdapter.cc:



7.3.1 Detailed Description

TODO.

Author

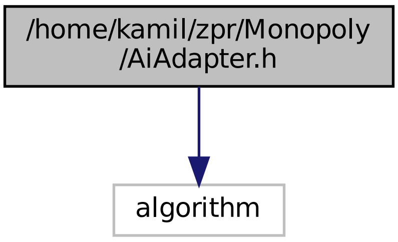
Kamil Kosnik, Kacper Radzikowski

7.4 /home/kamil/zpr/Monopoly/AiAdapter.h File Reference

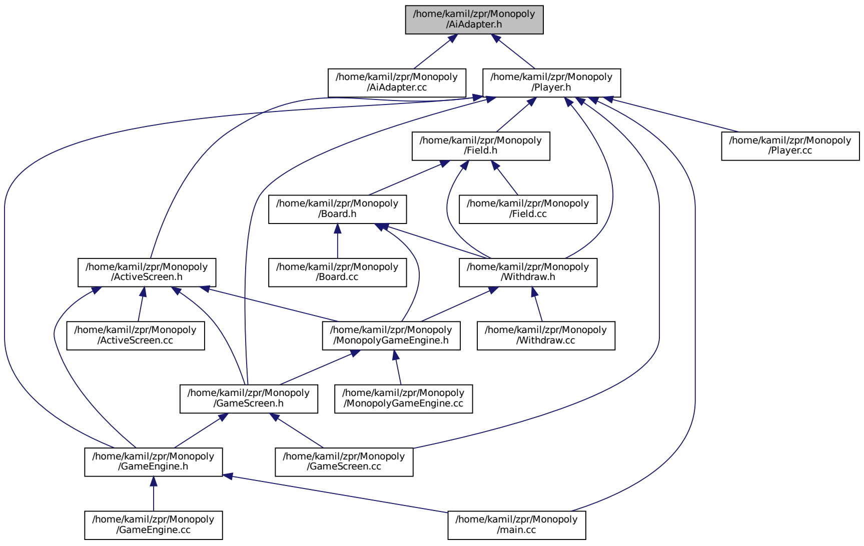
TODO.

`#include <algorithm>`

Include dependency graph for AiAdapter.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/AiAdapter.h
/home/kamil/zpr/Monopoly/AiAdapter.cc
/home/kamil/zpr/Monopoly/Player.h
/home/kamil/zpr/Monopoly/Field.h
/home/kamil/zpr/Monopoly/Player.cc
/home/kamil/zpr/Monopoly/Board.h
/home/kamil/zpr/Monopoly/Field.cc
/home/kamil/zpr/Monopoly/ActiveScreen.h
/home/kamil/zpr/Monopoly/Board.cc
/home/kamil/zpr/Monopoly/Withdraw.h
/home/kamil/zpr/Monopoly/ActiveScreen.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.h
/home/kamil/zpr/Monopoly/Withdraw.cc
/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.cc
/home/kamil/zpr/Monopoly/main.cc

Classes

- class AiAdapter

  Represents an adapter for AI input data.

7.4.1 Detailed Description

TODO.

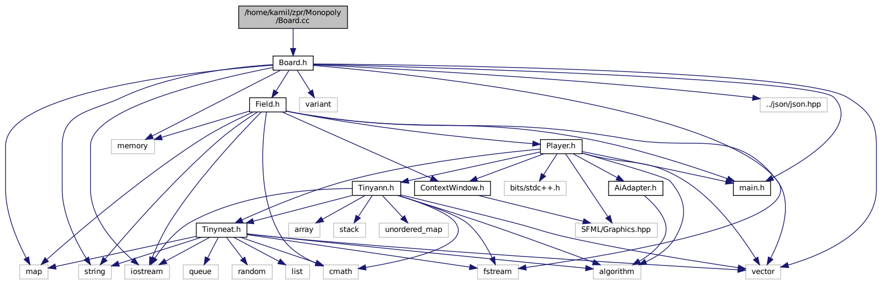
Author

Kamil Kosnik, Kacper Radzikowski

7.5 /home/kamil/zpr/Monopoly/Board.cc File Reference

Source file for monopoly game board, creation is based on json file 'board.json'. Mainly handles all types of fields and their usage.

```
#include "Board.h"
```
Include dependency graph for Board.cc:



Typedefs

- using json = nlohmann::json

Functions

- std::map< StreetTiers, unsigned int > jsonToStreetRent (const json &element)

  Converts JSON data to a map of rent values for street properties.
- std::map< StationTiers, unsigned int > jsonToStationRent (const json &element)

  Converts JSON data to a map of rent values for station properties.
- std::map< UtilityTiers, unsigned int > jsonToUtilityRent (const json &element)

  Converts JSON data to a map of rent multipliers for utility properties.

7.5.1 Detailed Description

Source file for monopoly game board, creation is based on json file 'board.json'. Mainly handles all types of fields and their usage.

Author

Kamil Kosnik, Kacper Radzikowski

7.5.2 Typedef Documentation

7.5.2.1 json

```
using json = nlohmann::json
```

7.5.3 Function Documentation

7.5.3.1 jsonToStationRent()

```
std::map<StationTiers, unsigned int> jsonToStationRent (
            const json & element )
```

Converts JSON data to a map of rent values for station properties.

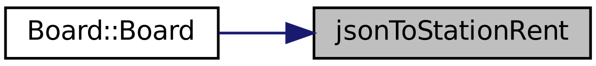
Parameters

element	The JSON data containing rent values.

Returns

A map of rent values for station properties.

Here is the caller graph for this function:



7.5.3.2 jsonToStreetRent()

```
std::map<StreetTiers, unsigned int> jsonToStreetRent (
            const json & element )
```

Converts JSON data to a map of rent values for street properties.

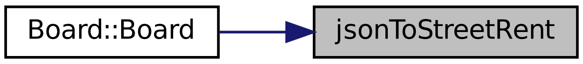
Parameters

element	The JSON data containing rent values.

Returns

A map of rent values for street properties.

Here is the caller graph for this function:



7.5.3.3 jsonToUtilityRent()

```
std::map<UtilityTiers, unsigned int> jsonToUtilityRent (
            const json & element )
```

Converts JSON data to a map of rent multipliers for utility properties.

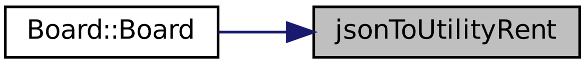
Parameters

element	The JSON data containing rent multipliers.

Returns

A map of rent multipliers for utility properties.

Here is the caller graph for this function:

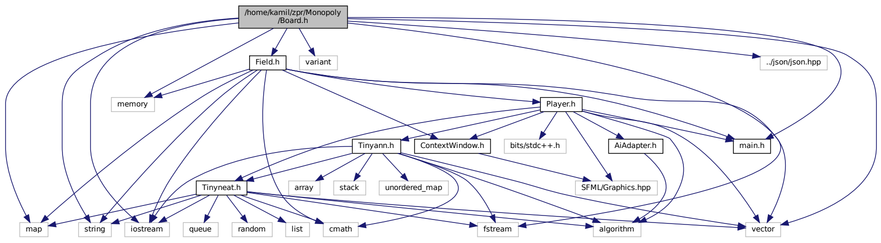


7.6 /home/kamil/zpr/Monopoly/Board.h File Reference

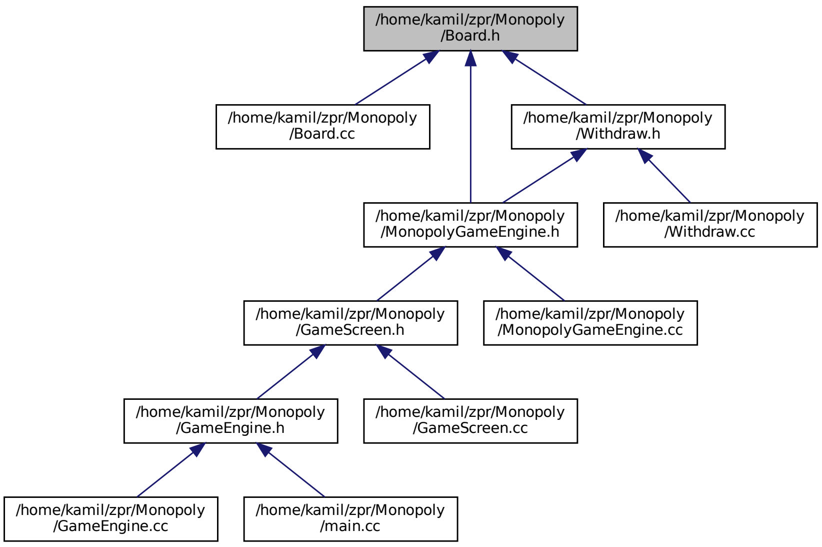
Header file for monopoly game board, creation is based on json file 'board.json'. Mainly handles all types of fields and their usage.

```
#include <fstream>
#include <iostream>
#include <map>
#include <memory>
#include <string>
#include <variant>
#include <vector>
#include "../json/json.hpp"
#include "Field.h"
#include "main.h"
```

Include dependency graph for Board.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly /Board.h
/home/kamil/zpr/Monopoly /Board.cc
/home/kamil/zpr/Monopoly /Withdraw.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /Withdraw.cc
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class Board

    Class representing the monopoly game board.

