DistanzSpiel

Erzeugt von Doxygen 1.8.9.1

Son Jun 21 2015 21:42:59

Inhaltsverzeichnis

Kapitel 1

Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

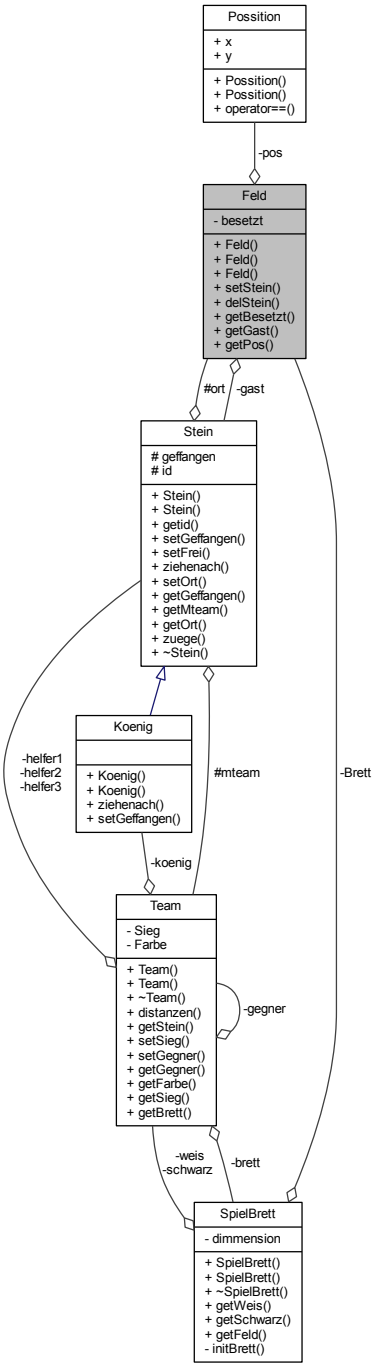
Kapitel 4

Klassen-Dokumentation

4.1 Feld Klassenreferenz

```
#include <Feld.h>
```

Zusammengehörigkeiten von Feld:



Öffentliche Methoden

- Feld ()
- Feld (short nx, short ny)
- Feld (Feld &f)
- void setStein (Stein *newstein)
- void delStein ()

- bool getBesetzt ()
- Stein ∗ getGast ()
- Possition getPos ()

Private Attribute

- bool besetzt
- Possition pos
- Stein ∗ gast =nullptr

4.1.1 Ausführliche Beschreibung

class Feld Diese Klasse Symbolisiert ein Feld auf einem Spielbrett.

4.1.2 Beschreibung der Konstruktoren und Destruktoren

4.1.2.1 Feld::Feld ( )

4.1.2.2 Feld::Feld (short *nx*, short *ny*)

Feld Konstruktor

Parameter

in	*nx*	x Koordinaten des Feldes
in	*ny*	y Koordinaten des Feldes

4.1.2.3 Feld::Feld (Feld & *f*)

4.1.3 Dokumentation der Elementfunktionen

4.1.3.1 void Feld::delStein ( )

delStein Loescht Zeiger auf Gast Setzt besetzt auf false

4.1.3.2 bool Feld::getBesetzt ( )

Get the value of besetzt.

Rückgabe

the value of besetzt.

4.1.3.3 Stein ∗ Feld::getGast ( )

getGast

Rückgabe

Gibt einen Pointer auf den Gast zurueck.

4.1.3.4 Possition Feld::getPos ( )

getPos

Rückgabe

the value of pos.

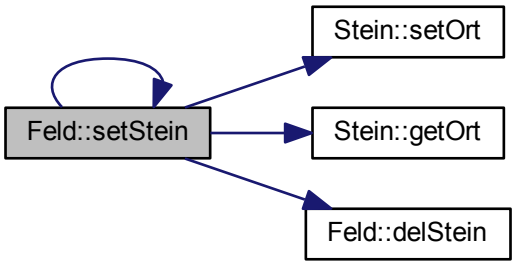
4.1.3.5 void Feld::setStein (Stein * *newstein*)

Setzt Stein auf das Feld und Markiert das Feld als Besetzt. Falls das Feld besetzt ist, werden die Gaeste/Steine getauscht.

Parameter

[in/out]	*newstein pointer auf den zu setzenden Stein.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.1.4 Dokumentation der Datenelemente

4.1.4.1 bool Feld::besetzt [private]

4.1.4.2 Stein* Feld::gast =nullptr [private]

4.1.4.3 Possition Feld::pos [private]

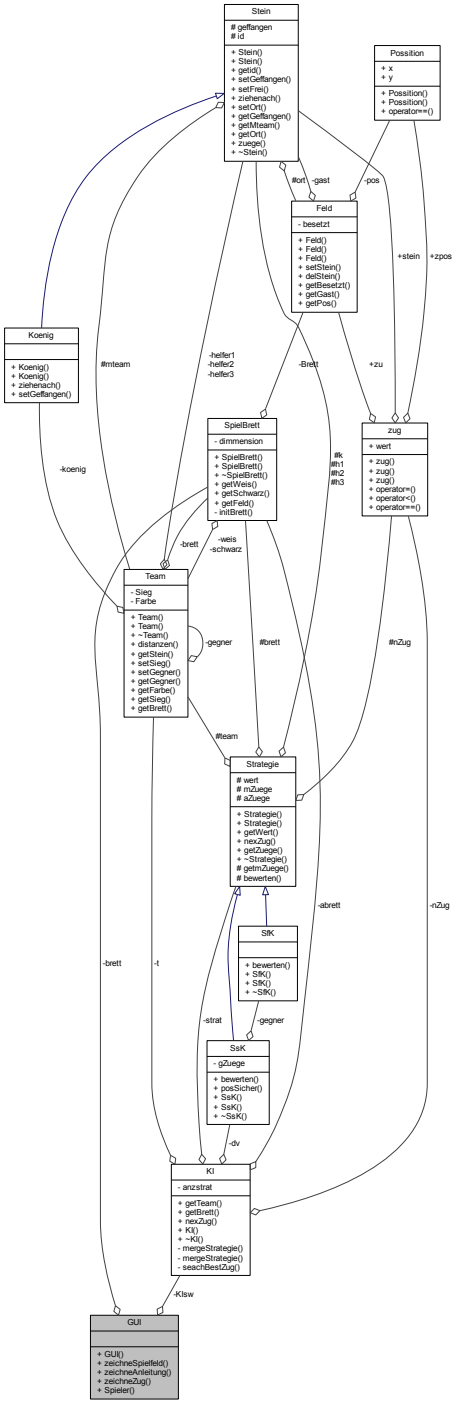
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Feld.h
- C:/Users/franz/workspace/distanz/src/cpp/Feld.cpp

4.2 GUI Klassenreferenz

```
#include <GUI.h>
```

Zusammengehörigkeiten von GUI:



Öffentliche Methoden

- GUI (SpielBrett ∗br, KI ∗ki)
- void zeichneSpielfeld (int zug, int spieler)
- void zeichneAnleitung ()
- void zeichneZug (int zug, int spieler, int zeile, int spalte)
- void Spieler (bool farbe, int zug, int spieler)

Private Attribute

- SpielBrett ∗ brett
- KI ∗ KIsw

4.2.1 Ausführliche Beschreibung

class GUI Die Abkürzung GUI steht für graphical user interface bzw. grafische Benutzeroberfläche. Durch diese Klasse wird dem Benutzer eine grafische Oberfläche zur Verfügung gestellt, über die er mit dem Programm interagierien kann. Alle Benutzereingaben erfolgen ausschließlich über die Tastatur. Unterstützend wird die Strucktur der erwarteten Eingabe in Klammern mit angegeben. Sollte dennoch der Benutzer eine Falsch Eingabe tätigen, so wird er darauf hingewiesen und kann seine Eingabe nach 3 Sekunken wiederholen. Die Darstellung des Spielfeldes und wichtiger Spielparameter erfolgt in der Windows Konsole über ANSI-Zeichen.

4.2.2 Beschreibung der Konstruktoren und Destruktoren

4.2.2.1 GUI::GUI (SpielBrett ∗ *br,* KI ∗ *ki*)

4.2.3 Dokumentation der Elementfunktionen

4.2.3.1 void GUI::Spieler (bool *farbe,* int *zug,* int *spieler*)

zeichneZug(zug,spieler,zeile,spalte) Mit dieser Funktion wird dem Spieler alle zulässigen Züge des ausgewählten Steins angezeigt.

Parameter

[int]	zug gibt den aktuellen Zug an
[int]	spieler gibt an, welcher Spieler gerade am Zug ist
[int]	zeile Zeile des ausgewählten Steins
[int]	spalte Spalte des ausgewählten Steins

4.2.3.2 void GUI::zeichneAnleitung ( )

zeichneSpielfeld(zug,spieler) Mit dieser Funktion wird das Spielfeld grafisch für den Spieler aufbereitet.

Parameter

[int]	zug gibt den aktuelle Zug an
[int]	spieler gibt an, welcher Spieler an Zug ist

4.2.3.3 void GUI::zeichneSpielfeld (int *zug,* int *spieler*)

GUI Diese Funktion ist ein Konstructor für eine Instanz von der Klasse Spielbrett.

4.2.3.4 void GUI::zeichneZug (int *zug,* int *spieler,* int *zeile,* int *spalte*)

zeichenAnleitung() Diese Funktion gibt dem Benutzer Auskunft über die Spielregeln.

4.2.4 Dokumentation der Datenelemente

4.2.4.1 SpielBrett∗ GUI::brett [private]

4.2.4.2 KI* GUI::Klsw [private]

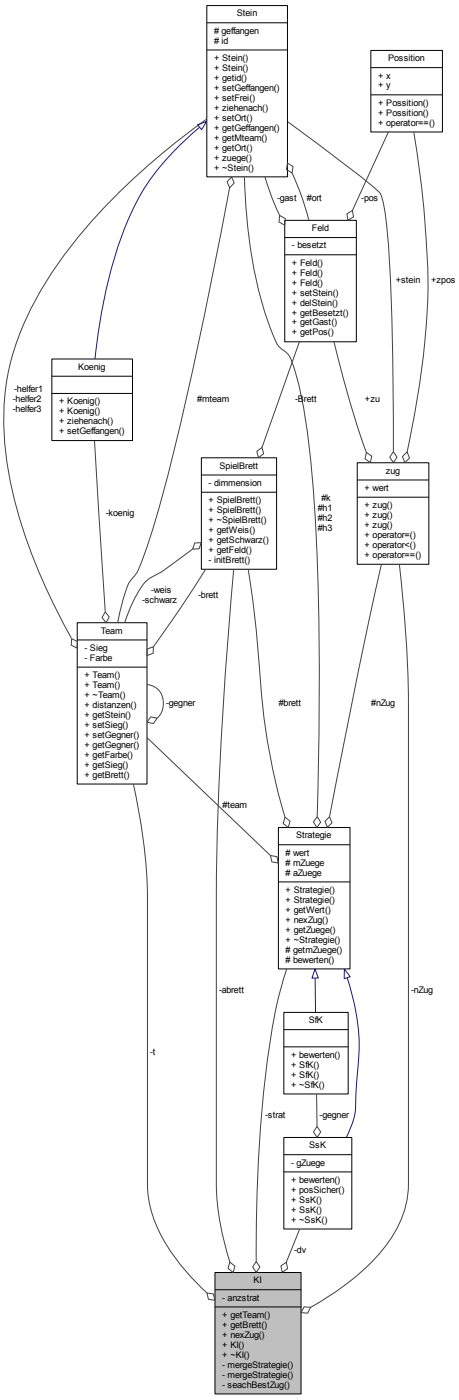
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- GUI.h
- C:/Users/franz/workspace/distanz/src/cpp/GUI.cpp

4.3 KI Klassenreferenz

```
#include <KI.h>
```

Zusammengehörigkeiten von KI:



Öffentliche Methoden

- Team & getTeam ()
- SpielBrett & getBrett ()
- void nexZug ()
- KI (Team &t)
- virtual ~KI ()

Private Methoden

- std::vector< zug > mergeStrategie (Strategie *st1, std::vector< zug > st2Zuege)
- std::vector< zug > mergeStrategie (Strategie *st1, Strategie *st2)
- void seachBestZug ()

Private Attribute

- Team & t
- SpielBrett & abrett
- Strategie * strat [anzstrat]
- SsK * dv
- zug nZug

Statische, private Attribute

- static const int anzstrat =4

4.3.1 Ausführliche Beschreibung

class KI Ist eine Klasse die aus den möglichen Spielzügen den besten auswählt. Sie ist mit zusätzlichen Strategien erweiterbar.

4.3.2 Beschreibung der Konstruktoren und Destruktoren

4.3.2.1 KI::KI (Team & *t*)

KI Konstruktor

Parameter

in,out	*t*	Referenz auf das Team, das gesteuert werden soll

4.3.2.2 KI::~KI ( ) [virtual]

4.3.3 Dokumentation der Elementfunktionen

4.3.3.1 SpielBrett & KI::getBrett ( )

getBrett

Rückgabe

Referenz auf das Spielbrett

4.3.3.2 Team & KI::getTeam ( )

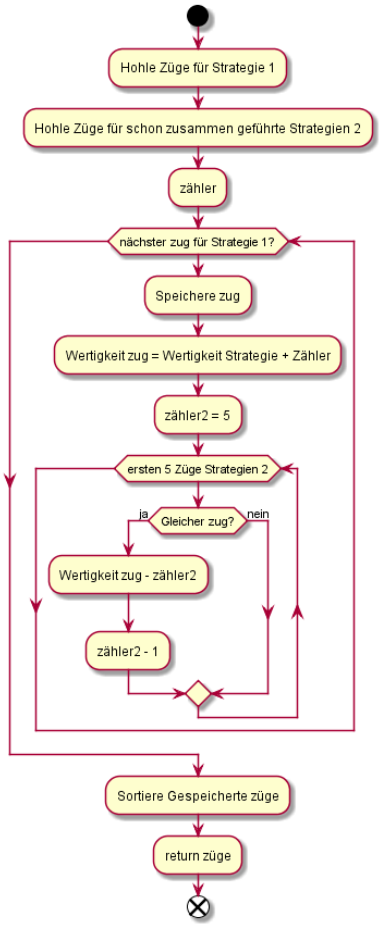
getTeam

Rückgabe

Referenz auf das gesteuerte Team

4.3.3.3 std::vector< zug > KI::mergeStrategie (Strategie * *st1*, std::vector< zug > *st2Zuege*) [private]

mergeStrategie Vereint zwei Strategien und führt die Wertigkeiten zusammen. Je kleiner die Wertigkeits-Zahl desto besser ist der zug.



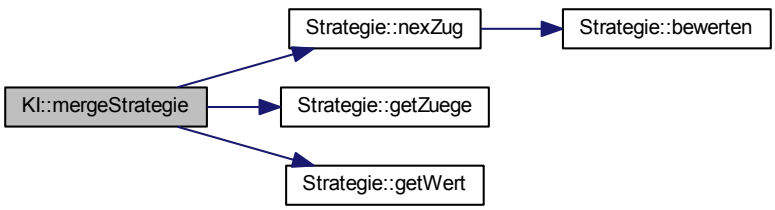
Parameter

in	*st1*	Pointer auf eine Strategie
in	*st2Zuege*	Vector mit Zügen.

Rückgabe

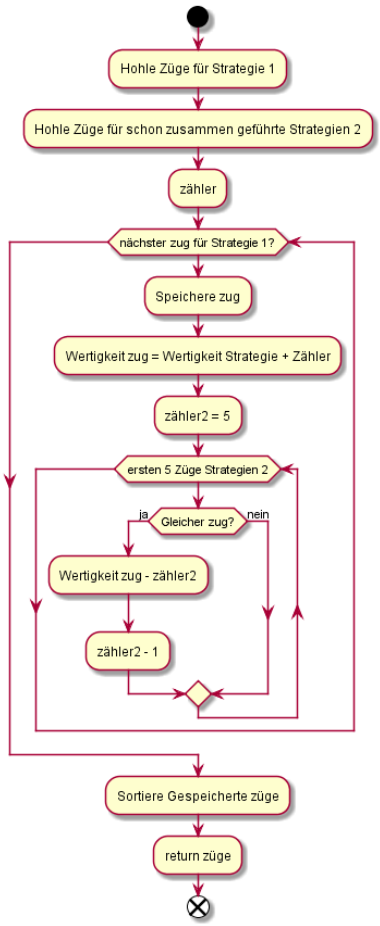
Vector mit nach wertigkeit sortierten zügen.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.3.3.4 std::vector< zug > KI::mergeStrategie (Strategie * *st1*, Strategie * *st2*) [private]

- mergeStrategie Vereint zwei Strategien und fürt die Wertigkeiten zusammen. Je kleiner die wertigkeits Zahl desto besser ist der Zug.



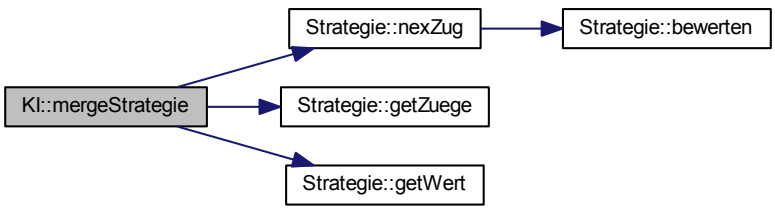
Parameter

in	*st1*	Pointer auf eine Strategie.
in	*st2*	Pointer auf eine Strategie.

