Truco Paulista

1.0

Gerado por Doxygen 1.10.0

Capítulo 1

Namespaces

1.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

Capítulo 2

Índice Hierárquico

2.1 Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Capítulo 3

Índice dos Componentes

3.1 Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Capítulo 5

Namespace

5.1 Refência do Namespace GameEngine

Componentes

- class AIPlayer
  Classe base que representa o avatar controlado por AI.
- class AIPlayerController
  Classe base que representa um avatar controlado por AI.
- class AIPlayerState
  Classe base responsavel por gerenciar o estado do Player controlado por AI.
- class GameInstance
  Classe base que gerencia de forma geral uma instancia do jogo em execucao.
- class GameMode
  Classe base responsavel por gerenciar as regras do jogo.
- class GameState
  Classe base responsavel por gerenciar o estado do jogo.
- class Player
  Classe base que representa o avatar de um jogador.
- class PlayerController
  Classe base que representa um jogador.
- class PlayerState
  Classe base responsavel por gerenciar o estado do Player durante as partidas.
- class Result
  Classe que representa o resultado de uma operacao que pode falhar de forma aceitavel.

Enumerações

- enum ResultCode {
  Undefined , Success , Failed , GameIsFull ,
  PlayerAlreadyJoined }
  Enumeracao que representa os possiveis retornos de uma operacao que retorna um Result.

5.1.1 Enumerações

5.1.1.1 ResultCode

`enum GameEngine::ResultCode`

Enumeracao que representa os possiveis retornos de uma operacao que retorna um Result.

Enumeradores

Undefined	
Success	
Failed	
GameIsFull	
PlayerAlreadyJoined	

Definição na linha 9 do arquivo Result.h.

5.2 Refência do Namespace TrucoGame

Componentes

- class Card
  Classe que representa uma carte de baralho.
- class Deck
  Classe que representa o conjunto de cartas de baralho que estao disponiveis no jogo.
- class Hand
  Classe que representa as cartas de baralho que estao em posse (na mao) de um jogador.
- class TrucoAIPlayer
  Especializacao da classe AIPlayer para um jogo de truco.
- class TrucoAIPlayerState
  Especializacao da classe AIPlayerState para um jogo de truco.
- class TrucoGameInstance
  Expecializacao da classe GameInstance para um jogo de truco.
- class TrucoGameMode
  Especializacao da classe GameMode para um jogo de truco.
- class TrucoGameState
  Especializacao da classe GameState para um jogo de truco.
- class TrucoPlayer
  Especializacao da classe Player para um jogo de truco.
- class TrucoPlayerState
  Especializacao da classe PlayerState para uma partida de truco.

Enumerações

- enum Naipes : int {
  Paus , Ouros , Copas , Espadas ,
  Last }
  Enumeracao que representa os possiveis naipes de um baralho.

5.2.1 Enumerações

5.2.1.1 Naipes

```
enum TrucoGame::Naipes : int
```

Enumeracao que representa os possiveis naipes de um baralho.

Enumeradores

Paus	
Ouros	
Copas	
Espadas	
Last	

Definição na linha 8 do arquivo Card.h.

Capítulo 6

Classes

6.1 Referência da Classe GameEngine::AIPlayer

Classe base que representa o avatar controlado por AI.

`#include <AIPlayer.h>`

Diagrama de hierarquia da classe GameEngine::AIPlayer:

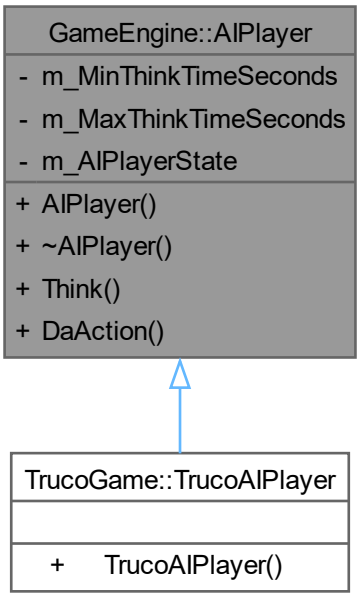
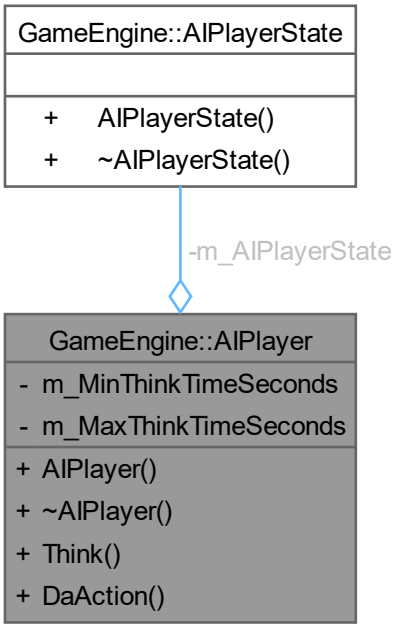


Diagrama de colaboração para GameEngine::AIPlayer:



Membros Públicos

- AIPlayer (AIPlayerState ∗aiPlayerState, double minThinkTimeSec, double maxThinkTimeSec)
- virtual ∼AIPlayer ()
- virtual void Think ()
- virtual void DaAction ()

Atributos Privados

- double m_MinThinkTimeSeconds
- double m_MaxThinkTimeSeconds
- AIPlayerState ∗ m_AIPlayerState

6.1.1 Descrição detalhada

Classe base que representa o avatar controlado por AI.

Aqui teremos as informacoes sobre o avatar que nao mudam entre uma partida e outra.

Definição na linha 12 do arquivo AIPlayer.h.

6.1.2 Construtores e Destrutores

6.1.2.1 AIPlayer()

```
AIPlayer::AIPlayer (
            AIPlayerState * aiPlayerState,
            double minThinkTimeSec,
            double maxThinkTimeSec )
```

Definição na linha 6 do arquivo AIPlayer.cpp.

6.1.2.2 ~AIPlayer()

```
AIPlayer::~AIPlayer ( )  [virtual]
```

Definição na linha 14 do arquivo AIPlayer.cpp.

6.1.3 Documentação das funções

6.1.3.1 DaAction()

```
void AIPlayer::DaAction ( )  [virtual]
```

Definição na linha 26 do arquivo AIPlayer.cpp.

6.1.3.2 Think()

```
void AIPlayer::Think ( )  [virtual]
```

Definição na linha 22 do arquivo AIPlayer.cpp.

6.1.4 Atributos

6.1.4.1 m_AIPlayerState

```
AIPlayerState* GameEngine::AIPlayer::m_AIPlayerState  [private]
```

Definição na linha 17 do arquivo AIPlayer.h.

6.1.4.2 m_MaxThinkTimeSeconds

```
double GameEngine::AIPlayer::m_MaxThinkTimeSeconds  [private]
```

Definição na linha 16 do arquivo AIPlayer.h.

6.1.4.3 m_MinThinkTimeSeconds

```
double GameEngine::AIPlayer::m_MinThinkTimeSeconds  [private]
```

Definição na linha 15 do arquivo AIPlayer.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayer.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayer.cpp

6.2 Referência da Classe GameEngine::AIPlayerController

Classe base que representa um avatar controlado por AI.

```
#include <AIPlayerController.h>
```

Diagrama de hierarquia da classe GameEngine::AIPlayerController:

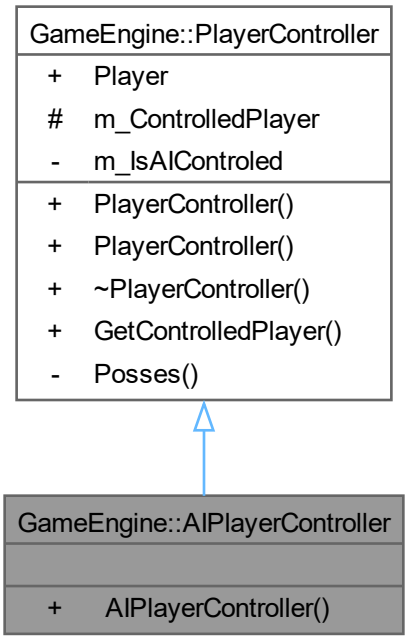
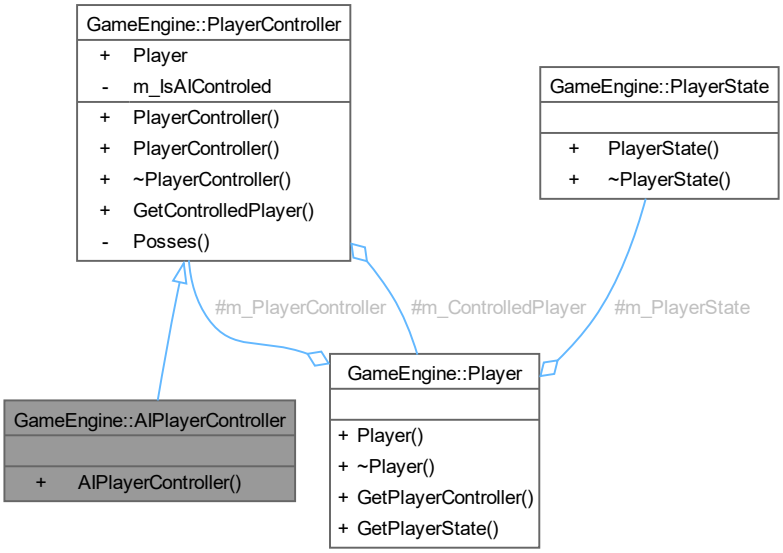


Diagrama de colaboração para GameEngine::AIPlayerController:



Membros Públicos

- AIPlayerController ()

Membros Públicos herdados de GameEngine::PlayerController

- PlayerController ()
- PlayerController (bool isAIControlled)
- virtual ~PlayerController ()
- Player * GetControlledPlayer ()

Outros membros herdados

Atributos Públicos herdados de GameEngine::PlayerController

- friend Player

Atributos Protegidos herdados de GameEngine::PlayerController

- Player * m_ControlledPlayer

6.2.1 Descrição detalhada

Classe base que representa um avatar controlado por AI.

Ela e responsavel por controlar um avatar atraves de inputs gerados por arvore de decisao, maquina de estado finita, entre outros. Pode se considerar como cerebro do AIPlayer.

Definição na linha 13 do arquivo AIPlayerController.h.

6.2.2 Construtores e Destrutores

6.2.2.1 AIPlayerController()

```
AIPlayerController::AIPlayerController ( )
```

Definição na linha 5 do arquivo AIPlayerController.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayerController.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayerController.cpp

6.3 Referência da Classe GameEngine::AIPlayerState

Classe base responsavel por gerenciar o estado do Player controlado por AI.

`#include <AIPlayerState.h>`

Diagrama de hierarquia da classe GameEngine::AIPlayerState:

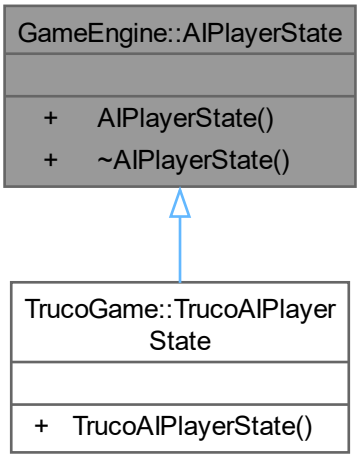
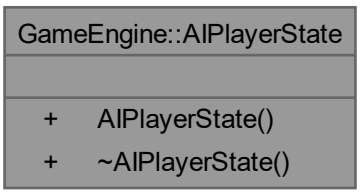


Diagrama de colaboração para GameEngine::AIPlayerState:



Membros Públicos

- AIPlayerState ()
- virtual ~AIPlayerState ()=default

6.3.1 Descrição detalhada

Classe base responsavel por gerenciar o estado do Player controlado por AI.

Aqui teremos informacoes pertinentes ao estado da AI durante as partidas.

Definição na linha 10 do arquivo AIPlayerState.h.

6.3.2 Construtores e Destrutores

6.3.2.1 AIPlayerState()

```
AIPlayerState::AIPlayerState ( )
```

Definição na linha 5 do arquivo AIPlayerState.cpp.

6.3.2.2 ~AIPlayerState()

```
virtual GameEngine::AIPlayerState::~AIPlayerState ( )  [virtual], [default]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

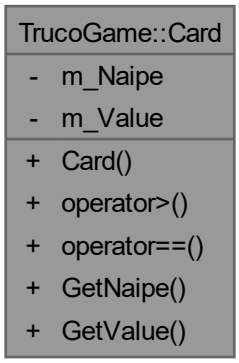
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayerState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayerState.cpp

6.4 Referência da Classe TrucoGame::Card

Classe que representa uma carte de baralho.

```
#include <Card.h>
```

Diagrama de colaboração para TrucoGame::Card:



Membros Públicos

- Card (Naipes naipe, int value)
- bool operator> (const Card &card) const
- bool operator== (const Card &card) const
- Naipes GetNaipe ()
- int GetValue ()

Atributos Privados

- Naipes m_Naipe
- int m_Value

6.4.1 Descrição detalhada

Classe que representa uma carte de baralho.

Definição na linha 20 do arquivo Card.h.

6.4.2 Construtores e Destrutores

6.4.2.1 Card()

```
Card::Card (
            Naipes naipe,
            int value )
```

Definição na linha 5 do arquivo Card.cpp.

6.4.3 Documentação das funções

6.4.3.1 GetNaipe()

```
Naipes Card::GetNaipe ( )
```

Definição na linha 51 do arquivo Card.cpp.

6.4.3.2 GetValue()

```
int Card::GetValue ( )
```

Definição na linha 56 do arquivo Card.cpp.

6.4.3.3 operator==()

```
bool Card::operator== (
            const Card & card ) const
```

Definição na linha 39 do arquivo Card.cpp.

6.4.3.4 operator>()

```
bool Card::operator> (
            const Card & card ) const
```

Definição na linha 11 do arquivo Card.cpp.

6.4.4 Atributos

6.4.4.1 m_Naipe

```
Naipes TrucoGame::Card::m_Naipe  [private]
```

Definição na linha 23 do arquivo Card.h.

6.4.4.2 m_Value

```
int TrucoGame::Card::m_Value  [private]
```

Definição na linha 24 do arquivo Card.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

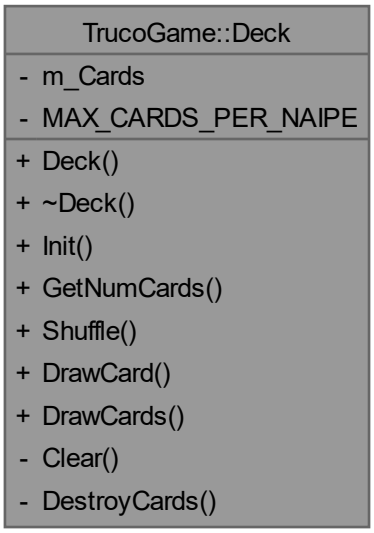
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Card.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Card.cpp

6.5 Referência da Classe TrucoGame::Deck

Classe que representa o conjunto de cartas de baralho que estao disponiveis no jogo.

`#include <Deck.h>`

Diagrama de colaboração para TrucoGame::Deck:



Membros Públicos

- Deck ()
- ~Deck ()
- void Init ()
- int GetNumCards ()
- void Shuffle ()
- Card * DrawCard ()
- std::vector< Card * > DrawCards (int numCards)

Membros privados

- void Clear ()
- void DestroyCards ()

Atributos Privados

- std::stack< Card * > m_Cards
- const int MAX_CARDS_PER_NAIPE = 10

6.5.1 Descrição detalhada

Classe que representa o conjunto de cartas de baralho que estao disponiveis no jogo.

Definição na linha 13 do arquivo Deck.h.

6.5.2 Construtores e Destrutores

6.5.2.1 Deck()

```
Deck::Deck ( )
```

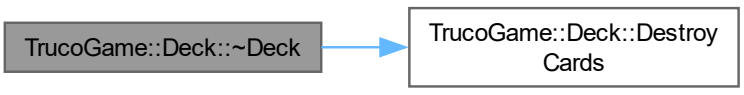
Definição na linha 41 do arquivo Deck.cpp.

6.5.2.2 ~Deck()

```
Deck::~Deck ( )
```

Definição na linha 45 do arquivo Deck.cpp.

Este é o diagrama das funções utilizadas por essa função:



6.5.3 Documentação das funções

6.5.3.1 Clear()

```
void Deck::Clear ( )  [private]
```

Definição na linha 22 do arquivo Deck.cpp.

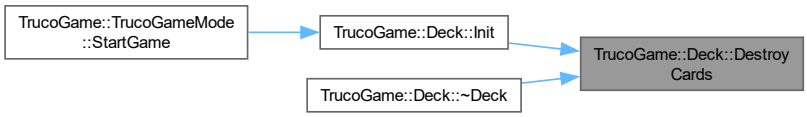
Esse é o diagrama das funções que utilizam essa função:

6.5.3.2 DestroyCards()

```
void Deck::DestroyCards ( )  [private]
```

Definição na linha 30 do arquivo Deck.cpp.

Esse é o diagrama das funções que utilizam essa função:

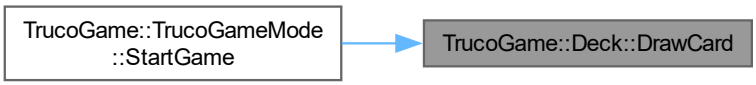


6.5.3.3 DrawCard()

```
Card * Deck::DrawCard ( )
```

Definição na linha 74 do arquivo Deck.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.5.3.4 DrawCards()

```
std::vector< Card * > Deck::DrawCards (
            int numCards )
```

Definição na linha 82 do arquivo Deck.cpp.

6.5.3.5 GetNumCards()

```
int Deck::GetNumCards ( )
```

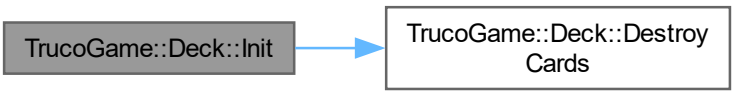
Definição na linha 50 do arquivo Deck.cpp.

6.5.3.6 Init()

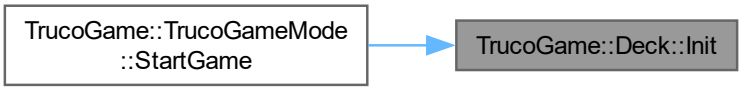
```
void Deck::Init ( )
```

Definição na linha 8 do arquivo Deck.cpp.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



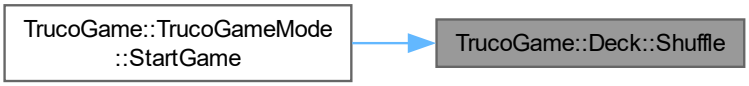
6.5.3.7 Shuffle()

```
void Deck::Shuffle ( )
```

Definição na linha 55 do arquivo Deck.cpp.

Este é o diagrama das funções utilizadas por essa função:

Esse é o diagrama das funções que utilizam essa função:



6.5.4 Atributos

6.5.4.1 m_Cards

```
std::stack<Card*> TrucoGame::Deck::m_Cards  [private]
```

Definição na linha 16 do arquivo Deck.h.

6.5.4.2 MAX_CARDS_PER_NAIPE

```
const int TrucoGame::Deck::MAX_CARDS_PER_NAIPE = 10  [private]
```

Definição na linha 17 do arquivo Deck.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Deck.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Deck.cpp

6.6 Referência da Classe GameEngine::GameInstance

Classe base que gerencia de forma geral uma instancia do jogo em execucao.

`#include <GameInstance.h>`

Diagrama de hierarquia da classe GameEngine::GameInstance:

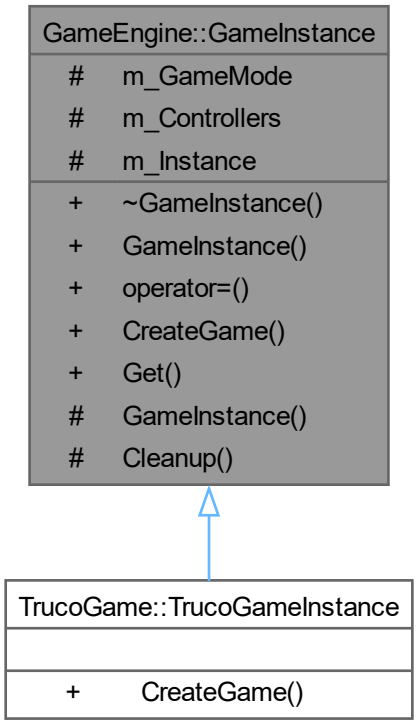
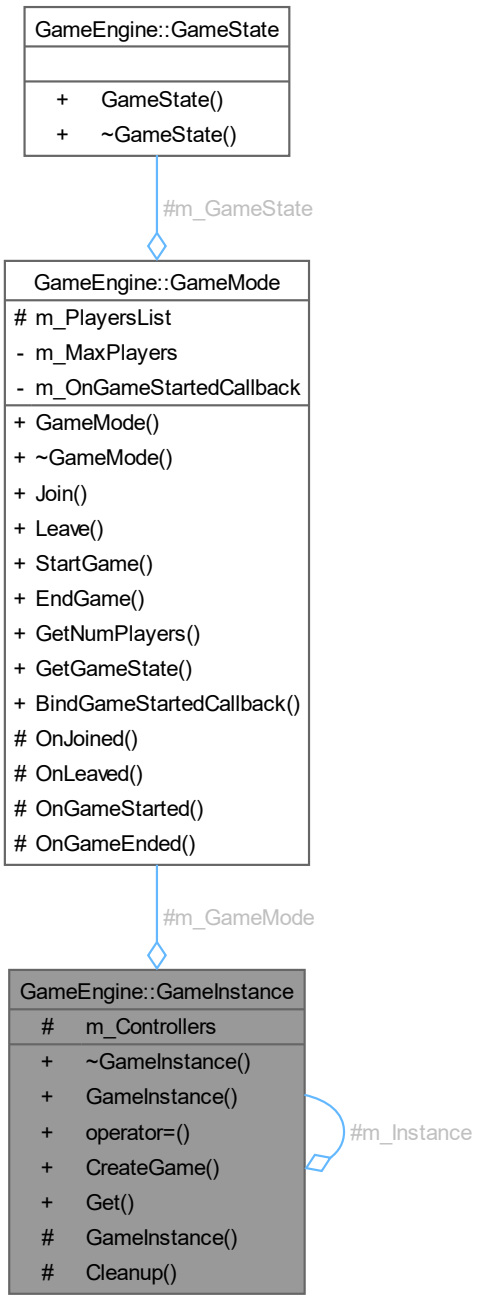


Diagrama de colaboração para GameEngine::GameInstance:



Membros Públicos

- ~GameInstance ()
- GameInstance (GameInstance &other)=delete
- void operator= (const GameInstance &)=delete
- virtual void CreateGame (int numPlayers, int numAIPlayers, GameMode *pGameMode)

Membros públicos estáticos

- static GameInstance * Get ()

Membros protegidos

- GameInstance ()
- void Cleanup ()

Atributos Protegidos

- GameMode * m_GameMode
- std::vector< PlayerController * > m_Controllers

Atributos Protegidos Estáticos

- static GameInstance * m_Instance = nullptr

6.6.1 Descrição detalhada

Classe base que gerencia de forma geral uma instancia do jogo em execucao.

Aqui teremos controle sobre o inicio de partidas, acesso aos jogadores, configuracao de jogo, e etc. E o ponto de entrada de gerenciamento e configuracao das partidas.

Definição na linha 16 do arquivo GameInstance.h.

6.6.2 Construtores e Destrutores

6.6.2.1 GameInstance() [1/2]

