PingForce TD

Készítette Doxygen 1.9.4

1. fejezet

Névtérmutató

1.1. Névtérlista

Az összes névtér listája rövid leírásokkal:

2. fejezet

Hierarchikus mutató

2.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

3. fejezet

Adatszerkezet-mutató

3.1. Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

4. fejezet

Fájlmutató

4.1. Fájllista

Az összes fájl listája rövid leírásokkal:

5. fejezet

Névterek dokumentációja

5.1. pftd névtér-referencia

Névterek

- namespace gr
- namespace utils

Adatszerkezetek

- class App
- class Clickable
- class Cub
- class Entity
- struct FollowPath
- class FZC
- class GameScene
- class IcicleStabber
- class Level
- struct LoadError
- class MenuScene
- class Object
- class Projectile
- class RegularSeal
- class Renderer
- class ResourceManager
- class Scene
- struct SceneError
- class Seal
- class Snowball
- class Snowballer
- class Tower
- class ZombieSeal

Típusdefiníciók

- using EndPoint = utils::Vec2f

Enumerációk

- enum class ProjectileID { SNOWBALL = 0 }
- enum class SealID { REGULAR = 0 , CUB , ZOMBIE , FZC }
- enum class TowerID { SNOWBALLER = 0 , ICICLE_STABBER }

5.1.1. Típusdefiníciók dokumentációja

5.1.1.1. EndPoint

```
using pftd::EndPoint = typedef utils::Vec2f
```

5.1.2. Enumerációk dokumentációja

5.1.2.1. ProjectileID

```
enum class pftd::ProjectileID  [strong]
```

Enumeráció-értékek

SNOWBALL	

5.1.2.2. SealID

```
enum class pftd::SealID  [strong]
```

Enumeráció-értékek

REGULAR	
CUB	
ZOMBIE	
FZC	

5.1.2.3. TowerID

```
enum class pftd::TowerID  [strong]
```

Enumeráció-értékek

SNOWBALLER	
ICICLE_STABBER	

5.2. pftd::gr névtér-referencia

Adatszerkezetek

- class Button
- class Label
- class Sprite

5.3. pftd::utils névtér-referencia

Névterek

- namespace parser

Adatszerkezetek

- class Container
- class Random
- class Serializable
- struct Vec2

Típusdefiníciók

- using Vec2i = Vec2< int >
- using Vec2f = Vec2< float >

5.3.1. Típusdefiníciók dokumentációja

5.3.1.1. Vec2f

```
using pftd::utils::Vec2f = typedef Vec2<float>
```

5.3.1.2. Vec2i

```
using pftd::utils::Vec2i = typedef Vec2<int>
```

5.4. pftd::utils::parser névtér-referencia

Adatszerkezetek

- class LevelConfigParser
- struct ParseError
- class Parser
- class SaveFileParser

6. fejezet

Adatszerkezetek dokumentációja

6.1. pftd::App osztályreferencia

`#include <app.hpp>`

Publikus tagfüggvények

- App (App const &)=delete
- App (App &&)=delete
- ~App ()

  Felszabadítja a kezelt erőforrásokat.
- void run ()

  A fő programciklus.
- void addScene (std::string id, Scene *scene, bool active=false)

  Hozzáad egy új nézetet.
- bool changeScene (std::string id, Scene::StateFlag flag=Scene::StateFlag::NONE)

  Nézetet vált (aktívvá tesz egy másikat).
- auto isRunning () const

  `m_running` *getter.*
- int getWindowWidth () const
- int getWindowHeight () const
- App & operator= (App const &)=delete

Statikus publikus tagfüggvények

- static App * getInstance ()

  A singleton instance megszerzése.
- static App * create (unsigned int width, unsigned int height, std::string const &windowTitle)

  Inicializálja a különböző programelemeket.
- static void destroy ()

  Törli a singleton instance-t.

Privát tagfüggvények

- App ()=default

Privát attribútumok

- bool m_running = false
- Renderer * m_renderer = nullptr
- std::map< std::string, Scene * > m_scenes
- std::string m_activeSceneID

Statikus privát attribútumok

- static App * m_instance

6.1.1. Részletes leírás

Az egész játékért felelő osztály.

6.1.2. Konstruktorok és destruktorok dokumentációja

6.1.2.1. App() [1/3]

```
pftd::App::App (
            App const &  )  [delete]
```

6.1.2.2. App() [2/3]

```
pftd::App::App (
            App &&  )  [delete]
```

6.1.2.3. ~App()

```
pftd::App::~App ( )
```

Felszabadítja a kezelt erőforrásokat.

Törli a renderer-t, az erőforrás-kezelőt és a nézeteket.

6.1.2.4. App() [3/3]

```
pftd::App::App ( )  [private], [default]
```

6.1.3. Tagfüggvények dokumentációja

6.1.3.1. addScene()

```
void pftd::App::addScene (
            std::string id,
            Scene * scene,
            bool active = false )
```

Hozzáad egy új nézetet.

Ha az `active` igaz, akkor meghívja a `changeScene`-t a megadott `id`-ra. Megjegyzés: Ha már létezik ilyen nézet a megadott azonosítóval, akkor `SceneError`-t dob.

Paraméterek

id	A nézet azonosítója.
scene	Maga a nézet osztály egy példánya (pl.: MenuScene, GameScene).
active	Ez legyen e az aktív nézet.

6.1.3.2. changeScene()

```
bool pftd::App::changeScene (
            std::string id,
            Scene::StateFlag flag = Scene::StateFlag::NONE )
```

Nézetet vált (aktívvá tesz egy másikat).

Ha megegyezik az eddigi és a kért nézet, akkor nem csinál semmit. Egyébként meg az eddigi nézetet deaktiválja, és az újat pedig aktiválja. Megjegyzés: Ha már létezik ilyen nézet a megadott azonosítóval, akkor `SceneError`-t dob.

Paraméterek

id	Az új nézet azonosítója.
flag	Egyéb "kérést" közlő opció flag.

Visszatérési érték

Sikerült e váltani.

6.1.3.3. create()

```
static App * pftd::App::create (
            unsigned int width,
            unsigned int height,
            std::string const & windowTitle )  [static]
```

Inicializálja a különböző programelemeket.

Megjegyzés: Meg kell hívni a `destroy()`-t, ha már nincs szükség rá!

Paraméterek

width	Az ablak szélessége.
height	Az ablak magassága.
windowTitle	Az ablak címe.

Visszatérési érték

Az osztályhoz tartozó singleton instance.

6.1.3.4. destroy()

```
static void pftd::App::destroy ( )  [inline], [static]
```

Törli a singleton instance-t.

6.1.3.5. getInstance()

```
static App * pftd::App::getInstance ( )  [inline], [static]
```

A singleton instance megszerzése.

Visszatérési érték

Az osztályhoz tartozó singleton instance.

6.1.3.6. getWindowHeight()

```
int pftd::App::getWindowHeight ( ) const  [inline]
```

Visszatérési érték

A programhoz tartozó ablak magassága.

6.1.3.7. getWindowWidth()

```
int pftd::App::getWindowWidth ( ) const  [inline]
```

Visszatérési érték

A programhoz tartozó ablak szélessége.

6.1.3.8. isRunning()

```
auto pftd::App::isRunning ( ) const  [inline]
```

`m_running` getter.

Visszatérési érték

Fut e a program.

6.1.3.9. operator=()

```
App & pftd::App::operator= (
            App const &  )  [delete]
```

6.1.3.10. run()

```
void pftd::App::run ( )
```

A fő programciklus.

6.1.4. Adatmezők dokumentációja

6.1.4.1. m_activeSceneID

```
std::string pftd::App::m_activeSceneID  [private]
```

Az aktív nézet azonosítója.

6.1.4.2. m_instance

```
App* pftd::App::m_instance  [static], [private]
```

A singleton instance.

6.1.4.3. m_renderer

```
Renderer* pftd::App::m_renderer = nullptr  [private]
```

A renderer.

6.1.4.4. m_running

```
bool pftd::App::m_running = false  [private]
```

Fut e a program.

6.1.4.5. m_scenes

```
std::map<std::string, Scene*> pftd::App::m_scenes  [private]
```

A nézetek: [id, nézet].

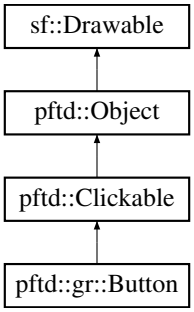
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/app.hpp

6.2. pftd::gr::Button osztályreferencia

```
#include <button.hpp>
```

A pftd::gr::Button osztály származási diagramja:



Publikus tagfüggvények

- Button (gr::Label const &label, utils::Vec2f const &position, utils::Vec2f const &size, bool active=true, int zIndex=0)
- virtual ~Button ()
- void setSound (std::string const &src)

  Kattintás hangeffekt beállítása.
- void setBackground (std::string const &src)

  Háttérkép beállítása.
- std::string getLabel () const

  Ráírt szöveg.
- virtual void handleClick (utils::Vec2i const &clickCoords) override

  Kattintás kezelése.
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Védett attribútumok

- Label label

Privát attribútumok

- sf::FloatRect m_rect
- Sprite * m_background = nullptr
- sf::Sound m_clickSound

További örökölt tagok

6.2.1. Részletes leírás

Gomb GUI elem.

6.2.2. Konstruktorok és destruktorok dokumentációja

6.2.2.1. Button()

```
pftd::gr::Button::Button (
            gr::Label const & label,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            bool active = true,
            int zIndex = 0 )
```

Paraméterek

label	Szöveg objektum: maga a szöveg, betűtípus és betűméret.
position	Pozíció.
size	Méret.
active	Aktív e? Rá lehet e kattintani.
zIndex	Z koordináta.

6.2.2.2. ~Button()

```
virtual pftd::gr::Button::~Button ( )  [virtual]
```

6.2.3. Tagfüggvények dokumentációja

6.2.3.1. draw()

```
virtual void pftd::gr::Button::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.2.3.2. getLabel()

```
std::string pftd::gr::Button::getLabel ( ) const  [inline]
```

Ráírt szöveg.

Visszatérési érték

A tárolt címkére írt szöveg.

6.2.3.3. handleClick()

```
virtual void pftd::gr::Button::handleClick (
            utils::Vec2i const & clickCoords )  [override], [virtual]
```

Kattintás kezelése.

Paraméterek

clickCoords	Kurzor koordinátái kattintáskor.

Újraimplementált ősök: pftd::Clickable.

6.2.3.4. setBackground()

```
void pftd::gr::Button::setBackground (
            std::string const & src )
```

Háttérkép beállítása.

Paraméterek

src	Háttárkép elérési útvonala.

6.2.3.5. setSound()

```
void pftd::gr::Button::setSound (
            std::string const & src )
```

Kattintás hangeffekt beállítása.

Paraméterek

src	Hangfájl elérési útvonala.

6.2.4. Adatmezők dokumentációja

6.2.4.1. label

```
Label pftd::gr::Button::label  [protected]
```

Címke.

6.2.4.2. m_background

```
Sprite* pftd::gr::Button::m_background = nullptr  [private]
```

Háttér.

6.2.4.3. m_clickSound

```
sf::Sound pftd::gr::Button::m_clickSound  [private]
```

Kattintás hangeffekt.

6.2.4.4. m_rect

```
sf::FloatRect pftd::gr::Button::m_rect  [private]
```

Minden gomb egy téglalap: ennek a pozíciója és mérete.

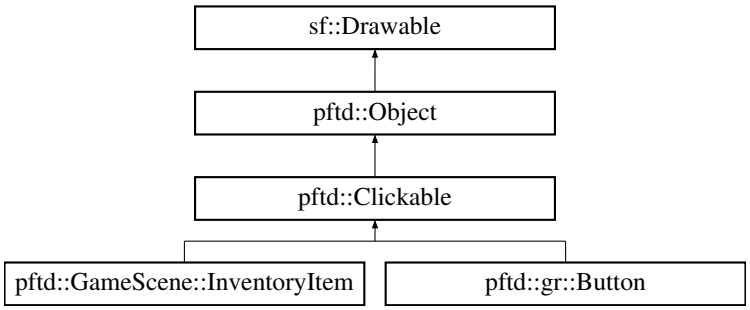
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/gui/button.hpp

6.3. pftd::Clickable osztályreferencia

```
#include <clickable.hpp>
```

A pftd::Clickable osztály származási diagramja:



Publikus tagfüggvények

- Clickable (bool active=true)
- Clickable (utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0, bool active=true)
- virtual void setCallback (std::function< void()> callback)

  Callback beállítása.
- virtual void handleClick (utils::Vec2i const &clickCoords)

  Kattintás kezelése.

Adatmezők

- bool isActive

Védett attribútumok

- std::function< void()> m_callback

6.3.1. Részletes leírás

Kattintható (interaktálható) objektum.

6.3.2. Konstruktorok és destruktorok dokumentációja

6.3.2.1. Clickable() [1/2]

```
pftd::Clickable::Clickable (
            bool active = true )
```

6.3.2.2. Clickable() [2/2]

```
pftd::Clickable::Clickable (
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0,
            bool active = true )
```

6.3.3. Tagfüggvények dokumentációja

6.3.3.1. handleClick()

```
virtual void pftd::Clickable::handleClick (
            utils::Vec2i const & clickCoords )  [virtual]
```

Kattintás kezelése.

Paraméterek

clickCoords	Kurzor koordinátái kattintáskor.

Újraimplementáló leszármazottak: pftd::gr::Button.

6.3.3.2. setCallback()

```
virtual void pftd::Clickable::setCallback (
            std::function< void()> callback )  [virtual]
```

Callback beállítása.

Paraméterek

callback	A kattintásra lefuttatni kívánt függvény.

6.3.4. Adatmezők dokumentációja

6.3.4.1. isActive

```
bool pftd::Clickable::isActive
```

Aktív e.

6.3.4.2. m_callback

```
std::function<void()> pftd::Clickable::m_callback  [protected]
```

Callback: akkor fut le, ha az objektumra rákattintunk.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/clickable.hpp

6.4. pftd::Object::Compare struktúrareferencia

```
#include <object_base.hpp>
```

Publikus tagfüggvények

- bool operator() (Object const *o1, Object const *o2) const

6.4.1. Részletes leírás

Priority queue miatt szükséges segédosztály.

6.4.2. Tagfüggvények dokumentációja

6.4.2.1. operator()()

```
bool pftd::Object::Compare::operator() (
            Object const * o1,
            Object const * o2 ) const  [inline]
```

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/objects/object_base.hpp

6.5. pftd::utils::Container< T, C > osztálysablon-referencia

```
#include <hetero_collection.hpp>
```

Publikus tagfüggvények

- Container ()=default
- virtual ~Container ()
- T *const append (T *elem)

  Új elem hozzáadása.
- std::size_t size () const

  A tároló mérete.
- C const & getContainer () const

  Konstans `m_container` getter.

Privát attribútumok

- C m_container

6.5.1. Részletes leírás

template<typename T, typename C = std::vector<T*>>
class pftd::utils::Container< T, C >

Olyan osztályok, amelyek tárolóként (is) használhatók. Fontos: nem egy tárolót tartalmaznak, hanem azok maguk a tárolók (öröklés, nem kompozíció)! Ilyen például: `FollowPath`, `Inventory`.

Megjegyzés: iterátorokat nem tartalmaz, de elérhető a `getContainer()`.

6.5.2. Konstruktorok és destruktorok dokumentációja

6.5.2.1. Container()

```
template<typename T , typename C = std::vector<T*>>
pftd::utils::Container< T, C >::Container ( )  [default]
```

6.5.2.2. ~Container()

```
template<typename T , typename C = std::vector<T*>>
virtual pftd::utils::Container< T, C >::~Container ( )  [inline], [virtual]
```

6.5.3. Tagfüggvények dokumentációja

6.5.3.1. append()

```
template<typename T , typename C = std::vector<T*>>
T *const pftd::utils::Container< T, C >::append (
            T * elem )  [inline]
```

Új elem hozzáadása.

Paraméterek

elem	A hozzáadni kívánt dolog.

Visszatérési érték

Az imént hozzáadott dolog.

6.5.3.2. getContainer()

```
template<typename T , typename C = std::vector<T*>>
C const & pftd::utils::Container< T, C >::getContainer ( ) const  [inline]
```

Konstans `m_container` getter.

Visszatérési érték

A tároló.

6.5.3.3. size()

```
template<typename T , typename C = std::vector<T*>>
std::size_t pftd::utils::Container< T, C >::size ( ) const  [inline]
```

A tároló mérete.

Visszatérési érték

A tároló elemeinek száma.

6.5.4. Adatmezők dokumentációja

6.5.4.1. m_container

```
template<typename T , typename C = std::vector<T*>>
C pftd::utils::Container< T, C >::m_container  [private]
```

A tároló.

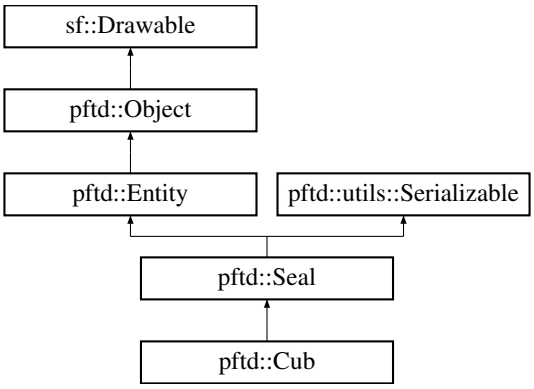
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/hetero_collection.hpp

6.6. pftd::Cub osztályreferencia

```
#include <cub.hpp>
```

A pftd::Cub osztály származási diagramja:



Publikus tagfüggvények

- Cub (FollowPath const &followpath)

  A textúra hard code-olt, a kezdő pozíciót pedig a `followPath` határozza meg.
- Seal * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

További örökölt tagok

6.6.1. Részletes leírás

Kicsiny.

6.6.2. Konstruktorok és destruktorok dokumentációja

6.6.2.1. Cub()

```
pftd::Cub::Cub (
            FollowPath const & followpath )
```

A textúra hard code-olt, a kezdő pozíciót pedig a `followPath` határozza meg.

Paraméterek

followpath	Az útvonal amin végig kell menjen (majd pedig vissza).

6.6.3. Tagfüggvények dokumentációja

6.6.3.1. clone()

```
Seal * pftd::Cub::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Seal.

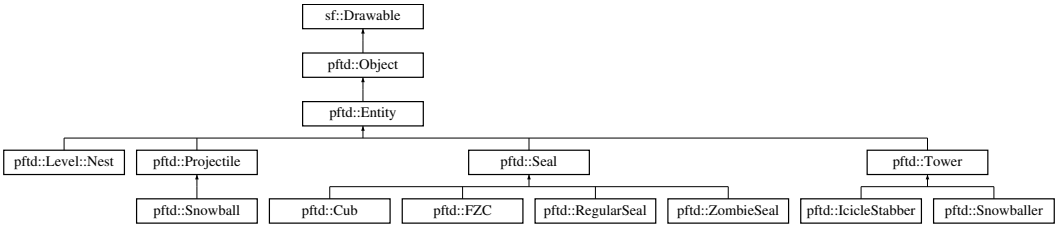
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/seals/cub.hpp

6.7. pftd::Entity osztályreferencia

`#include <entity_base.hpp>`

A pftd::Entity osztály származási diagramja:



Publikus tagfüggvények

- Entity (std::string const &spriteSheetSrc, utils::Vec2i spriteSize, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0)
- Entity (sf::Texture const &texture, utils::Vec2i spriteSize, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0)
- Entity (std::string const &spriteSrc, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0)
- Entity (Entity const &other)
- virtual ~Entity ()
- virtual Entity * clone () const =0

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.
- virtual void update (float dt)

  Update.
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.
- void setPosition (utils::Vec2f position)

  Pozíció beállítása.
- auto getPosition () const

  `position` *getter.*
- gr::Sprite const * getSprite () const

  Konstans `currentSprite` *getter.*
- gr::Sprite * getSprite ()

  `currentSprite` *getter.*
- auto & getSpriteSheet () const

  `spriteSheet` *getter.*

Adatmezők

- bool isAnimated = true

Védett tagfüggvények

- virtual void advanceAnimationFrame ()

  Animáció: képkocka léptetése.
- virtual void resetAnimation ()

  Visszaállítja a jelenlegi képkockát az elsőre, és nullázza az időt.

