SFML-Space

1.0pre

Erzeugt von Doxygen 1.5.6

Thu Jun 12 18:19:44 2008

Inhaltsverzeichnis

Kapitel 1

Ausstehende Aufgaben

Klasse AHUDElement Kurze dokumentation

Element ANPC::get_fancy(float *farray, std::size_t start, std::size_t end) Das hier ist FALSCH! Signed ist nötig

Element CBriefing::CBriefing() initialisierungsliste

Element CEnemyManager::init(std::string filename_waves, std::string filename_parameter, global_data_pointers Data to be removed

Element CHUDManager::registerHUD(const std::string *data, HUDType Type, int posX, int posY, CBitmapFont *bmF ???

Element CMenu::CMenu() Initialisierungsliste

Element CParticle::Init(const std::string &TexName, float FrameWidth=-1, float FrameHeight=-1, int FrameCount=1) m_Sprite.Init(lpDevice,TexManager,TexName,FrameWidth,FrameHeight,FrameCount);

Element CParticle::Update() m_Sprite.IncFrame();

Element CPowerUp::Update() m_Sprite.IncFrame();

Element CProjectile::Update() Finde eine Schlaue Idee wegen IncFrame()

Element CShip::EquipWeapon(int GroupID) genauer überfliegen, evtl umschreiben

Element CStar::InitAsteroid(AsteroidData *Data, int Layer, float XSpeed, float YSpeed, float X, float Y) m_Star.Init(Data->FileName, Data->FrameX, Data->FrameY, Data->FrameCount);

Kapitel 2

Klassen-Verzeichnis

2.1 Klassenhierarchie

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

Kapitel 3

Klassen-Verzeichnis

3.1 Auflistung der Klassen

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

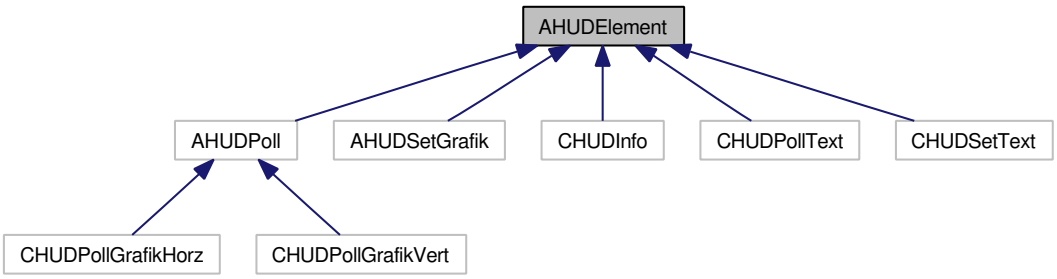
Kapitel 4

Klassen-Dokumentation

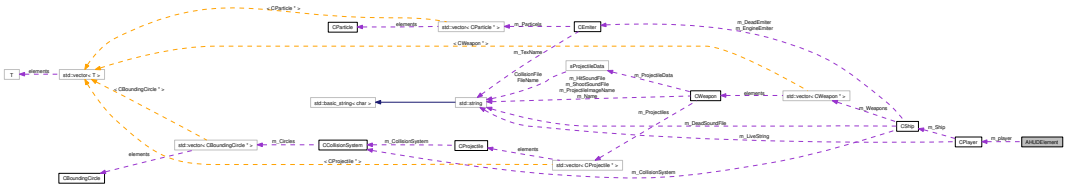
4.1 AHUDElement Klassenreferenz

`#include <HUD-Element.hpp>`

Klassendiagramm für AHUDElement:



Zusammengehörigkeiten von AHUDElement:



Öffentliche Methoden

- void **SetPosition** (int X, int Y)
- void **SetSize** (int Width, int Height)
- int **GetPosition_X** ()
- int **GetPosition_Y** ()
- int **GetSize_Width** ()
- int **GetSize_Height** ()
- virtual void **draw** ()=0

Geschützte Attribute

- int **m_X**
- int **m_Y**
- int **m_Width**
- int **m_Height**
- CPlayer * **m_player**

4.1.1 Ausführliche Beschreibung

Noch zu erledigen

Kurze dokumentation

Definiert in Zeile 33 der Datei HUD-Element.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

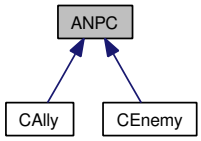
- HUD-Element.hpp
- HUD-Element.cpp

4.2 ANPC Klassenreferenz

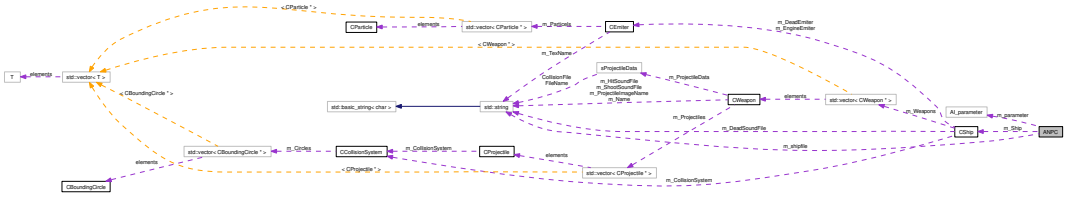
Basisklasse für NPCs.

`#include <NPC.hpp>`

Klassendiagramm für ANPC:



Zusammengehörigkeiten von ANPC:



Öffentliche Typen

- enum **Strength** {
  EMPTY, **WEAK_ENEMY**, **MEDIUMWEAK_ENEMY**, **AVERAGE_ENEMY**,
  MEDIUMHARD_ENEMY, **HARD_ENEMY**, **WEAK_BOSS** = 101, **MEDIUMWEAK_BOSS** = 102,
  AVERAGE_BOSS = 103, **MEDIUMHARD_BOSS** = 104, **HARD_BOSS** = 105 }

Öffentliche Methoden

- **ANPC** (unsigned int starttime, int posx, int posy, Strength theStrength, std::string shipfile, AI_-parameter ∗parameter)
- bool **init** (global_data_pointers Data, strength ∗tStrength)
- void **calculate** (float ∗fancy, std::list< projectileData > &r_Projectiles, std::list< ANPC ∗ > &r_-myEnemys, std::list< ANPC ∗ > &r_myAllys)
- void **update** (const std::bitset< SCREEN_X_SIZE-HUD_SIZE_X > &owned)
- void **Draw** ()
- void **printposition** ()
- CShip ∗ **getShip** ()
- unsigned int **getStarttime** ()
- int **getPower** ()
- int **getStrength** ()

Geschützte Methoden

- float get_fancy (float ∗farray, std::size_t start, std::size_t end)

Geschützte Attribute

- int **des_x**
- int **des_y**
- AI_parameter * **m_parameter**
- int **desired_distance**
- int **desired_center**
- CShip * **m_Ship**
- int **m_weapon_offset**

4.2.1 Ausführliche Beschreibung

Basisklasse für NPCs.

Autor:

Christoph Egger

Definiert in Zeile 32 der Datei NPC.hpp.

4.2.2 Dokumentation der Elementfunktionen

4.2.2.1 float ANPC::get_fancy (float * *farray*, std::size_t *start*, std::size_t *end*) [protected]

Noch zu erledigen

Das hier ist FALSCH! Signed ist nötig

Definiert in Zeile 310 der Datei NPC.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

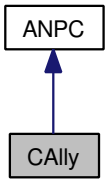
- NPC.hpp
- NPC.cpp

4.3 CAlly Klassenreferenz

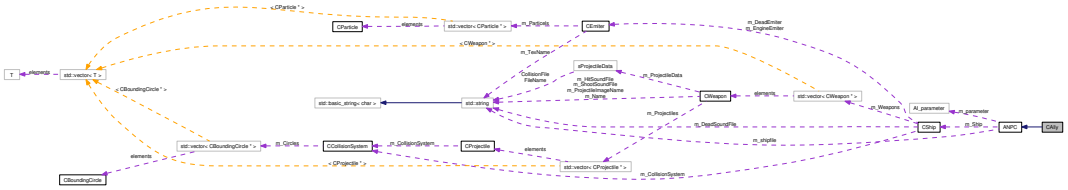
Verbündeter NPC.

`#include <Ally.hpp>`

Klassendiagramm für CAlly:



Zusammengehörigkeiten von CAlly:



Öffentliche Methoden

- **CAlly** (unsigned int starttime, int posx, int posy, Strength theStrength, std::string shipfile, AI_-parameter ∗parameter)

4.3.1 Ausführliche Beschreibung

Verbündeter NPC.

Autor:

Christoph Egger

Definiert in Zeile 27 der Datei Ally.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

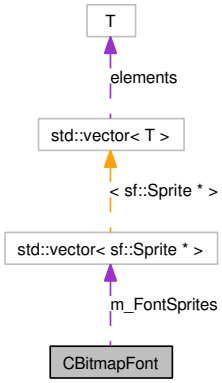
- Ally.hpp
- Ally.cpp

4.4 CBitmapFont Klassenreferenz

Bitmap Fonts.

`#include <Bitmapfont.hpp>`

Zusammengehörigkeiten von CBitmapFont:



Öffentliche Methoden

- void **Init** ()
- void **DrawText** (float x, float y, std::string Text, float PixelHeight=0.0f, float RectXSize=10000.0f, float RectYSize=10000.0f, int ScrollY=0, bool *CanScroll=NULL, int Alpha=255, bool CenterdX=false)
- float **GetFontHeight** ()

4.4.1 Ausführliche Beschreibung

Bitmap Fonts.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 40 der Datei Bitmapfont.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Bitmapfont.hpp
- Bitmapfont.cpp

4.5 CBoundingCircle Klassenreferenz

Bounding Circles für das Collision System.