```
GameInstance::GameInstance ( )  [protected]
```

Definição na linha 27 do arquivo GameInstance.cpp.

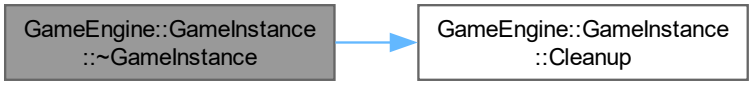
Esse é o diagrama das funções que utilizam essa função:

6.6.2.2 ~GameInstance()

```
GameInstance::~GameInstance ( )
```

Definição na linha 31 do arquivo GameInstance.cpp.

Este é o diagrama das funções utilizadas por essa função:



6.6.2.3 GameInstance() [2/2]

```
GameEngine::GameInstance::GameInstance (
            GameInstance & other )  [delete]
```

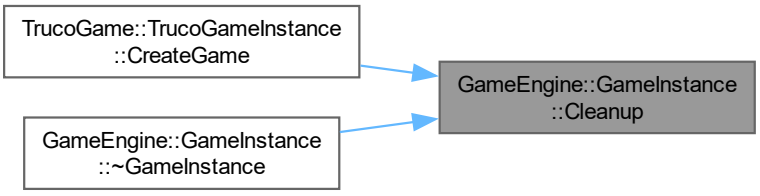
6.6.3 Documentação das funções

6.6.3.1 Cleanup()

```
void GameInstance::Cleanup ( )  [protected]
```

Definição na linha 13 do arquivo GameInstance.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.6.3.2 CreateGame()

```
void GameInstance::CreateGame (
            int numPlayers,
            int numAIPlayers,
            GameMode * pGameMode )  [virtual]
```

Reimplementado por TrucoGame::TrucoGameInstance.

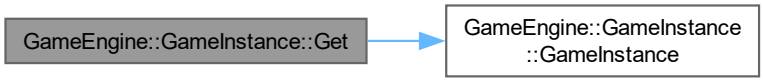
Definição na linha 51 do arquivo GameInstance.cpp.

6.6.3.3 Get()

```
GameInstance * GameInstance::Get ( )  [static]
```

Definição na linha 41 do arquivo GameInstance.cpp.

Este é o diagrama das funções utilizadas por essa função:



6.6.3.4 operator=()

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

6.6.4 Atributos

6.6.4.1 m_Controllers

```
std::vector<PlayerController*> GameEngine::GameInstance::m_Controllers  [protected]
```

Definição na linha 20 do arquivo GameInstance.h.

6.6.4.2 m_GameMode

```
GameMode* GameEngine::GameInstance::m_GameMode  [protected]
```

Definição na linha 19 do arquivo GameInstance.h.

6.6.4.3 m_Instance

```
GameInstance * GameInstance::m_Instance = nullptr  [static], [protected]
```

Definição na linha 25 do arquivo GameInstance.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameInstance.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameInstance.cpp

6.7 Referência da Classe GameEngine::GameMode

Classe base responsavel por gerenciar as regras do jogo.

```
#include <GameMode.h>
```

Diagrama de hierarquia da classe GameEngine::GameMode:

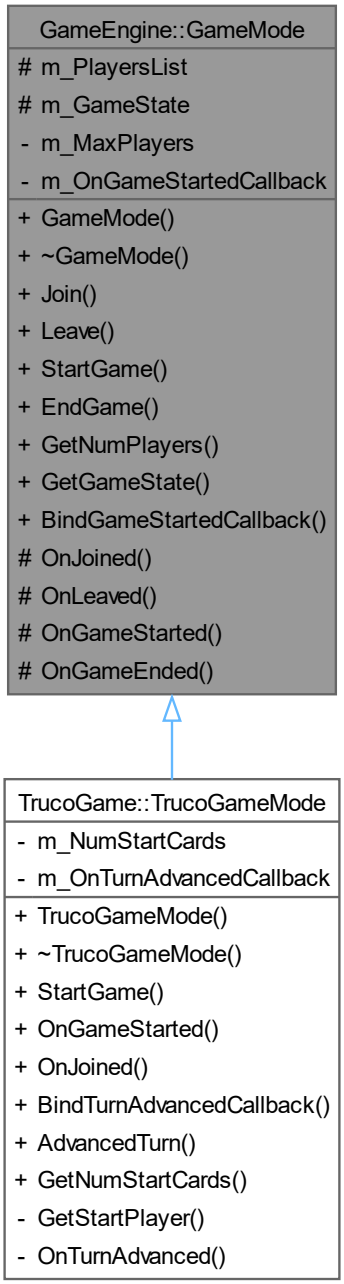
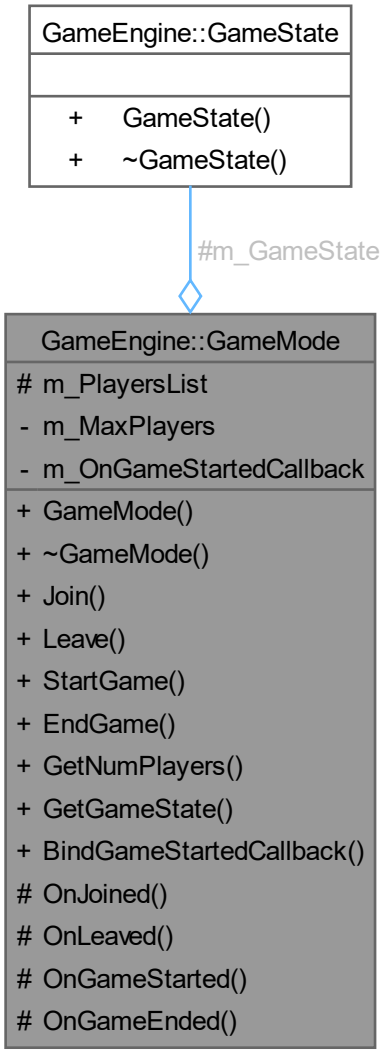


Diagrama de colaboração para GameEngine::GameMode:



Membros Públicos

- GameMode (int maxPlayers, GameState ∗pGameState)
- virtual ∼GameMode ()
- Result Join (PlayerController ∗pController, bool isAIControlled=false)
- void Leave (PlayerController ∗pController)
- virtual void StartGame ()
- virtual void EndGame ()
- int GetNumPlayers ()
- GameState ∗ GetGameState ()
- void BindGameStartedCallback (std::function< void(void)> func)

Membros protegidos

- virtual void OnJoined (PlayerController *pController, bool isAIControlled=false)
- virtual void OnLeaved (PlayerController *pController)
- virtual void OnGameStarted ()
- virtual void OnGameEnded ()

Atributos Protegidos

- std::vector< PlayerController * > m_PlayersList
- GameState * m_GameState

Atributos Privados

- int m_MaxPlayers = 1
- std::function< void(void)> m_OnGameStartedCallback

6.7.1 Descrição detalhada

Classe base responsavel por gerenciar as regras do jogo.

Definição na linha 15 do arquivo GameMode.h.

6.7.2 Construtores e Destrutores

6.7.2.1 GameMode()

```
GameMode::GameMode (
            int maxPlayers,
            GameState * pGameState )
```

Definição na linha 10 do arquivo GameMode.cpp.

6.7.2.2 ~GameMode()

```
GameMode::~GameMode ( )  [virtual]
```

Definição na linha 18 do arquivo GameMode.cpp.

6.7.3 Documentação das funções

6.7.3.1 BindGameStartedCallback()

```
void GameMode::BindGameStartedCallback (
            std::function< void(void)> func )
```

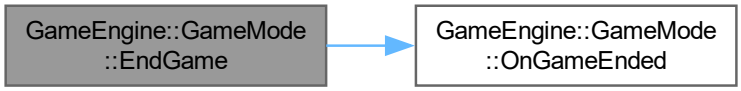
Definição na linha 102 do arquivo GameMode.cpp.

6.7.3.2 EndGame()

```
void GameMode::EndGame ( )  [virtual]
```

Definição na linha 87 do arquivo GameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



6.7.3.3 GetGameState()

```
GameState * GameMode::GetGameState ( )
```

Definição na linha 97 do arquivo GameMode.cpp.

6.7.3.4 GetNumPlayers()

```
int GameMode::GetNumPlayers ( )
```

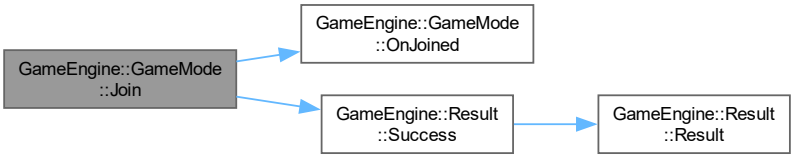
Definição na linha 92 do arquivo GameMode.cpp.

6.7.3.5 Join()

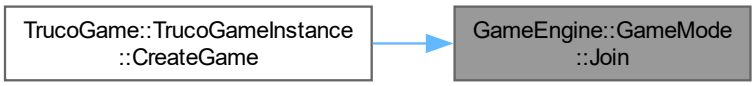
```
Result GameMode::Join (
            PlayerController * pController,
            bool isAIControlled = false )
```

Definição na linha 46 do arquivo GameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:

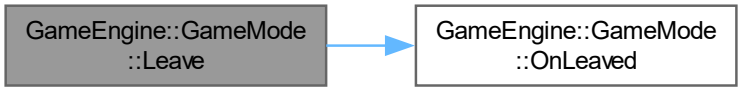


6.7.3.6 Leave()

```
void GameMode::Leave (
            PlayerController * pController )
```

Definição na linha 69 do arquivo GameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:

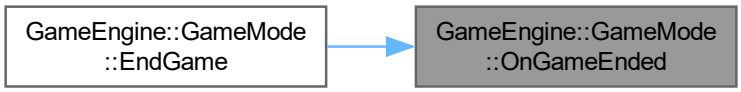


6.7.3.7 OnGameEnded()

```
void GameMode::OnGameEnded ( )  [protected], [virtual]
```

Definição na linha 42 do arquivo GameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:



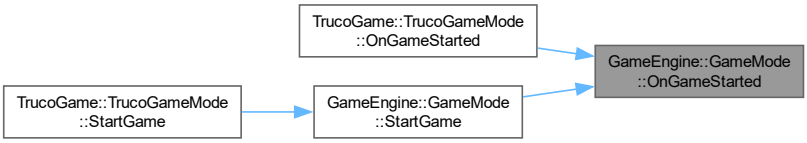
6.7.3.8 OnGameStarted()

```
void GameMode::OnGameStarted ( )  [protected], [virtual]
```

Reimplementado por TrucoGame::TrucoGameMode.

Definição na linha 34 do arquivo GameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:



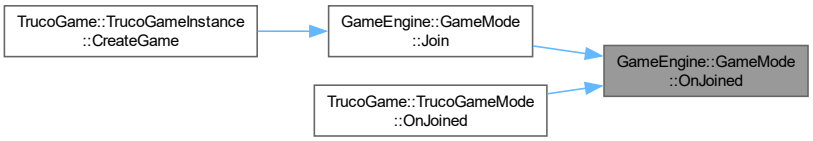
6.7.3.9 OnJoined()

```
void GameMode::OnJoined (
            PlayerController * pController,
            bool isAIControlled = false )  [protected], [virtual]
```

Reimplementado por TrucoGame::TrucoGameMode.

Definição na linha 26 do arquivo GameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:

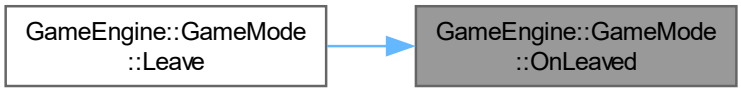


6.7.3.10 OnLeaved()

```
void GameMode::OnLeaved (
            PlayerController * pController )  [protected], [virtual]
```

Definição na linha 30 do arquivo GameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:



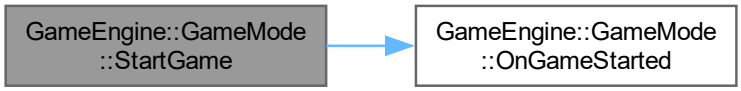
6.7.3.11 StartGame()

```
void GameMode::StartGame ( )  [virtual]
```

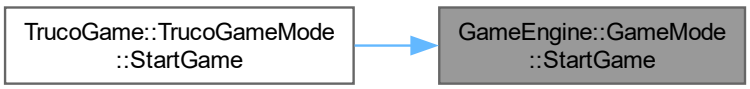
Reimplementado por TrucoGame::TrucoGameMode.

Definição na linha 82 do arquivo GameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



6.7.4 Atributos

6.7.4.1 m_GameState

```
GameState* GameEngine::GameMode::m_GameState  [protected]
```

Definição na linha 23 do arquivo GameMode.h.

6.7.4.2 m_MaxPlayers

```
int GameEngine::GameMode::m_MaxPlayers = 1  [private]
```

Definição na linha 18 do arquivo GameMode.h.

6.7.4.3 m_OnGameStartedCallback

```
std::function<void(void)> GameEngine::GameMode::m_OnGameStartedCallback  [private]
```

Definição na linha 19 do arquivo GameMode.h.

6.7.4.4 m_PlayersList

```
std::vector<PlayerController*> GameEngine::GameMode::m_PlayersList  [protected]
```

Definição na linha 22 do arquivo GameMode.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameMode.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameMode.cpp

6.8 Referência da Classe GameEngine::GameState

Classe base responsavel por gerenciar o estado do jogo.

```
#include <GameState.h>
```

Diagrama de hierarquia da classe GameEngine::GameState:

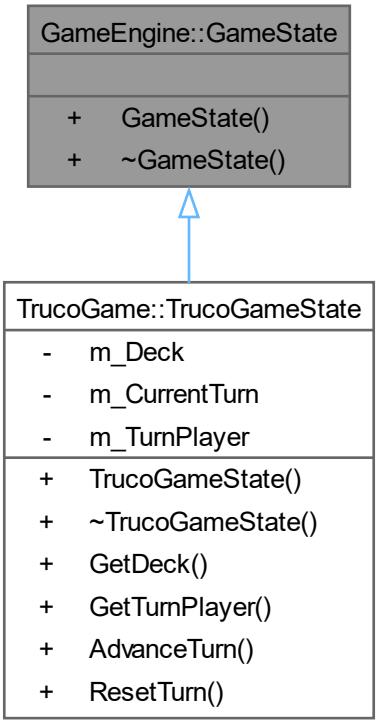
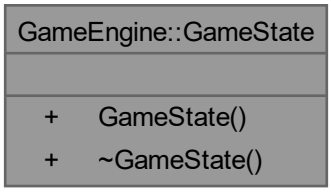


Diagrama de colaboração para GameEngine::GameState:



Membros Públicos

- GameState ()
- virtual ~GameState ()=default

6.8.1 Descrição detalhada

Classe base responsavel por gerenciar o estado do jogo.

Aqui teremos as informacoes que sao relevantes apenas para a partida em curso.

Definição na linha 10 do arquivo GameState.h.

6.8.2 Construtores e Destrutores

6.8.2.1 GameState()

```
GameState::GameState ( )
```

Definição na linha 5 do arquivo GameState.cpp.

6.8.2.2 ~GameState()

```
virtual GameEngine::GameState::~GameState ( )  [virtual], [default]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameState.cpp

6.9 Referência da Classe TrucoGame::Hand

Classe que representa as cartas de baralho que estao em posse (na mao) de um jogador.

```
#include <Hand.h>
```

Diagrama de colaboração para TrucoGame::Hand:

Membros Públicos

- Hand ()
- ~Hand ()
- void SetInitalCards (std::vector< Card * > cards)
- void AddCard (Card *card)
- Card * PlayCard (int index)
- std::vector< Card * > GetCards ()

Atributos Privados

- std::vector< Card * > m_Cards

6.9.1 Descrição detalhada

Classe que representa as cartas de baralho que estao em posse (na mao) de um jogador.

Definição na linha 12 do arquivo Hand.h.

6.9.2 Construtores e Destrutores

6.9.2.1 Hand()

```
Hand::Hand ( )
```

Definição na linha 6 do arquivo Hand.cpp.

6.9.2.2 ~Hand()

```
Hand::~Hand ( )
```

Definição na linha 10 do arquivo Hand.cpp.

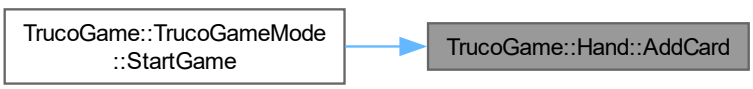
6.9.3 Documentação das funções

6.9.3.1 AddCard()

```
void Hand::AddCard (
            Card * card )
```

Definição na linha 29 do arquivo Hand.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.9.3.2 GetCards()

```
std::vector< Card * > Hand::GetCards ( )
```

Definição na linha 42 do arquivo Hand.cpp.

6.9.3.3 PlayCard()

```
Card * Hand::PlayCard (
            int index )
```

Definição na linha 34 do arquivo Hand.cpp.

6.9.3.4 SetInitalCards()

```
void Hand::SetInitalCards (
            std::vector< Card * > cards )
```

Definição na linha 23 do arquivo Hand.cpp.

6.9.4 Atributos

6.9.4.1 m_Cards

```
std::vector<Card*> TrucoGame::Hand::m_Cards  [private]
```

Definição na linha 15 do arquivo Hand.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Hand.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Hand.cpp

6.10 Referência da Classe GameEngine::Player

Classe base que representa o avatar de um jogador.

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

Diagrama de hierarquia da classe GameEngine::Player:

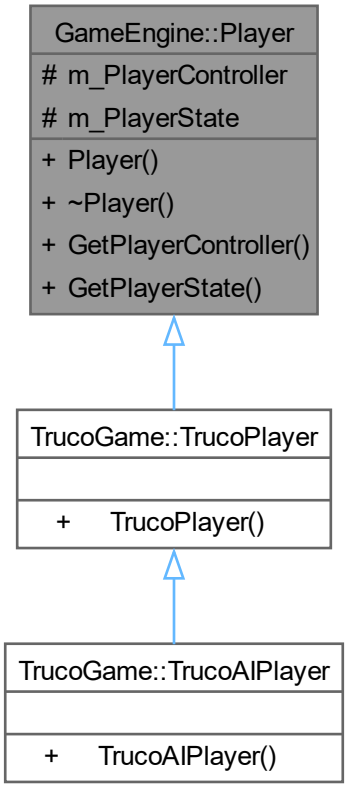
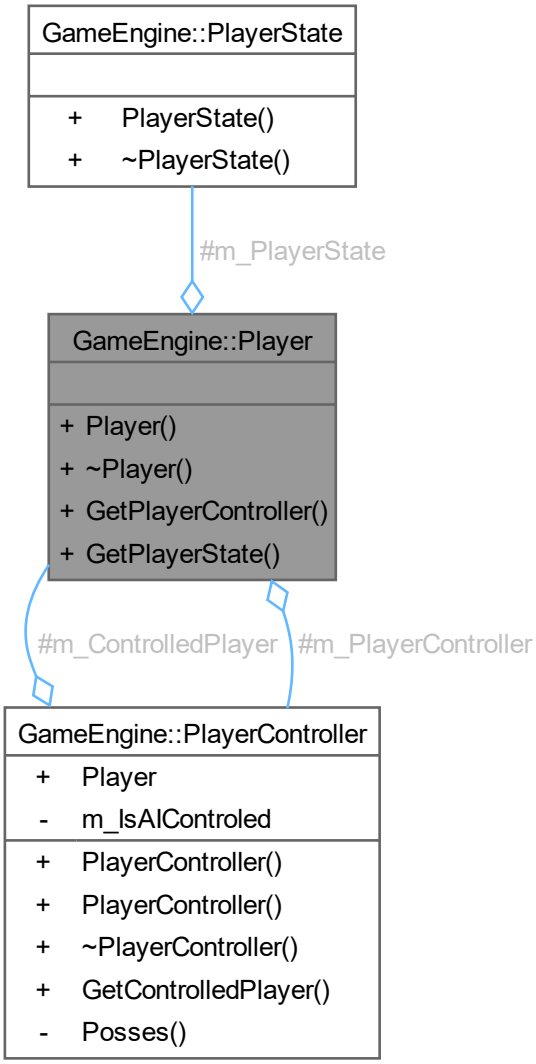


Diagrama de colaboração para GameEngine::Player:



Membros Públicos

- Player (PlayerState ∗pPlayerState, PlayerController ∗pPlayerController)
- virtual ∼Player ()
- PlayerController ∗ GetPlayerController ()
- PlayerState ∗ GetPlayerState ()

Atributos Protegidos

- PlayerController ∗ m_PlayerController
- PlayerState ∗ m_PlayerState

6.10.1 Descrição detalhada

Classe base que representa o avatar de um jogador.

