Házi Feladat

Programozás alapjai 2.

Takács András

1 Feladat:

A feladat egy 2 dimenziós játék, amiben kettő játékos versenyzik azon, hogy a másik játékos tetejére ugorjon ezzel pontot szerezzen. A játékosokat a billentyűzetről lehet irányítani és egy grafikus ablakban jelenik meg. A játék rendelkezik előre elkészített pályákkal, de a felhasználó is készíthet extra pályákat.

2 Feladatspecifikáció:

A feladat egy két fős játék elkészítése, melyben a két játékos egy-egy színes négyzetet irányít és a játék célja az, hogy a játékos a másik játékos által irányított négyzet tetejére ugorjon, ezzel pontot szerezve.

Amikor egy játékos pontot szerezt azt a játék egy képernyő közepére, a játékos színével megegyező, kiírt szöveggel jelzi, kiírja a játékos pontjainak a számát és egy rövid idő után visszaállítja a két játékost az eredeti helyükre. Amennyiben egyik játékos eléri a pályához tartozó győzelemhez szükséges pontszámot a program ezt szintén kiírással jelzi és egy új pályát tölt be nullázott pontokkal.

A két játékos egyazon számítógépről játszik, egy billentyűzeten osztozkodva, a kék játékos az A és D billentyűket használhatja horizontális mozgáshoz, a space-t ugráshoz és az S-et egy azonnali lefele gyorsuláshoz, míg a piros játékos esetében ezek megfelelője a két oldal nyíl a bal shift és a lefele nyíl.

A pálya, a játékosok és a szövegek grafikusan az SDL3 könyvtár segítségével vannak megjelenítve és a billentyűzet kezelésében is ez a könyvtár fog segíteni. A két játékos fizika szimulációjához a kód számontartja a gyorsulásukat és sebességüket, emellett biztosítja a helyes ütközéseket a másik játékossal és a

pályaelemekkel. Ha egy játékos kimegy a pálya oldalán akkor a pálya másik oldalán bukkan fel, ha kiesik a pálya alján akkor a pálya tetejére kerül. Mivel a két játékos és a pályák is téglalapok, ezért az ütközések detektáláshoz elég azt ellenőrizni, hogy tetszőleges két tengelyhez képest el nem forgatott téglalap metszi-e egymást. Egyes pályaelemekről az ütközéskor visszapattannak a játékosok, más pályaelemek megölik a játékost.

A program a grafikus megjelenítés mellett egy parancssorban rögzíti a fontosabb eseményeket/fellépő hibákat. Egy ilyen lehetséges hiba például egy hibás pályafájl, ebben az esetben a program a terminálban jelzi, hogy melyik fájl, mely sorában keresendő a hiba. A pályákat „.gamemap” kiterjesztésű fájlokból az induláskor tölti be a program, minden a játékkal egy mappában lévő ilyen fájlt betöltésével megpróbálkozik. A formátum pontosabb specifikációja majd a részletesebb tervben lesz megtalálható.

3 Pontosított specifikáció:

A pályafájlok formátuma az alábbiaknak fog megfelelni:

Pálya magassága(double, a tényleges magasság ennek a kétszeres)

Pálya háttérszíne(3 x byte)