Typedefs

- using json = nlohmann::json
- using PossibleFields = std::variant< Field, PropertyField, StreetField, StationField, UtilityField, TaxField >

    A variant type representing different types of fields on the monopoly board.

Functions

- std::map< StreetTiers, unsigned int > jsonToStreetRent (const json &element)

  Converts JSON data to a map of rent values for street properties.
- std::map< StationTiers, unsigned int > jsonToStationRent (const json &element)

  Converts JSON data to a map of rent values for station properties.
- std::map< UtilityTiers, unsigned int > jsonToUtilityRent (const json &element)

  Converts JSON data to a map of rent multipliers for utility properties.

7.6.1 Detailed Description

Header file for monopoly game board, creation is based on json file 'board.json'. Mainly handles all types of fields and their usage.

Author

Kamil Kosnik, Kacper Radzikowski

7.6.2 Typedef Documentation

7.6.2.1 json

```
using json = nlohmann::json
```

7.6.2.2 PossibleFields

```
PossibleFields
```

A variant type representing different types of fields on the monopoly board.

7.6.3 Function Documentation

7.6.3.1 jsonToStationRent()

```
std::map<StationTiers, unsigned int> jsonToStationRent (
            const json & element )
```

Converts JSON data to a map of rent values for station properties.

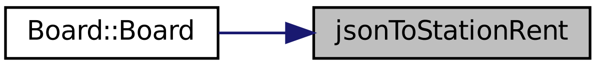
Parameters

element	The JSON data containing rent values.

Returns

A map of rent values for station properties.

Here is the caller graph for this function:



7.6.3.2 jsonToStreetRent()

```
std::map<StreetTiers, unsigned int> jsonToStreetRent (
            const json & element )
```

Converts JSON data to a map of rent values for street properties.

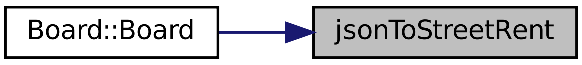
Parameters

element	The JSON data containing rent values.

Returns

A map of rent values for street properties.

Here is the caller graph for this function:



7.6.3.3 jsonToUtilityRent()

```
std::map<UtilityTiers, unsigned int> jsonToUtilityRent (
            const json & element )
```

Converts JSON data to a map of rent multipliers for utility properties.

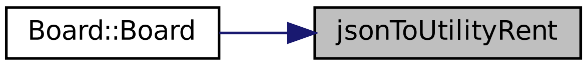
Parameters

element	The JSON data containing rent multipliers.

Returns

A map of rent multipliers for utility properties.

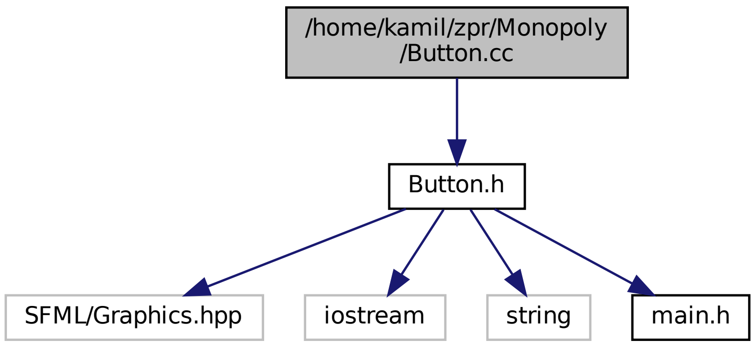
Here is the caller graph for this function:



7.7 /home/kamil/zpr/Monopoly/Button.cc File Reference

```
#include "Button.h"
```

Include dependency graph for Button.cc:

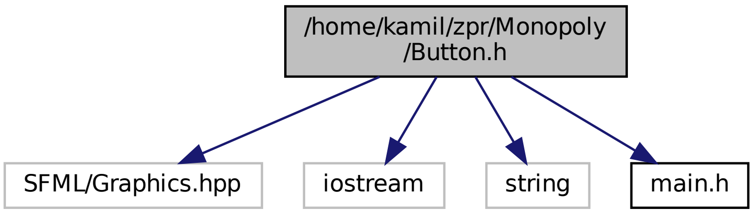


7.8 /home/kamil/zpr/Monopoly/Button.h File Reference

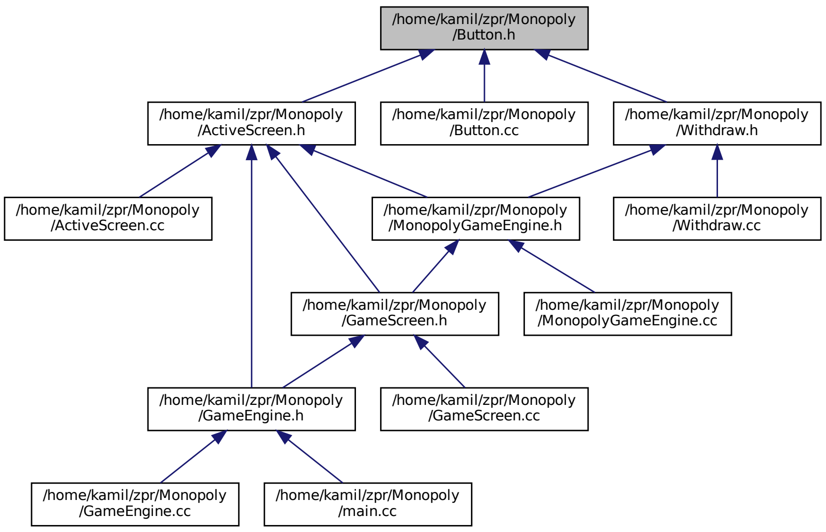
Source file for handling button objects actions used to communicate with user.

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <string>
#include "main.h"
```

Include dependency graph for Button.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly /Button.h
/home/kamil/zpr/Monopoly /ActiveScreen.h
/home/kamil/zpr/Monopoly /Button.cc
/home/kamil/zpr/Monopoly /Withdraw.h
/home/kamil/zpr/Monopoly /ActiveScreen.cc
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /Withdraw.cc
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class Button

    Represents a button for handling user interactions.

7.8.1 Detailed Description

Source file for handling button objects actions used to communicate with user.

Header file for handling button objects actions used to communicate with user.

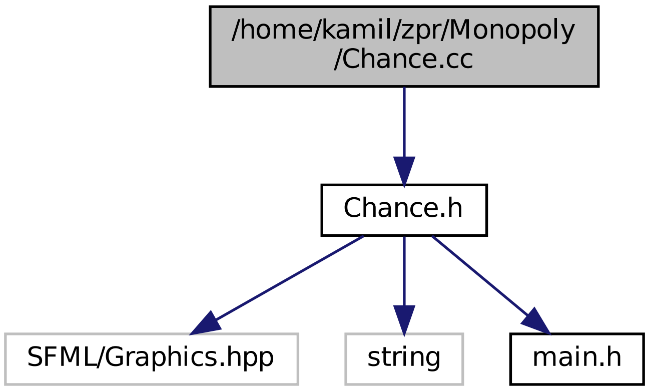
Author

Kamil Kosnik, Kacper Radzikowski

7.9 /home/kamil/zpr/Monopoly/Chance.cc File Reference

Source file for monopoly game chance cards, their types, actions...

`#include "Chance.h"`
Include dependency graph for Chance.cc:



7.9.1 Detailed Description

Source file for monopoly game chance cards, their types, actions...

Author

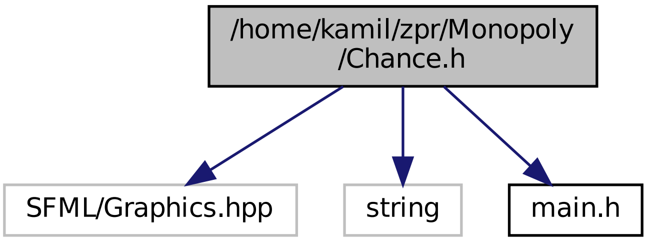
Kamil Kosnik, Kacper Radzikowski

7.10 /home/kamil/zpr/Monopoly/Chance.h File Reference

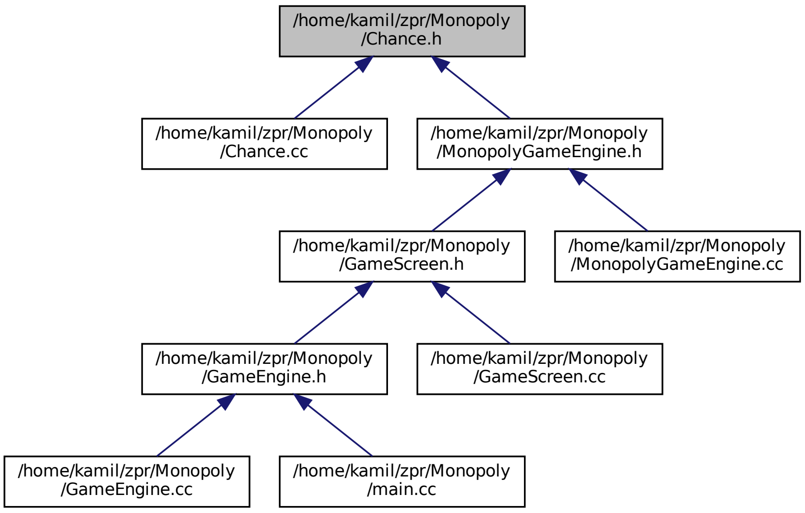
Header file for monopoly game chance cards, their types, actions...