Aqui teremos as informacoes sobre o jogador que nao mudam entre uma partida e outra.

Definição na linha 13 do arquivo Player.h.

6.10.2 Construtores e Destrutores

6.10.2.1 Player()

```
Player::Player (
            PlayerState * pPlayerState,
            PlayerController * pPlayerController )
```

Definição na linha 7 do arquivo Player.cpp.

Este é o diagrama das funções utilizadas por essa função:

6.10.2.2 ~Player()

```
Player::~Player ( )  [virtual]
```

Definição na linha 15 do arquivo Player.cpp.

6.10.3 Documentação das funções

6.10.3.1 GetPlayerController()

```
PlayerController * Player::GetPlayerController ( )
```

Definição na linha 23 do arquivo Player.cpp.

6.10.3.2 GetPlayerState()

```
PlayerState * Player::GetPlayerState ( )
```

Definição na linha 28 do arquivo Player.cpp.

6.10.4 Atributos

6.10.4.1 m_PlayerController

```
PlayerController* GameEngine::Player::m_PlayerController  [protected]
```

Definição na linha 16 do arquivo Player.h.

6.10.4.2 m_PlayerState

```
PlayerState* GameEngine::Player::m_PlayerState  [protected]
```

Definição na linha 17 do arquivo Player.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/Player.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/Player.cpp

6.11 Referência da Classe GameEngine::PlayerController

Classe base que representa um jogador.

```
#include <PlayerController.h>
```

Diagrama de hierarquia da classe GameEngine::PlayerController:

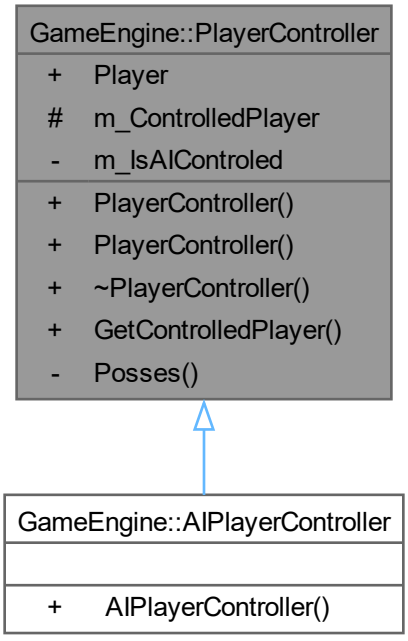
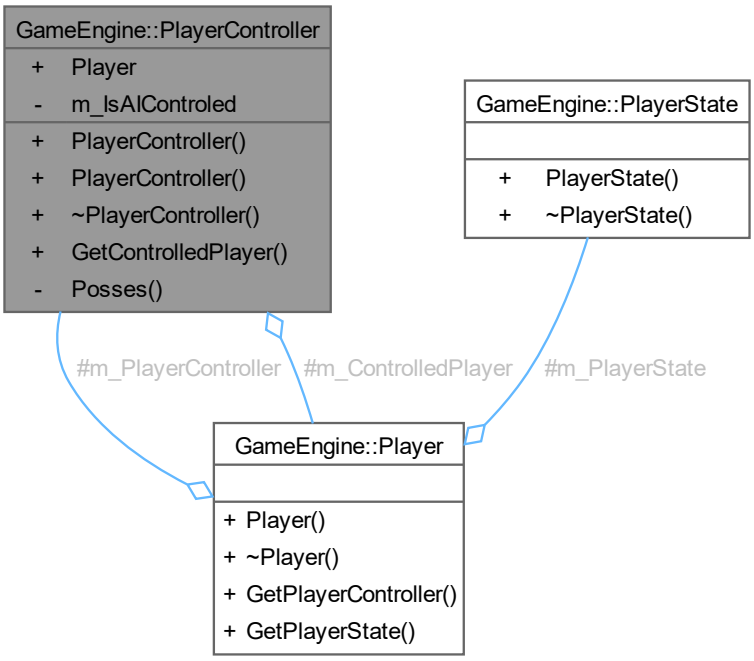


Diagrama de colaboração para GameEngine::PlayerController:



Membros Públicos

- PlayerController ()
- PlayerController (bool isAIControlled)
- virtual ~PlayerController ()
- Player * GetControlledPlayer ()

Atributos Públicos

- friend Player

Atributos Protegidos

- Player * m_ControlledPlayer

Membros privados

- void Posses (Player *pPlayer)

Atributos Privados

- bool m_IsAIControled

6.11.1 Descrição detalhada

Classe base que representa um jogador.

Ela e responsavel por controlar um Player atraves de inputs recebidos, pode se considerar como cerebro do Player.

Definição na linha 12 do arquivo PlayerController.h.

6.11.2 Construtores e Destrutores

6.11.2.1 PlayerController() [1/2]

```
PlayerController::PlayerController ( )
```

Definição na linha 6 do arquivo PlayerController.cpp.

6.11.2.2 PlayerController() [2/2]

```
PlayerController::PlayerController (
            bool isAIControlled )
```

Definição na linha 12 do arquivo PlayerController.cpp.

6.11.2.3 ~PlayerController()

```
PlayerController::~PlayerController ( )  [virtual]
```

Definição na linha 18 do arquivo PlayerController.cpp.

6.11.3 Documentação das funções

6.11.3.1 GetControlledPlayer()

```
Player * PlayerController::GetControlledPlayer ( )
```

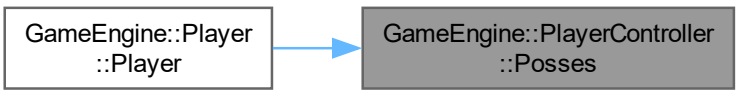
Definição na linha 31 do arquivo PlayerController.cpp.

6.11.3.2 Posses()

```
void PlayerController::Posses (
            Player * pPlayer )  [private]
```

Definição na linha 26 do arquivo PlayerController.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.11.4 Atributos

6.11.4.1 m_ControlledPlayer

```
Player* GameEngine::PlayerController::m_ControlledPlayer  [protected]
```

Definição na linha 19 do arquivo PlayerController.h.

6.11.4.2 m_IsAIControled

```
bool GameEngine::PlayerController::m_IsAIControled  [private]
```

Definição na linha 15 do arquivo PlayerController.h.

6.11.4.3 Player

```
friend GameEngine::PlayerController::Player
```

Definição na linha 28 do arquivo PlayerController.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/PlayerController.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/PlayerController.cpp

6.12 Referência da Classe GameEngine::PlayerState

Classe base responsavel por gerenciar o estado do Player durante as partidas.

```
#include <PlayerState.h>
```

Diagrama de hierarquia da classe GameEngine::PlayerState:

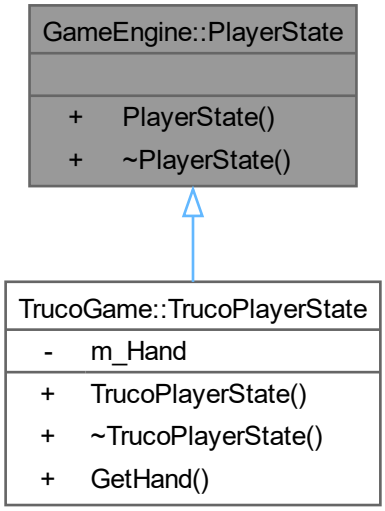
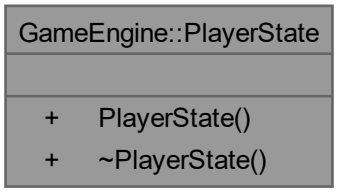


Diagrama de colaboração para GameEngine::PlayerState:



Membros Públicos

- PlayerState ()
- virtual ~PlayerState ()=default

6.12.1 Descrição detalhada

Classe base responsavel por gerenciar o estado do Player durante as partidas.

Aqui teremos as informacoes que sao relevantes apenas para a partida em curso.

Definição na linha 10 do arquivo PlayerState.h.

6.12.2 Construtores e Destrutores

6.12.2.1 PlayerState()

```
PlayerState::PlayerState ( )
```

Definição na linha 5 do arquivo PlayerState.cpp.

6.12.2.2 ~PlayerState()

```
virtual GameEngine::PlayerState::~PlayerState ( )  [virtual], [default]
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

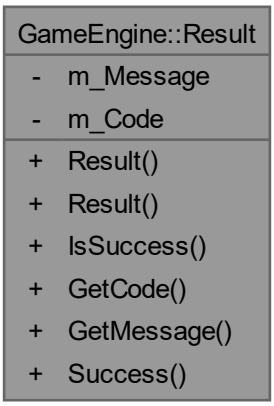
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/PlayerState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/PlayerState.cpp

6.13 Referência da Classe GameEngine::Result

Classe que representa o resultado de uma operacao que pode falhar de forma aceitavel.

`#include <Result.h>`

Diagrama de colaboração para GameEngine::Result:



Membros Públicos

- Result ()
- Result (ResultCode code, std::string message)
- bool IsSuccess ()
- ResultCode GetCode ()
- std::string GetMessage ()

Membros públicos estáticos

- static Result Success ()

Atributos Privados

- std::string m_Message
- ResultCode m_Code

6.13.1 Descrição detalhada

Classe que representa o resultado de uma operacao que pode falhar de forma aceitavel.

Definição na linha 21 do arquivo Result.h.

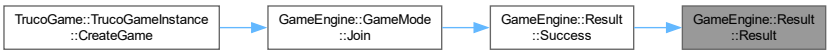
6.13.2 Construtores e Destrutores

6.13.2.1 Result() [1/2]

```
GameEngine::Result::Result ( )  [inline]
```

Definição na linha 28 do arquivo Result.h.

Esse é o diagrama das funções que utilizam essa função:



6.13.2.2 Result() [2/2]

```
GameEngine::Result::Result (
            ResultCode code,
            std::string message )  [inline]
```

Definição na linha 29 do arquivo Result.h.

6.13.3 Documentação das funções

6.13.3.1 GetCode()

```
ResultCode GameEngine::Result::GetCode ( )  [inline]
```

Definição na linha 34 do arquivo Result.h.

6.13.3.2 GetMessage()

```
std::string GameEngine::Result::GetMessage ( )  [inline]
```

Definição na linha 35 do arquivo Result.h.

6.13.3.3 IsSuccess()

```
bool GameEngine::Result::IsSuccess ( )  [inline]
```

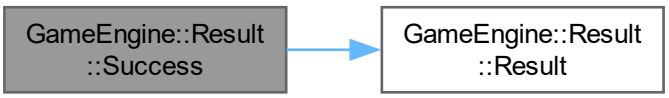
Definição na linha 33 do arquivo Result.h.

6.13.3.4 Success()

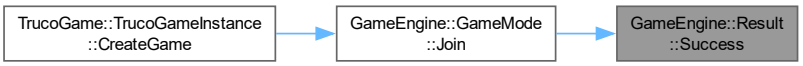
```
static Result GameEngine::Result::Success ( )  [inline], [static]
```

Definição na linha 31 do arquivo Result.h.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



6.13.4 Atributos

6.13.4.1 m_Code

```
ResultCode GameEngine::Result::m_Code  [private]
```

Definição na linha 25 do arquivo Result.h.

6.13.4.2 m_Message

```
std::string GameEngine::Result::m_Message  [private]
```

Definição na linha 24 do arquivo Result.h.

A documentação para essa classe foi gerada a partir do seguinte arquivo:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/Result.h

6.14 Referência da Classe TrucoGame::TrucoAIPlayer

Especializacao da classe AIPlayer para um jogo de truco.

`#include <TrucoAIPlayer.h>`

Diagrama de hierarquia da classe TrucoGame::TrucoAIPlayer:

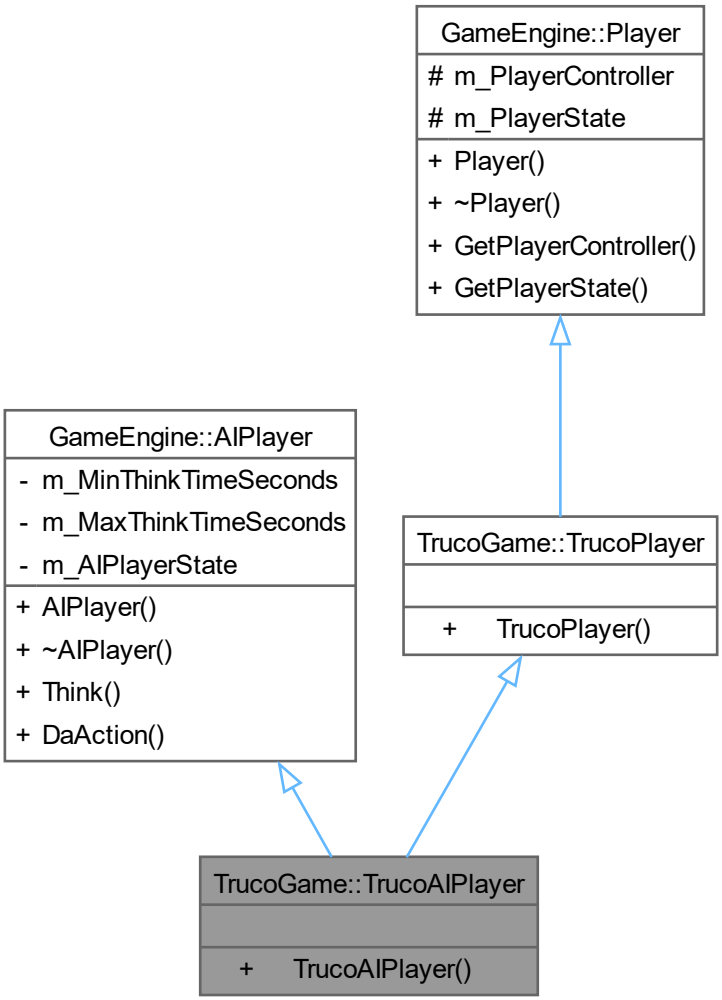
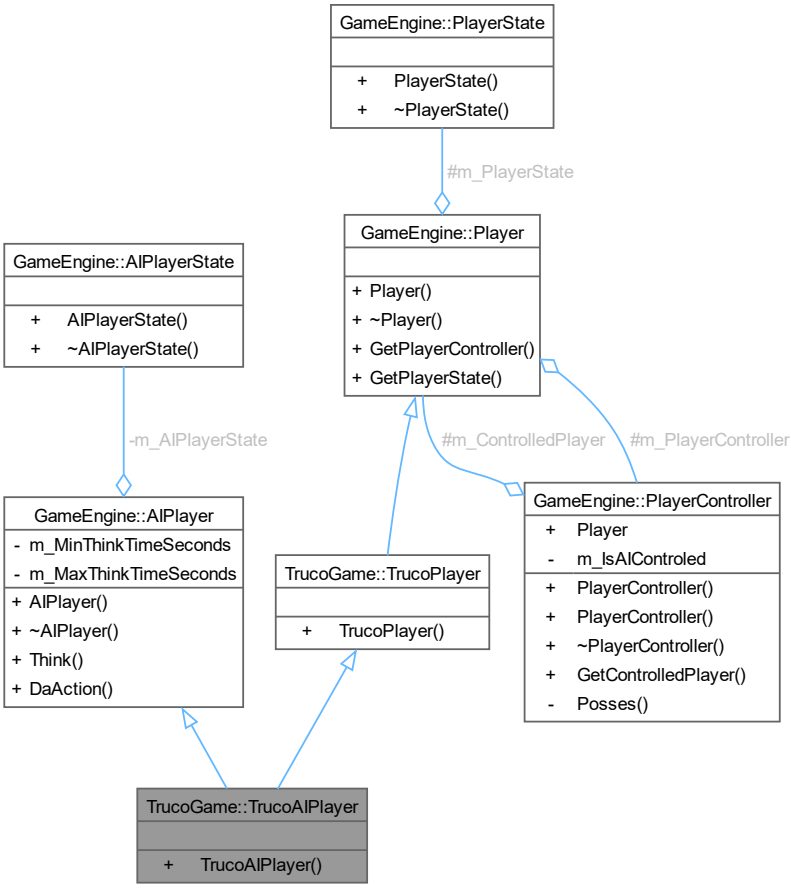


Diagrama de colaboração para TrucoGame::TrucoAIPlayer:



Membros Públicos

- TrucoAIPlayer (TrucoPlayerState ∗pPlayerState, TrucoAIPlayerState ∗pAIPlayerState, GameEngine::AIPlayerController ∗pAIPlayerController, double minTinkTimeSec, double maxThinkTimeSec)

Membros Públicos herdados de GameEngine::AIPlayer

- AIPlayer (AIPlayerState ∗aiPlayerState, double minThinkTimeSec, double maxThinkTimeSec)
- virtual ∼AIPlayer ()
- virtual void Think ()
- virtual void DaAction ()

Membros Públicos herdados de TrucoGame::TrucoPlayer

- TrucoPlayer (TrucoPlayerState ∗pTrucoPlayerState, GameEngine::PlayerController ∗pPlayerController)

Membros Públicos herdados de GameEngine::Player

- Player (PlayerState *pPlayerState, PlayerController *pPlayerController)
- virtual ~Player ()
- PlayerController * GetPlayerController ()
- PlayerState * GetPlayerState ()

Outros membros herdados

Atributos Protegidos herdados de GameEngine::Player

- PlayerController * m_PlayerController
- PlayerState * m_PlayerState

6.14.1 Descrição detalhada

Especializacao da classe AIPlayer para um jogo de truco.

Aqui teremos as informacoes sobre o avatar que nao mudam entre uma partida e outra.

Definição na linha 16 do arquivo TrucoAIPlayer.h.

6.14.2 Construtores e Destrutores

6.14.2.1 TrucoAIPlayer()

```
TrucoAIPlayer::TrucoAIPlayer (
            TrucoPlayerState * pPlayerState,
            TrucoAIPlayerState * pAIPlayerState,
            GameEngine::AIPlayerController * pAIPlayerController,
            double minTinkTimeSec,
            double maxThinkTimeSec )
```

Definição na linha 7 do arquivo TrucoAIPlayer.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoAIPlayer.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoAIPlayer.cpp

6.15 Referência da Classe TrucoGame::TrucoAIPlayerState

Especializacao da classe AIPlayerState para um jogo de truco.

`#include <TrucoAIPlayerState.h>`

Diagrama de hierarquia da classe TrucoGame::TrucoAIPlayerState:

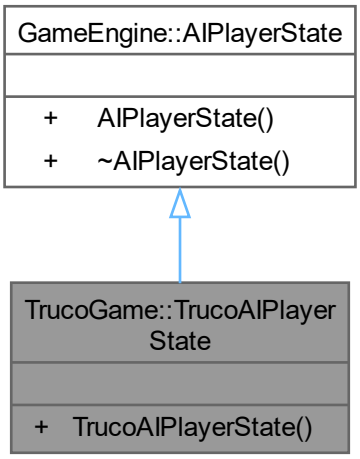
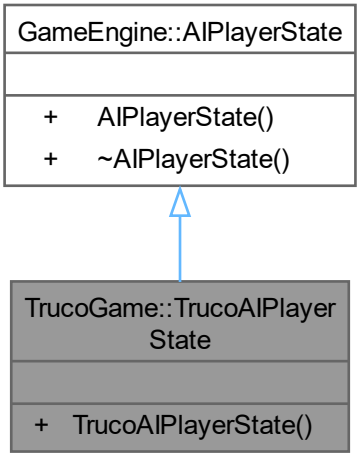


Diagrama de colaboração para TrucoGame::TrucoAIPlayerState:



Membros Públicos

- TrucoAIPlayerState ()

Membros Públicos herdados de GameEngine::AIPlayerState

- AIPlayerState ()
- virtual ~AIPlayerState ()=default

6.15.1 Descrição detalhada

Especializacao da classe AIPlayerState para um jogo de truco.

Aqui teremos informacoes pertinentes ao estado da AI durante as partidas.

Definição na linha 12 do arquivo TrucoAIPlayerState.h.

6.15.2 Construtores e Destrutores

6.15.2.1 TrucoAIPlayerState()

```
TrucoAIPlayerState::TrucoAIPlayerState ( )
```

Definição na linha 5 do arquivo TrucoAIPlayerState.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoAIPlayerState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoAIPlayerState.cpp

6.16 Referência da Classe TrucoGame::TrucoGameInstance

Especializacao da classe GameInstance para um jogo de truco.