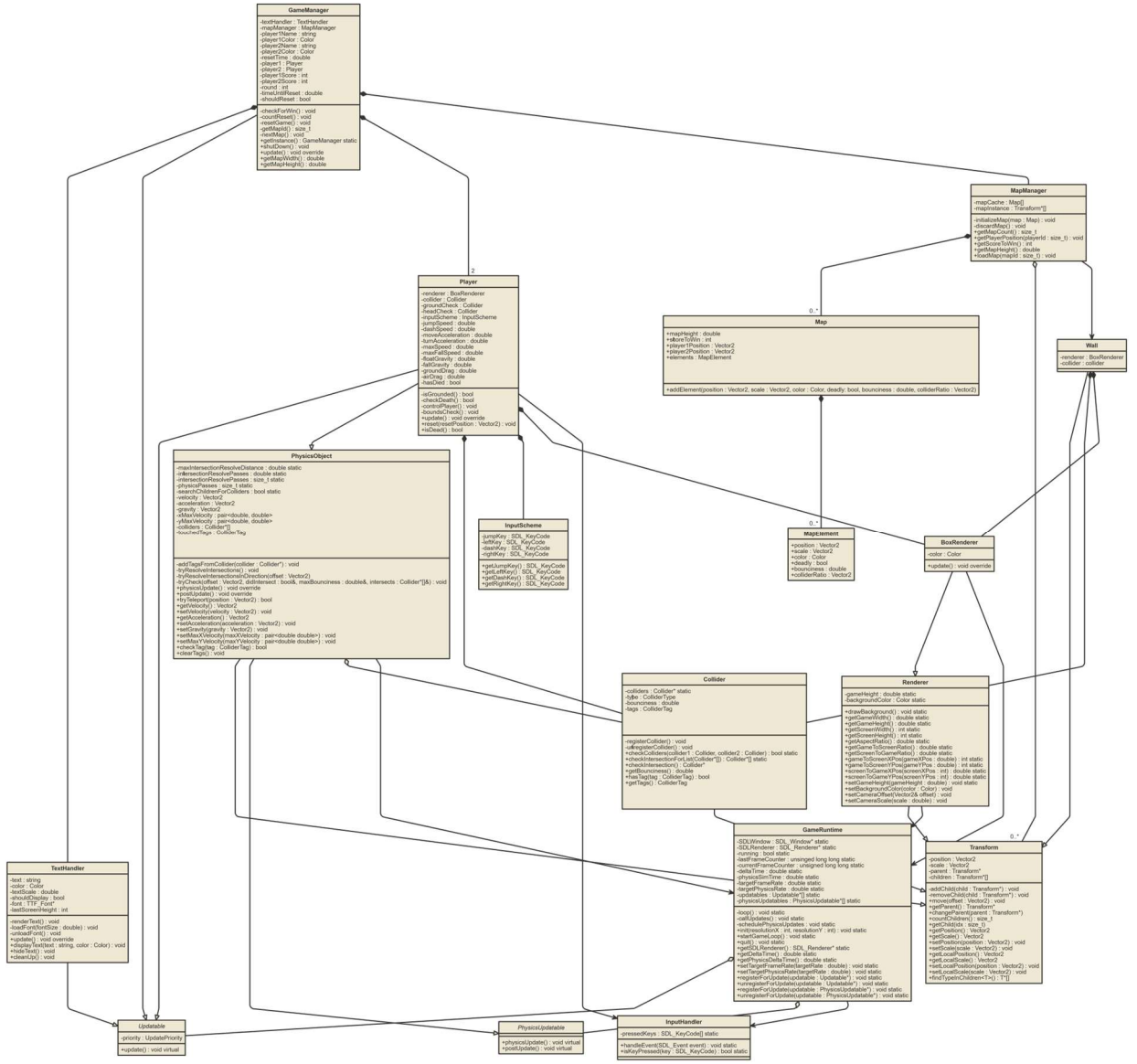
Első játékos kezdőkoordinátái(float float)

Második játékos kezdőkoordinátái(float float)

N db falelem:(pozíció: float float, méret: float float, szín: 3 x byte, visszapattanás mértéke: float, fizikai kiterjedés aránya: float, float)

4 Terv:

Az UML diagramot a https://www.nomnoml.com segítségével hoztam létre, részletezi a projekthez tartozó osztályokat és a közöttük lévő kapcsolatokat.



(Bele kell zoomolni hogy olvasható legyen)

5 Fontosabb algoritmusok:

5.1 Kettő el nem forgatott téglalap metszi-e egymást

Ezt az algoritmust használom arra, hogy ellenőrizzem, hogy két játékbeli objektum ütközik-e.