Védett attribútumok

- sf::Texture const & spriteSheet
- utils::Vec2i cellSize
- size_t const CELL_N
- size_t currentCell = 0U
- gr::Sprite * currentSprite = nullptr
- float frameDurationSec = 1.0f
- float totalElapsedSec = 0.0f

6.7.1. Részletes leírás

Mindenféle "actor" ősosztálya: tornyok, fólák és lövedékek.

6.7.2. Konstruktorok és destruktorok dokumentációja

6.7.2.1. Entity() [1/4]

```
pftd::Entity::Entity (
            std::string const & spriteSheetSrc,
            utils::Vec2i spriteSize,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0 )
```

6.7.2.2. Entity() [2/4]

```
pftd::Entity::Entity (
            sf::Texture const & texture,
            utils::Vec2i spriteSize,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0 )
```

6.7.2.3. Entity() [3/4]

```
pftd::Entity::Entity (
            std::string const & spriteSrc,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0 )
```

6.7.2.4. Entity() [4/4]

```
pftd::Entity::Entity (
            Entity const & other )
```

6.7.2.5. ~Entity()

```
virtual pftd::Entity::~Entity ( )  [virtual]
```

6.7.3. Tagfüggvények dokumentációja

6.7.3.1. advanceAnimationFrame()

```
virtual void pftd::Entity::advanceAnimationFrame ( )  [protected], [virtual]
```

Animáció: képkocka léptetése.

Újraimplementáló leszármazottak: pftd::Seal és pftd::Tower.

6.7.3.2. clone()

```
virtual Entity * pftd::Entity::clone ( ) const  [pure virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítják a következők: pftd::Level::Nest, pftd::Snowball, pftd::Cub, pftd::FZC, pftd::RegularSeal, pftd::ZombieSeal, pftd::IcicleStabber, pftd::Snowballer, pftd::Seal és pftd::Tower.

6.7.3.3. draw()

```
virtual void pftd::Entity::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.7.3.4. getPosition()

```
auto pftd::Entity::getPosition ( ) const  [inline]
```

`position` getter.

Visszatérési érték

Az entitás jelenlegi pozíciója.

6.7.3.5. getSprite() [1/2]

```
gr::Sprite * pftd::Entity::getSprite ( )  [inline]
```

`currentSprite` getter.

Visszatérési érték

A jelenlegi sprite.

6.7.3.6. getSprite() [2/2]

```
gr::Sprite const * pftd::Entity::getSprite ( ) const  [inline]
```

Konstans `currentSprite` getter.

Visszatérési érték

A jelenlegi sprite.

6.7.3.7. getSpriteSheet()

```
auto & pftd::Entity::getSpriteSheet ( ) const  [inline]
```

`spriteSheet` getter.

Visszatérési érték

A sprite sheet.

6.7.3.8. resetAnimation()

```
virtual void pftd::Entity::resetAnimation ( )  [protected], [virtual]
```

Visszaállítja a jelenlegi képkockát az elsőre, és nullázza az időt.

6.7.3.9. setPosition()

```
void pftd::Entity::setPosition (
            utils::Vec2f position )
```

Pozíció beállítása.

Paraméterek

position	Az új pozíció.

6.7.3.10. update()

```
virtual void pftd::Entity::update (
            float dt )  [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementáló leszármazottak: pftd::Projectile, pftd::Seal, pftd::IcicleStabber, pftd::Snowballer és pftd::Tower.

6.7.4. Adatmezők dokumentációja

6.7.4.1. CELL_N

```
size_t const pftd::Entity::CELL_N  [protected]
```

Képkockák száma.

6.7.4.2. cellSize

```
utils::Vec2i pftd::Entity::cellSize  [protected]
```

Képkockák mérete (fix).

6.7.4.3. currentCell

```
size_t pftd::Entity::currentCell = 0U  [protected]
```

A sprite sheet aktív képkockája.

6.7.4.4. currentSprite

```
gr::Sprite* pftd::Entity::currentSprite = nullptr  [protected]
```

A jelenlegi képkocka által meghatározott sprite.

6.7.4.5. frameDurationSec

```
float pftd::Entity::frameDurationSec = 1.0f  [protected]
```

Ennyi ideig "tart" egy képkocka, ezután váltani kell.

6.7.4.6. isAnimated

```
bool pftd::Entity::isAnimated = true
```

Animált e?

6.7.4.7. spriteSheet

```
sf::Texture const& pftd::Entity::spriteSheet  [protected]
```

A sprite sheet.

6.7.4.8. totalElapsedSec

```
float pftd::Entity::totalElapsedSec = 0.0f  [protected]
```

Eddig eltelt idő.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/entity_base.hpp

6.8. pftd::utils::parser::SaveFileParser::EntityInfo struktúrareferencia

```
#include <parsers.hpp>
```

Adatmezők

- EntityType entityType
- utils::Vec2f position
- struct {
    unsigned int sealID
    float lerpParam
    bool goingBackwards
    unsigned int hp
  } seal

- struct {
    unsigned int projID
    utils::Vec2f direction
    float speed
  } proj

- unsigned int towerID

6.8.1. Részletes leírás

Mentett játék betöltéséhez használt entitás infó.

6.8.2. Adatmezők dokumentációja

6.8.2.1. direction

```
utils::Vec2f pftd::utils::parser::SaveFileParser::EntityInfo::direction
```

6.8.2.2. entityType

```
EntityType pftd::utils::parser::SaveFileParser::EntityInfo::entityType
```

Típusa.

6.8.2.3. goingBackwards

```
bool pftd::utils::parser::SaveFileParser::EntityInfo::goingBackwards
```

6.8.2.4. hp

```
unsigned int pftd::utils::parser::SaveFileParser::EntityInfo::hp
```

6.8.2.5. lerpParam

```
float pftd::utils::parser::SaveFileParser::EntityInfo::lerpParam
```

6.8.2.6. position

```
utils::Vec2f pftd::utils::parser::SaveFileParser::EntityInfo::position
```

Pozíciója.

6.8.2.7.

```
struct { ... } pftd::utils::parser::SaveFileParser::EntityInfo::proj
```

Lövedék típusra valló adatok.

6.8.2.8. projID

```
unsigned int pftd::utils::parser::SaveFileParser::EntityInfo::projID
```

6.8.2.9.

```
struct { ... } pftd::utils::parser::SaveFileParser::EntityInfo::seal
```

Típustól függő egyéb adat.

Ellenfél típusra valló adatok.

6.8.2.10. sealID

```
unsigned int pftd::utils::parser::SaveFileParser::EntityInfo::sealID
```

6.8.2.11. speed

```
float pftd::utils::parser::SaveFileParser::EntityInfo::speed
```

6.8.2.12. towerID

```
unsigned int pftd::utils::parser::SaveFileParser::EntityInfo::towerID
```

Torony típusra valló adatok.

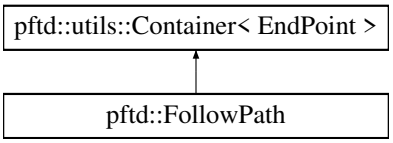
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.9. pftd::FollowPath struktúrareferencia

```
#include <followpath.hpp>
```

A pftd::FollowPath osztály származási diagramja:



Publikus tagfüggvények

- FollowPath ()=default

  A `Level` - amihez tartozik - "építi fel.".

6.9.1. Részletes leírás

Az ellenfelek által követett út.

6.9.2. Konstruktorok és destruktorok dokumentációja

6.9.2.1. FollowPath()

```
pftd::FollowPath::FollowPath ( )  [default]
```

A Level - amihez tartozik - "építi fel.".

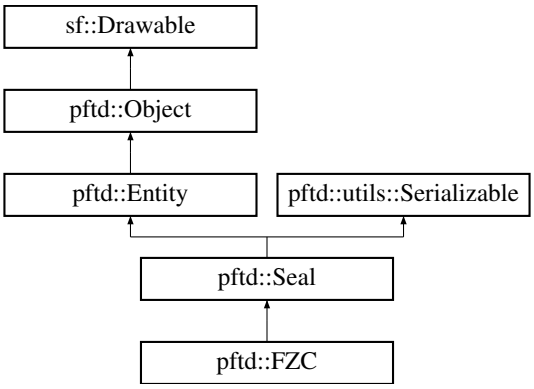
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/objects/entities/seals/followpath.hpp

6.10. pftd::FZC osztályreferencia

```
#include <fortified_zombie_cub.hpp>
```

A pftd::FZC osztály származási diagramja:



Publikus tagfüggvények

- FZC (FollowPath const &followpath)
- Seal * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

További örökölt tagok

6.10.1. Részletes leírás

Páncélos zombi kicsiny.

6.10.2. Konstruktorok és destruktorok dokumentációja

6.10.2.1. FZC()

```
pftd::FZC::FZC (
            FollowPath const & followpath )
```

Paraméterek

followpath	Az útvonal amin végig kell menjen (majd pedig vissza).

6.10.3. Tagfüggvények dokumentációja

6.10.3.1. clone()

```
Seal * pftd::FZC::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Seal.

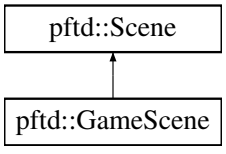
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/seals/fortified_zombie_cub.hpp

6.11. pftd::GameScene osztályreferencia

```
#include <game.hpp>
```

A pftd::GameScene osztály származási diagramja:



Adatszerkezetek

- struct Inventory
- struct InventoryItem

Publikus tagfüggvények

- GameScene ()
- ~GameScene ()
- void onEvent (sf::Event const &event) override

  Delegált események kezelése.
- void update (float dt) override

  Update.
- void toggleActive (Scene::StateFlag flag=Scene::StateFlag::NONE) override

  Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.
- void startGame ()

  Játék elindítása.
- void updateScore ()

  Pontszám felirat frissítése.
- void updateWealth ()

  Pénz felirat frissítése.

Statikus publikus attribútumok

- static constexpr char const * SAVE_FILE_PATH = "res/data/save.dat"

Privát tagfüggvények

- void _constructInventory ()

  Hozzáadja az `m_inventory`-hoz az `InventoryItem`-eket.

Privát attribútumok

- bool m_shouldLoadSaved = false
- gr::Button * m_saveButt = nullptr
- gr::Label * m_moneyCounter = nullptr
- gr::Label * m_scoreCounter = nullptr
- sf::Sound m_hornSound = sf::Sound{ResourceManager::getInstance()->getSound("res/audio/ready_for_↩ battle.mp3")}
- sf::Sound m_gameoverSound = sf::Sound{ResourceManager::getInstance()->getSound("res/audio/gameover.↩ mp3")}
- Inventory * m_inventory = nullptr
- Level * m_level = nullptr

További örökölt tagok

6.11.1. Részletes leírás

Játék nézet.

6.11.2. Konstruktorok és destruktorok dokumentációja

6.11.2.1. GameScene()

```
pftd::GameScene::GameScene ( )
```

6.11.2.2. ~GameScene()

```
pftd::GameScene::~GameScene ( )
```

6.11.3. Tagfüggvények dokumentációja

6.11.3.1. _constructInventory()

```
void pftd::GameScene::_constructInventory ( )  [private]
```

Hozzáadja az `m_inventory`-hoz az `InventoryItem`-eket.

6.11.3.2. onEvent()

```
void pftd::GameScene::onEvent (
            sf::Event const & event )  [override], [virtual]
```

Delegált események kezelése.

Paraméterek

event	Kezelni kívánt esemény.

Megvalósítja a következőket: pftd::Scene.

6.11.3.3. startGame()

```
void pftd::GameScene::startGame ( )
```

Játék elindítása.

Inicializálja a játék kezdetét: létrehozza a "szintet."

6.11.3.4. toggleActive()

```
void pftd::GameScene::toggleActive (
            Scene::StateFlag flag = Scene::StateFlag::NONE )  [override], [virtual]
```

Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.

Paraméterek

flag	Egyéb "kérést" közlő opció flag.

Újraimplementált ősök: pftd::Scene.

6.11.3.5. update()

```
void pftd::GameScene::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Megvalósítja a következőket: pftd::Scene.

6.11.3.6. updateScore()

```
void pftd::GameScene::updateScore ( )
```

Pontszám felirat frissítése.

6.11.3.7. updateWealth()

```
void pftd::GameScene::updateWealth ( )
```

Pénz felirat frissítése.

6.11.4. Adatmezők dokumentációja

6.11.4.1. m_gameoverSound

```
sf::Sound pftd::GameScene::m_gameoverSound = sf::Sound{ResourceManager::getInstance()->get↩
Sound("res/audio/gameover.mp3")}  [private]
```

Játék vége hangeffekt.

6.11.4.2. m_hornSound

```
sf::Sound pftd::GameScene::m_hornSound = sf::Sound{ResourceManager::getInstance()->getSound("res/audio/ready←
_for_battle.mp3")}  [private]
```

Kürt hang. Ez a játék elindításánál játszódik be.

6.11.4.3. m_inventory

```
Inventory* pftd::GameScene::m_inventory = nullptr  [private]
```

Az "eszköztár."

6.11.4.4. m_level

```
Level* pftd::GameScene::m_level = nullptr  [private]
```

Játéklogikát tartalmazó "szint."

6.11.4.5. m_moneyCounter

```
gr::Label* pftd::GameScene::m_moneyCounter = nullptr  [private]
```

Pénzszámláló felirat.

6.11.4.6. m_saveButt

```
gr::Button* pftd::GameScene::m_saveButt = nullptr  [private]
```

Mentés gomb.

6.11.4.7. m_scoreCounter

```
gr::Label* pftd::GameScene::m_scoreCounter = nullptr  [private]
```

Pontszámláló felirat.

6.11.4.8. m_shouldLoadSaved

```
bool pftd::GameScene::m_shouldLoadSaved = false  [private]
```

Be kell e tölteni a mentett játékállást.

6.11.4.9. SAVE_FILE_PATH

```
constexpr char const* pftd::GameScene::SAVE_FILE_PATH = "res/data/save.dat"  [static], [constexpr]
```

Mentett játékállás fájlja.

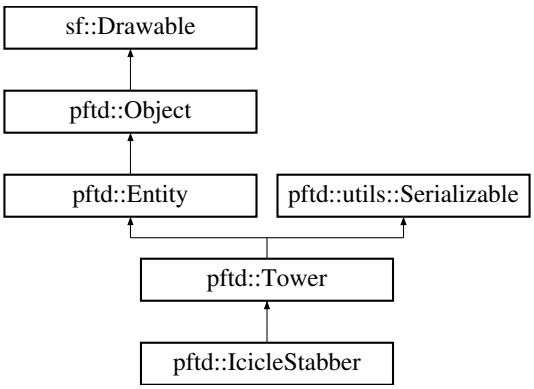
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/scenes/game.hpp

6.12. pftd::IcicleStabber osztályreferencia

```
#include <iciclestabber.hpp>
```

A pftd::IcicleStabber osztály származási diagramja:



Publikus tagfüggvények

- IcicleStabber (utils::Vec2f const &position, int zIndex=0)
- IcicleStabber ()
- ~IcicleStabber ()=default
- Tower * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.
- void update (float dt) override

  Update.

További örökölt tagok

6.12.1. Részletes leírás

Jégcsap kardos.

6.12.2. Konstruktorok és destruktorok dokumentációja

6.12.2.1. IcicleStabber() [1/2]

```
pftd::IcicleStabber::IcicleStabber (
            utils::Vec2f const & position,
            int zIndex = 0 )
```

6.12.2.2. IcicleStabber() [2/2]

```
pftd::IcicleStabber::IcicleStabber ( )
```

6.12.2.3. ~IcicleStabber()

```
pftd::IcicleStabber::~IcicleStabber ( )  [default]
```

6.12.3. Tagfüggvények dokumentációja

6.12.3.1. clone()

```
Tower * pftd::IcicleStabber::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Tower.

6.12.3.2. update()

```
void pftd::IcicleStabber::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementált ősök: pftd::Tower.

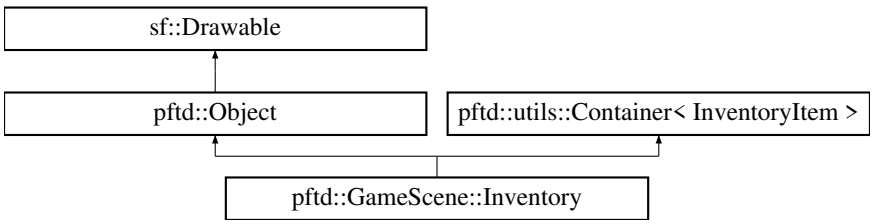
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/towers/iciclestabber.hpp

6.13. pftd::GameScene::Inventory struktúrareferencia

`#include <game.hpp>`

A pftd::GameScene::Inventory osztály származási diagramja:



Publikus tagfüggvények

- Inventory (std::string const &backgroundImageSrc)
- ~Inventory ()=default
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Adatmezők

- gr::Sprite background

További örökölt tagok

6.13.1. Részletes leírás

"Eszköztár": innen lehet megvenni a tornyokat.

6.13.2. Konstruktorok és destruktorok dokumentációja

6.13.2.1. Inventory()

```
pftd::GameScene::Inventory::Inventory (
            std::string const & backgroundImageSrc )
```

Paraméterek

backgroundImageSrc	Háttérkép elérési útvonala.

6.13.2.2. ~Inventory()

```
pftd::GameScene::Inventory::~Inventory ( )  [default]
```

6.13.3. Tagfüggvények dokumentációja

6.13.3.1. draw()

```
void pftd::GameScene::Inventory::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.13.4. Adatmezők dokumentációja

6.13.4.1. background

```
gr::Sprite pftd::GameScene::Inventory::background
```

Háttér.

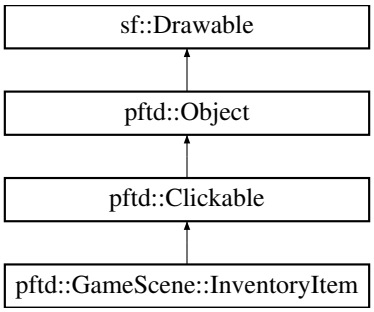
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/scenes/game.hpp

6.14. pftd::GameScene::InventoryItem struktúrareferencia

`#include <game.hpp>`

A pftd::GameScene::InventoryItem osztály származási diagramja:



Publikus tagfüggvények

- InventoryItem (Tower *tower, Level *const level, utils::Vec2f const &position, utils::Vec2f const &size)
- ~InventoryItem ()
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Adatmezők

- gr::Sprite frame
- gr::Sprite icon
- Tower * towerToSpawn = nullptr
- gr::Label priceLabel

További örökölt tagok

6.14.1. Részletes leírás

Az "eszköztárban" tárolt torony.