```
#include <SFML/Graphics.hpp>
#include <string>
#include "main.h"
```

Include dependency graph for Chance.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ChanceCard

  Represents a chance card in the monopoly game.

Enumerations

- enum ChanceType {
  MOVEMENT_TO_PROPERTY , MOVEMENT_WITH_BUY_OR_PAY , BANK_PAYS_YOU , GET_OUT_OF_JAIL_CARD
  ,
  MOVEMENT_SPACES , GO_TO_JAIL_CARD , PAY_FOR_HOUSE_HOTEL , TAX_CARD ,
  PAY_PLAYERS }

  Enumeration representing different types of chance cards.

7.10.1 Detailed Description

Header file for monopoly game chance cards, their types, actions...

Author

Kamil Kosnik, Kacper Radzikowski

7.10.2 Enumeration Type Documentation

7.10.2.1 ChanceType

```
enum ChanceType
```

Enumeration representing different types of chance cards.

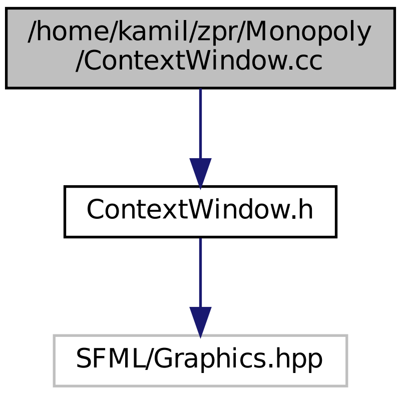
Enumerator

MOVEMENT_TO_PROPERTY	Move to a specific property.
MOVEMENT_WITH_BUY_OR_PAY	Move with the option to buy or pay.
BANK_PAYS_YOU	Receive money from the bank.
GET_OUT_OF_JAIL_CARD	Receive a "Get Out of Jail Free" card.
MOVEMENT_SPACES	Move a certain number of spaces.
GO_TO_JAIL_CARD	Go directly to jail.
PAY_FOR_HOUSE_HOTEL	Pay for each house and hotel owned.
TAX_CARD	Pay a tax.
PAY_PLAYERS	Pay other players.

7.11 /home/kamil/zpr/Monopoly/ContextWindow.cc File Reference

Source file for context Window class It is Singleton class type used mainly for handling SFML window operations between other classes.

`#include "ContextWindow.h"`
Include dependency graph for ContextWindow.cc:



7.11.1 Detailed Description

Source file for context Window class It is Singleton class type used mainly for handling SFML window operations between other classes.

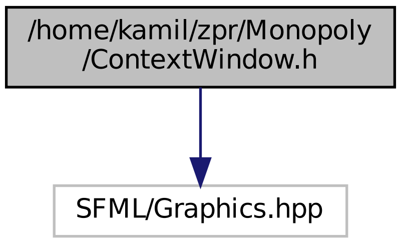
Author

Kamil Kosnik, Kacper Radzikowski

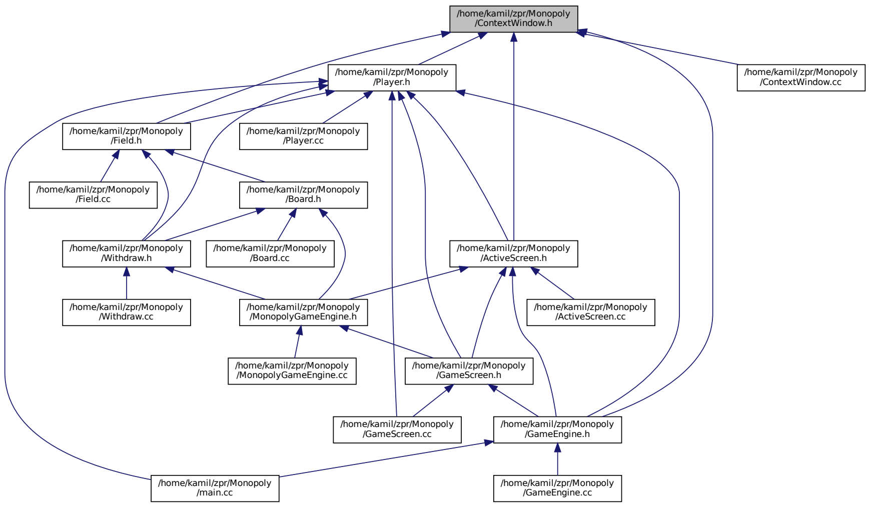
7.12 /home/kamil/zpr/Monopoly/ContextWindow.h File Reference

Source file for context Window class It is Singleton class type used mainly for handling SFML window operations between other classes.

`#include <SFML/Graphics.hpp>`
Include dependency graph for ContextWindow.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/ContextWindow.h
/home/kamil/zpr/Monopoly/Player.h
/home/kamil/zpr/Monopoly/ContextWindow.cc
/home/kamil/zpr/Monopoly/Field.h
/home/kamil/zpr/Monopoly/Player.cc
/home/kamil/zpr/Monopoly/Field.cc
/home/kamil/zpr/Monopoly/Board.h
/home/kamil/zpr/Monopoly/Withdraw.h
/home/kamil/zpr/Monopoly/Board.cc
/home/kamil/zpr/Monopoly/ActiveScreen.h
/home/kamil/zpr/Monopoly/Withdraw.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.h
/home/kamil/zpr/Monopoly/ActiveScreen.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/main.cc
/home/kamil/zpr/Monopoly/GameEngine.cc

Classes

- class ContextWindow

  Represents a Singleton class for handling SFML window operations.

7.12.1 Detailed Description

Source file for context Window class It is Singleton class type used mainly for handling SFML window operations between other classes.

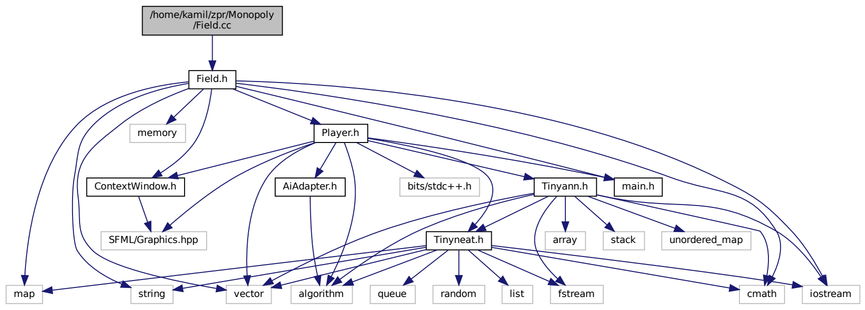
Author

Kamil Kosnik, Kacper Radzikowski

7.13 /home/kamil/zpr/Monopoly/Field.cc File Reference

Source file containing staff for single monopoly board game field. Separated to many deriving classes types each for specific field type.

```
#include "Field.h"
```
Include dependency graph for Field.cc:



7.13.1 Detailed Description

Source file containing staff for single monopoly board game field. Separated to many deriving classes types each for specific field type.

Author

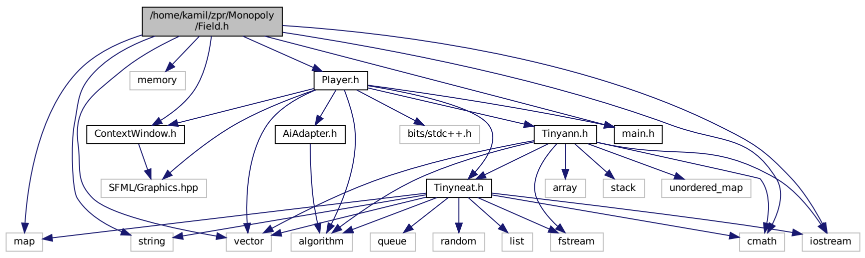
Kamil Kosnik, Kacper Radzikowski

7.14 /home/kamil/zpr/Monopoly/Field.h File Reference

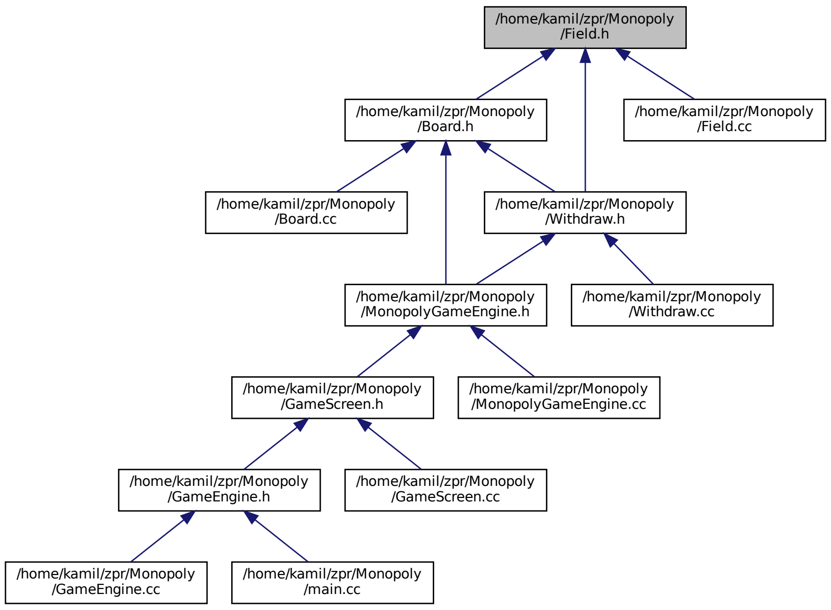
Header file containing staff for single monopoly board game field. Separated to many deriving classes types each for specific field type.