Tudjuk: center1, oldalhosszak1, center2, oldalhosszak2

//ha az első téglalap alja magasabban van a második téglalap tetejénél, így biztos nem metszgetik egymást

IF center1.y - oldalhosszak1.y/2 > center2.y + oldalhosszak2.y/2:

    RETURN HAMIS

//ha az első téglalap teteje alacsonyabban van a második téglalap aljánál, így biztos nem metszhetik egymást

IF center1.y + oldalhosszak1.y/2 <center2.y – oldalhosszak2.y/2:

    RETURN HAMIS

//ha az első téglalap bal széle jobbra van a második téglalap jobb oldalánál, így biztos nem metszhetik egymást

IF center1.x - oldalhosszak1.x/2 > center2.x + oldalhosszak2.x/2:

    RETURN HAMIS

//ha az első téglalap jobb széle balra van a második téglalap bal oldalánál, így biztos nem metszhetik egymást

IF center1.x + oldalhosszak1.x/2 <center2.x - oldalhosszak2.x/2:

RETURN HAMIS

//különben kizárásos alapon a kettő téglalap metszi egymást

RETURN IGAZ

5.2 Fizikai objektum szimulációja:

A fizikai objektum rendelkezik sebesség- és gyorsulásvektorral, minden szimulációs lépés elején a sebességhez hozzáadja a gyorsulást szorozva a szimulációk gyakoriságának reciprokával, hogy konzisztens viselkedést biztosítson a képkockák számától függetlenül. Mivel a gyorsulás négyzetes kapcsolatban van a megtett úttal a szimulációs rátát konzisztensen tartjuk, hogy az ebből adódó hibákat elkerüljük.

Kiszámoljuk a lehetséges mozgatás maximumát: ez sebességvektor szorozva a szimulációk gyakoriságának reciprokával.

Felvesszük a próbamozgatás nagyságát: először ½, aztán 1/4 ... ½^n

Először a x tengely mentén mozgatjuk a próbamozgatás x komponensével, ha nem ütközik semmi mással maradhat, ha nem visszaléptetjük a jelenlegi próbamozgatással

Ezt az y tengely mentén is megtesszük

Ezután felezzük a próbamozgatást, amíg el nem érünk ½^n-ig

Amennyiben a feljebb lévő algoritmus során ütköztünk valamivel, nullázzuk a sebességet és gyorsulást, amennyiben volt az ütközések között visszapattanós tulajdonságú objektum, akkor ennek arányában megfordítjuk a sebességet.

Az algoritmus garantálja, hogy az objektumok relatíve jól hozzásimulnak egymáshoz, n számától a lebegőpontos számítás hibáitól függően.

5.3 Két fizikai objektum metszésének a feloldása:

Bizonyos esetekben (leggyakrabban egy pálya betöltésekor) előfordulhat, hogy két játékos vagy egy játékos és egy pályaelem, ugyanazt a pozíciót veszik fel, a fenti algoritmus segítségével az alábbi helyzet csak akkor oldódhatna fel ha a játékos(ok) egy képkockán belül tudnak annyit mozogni, hogy a helyzet megszűnjön. Ez magas precizitással csak ritkán fordul elő, maximum csak akkor a két objektum nincs nagyon mélyen egymásba ágyazódva.

Az ilyen helyzetek feloldásának érdekében minden fizikai objektum a szimulációs lépése elején ellenőrzi, hogy a fenti eset fenn áll-e és amennyiben fennáll az alábbi módon kezeli:

Minden kardináls(fel, le, jobbra, balra): Megpróbálja végrehajtani az álábbi algoritmus, amennyiben az adott irány sikeres kilép, ha nem visszállítja és a következő iránnyal is megpróbálja:

Az objektumot elmozgatja a maximum feloldási távolsággal az adott irányba, amennyiben a metszés megszűnt sikeresnek találja, ha nem nem. Az objektumot visszamozgatja az előző távolság felével és ellenőrzi, hogy így már jó-e, ha jó eltárolja a távolságot és feljegyzi, hogy jó, ha nem nem. Ezt a bináris keresés jellegű szűkítést megteszi n db alkalommal és folyamatosan számontartja, hogy volt-e olyan távolság, ami fel tudja oldani a metszést és hogy mi volt a legkisebb ilyen távolság.

Megjegyzés: A fentebbi algoritmus célja főként az, hogy hibásan megtervezett pályákon (ahol a játékos beragadna a falba) se haljon meg a játék és hogy a váratlan kisebb mozgási hibákat kezelni tudja. Az algoritmus, sok triviális szemmel megoldható esetet nem fed le, viszont segíteni tud a tesztelés során megtapasztalt valós helyzetekkel.