6.14.2. Konstruktorok és destruktorok dokumentációja

6.14.2.1. InventoryItem()

```
pftd::GameScene::InventoryItem::InventoryItem (
            Tower * tower,
            Level *const level,
            utils::Vec2f const & position,
            utils::Vec2f const & size )
```

6.14.2.2. ~InventoryItem()

```
pftd::GameScene::InventoryItem::~InventoryItem ( )
```

6.14.3. Tagfüggvények dokumentációja

6.14.3.1. draw()

```
virtual void pftd::GameScene::InventoryItem::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.14.4. Adatmezők dokumentációja

6.14.4.1. frame

```
gr::Sprite pftd::GameScene::InventoryItem::frame
```

Keret.

6.14.4.2. icon

```
gr::Sprite pftd::GameScene::InventoryItem::icon
```

Ikon.

6.14.4.3. priceLabel

```
gr::Label pftd::GameScene::InventoryItem::priceLabel
```

Árcímke (felirat).

6.14.4.4. towerToSpawn

```
Tower* pftd::GameScene::InventoryItem::towerToSpawn = nullptr
```

Torony amit meg szeretnénk venni.

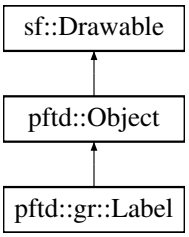
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/scenes/game.hpp

6.15. pftd::gr::Label osztályreferencia

```
#include <label.hpp>
```

A pftd::gr::Label osztály származási diagramja:



Publikus tagfüggvények

- Label (std::wstring const &label, sf::Font const &font, unsigned int cSize, sf::Color const &color=sf::Color::↩ White)
- Label (std::wstring const &label, sf::Font const &font, unsigned int cSize, utils::Vec2f position, int zIndex=0, sf::Color const &color=sf::Color::White)
- void setLabel (std::string const &newLabel)

  Új szöveg beállítása.
- void setOutline (sf::Color color, float thickness=1.0f)

  Szöveg outline beállítása.
- sf::Text & getText ()

  `m_text` *getter.*
- sf::Text const & getText () const

  Konstans `m_text` *getter.*
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Privát attribútumok

- sf::Text m_text

További örökölt tagok

6.15.1. Részletes leírás

Felirat GUI elem.

6.15.2. Konstruktorok és destruktorok dokumentációja

6.15.2.1. Label() [1/2]

```
pftd::gr::Label::Label (
            std::wstring const & label,
            sf::Font const & font,
            unsigned int cSize,
            sf::Color const & color = sf::Color::White )
```

Paraméterek

label	Szöveg.
font	Betűtípus.
cSize	Betűméret.
color	Betűszín.

6.15.2.2. Label() [2/2]

```
pftd::gr::Label::Label (
            std::wstring const & label,
            sf::Font const & font,
            unsigned int cSize,
            utils::Vec2f position,
            int zIndex = 0,
            sf::Color const & color = sf::Color::White )
```

Paraméterek

label	Szöveg.
font	Betűtípus.
cSize	Betűméret.
position	Pozíció.
zIndex	Z koordináta.
color	Betűszín.

6.15.3. Tagfüggvények dokumentációja

6.15.3.1. draw()

```
void pftd::gr::Label::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [inline], [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.15.3.2. getText() [1/2]

```
sf::Text & pftd::gr::Label::getText ( )  [inline]
```

`m_text` getter.

Visszatérési érték

A tárolt szöveg objektum.

6.15.3.3. getText() [2/2]

```
sf::Text const & pftd::gr::Label::getText ( ) const  [inline]
```

Konstans `m_text` getter.

Visszatérési érték

A tárolt szöveg objektum.

6.15.3.4. setLabel()

```
void pftd::gr::Label::setLabel (
            std::string const & newLabel )  [inline]
```

Új szöveg beállítása.

Paraméterek

newLabel	Az új szöveg.

6.15.3.5. setOutline()

```
void pftd::gr::Label::setOutline (
            sf::Color color,
            float thickness = 1.0f )  [inline]
```

Szöveg outline beállítása.

Paraméterek

color	Színe.
thickness	Vastagsága.

6.15.4. Adatmezők dokumentációja

6.15.4.1. m_text

```
sf::Text pftd::gr::Label::m_text  [private]
```

Tárolt szöveg objektum.

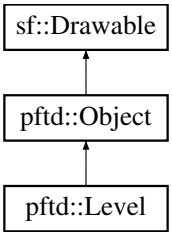
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/gui/label.hpp

6.16. pftd::Level osztályreferencia

```
#include <level.hpp>
```

A pftd::Level osztály származási diagramja:



Adatszerkezetek

- struct Nest
- struct Stats

Publikus tagfüggvények

- Level (std::string const &saveFile, Stats stats)
- Level (std::string const &saveFile)

  Ilyenkor a játékállást is a mentésfájlból olvassa vissza.
- virtual ~Level ()
- virtual void loseHP (int hpLost=1)

  HP vesztés.
- virtual bool placeTower ()

  Kiválasztott torny lehelyezése.
- void deselectTower ()

  Kiválasztott torony törlése.
- void selectTower (Tower *newTower)

  Torony kiválasztása.
- void update (float dt)

  Update.
- bool isGameOver () const

  Vége van e játéknak?
- void save () const

  Játékállás mentése.
- void reset (Stats stats)

  Játék újraindítása.
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Adatmezők

- Stats stats = Stats{}
- Tower * selectedTower = nullptr

Védett tagfüggvények

- void spawnSeal ()

  Ellenfél spawnolás.

Védett attribútumok

- Nest * nest = nullptr
- FollowPath followPath
- std::vector< Tower * > towers
- std::vector< Seal * > seals
- std::vector< Projectile * > projectiles
- utils::parser::LevelConfigParser config
- std::string saveFile

Privát tagfüggvények

- void _spawnSeal (float dt)

  spawnSeal-t hívja meg, ha eltelt megfelelő mennyiségű idő.
- void _updateSeals (float dt)

  Kezeli az ellenfeleket.
- void _updateTowers (float dt)

  Kezeli a tornyokat.
- void _updateProjectiles (float dt)

  Kezeli a lövedékeket.

Privát attribútumok

- float m_accuTimeSpawnSec = 0.0f

6.16.1. Részletes leírás

Játéklogikát és entitásokat tartalmazó osztály.

6.16.2. Konstruktorok és destruktorok dokumentációja

6.16.2.1. Level() [1/2]

```
pftd::Level::Level (
            std::string const & saveFile,
            Stats stats )
```

Paraméterek

saveFile	Mentésfájl elérési útvonala.
stats	Kezdeti játékállás.

6.16.2.2. Level() [2/2]

```
pftd::Level::Level (
            std::string const & saveFile )
```

Ilyenkor a játékállást is a mentésfájlból olvassa vissza.

Paraméterek

saveFile	Mentésfájl elérési útvonala.

6.16.2.3. ∼Level()

```
virtual pftd::Level::∼Level ( )  [virtual]
```

6.16.3. Tagfüggvények dokumentációja

6.16.3.1. _spawnSeal()

```
void pftd::Level::_spawnSeal (
            float dt )  [private]
```

spawnSeal-t hívja meg, ha eltelt megfelelő mennyiségű idő.

6.16.3.2. _updateProjectiles()

```
void pftd::Level::_updateProjectiles (
            float dt )  [private]
```

Kezeli a lövedékeket.

6.16.3.3. _updateSeals()

```
void pftd::Level::_updateSeals (
            float dt )  [private]
```

Kezeli az ellenfeleket.

6.16.3.4. _updateTowers()

```
void pftd::Level::_updateTowers (
            float dt )  [private]
```

Kezeli a tornyokat.

6.16.3.5. deselectTower()

```
void pftd::Level::deselectTower ( )
```

Kiválasztott torony törlése.

6.16.3.6. draw()

```
virtual void pftd::Level::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.16.3.7. isGameOver()

```
bool pftd::Level::isGameOver ( ) const  [inline]
```

Vége van e játéknak?

Visszatérési érték

Vesztett e a játékos?

6.16.3.8. loseHP()

```
virtual void pftd::Level::loseHP (
            int hpLost = 1 )  [virtual]
```

HP vesztés.

Paraméterek

hpLost	Ennyi HP-t veszít a játékos.

6.16.3.9. placeTower()

```
virtual bool pftd::Level::placeTower ( )  [virtual]
```

Kiválasztott torny lehelyezése.

Visszatérési érték

Sikerült e lehelyezni?

6.16.3.10. reset()

```
void pftd::Level::reset (
            Stats stats )
```

Játék újraindítása.

"Object Pool" nem használata miatt van.

6.16.3.11. save()

```
void pftd::Level::save ( ) const
```

Játékállás mentése.

LoadError-t dobhat, ha nem sikerül megnyitnia a fájlt / nincs engedélye hozzá.

6.16.3.12. selectTower()

```
void pftd::Level::selectTower (
            Tower * newTower )
```

Torony kiválasztása.

Ezt a játék nézet "eszköztárja" adja át.

Paraméterek

newTower	Lehelyezni kívánt torony.

6.16.3.13. spawnSeal()

```
void pftd::Level::spawnSeal ( )  [protected]
```

Ellenfél spawnolás.

6.16.3.14. update()

```
void pftd::Level::update (
            float dt )
```

Update.

Paraméterek

dt	Delta idő.

6.16.4. Adatmezők dokumentációja

6.16.4.1. config

```
utils::parser::LevelConfigParser pftd::Level::config  [protected]
```

Inicializáláshoz szükséges konfiguráció.

6.16.4.2. followPath

```
FollowPath pftd::Level::followPath  [protected]
```

Ellenségek által követett útvonal.

6.16.4.3. m_accuTimeSpawnSec

```
float pftd::Level::m_accuTimeSpawnSec = 0.0f  [private]
```

Előző spawn óta eltelt idő.

6.16.4.4. nest

```
Nest* pftd::Level::nest = nullptr  [protected]
```

A fészek (cél).

6.16.4.5. projectiles

```
std::vector<Projectile*> pftd::Level::projectiles  [protected]
```

Lövedékek.

6.16.4.6. saveFile

```
std::string pftd::Level::saveFile  [protected]
```

Mentésfájl.

6.16.4.7. seals

```
std::vector<Seal*> pftd::Level::seals  [protected]
```

Élő ellenfelek.

6.16.4.8. selectedTower

```
Tower* pftd::Level::selectedTower = nullptr
```

Lehelyezni kívánt torony.

6.16.4.9. stats

```
Stats pftd::Level::stats = Stats{}
```

Játékos statisztika.

6.16.4.10. towers

```
std::vector<Tower*> pftd::Level::towers  [protected]
```

Lehelyezett tornyok.

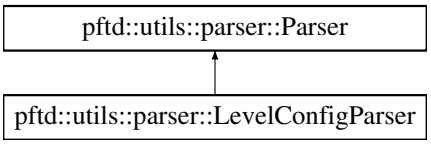
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/game/level.hpp

6.17. pftd::utils::parser::LevelConfigParser osztályreferencia

```
#include <parsers.hpp>
```

A pftd::utils::parser::LevelConfigParser osztály származási diagramja:



Publikus tagfüggvények

- LevelConfigParser (std::string const &sourceFile)
- ~LevelConfigParser ()=default
- void parse () override

  A parse főciklus.
- std::vector< Vec2f > getAttribute (std::string name) const

  Mentett attribútum értékének kiolvasása.

Privát tagfüggvények

- std::string _getAttribute ()

  Következő attribútum beolvasása a konfigfájlból.

Privát attribútumok

- std::unordered_map< std::string, std::vector< Vec2f > > m_attribs

További örökölt tagok

6.17.1. Részletes leírás

Egy szint alap beállításait tudjuk vele betölteni: az ellenfelek által követett útvonalat és a fészek helyzetét.

6.17.2. Konstruktorok és destruktorok dokumentációja

6.17.2.1. LevelConfigParser()

```
pftd::utils::parser::LevelConfigParser::LevelConfigParser (
            std::string const & sourceFile )
```

Paraméterek

sourceFile	A fájlnak az elérési útvonala, ahonnan a konfigot betöltjük.

6.17.2.2. ~LevelConfigParser()

```
pftd::utils::parser::LevelConfigParser::~LevelConfigParser ( )  [default]
```

6.17.3. Tagfüggvények dokumentációja

6.17.3.1. _getAttribute()

```
std::string pftd::utils::parser::LevelConfigParser::_getAttribute ( )  [private]
```

Következő attribútum beolvasása a konfigfájlból.

Visszatérési érték

A kapott attribútum neve.

6.17.3.2. getAttribute()

```
std::vector< Vec2f > pftd::utils::parser::LevelConfigParser::getAttribute (
            std::string name ) const
```

Mentett attribútum értékének kiolvasása.

Paraméterek

name	Az attribútum neve.

Visszatérési érték

Az attribútum értéke (mindig egy array).

6.17.3.3. parse()

```
void pftd::utils::parser::LevelConfigParser::parse ( )  [override], [virtual]
```

A parse főciklus.

Megvalósítja a következőket: pftd::utils::parser::Parser.

6.17.4. Adatmezők dokumentációja

6.17.4.1. m_attribs

```
std::unordered_map<std::string, std::vector<Vec2f> > pftd::utils::parser::LevelConfigParser↩
::m_attribs  [private]
```

Mentett [attribútum, érték] párok.

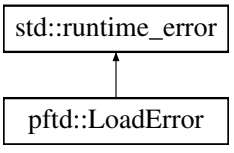
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.18. pftd::LoadError struktúrareferencia

```
#include <resources.hpp>
```

A pftd::LoadError osztály származási diagramja:



6.18.1. Részletes leírás

Erőforrás-betöltés hiba.

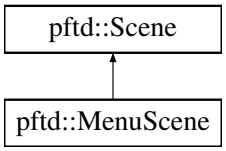
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/resources.hpp

6.19. pftd::MenuScene osztályreferencia

`#include <menu.hpp>`

A pftd::MenuScene osztály származási diagramja:



Publikus tagfüggvények

- MenuScene ()
- ~MenuScene ()=default
- void onEvent (sf::Event const &event) override

  Delegált események kezelése.
- void update (float dt) override

  Update.
- void toggleActive (Scene::StateFlag flag=Scene::StateFlag::NONE) override

  Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.

Privát tagfüggvények

- bool _isSaveFileAvailable ()

  Megnézi, hogy elérhető e a mentés fájl (`GameScene::SAVE_FILE_PATH` alapján).

Privát attribútumok

- std::vector< gr::Button * > m_buttons

További örökölt tagok

6.19.1. Részletes leírás

Menü nézet.

6.19.2. Konstruktorok és destruktorok dokumentációja

6.19.2.1. MenuScene()

```
pftd::MenuScene::MenuScene ( )
```

6.19.2.2. ~MenuScene()

```
pftd::MenuScene::~MenuScene ( )  [default]
```

6.19.3. Tagfüggvények dokumentációja

6.19.3.1. _isSaveFileAvailable()

```
bool pftd::MenuScene::_isSaveFileAvailable ( )  [private]
```

Megnézi, hogy elérhető e a mentés fájl (GameScene::SAVE_FILE_PATH alapján).

Visszatérési érték

Elérhető e a fájl.

6.19.3.2. onEvent()

```
void pftd::MenuScene::onEvent (
            sf::Event const & event )  [override], [virtual]
```

Delegált események kezelése.

Paraméterek

event	Kezelni kívánt esemény.

Megvalósítja a következőket: pftd::Scene.

6.19.3.3. toggleActive()

```
void pftd::MenuScene::toggleActive (
            Scene::StateFlag flag = Scene::StateFlag::NONE )  [override], [virtual]
```

Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.

Paraméterek

flag	Egyéb "kérést" közlő opció flag.

Újraimplementált ősök: pftd::Scene.

6.19.3.4. update()

```
void pftd::MenuScene::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Megvalósítja a következőket: pftd::Scene.

6.19.4. Adatmezők dokumentációja

6.19.4.1. m_buttons

```
std::vector<gr::Button*> pftd::MenuScene::m_buttons  [private]
```

Gombok.

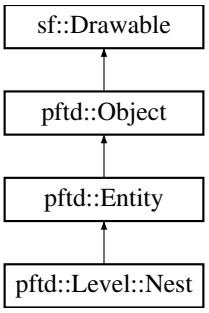
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/scenes/menu.hpp

6.20. pftd::Level::Nest struktúrareferencia

`#include <level.hpp>`

A pftd::Level::Nest osztály származási diagramja:



Publikus tagfüggvények

- Nest (utils::Vec2f const &position)

  A pozíción kívül minden más hard code-olt.
- Nest (Nest const &other)=default
- ~Nest ()=default
- Nest * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Adatmezők

- int const radiusPixel = 110

További örökölt tagok

6.20.1. Részletes leírás

Fészek.

6.20.2. Konstruktorok és destruktorok dokumentációja

6.20.2.1. Nest() [1/2]

```
pftd::Level::Nest::Nest (
            utils::Vec2f const & position )  [explicit]
```

A pozíción kívül minden más hard code-olt.

Paraméterek

position	Pozíció.

6.20.2.2. Nest() [2/2]

```
pftd::Level::Nest::Nest (
            Nest const & other )  [default]
```

6.20.2.3. ~Nest()

```
pftd::Level::Nest::~Nest ( )  [default]
```

6.20.3. Tagfüggvények dokumentációja

6.20.3.1. clone()

```
Nest * pftd::Level::Nest::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Entity.

6.20.4. Adatmezők dokumentációja

6.20.4.1. radiusPixel

```
int const pftd::Level::Nest::radiusPixel = 110
```

Az a környezete, amelybe tornyot nem lehet tenni.

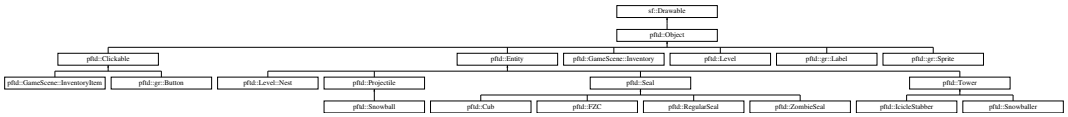
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/level.hpp

6.21. pftd::Object osztályreferencia

`#include <object_base.hpp>`

A pftd::Object osztály származási diagramja:



Adatszerkezetek

- struct Compare

Publikus tagfüggvények

- Object ()=default
- Object (utils::Vec2f position, utils::Vec2f size, int zIndex=0)
- Object (Object const &)=default
- Object (Object &&) noexcept=default
- virtual ~Object ()=default
- virtual void draw (sf::RenderTarget &target, sf::RenderStates states) const =0

  Objektum megjelenítése.

Védett attribútumok

- int zIndex = 0
- utils::Vec2f position
- utils::Vec2f size

6.21.1. Részletes leírás

Programelemek.

6.21.2. Konstruktorok és destruktorok dokumentációja

6.21.2.1. Object() [1/4]

```
pftd::Object::Object ( )  [default]
```

6.21.2.2. Object() [2/4]

```
pftd::Object::Object (
            utils::Vec2f position,
            utils::Vec2f size,
            int zIndex = 0 )  [inline]
```

Paraméterek

position	Pozíció koordinátái.
size	Méret.
zIndex	Z koordináta: "mélységi index."

6.21.2.3. Object() [3/4]

```
pftd::Object::Object (
            Object const &  )  [default]
```

6.21.2.4. Object() [4/4]

```
pftd::Object::Object (
            Object &&  )  [default], [noexcept]
```

6.21.2.5. ~Object()

```
virtual pftd::Object::~Object ( )  [virtual], [default]
```

6.21.3. Tagfüggvények dokumentációja

6.21.3.1. draw()

```
virtual void pftd::Object::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [pure virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítják a következők: pftd::Level, pftd::Entity, pftd::gr::Button, pftd::gr::Sprite, pftd::gr::Label, pftd::GameScene::InventoryItem és pftd::GameScene::Inventory.

6.21.4. Adatmezők dokumentációja

6.21.4.1. position

```
utils::Vec2f pftd::Object::position  [protected]
```

Pozíció.

6.21.4.2. size

```
utils::Vec2f pftd::Object::size  [protected]
```

Méret (szélesség x magasság).

6.21.4.3. zIndex

```
int pftd::Object::zIndex = 0  [protected]
```

Z koordináta.

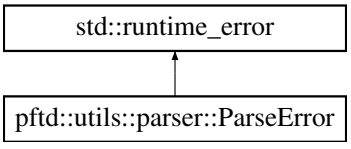
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/object_base.hpp

6.22. pftd::utils::parser::ParseError struktúrareferencia

```
#include <parsers.hpp>
```

A pftd::utils::parser::ParseError osztály származási diagramja:



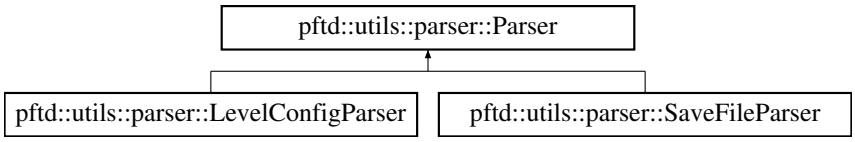
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.23. pftd::utils::parser::Parser osztályreferencia

`#include <parsers.hpp>`

A pftd::utils::parser::Parser osztály származási diagramja:



Publikus tagfüggvények

- Parser (std::string const &sourceFile, std::string label="")
- virtual ~Parser ()
- void reset ()

  Reseteli az IO flageket, és a stream elejére mozgatja a kurzort.
- bool isLabelValid (bool skip=true)

  Megmondja, hogy tényleg a megadott karaktersorral kezdődik e a stream.
- void skip (size_t howMany, char until='\n')

  Ugorjon át valamennyi karaktert, de legfeljebb egy megadottig.
- void skip (char while_)

  Ugorjon át mindent amíg a következő karakter egy megadott.
- template<typename T >
  T get ()

  Beolvas egy megadott típusú adatot.
- char peekAhead ()

  Megnézi, hogy milyen karakter van a kurzornál (lásd: megjegyzés).
- virtual void parse ()=0

  A parse főciklus.