`#include <BoundingCircle.hpp>`

Öffentliche Methoden

- CBoundingCircle ()

  Konstrukor und Destruktor.

- void SetRadius (float Radius)

  Diese Funktion legt den Radius des BoundingCircles fest.

- float GetRadius () const

  Diese Funktion gibt den Radius des BoundingCircles zurück.

- void SetPosition (float X, float Y)

  Diese Funktion legt die Position des BoundingCircles fest.

- float GetRealXPosition () const

  Diese Funktion gibt die X-Position des BoundingCircles zurück.

- float **GetXPosition** () const
- float GetRealYPosition ()

  Diese Funktion gibt die X-Position des BoundingCircles zurück.

- float **GetYPosition** ()
- void SetPositionAdd (float XAdd, float YAdd)

  Diese Funktion legt fest was auf den Position des BoundingCircles addiert wird.

- float GetXPositionAdd () const

  Diese Funktion gibt den Wert der auf X-Position des BoundingCircles addiert wird zurück.

- float GetYPositionAdd () const

  Diese Funktion gibt den Wert der auf Y-Position des BoundingCircles addiert wird zurück.

- bool Overlap (CBoundingCircle *Counterpart) const

  Diese Funktion überprüft ob sich die beiden BoundCirlces überschneiden.

- void **Draw** ()

4.5.1 Ausführliche Beschreibung

Bounding Circles für das Collision System.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 39 der Datei BoundingCircle.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

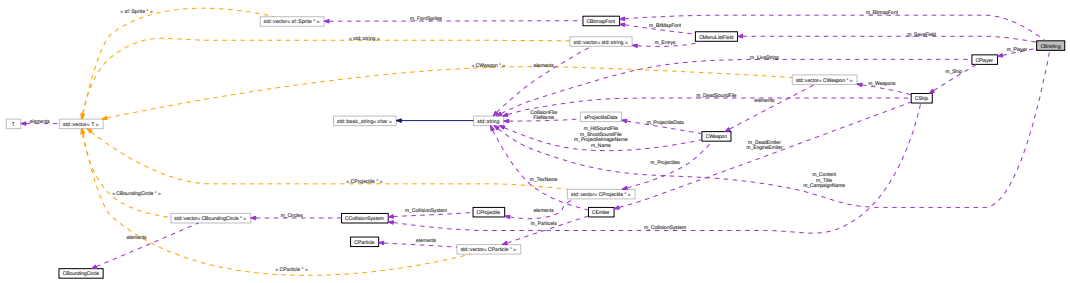
- BoundingCircle.hpp

4.6 CBriefing Klassenreferenz

Briefing.

`#include <Briefing.hpp>`

Zusammengehörigkeiten von CBriefing:



Öffentliche Methoden

- CBriefing ()

  Konstruktor und Destruktor.

- void Init (CBitmapFont *BitMapFont, int *MouseX, int *MouseY, bool *MouseDown, CPlayer *Player)

  Diese Funktion intialisiert das Briefing.

- void SetContent (std::string Title, std::string Content, std::string FileName, int ButtonType=BRIEF_-BUTTON_START, unsigned int FrameCount=1, int FrameDelay=50)

  Diese Funktion legt den Title des Briefings, den Inhalt des Briefings und das Logo fest.

- void Update ()

  Diese Funktion öffnet das Menü.

- bool Close ()

  Diese Funktion gibt zurück ob das Menü geschlossen werden soll.

- int GetButtonType () const

  Diese Funktion gibt den Button Type zurück.

- void SetCampaign (std::string Campaign)

  Diese Funktion legt die aktuelle Kampagne fest.

- std::string GetCampaign () const

  Diese Funktion gibt die aktuelle Kampagne zurück.

- void SetCurMission (int Mission)

  Diese Funktion legt die aktuelle Mission fest.

- void IncCurMission ()

Diese Funktion inkrementiert die aktuelle Mission.

- int GetCurMission () const

  Diese Funktion gibt die aktuelle Mission zurück.

4.6.1 Ausführliche Beschreibung

Briefing.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 54 der Datei Briefing.hpp.

4.6.2 Beschreibung der Konstruktoren und Destruktoren

4.6.2.1 CBriefing::CBriefing ()

Konstruktor und Destruktor.

Noch zu erledigen

initialisierungsliste

Definiert in Zeile 25 der Datei Briefing.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

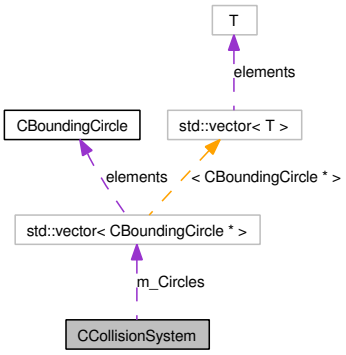
- Briefing.hpp
- Briefing.cpp

4.7 CCollisionSystem Klassenreferenz

Kollisionssystem.

`#include <CollisionSystem.hpp>`

Zusammengehörigkeiten von CCollisionSystem:



Öffentliche Methoden

- CCollisionSystem ()

  Konstrukor und Destruktor.

- void Init (std::string FileName)

  Diese Funktion intialsiert das KollisionsSystem aus der angeben Datei.

- void SetPosition (float X, float Y)

  Diese Funktion legt die Position des KollisionSystem fest.

- bool CheckCollision (const CCollisionSystem *CounterPart) const

  Diese Funktion überprüft ob zwei BoundingCircle der angeben KollisionsSysteme miteinander kollidieren.

- std::size_t GetCircleCount () const

  Diese Funktion gibt diee Anzahl der BoundingCircles des KollisionsSystems zurück.

- const CBoundingCircle * **GetCircle** (int Index) const
- float **GetWidth** () const
- void **Draw** ()

4.7.1 Ausführliche Beschreibung

Kollisionssystem.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 41 der Datei CollisionSystem.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- CollisionSystem.hpp
- CollisionSystem.cpp

4.8 CConsole Klassenreferenz

Konsole.

`#include <Console.hpp>`

Öffentliche Methoden

- void **printString** (std::string Out)
- std::string **readString** ()

4.8.1 Ausführliche Beschreibung

Konsole.

Autor:

Christoph Egger

Obsolete Klasse, to be removed

Definiert in Zeile 31 der Datei Console.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

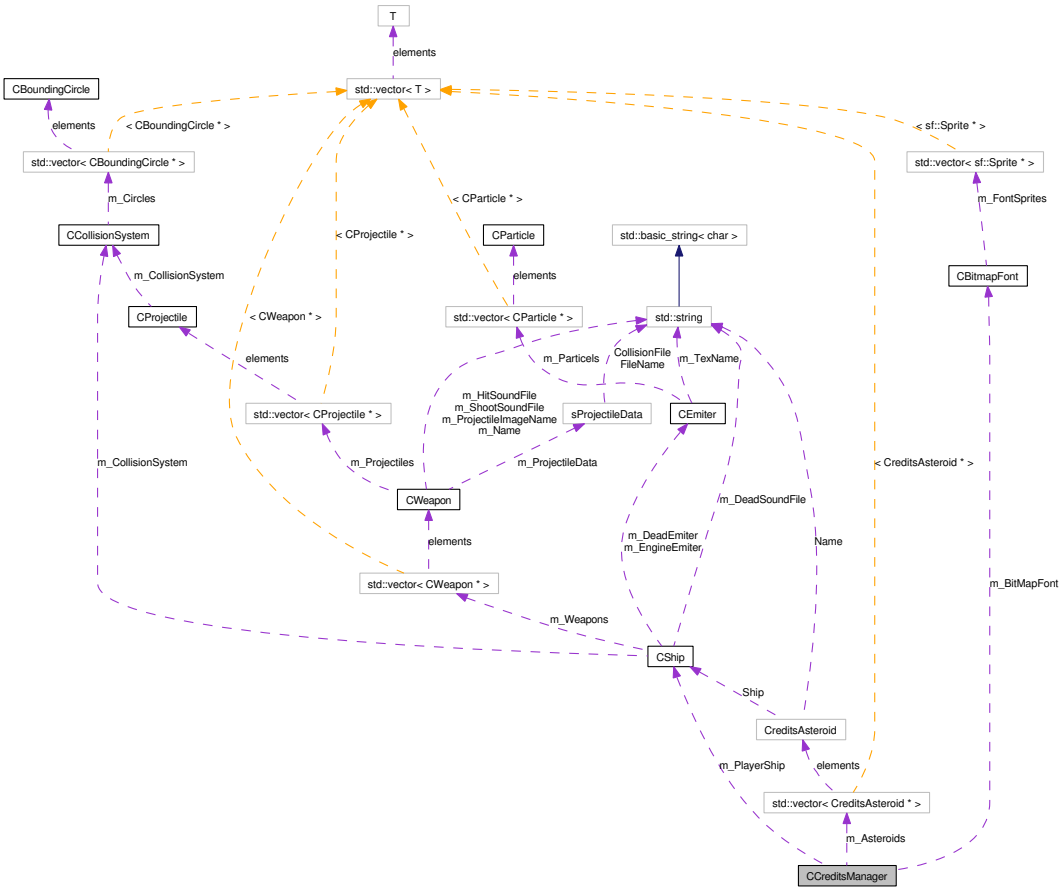
- Console.hpp
- Console.cpp

4.9 CCreditsManager Klassenreferenz

Credits Manager.