Rückgabe

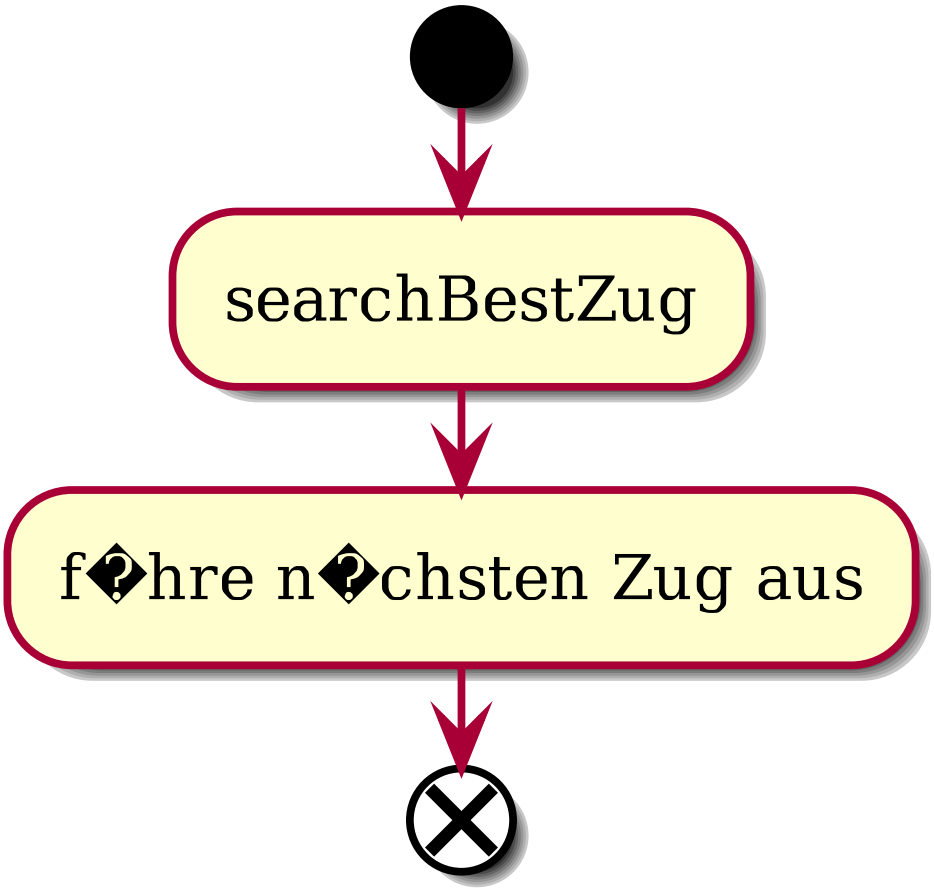
Vector mit nach Wertigkeit sortierten Zügen.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

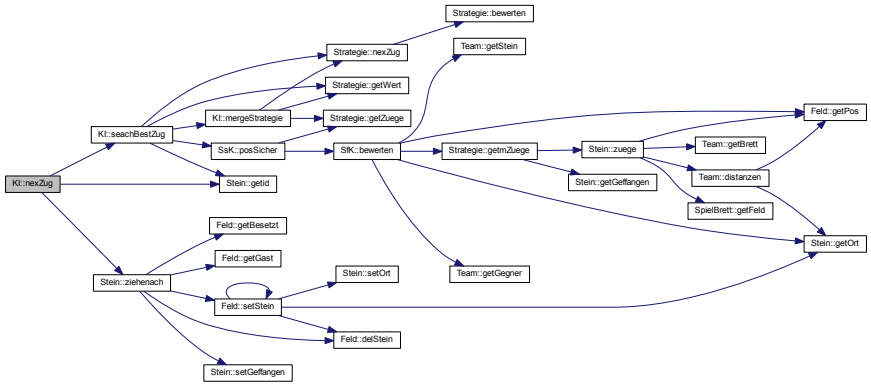


4.3.3.5 void Kl::nexZug ( )

nexZug Führt den nächsten Zug aus

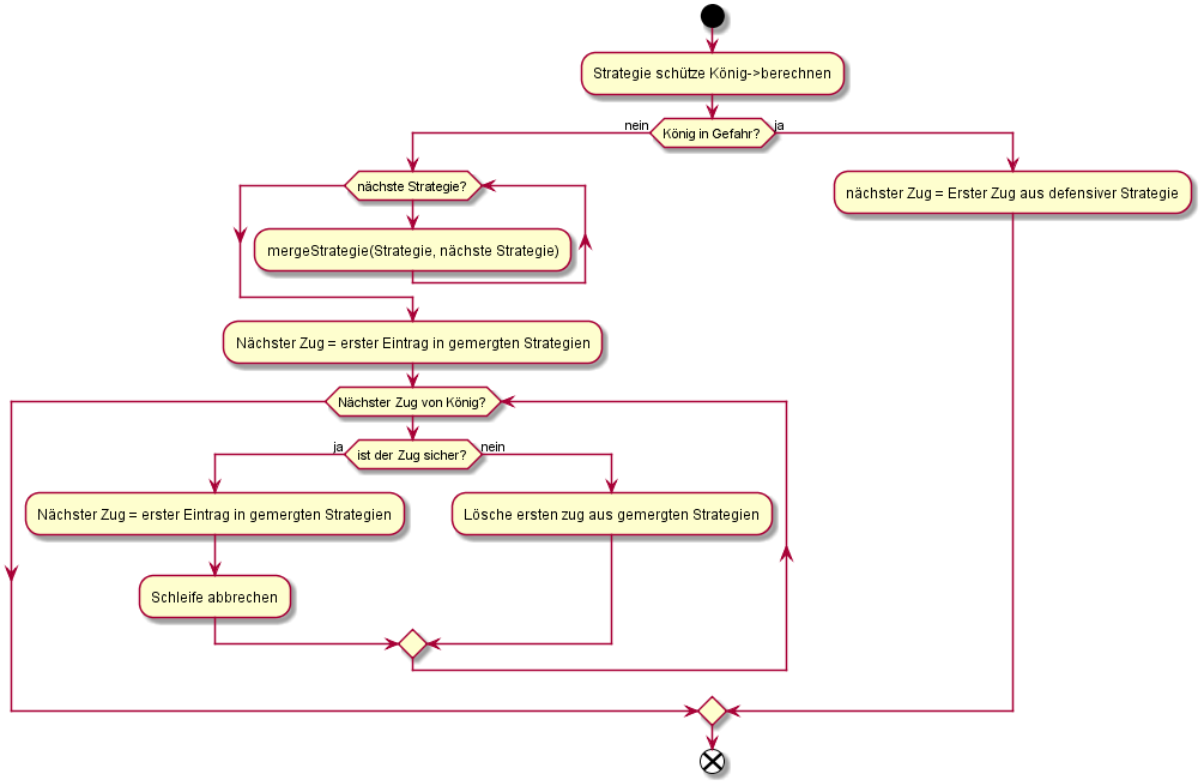


Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

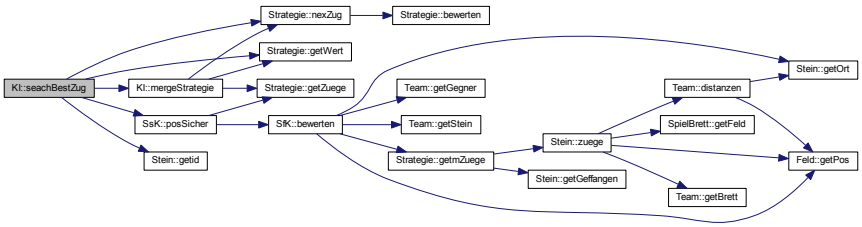


4.3.3.6 void Kl::seachBestZug ( ) [private]

seachBestZug Wählt aus den zusammengefürten Strategien den besten Zug aus. Und stelt sicher das der König nicht in Gefahr ist bzw. kommt.



Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.3.4 Dokumentation der Datenelemente

4.3.4.1 SpielBrett& KI::abrett [private]

4.3.4.2 const int KI::anzstrat =4 [static], [private]

4.3.4.3 SsK* KI::dv [private]

4.3.4.4 zug KI::nZug [private]

4.3.4.5 Strategie* KI::strat[anzstrat] [private]

4.3.4.6 Team& KI::t [private]

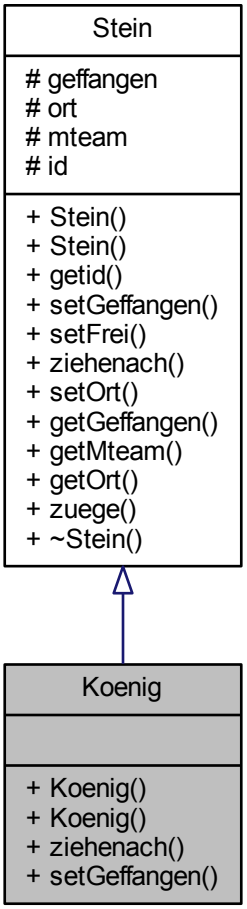
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- KI.h
- C:/Users/franz/workspace/distanz/src/cpp/KI.cpp

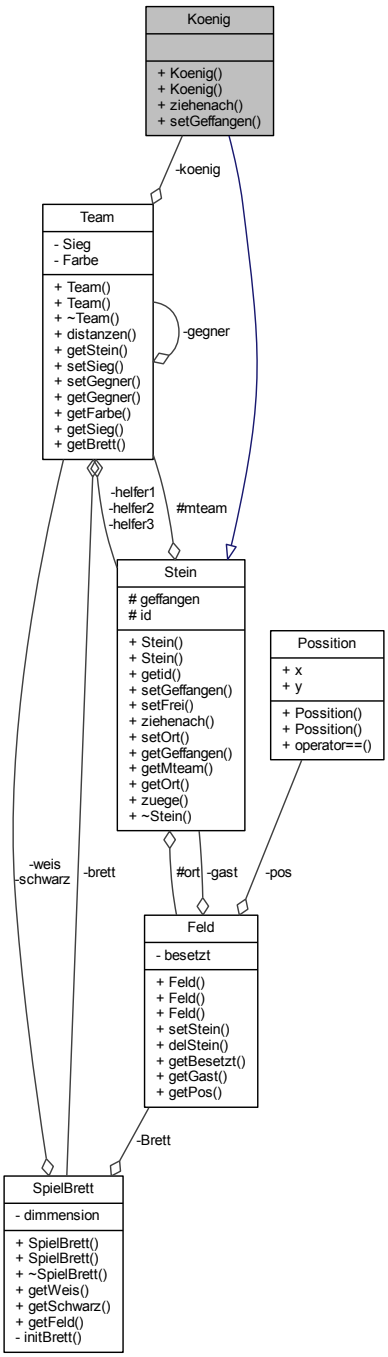
4.4 Koenig Klassenreferenz

`#include <Koenig.h>`

Klassendiagramm für Koenig:



Zusammengehörigkeiten von Koenig:



Öffentliche Methoden

- Koenig ()
- Koenig (int id, Feld ∗startplatz, Team ∗mt)
- virtual bool ziehenach (Feld ∗ziehe) override
- virtual void setGeffangen () override

Weitere Geerbte Elemente

4.4.1 Beschreibung der Konstruktoren und Destruktoren

4.4.1.1 Koenig::Koenig ( )

4.4.1.2 Koenig::Koenig (int *id,* Feld * *startplatz,* Team * *mt*)

4.4.2 Dokumentation der Elementfunktionen

4.4.2.1 void Koenig::setGeffangen ( ) [override], [virtual]

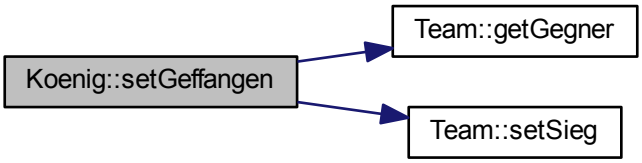
ziehenach(Feld) Mit dieser Methode hat der König die Möglichkeit seine Helfer zu befreien und sich auf dem Spielfeld zu bewegen.

Parameter

[Feld]	*ziehe hierbei handelt sich um einen Pointer auf das Feld, das er springen soll

Erneute Implementation von Stein.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

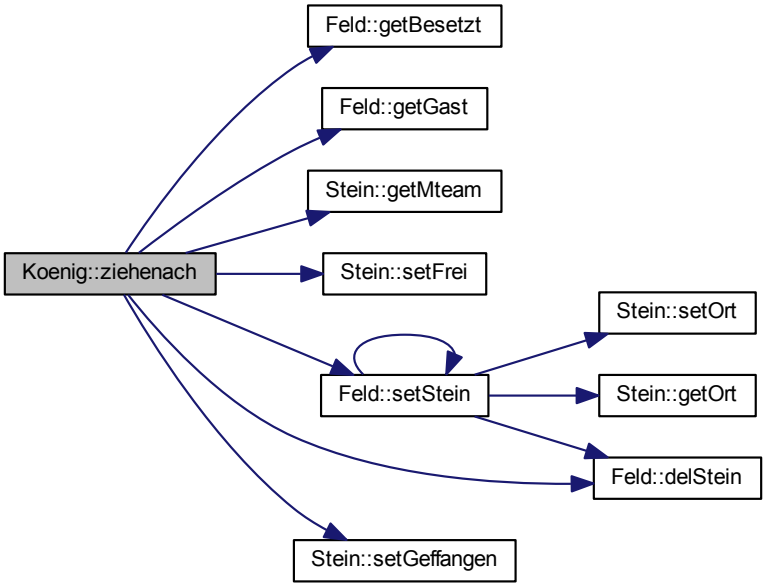


4.4.2.2 bool Koenig::ziehenach (Feld * *ziehe*) [override], [virtual]

Implementiert den Startplatz des Königs.

Erneute Implementation von Stein.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



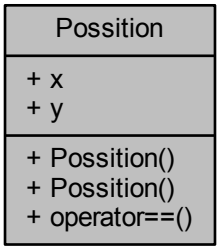
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Koenig.h
- C:/Users/franz/workspace/distanz/src/cpp/Koenig.cpp

4.5 Possition Strukturreferenz

`#include <Possition.h>`

Zusammengehörigkeiten von Possition:



Öffentliche Methoden

- Possition (short int x, short int y)
- Possition ()
- bool operator== (const Possition &p) const

Öffentliche Attribute

- short int x
- short int y

4.5.1 Ausführliche Beschreibung

struct Position

4.5.2 Beschreibung der Konstruktoren und Destruktoren

4.5.2.1 Possition::Possition (short int *x*, short int *y*) [inline]

4.5.2.2 Possition::Possition ( ) [inline]

4.5.3 Dokumentation der Elementfunktionen

4.5.3.1 bool Possition::operator== (const Possition & *p*) const [inline]

4.5.4 Dokumentation der Datenelemente

4.5.4.1 short int Possition::x

4.5.4.2 short int Possition::y

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

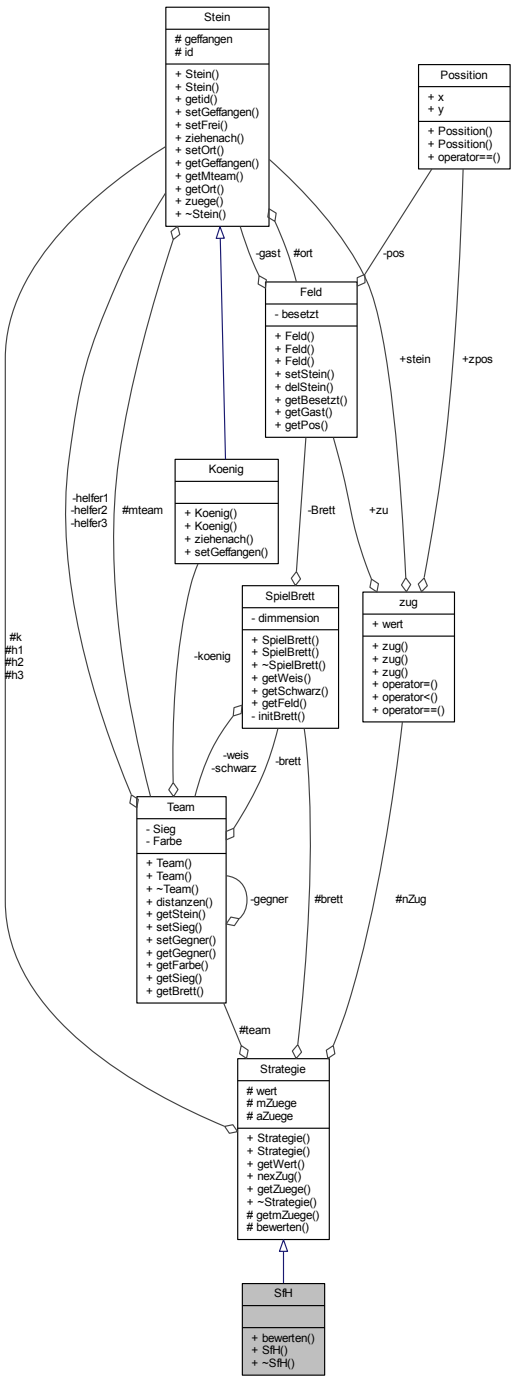
- Possition.h

4.6 SfH Klassenreferenz

```
#include <SfH.h>
```

Klassendiagramm für SfH:

Zusammengehörigkeiten von SfH:



Öffentliche Methoden

- virtual void bewerten () override
- SfH (Team &team, SpielBrett &b)
- virtual ~SfH ()

Weitere Geerbte Elemente

4.6.1 Ausführliche Beschreibung

class SfH (Strategie fange Helfer) implementiert die Methode bewerten();

Diese Strategie sorgt dafür, dass die gegnerischen Helfer festgesetzt/gefangen werden. Ein festgesetzter/gefangener Helfer stellt insofern keine Bedrohung mehr dar, bis er wieder vom Koenig befreit wird. Dies gilt es durch andere Strategien zu verhindern.

Parameter

&team	Referenz auf Instanz von Team
&b	Referenz auf Instanz von SpielBrett

4.6.2 Beschreibung der Konstruktoren und Destruktoren

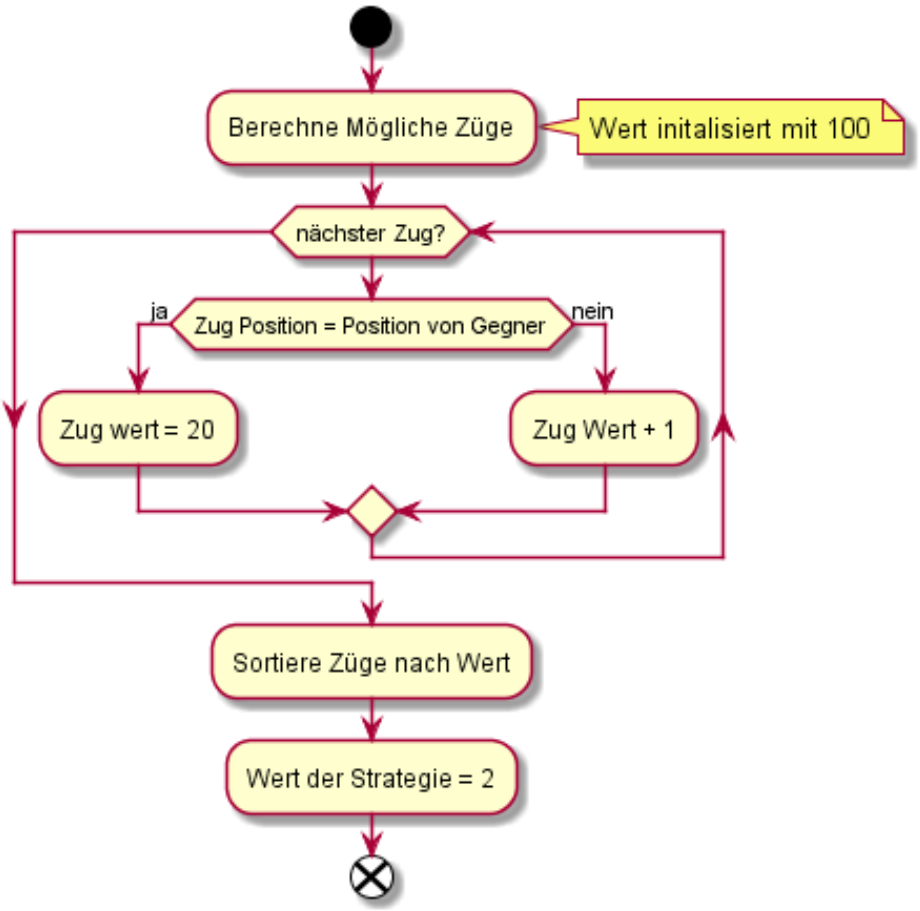
4.6.2.1 SfH::SfH (Team & *team,* SpielBrett & *b*)

4.6.2.2 SfH::~SfH ( ) [virtual]

4.6.3 Dokumentation der Elementfunktionen

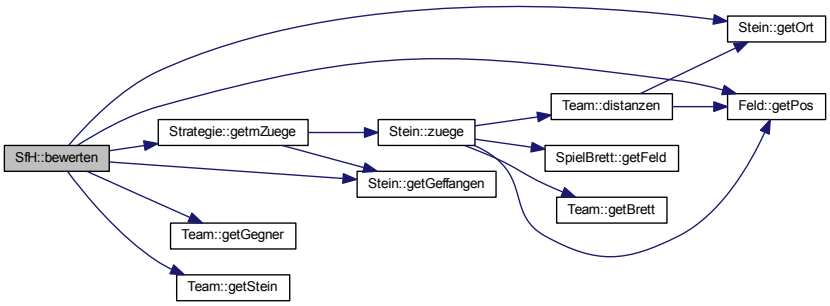
4.6.3.1 void SfH::bewerten ( ) [override], [virtual]

bewerten() Bewertet mögliche Züge nach der Möglichkeit gegnerische Helfer zu fangen.