Védett attribútumok

- std::ifstream sourceStream
- std::string validLabel
- char commentDenoter = '#'

Privát tagfüggvények

- void _skipLine ()

  Egész sor átugrása.
- void _skipWhitespace ()

  Whitespace-ek átugrása.

6.23.1. Konstruktorok és destruktorok dokumentációja

6.23.1.1. Parser()

```
pftd::utils::parser::Parser::Parser (
            std::string const & sourceFile,
            std::string label = "" )
```

Paraméterek

sourceFile	A fájlnak az elérési útvonala, ahonnan olvasni szeretnénk.
label	A karaktersor, amellyel kötelezően kezdődnie kell a fájlnak.

6.23.1.2. ~Parser()

```
virtual pftd::utils::parser::Parser::~Parser ( )  [virtual]
```

6.23.2. Tagfüggvények dokumentációja

6.23.2.1. _skipLine()

```
void pftd::utils::parser::Parser::_skipLine ( )  [private]
```

Egész sor átugrása.

6.23.2.2. _skipWhitespace()

```
void pftd::utils::parser::Parser::_skipWhitespace ( )  [private]
```

Whitespace-ek átugrása.

6.23.2.3. get()

```
template<typename T >
T pftd::utils::parser::Parser::get ( )  [inline]
```

Beolvas egy megadott típusú adatot.

Megjegyzés: a kommenteket átugorja.

Visszatérési érték

A beolvasott adat.

6.23.2.4. isLabelValid()

```
bool pftd::utils::parser::Parser::isLabelValid (
            bool skip = true )
```

Megmondja, hogy tényleg a megadott karaktersorral kezdődik e a stream.

Paraméterek

skip	Ha beolvasta a karaktersort, akkor skippelje e, vagy visszamozogjon.

Visszatérési érték

Helyes e a karaktersor.

6.23.2.5. parse()

```
virtual void pftd::utils::parser::Parser::parse ( )  [pure virtual]
```

A parse főciklus.

Megvalósítják a következők: pftd::utils::parser::LevelConfigParser és pftd::utils::parser::SaveFileParser.

6.23.2.6. peekAhead()

```
char pftd::utils::parser::Parser::peekAhead ( )
```

Megnézi, hogy milyen karakter van a kurzornál (lásd: megjegyzés).

Megjegyzés: átugorja a white space-eket és a kommenteket, és csak az azutáni karaktert nézi meg.

Visszatérési érték

A kurzornál levő karakter.

6.23.2.7. reset()

```
void pftd::utils::parser::Parser::reset ( )
```

Reseteli az IO flageket, és a stream elejére mozgatja a kurzort.

6.23.2.8. skip() [1/2]

```
void pftd::utils::parser::Parser::skip (
            char while_ )
```

Ugorjon át mindent amíg a következő karakter egy megadott.

Paraméterek

while↩_	Addig ugor át mindent, amíg ez a következő karakter.

6.23.2.9. skip() [2/2]

```
void pftd::utils::parser::Parser::skip (
            size_t howMany,
            char until = '\n' )
```

Ugorjon át valamennyi karaktert, de legfeljebb egy megadottig.

Paraméterek

howMany	Ennyi karaktert ugrik át legfeljebb.
until	Eddig a karakterig ugrik át mindent, ha még nem ugrott át `howMany` számút (ezt is beleértve).

6.23.3. Adatmezők dokumentációja

6.23.3.1. commentDenoter

```
char pftd::utils::parser::Parser::commentDenoter = '#'  [protected]
```

Kommentet jelölő karakter (ezt követően a sorban minden karakter a komment része).

6.23.3.2. sourceStream

```
std::ifstream pftd::utils::parser::Parser::sourceStream  [protected]
```

Input stream.

6.23.3.3. validLabel

```
std::string pftd::utils::parser::Parser::validLabel  [protected]
```

Elvárt karaktersor, amivel a streamnek kezdődnie kell.

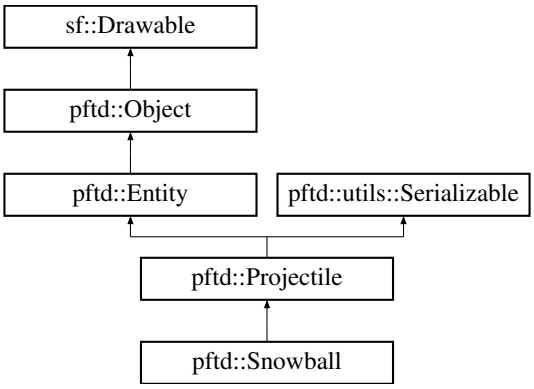
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.24. pftd::Projectile osztályreferencia

`#include <projectile_base.hpp>`

A pftd::Projectile osztály származási diagramja:



Publikus tagfüggvények

- virtual ~Projectile ()=default
- virtual void update (float dt) override

  Update.
- void serialize (std::ostream &out) const override

  Kiír egy streamre.

Adatmezők

- utils::Vec2f direction
- float linearSpeed
- float angularVelocityRadPerSec

Védett tagfüggvények

- Projectile (std::string const &spriteSrc, utils::Vec2f const &position, utils::Vec2f const &size, utils::Vec2f const &direction, float speed, float angularSpeed=0.0f, int zIndex=0)

Védett attribútumok

- ProjectileID id

6.24.1. Konstruktorok és destruktorok dokumentációja

6.24.1.1. ~Projectile()

```
virtual pftd::Projectile::~Projectile ( )  [virtual], [default]
```

6.24.1.2. Projectile()

```
pftd::Projectile::Projectile (
            std::string const & spriteSrc,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            utils::Vec2f const & direction,
            float speed,
            float angularSpeed = 0.0f,
            int zIndex = 0 )  [protected]
```

Paraméterek

spriteSrc	Sprite eleérési útvonal.
position	Pozíció.
size	Méret.
direction	Irány.
speed	Lineáris gyorsaság.
angularSpeed	Forgási sebesség (radián / mp).
zIndex	Z koordináta.

6.24.2. Tagfüggvények dokumentációja

6.24.2.1. serialize()

```
void pftd::Projectile::serialize (
            std::ostream & out ) const  [override], [virtual]
```

Kiír egy streamre.

Paraméterek

out	A stream.

Megvalósítja a következőket: pftd::utils::Serializable.

6.24.2.2. update()

```
virtual void pftd::Projectile::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementált ősök: pftd::Entity.

6.24.3. Adatmezők dokumentációja

6.24.3.1. angularVelocityRadPerSec

```
float pftd::Projectile::angularVelocityRadPerSec
```

Forgási sebesség.

6.24.3.2. direction

```
utils::Vec2f pftd::Projectile::direction
```

Irány (normalizált).

6.24.3.3. id

```
ProjectileID pftd::Projectile::id  [protected]
```

Mentéshez és betöltéshez használd azonosító.

6.24.3.4. linearSpeed

```
float pftd::Projectile::linearSpeed
```

Lineáris gyorsaság.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/projectiles/projectile_base.hpp

6.25. pftd::utils::Random osztályreferencia

`#include <random_gen.hpp>`

Publikus tagfüggvények

- Random ()=delete

Statikus publikus tagfüggvények

- template<typename RandomDist >
  static auto generate (RandomDist distribution)

  Random szám generálása megadott eloszlás szerint.

Statikus publikus attribútumok

- static std::default_random_engine randomEngine

6.25.1. Részletes leírás

Random szám generáláshoz használt segédosztály. Ez egy teljesen statikus osztály.

6.25.2. Konstruktorok és destruktorok dokumentációja

6.25.2.1. Random()

```
pftd::utils::Random::Random ( )  [delete]
```

6.25.3. Tagfüggvények dokumentációja

6.25.3.1. generate()

```
template<typename RandomDist >
static auto pftd::utils::Random::generate (
            RandomDist distribution )  [inline], [static]
```

Random szám generálása megadott eloszlás szerint.

Paraméterek

distribution	Generálandó számok eloszlása.

Visszatérési érték

Generált szám.

6.25.4. Adatmezők dokumentációja

6.25.4.1. randomEngine

```
std::default_random_engine pftd::utils::Random::randomEngine  [static]
```

Felhasznált random engine. (Az alapértelmezett.)

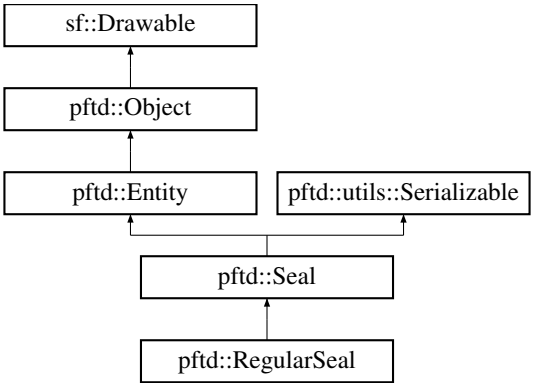
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/random_gen.hpp

6.26. pftd::RegularSeal osztályreferencia

```
#include <regular.hpp>
```

A pftd::RegularSeal osztály származási diagramja:



Publikus tagfüggvények

- RegularSeal (FollowPath const &followpath)
- Seal * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

További örökölt tagok

6.26.1. Részletes leírás

"Normális" fóka.

6.26.2. Konstruktorok és destruktorok dokumentációja

6.26.2.1. RegularSeal()

```
pftd::RegularSeal::RegularSeal (
            FollowPath const & followpath )
```

Paraméterek

followpath	Az útvonal amin végig kell menjen (majd pedig vissza).

6.26.3. Tagfüggvények dokumentációja

6.26.3.1. clone()

```
Seal * pftd::RegularSeal::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Seal.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/seals/regular.hpp

6.27. pftd::Renderer osztályreferencia

```
#include <app.hpp>
```

Publikus tagfüggvények

- Renderer (unsigned int width, unsigned int height, std::string const &windowTitle)

  Létrehozza az ablakot a rendering kontextussal együtt.
- Renderer (Renderer const &)=delete
- Renderer (Renderer &&)=delete
- ~Renderer ()

  Felszabadítja az ablak erőforrásait.
- void render ()

  Átmásolja a frame buffer-re az `m_queue`-ban tárolt objektumokat.
- void clear () const

  Törli a frame buffer-t.
- void display () const

  Megjeleníti a frame buffer tartalmát.
- void pushQueue (Object const *o)

  Hozzáad az `m_queue`-hoz egy megjelenítendő objektumot.
- sf::RenderWindow * getWindow () const

  `m_window` getter.

Privát attribútumok

- sf::RenderWindow * m_window = nullptr
- unsigned int m_width
- unsigned int m_height
- std::priority_queue< Object const *, std::vector< Object const * >, Object::Compare > m_queue

Barátok

- class App

6.27.1. Részletes leírás

A renderer.

6.27.2. Konstruktorok és destruktorok dokumentációja

6.27.2.1. Renderer() [1/3]

```
pftd::Renderer::Renderer (
            unsigned int width,
            unsigned int height,
            std::string const & windowTitle )
```

Létrehozza az ablakot a rendering kontextussal együtt.

Paraméterek

width	Az ablak szélessége.
height	Az ablak magassága.
windowTitle	Az ablak címe.

6.27.2.2. Renderer() [2/3]

```
pftd::Renderer::Renderer (
            Renderer const &  )  [delete]
```

6.27.2.3. Renderer() [3/3]

```
pftd::Renderer::Renderer (
            Renderer &&  )  [delete]
```

6.27.2.4. ~Renderer()

```
pftd::Renderer::~Renderer ( )
```

Felszabadítja az ablak erőforrásait.

6.27.3. Tagfüggvények dokumentációja

6.27.3.1. clear()

```
void pftd::Renderer::clear ( ) const  [inline]
```

Törli a frame buffer-t.

6.27.3.2. display()

```
void pftd::Renderer::display ( ) const  [inline]
```

Megjeleníti a frame buffer tartalmát.

6.27.3.3. getWindow()

```
sf::RenderWindow * pftd::Renderer::getWindow ( ) const  [inline]
```

`m_window` getter.

Visszatérési érték

A kezelt ablak.

6.27.3.4. pushQueue()

```
void pftd::Renderer::pushQueue (
            Object const * o )  [inline]
```

Hozzáad az `m_queue`-hoz egy megjelenítendő objektumot.

Paraméterek

o	Az objektum amit megjeleníteni szeretnénk.

6.27.3.5. render()

```
void pftd::Renderer::render ( )
```

Átmásolja a frame buffer-re az `m_queue`-ban tárolt objektumokat.

6.27.4. Barát és kapcsolódó függvények dokumentációja

6.27.4.1. App

```
friend class App  [friend]
```

6.27.5. Adatmezők dokumentációja

6.27.5.1. m_height

```
unsigned int pftd::Renderer::m_height  [private]
```

Az ablak magassága.

6.27.5.2. m_queue

```
std::priority_queue<Object const*, std::vector<Object const*>, Object::Compare> pftd::↩
Renderer::m_queue  [private]
```

A render queue.

6.27.5.3. m_width

```
unsigned int pftd::Renderer::m_width  [private]
```

Az ablak szélessége.

6.27.5.4. m_window

```
sf::RenderWindow* pftd::Renderer::m_window = nullptr  [private]
```

A kezelt ablak.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/app.hpp

6.28. pftd::ResourceManager osztályreferencia

```
#include <resources.hpp>
```

Publikus tagfüggvények

- ResourceManager (ResourceManager const &)=delete
- ResourceManager (ResourceManager &&)=delete
- ~ResourceManager ()=default

  Felszabadítja a tárolt erőforrásokat.
- void loadDefaultFont (std::string const &path)

  Az alapértelmezett (valójában csak ez van) betűtípus betöltése.
- sf::Texture const & getTexture (std::string const &source)

  Új vagy tárolt textúra megszerzése.
- sf::SoundBuffer const & getSound (std::string const &source)

  Új vagy tárolt audio buffer megszerzése.
- sf::Font const & getDefaultFont () const

  `m_defaultFont` *getter.*

Statikus publikus tagfüggvények

- static ResourceManager * getInstance ()

  A singleton instance megszerzése.
- static ResourceManager * create ()

  Létrehozza a singleton instance-t.
- static void destroy ()

  Törli a singleton instance-t.

Privát tagfüggvények

- ResourceManager ()=default

Privát attribútumok

- sf::Font m_defaultFont
- std::unordered_map< std::string, sf::Texture > m_textures
- std::unordered_map< std::string, sf::SoundBuffer > m_sounds

Statikus privát attribútumok

- static ResourceManager * m_instance

6.28.1. Részletes leírás

Az erőforrás-kezelő.

6.28.2. Konstruktorok és destruktorok dokumentációja

6.28.2.1. ResourceManager() [1/3]

```
pftd::ResourceManager::ResourceManager (
            ResourceManager const &  )  [delete]
```

6.28.2.2. ResourceManager() [2/3]

```
pftd::ResourceManager::ResourceManager (
            ResourceManager &&  )  [delete]
```

6.28.2.3. ~ResourceManager()

```
pftd::ResourceManager::~ResourceManager ( )  [default]
```

Felszabadítja a tárolt erőforrásokat.

6.28.2.4. ResourceManager() [3/3]

```
pftd::ResourceManager::ResourceManager ( )  [private], [default]
```

6.28.3. Tagfüggvények dokumentációja

6.28.3.1. create()

```
static ResourceManager * pftd::ResourceManager::create ( )  [static]
```

Létrehozza a singleton instance-t.

Megjegyzés: Meg kell hívni a `destroy()`-t, ha már nincs szükség rá!

Visszatérési érték

Az osztályhoz tartozó singleton instance.

6.28.3.2. destroy()

```
static void pftd::ResourceManager::destroy ( )  [inline], [static]
```

Törli a singleton instance-t.

6.28.3.3. getDefaultFont()

```
sf::Font const & pftd::ResourceManager::getDefaultFont ( ) const  [inline]
```

`m_defaultFont` getter.

Visszatérési érték

Az alapértelmezett betűtípus.

6.28.3.4. getInstance()

```
static ResourceManager * pftd::ResourceManager::getInstance ( )  [inline], [static]
```

A singleton instance megszerzése.

Visszatérési érték

Az osztályhoz tartozó singleton instance.

6.28.3.5. getSound()

```
sf::SoundBuffer const & pftd::ResourceManager::getSound (
            std::string const & source )
```

Új vagy tárolt audio buffer megszerzése.

Megjegyzés: LoadError-t dob, ha nem találja.

Paraméterek

source	A hangfájl elérési útvonala / azonosítója (megegyezik).

Visszatérési érték

A kért audio buffer.

6.28.3.6. getTexture()

```
sf::Texture const & pftd::ResourceManager::getTexture (
            std::string const & source )
```

Új vagy tárolt textúra megszerzése.

Megjegyzés: LoadError-t dob, ha nem találja.

Paraméterek

source	A textúra elérési útvonala / azonosítója (megegyezik).

Visszatérési érték

A kért textúra.

6.28.3.7. loadDefaultFont()

```
void pftd::ResourceManager::loadDefaultFont (
            std::string const & path )
```

Az alapértelmezett (valójában csak ez van) betűtípus betöltése.

Megjegyzés: LoadError-t dob, ha nem tudja betölteni bármi miatt is.

Paraméterek

path	A betűtípus elérési útvonala.

6.28.4. Adatmezők dokumentációja

6.28.4.1. m_defaultFont

```
sf::Font pftd::ResourceManager::m_defaultFont  [private]
```

Az alapértelmzett betűtípus.

6.28.4.2. m_instance

```
ResourceManager* pftd::ResourceManager::m_instance  [static], [private]
```

A singleton instance.

6.28.4.3. m_sounds

```
std::unordered_map<std::string, sf::SoundBuffer> pftd::ResourceManager::m_sounds  [private]
```

Audio: [source, hang buffer példány]

6.28.4.4. m_textures

```
std::unordered_map<std::string, sf::Texture> pftd::ResourceManager::m_textures  [private]
```

Textúrák: [source, textúra példány]

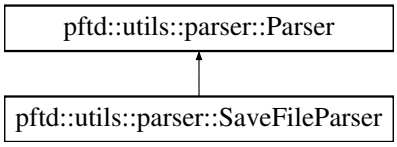
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/resources.hpp

6.29. pftd::utils::parser::SaveFileParser osztályreferencia

`#include <parsers.hpp>`

A pftd::utils::parser::SaveFileParser osztály származási diagramja:



Adatszerkezetek

- struct EntityInfo
- struct StatsInfo

Publikus típusok

- enum class EntityType { TOWER = 0U , SEAL , PROJECTILE }

Publikus tagfüggvények

- SaveFileParser (std::string const &sourceFile)
- ~SaveFileParser ()=default
- void parse () override

  A parse főciklus.
- StatsInfo const & getStats () const

  Mentett statisztika beolvasása: pont, pénz, HP.
- std::vector< EntityInfo > const & getEntities () const

  Mentett entitások beolvasása.

Privát tagfüggvények

- void _getStats ()

  Statisztika beolvasása a fájlból.
- void _getEntity ()

  Következ entitás beolvasása a fájlból.

Privát attribútumok

- StatsInfo m_readStats
- std::vector< EntityInfo > m_entities

További örökölt tagok

6.29.1. Részletes leírás

Mentett játékállás betöltésére szolgál.

6.29.2. Enumeráció-tagok dokumentációja

6.29.2.1. EntityType

```
enum class pftd::utils::parser::SaveFileParser::EntityType  [strong]
```

Mentett játék betöltéséhez használt entitás típusok.