`#include <CreditsManager.hpp>`

Zusammengehörigkeiten von CCreditsManager:



Öffentliche Methoden

- void **Init** (CBitmapFont *BitMapFont)
- void **Draw** ()
- bool **IsFinished** () const

4.9.1 Ausführliche Beschreibung

Credits Manager.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 58 der Datei CreditsManager.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

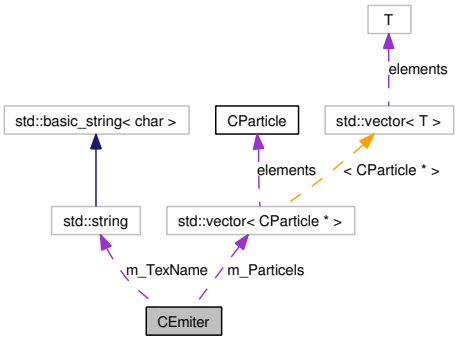
- CreditsManager.hpp
- CreditsManager.cpp

4.10 CEmiter Klassenreferenz

Partikel Emitter.

`#include <Emiter.hpp>`

Zusammengehörigkeiten von CEmiter:



Öffentliche Methoden

- CEmiter ()

  Konstruktor und Destruktor.

- void Init (const std::string &TexName, float FrameWidth=-1, float FrameHeight=-1, int FrameCount=1)

  Diese Funktion intialisiert den Emiter.

- void InitFromFile (const std::string &FileName)

  Diese Funktion intialisiert den Emiter aus der angebenen Datei.

- void Activate (bool Activate=true)

  Diese Funktion aktiviert den Emiter.

- bool IsActivate () const

  Diese Funktion gibt zurück ob der Emiter aktiv ist.

- void SetPosition (float X, float Y)

  Diese Funktion legt die EmiterPosition fest.

- float GetXPosition () const

  Diese Funktion gibt die X-Position zurück.

- float GetYPosition () const

  Diese Funktion gibt die Y-Position zurück.

- void SetSize (float X, float Y)

  Diese Funktion legt die Größe des Emiters fest.

- float GetXSize () const

Diese Funktion gibt die X-Größe des Emiters zurück.

- float GetYSize () const

  Diese Funktion gibt die Y-Größe des Emiters zurück.

- void Update ()

  Diese Funktion aktualisiert den Emiter.

- void Draw ()

  Diese Funktion zeichnet die Partikel des Emiters.

- int GetParticleCount () const

  Diese Funktion gibt die Anzahl der Partikel des Emiters zurück.

- void SetSpawnDelay (int SpawnDelay)

  Diese Funktion legt die Zeit die zwischen dem erstellen von 2 Partikeln gewaretet wird fest.

- int GetSpawnDelay () const

  Diese Funktion gibt die Zeit die zwischen dem erstellen von 2 Partikeln gewaretet wird zurück.

- void SetFrameDelay (int FrameDelay)

  Diese Funktion legt die Zeit in Millisekunden die zwischen dem inkrementieren der PartikelFrames gewartet wird fest.

- int GetFrameDelay () const

  Diese Funktion gibt die Zeit in Millisekunden die zwischen dem inkrementieren der PartikelFrames gewartet wird zurück.

- void SetLifeSpan (int LifeSpan)

  Diese Funktion legt die Lebensspanne der Partikel fest.

- int GetLifeSpan () const

  Diese Funktion gibt die Lebensspanne der Partikel zurück.

- void SetStartScale (float ScaleX, float ScaleY)

  Diese Funktion legt die StartSkalirung der Partikel fest.

- float GetStartScaleX () const

  Diese Funktion gibt die X-Achsen Startskalriung der Partikel zurück.

- float GetStartScaleY () const

  Diese Funktion gibt die Y-Achsen Startskalriung der Partikel zurück.

- void SetTargetScale (float ScaleX, float ScaleY)

  Diese Funktion legt die StartSkalirung der Partikel fest.

- float GetTargetScaleX () const

  Diese Funktion gibt die X-Achsen Zielskalriung der Partikel zurück.

- float GetTargetScaleY () const

Diese Funktion gibt die Y-Achsen Zielskalriung der Partikel zurück.

- void SetStartAlpha (int Alpha)

  Diese Funktion legt den StartAlpha wert der Partikel fest.

- int GetStartAlpha () const

  Diese Funktion gibt den StartAlpha wert der Partikel zurück.

- void SetEndAlpha (int Alpha)

  Diese Funktion legt den StartAlpha wert der Partikel fest.

- int GetEndAlpha () const

  Diese Funktion gibt den StartAlpha wert der Partikel zurück.

- void SetStartColor (int r, int g, int b)

  Diese Funktion legt die StartFarbwerte der Partikel fest.

- int GetStartColorR () const

  Diese Funktion gibt den StartRot wert der Partikel zurück.

- int GetStartColorG () const

  Diese Funktion gibt den StartGrün wert der Partikel zurück.

- int GetStartColorB () const

  Diese Funktion gibt den StartBlau wert der Partikel zurück.

- void SetEndColor (int r, int g, int b)

  Diese Funktion legt die EndFarbwerte der Partikel fest.

- int GetEndColorR () const

  Diese Funktion gibt den EndRot wert der Partikel zurück.

- int GetEndColorG () const

  Diese Funktion gibt den EndGrün wert der Partikel zurück.

- int GetEndColorB () const

  Diese Funktion gibt den EndBlau wert der Partikel zurück.

- void SetStartVelX (float StartVelX)

  Diese Funktion legt die Startgeschwindigkeit der Partikel fest.

- float GetStartVelX () const

  Diese Funktion gibt die Startgeschwindigkeit der Partikel zurück.

- void SetEndVelX (float EndVelX)

  Diese Funktion legt die Endgeschwindigkeit der Partikel fest.

- float GetEndVelX () const

  Diese Funktion gibt die Endgeschwindigkeit der Partikel zurück.

- void SetStartVelY (float StartVelY)

  Diese Funktion legt die Startgeschwindigkeit der Partikel fest.

- float GetStartVelY () const

  Diese Funktion gibt die Startgeschwindigkeit der Partikel zurück.

- void SetEndVelY (float EndVelY)

  Diese Funktion legt die Endgeschwindigkeit der Partikel fest.

- float GetEndVelY () const

  Diese Funktion gibt die Endgeschwindigkeit der Partikel zurück.

- void SetVelXMulti (float Low, float High)

  Diese Funktion legt die Mulitplikationen der Geschwindigkiet auf der X-Achse fest.

- float GetLowVelXMulti () const

  Diese Funktion gibt die untere Multiplikation der Geschwindigkiet auf der X-Achse zurück.

- float GetHighVelXMulti () const

  Diese Funktion gibt die obere Multiplikation der Geschwindigkiet auf der X-Achse zurück.

- void SetVelYMulti (float Low, float High)

  Diese Funktion legt die Mulitplikationen der Geschwindigkiet auf der Y-Achse fest.

- float GetLowVelYMulti () const

  Diese Funktion gibt die untere Multiplikation der Geschwindigkiet auf der Y-Achse zurück.

- float GetHighVelYMulti () const

  Diese Funktion gibt die obere Multiplikation der Geschwindigkiet auf der Y-Achse zurück.

4.10.1 Ausführliche Beschreibung

Partikel Emitter.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 43 der Datei Emiter.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

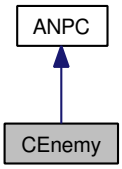
- Emiter.hpp
- Emiter.cpp

4.11 CEnemy Klassenreferenz

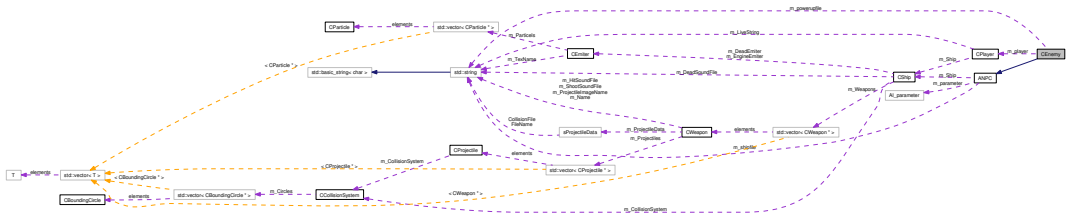
Feindlicher NPC.

`#include <Enemy.hpp>`

Klassendiagramm für CEnemy:



Zusammengehörigkeiten von CEnemy:



Öffentliche Methoden

- **CEnemy** (unsigned int starttime, int posx, int posy, Strength theStrength, std::string shipfile, std::string powerupfile, AI_parameter ∗parameter)
- std::string **getPowerUp** ()
- bool **init** (global_data_pointers Data, CPlayer ∗player, strength ∗Strength)

4.11.1 Ausführliche Beschreibung

Feindlicher NPC.

Autor:

Christoph Egger

Definiert in Zeile 39 der Datei Enemy.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

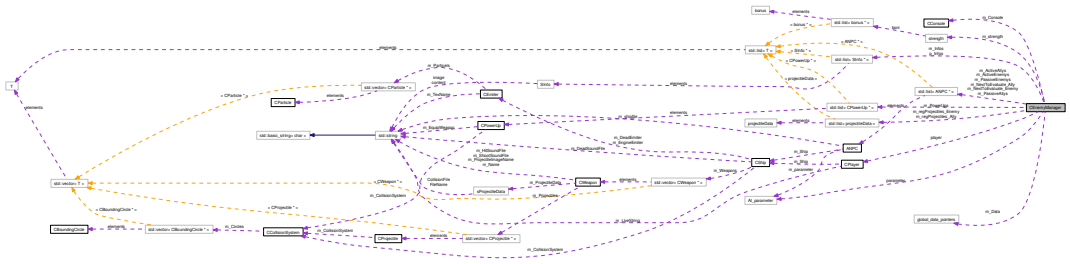
- Enemy.hpp
- Enemy.cpp

4.12 CEnemyManager Klassenreferenz

Verwaltung der NPCs.