```
#include <TrucoGameInstance.h>
```

Diagrama de hierarquia da classe TrucoGame::TrucoGameInstance:

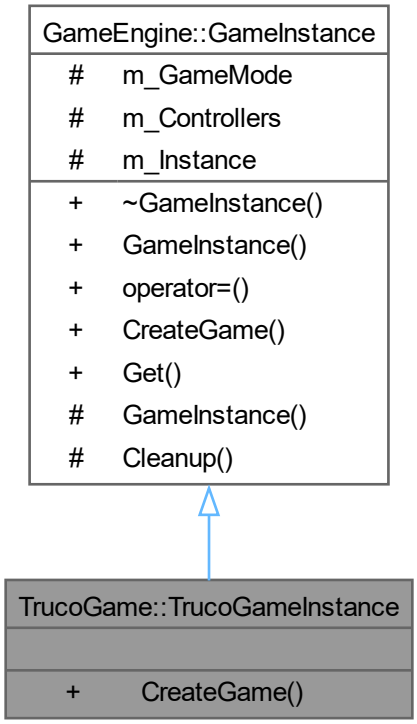
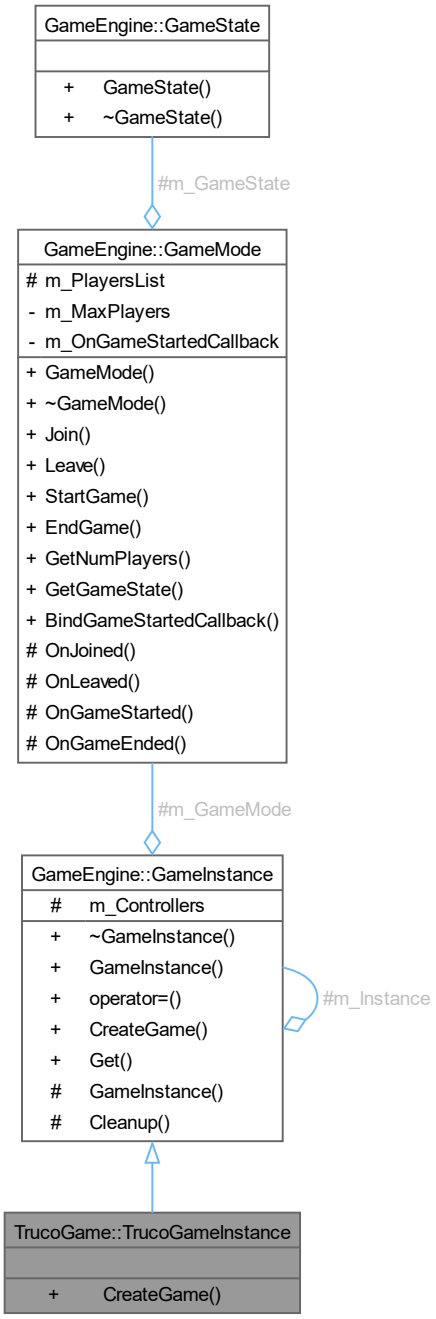


Diagrama de colaboração para TrucoGame::TrucoGameInstance:



Membros Públicos

- void CreateGame (int numPlayers, int numAIPlayers, GameEngine::GameMode ∗pGameMode) override

Membros Públicos herdados de GameEngine::GameInstance

- ∼GameInstance ()

- GameInstance (GameInstance &other)=delete
- void operator= (const GameInstance &)=delete

Outros membros herdados

Membros públicos estáticos herdados de GameEngine::GameInstance

- static GameInstance * Get ()

Membros protegidos herdados de GameEngine::GameInstance

- GameInstance ()
- void Cleanup ()

Atributos Protegidos herdados de GameEngine::GameInstance

- GameMode * m_GameMode
- std::vector< PlayerController * > m_Controllers

Atributos Protegidos Estáticos herdados de GameEngine::GameInstance

- static GameInstance * m_Instance = nullptr

6.16.1 Descrição detalhada

Expecializacao da classe GameInstance para um jogo de truco.

Aqui teremos controle sobre o inicio de partidas, acesso aos jogadores, configuracao de jogo, e etc. E o ponto de entrada de gerenciamento e configuracao das partidas.

Definição na linha 13 do arquivo TrucoGameInstance.h.

6.16.2 Documentação das funções

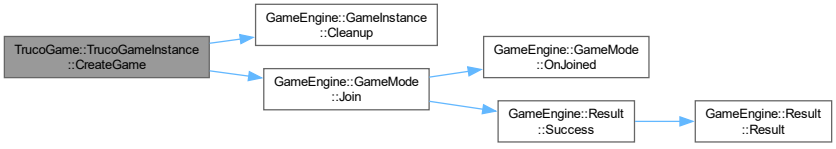
6.16.2.1 CreateGame()

```
void TrucoGameInstance::CreateGame (
            int numPlayers,
            int numAIPlayers,
            GameEngine::GameMode * pGameMode )  [override], [virtual]
```

Reimplementa GameEngine::GameInstance.

Definição na linha 12 do arquivo TrucoGameInstance.cpp.

Este é o diagrama das funções utilizadas por essa função:



A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoGameInstance.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoGameInstance.cpp

6.17 Referência da Classe TrucoGame::TrucoGameMode

Especializacao da classe GameMode para um jogo de truco.

`#include <TrucoGameMode.h>`

Diagrama de hierarquia da classe TrucoGame::TrucoGameMode:

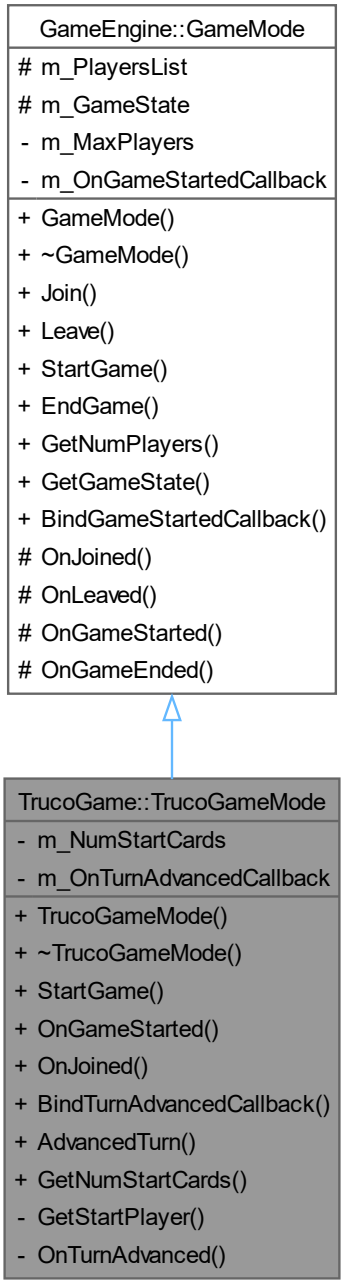
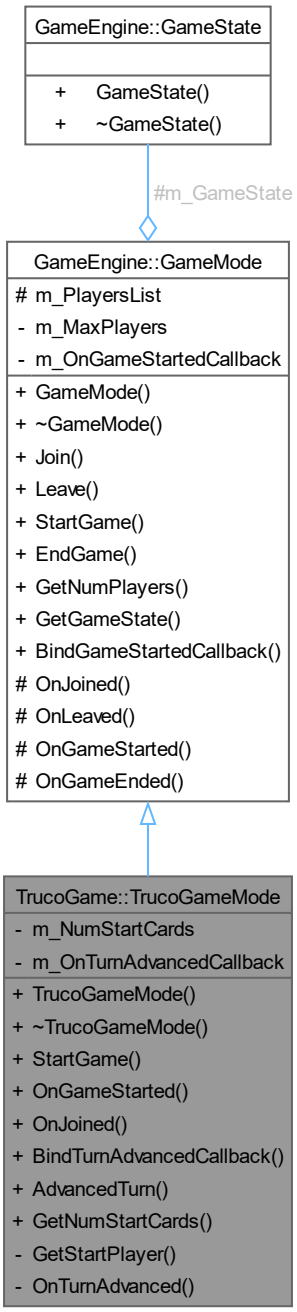


Diagrama de colaboração para TrucoGame::TrucoGameMode:



Membros Públicos

- TrucoGameMode (int numPlayers, TrucoGameState ∗gameState)
- ∼TrucoGameMode ()
- void StartGame () override
- void OnGameStarted () override
- void OnJoined (GameEngine::PlayerController ∗pPlayerController, bool isAIControlled=false) override

- void BindTurnAdvancedCallback (std::function< void(TrucoPlayer *)> callback)
- void AdvancedTurn (TrucoPlayer *turnPlayer)
- int GetNumStartCards ()

Membros Públicos herdados de GameEngine::GameMode

- GameMode (int maxPlayers, GameState *pGameState)
- virtual ~GameMode ()
- Result Join (PlayerController *pController, bool isAIControlled=false)
- void Leave (PlayerController *pController)
- virtual void EndGame ()
- int GetNumPlayers ()
- GameState * GetGameState ()
- void BindGameStartedCallback (std::function< void(void)> func)

Membros privados

- TrucoPlayer * GetStartPlayer ()
- void OnTurnAdvanced (TrucoPlayer *turnPlayer)

Atributos Privados

- int m_NumStartCards = 3
- std::function< void(TrucoPlayer *) m_OnTurnAdvancedCallback)

Outros membros herdados

Membros protegidos herdados de GameEngine::GameMode

- virtual void OnLeaved (PlayerController *pController)
- virtual void OnGameEnded ()

Atributos Protegidos herdados de GameEngine::GameMode

- std::vector< PlayerController * > m_PlayersList
- GameState * m_GameState

6.17.1 Descrição detalhada

Especializacao da classe GameMode para um jogo de truco.

Definição na linha 14 do arquivo TrucoGameMode.h.

6.17.2 Construtores e Destrutores

6.17.2.1 TrucoGameMode()

```
TrucoGameMode::TrucoGameMode (
            int numPlayers,
            TrucoGameState * gameState )
```

Definição na linha 28 do arquivo TrucoGameMode.cpp.

6.17.2.2 ~TrucoGameMode()

```
TrucoGameMode::~TrucoGameMode ( )
```

Definição na linha 33 do arquivo TrucoGameMode.cpp.

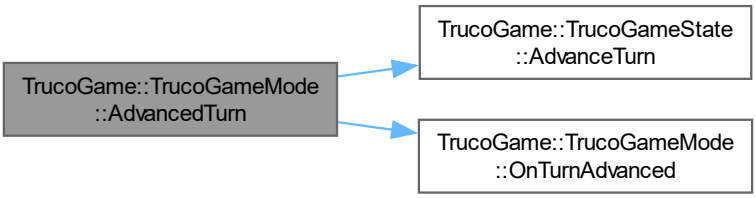
6.17.3 Documentação das funções

6.17.3.1 AdvancedTurn()

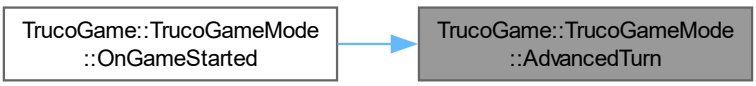
```
void TrucoGameMode::AdvancedTurn (
            TrucoPlayer * turnPlayer )
```

Definição na linha 121 do arquivo TrucoGameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



Esse é o diagrama das funções que utilizam essa função:



6.17.3.2 BindTurnAdvancedCallback()

```
void TrucoGameMode::BindTurnAdvancedCallback (
            std::function< void(TrucoPlayer *)> callback )
```

Definição na linha 116 do arquivo TrucoGameMode.cpp.

6.17.3.3 GetNumStartCards()

```
int TrucoGame::TrucoGameMode::GetNumStartCards ( )  [inline]
```

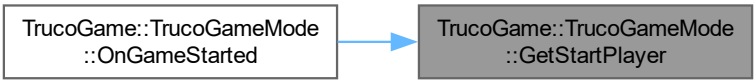
Definição na linha 35 do arquivo TrucoGameMode.h.

6.17.3.4 GetStartPlayer()

```
TrucoPlayer * TrucoGameMode::GetStartPlayer ( )  [private]
```

Definição na linha 16 do arquivo TrucoGameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:



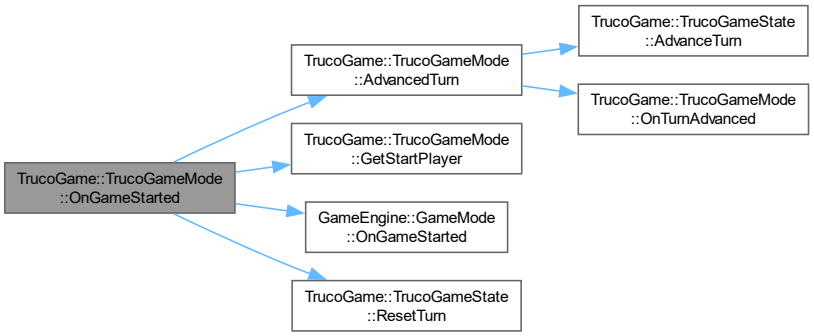
6.17.3.5 OnGameStarted()

```
void TrucoGameMode::OnGameStarted ( )  [override], [virtual]
```

Reimplementa GameEngine::GameMode.

Definição na linha 77 do arquivo TrucoGameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



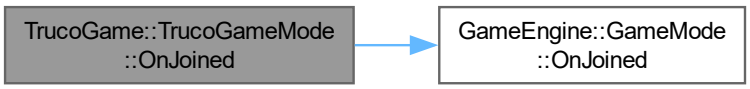
6.17.3.6 OnJoined()

```
void TrucoGameMode::OnJoined (
            GameEngine::PlayerController * pPlayerController,
            bool isAIControlled = false )  [override], [virtual]
```

Reimplementa GameEngine::GameMode.

Definição na linha 93 do arquivo TrucoGameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:

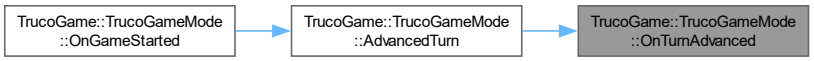


6.17.3.7 OnTurnAdvanced()

```
void TrucoGameMode::OnTurnAdvanced (
            TrucoPlayer * turnPlayer )  [private]
```

Definição na linha 108 do arquivo TrucoGameMode.cpp.

Esse é o diagrama das funções que utilizam essa função:



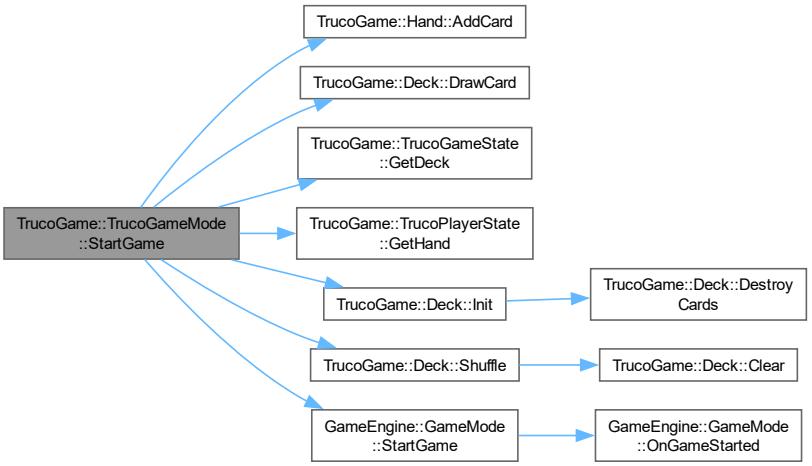
6.17.3.8 StartGame()

```
void TrucoGameMode::StartGame ( )  [override], [virtual]
```

Reimplementa GameEngine::GameMode.

Definição na linha 37 do arquivo TrucoGameMode.cpp.

Este é o diagrama das funções utilizadas por essa função:



6.17.4 Atributos

6.17.4.1 m_NumStartCards

```
int TrucoGame::TrucoGameMode::m_NumStartCards = 3  [private]
```

Definição na linha 17 do arquivo TrucoGameMode.h.

6.17.4.2 m_OnTurnAdvancedCallback

```
std::function<void(TrucoPlayer*) TrucoGame::TrucoGameMode::m_OnTurnAdvancedCallback)  [private]
```

Definição na linha 20 do arquivo TrucoGameMode.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoGameMode.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoGameMode.cpp

6.18 Referência da Classe TrucoGame::TrucoGameState

Especializacao da classe GameState para um jogo de truco.

```
#include <TrucoGameState.h>
```

Diagrama de hierarquia da classe TrucoGame::TrucoGameState:

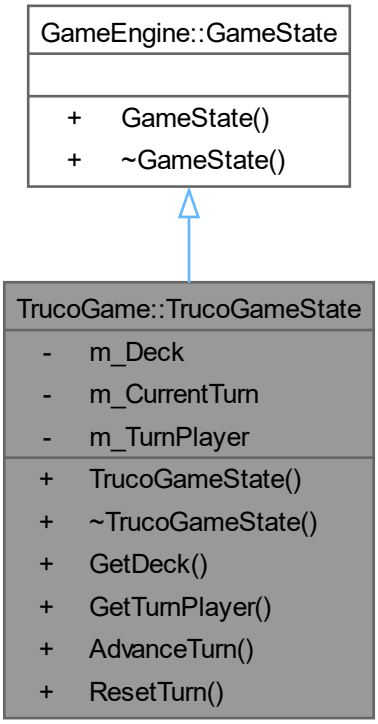
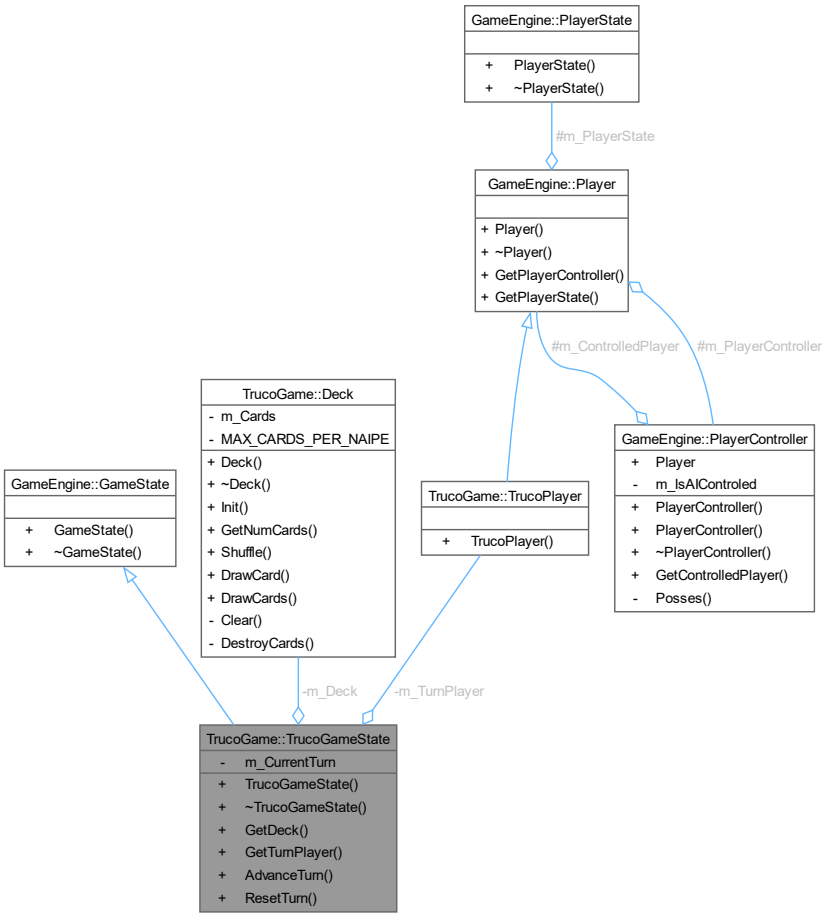


Diagrama de colaboração para TrucoGame::TrucoGameState:



Membros Públicos

- TrucoGameState (Deck ∗deck)
- ∼TrucoGameState ()
- Deck ∗ GetDeck ()
- TrucoPlayer ∗ GetTurnPlayer ()
- void AdvanceTurn (TrucoPlayer ∗turnPlayer)
- void ResetTurn ()

Membros Públicos herdados de GameEngine::GameState

- GameState ()
- virtual ∼GameState ()=default

Atributos Privados

- Deck ∗ m_Deck
- int m_CurrentTurn
- TrucoPlayer ∗ m_TurnPlayer

6.18.1 Descrição detalhada

Especializacao da classe GameState para um jogo de truco.

Aqui teremos as informacoes que sao relevantes apenas para a partida em curso.

Definição na linha 15 do arquivo TrucoGameState.h.

6.18.2 Construtores e Destrutores

6.18.2.1 TrucoGameState()

```
TrucoGameState::TrucoGameState (
            Deck * deck )
```

Definição na linha 6 do arquivo TrucoGameState.cpp.

6.18.2.2 ~TrucoGameState()

```
TrucoGameState::~TrucoGameState ( )
```

Definição na linha 12 do arquivo TrucoGameState.cpp.

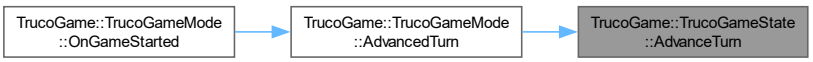
6.18.3 Documentação das funções

6.18.3.1 AdvanceTurn()

```
void TrucoGameState::AdvanceTurn (
            TrucoPlayer * turnPlayer )
```

Definição na linha 30 do arquivo TrucoGameState.cpp.

Esse é o diagrama das funções que utilizam essa função:

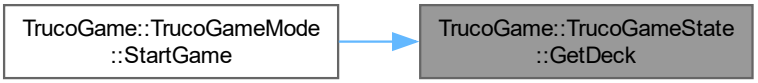


6.18.3.2 GetDeck()

```
Deck * TrucoGameState::GetDeck ( )
```

Definição na linha 20 do arquivo TrucoGameState.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.18.3.3 GetTurnPlayer()

```
TrucoPlayer * TrucoGameState::GetTurnPlayer ( )
```

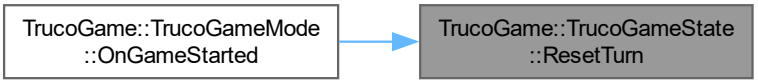
Definição na linha 25 do arquivo TrucoGameState.cpp.

6.18.3.4 ResetTurn()

```
void TrucoGameState::ResetTurn ( )
```

Definição na linha 36 do arquivo TrucoGameState.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.18.4 Atributos

6.18.4.1 m_CurrentTurn

```
int TrucoGame::TrucoGameState::m_CurrentTurn  [private]
```

Definição na linha 19 do arquivo TrucoGameState.h.

6.18.4.2 m_Deck

```
Deck* TrucoGame::TrucoGameState::m_Deck  [private]
```

Definição na linha 18 do arquivo TrucoGameState.h.

6.18.4.3 m_TurnPlayer

```
TrucoPlayer* TrucoGame::TrucoGameState::m_TurnPlayer  [private]
```

Definição na linha 21 do arquivo TrucoGameState.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoGameState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoGameState.cpp

6.19 Referência da Classe TrucoGame::TrucoPlayer

Especializacao da classe Player para um jogo de truco.