Implementiert Strategie.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



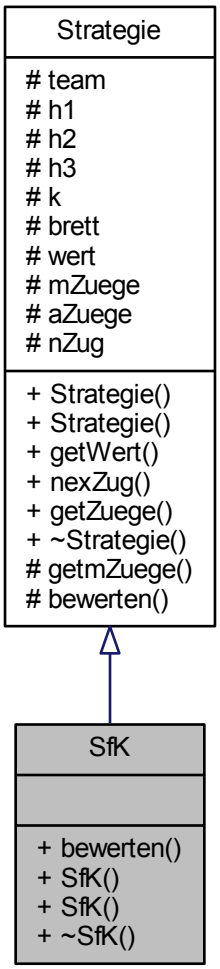
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SfH.h
- C:/Users/franz/workspace/distanz/src/cpp/SfH.cpp

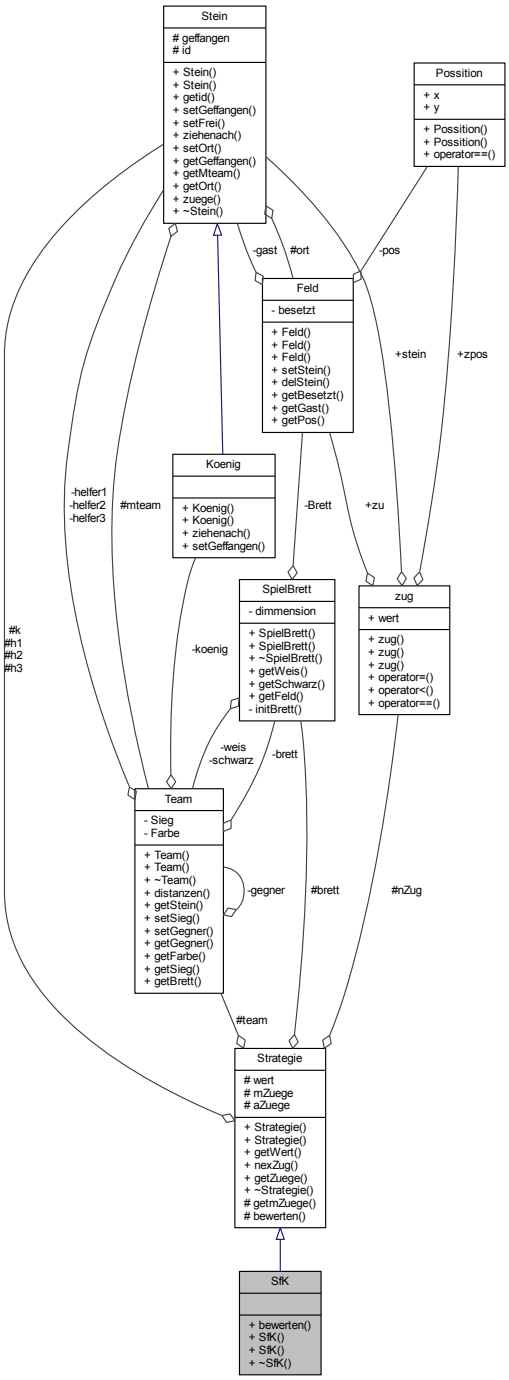
4.7 SfK Klassenreferenz

`#include <SfK.h>`

Klassendiagramm für SfK:



Zusammengehörigkeiten von SfK:



Öffentliche Methoden

- virtual void bewerten () override
- SfK (Team &team, SpielBrett &b)
- SfK ()
- virtual ~SfK ()

Weitere Geerbte Elemente

4.7.1 Ausführliche Beschreibung

class SfK (Strategie fange König) Ist eine Ableitung der abstrakten Klasse Strategie.

Diese Strategie sorgt dafür, dass sich die Spielfiguren dem gegnerischen König nähern, um in festsetzen/gefangen nehmen zu können. Ein festgesetzter/gefangener König bedeutet das Spielende. Ein Sieg wird erzielt, sobald der gegnerische König festgesetzt/gefangen ist.

Überschreibt/implementiert die Methode bewerten();

Parameter

&team	Referenz auf Instanz von Team
&b	Referenz auf Instanz von SpielBrett

4.7.2 Beschreibung der Konstruktoren und Destruktoren

4.7.2.1 SfK::SfK (Team & *team,* SpielBrett & *b*)

4.7.2.2 SfK::SfK ( )

4.7.2.3 SfK::~SfK ( ) [virtual]

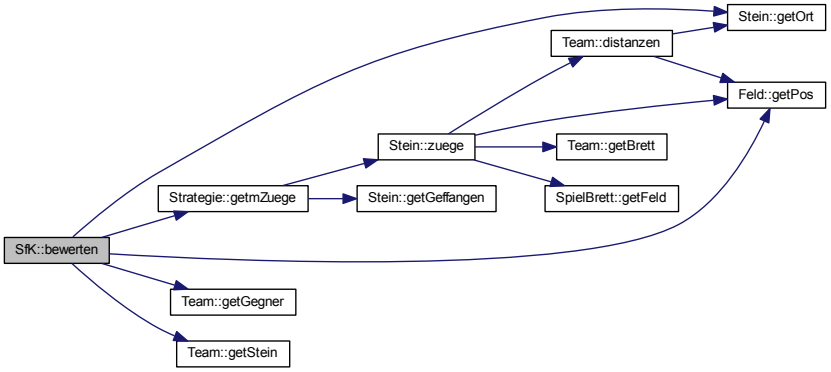
4.7.3 Dokumentation der Elementfunktionen

4.7.3.1 void SfK::bewerten ( ) [override], [virtual]

bewerten() Bewertet mögliche Züge nach der Möglichkeit gegnerischen König zu fangen.

Implementiert Strategie.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



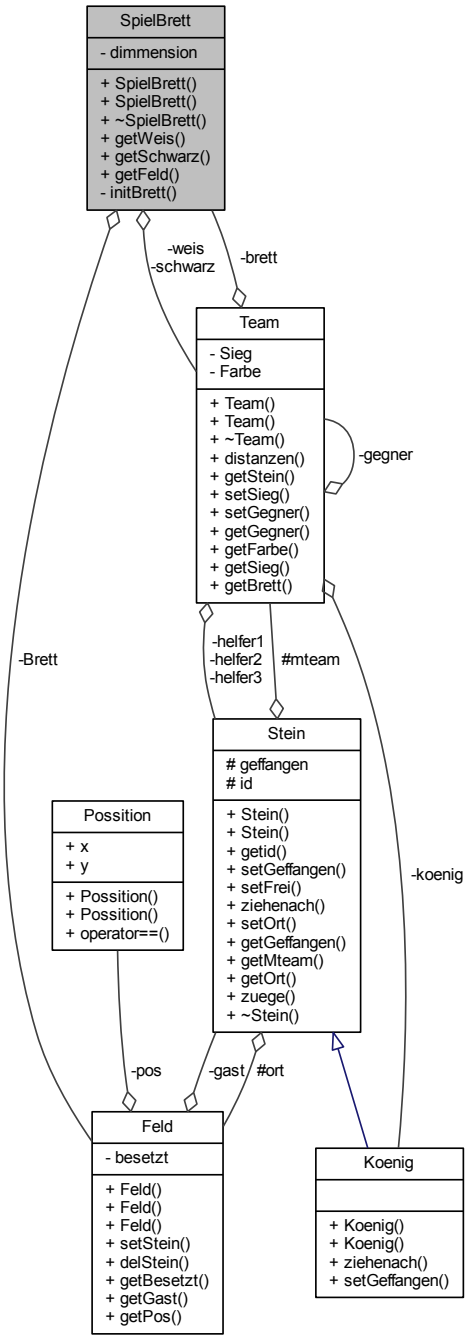
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SfK.h
- C:/Users/franz/workspace/distanz/src/cpp/SfK.cpp

4.8 SpielBrett Klassenreferenz

`#include <SpielBrett.h>`

Zusammengehörigkeiten von SpielBrett:



Öffentliche Methoden

- SpielBrett ()
- SpielBrett (const SpielBrett &sb)
- ~SpielBrett ()
- Team * getWeis () const
- Team * getSchwarz () const
- Feld * getFeld (int x, int y) const

Private Methoden

- void initBrett ()

Private Attribute

- Feld ** Brett =nullptr
- Team * schwarz =nullptr
- Team * weis =nullptr

Statische, private Attribute

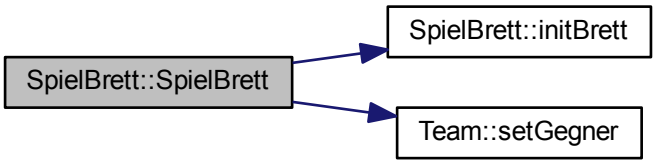
- static const short int dimmension = 8

4.8.1 Beschreibung der Konstruktoren und Destruktoren

4.8.1.1 SpielBrett::SpielBrett ( )

initBrett() Erzeugt das 8x8 großes Feld.

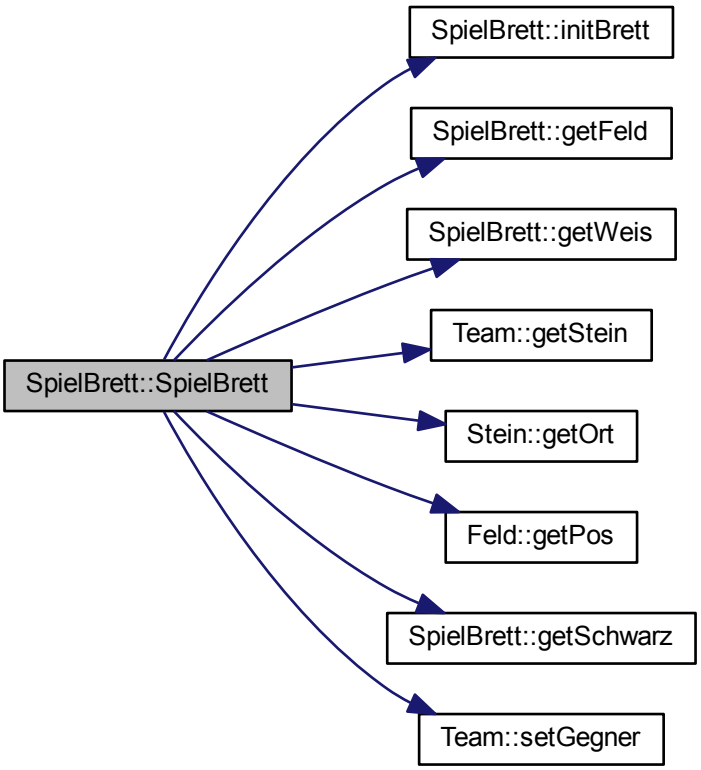
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.8.1.2 SpielBrett::SpielBrett (const SpielBrett & *sb*)

SpielBrett() Beinhaltet die Startaufstellung.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.8.1.3 SpielBrett::∼SpielBrett ( )

SpielBrett (const SpielBrett &sb) Verweißt auf die Pointer, der einzelnen Spielsteine.

4.8.2 Dokumentation der Elementfunktionen

4.8.2.1 Feld ∗ SpielBrett::getFeld (int *x*, int *y*) const

getSchwarz() Kennzeichnet die schwarzen Steine.

Rückgabe

schwarz

4.8.2.2 Team ∗ SpielBrett::getSchwarz ( ) const

getWeis() Kennzeichnet die weißen Steine.

Rückgabe

weiß

4.8.2.3 Team * SpielBrett::getWeis ( ) const

~SpielBrett Destruktor des Spiels

4.8.2.4 void SpielBrett::initBrett ( ) [private]

4.8.3 Dokumentation der Datenelemente

4.8.3.1 Feld** SpielBrett::Brett =nullptr [private]

4.8.3.2 const short int SpielBrett::dimmension = 8 [static], [private]

4.8.3.3 Team* SpielBrett::schwarz =nullptr [private]

4.8.3.4 Team * SpielBrett::weis =nullptr [private]

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

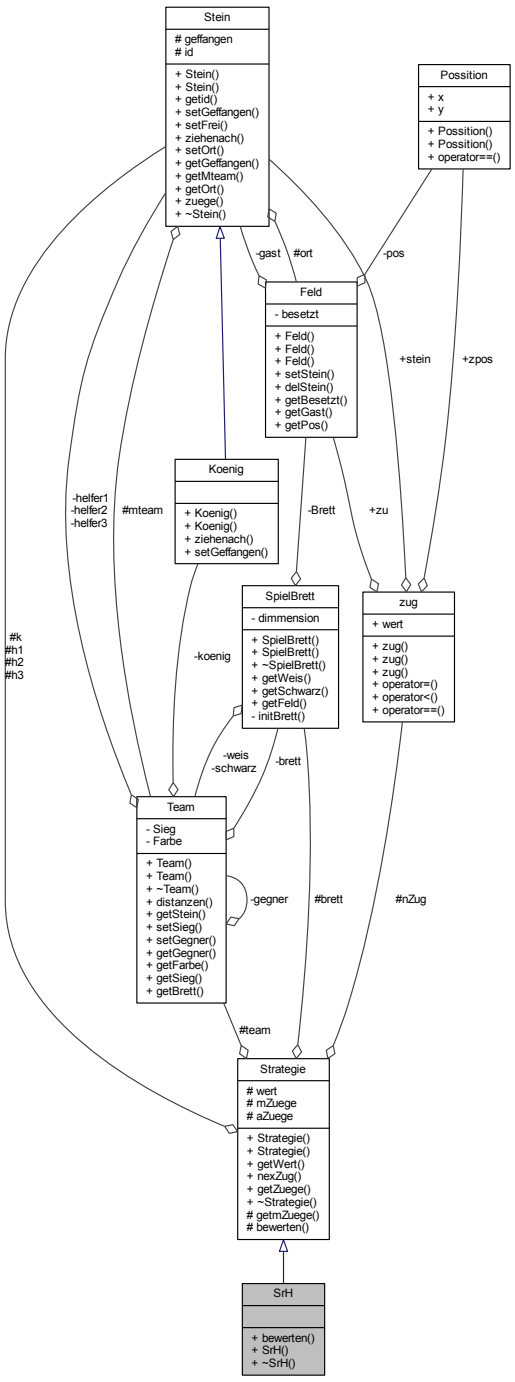
- SpielBrett.h
- C:/Users/franz/workspace/distanz/src/cpp/SpielBrett.cpp

4.9 SrH Klassenreferenz

```
#include <SrH.h>
```

Klassendiagramm für SrH:

Zusammengehörigkeiten von SrH:



Öffentliche Methoden

- virtual void bewerten () override
- SrH (Team &team, SpielBrett &b)
- virtual ~SrH ()

Weitere Geerbte Elemente

4.9.1 Ausführliche Beschreibung

class SrH (Strategie rette Helfer) Ist eine Ableitung der abstrakten Klasse Strategie.

Diese Strategie sorgt dafuer, dass der Koenig teameigene festgesetzte/gefangene Helfer befreit. Dies tut er allerdings nach Moeglichkeit erst dann, wenn sie sich auch in unmittelbarer Umgebung befinden, da der Koenig selber eine sehr defensive Rolle im Spielverlauf einnimmt.

Ueberschreibt/implementiert die Methode bewerten();

Parameter

&team	Referenz auf Instanz von Team
&b	Referenz auf Instanz von SpielBrett

4.9.2 Beschreibung der Konstruktoren und Destruktoren

4.9.2.1 SrH::SrH (Team & *team,* SpielBrett & *b*)

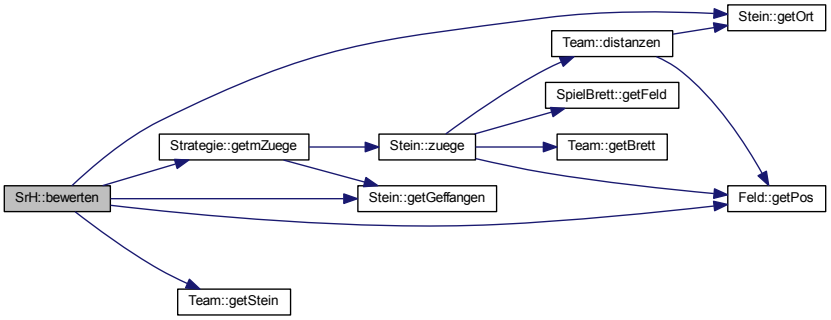
4.9.2.2 SrH::~SrH ( ) [virtual]

4.9.3 Dokumentation der Elementfunktionen

4.9.3.1 void SrH::bewerten ( ) [override], [virtual]

Implementiert Strategie.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



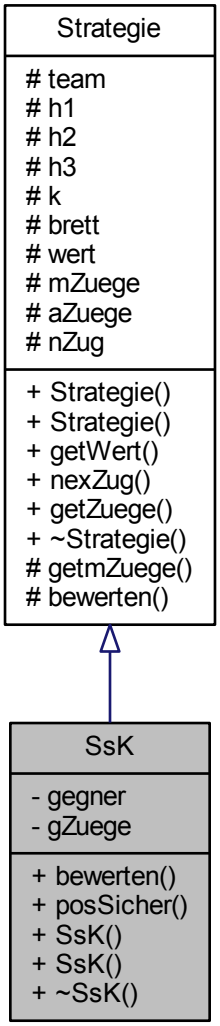
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SrH.h
- C:/Users/franz/workspace/distanz/src/cpp/SrH.cpp

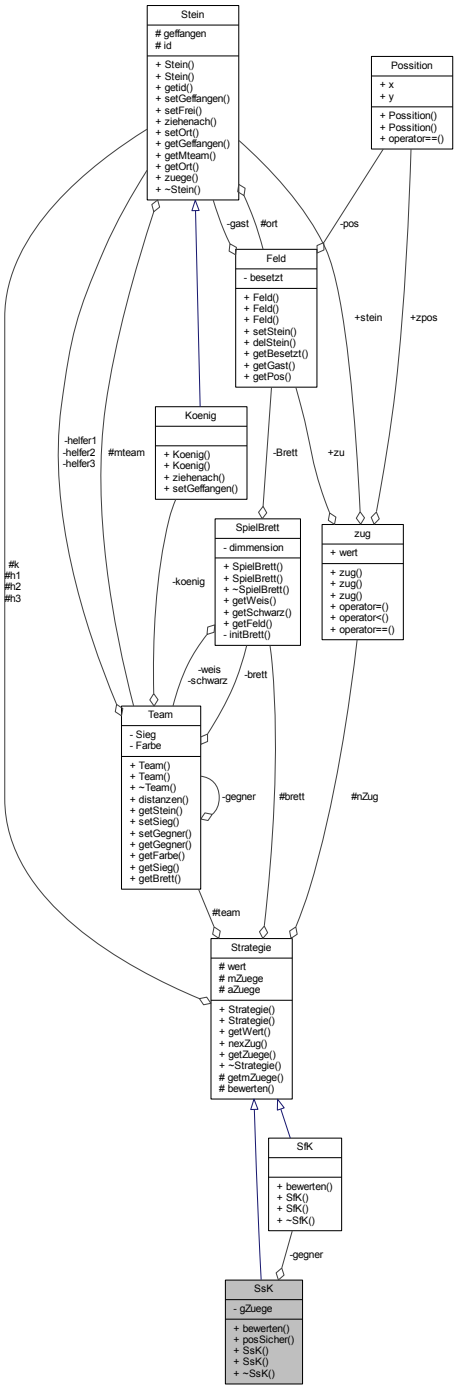
4.10 SsK Klassenreferenz

`#include <SsK.h>`

Klassendiagramm für SsK:



Zusammengehörigkeiten von SsK:



Öffentliche Methoden

- virtual void bewerten () override
- bool posSicher (Possition p)
- SsK (Team &team, SpielBrett &b)
- SsK ()
- virtual ~SsK ()

Private Attribute

- SfK gegner
- std::vector< zug > gZuege

Weitere Geerbte Elemente

4.10.1 Ausführliche Beschreibung

class SsK (Strategie schuetze Koenig) Ist eine Ableitung der abstrakten Klasse Strategie.