Enumeráció-értékek

TOWER	
SEAL	
PROJECTILE	

6.29.3. Konstruktorok és destruktorok dokumentációja

6.29.3.1. SaveFileParser()

```
pftd::utils::parser::SaveFileParser::SaveFileParser (
            std::string const & sourceFile )
```

6.29.3.2. ∼SaveFileParser()

```
pftd::utils::parser::SaveFileParser::∼SaveFileParser ( )  [default]
```

6.29.4. Tagfüggvények dokumentációja

6.29.4.1. _getEntity()

```
void pftd::utils::parser::SaveFileParser::_getEntity ( )  [private]
```

Következ entitás beolvasása a fájlból.

6.29.4.2. _getStats()

```
void pftd::utils::parser::SaveFileParser::_getStats ( )  [private]
```

Statisztika beolvasása a fájlból.

6.29.4.3. getEntities()

```
std::vector< EntityInfo > const & pftd::utils::parser::SaveFileParser::getEntities ( ) const
[inline]
```

Mentett entitások beolvasása.

Visszatérési érték

Ezeket az entitásokat (azoknak a lényeges információját) tartalmazó lista.

6.29.4.4. getStats()

```
StatsInfo const & pftd::utils::parser::SaveFileParser::getStats ( ) const  [inline]
```

Mentett statisztika beolvasása: pont, pénz, HP.

Visszatérési érték

A mentett statisztikát tartalmazó struktúra.

6.29.4.5. parse()

```
void pftd::utils::parser::SaveFileParser::parse ( )  [override], [virtual]
```

A parse főciklus.

Megvalósítja a következőket: pftd::utils::parser::Parser.

6.29.5. Adatmezők dokumentációja

6.29.5.1. m_entities

`std::vector<EntityInfo> pftd::utils::parser::SaveFileParser::m_entities [private]`

A beolvasott entitás infók.

6.29.5.2. m_readStats

`StatsInfo pftd::utils::parser::SaveFileParser::m_readStats [private]`

A beolvasott statisztika.

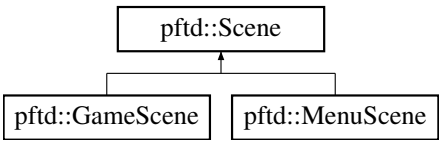
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.30. pftd::Scene osztályreferencia

`#include <scene.hpp>`

A pftd::Scene osztály származási diagramja:



Publikus típusok

- enum class StateFlag : uint8_t { NONE = 0 , LOAD_STATE = 1 << 0 }

Publikus tagfüggvények

- Scene ()=default
- Scene (Scene const &)=delete
- Scene (Scene &&) noexcept=delete
- virtual ~Scene ()

  Törli a kezelt objektumokat.
- ObjPtrVec const & getObjects () const

  `objects` getter (konstans).
- ObjPtrVec & getObjects ()

  `objects` getter.
- void setMusic (std::string const &source, float volume=100.0f)

  Zene beállítása (loopon megy amíg aktív a nézet).
- virtual void toggleActive (StateFlag flag=StateFlag::NONE)

  Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.
- virtual void onEvent (sf::Event const &event)=0

  Delegált események kezelése.
- virtual void update (float dt)=0

  Update.

Védett attribútumok

- bool isActive = false
- ObjPtrVec objects
- sf::Music * backgroundMusic = nullptr

Privát típusok

- using ObjPtrVec = std::vector< Object * >

6.30.1. Részletes leírás

Nézet ősosztály.

6.30.2. Típusdefiníció-tagok dokumentációja

6.30.2.1. ObjPtrVec

```
using pftd::Scene::ObjPtrVec = std::vector<Object*>  [private]
```

6.30.3. Enumeráció-tagok dokumentációja

6.30.3.1. StateFlag

```
enum class pftd::Scene::StateFlag :  uint8_t  [strong]
```

Plusz információt közlő opciók.

Enumeráció-értékek

NONE	
LOAD_STATE	

6.30.4. Konstruktorok és destruktorok dokumentációja

6.30.4.1. Scene() [1/3]

```
pftd::Scene::Scene ( )  [default]
```

6.30.4.2. Scene() [2/3]

```
pftd::Scene::Scene (
            Scene const &  )  [delete]
```

6.30.4.3. Scene() [3/3]

```
pftd::Scene::Scene (
            Scene &&  )  [delete], [noexcept]
```

6.30.4.4. ∼Scene()

```
virtual pftd::Scene::∼Scene ( )  [virtual]
```

Törli a kezelt objektumokat.

6.30.5. Tagfüggvények dokumentációja

6.30.5.1. getObjects() [1/2]

```
ObjPtrVec & pftd::Scene::getObjects ( )  [inline]
```

`objects` getter.

Visszatérési érték

A kezelt elemek listája.

6.30.5.2. getObjects() [2/2]

```
ObjPtrVec const & pftd::Scene::getObjects ( ) const  [inline]
```

`objects` getter (konstans).

Visszatérési érték

A kezelt elemek listája.

6.30.5.3. onEvent()

```
virtual void pftd::Scene::onEvent (
            sf::Event const & event )  [pure virtual]
```

Delegált események kezelése.

Paraméterek

event	Kezelni kívánt esemény.

Megvalósítják a következők: pftd::GameScene és pftd::MenuScene.

6.30.5.4. setMusic()

```
void pftd::Scene::setMusic (
            std::string const & source,
            float volume = 100.0f )
```

Zene beállítása (loopon megy amíg aktív a nézet).

Paraméterek

source	A hangfájl elérési útvonala.
volume	A zene hangereje.

6.30.5.5. toggleActive()

```
virtual void pftd::Scene::toggleActive (
            StateFlag flag = StateFlag::NONE )  [virtual]
```

Nézet aktiválása/deaktiválása: aktív -> nem aktív VAGY nem aktív -> aktív.

Paraméterek

flag	Egyéb "kérést" közlő opció flag.

Újraimplementáló leszármazottak: pftd::GameScene és pftd::MenuScene.

6.30.5.6. update()

```
virtual void pftd::Scene::update (
            float dt )  [pure virtual]
```

Update.

Paraméterek

dt	Delta idő.

Megvalósítják a következők: pftd::GameScene és pftd::MenuScene.

6.30.6. Adatmezők dokumentációja

6.30.6.1. backgroundMusic

```
sf::Music* pftd::Scene::backgroundMusic = nullptr  [protected]
```

Háttérzene.

6.30.6.2. isActive

```
bool pftd::Scene::isActive = false  [protected]
```

Aktív e.

6.30.6.3. objects

```
ObjPtrVec pftd::Scene::objects  [protected]
```

Kezelt objektumok.

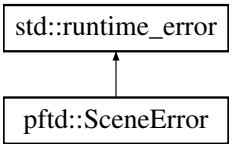
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/scene.hpp

6.31. pftd::SceneError struktúrareferencia

```
#include <app.hpp>
```

A pftd::SceneError osztály származási diagramja:



6.31.1. Részletes leírás

Nézethiba.

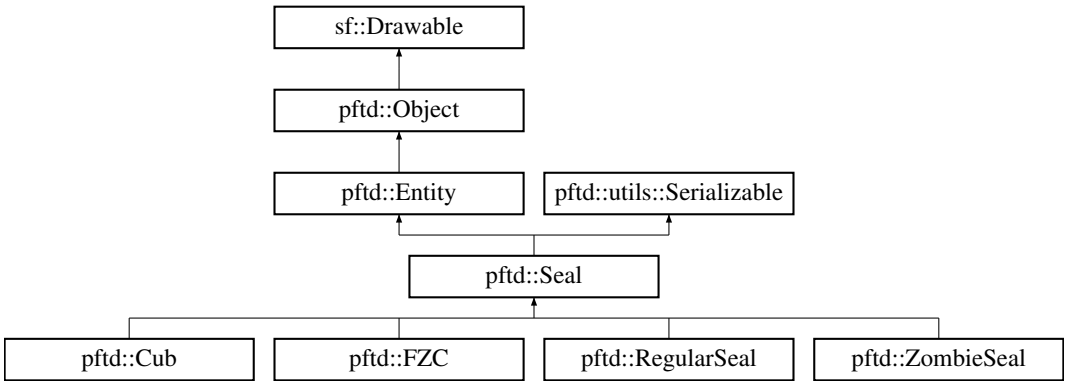
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/app.hpp

6.32. pftd::Seal osztályreferencia

`#include <seal_base.hpp>`

A pftd::Seal osztály származási diagramja:



Publikus tagfüggvények

- virtual ~Seal ()=default
- void lerpPath ()

  LERP paraméter frissítése.
- virtual Seal * clone () const override=0

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.
- void update (float dt) override

  Update.
- bool hasCompletedPath () const

  `returned` *getter.*
- bool hasReachedNest () const

  `reachedNest` *getter.*
- void setLerpState (float param, bool backwards=false)

  Szükség van rá, amikor mentett állást töltünk.
- void damage (int hpLost=1)

  Lesebzés.
- void serialize (std::ostream &out) const override

  Kiír egy streamre.

Adatmezők

- bool isCurrentlyStealing = false
- int hp
- float speed
- unsigned int value

Védett tagfüggvények

- Seal (FollowPath const &followPath, std::string const &spriteSrc, utils::Vec2f const &size, int hp, float speed, unsigned int value, int zIndex=0)

Védett attribútumok

- SealID id
- FollowPath const & followPath
- float lerpParam = 0.0f
- bool reachedNest = false
- bool returned = false
- EndPoint nextPoint

Privát tagfüggvények

- void advanceAnimationFrame () override

  Animáció: képkocka léptetése.

6.32.1. Részletes leírás

Fóka.

6.32.2. Konstruktorok és destruktorok dokumentációja

6.32.2.1. ~Seal()

```
virtual pftd::Seal::~Seal ( )  [virtual], [default]
```

6.32.2.2. Seal()

```
pftd::Seal::Seal (
            FollowPath const & followPath,
            std::string const & spriteSrc,
            utils::Vec2f const & size,
            int hp,
            float speed,
            unsigned int value,
            int zIndex = 0 )  [protected]
```

Paraméterek

followPath	Az útvonal, amit követnie kell.
spriteSrc	A sprite-jának az elérési útvonala.
size	Méret.
hp	HP.
speed	Gyorsaság.
value	Érték: ennyi pénzt kap a játékos ha megöli.
zIndex	Z koordináta.

6.32.3. Tagfüggvények dokumentációja

6.32.3.1. advanceAnimationFrame()

```
void pftd::Seal::advanceAnimationFrame ( )  [override], [private], [virtual]
```

Animáció: képkocka léptetése.

Újraimplementált ősök: pftd::Entity.

6.32.3.2. clone()

```
virtual Seal * pftd::Seal::clone ( ) const  [override], [pure virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Entity.

Megvalósítják a következők: pftd::Cub, pftd::FZC, pftd::RegularSeal és pftd::ZombieSeal.

6.32.3.3. damage()

```
void pftd::Seal::damage (
            int hpLost = 1 )
```

Lesebzés.

Paraméterek

hpLost	Ennyi HP-ja fog lemenni a fókának.

6.32.3.4. hasCompletedPath()

```
bool pftd::Seal::hasCompletedPath ( ) const  [inline]
```

`returned` getter.

Visszatérési érték

Elment e a fészekig és vissza a kezdőhelyre.

6.32.3.5. hasReachedNest()

```
bool pftd::Seal::hasReachedNest ( ) const  [inline]
```

`reachedNest` getter.

Visszatérési érték

Elment e a fészekig.

6.32.3.6. lerpPath()

```
void pftd::Seal::lerpPath ( )
```

LERP paraméter frissítése.

A `followPath` pontjai közt lineárisan interpolál.

6.32.3.7. serialize()

```
void pftd::Seal::serialize (
            std::ostream & out ) const  [override], [virtual]
```

Kiír egy streamre.

Paraméterek

out	A stream.

Megvalósítja a következőket: pftd::utils::Serializable.

6.32.3.8. setLerpState()

```
void pftd::Seal::setLerpState (
            float param,
            bool backwards = false )  [inline]
```

Szükség van rá, amikor mentett állást töltünk.

6.32.3.9. update()

```
void pftd::Seal::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementált ősök: pftd::Entity.

6.32.4. Adatmezők dokumentációja

6.32.4.1. followPath

```
FollowPath const& pftd::Seal::followPath  [protected]
```

Az útvonal amin végigmegy.

6.32.4.2. hp

```
int pftd::Seal::hp
```

HP.

6.32.4.3. id

```
SealID pftd::Seal::id  [protected]
```

6.32.4.4. isCurrentlyStealing

```
bool pftd::Seal::isCurrentlyStealing = false
```

Éppen lopás közben van e?

6.32.4.5. lerpParam

```
float pftd::Seal::lerpParam = 0.0f  [protected]
```

0-tól 1-ig terjedő paraméter az interpolációhoz.

6.32.4.6. nextPoint

```
EndPoint pftd::Seal::nextPoint  [protected]
```

A következő pont az útvonalon.

6.32.4.7. reachedNest

```
bool pftd::Seal::reachedNest = false  [protected]
```

Elérte e a fészket.

6.32.4.8. returned

```
bool pftd::Seal::returned = false  [protected]
```

Visszatért e a kezdő ponthoz.

6.32.4.9. speed

```
float pftd::Seal::speed
```

Gyorsaság.

6.32.4.10. value

```
unsigned int pftd::Seal::value
```

Pénz amit ad halál után.

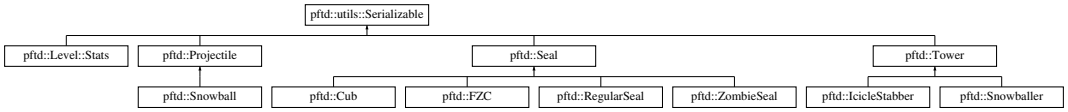
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/seals/seal_base.hpp

6.33. pftd::utils::Serializable osztályreferencia

```
#include <serializable.hpp>
```

A pftd::utils::Serializable osztály származási diagramja:



Publikus tagfüggvények

- virtual void serialize (std::ostream &out) const =0

  Kiír egy streamre.

6.33.1. Részletes leírás

Szérializálható osztályok.

6.33.2. Tagfüggvények dokumentációja

6.33.2.1. serialize()

```
virtual void pftd::utils::Serializable::serialize (
            std::ostream & out ) const  [pure virtual]
```

Kiír egy streamre.

Paraméterek

out	A stream.

Megvalósítják a következők: pftd::Level::Stats, pftd::Projectile, pftd::Seal és pftd::Tower.

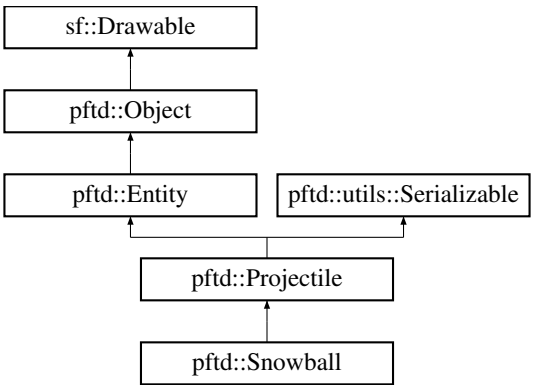
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/utils/serializable.hpp

6.34. pftd::Snowball osztályreferencia

```
#include <snowball.hpp>
```

A pftd::Snowball osztály származási diagramja:



Publikus tagfüggvények

- Snowball (utils::Vec2f const &position, utils::Vec2f direction, float speed=50.0f)

  A textúra hard code-olt.
- Snowball * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

További örökölt tagok

6.34.1. Részletes leírás

Hógolyó lövedék.

6.34.2. Konstruktorok és destruktorok dokumentációja

6.34.2.1. Snowball()

```
pftd::Snowball::Snowball (
            utils::Vec2f const & position,
            utils::Vec2f direction,
            float speed = 50.0f )
```

A textúra hard code-olt.

Paraméterek

position	Pozíció.
direction	Irány.
speed	Repülési gyorsaság.

6.34.3. Tagfüggvények dokumentációja

6.34.3.1. clone()

```
Snowball * pftd::Snowball::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Entity.

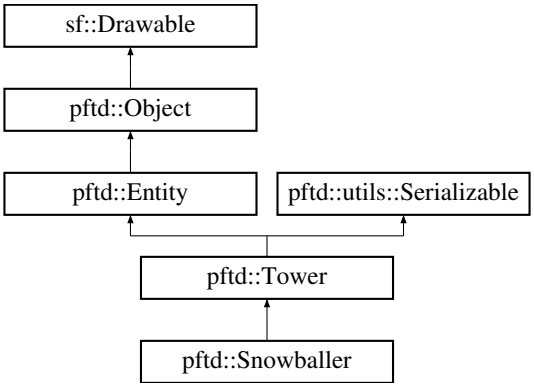
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/projectiles/snowball.hpp

6.35. pftd::Snowballer osztályreferencia

`#include <snowballer.hpp>`

A pftd::Snowballer osztály származási diagramja:



Publikus tagfüggvények

- Snowballer (utils::Vec2f const &position, int zIndex=0)
- Snowballer ()
- ~Snowballer ()=default
- Tower * clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.
- void update (float dt) override

  Update.
- void attack () override

  Megtámadja a `Tower::target`-et, ha van.

További örökölt tagok

6.35.1. Részletes leírás

Hógolyó dobáló.

6.35.2. Konstruktorok és destruktorok dokumentációja

6.35.2.1. Snowballer() [1/2]

```
pftd::Snowballer::Snowballer (
            utils::Vec2f const & position,
            int zIndex = 0 )
```

6.35.2.2. Snowballer() [2/2]

```
pftd::Snowballer::Snowballer ( )
```

6.35.2.3. ~Snowballer()

```
pftd::Snowballer::~Snowballer ( )  [default]
```

6.35.3. Tagfüggvények dokumentációja

6.35.3.1. attack()

```
void pftd::Snowballer::attack ( )  [override], [virtual]
```

Megtámadja a `Tower::target`-et, ha van.

Újraimplementált ősök: pftd::Tower.

6.35.3.2. clone()

```
Tower * pftd::Snowballer::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Tower.