6 Pálya fájlformátum specifikáció:

A pályafájlokat a játék az elinduláskor beolvassa, a benne talált értékeket Map adatstruktúrákban tárolja melyek, pályaspecifikus adatokat tartalmaznak; pl.: háttérszín, pálya magassága, míg az egyes elemek adatait MapElement adatstruktúrákban tárolja. A pályákat egy MapCache nevű Map-ekből álló listában tárolja és pályaváltáskor a kiválasztott pályát ez alapján inicializálja.

A Map adatstruktúra felépítése:

{

Pálya magassága: Lebegőpontos

Háttérszín: Szín

Nyeréshez szükséges pontszám: Egész

1. Játékos kezdőpozíciója: Lebegőpontos vektor
2. Játékos kezdőpozíciója: Lebegőpontos vektor

Pályaelemek: MapElementek listája

}

A MapElement adatstruktúra felépítése

{

Pozíció: Lebegőpontos vektor

Méret: Lebegőpontos vektor

Szín: Szín

Halálosság: Logikai

Visszapattanási együttható: Lebegőpontos

Collider arány: Lebegőpontos vektor(skalárokat tartalmaz)

}

Egy .gamemap fájl felépítése:

(Az elemek whitespacekkel vannak szeparálva, a konkrét fájlokban sortöréseket is használok, de ezt csak az olvashatóság kedvéért, a sortörés nélküli fájlok is validnak számítanak a program által, habár személy szerint én ellenzem őket.)

#INT/PÁLYAMAGASSÁG#

#INT/HÁTTÉRSZÍN_R# #INT/HÁTTÉRSZÍN_G#
#INT/HÁTTÉRSZÍN_B#

#INT/NYERÉSI_PONTSZÁM#

#FLOAT/JÁTÉKOS_1_POZ_X# #FLOAT/JÁTÉKOS_1_POZ_Y#

#FLOAT/JÁTÉKOS_2_POZ_X# #FLOAT/JÁTÉKOS_2_POZ_Y#

(itt következik tetszőleges mennyiségű pályaelem)

#FLOAT/ELEM_N_POZ_X# #FLOAT/ELEM_N_POZ_Y#
#FLOAT/ELEM_N_MÉRET_X# #FLOAT/ELEM_N_MÉRET_Y#
#INT/ELEM_N_SZÍN_R# #INT/ELEM_N_SZÍN_G#
#INT/ELEM_N_SZÍN_B# #BOOL/HALÁLOS_E#
#FLOAT/VISSZA_PATTANÁS# #FLOAT/COLLIDER_X#
#FLOAT/COLLIDER_Y#

(minden pályaelem értelemszerűen egy külön MapElement-be kerül, amiknek a listája a Map-hez tartozik)

7 Fordítás/futtatás:

A projektet g++-al fordítottam, CPORTA flaggel nem igényli az SDL3 és az egyéb szükséges fájlok jelenlétét, ekkor csak a két tesztpályafájl kell(más pályák nem is lehetnek, mert akkor nem fut le a teszt). Ekkor csak a tesztek és a teszteléshez szükséges részek fordulnak le. A játék így nem lesz játszható.

Ha CPORTA flag nélkül fordítunk, akkor egy játszható játékos kapunk, de ehhez szükséges:

- SDL3 és SDL3_ttf DLL-ek rendelkezésre állása és az -lSDL3 -lSDL3_ttf flagekkel való fordítás
- Játszható pályák a mappában(ezek lehetnek tesztpályák is)
- Egy „gamefont.ttf” betűtípus szintén a mappában legyen, hogy legyen mivel megjeleníteni a játékban lévő szövegeket

8 Megvalósítás:

A program megvalósítása nem különbözik jelentősen a fenti diagramban megadottól, a fő változás annyi, hogy bekerül egy teszt.cpp és egy teszt.h is, ezek a teszteléssel segítenek, valamint a CPORTA flag a program egyes részeit letiltja/engedélyezi.

A CPORTA-s verzióban dirent.h-t használok filesystem helyett mivel a filesystem-et a JPORTA nem fogadja el, annak ellenére, hogy elvileg CPP17-el fordít.