`#include <EnemyManager.hpp>`

Zusammengehörigkeiten von CEnemyManager:



Öffentliche Methoden

- bool **reset** ()
- int **clearPowerups** ()
- bool init (std::string filename_waves, std::string filename_parameter, global_data_pointers Data, CConsole *Console, CPlayer *player, bool *play_Info, std::list< SInfo * > *Infos)
- bool **update** ()
- bool **hasWon** ()
- strength * **getStrength** (int level)

Klassen

- struct **less_S_no_fight**
- struct **S_no_fight**

4.12.1 Ausführliche Beschreibung

Verwaltung der NPCs.

Autor:

Christoph Egger

Definiert in Zeile 50 der Datei EnemyManager.hpp.

4.12.2 Dokumentation der Elementfunktionen

4.12.2.1 bool CEnemyManager::init (std::string *filename_waves*, std::string *filename_parameter*, global_data_pointers *Data*, CConsole * *Console*, CPlayer * *player*, bool * *play_Info*, std::list< SInfo * > * *Infos*)

Parameter: Noch zu erledigen

Console to be removed

Definiert in Zeile 26 der Datei EnemyManager.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

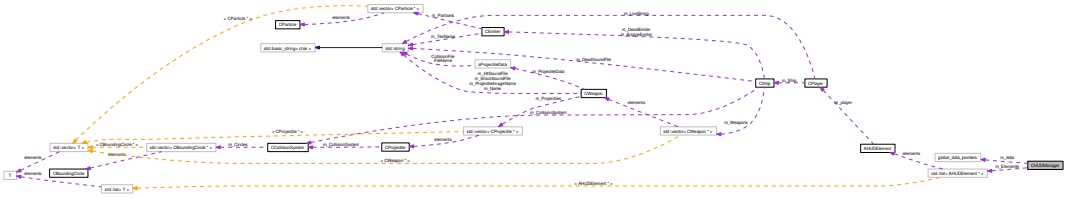
- EnemyManager.hpp
- EnemyManager.cpp

4.13 CHUDManager Klassenreferenz

Verwaltung der HUDs.

`#include <HUD-Manager.hpp>`

Zusammengehörigkeiten von CHUDManager:



Öffentliche Methoden

- void **init** (global_data_pointers pointers, CBitmapFont *bmFont)
- AHUDElement * **registerHUD** (int maxValue, const int *ActualValue, HUDType Type, int posX, int posY, int width, int height, int size=15)
- AHUDElement * **registerHUD** (float maxValue, const float *ActualValue, HUDType Type, int posX, int posY, int width, int height, int size=15)
- AHUDElement * registerHUD (const std::string *data, HUDType Type, int posX, int posY, CBitmapFont *bmFont, int fontsize=15, int size=138)
- std::string * **registerHUD** (int posX, int posY, CBitmapFont *bmFont, HUDType Type=SET_TEXT, int size=15)
- AHUDElement * **registerHUD** (bool *play_Info, std::list< SInfo * > *Infos, CBitmapFont *bmFont)
- void **draw** ()

4.13.1 Ausführliche Beschreibung

Verwaltung der HUDs.

Autor:

Christoph Egger

Definiert in Zeile 37 der Datei HUD-Manager.hpp.

4.13.2 Dokumentation der Elementfunktionen

4.13.2.1 AHUDElement * CHUDManager::registerHUD (const std::string * *data*, HUDType *Type*, int *posX*, int *posY*, CBitmapFont * *bmFont*, int *fontsize* = 15, int *size* = 138)

Noch zu erledigen

???

Definiert in Zeile 90 der Datei HUD-Manager.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

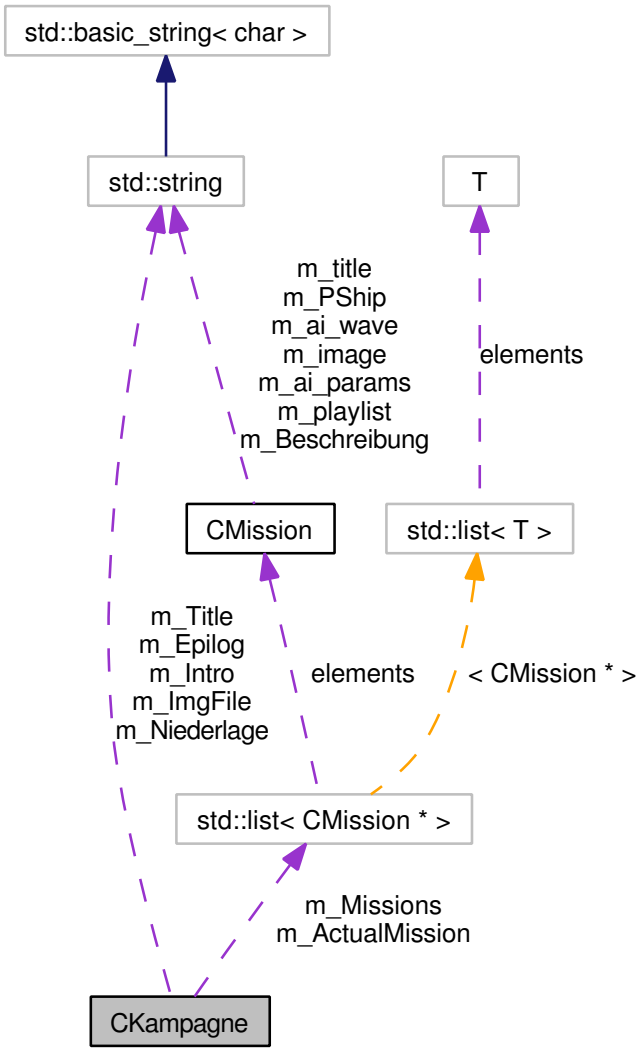
- HUD-Manager.hpp
- HUD-Manager.cpp

4.14 CKampagne Klassenreferenz

Kampagne.

`#include <Kampagne.hpp>`

Zusammengehörigkeiten von CKampagne:



Öffentliche Methoden

- bool **init** (std::string filename)
- CMission ∗ **get_Mission** () const
- bool **nextMission** ()
- std::string **get_Intro** ()
- std::string **get_Epilog** ()
- std::string **get_Niederlage** ()
- std::string **get_ImgFile** ()
- std::string **get_Title** ()

- int **getLeben** ()
- int **getFramecount** ()
- int **getFramedelay** ()
- std::size_t **get_MissionIndex** ()
- bool **set_MissionIndex** (int index)

4.14.1 Ausführliche Beschreibung

Kampagne.

Autor:

Christoph Egger

Definiert in Zeile 34 der Datei Kampagne.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

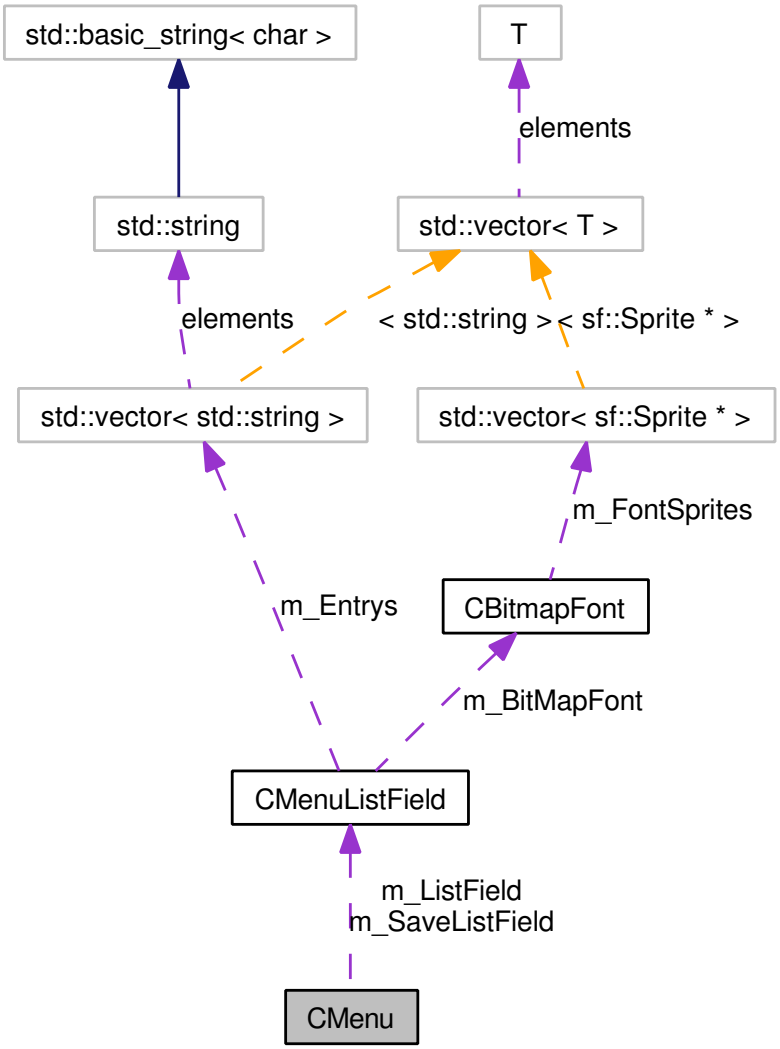
- Kampagne.hpp
- Kampagne.cpp

4.15 CMenu Klassenreferenz

Menü.