6.35.3.3. update()

```
void pftd::Snowballer::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementált ősök: pftd::Tower.

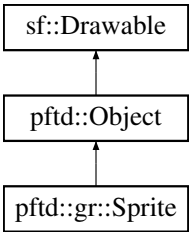
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/towers/snowballer.hpp

6.36. pftd::gr::Sprite osztályreferencia

`#include <image.hpp>`

A pftd::gr::Sprite osztály származási diagramja:



Publikus tagfüggvények

- Sprite (sf::Texture const &texture, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=-1)
- Sprite (sf::Texture const &texture, sf::IntRect const &textureRect, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=-1)
- Sprite (std::string const &imageSrc, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=-1)
- Sprite (Sprite const &other)
- Sprite (Sprite &&other) noexcept
- ~Sprite ()=default
- void modColor (sf::Color const &color)

  Sprite szín modulációjának beállítása.
- void flipY ()

  Y tengely mentén tükröz.
- void scale (utils::Vec2f const &factor)

  Nyújt.
- void setSpriteRect (sf::IntRect const &textureRect)

  A textúra csak egy részletének használata Sprite-ként.
- void setPosition (utils::Vec2f const &newPos)

  Beállítja az `m_sprite` pozícióját.
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Objektum megjelenítése.

Privát attribútumok

- sf::Texture const & m_texture
- sf::Sprite m_sprite

További örökölt tagok

6.36.1. Részletes leírás

Saját sprite osztály. Használható kép GUI elemként.

6.36.2. Konstruktorok és destruktorok dokumentációja

6.36.2.1. Sprite() [1/5]

```
pftd::gr::Sprite::Sprite (
            sf::Texture const & texture,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = -1 )
```

6.36.2.2. Sprite() [2/5]

```
pftd::gr::Sprite::Sprite (
            sf::Texture const & texture,
            sf::IntRect const & textureRect,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = -1 )
```

6.36.2.3. Sprite() [3/5]

```
pftd::gr::Sprite::Sprite (
            std::string const & imageSrc,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = -1 )
```

6.36.2.4. Sprite() [4/5]

```
pftd::gr::Sprite::Sprite (
            Sprite const & other )
```

6.36.2.5. Sprite() [5/5]

```
pftd::gr::Sprite::Sprite (
            Sprite && other )  [noexcept]
```

6.36.2.6. ~Sprite()

```
pftd::gr::Sprite::~Sprite ( )  [default]
```

6.36.3. Tagfüggvények dokumentációja

6.36.3.1. draw()

```
void pftd::gr::Sprite::draw (
            sf::RenderTarget & target,
            sf::RenderStates states ) const  [override], [virtual]
```

Objektum megjelenítése.

Paraméterek

target	Hol legyen megjelenítve? Ez általában egy ablak.
states	Render-eléshez szükséges egyéb állapotok.

Megvalósítja a következőket: pftd::Object.

6.36.3.2. flipY()

```
void pftd::gr::Sprite::flipY ( )
```

Y tengely mentén tükröz.

Nincs használva.

6.36.3.3. modColor()

```
void pftd::gr::Sprite::modColor (
            sf::Color const & color )
```

Sprite szín modulációjának beállítása.

Megjegyzés: Az SFML által biztosított modulációt használja.

Paraméterek

color	Szín, amivel szorzunk.

6.36.3.4. scale()

```
void pftd::gr::Sprite::scale (
            utils::Vec2f const & factor )
```

Nyújt.

Paraméterek

factor	Ennyivel nyújt az X-Y tengelyek mentén.

6.36.3.5. setPosition()

```
void pftd::gr::Sprite::setPosition (
            utils::Vec2f const & newPos )  [inline]
```

Beállítja az `m_sprite` pozícióját.

Paraméterek

newPos	Az új pozíció.

6.36.3.6. setSpriteRect()

```
void pftd::gr::Sprite::setSpriteRect (
            sf::IntRect const & textureRect )
```

A textúra csak egy részletének használata Sprite-ként.

Paraméterek

textureRect	A tárolt textúra része egy téglalappal megadva.

6.36.4. Adatmezők dokumentációja

6.36.4.1. m_sprite

```
sf::Sprite pftd::gr::Sprite::m_sprite  [private]
```

Sprite.

6.36.4.2. m_texture

```
sf::Texture const& pftd::gr::Sprite::m_texture  [private]
```

Sprite-hoz tartozó textúra.

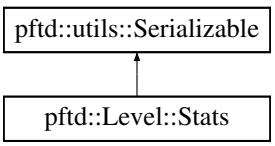
Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/gui/image.hpp

6.37. pftd::Level::Stats struktúrareferencia

```
#include <level.hpp>
```

A pftd::Level::Stats osztály származási diagramja:



Publikus tagfüggvények

- Stats ()=default
- Stats (int maxHp, int currentHp, unsigned int score, unsigned int wealth)
- void serialize (std::ostream &out) const override

  Kiír egy streamre.

Adatmezők

- int maxHp = 3
- int hp = maxHp
- unsigned int score = 0U
- unsigned int money = 100U

6.37.1. Részletes leírás

Játékállás / statisztika.

6.37.2. Konstruktorok és destruktorok dokumentációja

6.37.2.1. Stats() [1/2]

```
pftd::Level::Stats::Stats ( )  [explicit], [default]
```

6.37.2.2. Stats() [2/2]

```
pftd::Level::Stats::Stats (
            int maxHp,
            int currentHp,
            unsigned int score,
            unsigned int wealth )  [explicit]
```

6.37.3. Tagfüggvények dokumentációja

6.37.3.1. serialize()

```
void pftd::Level::Stats::serialize (
            std::ostream & out ) const  [override], [virtual]
```

Kiír egy streamre.

Paraméterek

out	A stream.

Megvalósítja a következőket: pftd::utils::Serializable.

6.37.4. Adatmezők dokumentációja

6.37.4.1. hp

```
int pftd::Level::Stats::hp = maxHp
```

Jelenlegi HP.

6.37.4.2. maxHp

```
int pftd::Level::Stats::maxHp = 3
```

A maximum HP.

6.37.4.3. money

```
unsigned int pftd::Level::Stats::money = 100U
```

Pénz.

6.37.4.4. score

```
unsigned int pftd::Level::Stats::score = 0U
```

Pontszám.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/level.hpp

6.38. pftd::utils::parser::SaveFileParser::StatsInfo struktúrareferencia

```
#include <parsers.hpp>
```

Adatmezők

- unsigned int score
- unsigned int wealth
- int maxHp
- int hp

6.38.1. Részletes leírás

Mentett játék betöltéséhez használt statisztika osztály.

6.38.2. Adatmezők dokumentációja

6.38.2.1. hp

```
int pftd::utils::parser::SaveFileParser::StatsInfo::hp
```

6.38.2.2. maxHp

```
int pftd::utils::parser::SaveFileParser::StatsInfo::maxHp
```

6.38.2.3. score

```
unsigned int pftd::utils::parser::SaveFileParser::StatsInfo::score
```

6.38.2.4. wealth

```
unsigned int pftd::utils::parser::SaveFileParser::StatsInfo::wealth
```

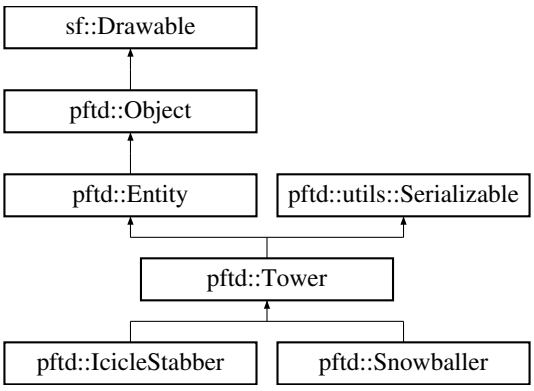
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/utils/parsers.hpp

6.39. pftd::Tower osztályreferencia

```
#include <tower_base.hpp>
```

A pftd::Tower osztály származási diagramja:



Adatszerkezetek

- struct TowerProperties

Publikus tagfüggvények

- Tower (TowerProperties const &props, std::string const &spriteSheetSrc, utils::Vec2i spriteSize, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0)
- Tower (TowerProperties const &props, std::string const &spriteSrc, utils::Vec2f const &position, utils::Vec2f const &size, int zIndex=0)
- Tower (Tower const &other)
- virtual ~Tower ()=default
- void setProjSpawnCb (ProjSpawnFunc callback)

  Lövedék kilövéséhez használt callback függvény beállítása.
- virtual Tower * clone () const override=0

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.
- virtual void attack ()

  Megtámadja a `target`-et, ha van.
- virtual bool lookForTarget (std::vector< Seal * > const &enemies)

  Megfelelő célpontot keres.
- virtual void update (float dt) override

  Update.
- void serialize (std::ostream &out) const override

  Kiír egy streamre.

Adatmezők

- struct pftd::Tower::TowerProperties properties

Védett tagfüggvények

- void advanceAnimationFrame () override

  Animáció: képkocka léptetése.

Védett attribútumok

- Seal * target = nullptr
- ProjSpawnFunc spawnProjectile
- float attackTimerSec = 0.0f

Privát típusok

- using ProjSpawnFunc = std::function< void(Projectile *)>

6.39.1. Részletes leírás

Torony (pingvin).

6.39.2. Típusdefiníció-tagok dokumentációja

6.39.2.1. ProjSpawnFunc

```
using pftd::Tower::ProjSpawnFunc = std::function<void(Projectile*)>  [private]
```

6.39.3. Konstruktorok és destruktorok dokumentációja

6.39.3.1. Tower() [1/3]

```
pftd::Tower::Tower (
            TowerProperties const & props,
            std::string const & spriteSheetSrc,
            utils::Vec2i spriteSize,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0 )
```

6.39.3.2. Tower() [2/3]

```
pftd::Tower::Tower (
            TowerProperties const & props,
            std::string const & spriteSrc,
            utils::Vec2f const & position,
            utils::Vec2f const & size,
            int zIndex = 0 )
```

6.39.3.3. Tower() [3/3]

```
pftd::Tower::Tower (
            Tower const & other )
```

6.39.3.4. ~Tower()

```
virtual pftd::Tower::~Tower ( )  [virtual], [default]
```

6.39.4. Tagfüggvények dokumentációja

6.39.4.1. advanceAnimationFrame()

```
void pftd::Tower::advanceAnimationFrame ( )  [override], [protected], [virtual]
```

Animáció: képkocka léptetése.

Újraimplementált ősök: pftd::Entity.

6.39.4.2. attack()

```
virtual void pftd::Tower::attack ( )  [virtual]
```

Megtámadja a `target`-et, ha van.

Újraimplementáló leszármazottak: pftd::Snowballer.

6.39.4.3. clone()

```
virtual Tower * pftd::Tower::clone ( ) const  [override], [pure virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Entity.

Megvalósítják a következők: pftd::IcicleStabber és pftd::Snowballer.

6.39.4.4. lookForTarget()

```
virtual bool pftd::Tower::lookForTarget (
            std::vector< Seal * > const & enemies )  [virtual]
```

Megfelelő célpontot keres.

Megjegyzés: csak akkor keres, ha a `target` egy nullptr.

Paraméterek

enemies	Ellenfelek, amik közt célpontot keres.

6.39.4.5. serialize()

```
void pftd::Tower::serialize (
            std::ostream & out ) const  [override], [virtual]
```

Kiír egy streamre.

Paraméterek

out	A stream.

Megvalósítja a következőket: pftd::utils::Serializable.

6.39.4.6. setProjSpawnCb()

```
void pftd::Tower::setProjSpawnCb (
            ProjSpawnFunc callback )
```

Lövedék kilövéséhez használt callback függvény beállítása.

Paraméterek

callback	A függvény.

6.39.4.7. update()

```
virtual void pftd::Tower::update (
            float dt )  [override], [virtual]
```

Update.

Paraméterek

dt	Delta idő.

Újraimplementált ősök: pftd::Entity.

Újraimplementáló leszármazottak: pftd::IcicleStabber és pftd::Snowballer.

6.39.5. Adatmezők dokumentációja

6.39.5.1. attackTimerSec

```
float pftd::Tower::attackTimerSec = 0.0f  [protected]
```

Támadások közti idő mérésére szolgáló számláló.

6.39.5.2. properties

```
struct pftd::Tower::TowerProperties pftd::Tower::properties
```

6.39.5.3. spawnProjectile

```
ProjSpawnFunc pftd::Tower::spawnProjectile  [protected]
```

Callback függvény, ha a torony távolsági (valamit kilő).

6.39.5.4. target

```
Seal* pftd::Tower::target = nullptr  [protected]
```

Célpont.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/towers/tower_base.hpp

6.40. pftd::Tower::TowerProperties struktúrareferencia

```
#include <tower_base.hpp>
```

Publikus tagfüggvények

- TowerProperties (TowerID id, float radius=120.0f, float attackRange=100.0f, float attackSpeed=1.0f, unsigned int attackDamage=1U, bool instant=false, unsigned int price=0U)

Adatmezők

- TowerID id
- float radiusPixel = 120.0f
- float attackRangePixel = 100.0f
- float attackSpeedSec = 1.0f
- unsigned int attackDamage = 1U
- bool instantAttack = false
- unsigned int price = 0U

6.40.1. Konstruktorok és destruktorok dokumentációja

6.40.1.1. TowerProperties()

```
pftd::Tower::TowerProperties::TowerProperties (
            TowerID id,
            float radius = 120.0f,
            float attackRange = 100.0f,
            float attackSpeed = 1.0f,
            unsigned int attackDamage = 1U,
            bool instant = false,
            unsigned int price = 0U )
```

6.40.2. Adatmezők dokumentációja

6.40.2.1. attackDamage

```
unsigned int pftd::Tower::TowerProperties::attackDamage = 1U
```

Támadás ereje: mennyi HP-t visz le.

6.40.2.2. attackRangePixel

```
float pftd::Tower::TowerProperties::attackRangePixel = 100.0f
```

Az a környezete, amiben célpontot keres.

6.40.2.3. attackSpeedSec

```
float pftd::Tower::TowerProperties::attackSpeedSec = 1.0f
```

Támadás gyorsasága.

6.40.2.4. id

```
TowerID pftd::Tower::TowerProperties::id
```

6.40.2.5. instantAttack

```
bool pftd::Tower::TowerProperties::instantAttack = false
```

Instant támadó (true)? Vagy lövedékkel (false)?

6.40.2.6. price

```
unsigned int pftd::Tower::TowerProperties::price = 0U
```

Az ára, amikor az "eszköztárban" van.

6.40.2.7. radiusPixel

```
float pftd::Tower::TowerProperties::radiusPixel = 120.0f
```

Az a környezete, amibe másik tornyot nem lehet lehelyezni.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/objects/entities/towers/tower_base.hpp

6.41. pftd::utils::Vec2< T > struktúrasablon-referencia

```
#include <substitute_types.hpp>
```

Publikus tagfüggvények

- Vec2 ()
- Vec2 (T x, T y)
- Vec2 (Vec2 const &right)
- Vec2 normalize () const

  Vektor normalizálása.
- Vec2 operator+ (Vec2 const &right) const
- Vec2 operator- (Vec2 const &right) const
- Vec2 operator/ (T scalar) const
- Vec2 operator∗ (T scalar) const
- bool operator== (Vec2 other) const
- bool operator!= (Vec2 other) const
- Vec2 & operator= (Vec2 const &right)

Statikus publikus tagfüggvények

- static float distance (Vec2 const &v1, Vec2 const &v2)

  Két vektor végpontja közti euklideszi távolság.

Adatmezők

- T x
- T y

Barátok

- std::ostream & operator<< (std::ostream &out, Vec2 const &vec)
- std::istream & operator>> (std::istream &in, Vec2 &vec)

6.41.1. Részletes leírás

template<typename T>
struct pftd::utils::Vec2< T >

sf::Vector2 helyett használt matematikai vektor implementáció.

6.41.2. Konstruktorok és destruktorok dokumentációja

6.41.2.1. Vec2() [1/3]

```
template<typename T >
pftd::utils::Vec2< T >::Vec2 ( )  [inline]
```

6.41.2.2. Vec2() [2/3]

```
template<typename T >
pftd::utils::Vec2< T >::Vec2 (
            T x,
            T y )  [inline]
```

6.41.2.3. Vec2() [3/3]

```
template<typename T >
pftd::utils::Vec2< T >::Vec2 (
            Vec2< T > const & right )  [inline]
```

6.41.3. Tagfüggvények dokumentációja

6.41.3.1. distance()

```
template<typename T >
static float pftd::utils::Vec2< T >::distance (
            Vec2< T > const & v1,
            Vec2< T > const & v2 )  [inline], [static]
```

Két vektor végpontja közti euklideszi távolság.

Paraméterek

v1	Egyik vektor.
v2	Másik vektor.

Visszatérési érték

A távolság.

6.41.3.2. normalize()

```
template<typename T >
Vec2 pftd::utils::Vec2< T >::normalize ( ) const  [inline]
```

Vektor normalizálása.

Ez a metódus nem módosítja a Vec2-t, hanem egy újat hoz létre belőle.

Visszatérési érték

Egy új, normalizált Vec2.

6.41.3.3. operator"!=()

```
template<typename T >
bool pftd::utils::Vec2< T >::operator!= (
            Vec2< T > other ) const  [inline]
```

6.41.3.4. operator*()

```
template<typename T >
Vec2 pftd::utils::Vec2< T >::operator* (
            T scalar ) const  [inline]
```

6.41.3.5. operator+()

```
template<typename T >
Vec2 pftd::utils::Vec2< T >::operator+ (
            Vec2< T > const & right ) const  [inline]
```

6.41.3.6. operator-()

```
template<typename T >
Vec2 pftd::utils::Vec2< T >::operator- (
            Vec2< T > const & right ) const  [inline]
```

6.41.3.7. operator/()

```
template<typename T >
Vec2 pftd::utils::Vec2< T >::operator/ (
            T scalar ) const  [inline]
```

6.41.3.8. operator=()

```
template<typename T >
Vec2 & pftd::utils::Vec2< T >::operator= (
            Vec2< T > const & right )  [inline]
```

6.41.3.9. operator==()

```
template<typename T >
bool pftd::utils::Vec2< T >::operator== (
            Vec2< T > other ) const  [inline]
```

6.41.4. Barát és kapcsolódó függvények dokumentációja

6.41.4.1. operator<<

```
template<typename T >
std::ostream & operator<< (
            std::ostream & out,
            Vec2< T > const & vec )  [friend]
```

6.41.4.2. operator>>

```
template<typename T >
std::istream & operator>> (
            std::istream & in,
            Vec2< T > & vec )  [friend]
```

6.41.5. Adatmezők dokumentációja

6.41.5.1. x

```
template<typename T >
T pftd::utils::Vec2< T >::x
```

X koordináta.

6.41.5.2. y

```
template<typename T >
T pftd::utils::Vec2< T >::y
```

Y koordináta.

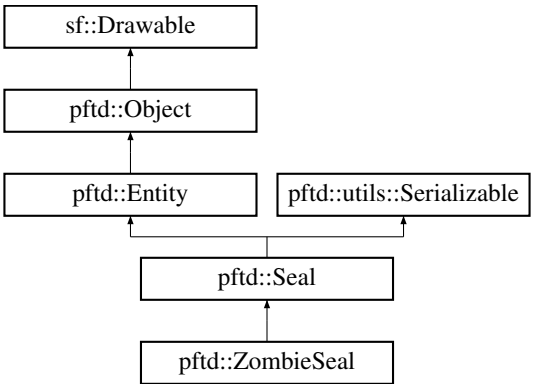
Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/utils/substitute_types.hpp

6.42. pftd::ZombieSeal osztályreferencia

`#include <zombie.hpp>`

A pftd::ZombieSeal osztály származási diagramja:



Publikus tagfüggvények

- ZombieSeal (FollowPath const &followpath)
- Seal ∗ clone () const override

  Egy dinamikus memóriában foglalt másolatot készít az entitásról.

További örökölt tagok

6.42.1. Részletes leírás

Zombi fóka.

6.42.2. Konstruktorok és destruktorok dokumentációja

6.42.2.1. ZombieSeal()

```
pftd::ZombieSeal::ZombieSeal (
            FollowPath const & followpath )
```

Paraméterek

followpath	Az útvonal amin végig kell menjen (majd pedig vissza).

6.42.3. Tagfüggvények dokumentációja

6.42.3.1. clone()

```
Seal * pftd::ZombieSeal::clone ( ) const  [override], [virtual]
```

Egy dinamikus memóriában foglalt másolatot készít az entitásról.

Visszatérési érték

A másolat.