9 Tesztelés:

A program a tesztjei a kb 1500 soros teszt.cpp file-ban találhatóak. A tesztek kiterjednek a:

- Vector2 adatstruktúra megvalósítására
- A Transform osztályra és ennek helyes viselkedésére (megfelelő lineáris transzformációk)
- A Collider osztályra és ennek helyes viselkedésére (megfelelő metszésdetektálás)
- A PhysicsObject osztályra és ennek helyes viselkedésére(egy módosított GameHandler segítségével konkrét esetek helyes szimulációja)
- A pályák betöltésére és hogy ezek megegyeznek a tesztesetben definiált értékekkel

10 Dokumentáció:

Platformer Game

Generated by Doxygen 1.13.2

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 gtest_lite Namespace Reference

gtest_lite: a keretrendszer függvényinek és objektumainak névtere

Classes

- class ostreamRedir
- struct Test

Functions

- template<typename T1, typename T2>
  std::ostream & **EXPECT_** (T1 exp, T2 act, bool(∗pred)(T1, T1), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  általános sablon a várt értékhez.
- template<typename T1, typename T2>
  std::ostream & **EXPECT_** (T1 ∗exp, T2 ∗act, bool(∗pred)(T1 ∗, T1 ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  pointerre specializált sablon a várt értékhez.
- std::ostream & EXPECTSTR (const char ∗exp, const char ∗act, bool(∗pred)(const char ∗, const char ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")
- template<typename T>
  bool eq (T a, T b)
- bool **eqstr** (const char ∗a, const char ∗b)
- bool **eqstrcase** (const char ∗a, const char ∗b)
- template<typename T>
  bool **ne** (T a, T b)
- bool **nestr** (const char ∗a, const char ∗b)
- template<typename T>
  bool **le** (T a, T b)
- template<typename T>
  bool **lt** (T a, T b)
- template<typename T>
  bool **ge** (T a, T b)
- template<typename T>
  bool **gt** (T a, T b)
- template<typename T>
  bool almostEQ (T a, T b)

5.1.1 Detailed Description

gtest_lite: a keretrendszer függvényinek és objektumainak névtere

5.1.2 Function Documentation

5.1.2.1 almostEQ()

```
template<typename T>
bool gtest_lite::almostEQ (
            T a,
            T b)
```

Segédsablon valós számok összehasonlításához Nem bombabiztos, de nekünk most jó lesz Elméleti hátér: http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm

5.1.2.2 eq()

```
template<typename T>
bool gtest_lite::eq (
            T a,
            T b)
```

segéd sablonok a relációkhoz. azért nem STL (algorithm), mert csak a függvény lehet, hogy menjen a deduckció

5.1.2.3 EXPECTSTR()

```
std::ostream & gtest_lite::EXPECTSTR (
            const char * exp,
            const char * act,
            bool(* pred )(const char *, const char *),
            const char * file,
            int line,
            const char * expr,
            const char * lhs = "elvart",
            const char * rhs = "aktual")  [inline]
```

stringek összehasonlításához. azért nem spec. mert a sima EQ-ra másként kell működnie.

Chapter 6

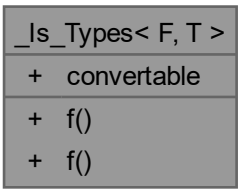
Class Documentation

6.1 _Is_Types< F, T > Struct Template Reference

Segédsablon típuskonverzió futás közbeni ellenőrzésere.

`#include <gtest_lite.h>`

Collaboration diagram for _Is_Types< F, T >:



Static Public Member Functions

- template<typename D>
  static char(& **f** (D))[1]
- template<typename D>
  static char(& **f** (...))[2]

Static Public Attributes

- static bool const **convertable** = sizeof(f<T>(F())) == 1

6.1.1 Detailed Description

template<typename F, typename T>
struct _Is_Types< F, T >

Segédsablon típuskonverzió futás közbeni ellenőrzésere.