`#include <Menu.hpp>`

Zusammengehörigkeiten von CMenu:



Öffentliche Methoden

- CMenu ()

  Konstruktor und Destruktor.

- void Init (CBitmapFont *BitMapFont, int *MouseX, int *MouseY, bool *MouseDown)

  Diese Funktion intialisiert das Menü.

- std::string Update ()

  Diese Funktion öffnet das Menü.

- int Close ()

  Diese Funktion gibt zurück ob das Menü geschlossen werden soll.

- void SetSubMenu (int Menu)

  Diese Funktion legt das Untermenü fest.

4.15.1 Ausführliche Beschreibung

Menü.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 53 der Datei Menu.hpp.

4.15.2 Beschreibung der Konstruktoren und Destruktoren

4.15.2.1 CMenu::CMenu ()

Konstruktor und Destruktor.

Noch zu erledigen

Initialisierungsliste

Definiert in Zeile 25 der Datei Menu.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

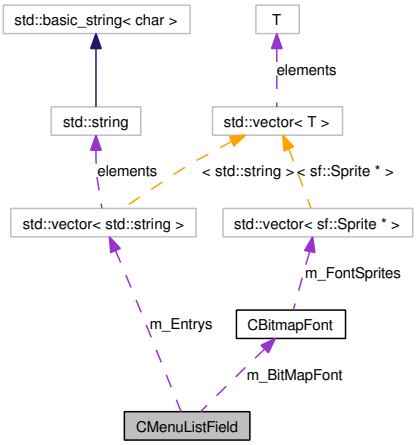
- Menu.hpp
- Menu.cpp

4.16 CMenuListField Klassenreferenz

Liste für die Menüs.

`#include <MenuListField.hpp>`

Zusammengehörigkeiten von CMenuListField:



Öffentliche Methoden

- void **Init** (CBitmapFont *BitMapFont, std::string ListSelectionName)
- void **AddEntry** (std::string Entry)
- void **SetPosition** (float X, float Y)
- float **GetXPosition** () const
- float **GetYPosition** () const
- void **SetSize** (float SizeX, float SizeY)
- float **GetXSize** ()
- float **GetYSize** ()
- void **SelectEntryWithScroll** (std::size_t Index)
- void **SelectEntryWithoutScroll** (std::size_t Index)
- void **Scroll** (int Scroll)
- void **SetScroll** (int Scroll)
- bool **CanScrollUp** () const
- bool **CanScrollDown** () const
- std::string **GetCurEntry** () const
- std::string **GetEntry** (std::size_t Index) const
- int **GetCurSelectIndex** () const
- void **Draw** ()
- void **Clear** ()
- std::size_t **GetEntryCount** () const

4.16.1 Ausführliche Beschreibung

Liste für die Menüs.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 44 der Datei MenuListField.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

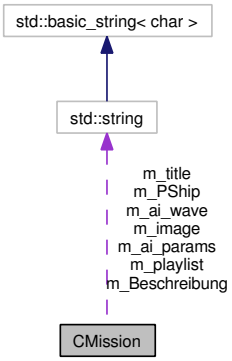
- MenuListField.hpp
- MenuListField.cpp

4.17 CMission Klassenreferenz

Einzelne Missionen.

`#include <Mission.hpp>`

Zusammengehörigkeiten von CMission:



Öffentliche Methoden

- **CMission** (std::string ai_wave, std::string ai_params, std::string Beschreibung, std::string Title, std::string PShip, std::string playlist[2], std::string image="", int framecount=1, int framedelay=10)
- std::string **get_ai_wave** ()
- std::string **get_ai_params** ()
- std::string **get_Beschreibung** ()
- std::string **get_Title** ()
- std::string **getPlayerShip** ()
- std::string **getBosPlaylist** ()
- std::string **getNormalPlaylist** ()
- std::string **getImage** ()
- int **getFramecount** ()
- int **getFramedelay** ()

4.17.1 Ausführliche Beschreibung

Einzelne Missionen.

Autor:

Christoph Egger

Definiert in Zeile 27 der Datei Mission.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- Mission.hpp

4.18 CParticle Klassenreferenz

Partikel.

`#include <Particle.hpp>`

Öffentliche Methoden

- CParticle ()

  Konstruktor und Destruktor.

- void Init (const std::string &TexName, float FrameWidth=-1, float FrameHeight=-1, int FrameCount=1)

  Diese Funktion intialisiert den Partikel.

- void SetPosition (float X, float Y)

  Diese Funktion legt die Position des Partikels fest.

- bool Remove ()

  Diese Funktion gibt true zurï¿½ck wenn der Partikel gelï¿½scht werden sollte.

- void Update ()

  Diese Funktion aktualisiert den Partikel.

- void Draw ()

  Diese Funktion zeichnet die Partikel.

- void SetLifeSpan (int LifeSpan)

  Diese Funktion legt die Lebensspanne des Partikels fest.

- void SetScale (float X, float Y)

  Diese Funktion legt die Skalirung des Partikels fest.

- void SetTargetScale (float X, float Y)

  Diese Funktion legt die EndSkalirung des Partikels fest.

- void SetStartAlpha (int Alpha)

  Diese Funktion legt den Startalpha wert des Partikels fest.

- void SetEndAlpha (int Alpha)

  Diese Funktion legt den Endalpha wert des Partikels fest.

- void SetStartRed (int Red)

  Diese Funktion legt den StartRotWert des Partikels fest.

- void SetEndRed (int Red)

  Diese Funktion legt den EndRotWert des Partikels fest.

- void SetStartGreen (int Green)

  Diese Funktion legt den StartGrï¿½nWert des Partikels fest.

- void SetEndGreen (int Green)

  Diese Funktion legt den EndGrï¿½nWert des Partikels fest.

- void SetStartBlue (int Blue)

  Diese Funktion legt den StartBlauWert des Partikels fest.

- void SetEndBlue (int Blue)

  Diese Funktion legt den EndBlauWert des Partikels fest.

- void SetStartVelX (float StartVelX)

  Diese Funktion legt die Startgeschwindigkeit des Partikels fest.

- void SetEndVelX (float EndVelX)

  Diese Funktion legt die Endgeschwindigkeit des Partikels fest.

- void SetStartVelY (float StartVelY)

  Diese Funktion legt die Startgeschwindigkeit des Partikels fest.

- void SetEndVelY (float EndVelY)

  Diese Funktion legt die Endgeschwindigkeit des Partikels fest.

- void SetAniSpeed (int AniSpeed)

  Diese Funktion legt die Zeit in Millisekunden zwischen dem inkrementieren des Frames gewartet wird fest.

4.18.1 Ausführliche Beschreibung

Partikel.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 41 der Datei Particle.hpp.

4.18.2 Dokumentation der Elementfunktionen

4.18.2.1 void CParticle::Init (const std::string & *TexName*, float *FrameWidth* = `-1`, float *FrameHeight* = `-1`, int *FrameCount* = `1`)

Diese Funktion intialisiert den Partikel.

Noch zu erledigen

m_Sprite.Init(lpDevice,TexManager,TexName,FrameWidth,FrameHeight,FrameCount);

Definiert in Zeile 39 der Datei Particle.cpp.

4.18.2.2 void CParticle::Update ()

Diese Funktion aktualisiert den Partikel.

Noch zu erledigen

m_Sprite.IncFrame();

Definiert in Zeile 52 der Datei Particle.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

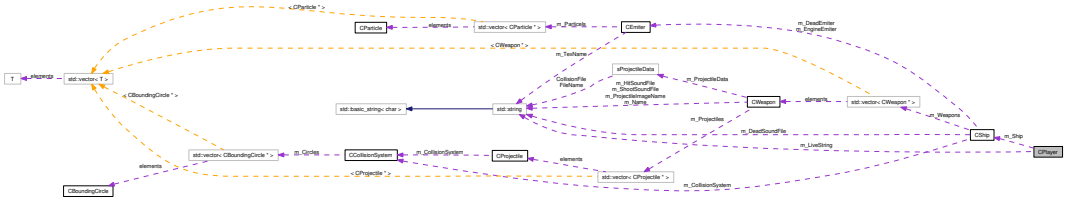
- Particle.hpp
- Particle.cpp

4.19 CPlayer Klassenreferenz

Spielerklasse.

`#include <Player.hpp>`

Zusammengehörigkeiten von CPlayer:



Öffentliche Methoden

- CPlayer ()

  Konstruktor und Destruktor.

- void Init (const std::string &FileName)

  Diese Funktion intialisiert den Spieler.

- void Draw ()

  Diese Funktion zeichnet das Schiff des Spielers.

- void Update ()

  Diese Funktion akualisiert den Spieler.

- float GetXPosition () const

  Diese Funktion gibt die X-Position des Spielers zurück.

- float GetYPosition () const

  Diese Funktion gibt die Y-Position des Spielers zurück.

- CShip * GetShip ()

  Diese Funktion gibt das Shiff zurück.

- const CShip * **GetShip** () const
- void EquipWeapon (std::string Name)

  Diese Funktion rüstet die angeben Waffe aus.

- void ResetWeapons ()

  Diese Funktion setzt die Ausrüstung zurück.

- bool CheckCollision (CPowerUp *CounterPart)

  Diese Funktion überprüft ob das angeben PowerUp mit dem Spieler kollidiert.

- bool CheckCollision (CShip *CounterPart)

- bool IsDead () const

  Diese Funktion gibt zurück ob der Spieler tot ist.