```
#include <TrucoPlayer.h>
```

Diagrama de hierarquia da classe TrucoGame::TrucoPlayer:

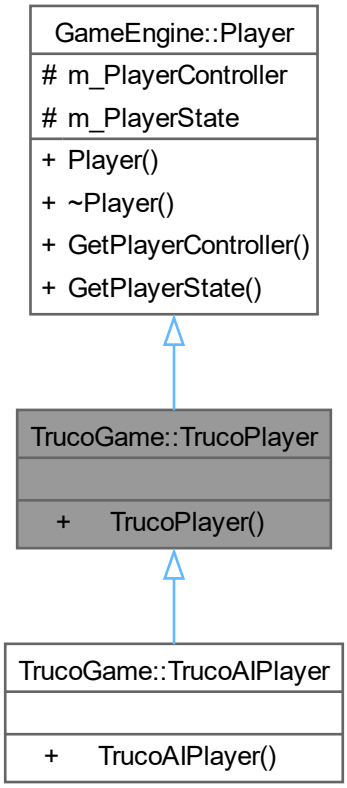
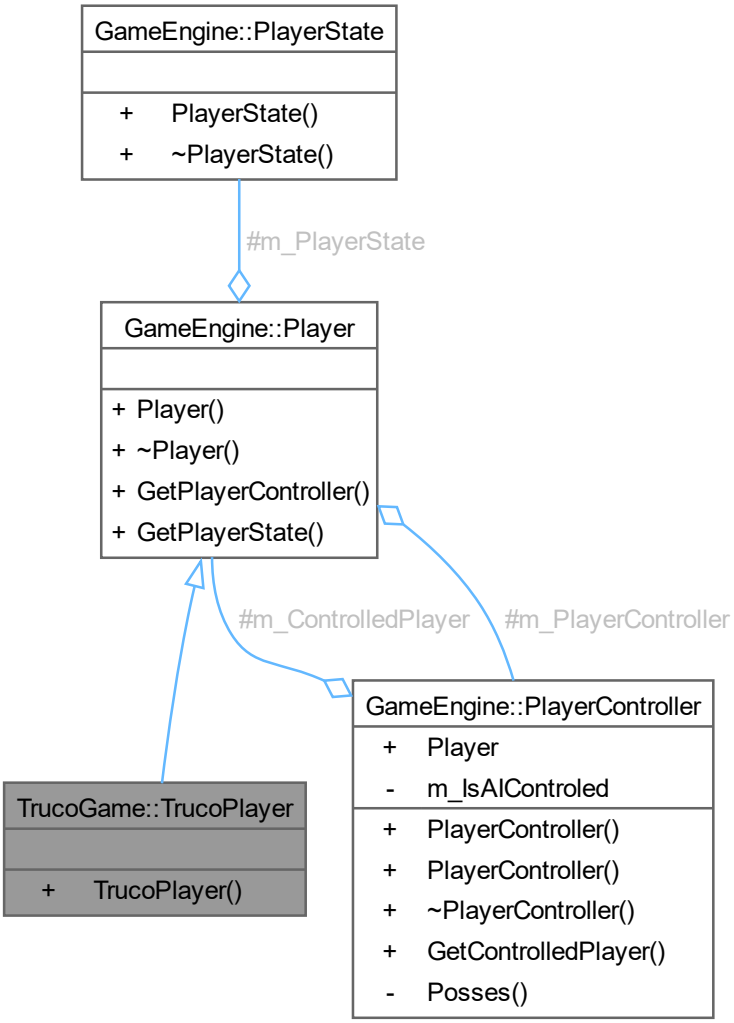


Diagrama de colaboração para TrucoGame::TrucoPlayer:



Membros Públicos

- TrucoPlayer (TrucoPlayerState ∗pTrucoPlayerState, GameEngine::PlayerController ∗pPlayerController)

Membros Públicos herdados de GameEngine::Player

- Player (PlayerState ∗pPlayerState, PlayerController ∗pPlayerController)
- virtual ∼Player ()
- PlayerController ∗ GetPlayerController ()
- PlayerState ∗ GetPlayerState ()

Outros membros herdados

Atributos Protegidos herdados de GameEngine::Player

- PlayerController * m_PlayerController
- PlayerState * m_PlayerState

6.19.1 Descrição detalhada

Especializacao da classe Player para um jogo de truco.

Aqui teremos as informacoes sobre o jogador que nao mudam entre uma partida e outra.

Definição na linha 14 do arquivo TrucoPlayer.h.

6.19.2 Construtores e Destrutores

6.19.2.1 TrucoPlayer()

```
TrucoPlayer::TrucoPlayer (
            TrucoPlayerState * pTrucoPlayerState,
            GameEngine::PlayerController * pPlayerController )
```

Definição na linha 7 do arquivo TrucoPlayer.cpp.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoPlayer.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoPlayer.cpp

6.20 Referência da Classe TrucoGame::TrucoPlayerState

Especializacao da classe PlayerState para uma partida de truco.

```
#include <TrucoPlayerState.h>
```

Diagrama de hierarquia da classe TrucoGame::TrucoPlayerState:

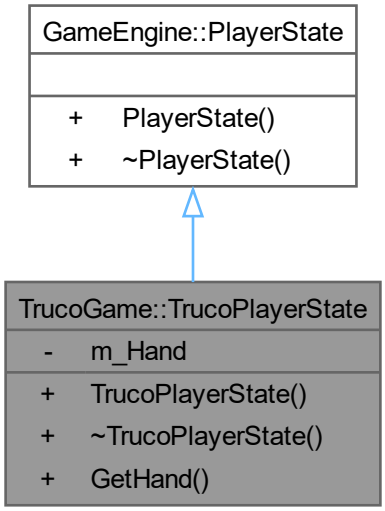
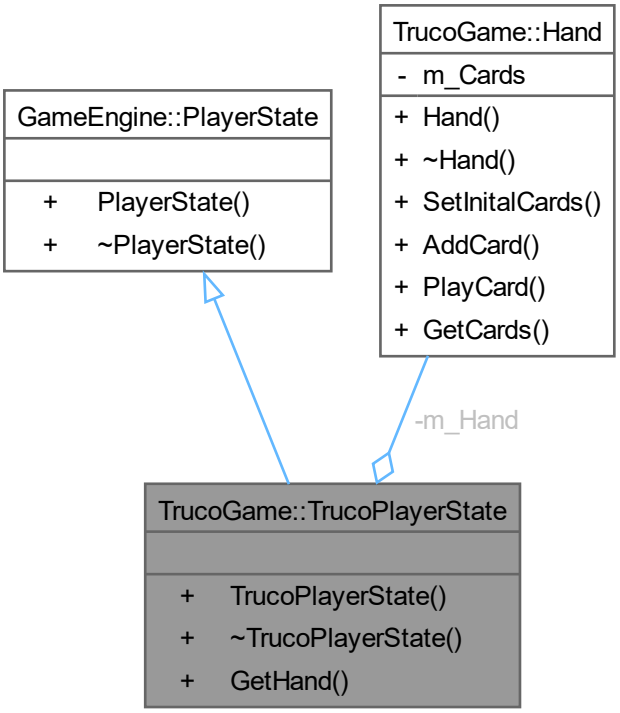


Diagrama de colaboração para TrucoGame::TrucoPlayerState:



Membros Públicos

- TrucoPlayerState (Hand *hand)
- ~TrucoPlayerState ()
- Hand * GetHand ()

Membros Públicos herdados de GameEngine::PlayerState

- PlayerState ()
- virtual ~PlayerState ()=default

Atributos Privados

- Hand * m_Hand

6.20.1 Descrição detalhada

Especializacao da classe PlayerState para uma partida de truco.

Aqui teremos as informacoes que sao relevantes apenas para a partida em curso.

Definição na linha 14 do arquivo TrucoPlayerState.h.

6.20.2 Construtores e Destrutores

6.20.2.1 TrucoPlayerState()

```
TrucoPlayerState::TrucoPlayerState (
            Hand * hand )
```

Definição na linha 6 do arquivo TrucoPlayerState.cpp.

6.20.2.2 ~TrucoPlayerState()

```
TrucoPlayerState::~TrucoPlayerState ( )
```

Definição na linha 11 do arquivo TrucoPlayerState.cpp.

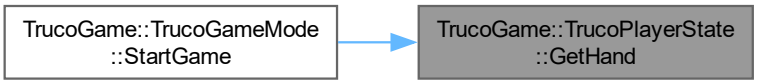
6.20.3 Documentação das funções

6.20.3.1 GetHand()

```
Hand * TrucoPlayerState::GetHand ( )
```

Definição na linha 19 do arquivo TrucoPlayerState.cpp.

Esse é o diagrama das funções que utilizam essa função:



6.20.4 Atributos

6.20.4.1 m_Hand

```
Hand* TrucoGame::TrucoPlayerState::m_Hand  [private]
```

Definição na linha 17 do arquivo TrucoPlayerState.h.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoPlayerState.h
- C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoPlayerState.cpp

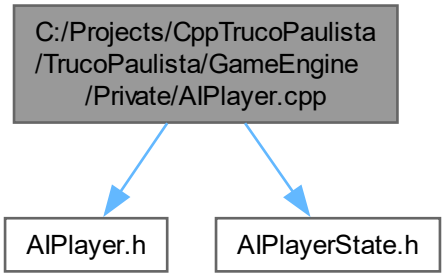
Capítulo 7

Arquivos

7.1 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayer.cpp

```
#include "AIPlayer.h"
#include "AIPlayerState.h"
```

Gráfico de dependência de inclusões para AIPlayer.cpp:



7.2 AIPlayer.cpp

Ir para a documentação desse arquivo.

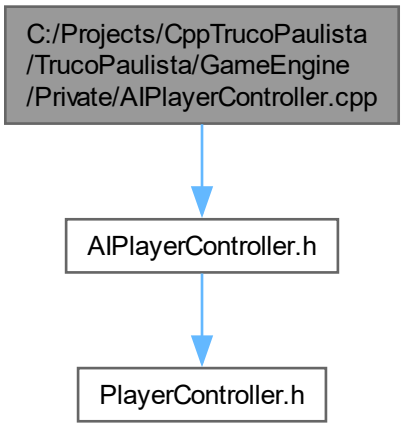
```
00001 #include "AIPlayer.h"
00002 #include "AIPlayerState.h"
00003
00004 using namespace GameEngine;
00005
00006 AIPlayer::AIPlayer(AIPlayerState* aiPlayerState, double minThinkTimeSec, double maxThinkTimeSec)
00007 {
00008     m_AIPlayerState = aiPlayerState;
00009
00010     m_MinThinkTimeSeconds = minThinkTimeSec;
00011     m_MaxThinkTimeSeconds = maxThinkTimeSec;
00012 }
00013
00014 AIPlayer::~AIPlayer()
```

```
00015 {
00016     if (m_AIPlayerState)
00017     {
00018         delete m_AIPlayerState;
00019     }
00020 }
00021
00022 void AIPlayer::Think()
00023 {
00024 }
00025
00026 void AIPlayer::DaAction()
00027 {
00028 }
```

7.3 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayerController.cpp

```
#include "AIPlayerController.h"
```

Gráfico de dependência de inclusões para AIPlayerController.cpp:



7.4 AIPlayerController.cpp

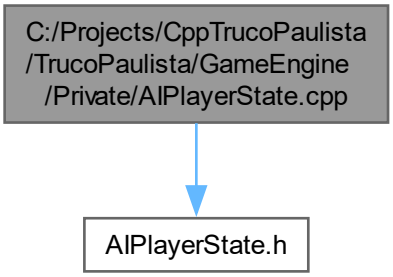
Ir para a documentação desse arquivo.

```
00001 #include "AIPlayerController.h"
00002
00003 using namespace GameEngine;
00004
00005 AIPlayerController::AIPlayerController() : PlayerController(true)
00006 {
00007 }
```

7.5 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/AIPlayerState.cpp

```
#include "AIPlayerState.h"
```

Gráfico de dependência de inclusões para AIPlayerState.cpp:



7.6 AIPlayerState.cpp

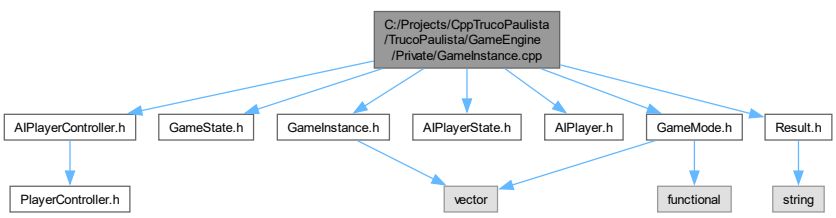
Ir para a documentação desse arquivo.

```
00001 #include "AIPlayerState.h"
00002
00003 using namespace GameEngine;
00004
00005 AIPlayerState::AIPlayerState()
00006 {
00007 }
```

7.7 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameInstance.cpp

```
#include "GameInstance.h"
#include "GameState.h"
#include "AIPlayerController.h"
#include "AIPlayerState.h"
#include "AIPlayer.h"
#include "Result.h"
#include "GameMode.h"
```

Gráfico de dependência de inclusões para GameInstance.cpp:



7.8 GameInstance.cpp

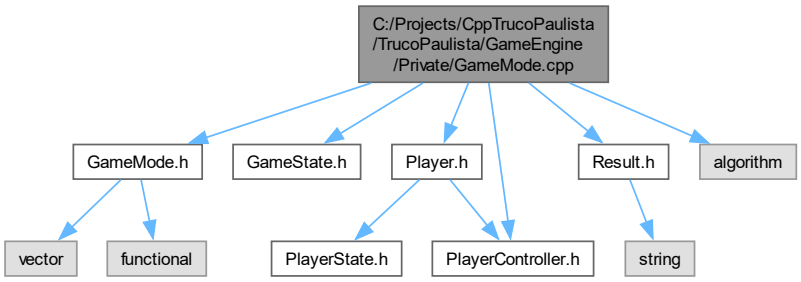
Ir para a documentação desse arquivo.

```
00001 #include "GameInstance.h"
00002 #include "GameState.h"
00003 #include "AIPlayerController.h"
00004 #include "AIPlayerState.h"
00005 #include "AIPlayer.h"
00006 #include "Result.h"
00007 #include "GameMode.h"
00008
00009 using namespace GameEngine;
00010
00011 GameInstance* GameInstance::m_Instance = nullptr;
00012
00013 void GameInstance::Cleanup()
00014 {
00015     if (m_GameMode)
00016     {
00017         delete m_GameMode;
00018     }
00019
00020     std::vector<PlayerController*>::iterator it;
00021     for (it = m_Controllers.begin(); it != m_Controllers.end(); it++)
00022     {
00023         delete* it;
00024     }
00025 }
00026
00027 GameInstance::GameInstance() : m_GameMode(nullptr)
00028 {
00029 }
00030
00031 GameInstance::~GameInstance()
00032 {
00033     Cleanup();
00034
00035     if (m_Instance != nullptr)
00036     {
00037         delete m_Instance;
00038     }
00039 }
00040
00041 GameInstance* GameInstance::Get()
00042 {
00043     if (m_Instance == nullptr)
00044     {
00045         m_Instance = new GameInstance();
00046     }
00047
00048     return m_Instance;
00049 }
00050
00051 void GameInstance::CreateGame(int numPlayers, int numAIPlayers, GameMode* pGameMode)
00052 {
00053 }
```

7.9 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameMode.cpp

```
#include "GameMode.h"
#include "GameState.h"
#include "Player.h"
#include "PlayerController.h"
#include "Result.h"
#include <algorithm>
```

Gráfico de dependência de inclusões para GameMode.cpp:



7.10 GameMode.cpp

Ir para a documentação desse arquivo.

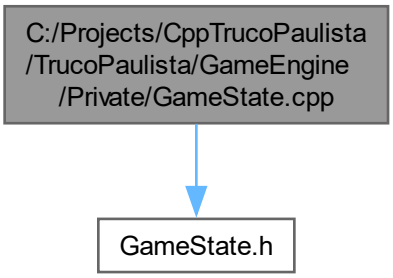
```
00001 #include "GameMode.h"
00002 #include "GameState.h"
00003 #include "Player.h"
00004 #include "PlayerController.h"
00005 #include "Result.h"
00006 #include <algorithm>
00007
00008 using namespace GameEngine;
00009
00010 GameMode::GameMode(int numPlayers, GameState* pGameState)
00011 {
00012     m_MaxPlayers = numPlayers;
00013     m_PlayersList.reserve(numPlayers);
00014     m_GameState = pGameState;
00015     m_OnGameStartedCallback = nullptr;
00016 }
00017
00018 GameMode::~GameMode()
00019 {
00020     if (m_GameState)
00021     {
00022         delete m_GameState;
00023     }
00024 }
00025
00026 void GameMode::OnJoined(PlayerController* pPlayerController, bool isAIControlled)
00027 {
00028 }
00029
00030 void GameMode::OnLeaved(PlayerController* pPlayerController)
00031 {
00032 }
00033
00034 void GameMode::OnGameStarted()
00035 {
00036     if (m_OnGameStartedCallback)
00037     {
00038         m_OnGameStartedCallback();
00039     }
00040 }
00041
00042 void GameMode::OnGameEnded()
00043 {
00044 }
00045
00046 Result GameMode::Join(PlayerController* pPlayerController, bool isAIControlled)
00047 {
00048     if (m_PlayersList.size() == m_MaxPlayers)
00049     {
00050         // TODO: Add log or warning
00051         return Result(ResultCode::GameIsFull, "This game already reached the maximum players limit.");
00052     }
00053
00054     std::vector<PlayerController*>::iterator it;
```

```
00055     it = std::find(m_PlayersList.begin(), m_PlayersList.end(), pPlayerController);
00056
00057     if (it != m_PlayersList.end())
00058     {
00059         return Result(ResultCode::PlayerAlreadyJoined, "This player is alreadi in the game.");
00060     }
00061
00062     m_PlayersList.push_back(pPlayerController);
00063
00064     OnJoined(pPlayerController, isAIControlled);
00065
00066     return Result::Success();
00067 }
00068
00069 void GameMode::Leave(PlayerController* pPlayerController)
00070 {
00071     std::vector<PlayerController*>::iterator it;
00072     it = std::find(m_PlayersList.begin(), m_PlayersList.end(), pPlayerController);
00073
00074     if (it != m_PlayersList.end())
00075     {
00076         OnLeaved(pPlayerController);
00077
00078         m_PlayersList.erase(it);
00079     }
00080 }
00081
00082 void GameMode::StartGame()
00083 {
00084     OnGameStarted();
00085 }
00086
00087 void GameMode::EndGame()
00088 {
00089     OnGameEnded();
00090 }
00091
00092 int GameMode::GetNumPlayers()
00093 {
00094     return (int)m_PlayersList.size();
00095 }
00096
00097 GameState* GameMode::GetGameState()
00098 {
00099     return m_GameState;
00100 }
00101
00102 void GameMode::BindGameStartedCallback(std::function<void(void)> func)
00103 {
00104     m_OnGameStartedCallback = func;
00105 }
```

7.11 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/GameState.cpp

```
#include "GameState.h"
```

Gráfico de dependência de inclusões para GameState.cpp:



7.12 GameState.cpp

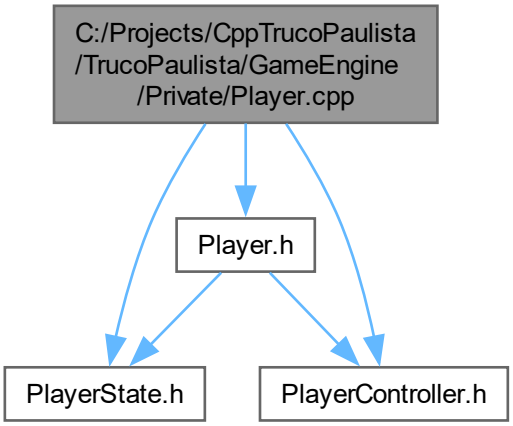
Ir para a documentação desse arquivo.

```
00001 #include "GameState.h"
00002
00003 using namespace GameEngine;
00004
00005 GameState::GameState()
00006 {
00007 }
```

7.13 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/Player.cpp

```
#include "Player.h"
#include "PlayerController.h"
#include "PlayerState.h"
```

Gráfico de dependência de inclusões para Player.cpp:



7.14 Player.cpp

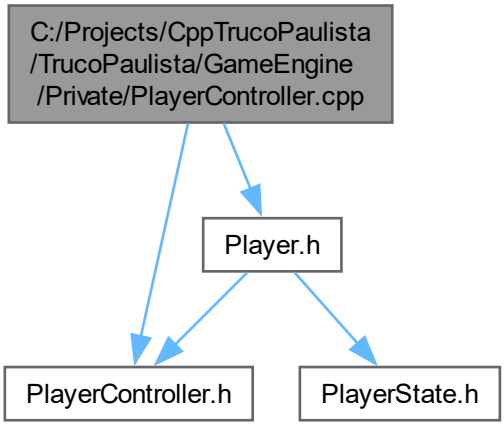
Ir para a documentação desse arquivo.

```
00001 #include "Player.h"
00002 #include "PlayerController.h"
00003 #include "PlayerState.h"
00004
00005 using namespace GameEngine;
00006
00007 Player::Player(PlayerState* pPlayerState, PlayerController* pPlayerController)
00008 {
00009     m_PlayerState = pPlayerState;
00010     m_PlayerController = pPlayerController;
00011
00012     pPlayerController->Posses(this);
00013 }
00014
00015 Player::~Player()
00016 {
00017     if (m_PlayerState)
00018     {
00019         delete m_PlayerState;
00020     }
00021 }
00022
00023 PlayerController* Player::GetPlayerController()
00024 {
00025     return m_PlayerController;
00026 }
00027
00028 PlayerState* Player::GetPlayerState()
00029 {
00030     return m_PlayerState;
00031 }
```

7.15 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/GameEngine/Private/PlayerController.cpp

```
#include "PlayerController.h"
#include "Player.h"
```

Gráfico de dependência de inclusões para PlayerController.cpp:



7.16 PlayerController.cpp

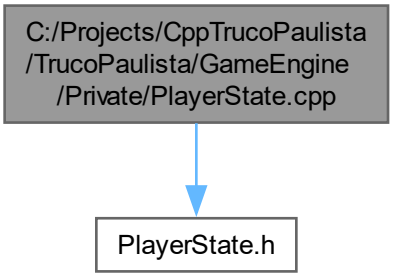
Ir para a documentação desse arquivo.