```
#include <cmath>
#include <iostream>
#include <map>
#include <memory>
#include <string>
#include <vector>
#include "ContextWindow.h"
#include "Player.h"
#include "main.h"
```

Include dependency graph for Field.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly /Field.h
/home/kamil/zpr/Monopoly /Board.h
/home/kamil/zpr/Monopoly /Field.cc
/home/kamil/zpr/Monopoly /Board.cc
/home/kamil/zpr/Monopoly /Withdraw.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /Withdraw.cc
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class HouseException

  Custom exception class for handling invalid house numbers.
- class Field

  Base class representing a generic game field.
- class PropertyField

  Derived class representing a property field on the game board.
- class StreetField

  Derived class representing a street field on the game board.
- class StationField

  Derived class representing a station field on the game board.
- class UtilityField
- class TaxField

  Derived class representing a tax field on the monopoly board.

7.14.1 Detailed Description

Header file containing staff for single monopoly board game field. Separated to many deriving classes types each for specific field type.

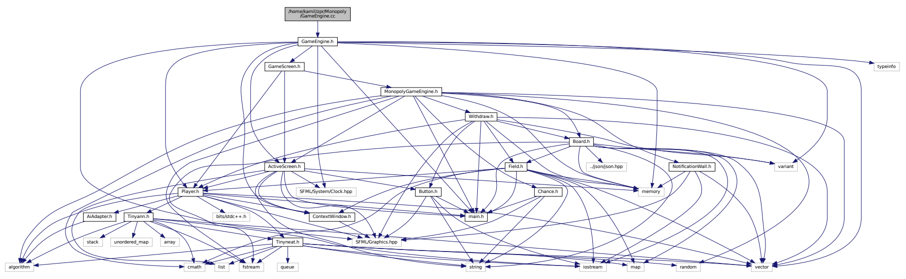
Author

Kamil Kosnik, Kacper Radzikowski

7.15 /home/kamil/zpr/Monopoly/GameEngine.cc File Reference

Source file for GameEngine class, used to handle lowes level program operations as input interactions (mouse, keyboard) or display window.

```
#include "GameEngine.h"
```
Include dependency graph for GameEngine.cc:



7.15.1 Detailed Description

Source file for GameEngine class, used to handle lowes level program operations as input interactions (mouse, keyboard) or display window.

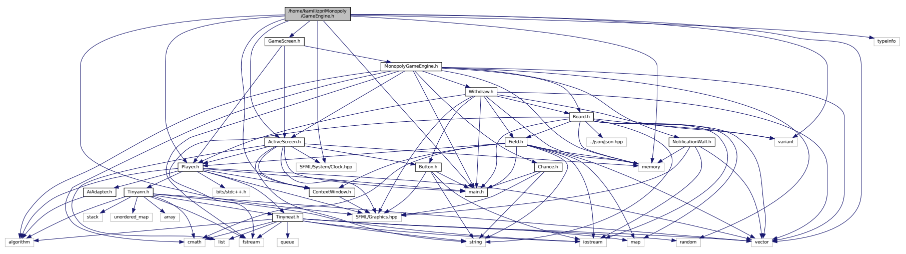
Author

Kamil Kosnik, Kacper Radzikowski

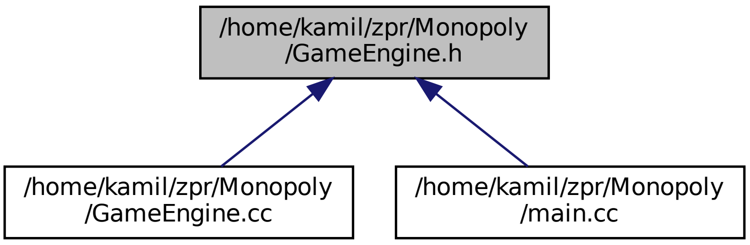
7.16 /home/kamil/zpr/Monopoly/GameEngine.h File Reference

Header file for GameEngine class, used to handle lowes level program operations as input interactions (mouse, keyboard) or display window.

```
#include <SFML/Graphics.hpp>
#include <SFML/System/Clock.hpp>
#include <memory>
#include <typeinfo>
#include <variant>
#include <vector>
#include "ActiveScreen.h"
#include "ContextWindow.h"
#include "GameScreen.h"
#include "Player.h"
#include "main.h"
```
Include dependency graph for GameEngine.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GameEngine

  Handles low-level program operations, including input interactions and window display.

7.16.1 Detailed Description

Header file for GameEngine class, used to handle lowes level program operations as input interactions (mouse, keyboard) or display window.

Author

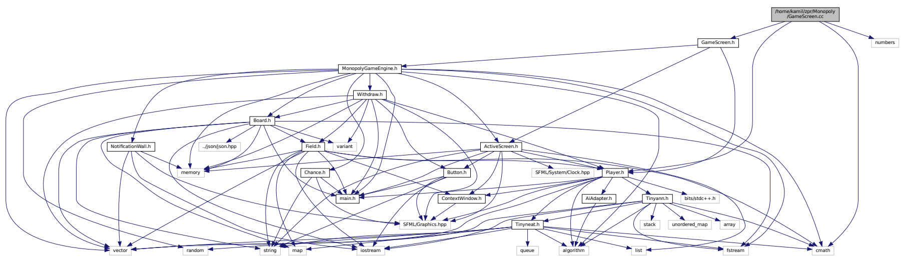
Kamil Kosnik, Kacper Radzikowski

7.17 /home/kamil/zpr/Monopoly/GameScreen.cc File Reference

Source file for game screen class deriving from ActiveScreen class. Used to handle monopoly game activities and drawing.

```
#include "GameScreen.h"
#include <cmath>
#include <numbers>
#include "Player.h"
```

Include dependency graph for GameScreen.cc:



7.17.1 Detailed Description

Source file for game screen class deriving from ActiveScreen class. Used to handle monopoly game activities and drawing.

Author

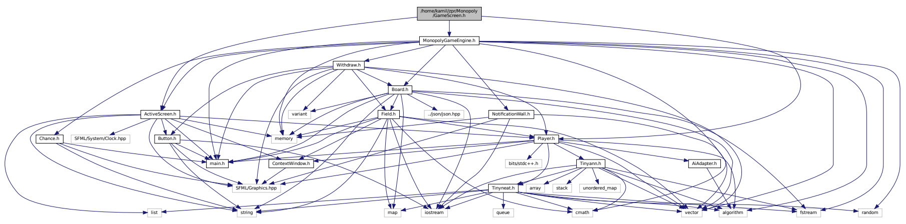
Kamil Kosnik, Kacper Radzikowski

7.18 /home/kamil/zpr/Monopoly/GameScreen.h File Reference

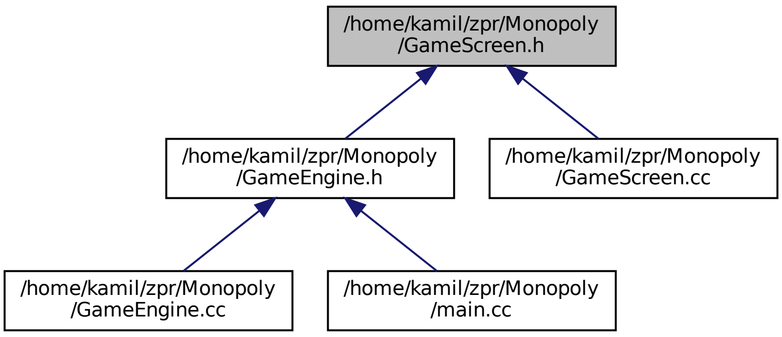
Header file for game screen class deriving from ActiveScreen class. Used to handle monopoly game activities and drawing.

```
#include "ActiveScreen.h"
#include "MonopolyGameEngine.h"
#include "Player.h"
```

Include dependency graph for GameScreen.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.cc
/home/kamil/zpr/Monopoly/main.cc

Classes

- class GameScreen

7.18.1 Detailed Description

Header file for game screen class deriving from ActiveScreen class. Used to handle monopoly game activities and drawing.

Author

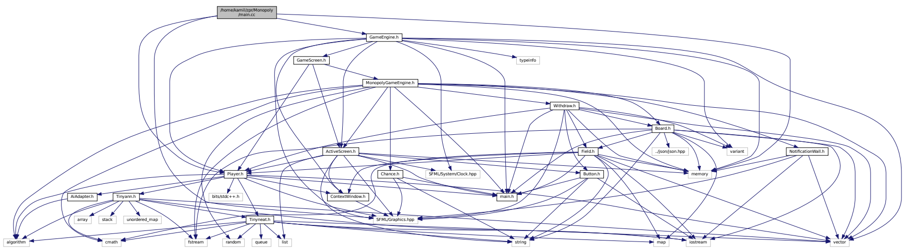
Kamil Kosnik, Kacper Radzikowski

7.19 /home/kamil/zpr/Monopoly/main.cc File Reference

Source file launching monopoly game vs AI project.