- std::size_t GetProjectilCount () const

  Diese Funktion gibt die ProjektilAnzahl des SpielerSchiffes zurück.

- float GetProjectilX (int Projectil) const

  Diese Funktion gibt die X-Koordinate des Angebenen Projektil des SpielerSchiffes zurück.

- float GetProjectilY (int Projectil) const

  Diese Funktion gibt die X-Koordinate des Angebenen Projektil des SpielerSchiffes zurück.

- float GetProjectilDamage (int Projectil) const

  Diese Funktion gibt die X-Koordinate des Angebenen Projektil des SpielerSchiffes zurück.

- float GetProjectilWidth (int Projectil) const

  Diese Funktion gibt die X-Koordinate des Angebenen Projektil des SpielerSchiffes zurück.

- std::size_t GetWeaponCount () const

  Diese Funktion gibt WaffenAnzahl des Spielerschiffs zurück.

- int GetWeaponDamage (int Weapon) const

  Diese Funktion gibt den WaffenSchaden der angeben Waffe zurück.

- float GetCollisionSystemWidth () const

  Diese Funktion gibt die bretie des KollisionSystems des SpielerSchiffes zurück.

- std::size_t GetWeaponTypeCount () const

  Diese Funktion gibt die Waffentypen zurück.

- std::string GetCurWeaponNameOfType (std::size_t GroupID)

  Diese Funktion gibt die momentan ausgerüstete Waffe des angeben Waffen types zurück.

- const CWeapon * GetCurWeaponOfType (std::size_t GroupID)

  Diese Funktion gibt die momentan ausgerüstete Waffe des angeben Waffen types zurück.

- bool CanBeRemoved ()

  Diese Funktion gibt zurück ob der Spieler entfernt werden kann ist.

- const std::string * GetLiveString () const

  Diese Funktion gibt Leben des Spielers als String zurück.

- int GetLives () const

  Diese Funktion gibt die Anzahl der Leben zurück.

- void SetLives (int Lives)

  Diese Funktion legt die Anzahl der Leben fest.

4.19.1 Ausführliche Beschreibung

Spielerklasse.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 45 der Datei Player.hpp.

4.19.2 Dokumentation der Elementfunktionen

4.19.2.1 bool CPlayer::CheckCollision (CShip * *CounterPart*) [inline]

Diese Funktion überprüft ob die Projektil des Spieler-Schiffes mit dem angebenen Schiff kollidieren wenn das Schiff dadurch sterben würde wird true zurück gegeben

Definiert in Zeile 83 der Datei Player.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

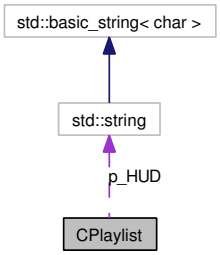
- Player.hpp
- Player.cpp

4.20 CPlaylist Klassenreferenz

Playlist für Hintergrundmusik.

`#include <Playlist.hpp>`

Zusammengehörigkeiten von CPlaylist:



Öffentliche Methoden

- int **init** (std::string filename)
- void **HUDout** (std::string *ausgabe)
- std::string **getRandSong** ()

Klassen

- struct **SSongInfo**

4.20.1 Ausführliche Beschreibung

Playlist für Hintergrundmusik.

Autor:

Christoph Egger

Definiert in Zeile 29 der Datei Playlist.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

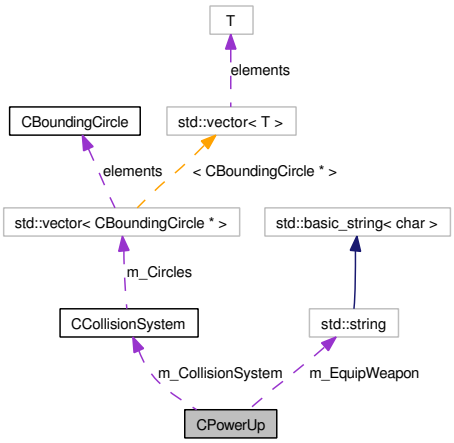
- Playlist.hpp
- Playlist.cpp

4.21 CPowerUp Klassenreferenz

Power Ups.

`#include <PowerUp.hpp>`

Zusammengehörigkeiten von CPowerUp:



Öffentliche Methoden

- CPowerUp ()
  Konstruktor und Destruktor.
- void Init (const std::string &FileName)
  Diese Funktion initalisiert das PowerUp.
- void SetPosition (float X, float Y)
  Diese Funktion legt die Position des PowerUps fest.
- std::string GetEquipWeapon () const
  Diese Funktion gibt die Waffe die ausgerüstete werden soll zurück.
- int GetWeaponGroupID () const
  Diese Funktion gibt die WaffenGruppenID zurück.
- int GetRepair () const
  Diese Funktion gibt zurück wie stark der Bonus das Schiff repariert.
- void Draw ()
  Diese Funktion zeichnet das PowerUp.
- void Update ()
  Diese Funktion aktualisiert das PowerUp.
- const CCollisionSystem * GetCollisionSystem () const
  Diese Funktion gibt das KollisionSystem zurück.

4.21.1 Ausführliche Beschreibung

Power Ups.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 41 der Datei PowerUp.hpp.

4.21.2 Dokumentation der Elementfunktionen

4.21.2.1 void CPowerUp::Update ()

Diese Funktion aktualisiert das PowerUp.

Noch zu erledigen

m_Sprite.IncFrame();

Definiert in Zeile 119 der Datei PowerUp.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

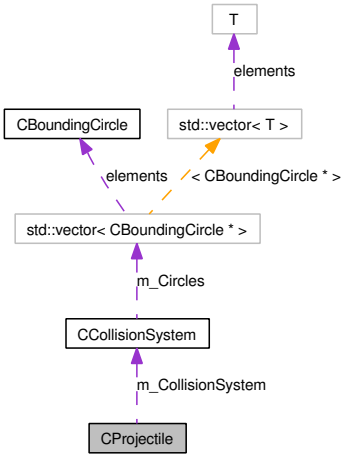
- PowerUp.hpp
- PowerUp.cpp

4.22 CProjectile Klassenreferenz

Projektile.

`#include <Projectile.hpp>`

Zusammengehörigkeiten von CProjectile:



Öffentliche Methoden

- void **Init** (sProjectileData ProjectileData)
- void **Draw** ()
- void Update ()
- float **GetXPosition** () const
- float **GetYPosition** () const
- const CCollisionSystem * **GetCollisionSystem** () const
- float **GetCollisionSystemWidth** () const

4.22.1 Ausführliche Beschreibung

Projektile.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 63 der Datei Projectile.hpp.

4.22.2 Dokumentation der Elementfunktionen

4.22.2.1 void CProjectile::Update ()

Noch zu erledigen

Finde eine Schlaue Idee wegen IncFrame()

Definiert in Zeile 76 der Datei Projectile.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

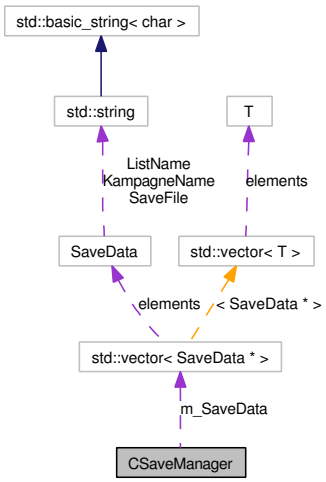
- Projectile.hpp
- Projectile.cpp

4.23 CSaveManager Klassenreferenz

Verwaltung gespeicherter Spielstände.

`#include <SaveManager.hpp>`

Zusammengehörigkeiten von CSaveManager:



Öffentliche Methoden

- void **Save** (std::string Folder, std::string Campaign, int Mission, int Lives)
- void **SaveFile** (std::string File, std::string Campaign, int Mission, int Lives)
- void **ReadSavesFromFolder** (std::string Folder)
- std::size_t **GetSaveCount** () const
- SaveData ∗ **GetSaveData** (std::size_t Index)
- SaveData ∗ **ReadSave** (std::string FileName)

4.23.1 Ausführliche Beschreibung

Verwaltung gespeicherter Spielstände.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 53 der Datei SaveManager.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

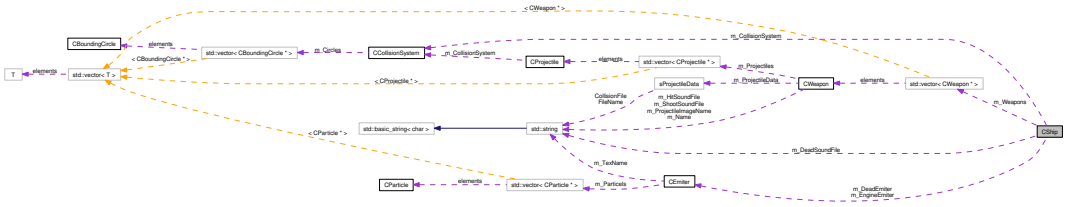
- SaveManager.hpp
- SaveManager.cpp

4.24 CShip Klassenreferenz

Shiffe.

`#include <Ship.hpp>`

Zusammengehörigkeiten von CShip:



Öffentliche Methoden

- CShip ()

  Konstruktor und Destruktor.

- void Init (const std::string &FileName)

  Diese Funktion intialisiert das Schiff.

- void Draw ()

  Diese Funktion zeichnet das Schiff.

- void Shoot ()

  Diese Funktion lässt das Schiff schießen.

- void SetPosition (float X, float Y)

  Diese Funktion legt die Position des Schiffs fest.

- void MoveX (int Direction)

  Diese Funktion bewegt das Schiff auf der X-Achse.