```
00001 #include "PlayerController.h"
00002 #include "Player.h"
00003
00004 using namespace GameEngine;
00005
00006 PlayerController::PlayerController() :
00007     m_IsAIControled(false),
00008     m_ControlledPlayer(nullptr)
00009 {
00010 }
00011
00012 PlayerController::PlayerController(bool isAIControlled) :
00013     m_IsAIControled(isAIControlled),
00014     m_ControlledPlayer(nullptr)
00015 {
00016 }
00017
00018 PlayerController::~PlayerController()
00019 {
00020     if (m_ControlledPlayer)
00021     {
00022         delete m_ControlledPlayer;
00023     }
00024 }
00025
00026 void PlayerController::Posses(Player* pPlayer)
00027 {
00028     m_ControlledPlayer = pPlayer;
00029 }
00030
00031 Player* PlayerController::GetControlledPlayer()
00032 {
00033     return m_ControlledPlayer;
00034 }
```

7.17 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Private/PlayerState.cpp

```
#include "PlayerState.h"
```

Gráfico de dependência de inclusões para PlayerState.cpp:



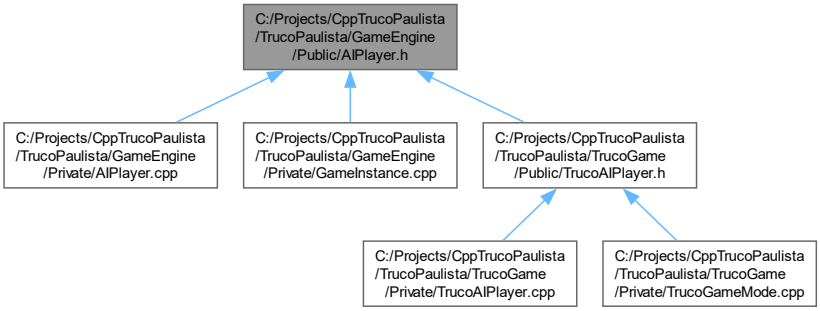
7.18 PlayerState.cpp

Ir para a documentação desse arquivo.

```
00001 #include "PlayerState.h"
00002
00003 using namespace GameEngine;
00004
00005 PlayerState::PlayerState()
00006 {
00007 }
```

7.19 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayer.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::AIPlayer

  Classe base que representa o avatar controlado por AI.

Namespaces

- namespace GameEngine

7.20 AIPlayer.h

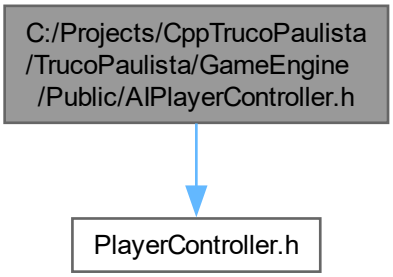
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace GameEngine
00004 {
00005     class AIPlayerState;
00006
00012     class AIPlayer
00013     {
00014     private:
00015         double m_MinThinkTimeSeconds;
00016         double m_MaxThinkTimeSeconds;
00017         AIPlayerState* m_AIPlayerState;
00018
00019     public:
00020         AIPlayer(AIPlayerState* aiPlayerState, double minThinkTimeSec, double maxThinkTimeSec);
00021         virtual ~AIPlayer();
00022
00023         virtual void Think();
00024         virtual void DaAction();
00025     };
00026 };
```

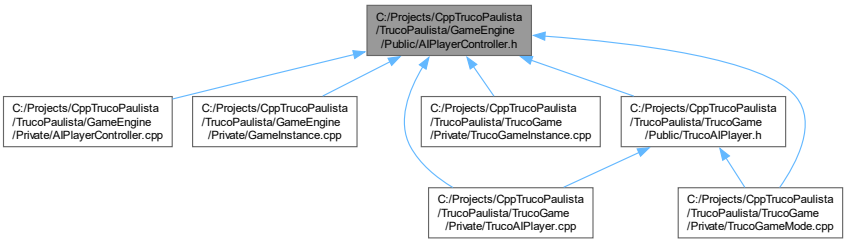
7.21 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayerController.h

```
#include "PlayerController.h"
```

Gráfico de dependência de inclusões para AIPlayerController.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::AIPlayerController

  Classe base que representa um avatar controlado por AI.

Namespaces

- namespace GameEngine

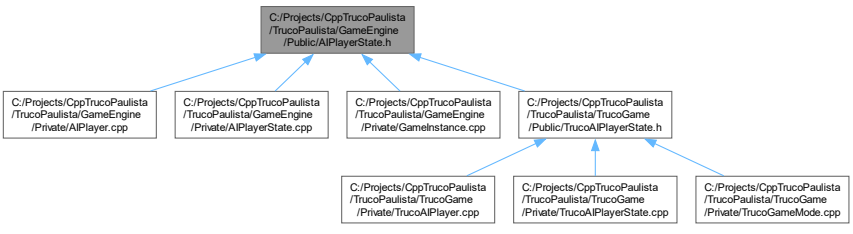
7.22 AIPlayerController.h

Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "PlayerController.h"
00004
00005 namespace GameEngine
00006 {
00013     class AIPlayerController :
00014         public PlayerController
00015     {
00016     public:
00017         AIPlayerController();
00018     };
00019 };
```

7.23 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/AIPlayerState.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::AIPlayerState

  Classe base responsavel por gerenciar o estado do Player controlado por AI.

Namespaces

- namespace GameEngine

7.24 AIPlayerState.h

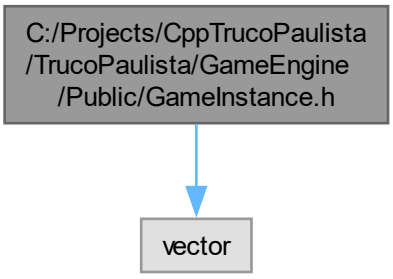
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace GameEngine
00004 {
00010     class AIPlayerState
00011     {
00012     public:
00013         AIPlayerState();
00014         virtual ~AIPlayerState() = default;
00015     };
00016 };
```

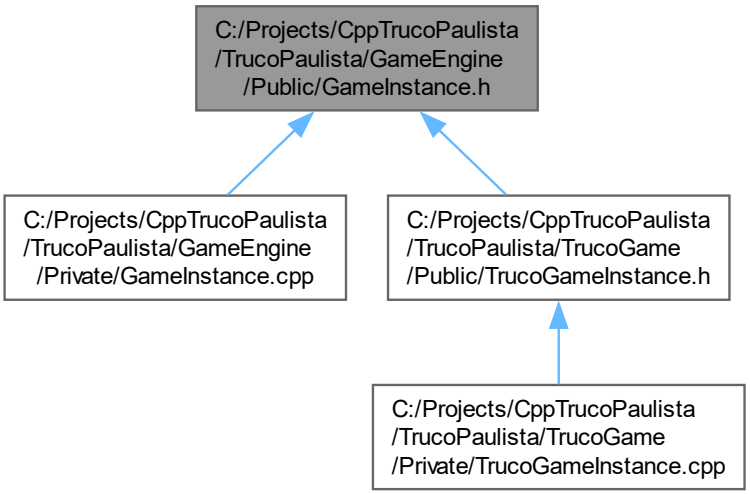
7.25 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameInstance.h

`#include <vector>`

Gráfico de dependência de inclusões para GameInstance.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::GameInstance

  Classe base que gerencia de forma geral uma instancia do jogo em execucao.

Namespaces

- namespace GameEngine

7.26 GameInstance.h

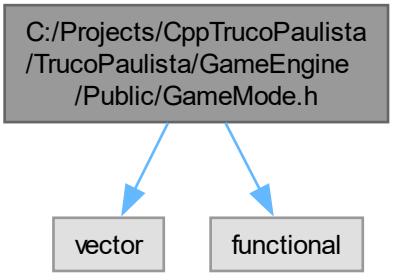
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include <vector>
00004
00005 namespace GameEngine
00006 {
00007     class GameMode;
00008     class PlayerController;
00009
00016     class GameInstance
00017     {
00018     protected:
00019         GameMode* m_GameMode;
00020         std::vector<PlayerController*> m_Controllers;
00021
00022         GameInstance();
00023         void Cleanup();
00024
00025         static GameInstance* m_Instance;
00026
00027     public:
00028         ~GameInstance();
00029         GameInstance(GameInstance& other) = delete;
00030         void operator = (const GameInstance&) = delete;
00031
00032         static GameInstance* Get();
00033
00034         virtual void CreateGame(int numPlayers, int numAIPlayers, GameMode* pGameMode);
00035     };
00036 };
```

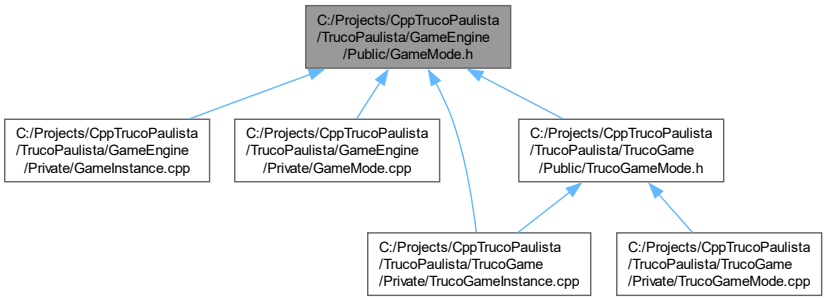
7.27 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameMode.h

```
#include <vector>
#include <functional>
```

Gráfico de dependência de inclusões para GameMode.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::GameMode

  Classe base responsavel por gerenciar as regras do jogo.

Namespaces

- namespace GameEngine

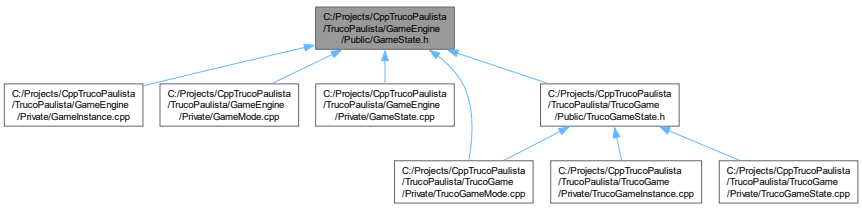
7.28 GameMode.h

Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include <vector>
00004 #include <functional>
00005
00006 namespace GameEngine
00007 {
00008     class Result;
00009     class GameState;
00010     class PlayerController;
00011
00015     class GameMode
00016     {
00017     private:
00018         int m_MaxPlayers = 1;
00019         std::function<void(void)> m_OnGameStartedCallback;
00020
00021     protected:
00022         std::vector<PlayerController*> m_PlayersList;
00023         GameState* m_GameState;
00024
00025         virtual void OnJoined(PlayerController* pController, bool isAIControlled = false);
00026         virtual void OnLeaved(PlayerController* pController);
00027
00028         virtual void OnGameStarted();
00029         virtual void OnGameEnded();
00030
00031     public:
00032         GameMode(int maxPlayers, GameState* pGameState);
00033         virtual ~GameMode();
00034
00035         Result Join(PlayerController* pController, bool isAIControlled = false);
00036         void Leave(PlayerController* pController);
00037         virtual void StartGame();
00038         virtual void EndGame();
00039
00040         int GetNumPlayers();
00041
00042         GameState* GetGameState();
00043
00044         void BindGameStartedCallback(std::function<void(void)> func);
00045     };
00046 };
```

7.29 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/GameState.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::GameState

  Classe base responsavel por gerenciar o estado do jogo.

Namespaces

- namespace GameEngine

7.30 GameState.h

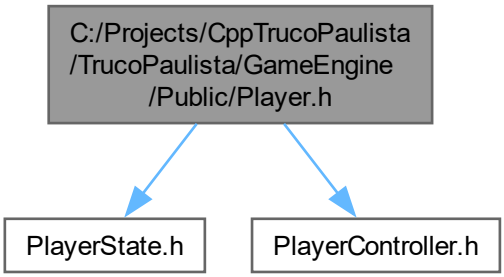
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace GameEngine
00004 {
00010     class GameState
00011     {
00012     public:
00013         GameState();
00014         virtual ~GameState() = default;
00015     };
00016 };
```

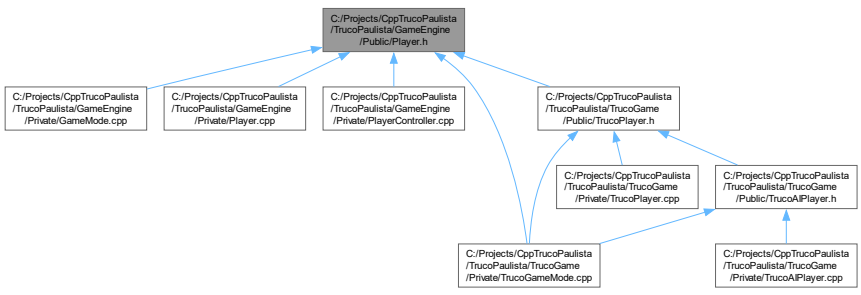
7.31 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/Player.h

```
#include "PlayerState.h"
#include "PlayerController.h"
```

Gráfico de dependência de inclusões para Player.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::Player

  Classe base que representa o avatar de um jogador.

Namespaces

- namespace GameEngine

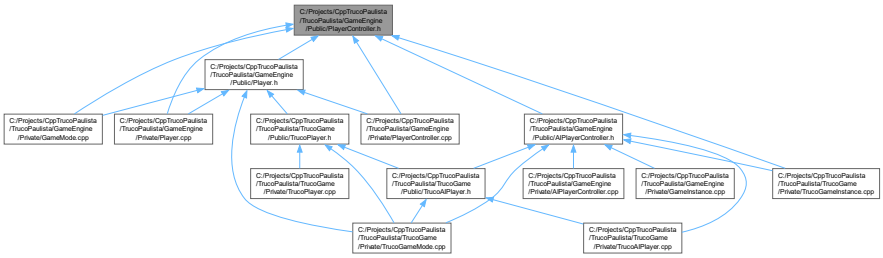
7.32 Player.h

Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "PlayerState.h"
00004 #include "PlayerController.h"
00005
00006 namespace GameEngine
00007 {
00013     class Player
00014     {
00015     protected:
00016         PlayerController* m_PlayerController;
00017         PlayerState* m_PlayerState;
00018
00019     public:
00020         Player(PlayerState* pPlayerState, PlayerController* pPlayerController);
00021         virtual ~Player();
00022
00023         PlayerController* GetPlayerController();
00024         PlayerState* GetPlayerState();
00025     };
00026 };
```

7.33 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/PlayerController.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::PlayerController

  Classe base que representa um jogador.

Namespaces

- namespace GameEngine

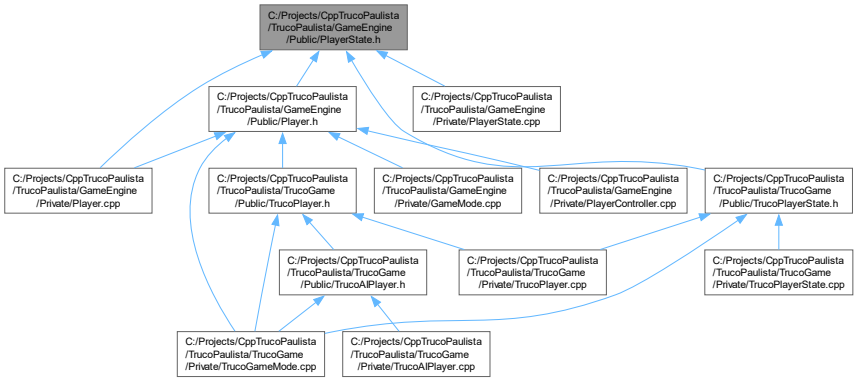
7.34 PlayerController.h

Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace GameEngine
00004 {
00005     class Player;
00006
00012     class PlayerController
00013     {
00014     private:
00015         bool m_IsAIControled;
00016         void Posses(Player* pPlayer);
00017
00018     protected:
00019         Player* m_ControlledPlayer;
00020
00021     public:
00022         PlayerController();
00023         PlayerController(bool isAIControlled);
00024         virtual ~PlayerController();
00025
00026         Player* GetControlledPlayer();
00027
00028         friend Player;
00029     };
00030 };
```

7.35 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/PlayerState.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::PlayerState

  Classe base responsavel por gerenciar o estado do Player durante as partidas.

Namespaces

- namespace GameEngine

7.36 PlayerState.h

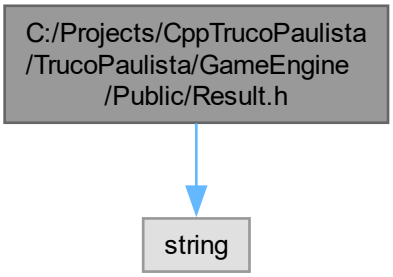
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace GameEngine
00004 {
00010     class PlayerState
00011     {
00012     public:
00013         PlayerState();
00014         virtual ~PlayerState() = default;
00015     };
00016 };
```

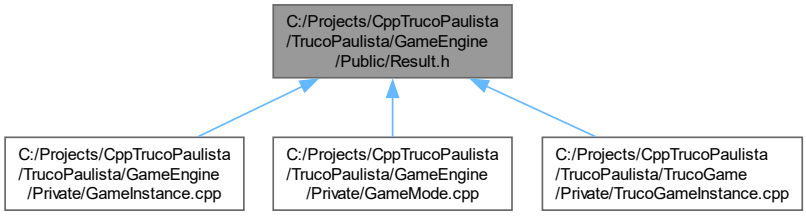
7.37 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/GameEngine/Public/Result.h

`#include <string>`

Gráfico de dependência de inclusões para Result.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class GameEngine::Result

  Classe que representa o resultado de uma operacao que pode falhar de forma aceitavel.

Namespaces

- namespace GameEngine

Enumerações

- enum GameEngine::ResultCode {
  GameEngine::Undefined , GameEngine::Success , GameEngine::Failed , GameEngine::GameIsFull ,
  GameEngine::PlayerAlreadyJoined }

  Enumeracao que representa os possiveis retornos de uma operacao que retorna um Result.

7.38 Result.h

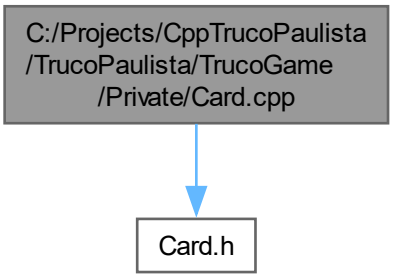
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002 #include <string>
00003
00004 namespace GameEngine
00005 {
00009     enum ResultCode
00010     {
00011         Undefined,
00012         Success,
00013         Failed,
00014         GameIsFull,
00015         PlayerAlreadyJoined
00016     };
00017
00021     class Result
00022     {
00023     private:
00024         std::string m_Message;
00025         ResultCode m_Code;
00026
00027     public:
00028         Result() : m_Code(ResultCode::Undefined), m_Message("") {}
00029         Result(ResultCode code, std::string message) : m_Code(code), m_Message(message) {}
00030
00031         static Result Success() { return Result(ResultCode::Success, ""); }
00032
00033         bool IsSuccess() { return m_Code == ResultCode::Success; }
00034         ResultCode GetCode() { return m_Code; }
00035         std::string GetMessage() { return m_Message; }
00036     };
00037 };
```

7.39 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Card.cpp

```
#include "Card.h"
```

Gráfico de dependência de inclusões para Card.cpp:



7.40 Card.cpp

Ir para a documentação desse arquivo.

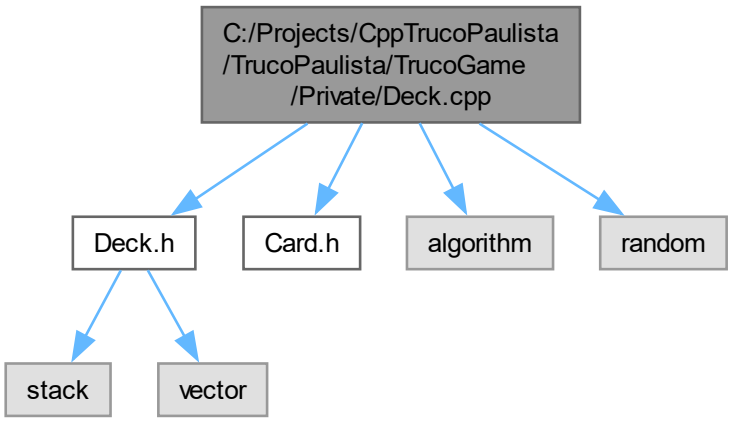
```
00001 #include "Card.h"
00002
00003 using namespace TrucoGame;
00004
00005 Card::Card(Naipes naipe, int value) :
00006     m_Naipe(naipe),
00007     m_Value(value)
00008 {
00009 }
00010
00011 bool Card::operator>(const Card& card) const
00012 {
00013     if (m_Value == 0 && m_Naipe == Naipes::Paus)
00014     {
00015         return true;
00016     }
00017     else if (m_Value == 3 && m_Naipe == Naipes::Copas && !(card.m_Value == 0 && card.m_Naipe ==
      Naipes::Paus))
00018     {
00019         return true;
00020     }
00021     else if (m_Value == 7 && m_Naipe == Naipes::Espadas && !((card.m_Value == 0 && card.m_Naipe ==
      Naipes::Paus) || (card.m_Value == 3 && card.m_Naipe == Naipes::Copas)))
00022     {
00023         return true;
00024     }
00025     else if (m_Value == 3 && m_Naipe == Naipes::Ouros && !((card.m_Value == 0 && card.m_Naipe ==
      Naipes::Paus) || (card.m_Value == 3 && card.m_Naipe == Naipes::Copas) || (card.m_Value == 7 &&
      card.m_Naipe == Naipes::Espadas)))
00026     {
00027         return true;
00028     }
00029     else if (!((card.m_Value == 0 && card.m_Naipe == Naipes::Paus) || (card.m_Value == 3 &&
      card.m_Naipe == Naipes::Copas) || (card.m_Value == 7 && card.m_Naipe == Naipes::Espadas) ||
      (card.m_Value == 3 && card.m_Naipe == Naipes::Ouros)))
00030     {
00031         return m_Value > card.m_Value;
00032     }
00033     else
00034     {
00035         return false;
00036     }
```

```
00037 }
00038
00039 bool Card::operator==(const Card& card) const
00040 {
00041     if (m_Naipe == card.m_Naipe && m_Value == card.m_Value)
00042     {
00043         return true;
00044     }
00045     else
00046     {
00047         return false;
00048     }
00049 }
00050
00051 Naipes Card::GetNaipe()
00052 {
00053     return m_Naipe;
00054 }
00055
00056 int Card::GetValue()
00057 {
00058     return m_Value;
00059 }
```

7.41 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Deck.cpp

```
#include "Deck.h"
#include "Card.h"
#include <algorithm>
#include <random>
```

Gráfico de dependência de inclusões para Deck.cpp:



7.42 Deck.cpp

Ir para a documentação desse arquivo.