Megvalósítja a következőket: pftd::Seal.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- include/objects/entities/seals/zombie.hpp

7. fejezet

Fájlok dokumentációja

7.1. include/all.hpp fájlreferencia

```
#include <cmath>
#include <vector>
#include <functional>
#include <optional>
#include <utility>
#include <unordered_map>
#include <map>
#include <queue>
#include <algorithm>
#include <ctime>
#include <ios>
#include <limits>
#include <stdexcept>
#include <random>
#include <fstream>
#include "SFML/Graphics.hpp"
#include "SFML/Audio.hpp"
#include "utils/logger.hpp"
#include "utils/substitute_types.hpp"
```

7.2. all.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 // Ez egy precompiled header fájl.
4
5 // STL
6 #include <cmath>
7 #include <vector>
8 #include <functional>
9 #include <optional>
10 #include <utility>
11 #include <unordered_map>
12 #include <map>
13 #include <queue>
14 #include <algorithm>
15 #include <ctime>
16 #include <ios>
```

```
17 #include <limits>
18 #include <stdexcept>
19 #include <random>
20 #include <fstream>
21
22 // SFML
23 #if not defined(_PFTD_TEST) && not defined(CPORTA)
24 #include "SFML/Graphics.hpp"
25 #include "SFML/Audio.hpp"
26 #endif
27
28 // Own
29 #include "utils/logger.hpp"
30 #include "utils/substitute_types.hpp"
```

7.3. include/app.hpp fájlreferencia

```
#include "objects/object_base.hpp"
#include "scene.hpp"
```

Adatszerkezetek

- class pftd::Renderer
- class pftd::App
- struct pftd::SceneError

Névterek

- namespace pftd

7.4. app.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/object_base.hpp"
6 #include "scene.hpp"
7
8 namespace pftd {
9
11 class Renderer final
12 {
13     friend class App;
14 public:
22     Renderer(unsigned int width, unsigned int height, std::string const& windowTitle);
23
24     Renderer(Renderer const&) = delete;
25     Renderer(Renderer&&) = delete;
26
30     ~Renderer();
31
35     void render();
36
40     void clear()const { m_window->clear(); }
41
45     void display()const { m_window->display(); }
46
52     void pushQueue(Object const* o) { m_queue.push(o); }
53
59     sf::RenderWindow* getWindow()const { return m_window; }
60
61 private:
```

```
63      sf::RenderWindow* m_window = nullptr;
64
66      unsigned int m_width;
67
69      unsigned int m_height;
70
72      std::priority_queue<Object const*, std::vector<Object const*>, Object::Compare> m_queue;
73
74  };
75
77  class App final
78  {
79  public:
80      App(App const&) = delete;
81      App(App&&) = delete;
82
88      ~App();
89
93      void run();
94
105      void addScene(std::string id, Scene* scene, bool active = false);
106
117      bool changeScene(std::string id, Scene::StateFlag flag = Scene::StateFlag::NONE);
118
124      auto isRunning()const { return this->m_running; }
125
131      int getWindowWidth()const { return this->m_renderer->m_width; }
132
138      int getWindowHeight()const { return this->m_renderer->m_height; }
139
140      App& operator=(App const&) = delete;
141
147      static App* getInstance() { return m_instance; }
148
159      static App* create(unsigned int width, unsigned int height, std::string const& windowTitle);
160
164      static void destroy() { delete m_instance; }
165
166  private:
167      // Az `m_instance` létrehozásához van használva.
168      App() = default;
169
171      static App* m_instance;
172
174      bool m_running = false;
175
177      Renderer* m_renderer = nullptr;
178
180      std::map<std::string, Scene*> m_scenes;
181
183      std::string m_activeSceneID;
184
185  };
186
188  struct SceneError :  public std::runtime_error
189  {
190      using std::runtime_error::runtime_error;
191  };
192
193  }
194
195  #endif
```

7.5. include/game/level.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/entities/projectiles/projectile_base.hpp"
#include "objects/entities/seals/seal_base.hpp"
#include "objects/entities/towers/tower_base.hpp"
#include "utils/serializable.hpp"
#include "utils/parsers.hpp"
```

Adatszerkezetek

- class pftd::Level

- struct pftd::Level::Nest
- struct pftd::Level::Stats

Névterek

- namespace pftd

7.6. level.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 #include "objects/entities/projectiles/projectile_base.hpp"
8 #include "objects/entities/seals/seal_base.hpp"
9 #include "objects/entities/towers/tower_base.hpp"
10 #include "utils/serializable.hpp"
11 #include "utils/parsers.hpp"
12
13 namespace pftd {
14
16 class Level :  public Object
17 {
18 public:
20     struct Nest :  public Entity
21     {
23         int const radiusPixel = 110;
24
30         explicit Nest(utils::Vec2f const& position);
31         Nest(Nest const& other) = default;
32
33         ~Nest() = default;
34
35         Nest* clone() const override;
36     };
37
39     struct Stats :  public utils::Serializable
40     {
42         int maxHp = 3;
43
45         int hp = maxHp;
46
48         unsigned int score = 0U;
49
51         unsigned int money = 100U;
52
53         explicit Stats() = default;
54         explicit Stats(int maxHp, int currentHp, unsigned int score, unsigned int wealth);
55
56         void serialize(std::ostream& out) const override;
57     };
58
60     Stats stats = Stats{};
61
63     Tower* selectedTower = nullptr;
64
71     Level(std::string const& saveFile, Stats stats);
72
80     Level(std::string const& saveFile);
81
82     virtual ~Level();
83
89     virtual void loseHP(int hpLost = 1);
90
96     virtual bool placeTower();
97
101     void deselectTower();
102
110     void selectTower(Tower* newTower);
111
117     void update(float dt);
118
124     bool isGameOver()const { return stats.hp <= 0; };
```

```
125
131     void save() const;
132
138     void reset(Stats stats);
139
140     virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
141
142 protected:
144     Nest* nest = nullptr;
145
147     FollowPath followPath;
148
150     std::vector<Tower*> towers;
151
153     std::vector<Seal*> seals;
154
156     std::vector<Projectile*> projectiles;
157
159     utils::parser::LevelConfigParser config;
160
162     std::string saveFile;
163
167     void spawnSeal();
168
169 private:
171     float m_accuTimeSpawnSec = 0.0f;
172
176     void _spawnSeal(float dt);
177
181     void _updateSeals(float dt);
182
186     void _updateTowers(float dt);
187
191     void _updateProjectiles(float dt);
192
193 };
194
195 }
196
197 #endif
```

7.7. include/objects/clickable.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/object_base.hpp"
```

Adatszerkezetek

- class pftd::Clickable

Névterek

- namespace pftd

7.8. clickable.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 #include "objects/object_base.hpp"
8
```

```
9 namespace pftd {
10
12 class Clickable :  public Object
13 {
14 public:
16     bool isActive;
17
18     Clickable(bool active = true);
19     Clickable(utils::Vec2f const& position, utils::Vec2f const& size, int zIndex = 0, bool active =
       true);
20
26     virtual void setCallback(std::function<void()> callback);
27
33     virtual void handleClick(utils::Vec2i const& clickCoords);
34
35 protected:
37     std::function<void()> m_callback;
38
39 };
40
41 }
42
43 #endif
```

7.9. include/objects/entities/all_entities.hpp fájlreferencia

```
#include "objects/entities/entity_base.hpp"
#include "objects/entities/projectiles/projectile_base.hpp"
#include "objects/entities/projectiles/snowball.hpp"
#include "objects/entities/seals/seal_base.hpp"
#include "objects/entities/seals/cub.hpp"
#include "objects/entities/seals/fortified_zombie_cub.hpp"
#include "objects/entities/seals/regular.hpp"
#include "objects/entities/seals/zombie.hpp"
#include "objects/entities/towers/tower_base.hpp"
#include "objects/entities/towers/iciclestabber.hpp"
#include "objects/entities/towers/snowballer.hpp"
```

7.10. all_entities.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "objects/entities/entity_base.hpp"
4
5 #include "objects/entities/projectiles/projectile_base.hpp"
6 #include "objects/entities/projectiles/snowball.hpp"
7
8 #include "objects/entities/seals/seal_base.hpp"
9 #include "objects/entities/seals/cub.hpp"
10 #include "objects/entities/seals/fortified_zombie_cub.hpp"
11 #include "objects/entities/seals/regular.hpp"
12 #include "objects/entities/seals/zombie.hpp"
13
14 #include "objects/entities/towers/tower_base.hpp"
15 #include "objects/entities/towers/iciclestabber.hpp"
16 #include "objects/entities/towers/snowballer.hpp"
```

7.11. include/objects/entities/entity_base.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/gui/image.hpp"
#include "objects/object_base.hpp"
```

Adatszerkezetek

- class pftd::Entity

Névterek

- namespace pftd

7.12. entity_base.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 #include "objects/gui/image.hpp"
8 #include "objects/object_base.hpp"
9
10 namespace pftd {
11
13 class Entity :  public Object
14 {
15 public:
17     bool isAnimated = true;
18
19     Entity(std::string const& spriteSheetSrc, utils::Vec2i spriteSize, utils::Vec2f const& position,
       utils::Vec2f const& size, int zIndex = 0);
20     Entity(sf::Texture const& texture, utils::Vec2i spriteSize, utils::Vec2f const& position,
       utils::Vec2f const& size, int zIndex = 0);
21     Entity(std::string const& spriteSrc, utils::Vec2f const& position, utils::Vec2f const& size, int
       zIndex = 0);
22     Entity(Entity const& other);
23     virtual ~Entity();
24
30     virtual Entity* clone() const = 0;
31
37     virtual void update(float dt);
38
39     virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
40
46     void setPosition(utils::Vec2f position);
47
53     auto getPosition()const { return position; }
54
60     gr::Sprite const* getSprite()const { return currentSprite; }
61
67     gr::Sprite* getSprite() { return currentSprite; }
68
74     auto& getSpriteSheet()const { return spriteSheet; }
75
76 protected:
78     sf::Texture const& spriteSheet; // Feltételezzük, hogy egy sorban vannak a sprite-ok
79
81     utils::Vec2i cellSize;
82
84     size_t const CELL_N;
85
87     size_t currentCell = 0U;
88
90     gr::Sprite* currentSprite = nullptr;
91
93     float frameDurationSec = 1.0f;
94
96     float totalElapsedSec = 0.0f; // Animációhoz van használva elsősorban.
97
101     virtual void advanceAnimationFrame();
102
106     virtual void resetAnimation();
107
108 };
109
110 }
111
112 #endif
```

7.13. include/objects/entities/projectiles/projectile_base.hpp fájlreferencia

```
#include "objects/entities/entity_base.hpp"
#include "utils/serializable.hpp"
#include "utils/substitute_types.hpp"
```

Adatszerkezetek

- class pftd::Projectile

Névterek

- namespace pftd

Enumerációk

- enum class pftd::ProjectileID { pftd::SNOWBALL = 0 }

7.14. projectile_base.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/entity_base.hpp"
6 #include "utils/serializable.hpp"
7 #include "utils/substitute_types.hpp"
8
9 namespace pftd {
10
11 enum class ProjectileID
12 {
13     SNOWBALL = 0,
14 };
15
16 class Projectile :  public Entity, public utils::Serializable
17 {
18 public:
20     utils::Vec2f direction;
21
23     float linearSpeed;
24
26     float angularVelocityRadPerSec;
27
28     virtual ~Projectile() = default;
29
30     virtual void update(float dt) override;
31
32     void serialize(std::ostream& out) const override;
33
34 protected:
36     ProjectileID id;
37
49     Projectile(std::string const& spriteSrc, utils::Vec2f const& position,
50         utils::Vec2f const& size, utils::Vec2f const& direction, float speed, float angularSpeed = 0.0f,
      int zIndex = 0);
51
52 };
53
54 }
55
56 #endif
```

7.15. include/objects/entities/projectiles/snowball.hpp fájlreferencia

```
#include "objects/entities/projectiles/projectile_base.hpp"
```

Adatszerkezetek

- class pftd::Snowball

Névterek

- namespace pftd

7.16. snowball.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/projectiles/projectile_base.hpp"
6
7 namespace pftd {
8
10 class Snowball :  public Projectile
11 {
12 public:
20     Snowball(utils::Vec2f const& position, utils::Vec2f direction, float speed = 50.0f);
21
22     Snowball* clone() const override;
23
24 };
25
26 }
27
28 #endif
```

7.17. include/objects/entities/seals/cub.hpp fájlreferencia

```
#include "objects/entities/seals/seal_base.hpp"
```

Adatszerkezetek

- class pftd::Cub

Névterek

- namespace pftd

7.18. cub.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/seals/seal_base.hpp"
6
7 namespace pftd {
8
10 class Cub :  public Seal
11 {
12 public:
18     Cub(FollowPath const& followpath);
19
20     Seal* clone() const override;
21
22 };
23
24 }
25
26 #endif
```

7.19. include/objects/entities/seals/followpath.hpp fájlreferencia

```
#include "utils/hetero_collection.hpp"
```

Adatszerkezetek

- struct pftd::FollowPath

Névterek

- namespace pftd

Típusdefiníciók

- using pftd::EndPoint = utils::Vec2f

7.20. followpath.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "utils/hetero_collection.hpp"
4
5 namespace pftd {
6
7 using EndPoint = utils::Vec2f;
8
10 struct FollowPath :  public utils::Container<EndPoint>
11 {
15     FollowPath() = default;
16 };
17
18 }
```

7.21. include/objects/entities/seals/fortified_zombie_cub.hpp fájlreferencia

```
#include "objects/entities/seals/seal_base.hpp"
```

Adatszerkezetek

- class pftd::FZC

Névterek

- namespace pftd

7.22. fortified_zombie_cub.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/seals/seal_base.hpp"
6
7 namespace pftd {
8
10 class FZC :  public Seal
11 {
12 public:
18     FZC(FollowPath const& followpath);
19
20     Seal* clone() const override;
21
22 };
23
24 }
25
26 #endif
```

7.23. include/objects/entities/seals/regular.hpp fájlreferencia

```
#include "objects/entities/seals/seal_base.hpp"
```

Adatszerkezetek

- class pftd::RegularSeal

Névterek

- namespace pftd

7.24. regular.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/seals/seal_base.hpp"
6
7 namespace pftd {
8
10 class RegularSeal :  public Seal
11 {
12 public:
18     RegularSeal(FollowPath const& followpath);
19
20     Seal* clone() const override;
21
22 };
23
24 }
25
26 #endif
```

7.25. include/objects/entities/seals/seal_base.hpp fájlreferencia

```
#include "objects/entities/entity_base.hpp"
#include "objects/entities/seals/followpath.hpp"
#include "utils/serializable.hpp"
#include "utils/substitute_types.hpp"
```

Adatszerkezetek

- class pftd::Seal

Névterek

- namespace pftd

Enumerációk

- enum class pftd::SealID { pftd::REGULAR = 0 , pftd::CUB , pftd::ZOMBIE , pftd::FZC }

7.26. seal_base.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/entity_base.hpp"
6 #include "objects/entities/seals/followpath.hpp"
7 #include "utils/serializable.hpp"
8 #include "utils/substitute_types.hpp"
9
10 namespace pftd {
11
12 enum class SealID
```

```
13 {
14     REGULAR = 0,
15     CUB,
16     ZOMBIE,
17     FZC
18 };
19
21 class Seal :  public Entity, public utils::Serializable
22 {
23 public:
25     bool isCurrentlyStealing = false;
26
28     int hp;
29
31     float speed;
32
34     unsigned int value;
35
36     virtual ~Seal() = default;
37
43     void lerpPath();
44
45     virtual Seal* clone() const override = 0;
46
47     void update(float dt) override;
48
54     bool hasCompletedPath()const { return returned; }
55
61     bool hasReachedNest()const { return reachedNest; }
62
68     void setLerpState(float param, bool backwards = false)
69     {
70         lerpParam = std::min(1.0f, std::max(param, 0.0f));
71         reachedNest = backwards;
72     }
73
79     void damage(int hpLost = 1);
80
81     void serialize(std::ostream& out) const override;
82
83 protected:
84     SealID id;
85
97     Seal(FollowPath const& followPath, std::string const& spriteSrc, utils::Vec2f const& size, int hp,
      float speed, unsigned int value, int zIndex = 0);
98
100      FollowPath const& followPath;
101
102     // Interpolációhoz
103
105     float lerpParam = 0.0f;
106
108     bool reachedNest = false;
109
111     bool returned = false;
112
114     EndPoint nextPoint;
115
116 private:
117     void advanceAnimationFrame() override;
118
119 };
120
121 }
122
123 #endif
```

7.27. include/objects/entities/seals/zombie.hpp fájlreferencia

```
#include "objects/entities/seals/seal_base.hpp"
```

Adatszerkezetek

- class pftd::ZombieSeal

Névterek

- namespace pftd

7.28. zombie.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/seals/seal_base.hpp"
6
7 namespace pftd {
8
10 class ZombieSeal :  public Seal
11 {
12 public:
18     ZombieSeal(FollowPath const& followpath);
19
20     Seal* clone() const override;
21
22 };
23
24 }
25
26 #endif
```

7.29. include/objects/entities/towers/iciclestabber.hpp fájlreferencia

```
#include "objects/entities/towers/tower_base.hpp"
```

Adatszerkezetek

- class pftd::IcicleStabber

Névterek

- namespace pftd

7.30. iciclestabber.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/towers/tower_base.hpp"
6
7 namespace pftd {
8
10 class IcicleStabber final :  public Tower
11 {
12 public:
13     IcicleStabber(utils::Vec2f const& position, int zIndex = 0);
14     IcicleStabber();
15     ~IcicleStabber() = default;
16
17     Tower* clone() const override;
18
19     void update(float dt) override;
20
21 };
22
23 }
24
25 #endif
```

7.31. include/objects/entities/towers/snowballer.hpp fájlreferencia

```
#include "objects/entities/towers/tower_base.hpp"
```

Adatszerkezetek

- class pftd::Snowballer

Névterek

- namespace pftd

7.32. snowballer.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/towers/tower_base.hpp"
6
7 namespace pftd {
8
10 class Snowballer final :  public Tower
11 {
12 public:
13     Snowballer(utils::Vec2f const& position, int zIndex = 0);
14     Snowballer();
15     ~Snowballer() = default;
16
17     Tower* clone() const override;
18
19     void update(float dt) override;
20
24     void attack() override;
25
26 };
27
28 }
29
30 #endif
```

7.33. include/objects/entities/towers/tower_base.hpp fájlreferencia

```
#include "objects/entities/entity_base.hpp"
#include "objects/entities/projectiles/projectile_base.hpp"
#include "objects/entities/seals/seal_base.hpp"
#include "utils/serializable.hpp"
```

Adatszerkezetek

- class pftd::Tower
- struct pftd::Tower::TowerProperties

Névterek

- namespace pftd

Enumerációk

- enum class pftd::TowerID { pftd::SNOWBALLER = 0 , pftd::ICICLE_STABBER }

7.34. tower_base.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/entities/entity_base.hpp"
6 #include "objects/entities/projectiles/projectile_base.hpp"
7 #include "objects/entities/seals/seal_base.hpp"
8 #include "utils/serializable.hpp"
9
10 namespace pftd {
11
12 enum class TowerID
13 {
14     SNOWBALLER = 0,
15     ICICLE_STABBER
16 };
17
19 class Tower :  public Entity, public utils::Serializable
20 {
21     using ProjSpawnFunc = std::function<void(Projectile*)>;
22 public:
23     struct TowerProperties
24     {
25         TowerID id;
26
28         float radiusPixel = 120.0f;
29
31         float attackRangePixel = 100.0f;
32
34         float attackSpeedSec = 1.0f;
35
37         unsigned int attackDamage = 1U;
38
40         bool instantAttack = false;
41
43         unsigned int price = 0U;
44
45         TowerProperties(TowerID id, float radius = 120.0f, float attackRange = 100.0f, float attackSpeed
    = 1.0f,
46             unsigned int attackDamage = 1U, bool instant = false, unsigned int price = 0U);
47
48     } properties;
49
50     Tower(TowerProperties const& props, std::string const& spriteSheetSrc, utils::Vec2i spriteSize,
    utils::Vec2f const& position, utils::Vec2f const& size, int zIndex = 0);
51     Tower(TowerProperties const& props, std::string const& spriteSrc, utils::Vec2f const& position,
    utils::Vec2f const& size, int zIndex = 0);
52     Tower(Tower const& other);
53     virtual ~Tower() = default;
54
60     void setProjSpawnCb(ProjSpawnFunc callback);
61
62     virtual Tower* clone() const override = 0;
63
67     virtual void attack();
68
76     virtual bool lookForTarget(std::vector<Seal*> const& enemies);
77
78     virtual void update(float dt) override;
79
80     void serialize(std::ostream& out) const override;
81
82 protected:
84     Seal* target = nullptr;
85
```

```
87     ProjSpawnFunc spawnProjectile;
88
90     float attackTimerSec = 0.0f;
91
92     void advanceAnimationFrame() override;
93 };
94
95 }
96
97 #endif
```