```
#include <memory>
#include "GameEngine.h"
#include "Player.h"
#include "SFML/Graphics.hpp"
```

Include dependency graph for main.cc:



Functions

- int main ()

Variables

- unsigned int WIDTH_MAX = 1920
- unsigned int HEIGHT_MAX = 1080
- unsigned int width
- unsigned int height
- unsigned int FRAMES_PER_SEC_MAX = 30
- bool TRAIN = false
- int GAMES_IN_ROUND = 5

7.19.1 Detailed Description

Source file launching monopoly game vs AI project.

Author

Kamil Kosnik, Kacper Radzikowski

7.19.2 Function Documentation

7.19.2.1 main()

```
int main ( )
```

7.19.3 Variable Documentation

7.19.3.1 FRAMES_PER_SEC_MAX

```
unsigned int FRAMES_PER_SEC_MAX = 30
```

7.19.3.2 GAMES_IN_ROUND

```
int GAMES_IN_ROUND = 5
```

7.19.3.3 height

```
unsigned int height
```

7.19.3.4 HEIGHT_MAX

```
unsigned int HEIGHT_MAX = 1080
```

7.19.3.5 TRAIN

```
bool TRAIN = false
```

7.19.3.6 width

```
unsigned int width
```

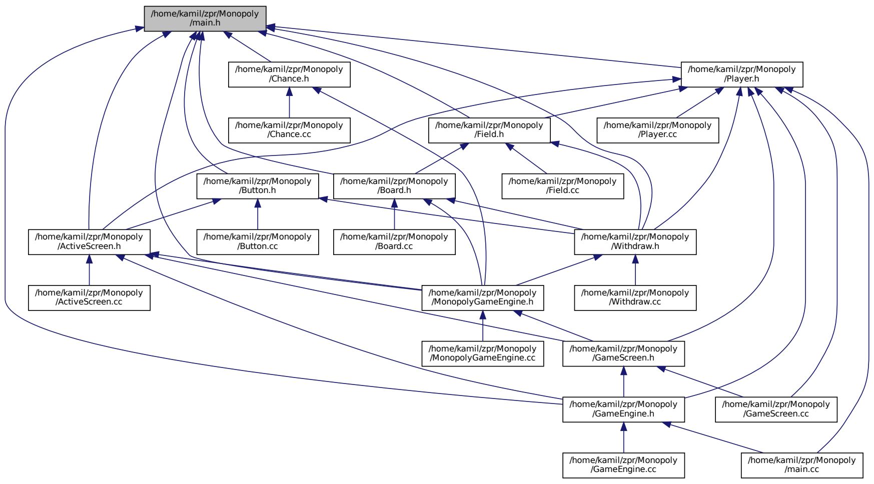
7.19.3.7 WIDTH_MAX

```
unsigned int WIDTH_MAX = 1920
```

7.20 /home/kamil/zpr/Monopoly/main.h File Reference

Header file containing structures shared between project files.

This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly /main.h
/home/kamil/zpr/Monopoly /Chance.h
/home/kamil/zpr/Monopoly /Player.h
/home/kamil/zpr/Monopoly /Chance.cc
/home/kamil/zpr/Monopoly /Field.h
/home/kamil/zpr/Monopoly /Player.cc
/home/kamil/zpr/Monopoly /Button.h
/home/kamil/zpr/Monopoly /Board.h
/home/kamil/zpr/Monopoly /Field.cc
/home/kamil/zpr/Monopoly /ActiveScreen.h
/home/kamil/zpr/Monopoly /Button.cc
/home/kamil/zpr/Monopoly /Board.cc
/home/kamil/zpr/Monopoly /Withdraw.h
/home/kamil/zpr/Monopoly /ActiveScreen.cc
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /Withdraw.cc
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class DimensionException

  Exception for handling passing wrong dimensions to any displayed object.
- class RotationException

  Exception for handling passing wrong rotation to any displayed object.
- class SpriteOffsetException

  Exception for handling wrong offset to any displayed object.
- struct playerSettings

  Struct describing player settings manipulted from game menu.

Enumerations

- enum TurnState {
ROLL_DICE , FIELD_ACTION , BUY_ACTION , PAY_RENT ,
TURN_END , WITHDRAW_ONGOING , RESULTS , NO_TURN }

    Enum describing monopoly game states.
- enum GameScreenType {
BOARDGAME , WITHDRAW_CHOOSE_PLAYER , WITHDRAW_ADD_VALUE , WITHDRAW_DECISION ,
AUCTION , RESULT }

    Enum describing possible showed screens in game screen object.
- enum ActiveScreenType { NONE , MAIN_MENU , GAME_MENU , MONOPOLY_GAME }

    Enum describing possible screen to be shown.
- enum ScreenEventType {
IDLE , EXIT , PLAY , RETURN_TO_MAIN_MENU ,
PLAYER_1_SET_NONE , PLAYER_2_SET_NONE , PLAYER_3_SET_NONE , PLAYER_4_SET_NONE ,
PLAYER_1_SET_HUMAN , PLAYER_2_SET_HUMAN , PLAYER_3_SET_HUMAN , PLAYER_4_SET_HUMAN
,
PLAYER_1_SET_AI , PLAYER_2_SET_AI , PLAYER_3_SET_AI , PLAYER_4_SET_AI ,
PLAYER_1_SET_AI_LEVEL_1 , PLAYER_2_SET_AI_LEVEL_1 , PLAYER_3_SET_AI_LEVEL_1 ,
PLAYER_4_SET_AI_LEVEL_1 ,
PLAYER_1_SET_AI_LEVEL_2 , PLAYER_2_SET_AI_LEVEL_2 , PLAYER_3_SET_AI_LEVEL_2 ,
PLAYER_4_SET_AI_LEVEL_2 ,
PLAYER_1_SET_AI_LEVEL_3 , PLAYER_2_SET_AI_LEVEL_3 , PLAYER_3_SET_AI_LEVEL_3 ,
PLAYER_4_SET_AI_LEVEL_3 ,
START_GAME , GAME_ENDED }

    Enum describing events that can be returned from showed screens of ActiveScreenType type.
- enum FieldType {
STREET , STATION , UTILITY , GO ,
CHANCE , COMMUNITY_CHEST , TAX , JAIL ,
FREE_PARKING , GO_TO_JAIL }

    Enum describing possible fields types in monopoly board.
- enum StreetTiers {
NO_HOUSES , ONE_HOUSE , TWO_HOUESES , THREE_HOUSES ,
FOUR_HOUSES , HOTEL }

    Enum describing possible street tiers (houses and hotel posessions)
- enum StationTiers { ONE_STATION , TWO_STATIONS , THREE_STATIONS , FOUR_STATIONS }

    Enum describing possible station tiers (how many is owned)
- enum UtilityTiers { ONE_UTILITY , TWO_UTILITIES }

    Enum describing possible utility tiers (how many is owned)
- enum Decision { YES , NO }

    Enums describing ai adapter decisions.
- enum JailDecision { ROLL , PAY , CARD }
- enum BuyDecision { BUY , RESIGN }

7.20.1 Detailed Description

Header file containing structures shared between project files.

Author

Kamil Kosnik, Kacper Radzikowski

7.20.2 Enumeration Type Documentation

7.20.2.1 ActiveScreenType

```
enum ActiveScreenType
```

Enum describing possible screen to be shown.

Enumerator

NONE	
MAIN_MENU	
GAME_MENU	
MONOPOLY_GAME	

7.20.2.2 BuyDecision

```
enum BuyDecision
```

Enumerator

BUY	
RESIGN	

7.20.2.3 Decision

```
enum Decision
```

Enums describing ai adapter decisions.

Enumerator

YES	
NO	

7.20.2.4 FieldType

```
enum FieldType
```

Enum describing possible fields types in monopoly board.

Enumerator

STREET	
STATION	
UTILITY	
GO	
CHANCE	
COMMUNITY_CHEST	
TAX	
JAIL	
FREE_PARKING	
GO_TO_JAIL	

7.20.2.5 GameScreenType

```
enum GameScreenType
```

Enum describing possible showed screens in game screen object.

Enumerator

BOARDGAME	
WITHDRAW_CHOOSE_PLAYER	
WITHDRAW_ADD_VALUE	
WITHDRAW_DECISION	
AUCTION	
RESULT	

7.20.2.6 JailDecision

```
enum JailDecision
```

Enumerator

ROLL	
PAY	
CARD	

7.20.2.7 ScreenEventType

```
enum ScreenEventType
```

Enum describing events that can be returned from showed screens of ActiveScreenType type.

Enumerator

IDLE	
EXIT	
PLAY	
RETURN_TO_MAIN_MENU	
PLAYER_1_SET_NONE	
PLAYER_2_SET_NONE	
PLAYER_3_SET_NONE	
PLAYER_4_SET_NONE	
PLAYER_1_SET_HUMAN	
PLAYER_2_SET_HUMAN	
PLAYER_3_SET_HUMAN	
PLAYER_4_SET_HUMAN	
PLAYER_1_SET_AI	
PLAYER_2_SET_AI	
PLAYER_3_SET_AI	
PLAYER_4_SET_AI	
PLAYER_1_SET_AI_LEVEL↩ _1	
PLAYER_2_SET_AI_LEVEL↩ _1	
PLAYER_3_SET_AI_LEVEL↩ _1	
PLAYER_4_SET_AI_LEVEL↩ _1	
PLAYER_1_SET_AI_LEVEL↩ _2	
PLAYER_2_SET_AI_LEVEL↩ _2	
PLAYER_3_SET_AI_LEVEL↩ _2	
PLAYER_4_SET_AI_LEVEL↩ _2	
PLAYER_1_SET_AI_LEVEL↩ _3	
PLAYER_2_SET_AI_LEVEL↩ _3	
PLAYER_3_SET_AI_LEVEL↩ _3	
PLAYER_4_SET_AI_LEVEL↩ _3	
START_GAME	
GAME_ENDED	

7.20.2.8 StationTiers