```
00001 #include "Deck.h"
00002 #include "Card.h"
00003 #include <algorithm>
00004 #include <random>
```

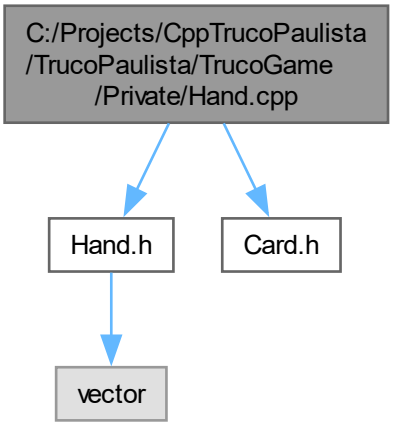
```
00005
00006 using namespace TrucoGame;
00007
00008 void Deck::Init()
00009 {
00010     DestroyCards();
00011
00012     for (int naipe = Naipes::Paus; naipe != Naipes::Last; naipe++)
00013     {
00014         for (int i = 1; i <= MAX_CARDS_PER_NAIPE; i++)
00015         {
00016             Card* card = new Card(static_cast<Naipes>(naipe), i);
00017             m_Cards.push(card);
00018         }
00019     }
00020 }
00021
00022 void Deck::Clear()
00023 {
00024     while (!m_Cards.empty())
00025     {
00026         m_Cards.pop();
00027     }
00028 }
00029
00030 void Deck::DestroyCards()
00031 {
00032     while (!m_Cards.empty())
00033     {
00034         Card* card = m_Cards.top();
00035         m_Cards.pop();
00036
00037         delete card;
00038     }
00039 }
00040
00041 Deck::Deck()
00042 {
00043 }
00044
00045 Deck::~Deck()
00046 {
00047     DestroyCards();
00048 }
00049
00050 int Deck::GetNumCards()
00051 {
00052     return (int)m_Cards.size();
00053 }
00054
00055 void Deck::Shuffle()
00056 {
00057     auto container = m_Cards._Get_container();
00058     std::vector<Card*> aux(container.begin(), container.end());
00059
00060     std::random_device rd;
00061     std::mt19937 g(rd());
00062
00063     std::shuffle(aux.begin(), aux.end(), g);
00064
00065     Clear();
00066
00067     std::for_each(aux.begin(), aux.end(), [=](Card* card)
00068         {
00069             m_Cards.push(card);
00070         }
00071     );
00072 }
00073
00074 Card* Deck::DrawCard()
00075 {
00076     Card* card = m_Cards.top();
00077     m_Cards.pop();
00078
00079     return card;
00080 }
00081
00082 std::vector<Card*> Deck::DrawCards(int numCards)
00083 {
00084     std::vector<Card*> cards(numCards);
00085
00086     if (!m_Cards.empty())
00087     {
00088         if (m_Cards.size() < numCards)
00089         {
00090             numCards = (int)m_Cards.size();
00091         }
```

```
00092
00093         while (numCards > 0)
00094         {
00095             Card* card = m_Cards.top();
00096             m_Cards.pop();
00097
00098             cards.push_back(card);
00099             numCards--;
00100         }
00101     }
00102
00103     return cards;
00104 }
```

7.43 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/Hand.cpp

```
#include "Hand.h"
#include "Card.h"
```

Gráfico de dependência de inclusões para Hand.cpp:



7.44 Hand.cpp

Ir para a documentação desse arquivo.

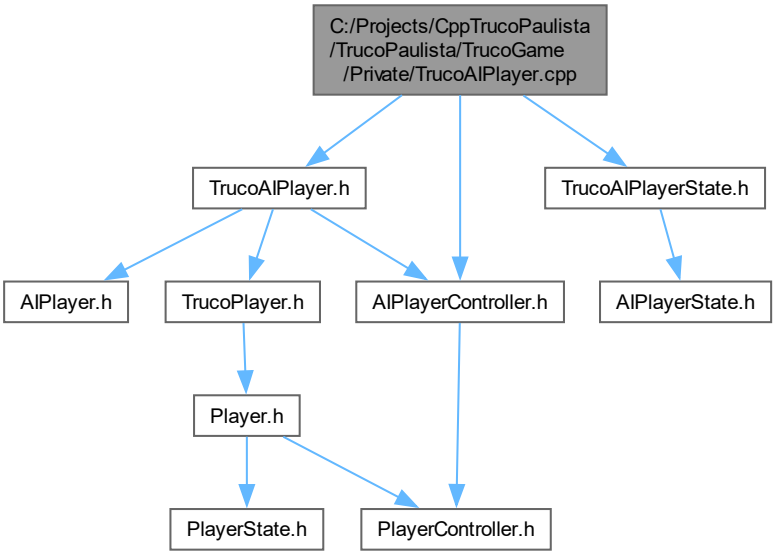
```
00001 #include "Hand.h"
00002 #include "Card.h"
00003
00004 using namespace TrucoGame;
00005
00006 Hand::Hand()
00007 {
00008 }
00009
00010 Hand::~Hand()
00011 {
00012     if (!m_Cards.empty())
00013     {
00014         for (Card* card : m_Cards)
00015         {
00016             delete card;
```

```
00017         }
00018
00019         m_Cards.clear();
00020     }
00021 }
00022
00023 void Hand::SetInitalCards(std::vector<Card*> cards)
00024 {
00025     m_Cards.clear();
00026     m_Cards.insert(m_Cards.end(), cards.begin(), cards.end());
00027 }
00028
00029 void Hand::AddCard(Card* card)
00030 {
00031     m_Cards.push_back(card);
00032 }
00033
00034 Card* Hand::PlayCard(int index)
00035 {
00036     Card* card = m_Cards[index];
00037     m_Cards.erase(m_Cards.begin() + index);
00038
00039     return card;
00040 }
00041
00042 std::vector<Card*> Hand::GetCards()
00043 {
00044     return m_Cards;
00045 }
```

7.45 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoAIPlayer.cpp

```
#include "TrucoAIPlayer.h"
#include "TrucoAIPlayerState.h"
#include "AIPlayerController.h"
```

Gráfico de dependência de inclusões para TrucoAIPlayer.cpp:



7.46 TrucoAIPlayer.cpp

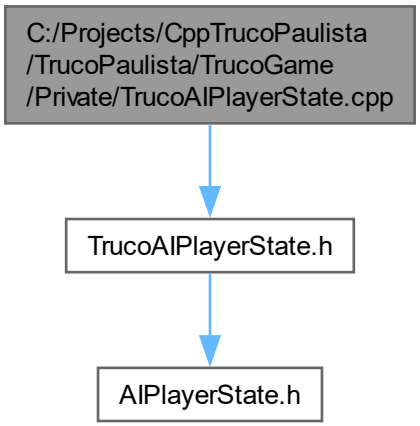
Ir para a documentação desse arquivo.

```
00001 #include "TrucoAIPlayer.h"
00002 #include "TrucoAIPlayerState.h"
00003 #include "AIPlayerController.h"
00004
00005 using namespace TrucoGame;
00006
00007 TrucoAIPlayer::TrucoAIPlayer(
00008     TrucoPlayerState* pPlayerState,
00009     TrucoAIPlayerState* pAIPlayerState,
00010     GameEngine::AIPlayerController* pAIPlayerController,
00011     double minTinkTimeSec,
00012     double maxThinkTimeSec) :
00013         GameEngine::AIPlayer(pAIPlayerState, minTinkTimeSec, maxThinkTimeSec),
00014         TrucoPlayer(pPlayerState, pAIPlayerController)
00015 {
00016 }
```

7.47 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoAIPlayerState.cpp

```
#include "TrucoAIPlayerState.h"
```

Gráfico de dependência de inclusões para TrucoAIPlayerState.cpp:



7.48 TrucoAIPlayerState.cpp

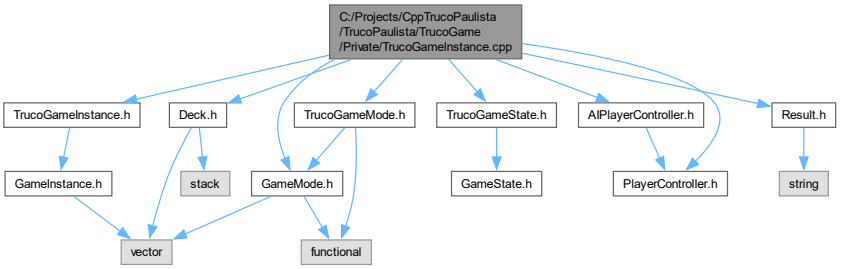
Ir para a documentação desse arquivo.

```
00001 #include "TrucoAIPlayerState.h"
00002
00003 using namespace TrucoGame;
00004
00005 TrucoAIPlayerState::TrucoAIPlayerState() : GameEngine::AIPlayerState()
00006 {
00007 }
```

7.49 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoGameInstance.cpp

```
#include "TrucoGameInstance.h"
#include "TrucoGameMode.h"
#include "TrucoGameState.h"
#include "Deck.h"
#include "AIPlayerController.h"
#include "PlayerController.h"
#include "GameMode.h"
#include "Result.h"
```

Gráfico de dependência de inclusões para TrucoGameInstance.cpp:



7.50 TrucoGameInstance.cpp

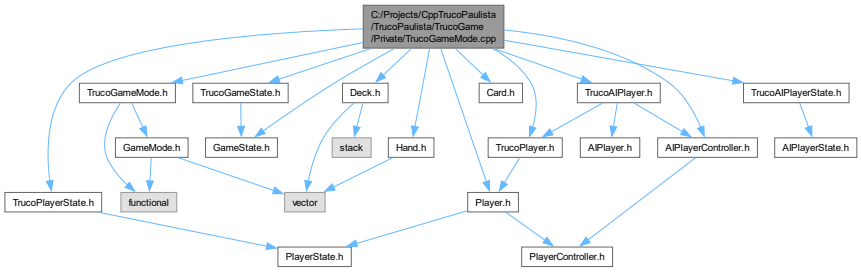
Ir para a documentação desse arquivo.

```
00001 #include "TrucoGameInstance.h"
00002 #include "TrucoGameMode.h"
00003 #include "TrucoGameState.h"
00004 #include "Deck.h"
00005 #include "AIPlayerController.h"
00006 #include "PlayerController.h"
00007 #include "GameMode.h"
00008 #include "Result.h"
00009
00010 using namespace TrucoGame;
00011
00012 void TrucoGameInstance::CreateGame(int numPlayers, int numAIPlayers, GameEngine::GameMode* pGameMode)
00013 {
00014     Cleanup();
00015
00016     int totalPlayers = numPlayers + numAIPlayers;
00017
00018     // Inicializa o modo de jogo
00019     m_GameMode = new TrucoGameMode(totalPlayers, new TrucoGameState(new Deck()));
00020
00021     // Inicializa os jogadores
00022     m_Controllers.reserve(totalPlayers);
00023
00024     for (size_t i = 0; i < numPlayers; i++)
00025     {
00026         GameEngine::PlayerController* pController = new GameEngine::PlayerController();
00027         m_Controllers.push_back(pController);
00028
00029         m_GameMode->Join(pController);
00030     }
00031
00032     for (size_t i = 0; i < numAIPlayers; i++)
00033     {
00034         GameEngine::AIPlayerController* pAIController = new GameEngine::AIPlayerController();
00035         m_Controllers.push_back(pAIController);
00036
00037         m_GameMode->Join(pAIController, true);
00038     }
00039 }
```

7.51 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Private/TrucoGameMode.cpp

```
#include "TrucoGameMode.h"
#include "TrucoGameState.h"
#include "GameState.h"
#include "TrucoPlayer.h"
#include "TrucoPlayerState.h"
#include "Player.h"
#include "Deck.h"
#include "Card.h"
#include "Hand.h"
#include "TrucoAIPlayer.h"
#include "TrucoAIPlayerState.h"
#include "AIPlayerController.h"
```

Gráfico de dependência de inclusões para TrucoGameMode.cpp:



7.52 TrucoGameMode.cpp

Ir para a documentação desse arquivo.

```
00001 #include "TrucoGameMode.h"
00002 #include "TrucoGameState.h"
00003 #include "GameState.h"
00004 #include "TrucoPlayer.h"
00005 #include "TrucoPlayerState.h"
00006 #include "Player.h"
00007 #include "Deck.h"
00008 #include "Card.h"
00009 #include "Hand.h"
00010 #include "TrucoAIPlayer.h"
00011 #include "TrucoAIPlayerState.h"
00012 #include "AIPlayerController.h"
00013
00014 using namespace TrucoGame;
00015
00016 TrucoPlayer* TrucoGameMode::GetStartPlayer()
00017 {
00018     TrucoPlayer* pStartPlayer = nullptr;
00019
00020     if (!m_PlayersList.empty())
00021     {
00022         pStartPlayer = dynamic_cast<TrucoPlayer*>(m_PlayersList[0]->GetControlledPlayer());
00023     }
00024
00025     return pStartPlayer;
00026 }
00027
00028 TrucoGameMode::TrucoGameMode(int numPlayers, TrucoGameState* gameState)
00029     : GameEngine::GameMode(numPlayers, gameState)
00030 {
00031 }
00032
00033 TrucoGameMode::~TrucoGameMode()
```

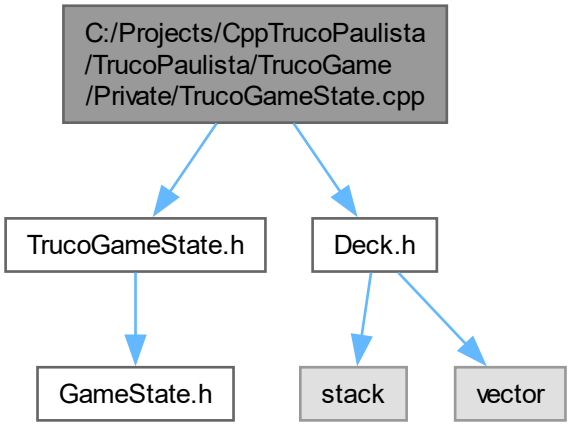
```
00034 {
00035 }
00036
00037 void TrucoGameMode::StartGame()
00038 {
00039     std::vector<TrucoPlayer*> players;
00040
00041     for (auto& pController : m_PlayersList)
00042     {
00043         TrucoPlayer* pPlayer = dynamic_cast<TrucoPlayer*>(pController->GetControlledPlayer());
00044         if (pPlayer)
00045         {
00046             players.push_back(pPlayer);
00047         }
00048     }
00049
00050     TrucoGameState* pTrucoGameState = dynamic_cast<TrucoGameState*>(m_GameState);
00051     if (pTrucoGameState)
00052     {
00053         Deck* deck = pTrucoGameState->GetDeck();
00054
00055         // Set initial players cards
00056         deck->Init();
00057         deck->Shuffle();
00058
00059         for (int i = 0; i < m_NumStartCards; i++)
00060         {
00061             for (auto& player : players)
00062             {
00063                 Card* card = deck->DrawCard();
00064                 TrucoPlayerState* pTrucoPlayerState =
    dynamic_cast<TrucoPlayerState*>(player->GetPlayerState());
00065
00066                 if (pTrucoPlayerState)
00067                 {
00068                     pTrucoPlayerState->GetHand()->AddCard(card);
00069                 }
00070             }
00071         }
00072
00073         GameEngine::GameMode::StartGame();
00074     }
00075 }
00076
00077 void TrucoGameMode::OnGameStarted()
00078 {
00079     GameEngine::GameMode::OnGameStarted();
00080
00081     // Initialize turn
00082     TrucoGameState* pTrucoGameState = dynamic_cast<TrucoGameState*>(m_GameState);
00083
00084     if (pTrucoGameState)
00085     {
00086         pTrucoGameState->ResetTurn();
00087
00088         TrucoPlayer* pStartPlayer = GetStartPlayer();
00089         AdvancedTurn(pStartPlayer);
00090     }
00091 }
00092
00093 void TrucoGameMode::OnJoined(GameEngine::PlayerController* pPlayerController, bool isAIControlled)
00094 {
00095     if (isAIControlled)
00096     {
00097         TrucoPlayerState* pTrucoPlayerState = new TrucoPlayerState(new Hand());
00098         TrucoAIPlayer* pAIPlayer = new TrucoAIPlayer(pTrucoPlayerState, new TrucoAIPlayerState(),
    dynamic_cast<GameEngine::AIPlayerController*>(pPlayerController), 2, 5);
00099     }
00100     else
00101     {
00102         TrucoPlayer* pPlayer = new TrucoPlayer(new TrucoPlayerState(new Hand()), pPlayerController);
00103     }
00104
00105     GameEngine::GameMode::OnJoined(pPlayerController, isAIControlled);
00106 }
00107
00108 void TrucoGameMode::OnTurnAdvanced(TrucoPlayer* turnPlayer)
00109 {
00110     if (m_OnTurnAdvancedCallback)
00111     {
00112         m_OnTurnAdvancedCallback(turnPlayer);
00113     }
00114 }
00115
00116 void TrucoGameMode::BindTurnAdvancedCallback(std::function<void(TrucoPlayer*)> callback)
00117 {
00118     m_OnTurnAdvancedCallback = callback;
```

```
00119 }
00120
00121 void TrucoGameMode::AdvancedTurn(TrucoPlayer* turnPlayer)
00122 {
00123     TrucoGameState* pTrucoGameState = dynamic_cast<TrucoGameState*>(m_GameState);
00124
00125     if (pTrucoGameState)
00126     {
00127         pTrucoGameState->AdvanceTurn(turnPlayer);
00128     }
00129
00130     OnTurnAdvanced(turnPlayer);
00131 }
```

7.53 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoGameState.cpp

```
#include "TrucoGameState.h"
#include "Deck.h"
```

Gráfico de dependência de inclusões para TrucoGameState.cpp:



7.54 TrucoGameState.cpp

Ir para a documentação desse arquivo.

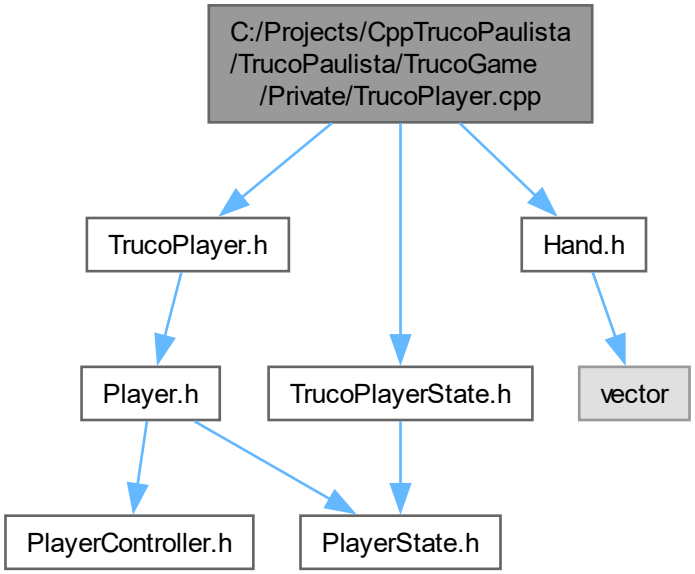
```
00001 #include "TrucoGameState.h"
00002 #include "Deck.h"
00003
00004 using namespace TrucoGame;
00005
00006 TrucoGameState::TrucoGameState(Deck* deck) : m_TurnPlayer(nullptr)
00007 {
00008     m_Deck = deck;
00009     m_CurrentTurn = 0;
00010 }
00011
00012 TrucoGameState::~TrucoGameState()
00013 {
00014     if (m_Deck)
00015     {
00016         delete m_Deck;
```

```
00017     }
00018 }
00019
00020 Deck* TrucoGameState::GetDeck()
00021 {
00022     return m_Deck;
00023 }
00024
00025 TrucoPlayer* TrucoGameState::GetTurnPlayer()
00026 {
00027     return m_TurnPlayer;
00028 }
00029
00030 void TrucoGameState::AdvanceTurn(TrucoPlayer* turnPlayer)
00031 {
00032     m_CurrentTurn++;
00033     m_TurnPlayer = turnPlayer;
00034 }
00035
00036 void TrucoGameState::ResetTurn()
00037 {
00038     m_TurnPlayer = nullptr;
00039     m_CurrentTurn = 0;
00040 }
```

7.55 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoPlayer.cpp

```
#include "TrucoPlayer.h"
#include "TrucoPlayerState.h"
#include "Hand.h"
```

Gráfico de dependência de inclusões para TrucoPlayer.cpp:



7.56 TrucoPlayer.cpp

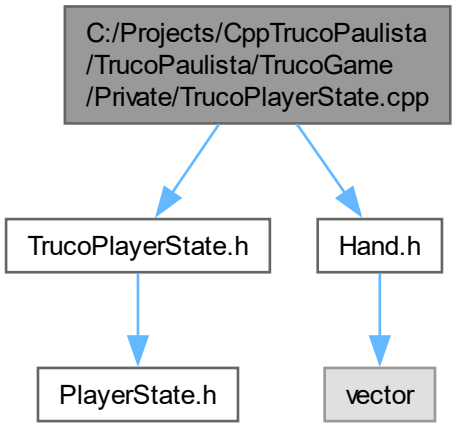
Ir para a documentação desse arquivo.