- void MoveY (int Direction)

  Diese Funktion bewegt das Schiff auf der Y-Achse.

- float GetXPosition () const

  Diese Funktion gibt die X-Position des Schiffes zurück.

- float GetYPosition () const

  Diese Funktion gibt die X-Position des Schiffes zurück.

- const int * GetCurrentArmor () const

  Gibt die verbleibende Rüstung / Lebenspunkte des Schiffs zurück.

- void SetCurArmor (int CurArmor)

  Diese Funktion legt die Rüstung vom Schiff fest.

- int GetMaxArmor () const

  Gibt die maximale Rüstung / Lebenspunkte des Schiffs zurück.

- const int * GetMaxArmorPtr () const

  Gibt die verbleibende Rüstung / Lebenspunkte des Schiffs zurück.

- void EquipWeapon (std::string Name)
- void EquipWeapon (int GroupID)
- void ResetWeapons ()

  Diese Funktion setzt die Waffenausrüstung wieder auf den Standard wert.

- bool CheckCollision (CPowerUp *CounterPart)

  Diese Funktion überprüft ob das angeben PowerUp mit dem Schiff kollidiert.

- bool CheckCollision (CShip *CounterPart)
- void Repair (int Repair)

  Diese Funktion repariert das Schiff.

- bool Damage (int Damage)

  Diese Funktion beschädigt das Schiff, wird es zerstört wird true zurückgegeben.

- CCollisionSystem * GetCollisionSystem ()

  Diese Funktion gibt das KollisionSystem des Schiffes zurück.

- float GetCollisionSystemWidth ()

  Diese Funktion gibt die KollisionSystem breite des Schiffes zurück.

- bool IsDead ()

  Diese Funktion gibt zurück ob das Schiff zerstört ist.

- bool CanBeRemoved () const

  Diese Funktion gibt zurück ob das Schiff gelöscht werden kann.

- int GetProjectilCount () const

  Diese Funktion gibt die Projetiklanzahl aller Waffen des Schiffes zurück.

- float GetProjectilXPosition (int Projectil) const

  Diese Funktion gibt X-Koordinate des angeben Projektils zurück.

- float GetProjectilYPosition (int Projectil) const

  Diese Funktion gibt X-Koordinate des angeben Projektils zurück.

- int GetProjectilDamage (int Projectil) const

  Diese Funktion gibt Schaden der durch das angeben Projektils verursacht wird.

- float GetProjectilWidth (int Projectil) const

  Diese Funktion gibt Breite des angeben Projektils zurück.

- int GetProjectileTarget (int YDistance) const

Diese Funktion gibt relativ zum Schiff den mittlewert der Schiffwaffentreffen zurück.

- int GetWeaponCount () const
  Diese Funktion gibt die Anzahl der Waffen zurück.
- int GetWeaponDamage (std::size_t Weapon) const
  Diese Funktion gibt den Schaden der durhc die angeben Waffe verursacht wird zurück.
- std::string GetWeaponName (std::size_t Weapon) const
  Diese Funktion gibt den Name der angeben Waffe zurück.
- int GetWeaponTypeCount () const
  Diese Funktion gibt die Waffentypen zurück.
- std::string GetCurWeaponNameOfType (std::size_t GroupID) const
  Diese Funktion gibt die momentan ausgerüstete Waffe des angeben Waffen types zurück.
- const CWeapon * GetCurWeaponOfType (std::size_t GroupID) const
  Diese Funktion gibt die momentan ausgerüstete Waffe des angeben Waffen types zurück.
- void ClearProjectils ()
  Diese Funktion löscht alle Projektile des Schiffs.
- void SetInvulnerable (bool Invulnerable)
  Diese Funktion legt fest ob das Schiff unverwundbar ist.
- bool IsInvulnerable () const
  Diese Funktion gibt zurück ob das Schiff unverwundbar ist.
- void SetAlpha (float Alpha)
  Diese Funktion legt den Alphawert des Schiffes fest.
- void SetAniDelay (int AniDelay)
  Diese Funktion legt fest wie lange zwischen den Animationsframes gewartet wird.
- int GetAniDelay () const
  Diese Funktion gibt zurück wie lange zwischen den Animationsframes gewartet wird.
- int GetDeadTime () const
  Diese Funktion gibt zurück wann das Schiff zerstört wurde.

4.24.1 Ausführliche Beschreibung

Shiffe.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 48 der Datei Ship.hpp.

4.24.2 Dokumentation der Elementfunktionen

4.24.2.1 void CShip::EquipWeapon (std::string *Name*)

Diese Funktion rüstet die Waffe mit dem angebenen Namen aus Waffen aus der selben GruppenID mit anderem Namen werden entrüstet

Definiert in Zeile 257 der Datei Ship.cpp.

4.24.2.2 void CShip::EquipWeapon (int *GroupID*)

Diese Funktion rüstet die Waffe mit dem angebenen Namen aus Waffen aus der selben GruppenID mit anderem Namen werden entrüstet

Noch zu erledigen

genauer überfliegen, evtl umschreiben

Definiert in Zeile 277 der Datei Ship.cpp.

4.24.2.3 bool CShip::CheckCollision (CShip * *CounterPart*)

Diese Funktion überprüft ob die Projektil des Schiffes mit dem angebenen Schiff kollidieren wenn das Schiff dadurch sterben würde wird true zurück gegeben

Definiert in Zeile 351 der Datei Ship.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Ship.hpp
- Ship.cpp

4.25 CStar Klassenreferenz

Stern.

`#include <Star.hpp>`

Öffentliche Methoden

- CStar ()

  Konstruktor und Destruktor.

- void Init (const std::string &FileName, int Layer, bool Planet=false, int FieldWidth=SCREEN_X_-SIZE-HUD_SIZE_X)

  Diese Funktion intialisiert den Stern.

- void InitAsteroid (AsteroidData ∗Data, int Layer, float XSpeed, float YSpeed, float X, float Y)

  Diese Funktion intialisiert einen Asteroiden.

- void Draw ()

  Diese Funktion zeichnet und aktualisiert den Stern.

- int GetLayer () const

  Diese Funktion gibt den Layer des Sterns zurück.

- bool IsPlanet () const

  Diese Funktion gibt zurück ob der Stern ein Planet ist.

- bool CanBeRemoved () const

  Diese Funktion gibt zurück ob der Stern entfernt werden kann.

- void SetFieldWidth (int FieldWidth)

  Diese Funktion legt die Sternenfeld Breite fest.

- void SetPosition (float X, float Y)

  Diese Funktion legt die Position fest.

4.25.1 Ausführliche Beschreibung

Stern.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 55 der Datei Star.hpp.

4.25.2 Dokumentation der Elementfunktionen

4.25.2.1 void CStar::Init (const std::string & *FileName*, int *Layer*, bool *Planet* = `false`, int *FieldWidth* = `SCREEN_X_SIZE - HUD_SIZE_X`)

Diese Funktion intialisiert den Stern.

m_Star.Init(lpDevice,TexManager, Data->FileName, Data->FrameX, Data->FrameY, Data->FrameCount);

Definiert in Zeile 38 der Datei Star.cpp.

4.25.2.2 void CStar::InitAsteroid (AsteroidData * *Data*, int *Layer*, float *XSpeed*, float *YSpeed*, float *X*, float *Y*)

Diese Funktion intialisiert einen Asteroiden.

Noch zu erledigen

m_Star.Init(Data->FileName, Data->FrameX, Data->FrameY, Data->FrameCount);

Definiert in Zeile 71 der Datei Star.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

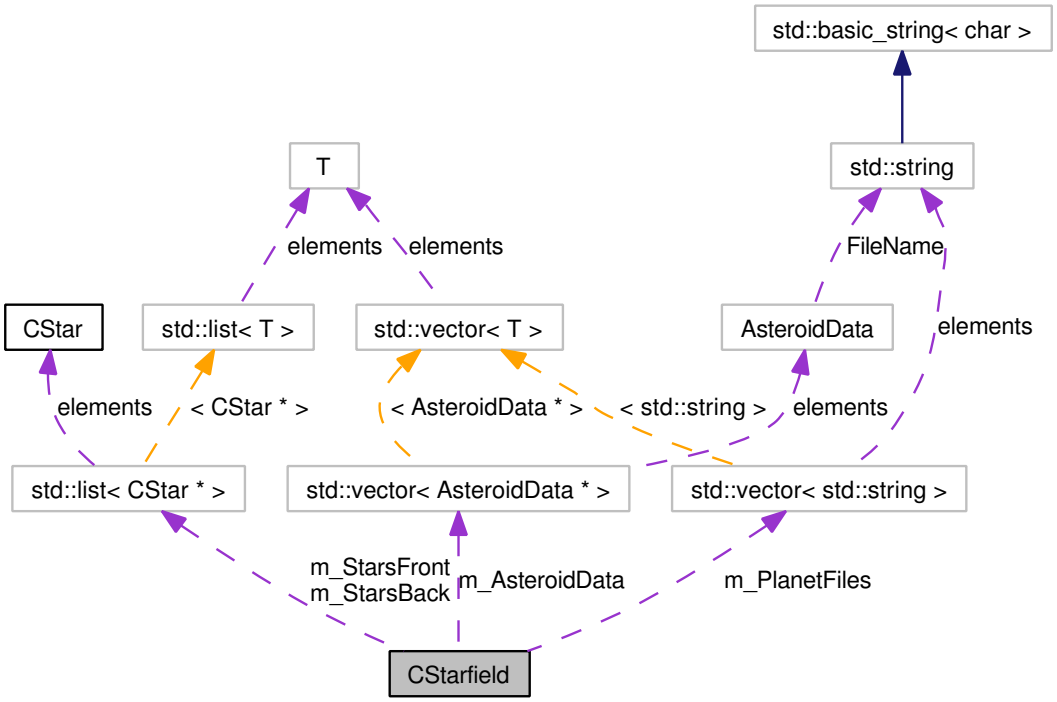
- Star.hpp
- Star.cpp

4.26 CStarfield Klassenreferenz

Sternenfeld.