Diese Strategie sorgt dafuer, dass der teameigene Koenig vor festsetzen/gefangen nehmen durch feindliche Spielfiguren geschuetzt wird. Zu beobachten ist hierbei das fangen von gegnerischen Spielfiguren, sobald sie dem König zu nahe kommen. Auch der Koenig selber nimmt ein sehr defensives Verhalten an und haelt sich von den Gegnern fern, um ein fruehzeitiges Ableben zu verhindern.

Ueberschreibt/implementiert die Methode bewerten();

Parameter

&team	Referenz auf Instanz von Team
&b	Referenz auf Instanz von SpielBrett

4.10.2 Beschreibung der Konstruktoren und Destruktoren

4.10.2.1 SsK::SsK (Team & *team,* SpielBrett & *b*)

4.10.2.2 SsK::SsK ( )

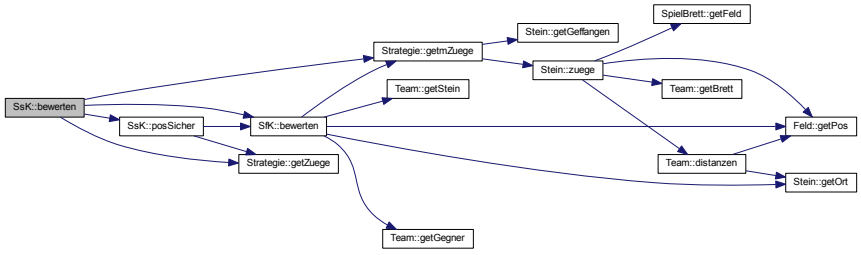
4.10.2.3 SsK::~SsK ( ) [virtual]

4.10.3 Dokumentation der Elementfunktionen

4.10.3.1 void SsK::bewerten ( ) [override], [virtual]

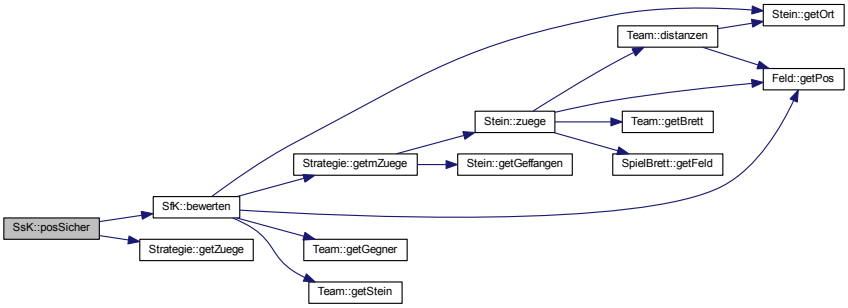
Implementiert Strategie.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.10.3.2 bool SsK::posSicher (Possition *p*)

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.10.4 Dokumentation der Datenelemente

4.10.4.1 SfK SsK::gegner [private]

4.10.4.2 std::vector<zug> SsK::gZuege [private]

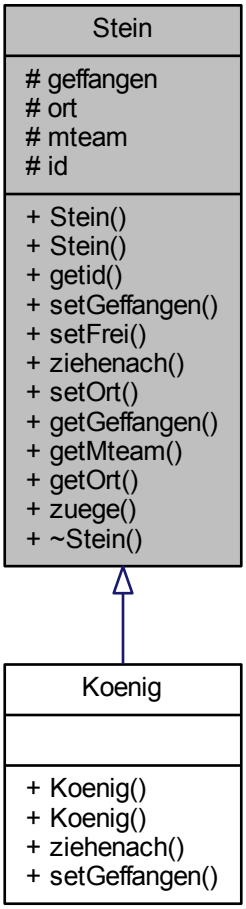
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- SsK.h
- C:/Users/franz/workspace/distanz/src/cpp/SsK.cpp

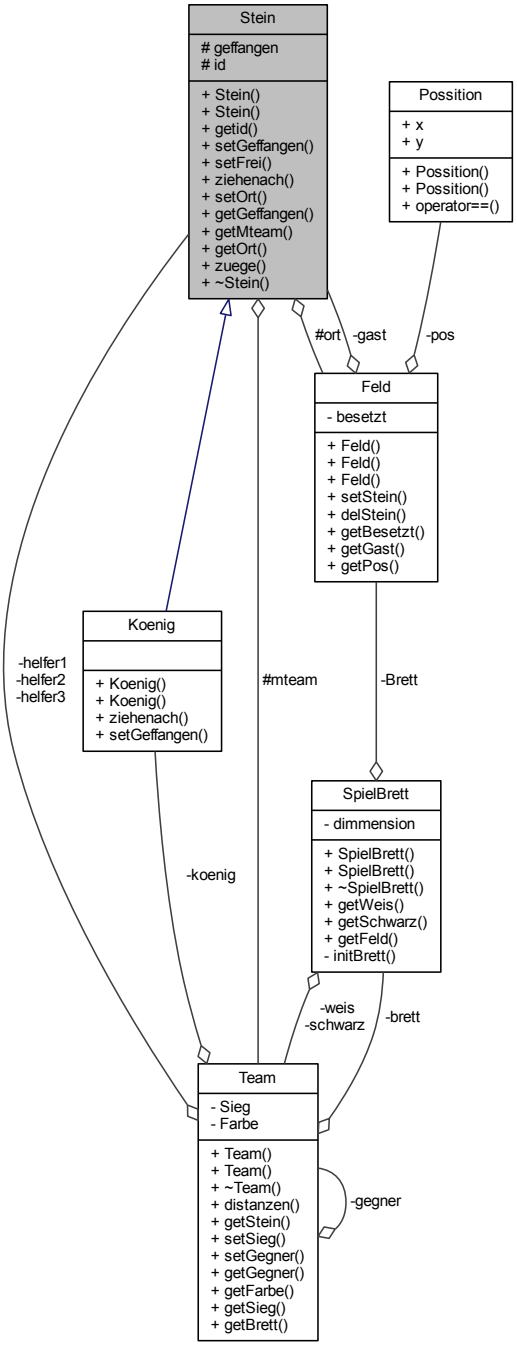
4.11 Stein Klassenreferenz

```
#include <Stein.h>
```

Klassendiagramm für Stein:



Zusammengehörigkeiten von Stein:



Öffentliche Methoden

- Stein ()
- Stein (int id, Feld ∗startplatz, Team ∗mt)
- int getid () const
- virtual void setGeffangen ()
- void setFrei ()

- virtual bool ziehenach (Feld ∗ziehl)
- void setOrt (Feld ∗o)
- bool getGeffangen ()
- Team ∗ getMteam ()
- Feld ∗ getOrt ()
- std::vector< Feld ∗ > zuege ()
- virtual ∼Stein ()=default

Geschützte Attribute

- bool geffangen =false
- Feld ∗ ort =nullptr
- Team ∗ mteam =nullptr
- const int id

4.11.1 Ausführliche Beschreibung

class Stein

Jedes Team besitzt drei Helfer. Sie können sich auf dem Spielfeld bewegen, festgesetzt (gefangen) werden, gegnerische Spielfiguren festsetzen, indem man sie ganz einfach auf das vom Gegner besetzte Feld schickt und in Verbindung mit dem teameigenen KÃ¶nig kÃ¶nnen sie auch selber befreit werden, sollte der Gegner sie gefangen genommen haben. Jede Spielfigur und damit auch jeder Helfer, bekommt bei Spielbeginn einen Platz mittels Pointern zugewiesen. Die Spielfigur-ID und die Spielfeld-ID bestimmen also, welche Spielfigur von welchem Team sich wo im Feld befindet.

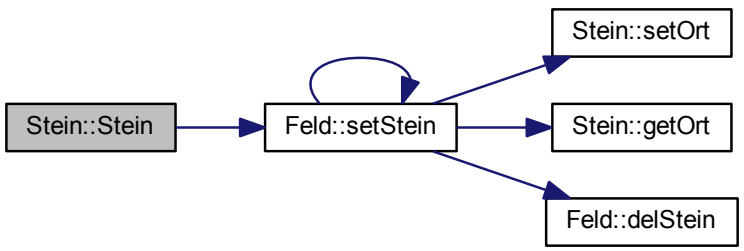
4.11.2 Beschreibung der Konstruktoren und Destruktoren

4.11.2.1 Stein::Stein ( )

Konstruktor

4.11.2.2 Stein::Stein (int *id,* Feld ∗ *startplatz,* Team ∗ *mt*)

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.11.2.3 virtual Stein::~Stein ( ) [virtual], [default]

4.11.3 Dokumentation der Elementfunktionen

4.11.3.1 bool Stein::getGeffangen ( )

getGeffangen() Die Funktion beschreibt, ob der Stein gefangen ist oder nicht.

Rückgabe

the value of gefangen

4.11.3.2 int Stein::getid ( ) const

getid() getid() Diese Funktion sagt aus, ob es sich hierbei um weiß oder schwarz handelt.

Rückgabe

id der Instanz

4.11.3.3 Team * Stein::getMteam ( )

4.11.3.4 Feld * Stein::getOrt ( )

4.11.3.5 void Stein::setFrei ( )

setFrei() Setzt den Stein frei Setzt gefangen -> false

4.11.3.6 void Stein::setGeffangen ( ) [virtual]

setGefangen() Setzt den Stein gefangen. gefangen -> true

Erneute Implementation in Koenig.

4.11.3.7 void Stein::setOrt (Feld * *o*)

4.11.3.8 bool Stein::ziehenach (Feld * *ziehl*) [virtual]

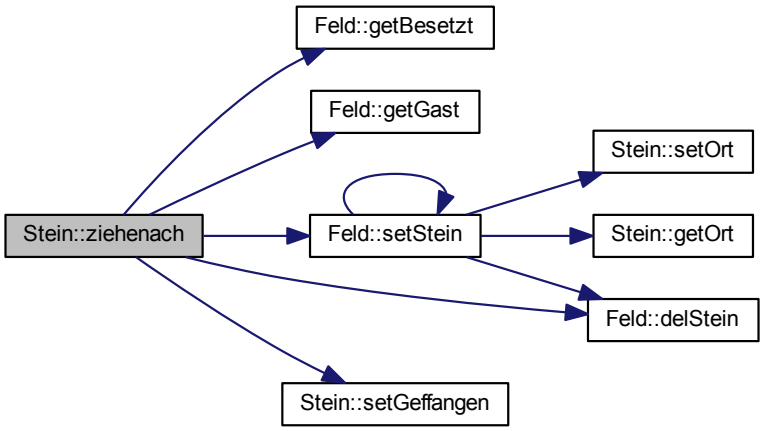
setOrt Rückt auf das übergebene Feld.

Parameter

[Feld]	gibt die neue Position an

Erneute Implementation in Koenig.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



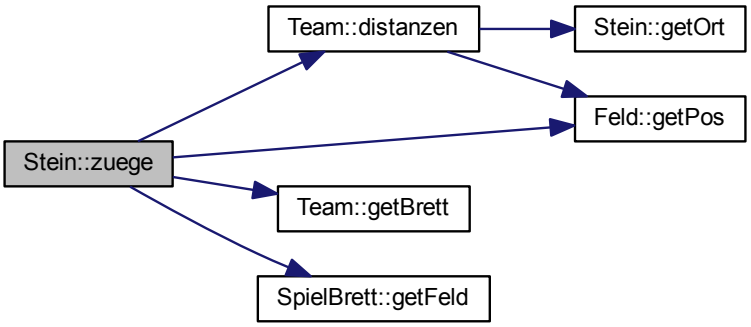
4.11.3.9 std::vector< Feld * > Stein::zuege ( )

zuege() Die Funktion Zuege ermittelt alle möglichen Züge und gibt diese als Vector zurück.

Rückgabe

Feld zue

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.11.4 Dokumentation der Datenelemente

4.11.4.1 bool Stein::geffangen =false [protected]

4.11.4.2 const int Stein::id [protected]

4.11.4.3 Team* Stein::mteam =nullptr [protected]

4.11.4.4 Feld* Stein::ort =nullptr [protected]

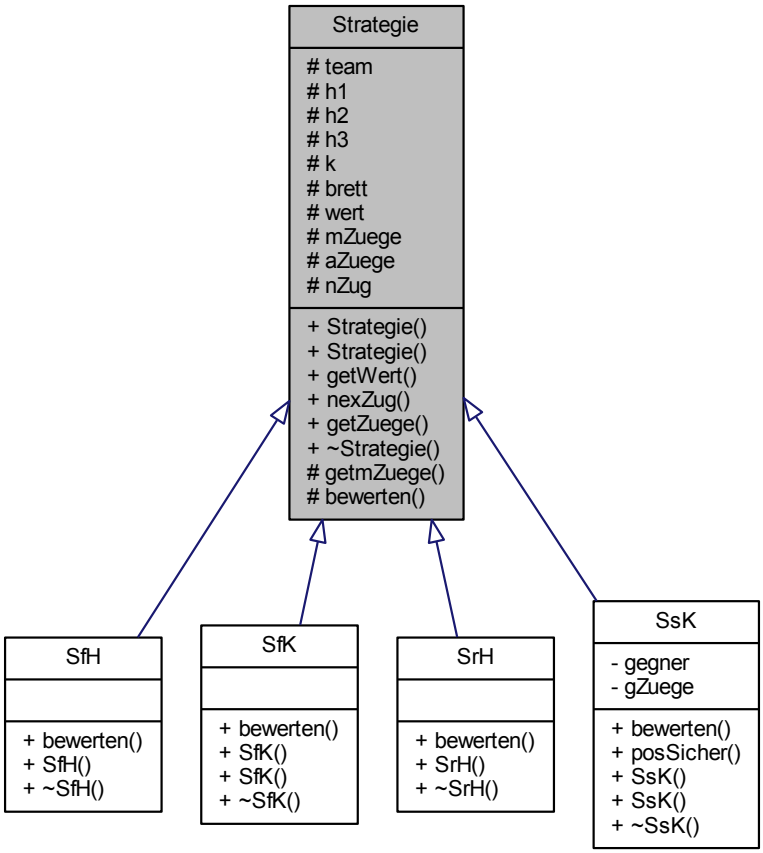
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Stein.h
- C:/Users/franz/workspace/distanz/src/cpp/Stein.cpp

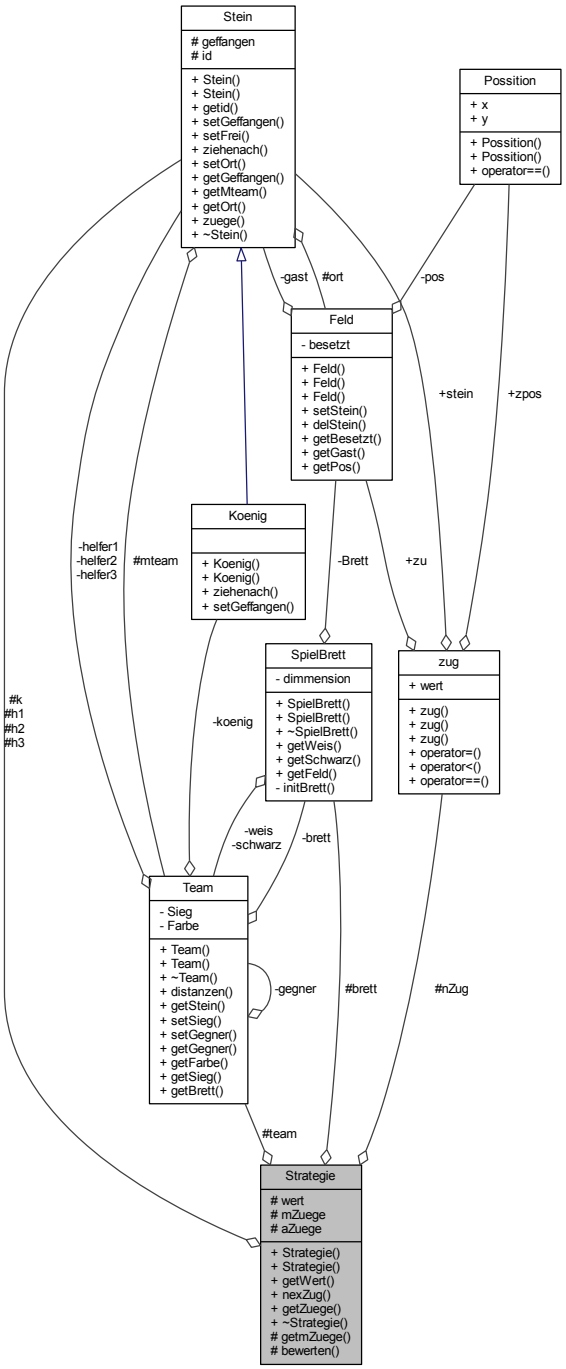
4.12 Strategie Klassenreferenz

`#include <Strategie.h>`

Klassendiagramm für Strategie:



Zusammengehörigkeiten von Strategie:



Öffentliche Methoden

- Strategie (Team &team, SpielBrett &b)
- Strategie ()
- int getWert () const
- zug nexZug ()
- std::vector< zug > getZuege () const
- virtual ~Strategie ()

Geschützte Methoden

- void getmZuege (std::vector< zug > &zuege)
- virtual void bewerten ()=0

Geschützte Attribute

- Team & team
- Stein & h1
- Stein & h2
- Stein & h3
- Stein & k
- SpielBrett & brett
- int wert
- std::vector< zug > mZuege
- std::vector< zug > aZuege
- zug nZug

4.12.1 Ausführliche Beschreibung

class Strategie

Abstrakte Klasse zur Erzeugung von speziellen Zug-Strategien.

Als Strategien sind jene Funktionen gemeint, welche neben der Bewegung im Feld, zusätzlich auch dafür sorgen, dass es zu einer Sieg/Niederlage Situation kommt. Sie stellen die Möglichkeiten dar, welche die Spielfiguren in den jeweiligen Momenten besitzen. Die Bewertung erfolgt in Echtzeit.

Wir programmierten 4 Strategien ein. Jede der 4 Strategien ist eine Vererbung dieser Klasse.

4.12.2 Beschreibung der Konstruktoren und Destruktoren

4.12.2.1 Strategie::Strategie (Team & *team,* SpielBrett & *b*)

4.12.2.2 Strategie::Strategie ( )

4.12.2.3 Strategie::~Strategie ( ) [virtual]

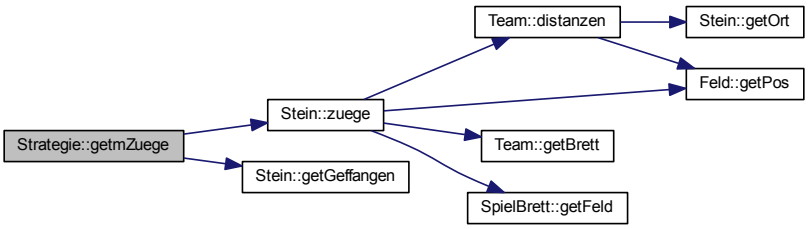
4.12.3 Dokumentation der Elementfunktionen

4.12.3.1 void Strategie::bewerten ( ) [protected], [pure virtual]

Implementiert in SfK, SfH, SsK und SrH.