```
00001 #include "TrucoPlayer.h"
00002 #include "TrucoPlayerState.h"
00003 #include "Hand.h"
00004
00005 using namespace TrucoGame;
00006
00007 TrucoPlayer::TrucoPlayer(TrucoPlayerState* pTrucoPlayerState, GameEngine::PlayerController*
      pPlayerController) :
00008     GameEngine::Player(pTrucoPlayerState, pPlayerController)
00009 {
00010 }
```

7.57 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Private/TrucoPlayerState.cpp

```
#include "TrucoPlayerState.h"
#include "Hand.h"
```

Gráfico de dependência de inclusões para TrucoPlayerState.cpp:



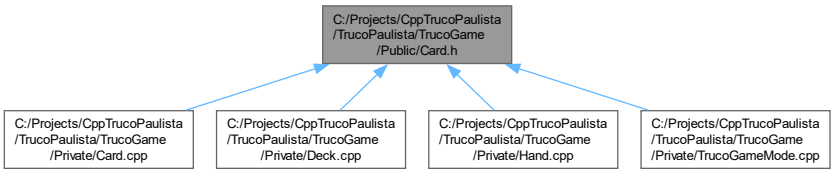
7.58 TrucoPlayerState.cpp

Ir para a documentação desse arquivo.

```
00001 #include "TrucoPlayerState.h"
00002 #include "Hand.h"
00003
00004 using namespace TrucoGame;
00005
00006 TrucoPlayerState::TrucoPlayerState(Hand* hand) : GameEngine::PlayerState()
00007 {
00008     m_Hand = hand;
00009 }
00010
00011 TrucoPlayerState::~TrucoPlayerState()
00012 {
00013     if (m_Hand)
00014     {
00015         delete m_Hand;
00016     }
00017 }
00018
00019 Hand* TrucoPlayerState::GetHand()
00020 {
00021     return m_Hand;
00022 }
```

7.59 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Card.h

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::Card

  Classe que representa uma carte de baralho.

Namespaces

- namespace TrucoGame

Enumerações

- enum TrucoGame::Naipes : int { TrucoGame::Paus , TrucoGame::Ouros , TrucoGame::Copas , TrucoGame::Espadas , TrucoGame::Last }

  Enumeracao que representa os possiveis naipes de um baralho.

7.60 Card.h

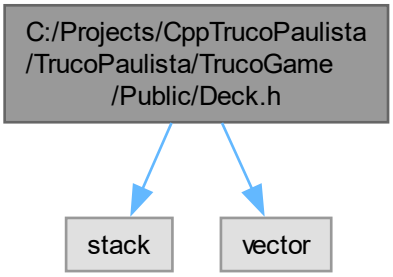
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 namespace TrucoGame
00004 {
00008     enum Naipes : int
00009     {
00010         Paus,
00011         Ouros,
00012         Copas,
00013         Espadas,
00014         Last
00015     };
00016
00020     class Card
00021     {
00022     private:
00023         Naipes m_Naipe;
00024         int m_Value;
00025
00026     public:
00027         Card(Naipes naipe, int value);
00028
00029         bool operator>(const Card& card) const;
00030         bool operator==(const Card& card) const;
00031
00032         Naipes GetNaipe();
00033         int GetValue();
00034     };
00035 };
```

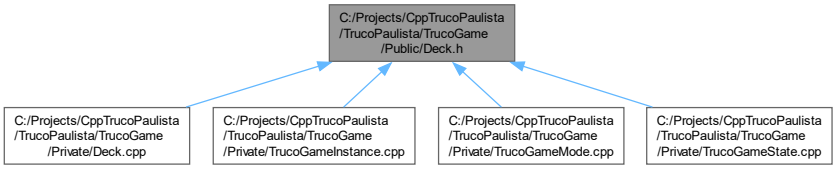
7.61 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Deck.h

```
#include <stack>
#include <vector>
```

Gráfico de dependência de inclusões para Deck.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::Deck

  Classe que representa o conjunto de cartas de baralho que estao disponiveis no jogo.

Namespaces

- namespace TrucoGame

7.62 Deck.h

Ir para a documentação desse arquivo.

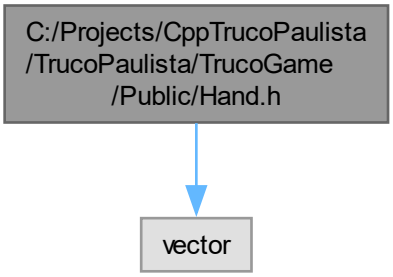
```
00001 #pragma once
00002
00003 #include <stack>
00004 #include <vector>
```

```
00005
00006 namespace TrucoGame
00007 {
00008     class Card;
00009
00013     class Deck
00014     {
00015     private:
00016         std::stack<Card*> m_Cards;
00017         const int MAX_CARDS_PER_NAIPE = 10;
00018
00019         void Clear();
00020         void DestroyCards();
00021
00022     public:
00023         Deck();
00024         ~Deck();
00025
00026         void Init();
00027         int GetNumCards();
00028         void Shuffle();
00029         Card* DrawCard();
00030         std::vector<Card*> DrawCards(int numCards);
00031     };
00032 };
```

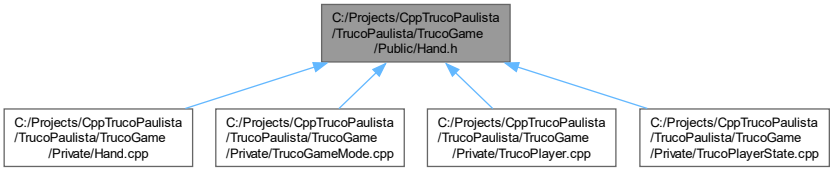
7.63 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/Hand.h

`#include <vector>`

Gráfico de dependência de inclusões para Hand.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::Hand

  Classe que representa as cartas de baralho que estao em posse (na mao) de um jogador.

Namespaces

- namespace TrucoGame

7.64 Hand.h

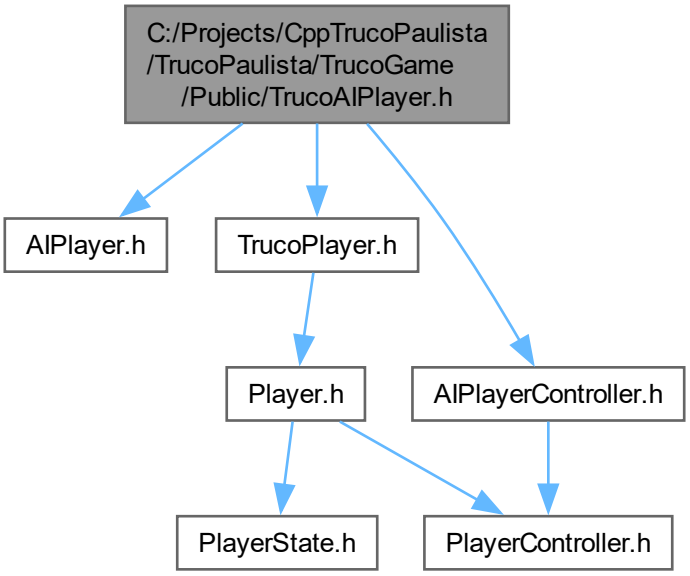
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include <vector>
00004
00005 namespace TrucoGame
00006 {
00007     class Card;
00008
00012     class Hand
00013     {
00014     private:
00015         std::vector<Card*> m_Cards;
00016
00017     public:
00018         Hand();
00019         ~Hand();
00020
00021         void SetInitalCards(std::vector<Card*> cards);
00022         void AddCard(Card* card);
00023         Card* PlayCard(int index);
00024         std::vector<Card*> GetCards();
00025     };
00026 };
```

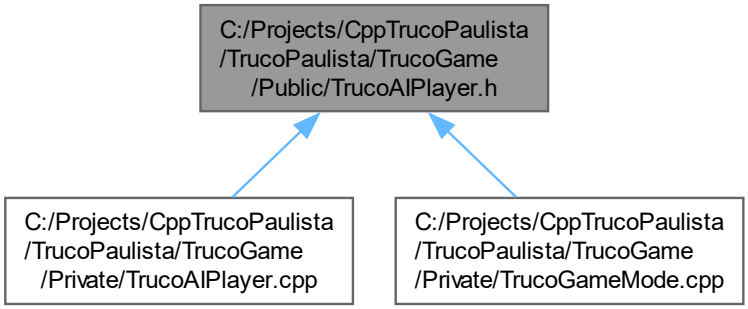
7.65 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoAIPlayer.h

```
#include "AIPlayer.h"
#include "TrucoPlayer.h"
#include "AIPlayerController.h"
```

Gráfico de dependência de inclusões para TrucoAIPlayer.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoAIPlayer

  Especializacao da classe AIPlayer para um jogo de truco.

Namespaces

- namespace TrucoGame

7.66 TrucoAIPlayer.h

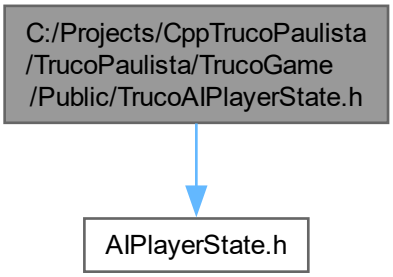
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "AIPlayer.h"
00004 #include "TrucoPlayer.h"
00005 #include "AIPlayerController.h"
00006
00007 namespace TrucoGame
00008 {
00009     class TrucoAIPlayerState;
00010
00016     class TrucoAIPlayer : public GameEngine::AIPlayer, public TrucoPlayer
00017     {
00018     public:
00019         TrucoAIPlayer(
00020             TrucoPlayerState* pPlayerState,
00021             TrucoAIPlayerState* pAIPlayerState,
00022             GameEngine::AIPlayerController* pAIPlayerController,
00023             double minTinkTimeSec,
00024             double maxThinkTimeSec);
00025     };
00026 };
```

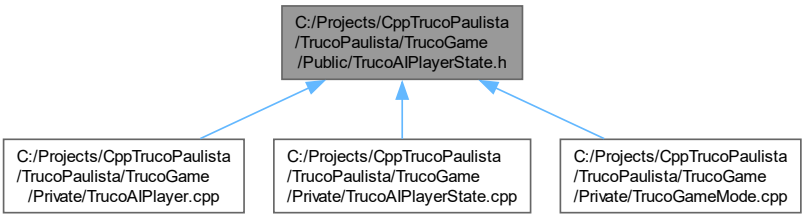
7.67 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Public/TrucoAIPlayerState.h

`#include "AIPlayerState.h"`

Gráfico de dependência de inclusões para TrucoAIPlayerState.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoAIPlayerState

  Especializacao da classe AIPlayerState para um jogo de truco.

Namespaces

- namespace TrucoGame

7.68 TrucoAIPlayerState.h

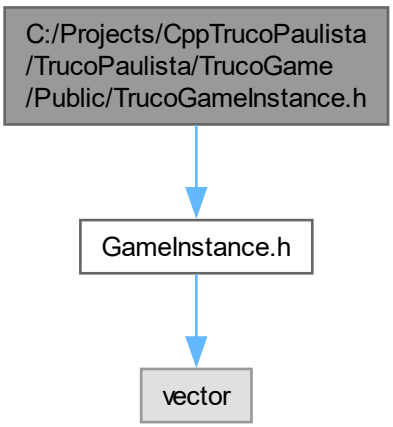
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "AIPlayerState.h"
00004
00005 namespace TrucoGame
00006 {
00012     class TrucoAIPlayerState : public GameEngine::AIPlayerState
00013     {
00014     public:
00015         TrucoAIPlayerState();
00016     };
00017 };
```

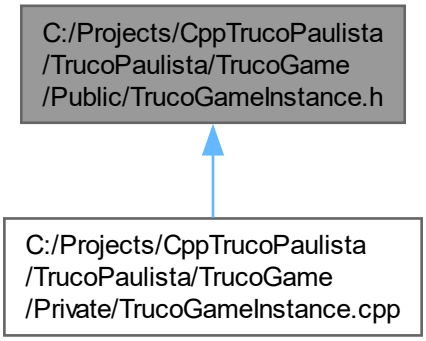
7.69 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Public/TrucoGameInstance.h

`#include "GameInstance.h"`

Gráfico de dependência de inclusões para TrucoGameInstance.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoGameInstance

  Expecializacao da classe GameInstance para um jogo de truco.

Namespaces

- namespace TrucoGame

7.70 TrucoGameInstance.h

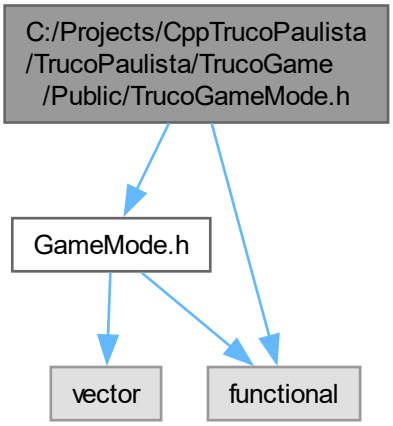
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "GameInstance.h"
00004
00005 namespace TrucoGame
00006 {
00013     class TrucoGameInstance : public GameEngine::GameInstance
00014     {
00015     public:
00016         void CreateGame(int numPlayers, int numAIPlayers, GameEngine::GameMode* pGameMode) override;
00017     };
00018 };
```

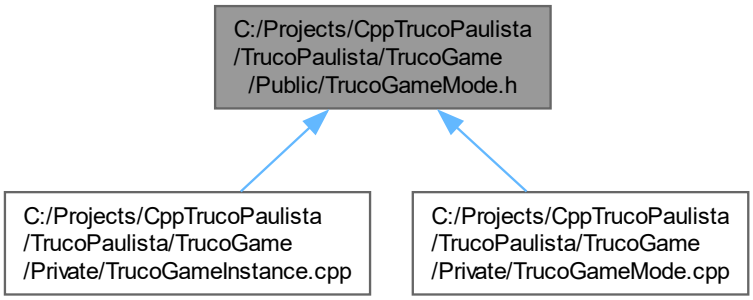
7.71 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Public/TrucoGameMode.h

```
#include "GameMode.h"
#include <functional>
```

Gráfico de dependência de inclusões para TrucoGameMode.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoGameMode

  Especializacao da classe GameMode para um jogo de truco.

Namespaces

- namespace TrucoGame

7.72 TrucoGameMode.h

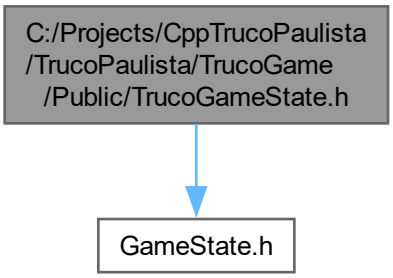
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002 #include "GameMode.h"
00003 #include <functional>
00004
00005 namespace TrucoGame
00006 {
00007     class Deck;
00008     class TrucoPlayer;
00009     class TrucoGameState;
00010
00014     class TrucoGameMode : public GameEngine::GameMode
00015     {
00016     private:
00017         int m_NumStartCards = 3;
00018
00019         TrucoPlayer* GetStartPlayer();
00020         std::function<void(TrucoPlayer*)> m_OnTurnAdvancedCallback;
00021
00022         void OnTurnAdvanced(TrucoPlayer* turnPlayer);
00023
00024     public:
00025         TrucoGameMode(int numPlayers, TrucoGameState* gameState);
00026         ~TrucoGameMode();
00027
00028         void StartGame() override;
00029         void OnGameStarted() override;
00030         void OnJoined(GameEngine::PlayerController* pPlayerController, bool isAIControlled = false)
      override;
00031
00032         void BindTurnAdvancedCallback(std::function<void(TrucoPlayer*)> callback);
00033         void AdvancedTurn(TrucoPlayer* turnPlayer);
00034
00035         int GetNumStartCards() { return m_NumStartCards; }
00036     };
00037 };
```

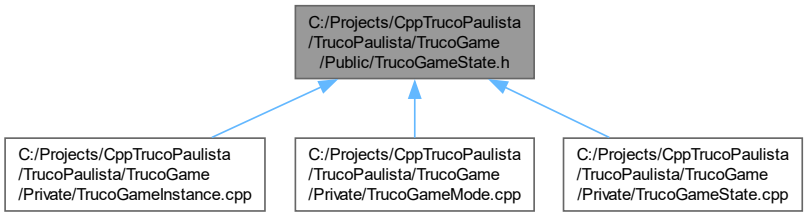
7.73 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Public/TrucoGameState.h

```
#include "GameState.h"
```

Gráfico de dependência de inclusões para TrucoGameState.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoGameState

  Especializacao da classe GameState para um jogo de truco.

Namespaces

- namespace TrucoGame

7.74 TrucoGameState.h

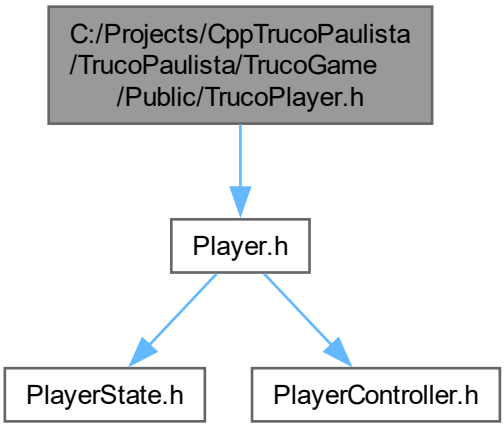
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "GameState.h"
00004
00005 namespace TrucoGame
00006 {
00007     class Deck;
00008     class TrucoPlayer;
00009
00015     class TrucoGameState : public GameEngine::GameState
00016     {
00017     private:
00018         Deck* m_Deck;
00019         int m_CurrentTurn;
00020
00021         TrucoPlayer* m_TurnPlayer;
00022
00023     public:
00024         TrucoGameState(Deck* deck);
00025         ~TrucoGameState();
00026
00027         Deck* GetDeck();
00028         TrucoPlayer* GetTurnPlayer();
00029
00030         void AdvanceTurn(TrucoPlayer* turnPlayer);
00031         void ResetTurn();
00032     };
00033 };
```

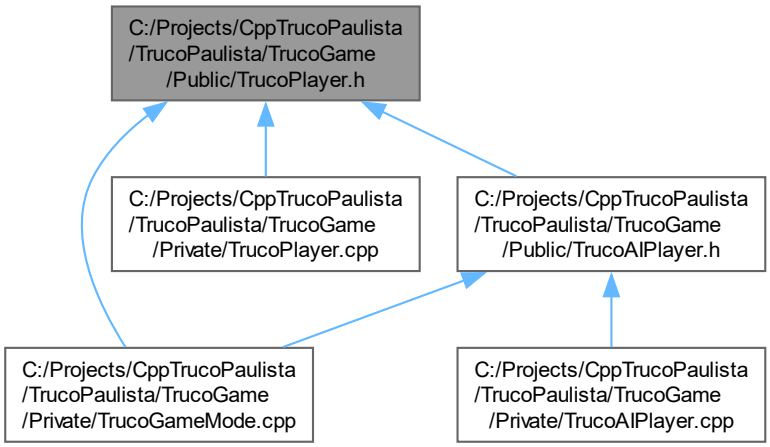
7.75 Referência do Arquivo C:/Projects/CppTrucoPaulista/TrucoPaulista/TrucoGame/Public/TrucoPlayer.h

```
#include "Player.h"
```

Gráfico de dependência de inclusões para TrucoPlayer.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoPlayer

  Especializacao da classe Player para um jogo de truco.

Namespaces

- namespace TrucoGame

7.76 TrucoPlayer.h

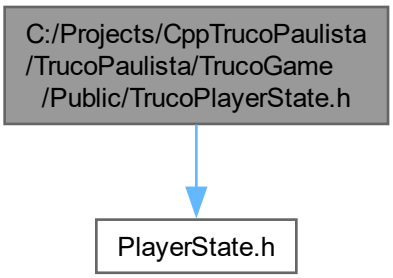
Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "Player.h"
00004
00005 namespace TrucoGame
00006 {
00007     class TrucoPlayerState;
00008
00014     class TrucoPlayer : public GameEngine::Player
00015     {
00016     public:
00017         TrucoPlayer(TrucoPlayerState* pTrucoPlayerState, GameEngine::PlayerController*
      pPlayerController);
00018     };
00019 };
```

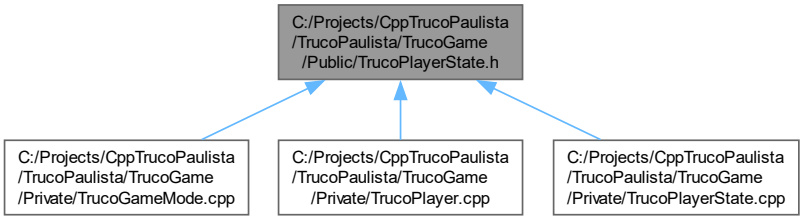
7.77 Referência do Arquivo C:/Projects/CppTrucoPaulista/Truco↩ Paulista/TrucoGame/Public/TrucoPlayerState.h

`#include "PlayerState.h"`

Gráfico de dependência de inclusões para TrucoPlayerState.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com esse arquivo:



Componentes

- class TrucoGame::TrucoPlayerState

  Especializacao da classe PlayerState para uma partida de truco.

Namespaces

- namespace TrucoGame

7.78 TrucoPlayerState.h

Ir para a documentação desse arquivo.

```
00001 #pragma once
00002
00003 #include "PlayerState.h"
00004
00005 namespace TrucoGame
00006 {
00007     class Hand;
00008
00014     class TrucoPlayerState : public GameEngine::PlayerState
00015     {
00016     private:
00017         Hand* m_Hand;
00018
00019     public:
00020         TrucoPlayerState(Hand* hand);
00021         ~TrucoPlayerState();
00022
00023         Hand* GetHand();
00024     };
00025 };
```