```
enum StationTiers
```

Enum describing possible station tiers (how many is owned)

Enumerator

ONE_STATION	
TWO_STATIONS	
THREE_STATIONS	
FOUR_STATIONS	

7.20.2.9 StreetTiers

```
enum StreetTiers
```

Enum describing possible street tiers (houses and hotel posessions)

Enumerator

NO_HOUSES	
ONE_HOUSE	
TWO_HOUESES	
THREE_HOUSES	
FOUR_HOUSES	
HOTEL	

7.20.2.10 TurnState

```
enum TurnState
```

Enum describing monopoly game states.

Enumerator

ROLL_DICE	
FIELD_ACTION	
BUY_ACTION	
PAY_RENT	
TURN_END	
WITHDRAW_ONGOING	
RESULTS	
NO_TURN	

7.20.2.11 UtilityTiers

```
enum UtilityTiers
```

Enum describing possible utility tiers (how many is owned)

Enumerator

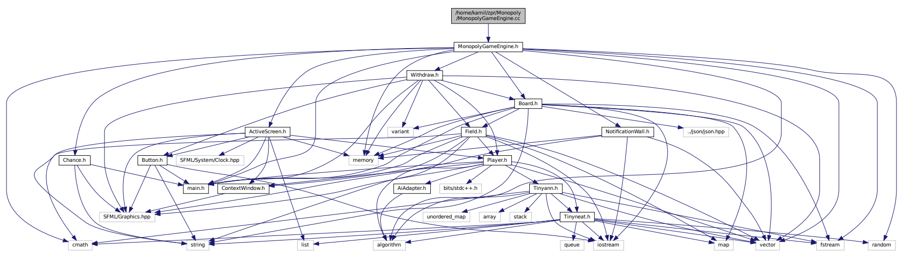
ONE_UTILITY	
TWO_UTILITIES	

7.21 /home/kamil/zpr/Monopoly/MonopolyGameEngine.cc File Reference

Source file of class used to handle whole monopoly game process, turns, actions with players, board etc.

```
#include "MonopolyGameEngine.h"
```

Include dependency graph for MonopolyGameEngine.cc:



7.21.1 Detailed Description

Source file of class used to handle whole monopoly game process, turns, actions with players, board etc.

Author

Kamil Kosnik, Kacper Radzikowski

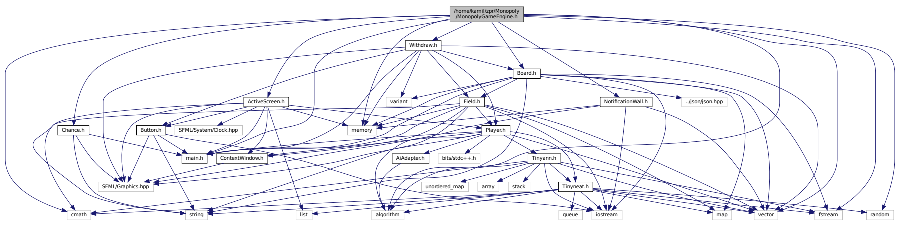
7.22 /home/kamil/zpr/Monopoly/MonopolyGameEngine.h File Reference

Header file of class used to handle whole monopoly game process, turns, actions with players, board etc.

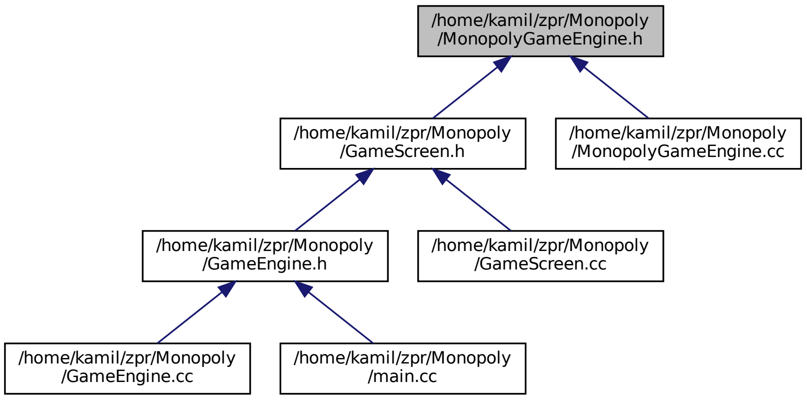
```
#include <algorithm>
#include <cmath>
#include <fstream>
#include <memory>
#include <random>
#include <vector>
#include "ActiveScreen.h"
#include "Board.h"
#include "Chance.h"
#include "NotificationWall.h"
#include "Withdraw.h"
```

`#include "main.h"`

Include dependency graph for MonopolyGameEngine.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MonopolyGameEngine

  Class representing the main game engine for the Monopoly game.

Enumerations

- enum AuctionState {
  NO_AUCTION , INITIALIZATION , PASS_BIDDING_TURN , BIDDING ,
  ENDING }

  Enum representing the state of an auction.

7.22.1 Detailed Description

Header file of class used to handle whole monopoly game process, turns, actions with players, board etc.

Author

Kamil Kosnik, Kacper Radzikowski

7.22.2 Enumeration Type Documentation

7.22.2.1 AuctionState

```
enum AuctionState
```

Enum representing the state of an auction.

Enumerator

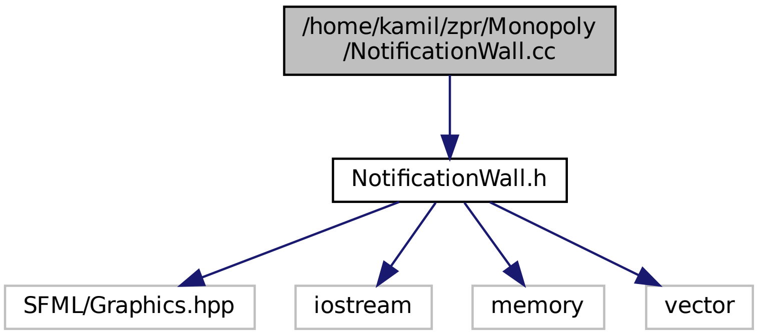
NO_AUCTION	
INITIALIZATION	
PASS_BIDDING_TURN	
BIDDING	
ENDING	

7.23 /home/kamil/zpr/Monopoly/NotificationWall.cc File Reference

Source file for the NotificationWall class.

```
#include "NotificationWall.h"
```

Include dependency graph for NotificationWall.cc:



7.23.1 Detailed Description

Source file for the NotificationWall class.

The NotificationWall class is used to display a list of messages/notifications as a finite list with a certain length. It automatically rolls over when it receives the next message.

Author

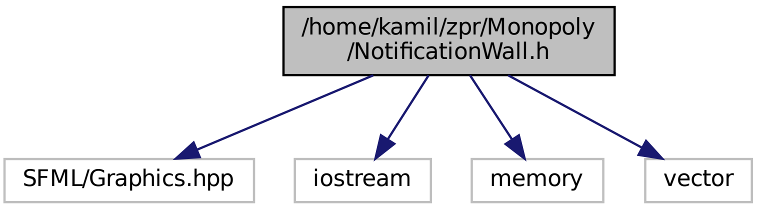
Kamil Kosnik, Kacper Radzikowski

7.24 /home/kamil/zpr/Monopoly/NotificationWall.h File Reference

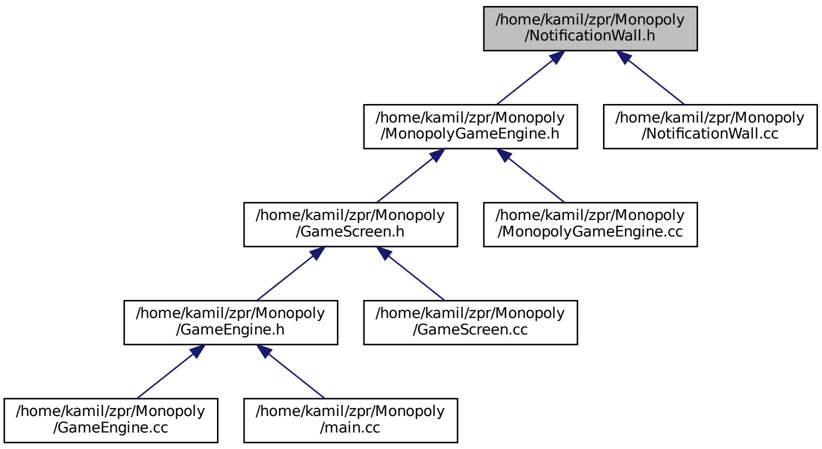
Header file for the NotificationWall class.

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <memory>
#include <vector>
```

Include dependency graph for NotificationWall.h:



This graph shows which files directly or indirectly include this file:



Classes

- class NotificationWall

    Represents a notification wall that displays messages.

7.24.1 Detailed Description

Header file for the NotificationWall class.

The NotificationWall class is used to display a list of messages/notifications as a finite list with a certain length. It automatically rolls over when it receives the next message.