4.12.3.2 void Strategie::getmZuege (std::vector< zug > & *zuege*) [protected]

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.12.3.3 int Strategie::getWert ( ) const

4.12.3.4 std::vector< zug > Strategie::getZuege ( ) const

4.12.3.5 zug Strategie::nexZug ( )

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

4.12.4 Dokumentation der Datenelemente

4.12.4.1 std::vector<zug> Strategie::aZuege [protected]

4.12.4.2 SpielBrett& Strategie::brett [protected]

4.12.4.3 Stein& Strategie::h1 [protected]

4.12.4.4 Stein & Strategie::h2 [protected]

4.12.4.5 Stein & Strategie::h3 [protected]

4.12.4.6 Stein & Strategie::k [protected]

4.12.4.7 std::vector<zug> Strategie::mZuege [protected]

4.12.4.8 zug Strategie::nZug [protected]

4.12.4.9 Team& Strategie::team [protected]

4.12.4.10 int Strategie::wert [protected]

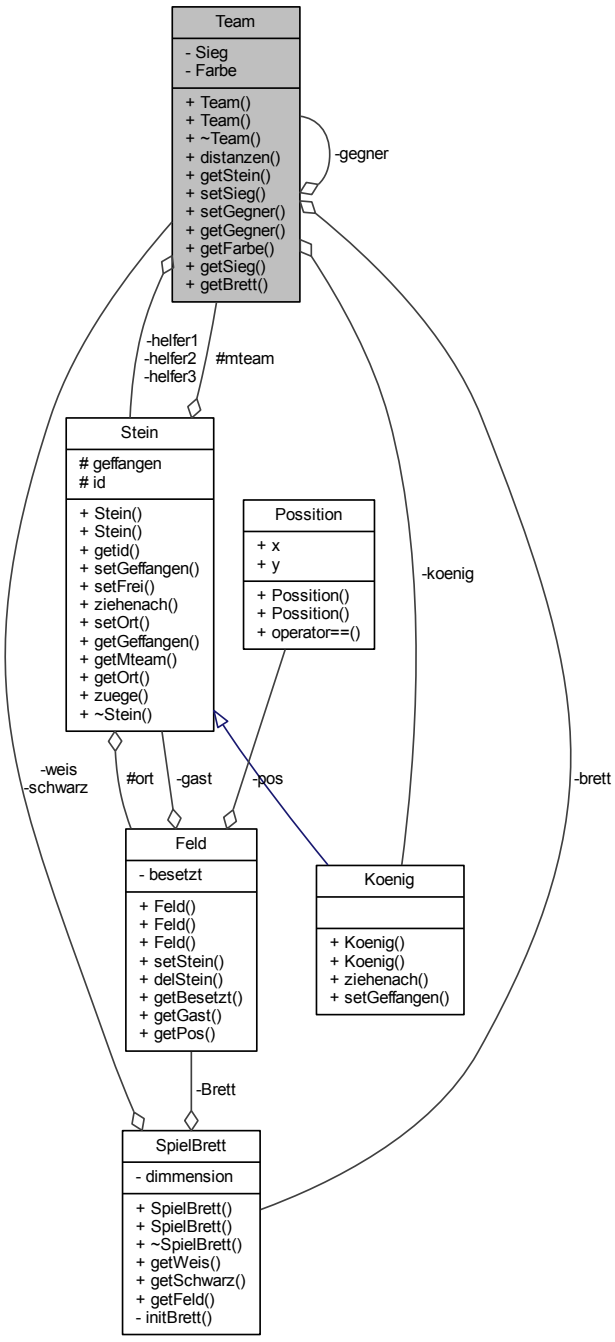
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Strategie.h
- C:/Users/franz/workspace/distanz/src/cpp/Strategie.cpp

4.13 Team Klassenreferenz

`#include <Team.h>`

Zusammengehörigkeiten von Team:



Öffentliche Methoden

- Team (SpielBrett ∗br, bool f, Feld ∗s1, Feld ∗s2, Feld ∗s3, Feld ∗k, Team ∗g)
- Team ()=default
- virtual ∼Team ()
- void distanzen (const Stein &anfrage, int ∗arr)
- Stein & getStein (int id) const

- void setSieg (bool new_var)
- void setGegner (Team ∗new_var)
- Team ∗ getGegner () const
- bool getFarbe () const
- bool getSieg ()
- SpielBrett ∗ getBrett () const

Private Attribute

- Stein ∗ helfer1 =nullptr
- Stein ∗ helfer2 =nullptr
- Stein ∗ helfer3 =nullptr
- Koenig ∗ koenig =nullptr
- bool Sieg =false
- Team ∗ gegner =nullptr
- SpielBrett ∗ brett =nullptr
- bool Farbe =false

4.13.1 Ausführliche Beschreibung

class Team

4.13.2 Beschreibung der Konstruktoren und Destruktoren

4.13.2.1 Team::Team (SpielBrett ∗ *br,* bool *f,* Feld ∗ *s1,* Feld ∗ *s2,* Feld ∗ *s3,* Feld ∗ *k,* Team ∗ *g* = `nullptr`)

Erzeugt Team.

4.13.2.2 Team::Team ( ) `[default]`

4.13.2.3 Team::∼Team ( ) `[virtual]`

4.13.3 Dokumentation der Elementfunktionen

4.13.3.1 void Team::distanzen (const Stein & *anfrage,* int ∗ *arr*)

distanzen() Trägt x und y Distanzen der "Anderen" Steine in einem Array ein. Array muss 6 Felder besitzen und vom Typ Integer sein.

Parameter

`in`	*&anfrage*	: Stein, [out] ∗arr : int array[6]

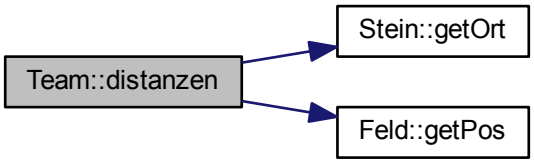
Rückgabe

unsigned short

Parameter

anfrage	

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



4.13.3.2 SpielBrett * Team::getBrett ( ) const

4.13.3.3 bool Team::getFarbe ( ) const

4.13.3.4 Team * Team::getGegner ( ) const

Gibt Pointer auf Gegnerisches Team aus.

4.13.3.5 bool Team::getSieg ( )

4.13.3.6 Stein & Team::getStein (int *id*) const

getStein Gibt Referenz auf Stein mit übergebener ID zurück, bei falschen IDŽs wird Referenz auf Koenig zurückgegeben. 1-3 -> Helfer 4 -> Koenig

Parameter

in	*id*	: int

Rückgabe

&Stein

4.13.3.7 void Team::setGegner (Team * *new_var*)

Setze Gegnerisches Team

4.13.3.8 void Team::setSieg (bool *new_var*)

Set the value of Sieg

Parameter

new_var	the new value of Sieg

4.13.4 Dokumentation der Datenelemente

4.13.4.1 SpielBrett* Team::brett =nullptr [private]

4.13.4.2 bool Team::Farbe =false [private]

4.13.4.3 Team* Team::gegner =nullptr [private]

4.13.4.4 Stein* Team::helfer1 =nullptr [private]

4.13.4.5 Stein * Team::helfer2 =nullptr [private]

4.13.4.6 Stein * Team::helfer3 =nullptr [private]

4.13.4.7 Koenig* Team::koenig =nullptr [private]

4.13.4.8 bool Team::Sieg =false [private]

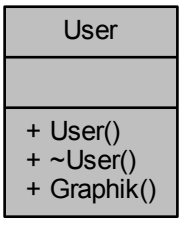
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Team.h
- C:/Users/franz/workspace/distanz/src/cpp/Team.cpp

4.14 User Klassenreferenz

`#include <User.h>`

Zusammengehörigkeiten von User:



Öffentliche Methoden

- User ()
- virtual ~User ()
- void Graphik ()

4.14.1 Ausführliche Beschreibung

class User

4.14.2 Beschreibung der Konstruktoren und Destruktoren

4.14.2.1 User::User ( )

Empty Constructor

4.14.2.2 User::~User ( ) [virtual]

Empty Destructor

4.14.3 Dokumentation der Elementfunktionen

4.14.3.1 void User::Graphik ( )

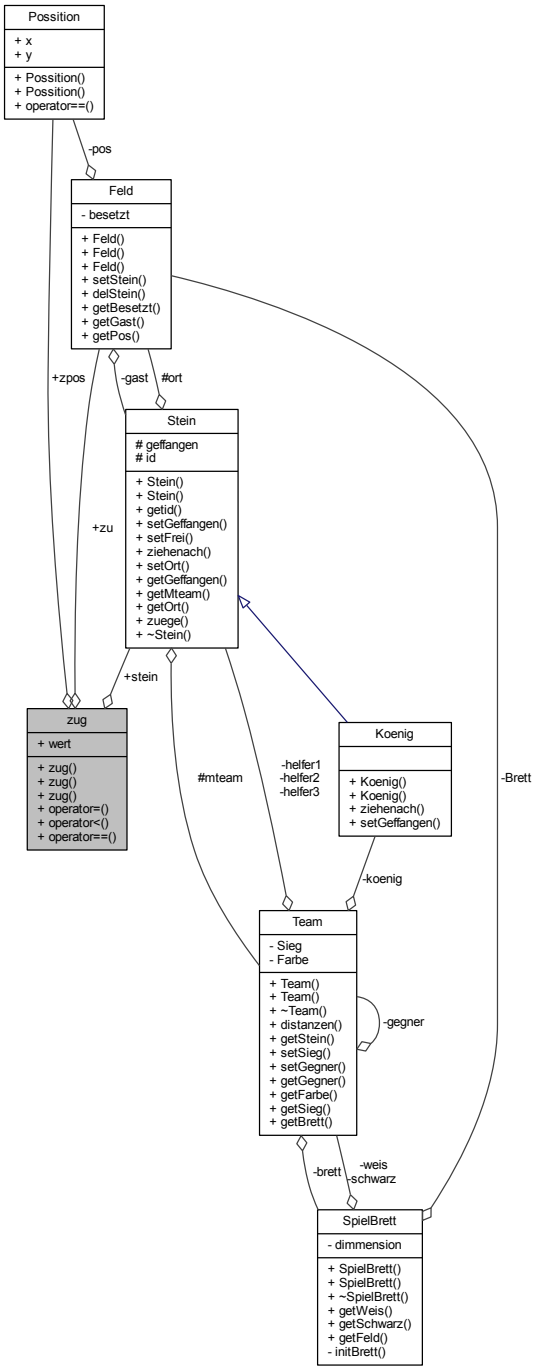
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- User.h
- C:/Users/franz/workspace/distanz/src/cpp/User.cpp

4.15 zug Strukturreferenz

```
#include <zug.h>
```

Zusammengehörigkeiten von zug:



Öffentliche Methoden

- zug ()=default
- zug (Feld ∗z, Stein ∗s)
- zug (const zug &z)
- zug & operator= (const zug &z)
- bool operator< (const zug &z) const
- bool operator== (const zug &z) const

Öffentliche Attribute

- Feld ∗ zu =nullptr
- Stein ∗ stein =nullptr
- int wert =100
- Possition zpos

4.15.1 Ausführliche Beschreibung

struct Zug Daten Struktur die einen Spiel-Zug Symbolysiert.

4.15.2 Beschreibung der Konstruktoren und Destruktoren

4.15.2.1 zug::zug ( ) [default]

4.15.2.2 zug::zug (Feld ∗ *z*, Stein ∗ *s*) [inline]

4.15.2.3 zug::zug (const zug & *z*) [inline]

4.15.3 Dokumentation der Elementfunktionen

4.15.3.1 bool zug::operator< (const zug & *z*) const [inline]

Kleiner als Operator Vergleicht Zuege nach Wertigkeit;

Parameter

z	

Rückgabe

4.15.3.2 zug& zug::operator= (const zug & *z*) [inline]

4.15.3.3 bool zug::operator== (const zug & *z*) const [inline]

Vergleichs-Operator Vergleicht Zuege auf gleiche Ziel-Position

Parameter

z	

Rückgabe

4.15.4 Dokumentation der Datenelemente

4.15.4.1 Stein∗ zug::stein =nullptr

4.15.4.2 int zug::wert =100

4.15.4.3 Possition zug::zpos

4.15.4.4 Feld* zug::zu =nullptr

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- zug.h

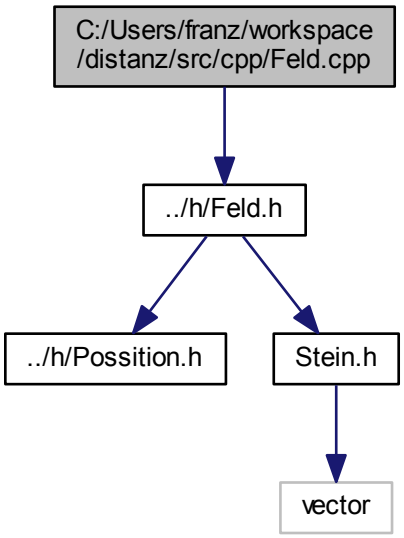
Kapitel 5

Datei-Dokumentation

5.1 C:/Users/franz/workspace/distanz/src/cpp/Feld.cpp-Dateireferenz

```
#include "../h/Feld.h"
```

Include-Abhängigkeitsdiagramm für Feld.cpp:



Makrodefinitionen

- #define STEIN_C

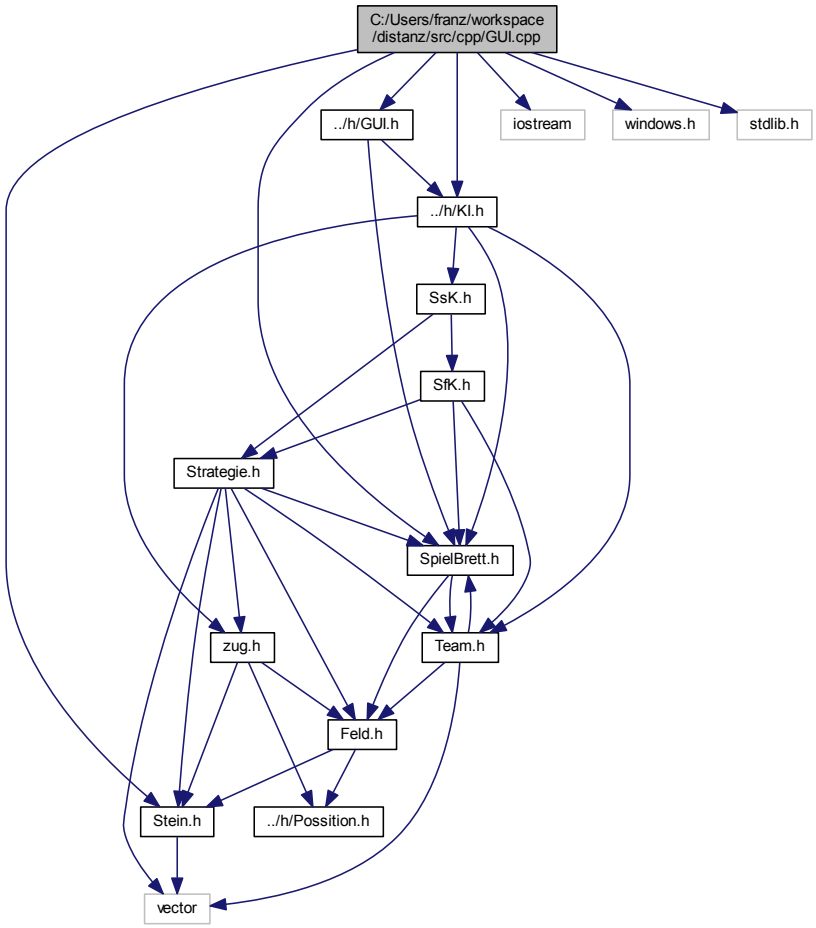
5.1.1 Makro-Dokumentation

5.1.1.1 #define STEIN_C

5.2 C:/Users/franz/workspace/distanz/src/cpp/GUI.cpp-Dateireferenz

```
#include "../h/GUI.h"
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include "../h/KI.h"
#include "../h/Spielbrett.h"
#include "../h/Stein.h"
```

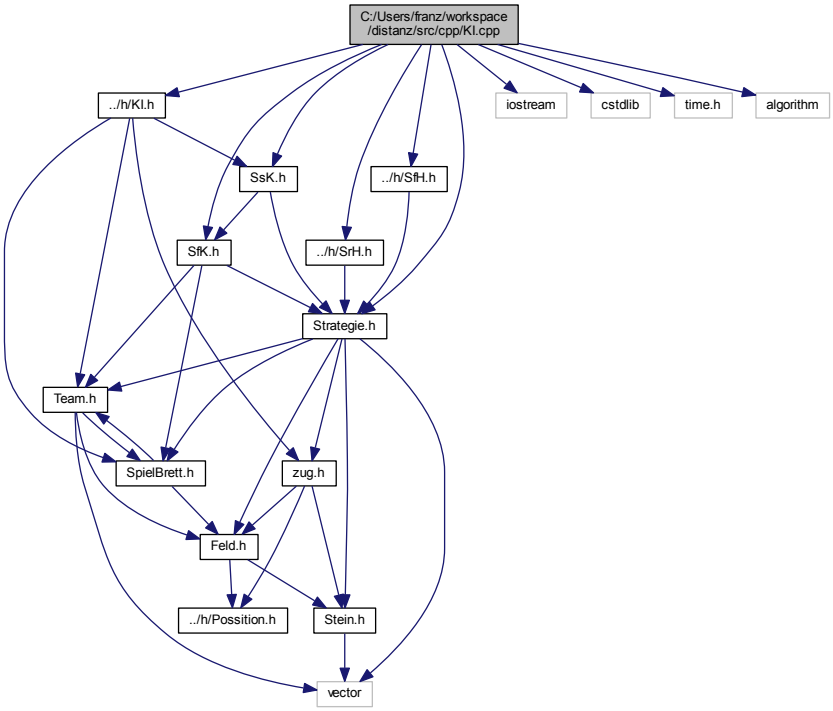
Include-Abhängigkeitsdiagramm für GUI.cpp:



5.3 C:/Users/franz/workspace/distanz/src/cpp/KI.cpp-Dateireferenz

```
#include "../h/KI.h"
#include "../h/SfK.h"
#include "../h/SsK.h"
#include "../h/SfH.h"
#include "../h/SrH.h"
#include "../h/Strategie.h"
#include <iostream>
#include <cstdlib>
#include <time.h>
#include <algorithm>
```