7.35. include/objects/gui/button.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/clickable.hpp"
#include "objects/gui/image.hpp"
#include "objects/gui/label.hpp"
```

Adatszerkezetek

- class pftd::gr::Button

Névterek

- namespace pftd
- namespace pftd::gr

7.36. button.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 #include "objects/clickable.hpp"
8 #include "objects/gui/image.hpp"
9 #include "objects/gui/label.hpp"
10
11 namespace pftd {
12 namespace gr {
13
15 class Button :  public Clickable
16 {
17 public:
27     Button(gr::Label const& label, utils::Vec2f const& position, utils::Vec2f const& size, bool active =
       true, int zIndex = 0);
28     virtual ~Button();
29
35     void setSound(std::string const& src);
36
42     void setBackground(std::string const& src);
43
49     std::string getLabel()const { return static_cast<std::string>(label.getText().getString()); };
50
51     virtual void handleClick(utils::Vec2i const& clickCoords) override;
52
53     virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
54
55 protected:
57     Label label;
58
```

```
59 private:
61     sf::FloatRect m_rect;
62
64     Sprite* m_background = nullptr;
65
67     sf::Sound m_clickSound;
68
69 };
70
71 }
72 }
73
74 #endif
```

7.37. include/objects/gui/image.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/object_base.hpp"
```

Adatszerkezetek

- class pftd::gr::Sprite

Névterek

- namespace pftd
- namespace pftd::gr

7.38. image.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 #include "objects/object_base.hpp"
8
9 namespace pftd {
10 namespace gr {
11
13 class Sprite final :  public Object
14 {
15 public:
16     Sprite(sf::Texture const& texture, utils::Vec2f const& position, utils::Vec2f const& size, int zIndex
      = -1);
17     Sprite(sf::Texture const& texture, sf::IntRect const& textureRect, utils::Vec2f const& position,
      utils::Vec2f const& size, int zIndex = -1);
18     Sprite(std::string const& imageSrc, utils::Vec2f const& position, utils::Vec2f const& size, int
      zIndex = -1);
19
20     Sprite(Sprite const& other);
21     Sprite(Sprite&& other) noexcept;
22
23     ~Sprite() = default;
24
32     void modColor(sf::Color const& color);
33
39     void flipY();
40
46     void scale(utils::Vec2f const& factor);
47
53     void setSpriteRect(sf::IntRect const& textureRect);
```

```
54
60      void setPosition(utils::Vec2f const& newPos) { m_sprite.setPosition({newPos.x, newPos.y}); }
61
62      void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
63
64  private:
66      sf::Texture const& m_texture;
67
69      sf::Sprite m_sprite;
70
71  };
72
73  }
74  }
75
76  #endif
```

7.39. include/objects/gui/label.hpp fájlreferencia

```
#include "all.hpp"
#include "objects/object_base.hpp"
```

Adatszerkezetek

- class pftd::gr::Label

Névterek

- namespace pftd
- namespace pftd::gr

7.40. label.hpp

Ugrás a fájl dokumentációjához.

```
1  #pragma once
2
3  #ifndef CPORTA
4
5  #include "all.hpp"
6
7  #include "objects/object_base.hpp"
8
9  namespace pftd {
10 namespace gr {
11
13 class Label final :  public Object
14 {
15 public:
24      Label(std::wstring const& label, sf::Font const& font, unsigned int cSize, sf::Color const& color =
       sf::Color::White);
25
36      Label(std::wstring const& label, sf::Font const& font, unsigned int cSize, utils::Vec2f position, int
       zIndex = 0, sf::Color const& color = sf::Color::White);
37
43      void setLabel(std::string const& newLabel) { m_text.setString(newLabel); }
44
51      void setOutline(sf::Color color, float thickness = 1.0f) { m_text.setOutlineColor(color);
       m_text.setOutlineThickness(thickness); }
52
58      sf::Text& getText() { return m_text; }
59
65      sf::Text const& getText()const { return m_text; }
66
67      void draw(sf::RenderTarget& target, sf::RenderStates states)const override
```

```
68 {
69         target.draw(m_text, states);
70     }
71
72 private:
74     sf::Text m_text;
75
76 };
77
78 }
79 }
80
81 #endif
```

7.41. include/objects/object_base.hpp fájlreferencia

```
#include "all.hpp"
```

Adatszerkezetek

- class pftd::Object
- struct pftd::Object::Compare

Névterek

- namespace pftd

7.42. object_base.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6
7 namespace pftd {
8
10 class Object :  public sf::Drawable
11 {
12 public:
13     Object() = default;
14
22     Object(utils::Vec2f position, utils::Vec2f size, int zIndex = 0):
23         zIndex{zIndex}, position{position}, size{size}
24     {}
25
26     Object(Object const&) = default;
27     Object(Object&&) noexcept = default;
28
29     virtual ~Object() = default;
30
37     virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const = 0;
38
40     struct Compare
41     {
42         // bool operator()(Object const& o1, Object const& o2) const
43         // {
44         //     return o1.zIndex > o2.zIndex;
45         // };
46
47         bool operator()(Object const* o1, Object const* o2)const
48 {
49             if(!o1 || !o2) return false;
50             return o1->zIndex > o2->zIndex;
```

```
51             //return this->operator()(*o1, *o2);
52         };
53     };
54
55 protected:
57     int zIndex = 0;
58
60     utils::Vec2f position;
61
63     utils::Vec2f size;
64
65 };
66
67 }
68
69 #endif
```

7.43. include/resources.hpp fájlreferencia

```
#include "SFML/Graphics.hpp"
#include "SFML/Audio.hpp"
#include "objects/gui/image.hpp"
```

Adatszerkezetek

- class pftd::ResourceManager
- struct pftd::LoadError

Névterek

- namespace pftd

7.44. resources.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "SFML/Graphics.hpp"
6 #include "SFML/Audio.hpp"
7 #include "objects/gui/image.hpp"
8
9 namespace pftd {
10
12 class ResourceManager final
13 {
14 public:
15     ResourceManager(ResourceManager const&) = delete;
16     ResourceManager(ResourceManager&&) = delete;
17
21     ~ResourceManager() = default;
22
30     void loadDefaultFont(std::string const& path);
31
40     sf::Texture const& getTexture(std::string const& source);
41
50     sf::SoundBuffer const& getSound(std::string const& source);
51
57     sf::Font const& getDefaultFont()const { return m_defaultFont; }
58
64     static ResourceManager* getInstance() { return m_instance; }
65
73     static ResourceManager* create();
```

```
74
78     static void destroy() { delete m_instance; }
79
80 private:
81     // Az `m_instance` létrehozásához van használva.
82     ResourceManager() = default;
83
85     static ResourceManager* m_instance;
86
88     sf::Font m_defaultFont;
89
91     std::unordered_map<std::string, sf::Texture> m_textures;
92
94     std::unordered_map<std::string, sf::SoundBuffer> m_sounds;
95
96 };
97
99 struct LoadError :  public std::runtime_error
100 {
101     using std::runtime_error::runtime_error;
102 };
103
104 }
105
106 #endif
```

7.45. include/scene.hpp fájlreferencia

```
#include "all.hpp"
#include "SFML/Window/Event.hpp"
#include "objects/object_base.hpp"
```

Adatszerkezetek

- class pftd::Scene

Névterek

- namespace pftd

7.46. scene.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "all.hpp"
6 #include "SFML/Window/Event.hpp"
7 #include "objects/object_base.hpp"
8
9 namespace pftd {
10
12 class Scene
13 {
14     using ObjPtrVec = std::vector<Object*>;
15 public:
17     enum class StateFlag :  uint8_t
18     {
19         NONE = 0,
20         LOAD_STATE = 1 « 0
21     };
22
23     Scene() = default;
24
```

```
25      Scene(Scene const&) = delete;
26      Scene(Scene&&) noexcept = delete;
27
31      virtual ~Scene();
32
38      ObjPtrVec const& getObjects()const { return objects; }
39
45      ObjPtrVec& getObjects() { return objects; }
46
53      void setMusic(std::string const& source, float volume = 100.0f);
54
60      virtual void toggleActive(StateFlag flag = StateFlag::NONE);
61
67      virtual void onEvent(sf::Event const& event) = 0;
68
74      virtual void update(float dt) = 0;
75
76  protected:
78      bool isActive = false;
79
81      ObjPtrVec objects;
82
84      sf::Music* backgroundMusic = nullptr;
85
86  };
87
88  }
89
90  #endif
```

7.47. include/scenes/game.hpp fájlreferencia

```
#include "objects/gui/image.hpp"
#include "objects/gui/button.hpp"
#include "utils/hetero_collection.hpp"
#include "game/level.hpp"
#include "resources.hpp"
#include "scene.hpp"
```

Adatszerkezetek

- class pftd::GameScene
- struct pftd::GameScene::InventoryItem
- struct pftd::GameScene::Inventory

Névterek

- namespace pftd

7.48. game.hpp

Ugrás a fájl dokumentációjához.

```
1  #pragma once
2
3  #ifndef CPORTA
4
5  #include "objects/gui/image.hpp"
6  #include "objects/gui/button.hpp"
7  #include "utils/hetero_collection.hpp"
8  #include "game/level.hpp"
9  #include "resources.hpp"
10 #include "scene.hpp"
```

```
11
12 namespace pftd {
13
15 class GameScene final :  public Scene
16 {
17 public:
19     static constexpr char const* SAVE_FILE_PATH = "res/data/save.dat";
20
22     struct InventoryItem :  public Clickable
23     {
25         gr::Sprite frame;
26
28         gr::Sprite icon;
29
31         Tower* towerToSpawn = nullptr;
32
34         gr::Label priceLabel;
35
36         InventoryItem(Tower* tower, Level * const level, utils::Vec2f const& position, utils::Vec2f
    const& size);
37         ~InventoryItem();
38
39         virtual void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
40     };
41
43     struct Inventory final :  public Object, public utils::Container<InventoryItem>
44     {
46         gr::Sprite background;
47
53         Inventory(std::string const& backgroundImageSrc);
54         ~Inventory() = default;
55
56         void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
57     };
58
59     GameScene();
60     ~GameScene();
61
62     void onEvent(sf::Event const& event) override;
63
64     void update(float dt) override;
65
66     void toggleActive(Scene::StateFlag flag = Scene::StateFlag::NONE) override;
67
73     void startGame();
74
78     void updateScore();
79
83     void updateWealth();
84
85 private:
87     bool m_shouldLoadSaved = false;
88
90     gr::Button* m_saveButt = nullptr;
91
93     gr::Label* m_moneyCounter = nullptr;
94
96     gr::Label* m_scoreCounter = nullptr;
97
99     sf::Sound m_hornSound =
    sf::Sound{ResourceManager::getInstance()->getSound("res/audio/ready_for_battle.mp3")};
100
102     sf::Sound m_gameoverSound =
    sf::Sound{ResourceManager::getInstance()->getSound("res/audio/gameover.mp3")};
103
105     Inventory* m_inventory = nullptr;
106
108     Level* m_level = nullptr;
109
113     void _constructInventory();
114
115 };
116
117 }
118
119 #endif
```

7.49. include/scenes/menu.hpp fájlreferencia

```
#include "objects/gui/button.hpp"
#include "scene.hpp"
```

Adatszerkezetek

- class pftd::MenuScene

Névterek

- namespace pftd

7.50. menu.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #ifndef CPORTA
4
5 #include "objects/gui/button.hpp"
6 #include "scene.hpp"
7
8 namespace pftd {
9
11 class MenuScene final :  public Scene
12 {
13 public:
14     MenuScene();
15     ~MenuScene() = default;
16
17     void onEvent(sf::Event const& event) override;
18
19     void update(float dt) override;
20
21     void toggleActive(Scene::StateFlag flag = Scene::StateFlag::NONE) override;
22
23 private:
25     std::vector<gr::Button*> m_buttons;
26
32     bool _isSaveFileAvailable();
33
34 };
35
36 }
37
38 #endif
```

7.51. include/utils/hetero_collection.hpp fájlreferencia

```
#include "all.hpp"
```

Adatszerkezetek

- class pftd::utils::Container< T, C >

Névterek

- namespace pftd
- namespace pftd::utils

7.52. hetero_collection.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "all.hpp"
4
5 namespace pftd {
6 namespace utils {
7
14 template<typename T, typename C = std::vector<T*»
15 class Container
16 {
17 public:
18     Container() = default;
19     virtual ~Container()
20     {
21         for(auto& elem :  m_container) {
22             delete elem;
23         }
24     }
25
32     /*virtual*/ T* const append(T* elem)
33     {
34         m_container.push_back(elem);
35         return elem;
36     }
37
43     std::size_t size()const { return m_container.size(); }
44
50     C const& getContainer()const { return m_container; }
51
52 private:
54     C m_container;
55
56 };
57
58 }
59 }
```

7.53. include/utils/logger.hpp fájlreferencia

Makródefiníciók

- #define print(x)
- #define where()

7.53.1. Makródefiníciók dokumentációja

7.53.1.1. print

```
#define print(
              x )
```

7.53.1.2. where

```
#define where( )
```

7.54. logger.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 // Zseniálisan egyszerű logger makrókkal.
4
5 #ifdef _PFTD_DEBUG
6
7 #include <iostream>
8
9 #define print(x) std::cout « x « '\n'
10 #define where() print("[CALLED IN] " « __PRETTY_FUNCTION__)
11
12 #else
13
14 #define print(x)
15 #define where()
16
17 #endif
```

7.55. include/utils/parsers.hpp fájlreferencia

```
#include "all.hpp"
```

Adatszerkezetek

- struct pftd::utils::parser::ParseError
- class pftd::utils::parser::Parser
- class pftd::utils::parser::LevelConfigParser
- class pftd::utils::parser::SaveFileParser
- struct pftd::utils::parser::SaveFileParser::StatsInfo
- struct pftd::utils::parser::SaveFileParser::EntityInfo

Névterek

- namespace pftd
- namespace pftd::utils
- namespace pftd::utils::parser

7.56. parsers.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "all.hpp"
4
5 namespace pftd {
6 namespace utils {
7 namespace parser {
8
9 struct ParseError :  std::runtime_error
10 {
11     using std::runtime_error::runtime_error;
12 };
13
14 class Parser
15 {
16 public:
```

```
23      Parser(std::string const& sourceFile, std::string label = "");
24
25      virtual ~Parser();
26
30      void reset();
31
38      bool isLabelValid(bool skip = true);
39
46      void skip(size_t howMany, char until = '\n');
47
53      void skip(char while_);
54
62      template<typename T> T get()
63      {
64          this->_skipWhitespace();
65          while(sourceStream.peek() == commentDenoter) {
66              this->_skipLine();
67              this->_skipWhitespace();
68          }
69
70
71          T got;
72          sourceStream » got;
73          if (sourceStream.fail() || sourceStream.bad()) {
74              throw ParseError{"(" + validLabel + ") Nem sikerült beolvasni a következő tokent!"};
75          }
76
77          return got;
78      }
79
87      char peekAhead();
88
92      virtual void parse() = 0;
93
94  protected:
96      std::ifstream sourceStream;
97
99      std::string validLabel;
100
102      char commentDenoter = '#';
103
104      //virtual bool validate() const = 0;
105
106 private:
110      void _skipLine();
111
115      void _skipWhitespace();
116
117 };
118
123 class LevelConfigParser final :  public Parser
124 {
125 public:
131      LevelConfigParser(std::string const& sourceFile);
132      ~LevelConfigParser() = default;
133
134      void parse() override;
135
142      std::vector<Vec2f> getAttribute(std::string name) const;
143
144 private:
146      std::unordered_map<std::string, std::vector<Vec2f» m_attribs;
147
153      std::string _getAttribute();
154
155 };
156
158 class SaveFileParser final :  public Parser
159 {
160 public:
162      struct StatsInfo
163      {
164          unsigned int score, wealth;
165          int maxHp, hp;
166      };
167
169      enum class EntityType
170      {
171          TOWER = 0U,
172          SEAL,
173          PROJECTILE
174      };
175
177      struct EntityInfo
178      {
180          EntityType entityType;
181
```

```
183         utils::Vec2f position;
184
186         //union {
188             struct {
189                 unsigned int sealID;
190                 float lerpParam;
191                 bool goingBackwards;
192                 unsigned int hp;
193             } seal;
194
196             struct {
197                 unsigned int projID;
198                 utils::Vec2f direction;
199                 float speed;
200             } proj;
201
203             unsigned int towerID;
204         //};
205     };
206
207 //public:
208     SaveFileParser(std::string const& sourceFile);
209     ~SaveFileParser() = default;
210
211     void parse() override;
212
218     StatsInfo const& getStats()const { return m_readStats; };
219
225     std::vector<EntityInfo> const& getEntities()const { return m_entities; };
226
227 private:
229     StatsInfo m_readStats;
230
232     std::vector<EntityInfo> m_entities;
233
237     void _getStats();
238
242     void _getEntity();
243 };
244
245 }
246 }
247 }
```

7.57. include/utils/random_gen.hpp fájlreferencia

```
#include "all.hpp"
```

Adatszerkezetek

- class pftd::utils::Random

Névterek

- namespace pftd
- namespace pftd::utils

7.58. random_gen.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "all.hpp"
4
5 namespace pftd {
```

```
6 namespace utils {
7
9 class Random
10 {
11 public:
12     Random() = delete;
13
15     static std::default_random_engine randomEngine;
16
23     template<typename RandomDist>
24     static auto generate(RandomDist distribution)
25     {
26         return distribution(randomEngine);
27     }
28 };
29
30 }
31 }
```

7.59. include/utils/serializable.hpp fájlreferencia

```
#include "all.hpp"
```

Adatszerkezetek

- class pftd::utils::Serializable

Névterek

- namespace pftd
- namespace pftd::utils

7.60. serializable.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include "all.hpp"
4
5 namespace pftd {
6 namespace utils {
7
9 class Serializable
10 {
11 public:
17     virtual void serialize(std::ostream& out) const = 0;
18
26     //virtual void deserialize(std::istream& in) = 0;
27 };
28
29 }
30 }
```

7.61. include/utils/substitute_types.hpp fájlreferencia

```
#include <cmath>
#include <iostream>
```

Adatszerkezetek

- struct pftd::utils::Vec2< T >

Névterek

- namespace pftd
- namespace pftd::utils

Típusdefiníciók

- using pftd::utils::Vec2i = Vec2< int >
- using pftd::utils::Vec2f = Vec2< float >

7.62. substitute_types.hpp

Ugrás a fájl dokumentációjához.

```
1 #pragma once
2
3 #include <cmath>
4 #include <iostream>
5
6 namespace pftd {
7 namespace utils {
8
10 template<typename T>
11 struct Vec2
12 {
14     T x;
15
17     T y;
18
19     Vec2():
20         x{0}, y{0}
21     {}
22     Vec2(T x, T y):
23         x{x}, y{y}
24     {}
25     Vec2(Vec2 const& right):
26         x{right.x}, y{right.y}
27     {}
28
36     static float distance(Vec2 const& v1, Vec2 const& v2)
37     {
38         return std::sqrt((v2.x-v1.x)*(v2.x-v1.x) + (v2.y-v1.y)*(v2.y-v1.y));
39     }
40
48     Vec2 normalize()const
49 {
50         return *this / Vec2::distance(*this, {0, 0});
51     }
52
53     Vec2 operator+(Vec2 const& right)const
54 {
55         return {this->x + right.x, this->y + right.y};
56     }
57
58     Vec2 operator-(Vec2 const& right)const
59 {
60         return {this->x - right.x, this->y - right.y};
61     }
62
63     Vec2 operator/(T scalar)const
64 {
65         return {this->x / scalar, this->y / scalar};
66     }
67     Vec2 operator*(T scalar)const
68 {
69         return {this->x * scalar, this->y * scalar};
70     }
71     bool operator==(Vec2 other)const
```

```
72 {
73         return this->x == other.x && this->y == other.y;
74     }
75     bool operator!=(Vec2 other)const
76 {
77         return !(this == other);
78     }
79 
80     Vec2& operator=(Vec2 const& right)
81     {
82         if(&right != this) {
83             this->x = right.x;
84             this->y = right.y;
85         }
86         return *this;
87     }
88 
89     friend std::ostream& operator«(std::ostream& out, Vec2 const& vec)
90     {
91         out « vec.x « " " « vec.y;
92         return out;
93     }
94 
95     friend std::istream& operator»(std::istream& in, Vec2& vec)
96     {
97         in » vec.x » vec.y;
98         return in;
99     }
100 };
101 
102 // Az SFML is ilyet csinál.
103 using Vec2i = Vec2<int>;
104 using Vec2f = Vec2<float>;
105 
106 }
107 }
```

Tárgymutató