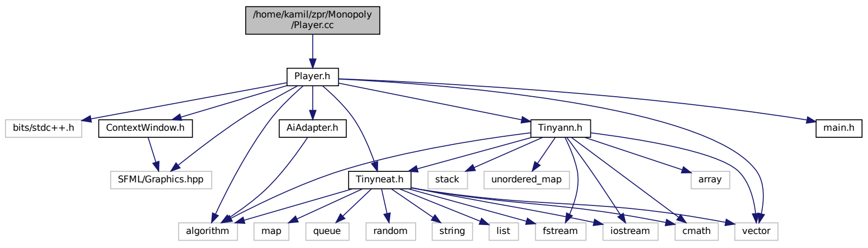
Author

Kamil Kosnik, Kacper Radzikowski

7.25 /home/kamil/zpr/Monopoly/Player.cc File Reference

Implementation file for Player class and AI Player class, containing data and methods for a player in a Monopoly game.

```
#include "Player.h"
```
Include dependency graph for Player.cc:



7.25.1 Detailed Description

Implementation file for Player class and AI Player class, containing data and methods for a player in a Monopoly game.

Author

Kamil Kosnik, Kacper Radzikowski

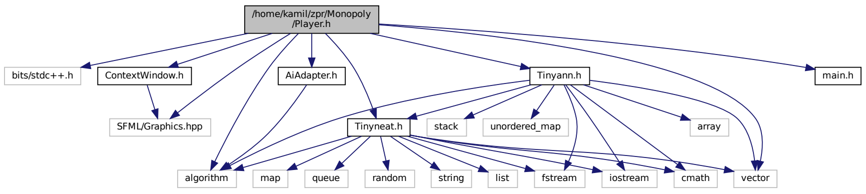
7.26 /home/kamil/zpr/Monopoly/Player.h File Reference

Implementation file for Player class and AI Player class, containing data and methods for a player in a Monopoly game.

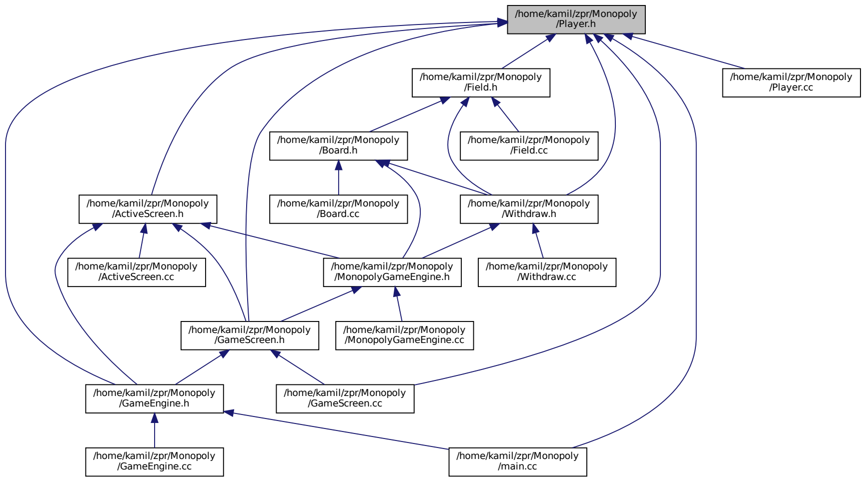
```
#include <bits/stdc++.h>
#include <SFML/Graphics.hpp>
#include <algorithm>
#include <vector>
#include "AiAdapter.h"
#include "ContextWindow.h"
```

```
#include "Tinyann.h"
#include "Tinyneat.h"
#include "main.h"
```

Include dependency graph for Player.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/Player.h
/home/kamil/zpr/Monopoly/Field.h
/home/kamil/zpr/Monopoly/Player.cc
/home/kamil/zpr/Monopoly/Board.h
/home/kamil/zpr/Monopoly/Field.cc
/home/kamil/zpr/Monopoly/ActiveScreen.h
/home/kamil/zpr/Monopoly/Board.cc
/home/kamil/zpr/Monopoly/Withdraw.h
/home/kamil/zpr/Monopoly/ActiveScreen.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.h
/home/kamil/zpr/Monopoly/Withdraw.cc
/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.cc
/home/kamil/zpr/Monopoly/main.cc

Classes

- class Player

    Represents a player in a Monopoly game.

- class AiPlayer

    Represents an AI player in a Monopoly game, inheriting from Player.

7.26.1 Detailed Description

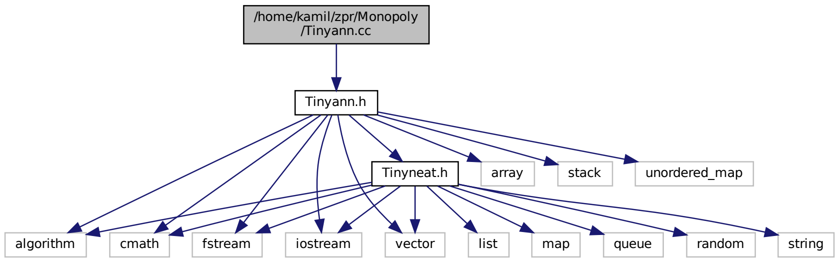
Implementation file for Player class and AI Player class, containing data and methods for a player in a Monopoly game.

Author

Kamil Kosnik, Kacper Radzikowski

7.27 /home/kamil/zpr/Monopoly/Tinyann.cc File Reference

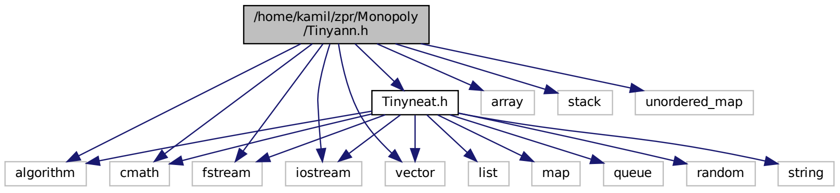
```
#include "Tinyann.h"
```
Include dependency graph for Tinyann.cc:



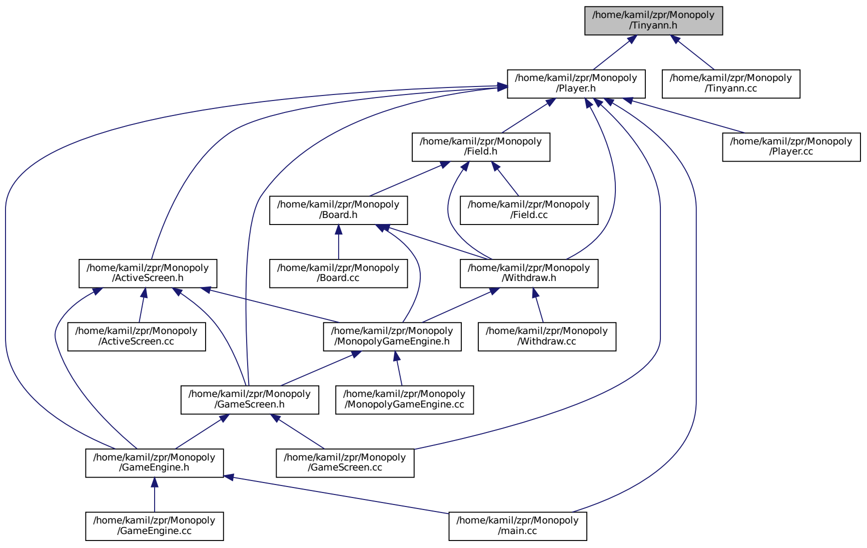
7.28 /home/kamil/zpr/Monopoly/Tinyann.h File Reference

Header file for AI implementation of tiny library from github user hav4ik, repository: https://github.com/hav4ik/tinyai.

```
#include <algorithm>
#include <array>
#include <cmath>
#include <fstream>
#include <iostream>
#include <stack>
#include <unordered_map>
#include <vector>
#include "Tinyneat.h"
```
Include dependency graph for Tinyann.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/Tinyann.h
/home/kamil/zpr/Monopoly/Player.h
/home/kamil/zpr/Monopoly/Tinyann.cc
/home/kamil/zpr/Monopoly/Field.h
/home/kamil/zpr/Monopoly/Player.cc
/home/kamil/zpr/Monopoly/Board.h
/home/kamil/zpr/Monopoly/Field.cc
/home/kamil/zpr/Monopoly/ActiveScreen.h
/home/kamil/zpr/Monopoly/Board.cc
/home/kamil/zpr/Monopoly/Withdraw.h
/home/kamil/zpr/Monopoly/ActiveScreen.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.h
/home/kamil/zpr/Monopoly/Withdraw.cc
/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.cc
/home/kamil/zpr/Monopoly/main.cc

Classes

- class ann::neuron
- class ann::neuralnet

Namespaces

- ann

Enumerations

- enum ann::type { ann::RECURRENT , ann::NON_RECURRENT }

7.28.1 Detailed Description

Header file for AI implementation of tiny library from github user hav4ik, repository: `https://github.↵com/hav4ik/tinyai`.