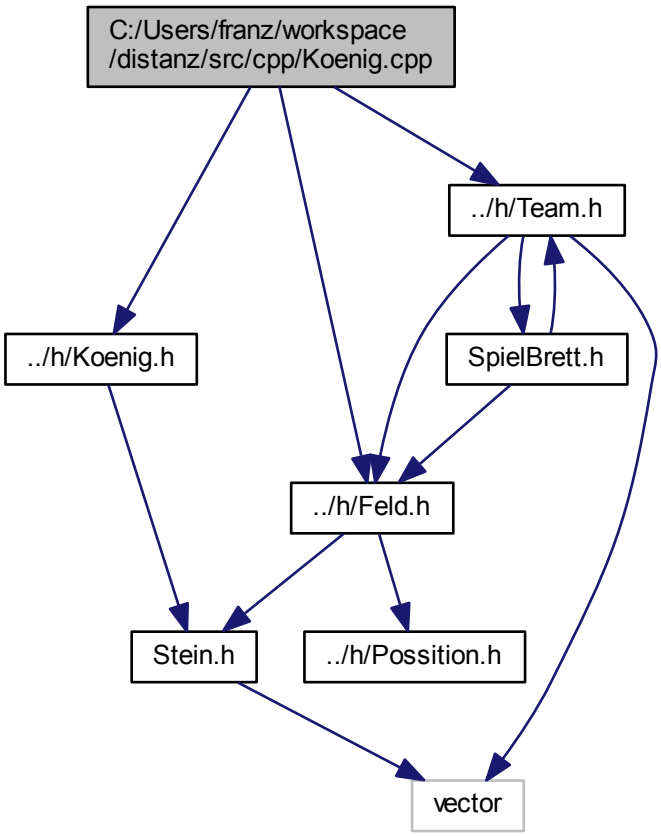
Include-Abhängigkeitsdiagramm für KI.cpp:



5.4 C:/Users/franz/workspace/distanz/src/cpp/Koenig.cpp-Dateireferenz

```
#include "../h/Koenig.h"
#include "../h/Feld.h"
#include "../h/Team.h"
```

Include-Abhängigkeitsdiagramm für Koenig.cpp:



Makrodefinitionen

- #define KOEING_C

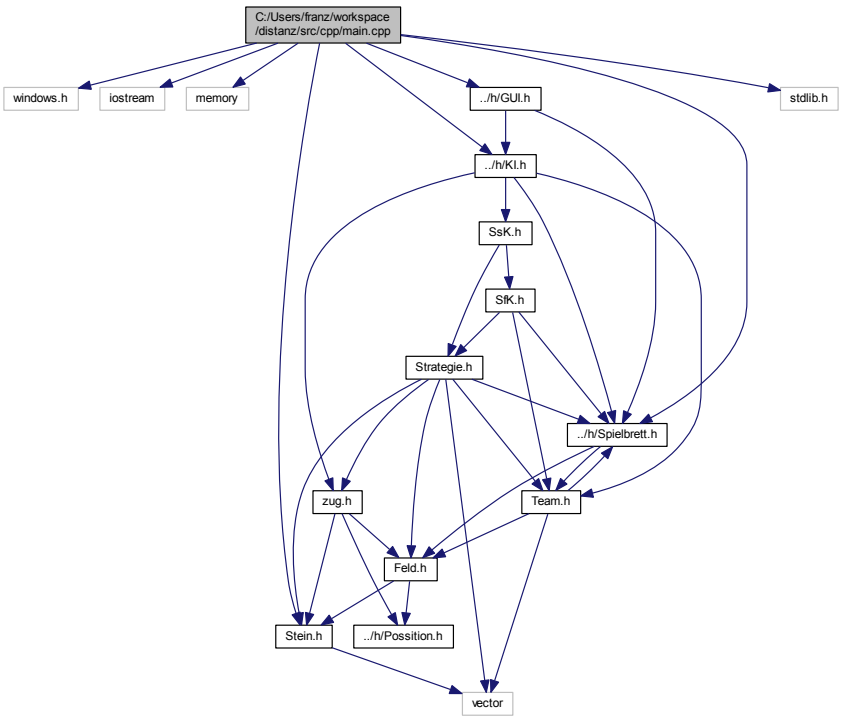
5.4.1 Makro-Dokumentation

5.4.1.1 #define KOEING_C

5.5 C:/Users/franz/workspace/distanz/src/cpp/main.cpp-Dateireferenz

```
#include <windows.h>
#include <iostream>
#include <memory>
#include "../h/Spielbrett.h"
#include <stdlib.h>
#include "../h/KI.h"
#include "../h/GUI.h"
#include "../h/Stein.h"
```

Include-Abhängigkeitsdiagramm für main.cpp:



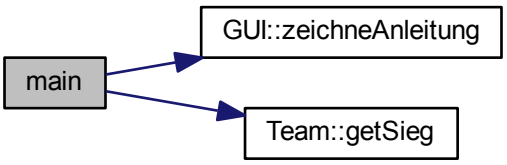
Funktionen

- int main (int _argc, char ∗argv[ ])

5.5.1 Dokumentation der Funktionen

5.5.1.1 int main (int *_argc,* char ∗ *argv[ ]*)

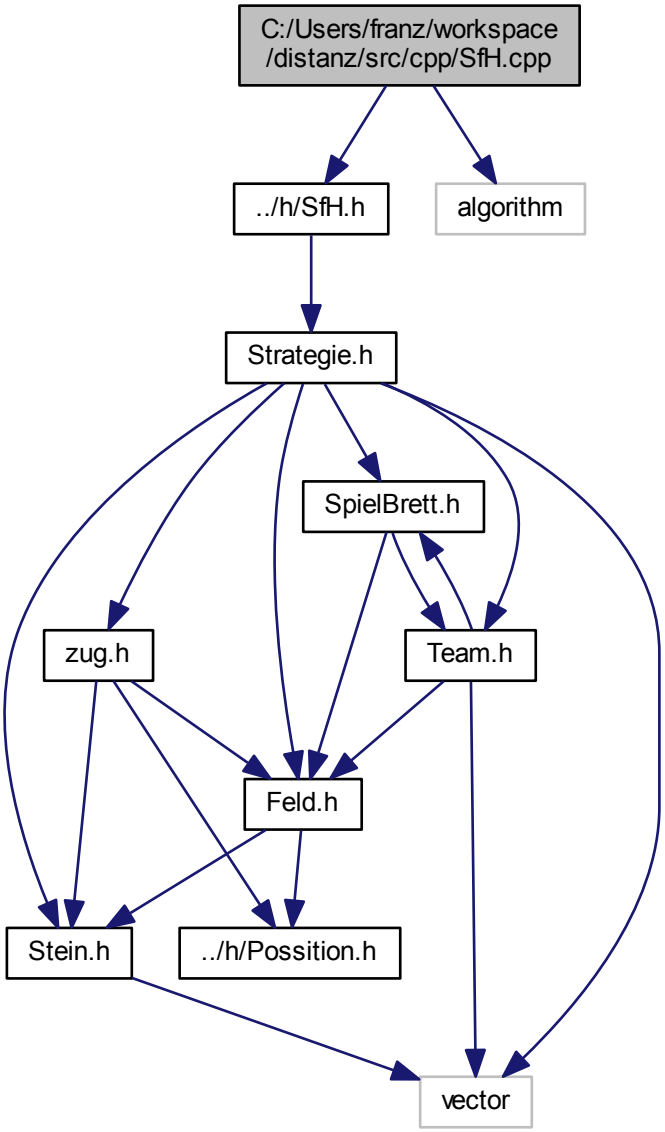
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



5.6 C:/Users/franz/workspace/distanz/src/cpp/SfH.cpp-Dateireferenz

```
#include "../h/SfH.h"
#include <algorithm>
```

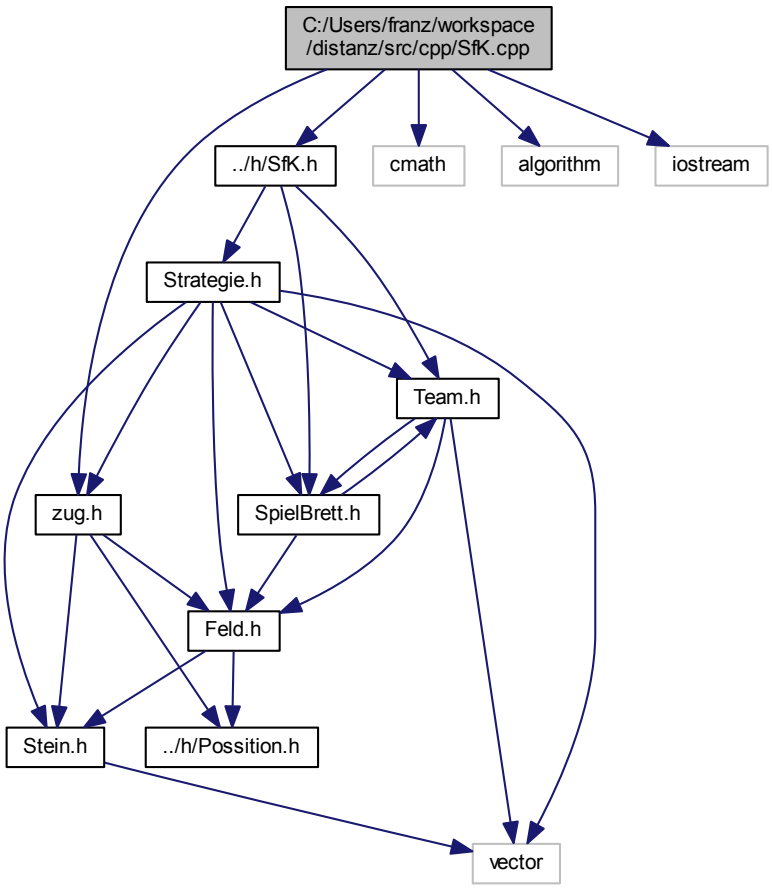
Include-Abhängigkeitsdiagramm für SfH.cpp:



5.7 C:/Users/franz/workspace/distanz/src/cpp/SfK.cpp-Dateireferenz

```
#include "../h/SfK.h"
#include <cmath>
#include <algorithm>
#include "../h/zug.h"
#include <iostream>
```

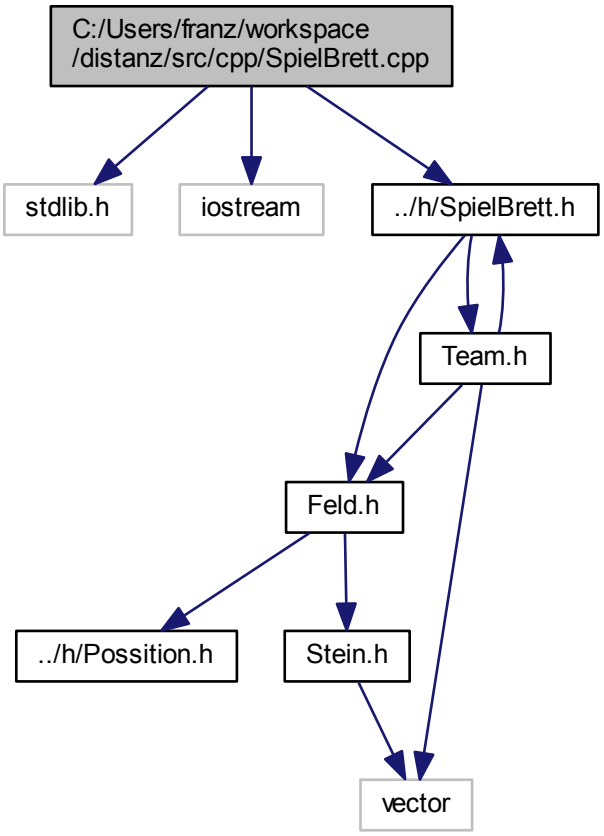
Include-Abhängigkeitsdiagramm für SfK.cpp:



5.8 C:/Users/franz/workspace/distanz/src/cpp/SpielBrett.cpp-Dateireferenz

```
#include <stdlib.h>
#include <iostream>
#include "../h/SpielBrett.h"
```

Include-Abhängigkeitsdiagramm für SpielBrett.cpp:



Makrodefinitionen

- #define SPIELBRETT_C

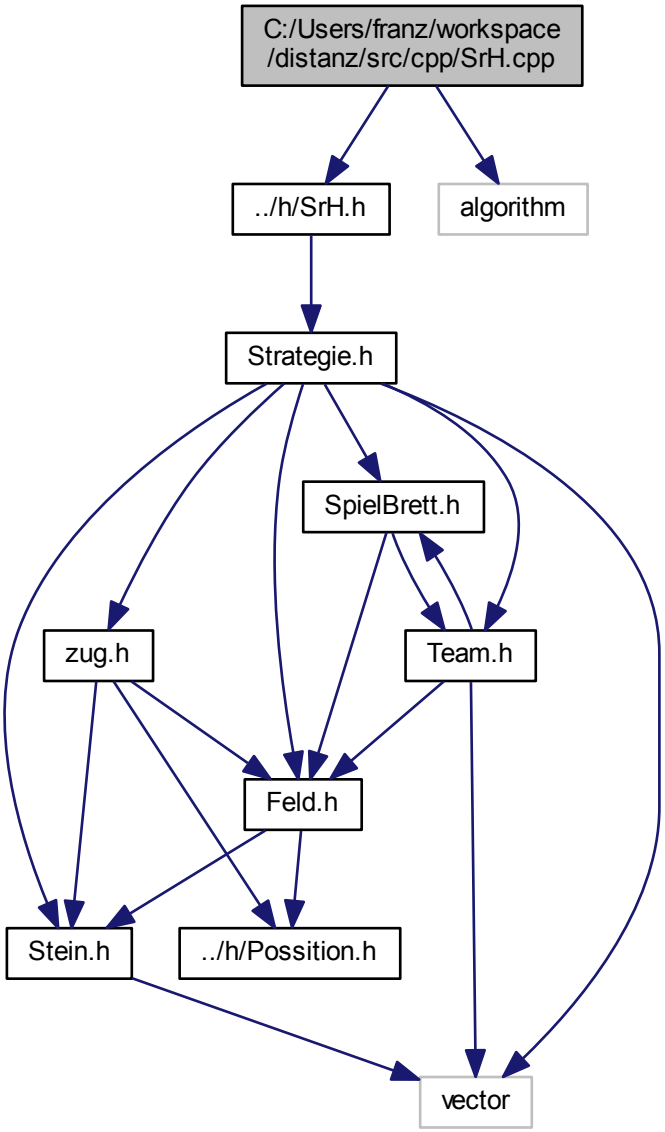
5.8.1 Makro-Dokumentation

5.8.1.1 #define SPIELBRETT_C

5.9 C:/Users/franz/workspace/distanz/src/cpp/SrH.cpp-Dateireferenz

```
#include "../h/SrH.h"
#include <algorithm>
```

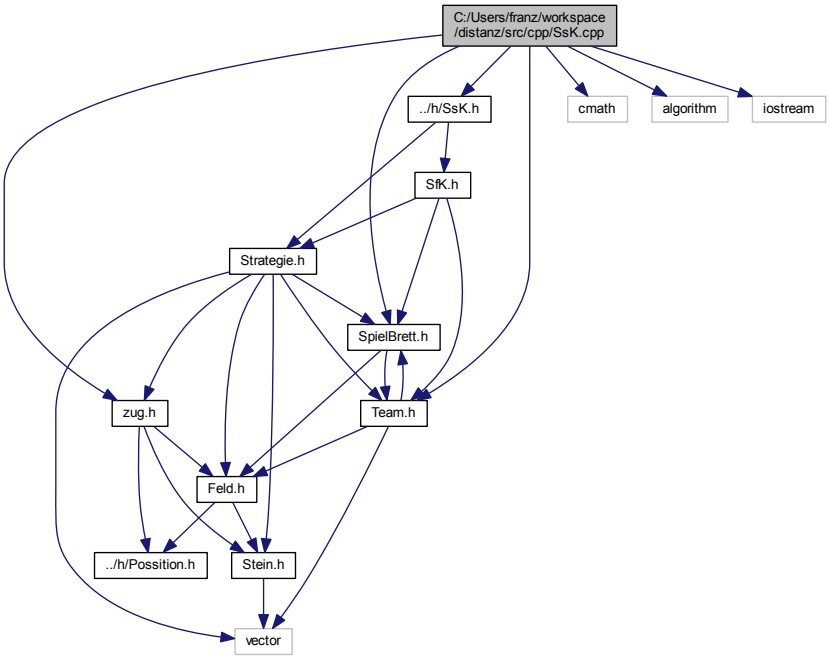
Include-Abhängigkeitsdiagramm für SrH.cpp:



5.10 C:/Users/franz/workspace/distanz/src/cpp/SsK.cpp-Dateireferenz

```
#include "../h/SsK.h"
#include <cmath>
#include <algorithm>
#include "../h/zug.h"
#include "../h/SpielBrett.h"
#include "../h/Team.h"
#include <iostream>
```

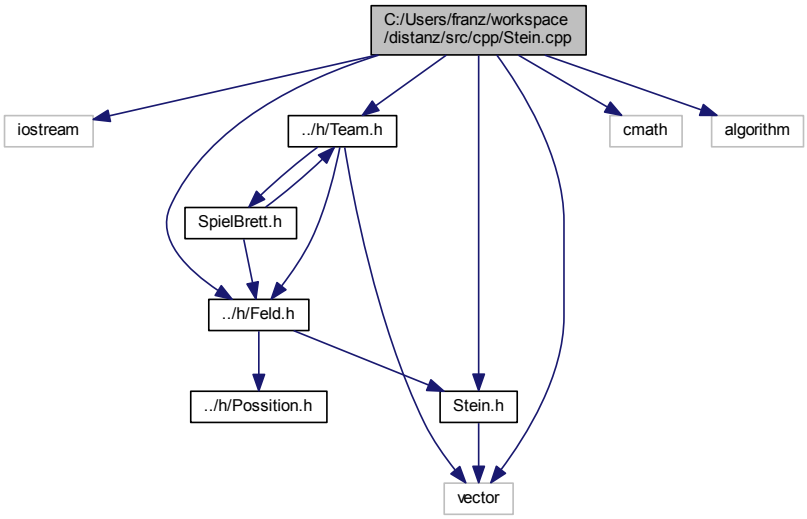
Include-Abhängigkeitsdiagramm für SsK.cpp:



5.11 C:/Users/franz/workspace/distanz/src/cpp/Stein.cpp-Dateireferenz

```
#include <iostream>
#include "../h/Feld.h"
#include "../h/Team.h"
#include "../h/Stein.h"
#include <cmath>
#include <vector>
#include <algorithm>
```

Include-Abhängigkeitsdiagramm für Stein.cpp:



Makrodefinitionen

- #define STEIN_C

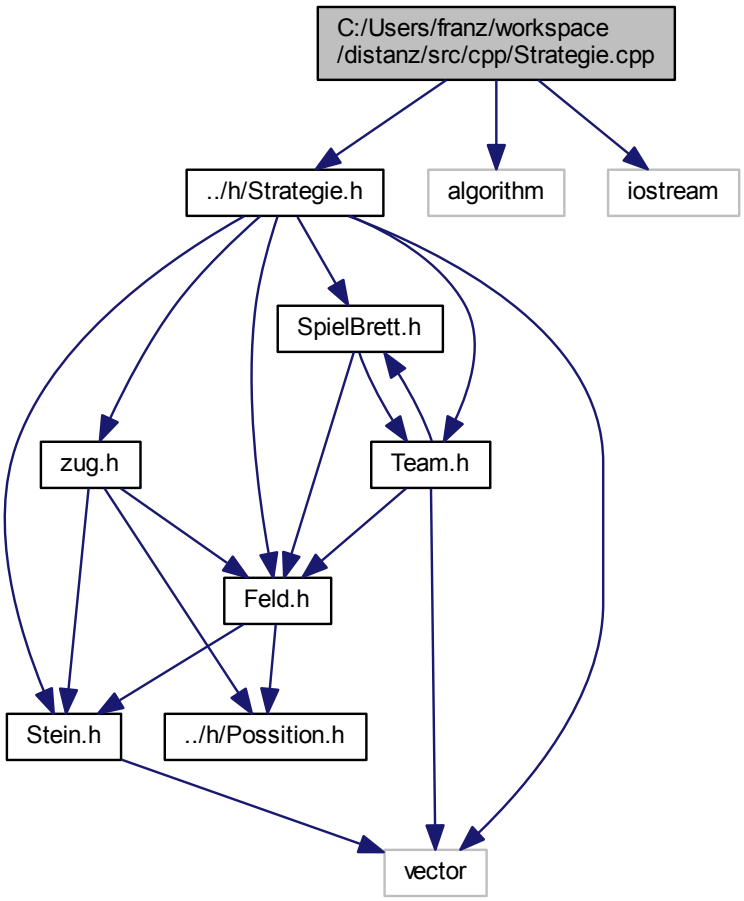
5.11.1 Makro-Dokumentation

5.11.1.1 #define STEIN_C

5.12 C:/Users/franz/workspace/distanz/src/cpp/Strategie.cpp-Dateireferenz

```
#include "../h/Strategie.h"
#include <algorithm>
#include <iostream>
```