`#include <Starfield.hpp>`

Zusammengehörigkeiten von CStarfield:



Öffentliche Methoden

- CStarfield ()

  Konstruktor und Destruktor.

- void Init (const std::string &FileName)

  Diese Funktion intialisiert das Sternenfeld.

- void DrawBack ()

  Diese Funktion zeichnet das hintere und aktualisiert das gesamte Sternenfeld.

- void DrawFront ()

  Diese Funktion zeichnet das vordere Sternenfeld.

- void RemovePlanets ()

  Diese Funktion entfernt alle Planeten.

- void SetSpawnPlanets (bool Spawning)

  Diese Funktion deaktiviert Planetenspawning.

- bool IsSpawningPlanets () const

  Diese Funktion gibt zurück Planeten gespawnt werden.

- void SetWidth (int Width)

  Diese Funktion legt die Breite des Sternenfeldes fest.

4.26.1 Ausführliche Beschreibung

Sternenfeld.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 52 der Datei Starfield.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

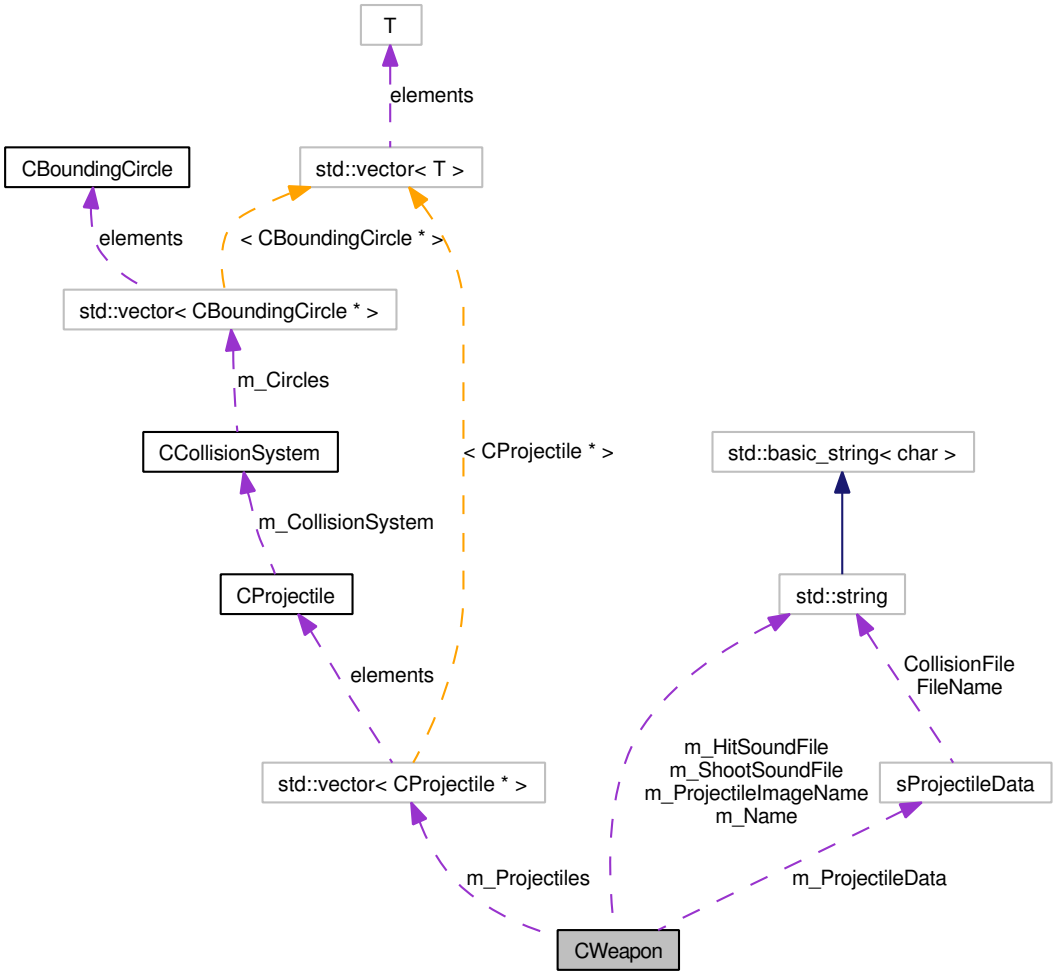
- Starfield.hpp
- Starfield.cpp

4.27 CWeapon Klassenreferenz

Waffe.

`#include <Weapon.hpp>`

Zusammengehörigkeiten von CWeapon:



Öffentliche Methoden

- CWeapon ()

  Konstrukor und Destruktor.

- void Init (const std::string &Name)

  Diese Funktion intialisiert die Waffe.

- const std::string & GetName () const

  Diese Funktion gibt den Namen der Waffe zurück.

- float GetProjectileSpeedX () const

Diese Funktion gibt die Geschwindigkeit der Projektile zurück.

- float GetProjectileSpeedY () const

  Diese Funktion gibt die Geschwindigkeit der Projektile zurück.

- int GetShootSpeed () const

  Diese Funktion gibt die Schussgeschwindigkeit der Waffe zurück.

- int GetDamage () const

  Diese Funktion gibt den Schaden der Waffe zurück.

- int GetGroupID () const

  Diese Funktion gibt die WaffenGruppenID zurück.

- void Equip (bool Equip=true)

  Diese Funktion rüstet die Waffe aus.

- bool IsEquiped () const

  Diese Funktion gibt zurück ob die Waffe ausgerüstet ist.

- void Reset ()

  Diese Funktion setzt die Waffe auf standard zurück.

- void Shoot (float X, float Y)

  Diese Funktion läßt die Waffe Schießen.

- int GetProjectilCount () const

  Diese Funktion gibt die ProjektilAnzahl der Waffe zurück.

- float GetProjectilX (int Projectil) const

  Diese Funktion gibt die X-Koordinate des angebne Projektils zurück.

- float GetProjectilY (int Projectil) const

  Diese Funktion gibt die Y-Koordinate des angebne Projektils zurück.

- float GetProjectilWidth (int Projectil) const

  Diese Funktion gibt die Breite des KollisionSystem des angebne Projektils zurück.

- void DrawProjectiles ()

  Diese Funktion zeichnet alle Schüße.

- int CheckCollision (const CCollisionSystem *CounterPart)
- void ClearProjectils ()

  Diese Funktion gibt alle Projektile der Waffe frei.

- int GetProjectilXRel (int YDistance) const

  Diese Funktion gibt die X-Projektile an der angeben Y-Koordinate, relativ zum Schiff.

- bool IsForeground () const

  Diese Funktion gibt zurück ob die Waffe im Vordergrund ist oder nicht.

4.27.1 Ausführliche Beschreibung

Waffe.

Autor:

Steffen Nörtershäuser

Definiert in Zeile 41 der Datei Weapon.hpp.

4.27.2 Dokumentation der Elementfunktionen

4.27.2.1 int CWeapon::CheckCollision (const CCollisionSystem * *CounterPart*)

Diese Funktion überprüft ob die Projektile mit dem angeben KollisionSystem kollidieren Der dabei insgesamt angerichtie Schaden wird zurückgegebb

Definiert in Zeile 315 der Datei Weapon.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

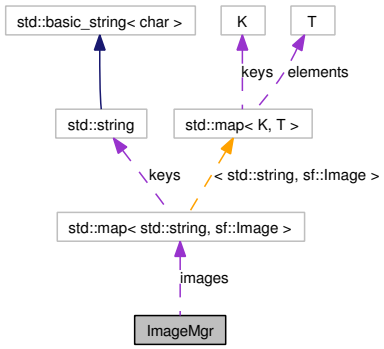
- Weapon.hpp
- Weapon.cpp

4.28 ImageMgr Klassenreferenz

Image Loader.

`#include <Images.hpp>`

Zusammengehörigkeiten von ImageMgr:



Öffentliche Methoden

- const sf::Image & **get_img** (const std::string &name)

4.28.1 Ausführliche Beschreibung

Image Loader.

Autor:

Christoph Egger

Definiert in Zeile 29 der Datei Images.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

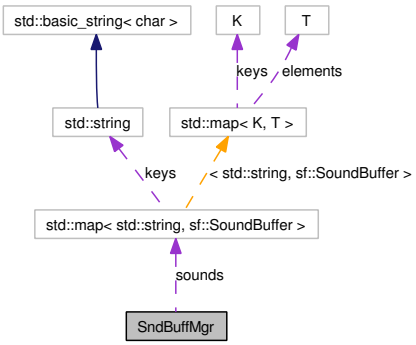
- Images.hpp

4.29 SndBuffMgr Klassenreferenz

Sound Loader.

`#include <SoundBuffers.hpp>`

Zusammengehörigkeiten von SndBuffMgr:



Öffentliche Methoden

- sf::SoundBuffer & **get_snd** (const std::string &name)

4.29.1 Ausführliche Beschreibung

Sound Loader.

Autor:

Christoph Egger

Definiert in Zeile 29 der Datei SoundBuffers.hpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- SoundBuffers.hpp

Index