Author

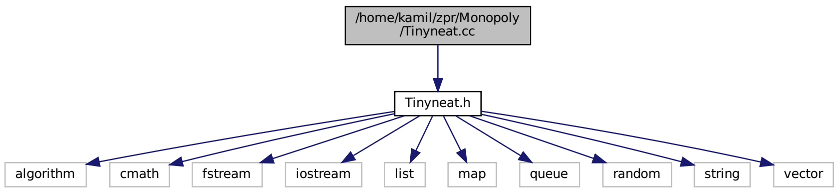
hav4ik

7.29 /home/kamil/zpr/Monopoly/Tinyneat.cc File Reference

Source file for AI implementation of tiny library from github user hav4ik, repository: https://github.↩ com/hav4ik/tinyai.

```
#include "Tinyneat.h"
```

Include dependency graph for Tinyneat.cc:



7.29.1 Detailed Description

Source file for AI implementation of tiny library from github user hav4ik, repository: https://github.↩ com/hav4ik/tinyai.

Author

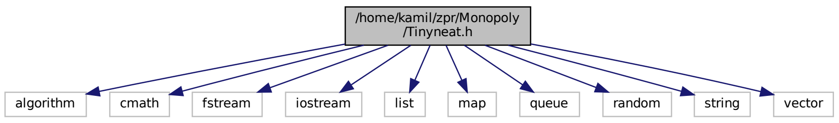
hav4ik

7.30 /home/kamil/zpr/Monopoly/Tinyneat.h File Reference

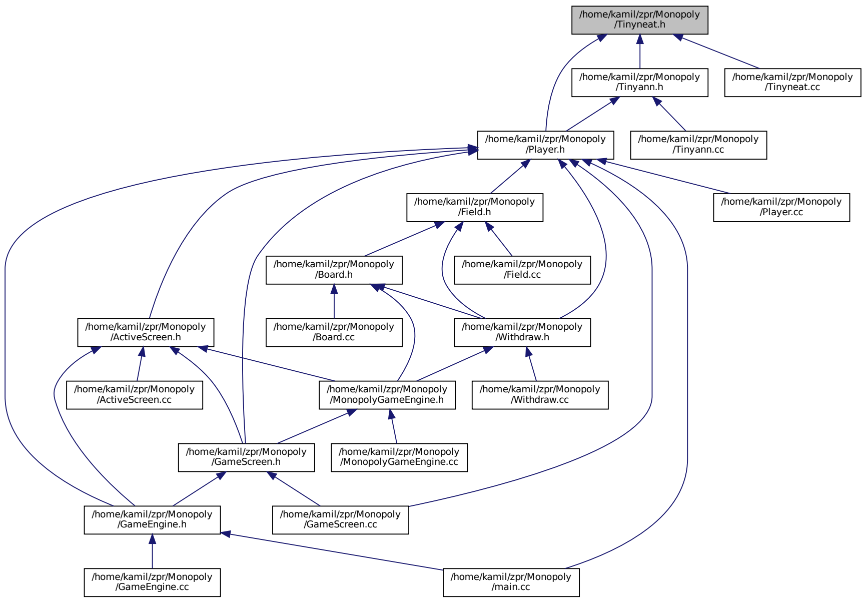
Header file for AI implementation of tiny library from github user hav4ik, repository: https://github.↩ com/hav4ik/tinyai.

```
#include <algorithm>
#include <cmath>
#include <fstream>
#include <iostream>
#include <list>
#include <map>
#include <queue>
#include <random>
#include <string>
#include <vector>
```

Include dependency graph for Tinyneat.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly/Tinyneat.h
/home/kamil/zpr/Monopoly/Tinyann.h
/home/kamil/zpr/Monopoly/Tinyneat.cc
/home/kamil/zpr/Monopoly/Player.h
/home/kamil/zpr/Monopoly/Tinyann.cc
/home/kamil/zpr/Monopoly/Field.h
/home/kamil/zpr/Monopoly/Player.cc
/home/kamil/zpr/Monopoly/Board.h
/home/kamil/zpr/Monopoly/Field.cc
/home/kamil/zpr/Monopoly/ActiveScreen.h
/home/kamil/zpr/Monopoly/Board.cc
/home/kamil/zpr/Monopoly/Withdraw.h
/home/kamil/zpr/Monopoly/ActiveScreen.cc
/home/kamil/zpr/Monopoly/MonopolyGameEngine.h
/home/kamil/zpr/Monopoly/Withdraw.cc
/home/kamil/zpr/Monopoly/GameScreen.h
/home/kamil/zpr/Monopoly/MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly/GameEngine.h
/home/kamil/zpr/Monopoly/GameScreen.cc
/home/kamil/zpr/Monopoly/GameEngine.cc
/home/kamil/zpr/Monopoly/main.cc

Classes

- struct neat::mutation_rate_container
- struct neat::speciating_parameter_container
- struct neat::network_info_container
- struct neat::gene
- class neat::genome
- struct neat::specie
- class neat::innovation_container
- class neat::pool

Namespaces

- neat

7.30.1 Detailed Description

Header file for AI implementation of tiny library from github user hav4ik, repository: `https://github.com/hav4ik/tinyai`.

Author

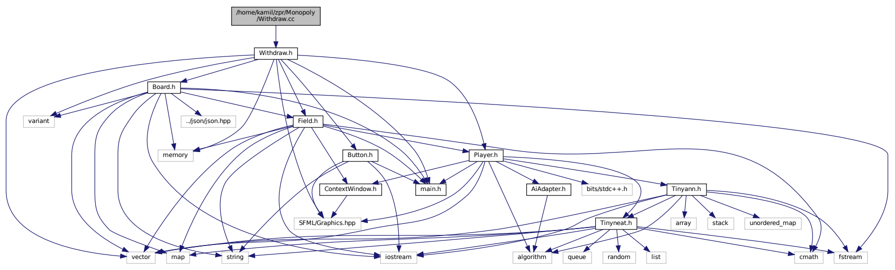
hav4ik

7.31 /home/kamil/zpr/Monopoly/Withdraw.cc File Reference

Source file for trade/withdraw mehanism in monopoly game between players.

```
#include "Withdraw.h"
```

Include dependency graph for Withdraw.cc:



7.31.1 Detailed Description

Source file for trade/withdraw mehanism in monopoly game between players.

Author

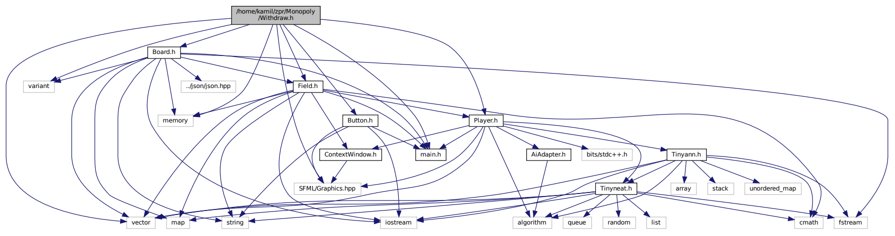
Kamil Kosnik, Kacper Radzikowski

7.32 /home/kamil/zpr/Monopoly/Withdraw.h File Reference

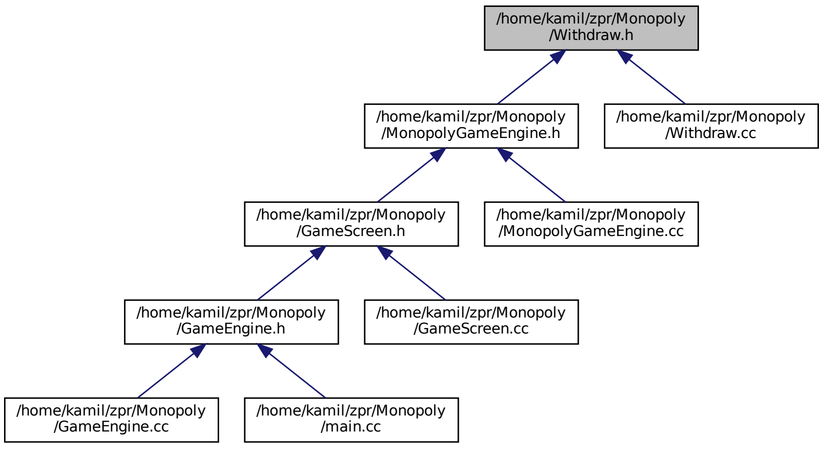
Header file for trade/withdraw mechanism in the monopoly game between players.

```
#include <SFML/Graphics.hpp>
#include <memory>
#include <variant>
#include <vector>
#include "Board.h"
#include "Button.h"
#include "Field.h"
#include "Player.h"
#include "main.h"
```

Include dependency graph for Withdraw.h:



This graph shows which files directly or indirectly include this file:


/home/kamil/zpr/Monopoly /Withdraw.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.h
/home/kamil/zpr/Monopoly /Withdraw.cc
/home/kamil/zpr/Monopoly /GameScreen.h
/home/kamil/zpr/Monopoly /MonopolyGameEngine.cc
/home/kamil/zpr/Monopoly /GameEngine.h
/home/kamil/zpr/Monopoly /GameScreen.cc
/home/kamil/zpr/Monopoly /GameEngine.cc
/home/kamil/zpr/Monopoly /main.cc

Classes

- class Withdraw

  Represents the trade and withdraw mechanism in a monopoly game.

7.32.1 Detailed Description

Header file for trade/withdraw mechanism in the monopoly game between players.

This file contains the declaration of the Withdraw class, which represents the trade and withdraw mechanism in a monopoly game between players. Player 1 - active in withdraw (offering one) Player 2 - passive in withdraw (decision maker)

Author

Kamil Kosnik, Kacper Radzikowski

Index