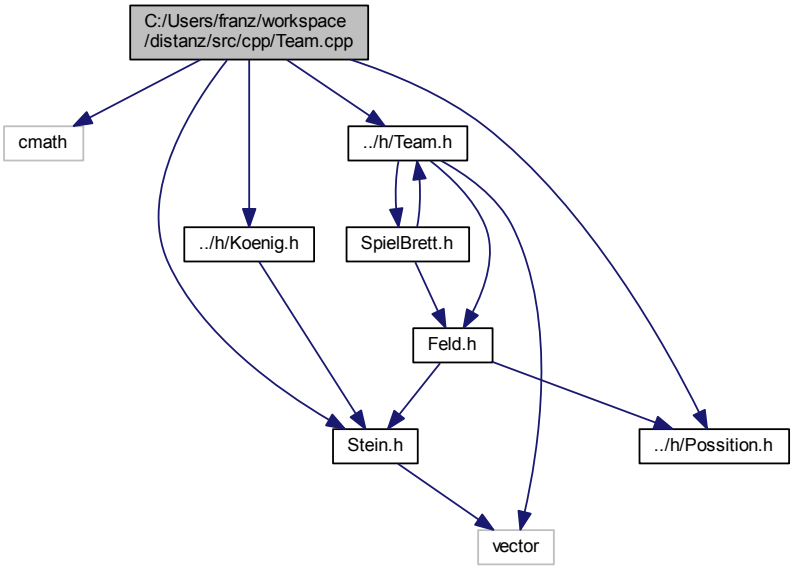
Include-Abhängigkeitsdiagramm für Strategie.cpp:



5.13 C:/Users/franz/workspace/distanz/src/cpp/Team.cpp-Dateireferenz

```
#include <cmath>
#include "../h/Team.h"
#include "../h/Stein.h"
#include "../h/Koenig.h"
#include "../h/Possition.h"
```

Include-Abhängigkeitsdiagramm für Team.cpp:



Makrodefinitionen

- #define TEAM_C

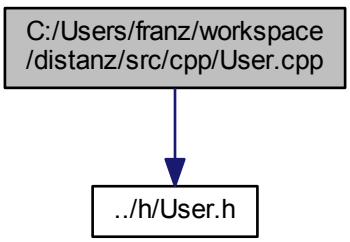
5.13.1 Makro-Dokumentation

5.13.1.1 #define TEAM_C

5.14 C:/Users/franz/workspace/distanz/src/cpp/User.cpp-Dateireferenz

```
#include "../h/User.h"
```

Include-Abhängigkeitsdiagramm für User.cpp:



Makrodefinitionen

- #define USER_C

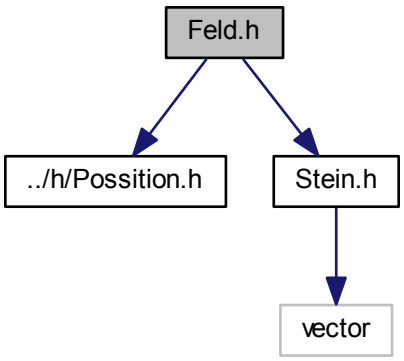
5.14.1 Makro-Dokumentation

5.14.1.1 #define USER_C

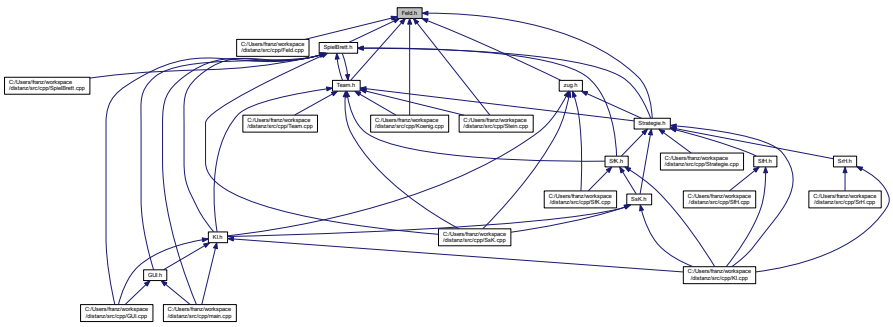
5.15 Feld.h-Dateireferenz

```
#include "../h/Possition.h"
#include "Stein.h"
```

Include-Abhängigkeitsdiagramm für Feld.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



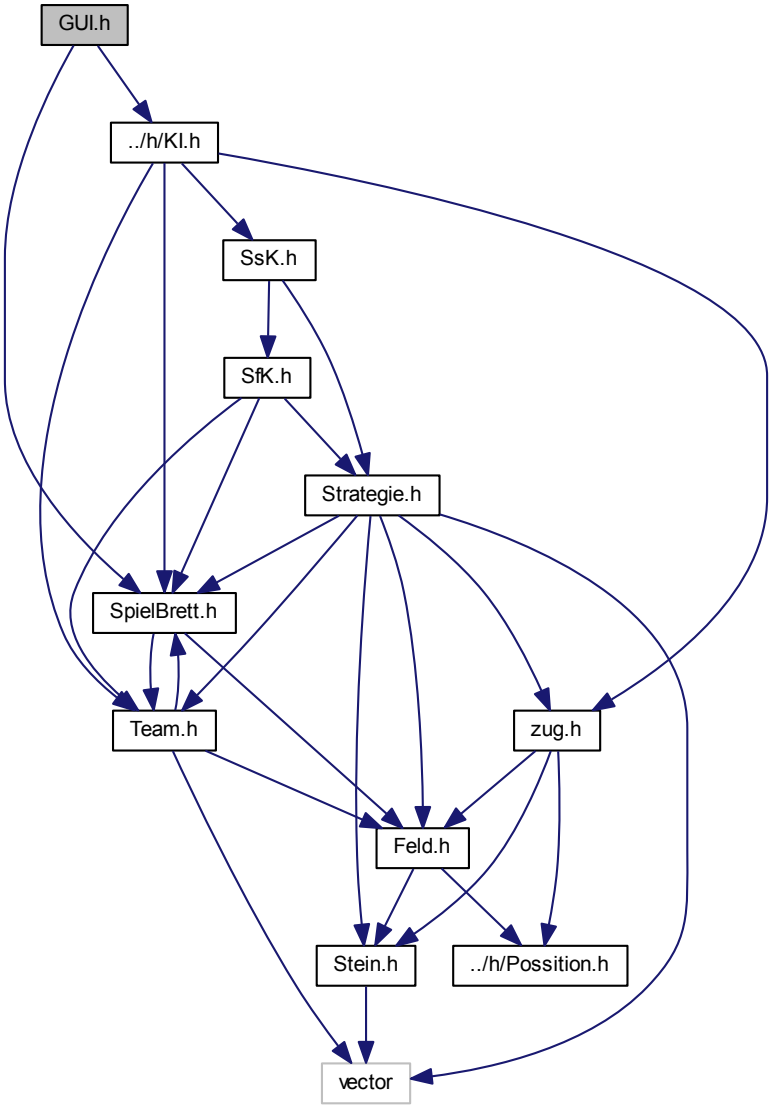
Klassen

- class Feld

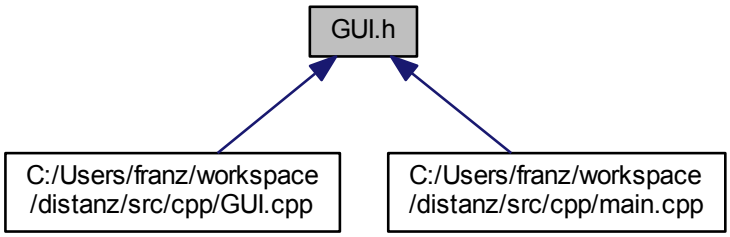
5.16 GUI.h-Dateireferenz

```
#include "SpielBrett.h"
#include "../h/KI.h"
```

Include-Abhängigkeitsdiagramm für GUI.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



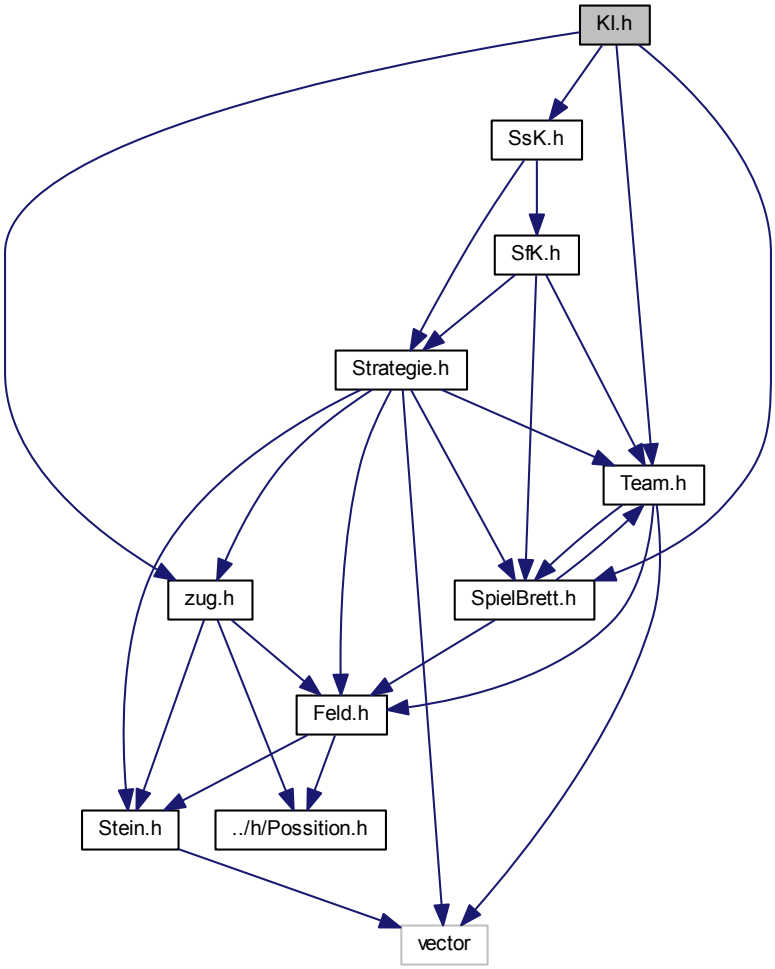
Klassen

- class GUI

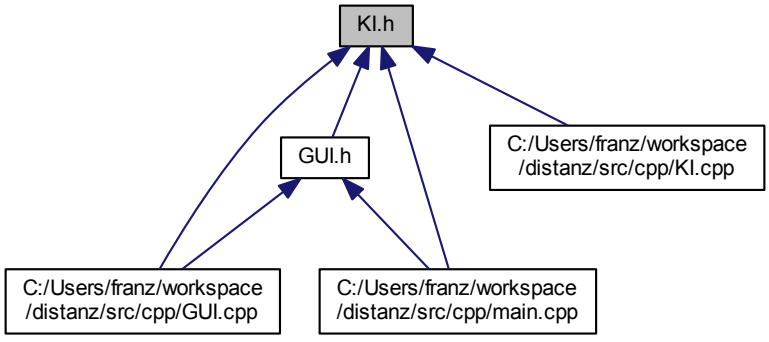
5.17 KI.h-Dateireferenz

```
#include "Team.h"
#include "SpielBrett.h"
#include "zug.h"
#include "SsK.h"
```

Include-Abhängigkeitsdiagramm für KI.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:

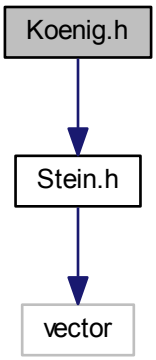


Klassen

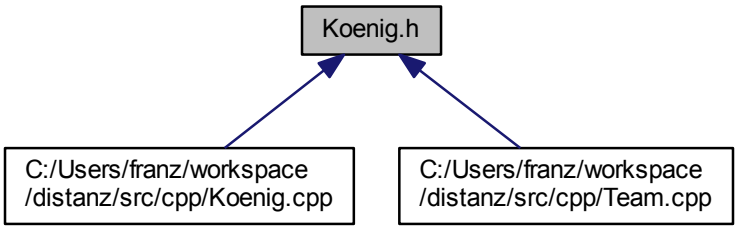
- class KI

5.18 Koenig.h-Dateireferenz

`#include "Stein.h"`
Include-Abhängigkeitsdiagramm für Koenig.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Koenig

Makrodefinitionen

- #define KOENIG_H

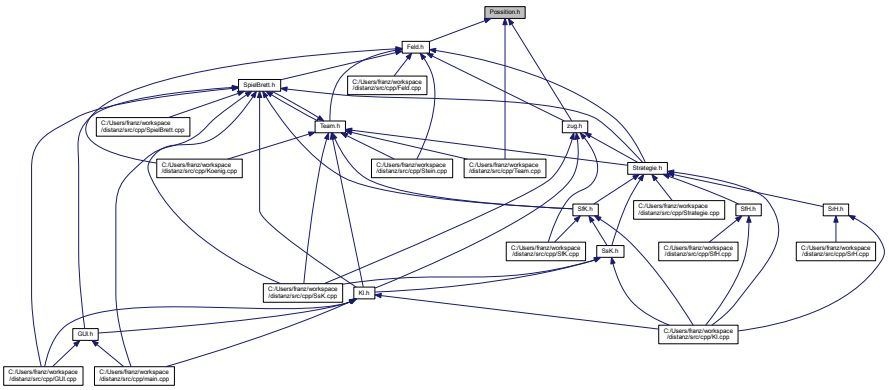
5.18.1 Makro-Dokumentation

5.18.1.1 #define KOENIG_H

5.19 Main.h-Dateireferenz

5.20 Possition.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct Possition

Makrodefinitionen

- #define POSSITION_H

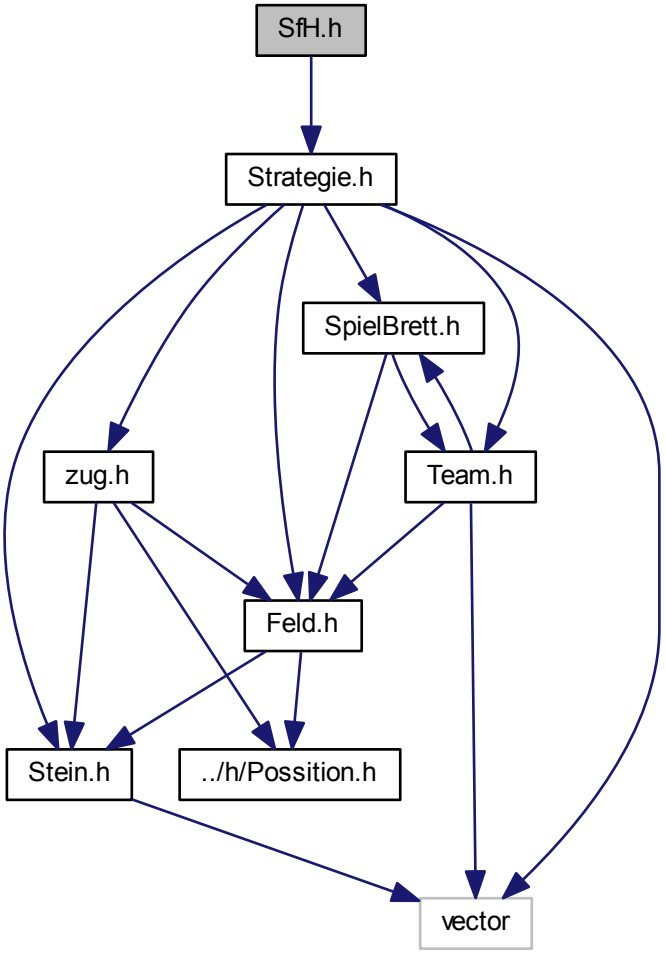
5.20.1 Makro-Dokumentation

5.20.1.1 #define POSSITION_H

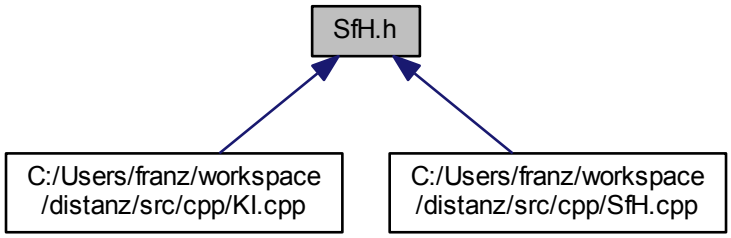
5.21 SfH.h-Dateireferenz

```
#include "Strategie.h"
```

Include-Abhängigkeitsdiagramm für SfH.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



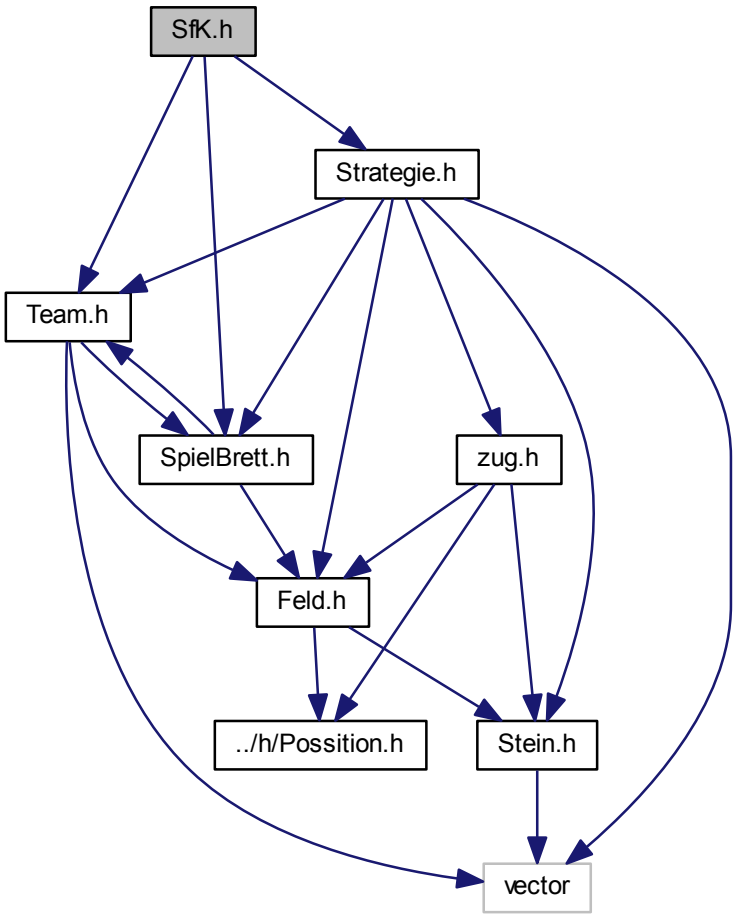
Klassen

- class SfH

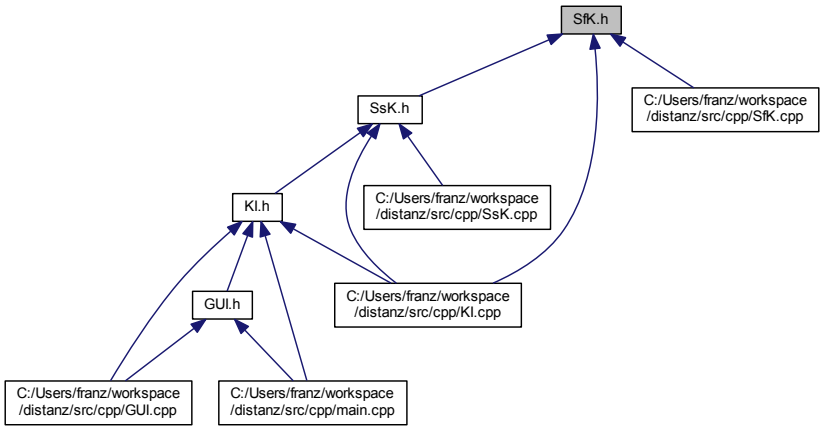
5.22 SfK.h-Dateireferenz

```
#include "Team.h"
#include "SpielBrett.h"
#include "Strategie.h"
```

Include-Abhängigkeitsdiagramm für SfK.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



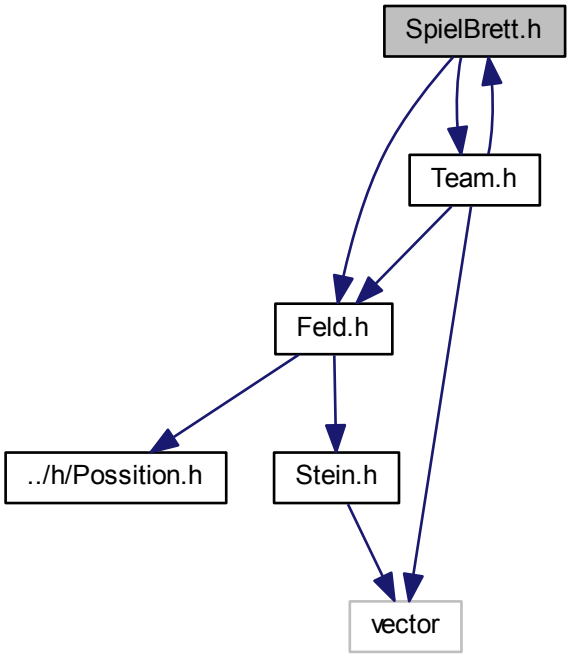
Klassen

- class SfK

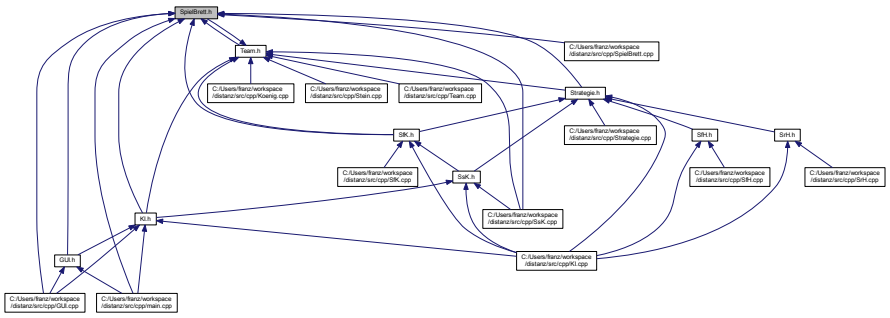
5.23 SpielBrett.h-Dateireferenz

```
#include "Feld.h"
#include "Team.h"
```

Include-Abhängigkeitsdiagramm für SpielBrett.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



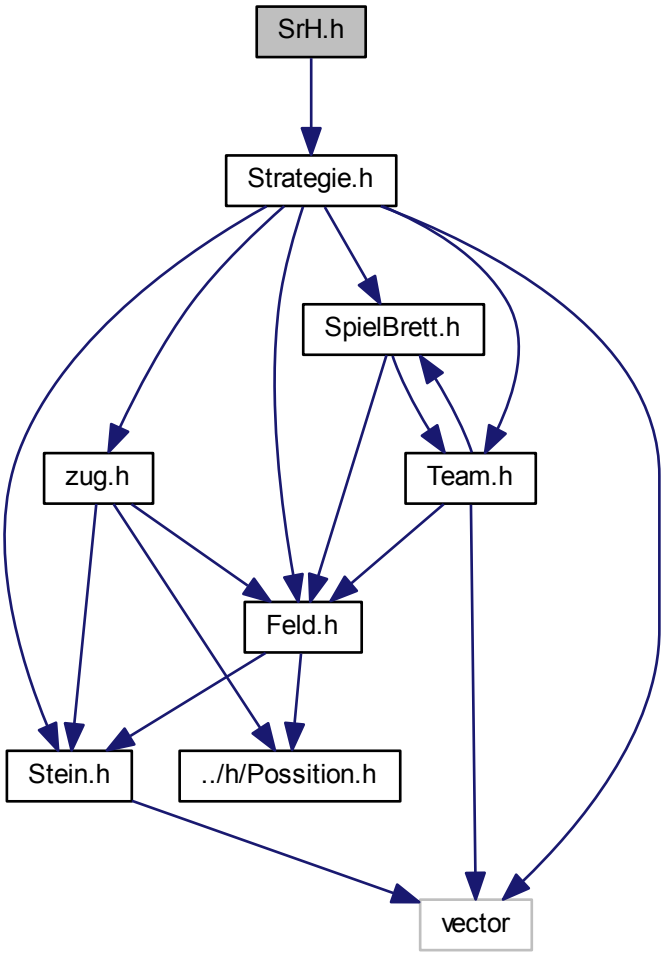
Klassen

- class SpielBrett

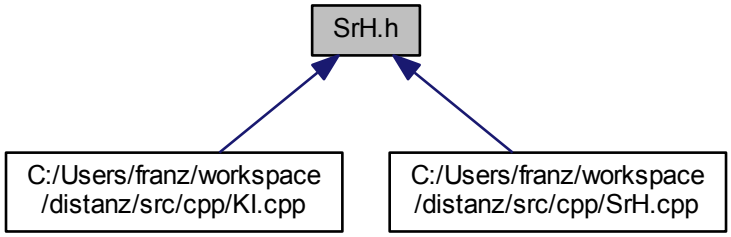
5.24 SrH.h-Dateireferenz

```
#include "Strategie.h"
```

Include-Abhängigkeitsdiagramm für SrH.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



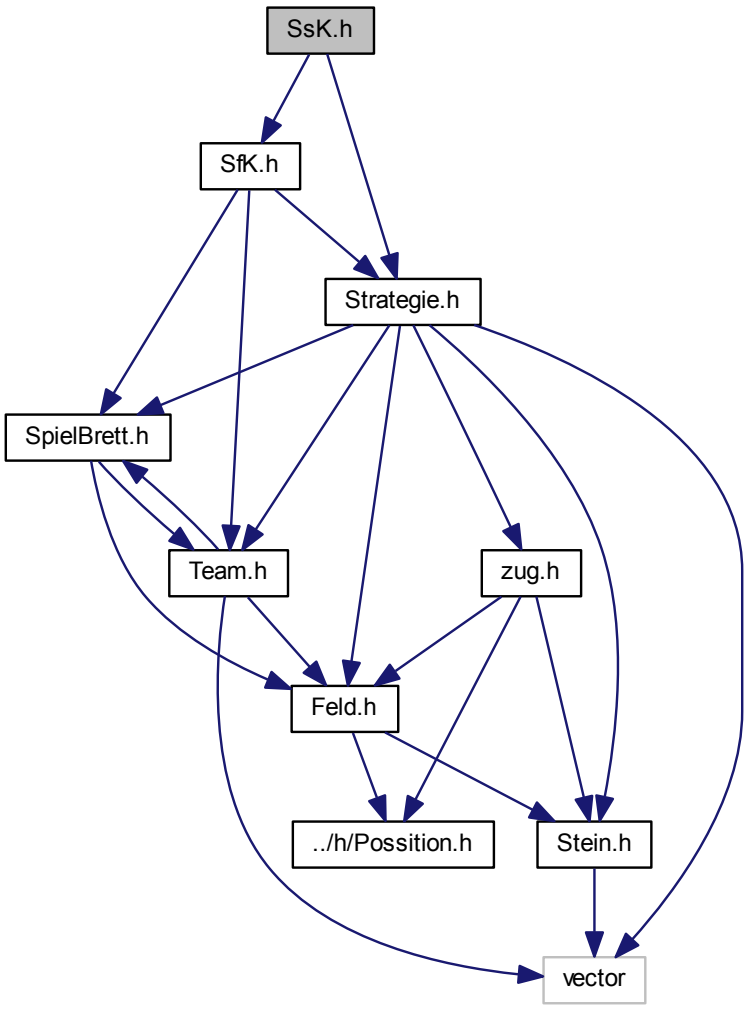
Klassen

- class SrH

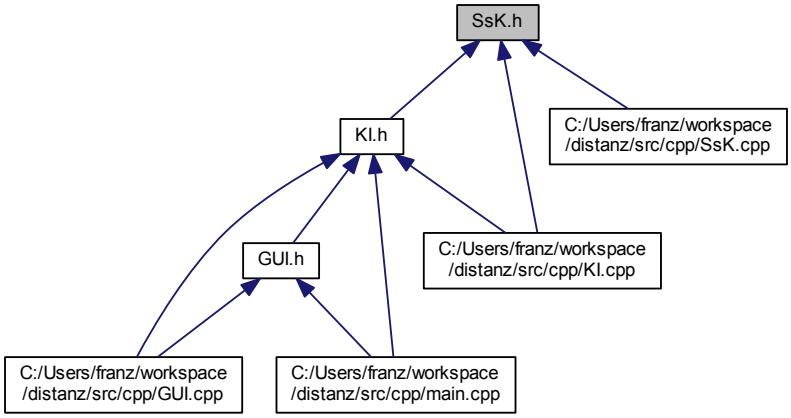
5.25 SsK.h-Dateireferenz

```
#include "Strategie.h"
#include "SfK.h"
```

Include-Abhängigkeitsdiagramm für SsK.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



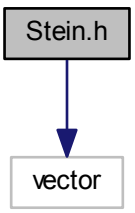
Klassen

- class SsK

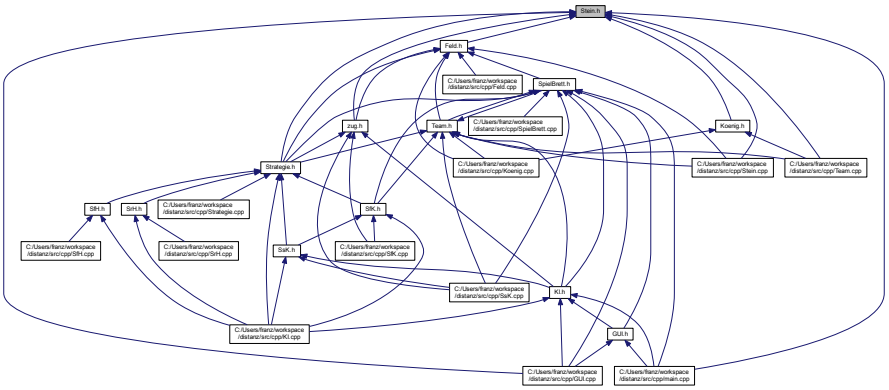
5.26 Stein.h-Dateireferenz

```
#include <vector>
```

Include-Abhängigkeitsdiagramm für Stein.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



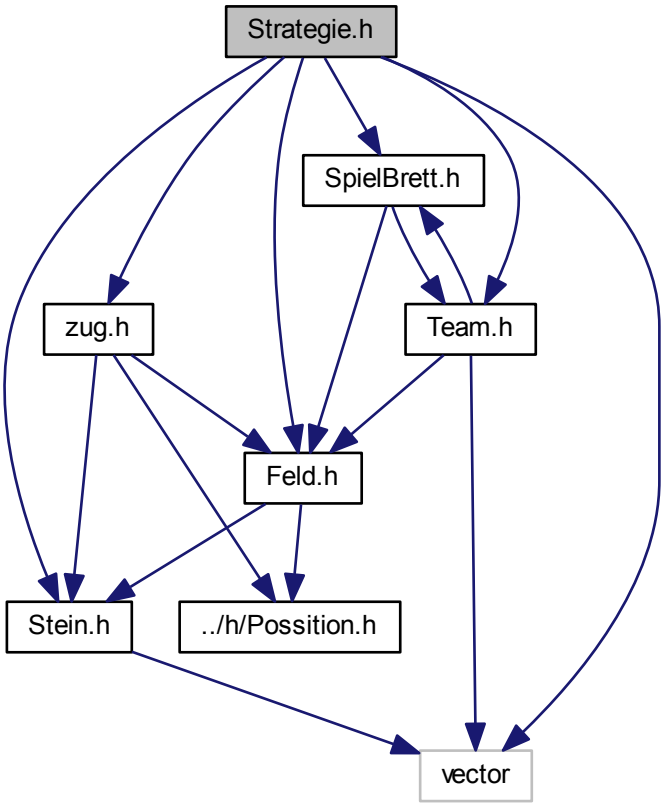
Klassen

- class Stein

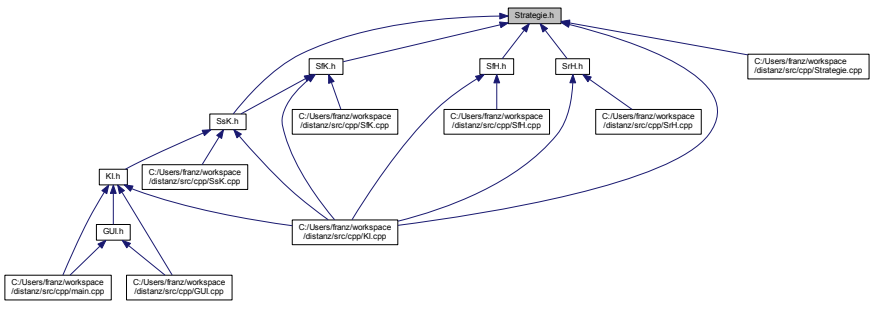
5.27 Strategie.h-Dateireferenz

```
#include <vector>
#include "Feld.h"
#include "SpielBrett.h"
#include "Team.h"
#include "Stein.h"
#include "zug.h"
```

Include-Abhängigkeitsdiagramm für Strategie.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



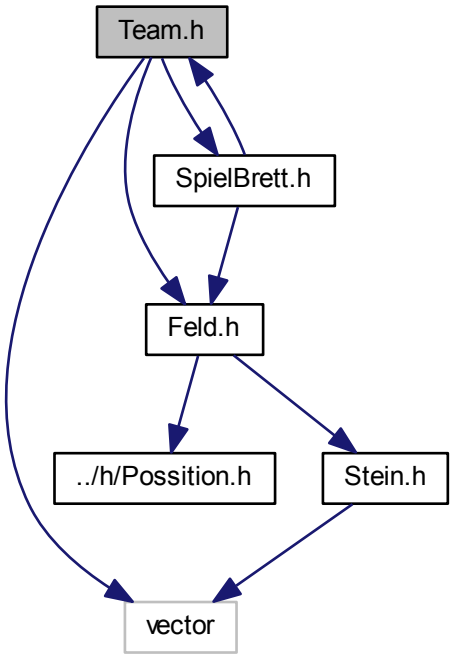
Klassen

- class Strategie

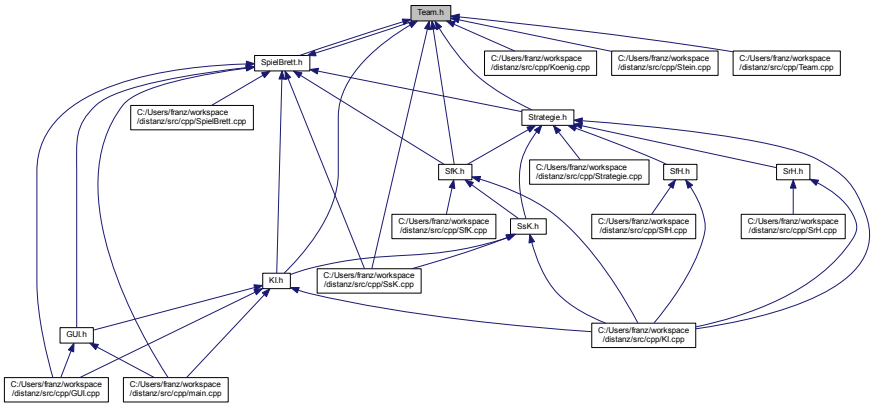
5.28 Team.h-Dateireferenz

```
#include <vector>
#include "Feld.h"
#include "SpielBrett.h"
```

Include-Abhängigkeitsdiagramm für Team.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:

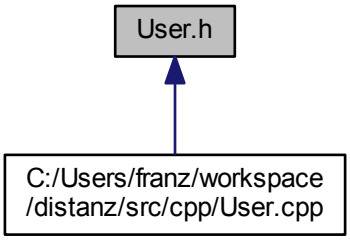


Klassen

- class Team

5.29 User.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



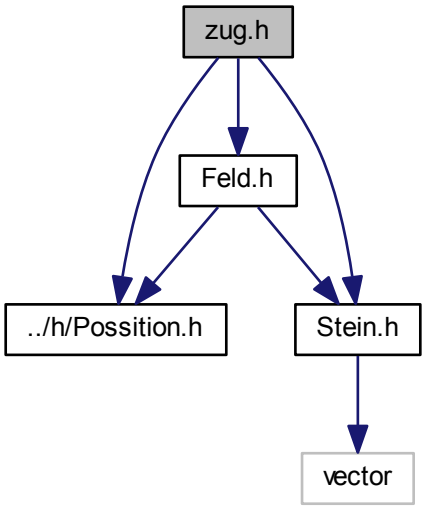
Klassen

- class User

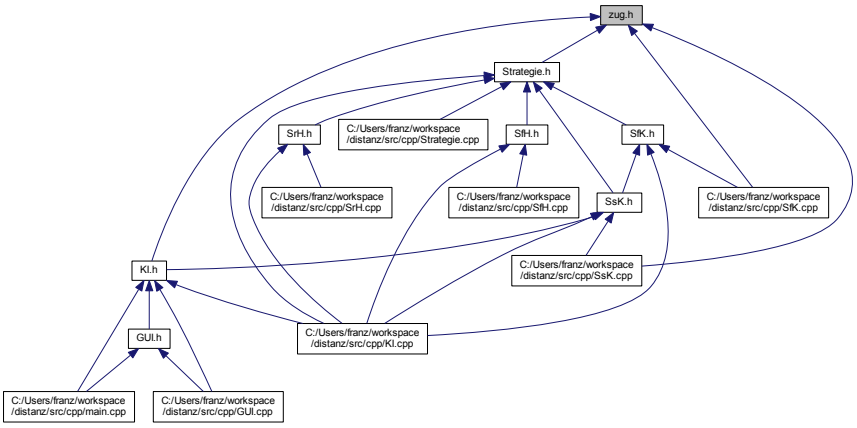
5.30 zug.h-Dateireferenz

```
#include "Feld.h"
#include "Stein.h"
#include "Possition.h"
```

Include-Abhängigkeitsdiagramm für zug.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct zug

Index