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

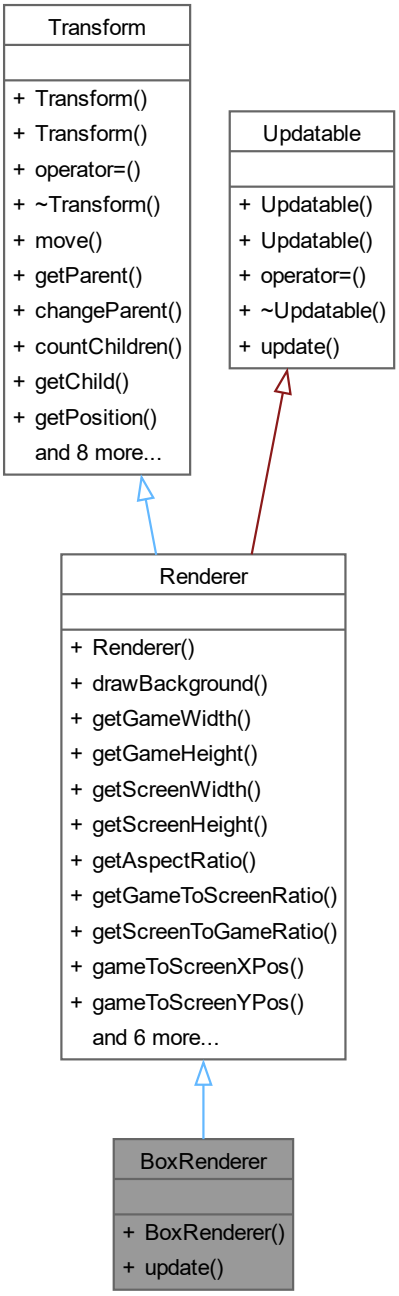
- gtest_lite.h

6.2 BoxRenderer Class Reference

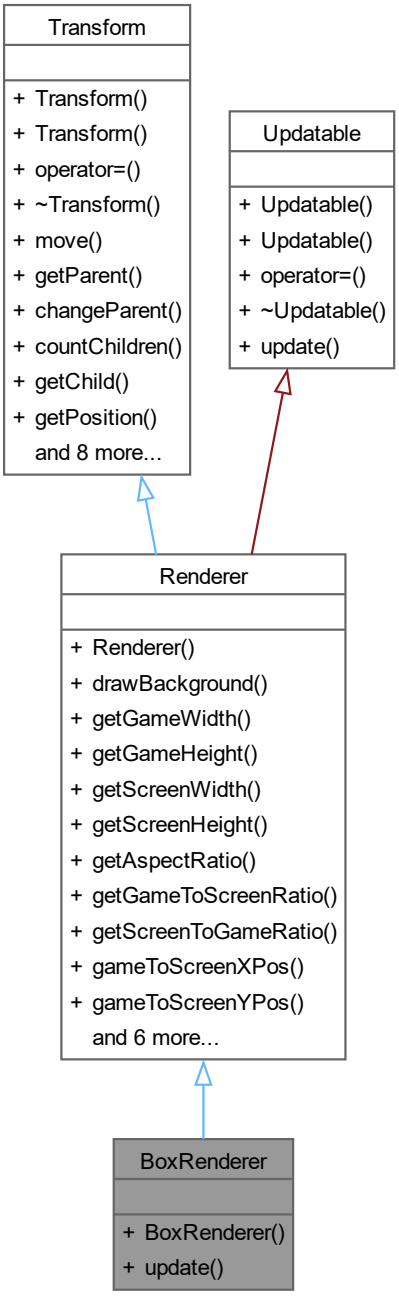
Egy egyszínű téglalapot kirajzoló objektum.

`#include <boxrenderer.h>`

Inheritance diagram for BoxRenderer:



Collaboration diagram for BoxRenderer:



Public Member Functions

- BoxRenderer (const Transform &transform, const Color &color, const UpdatePriority priority)

  Létrehoz egy BoxRenderer objektumot.
- void update () override

  Végrehatja a téglalap frissítését.

Public Member Functions inherited from Renderer

- Renderer (const Transform &transform, UpdatePriority priority)

  Létrehoz egy `Renderer` objektumot.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.
- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

Additional Inherited Members

Static Public Member Functions inherited from Renderer

- static void drawBackground ()

  Kirajzolja a játék hátterét.
- static double getGameWidth ()

  Visszaadja a játék világának szélességét.
- static double getGameHeight ()

  Visszaadja a játék világának magasságát.
- static int getScreenWidth ()

  Visszaadja a képernyő szélességét.
- static int getScreenHeight ()

  Visszaadja a képernyő magasságát.
- static double getAspectRatio ()

  Visszaadja a képernyő képarányát.
- static double getGameToScreenRatio ()

  Visszaadja a játék világának és a képernyőnek az arányát.
- static double getScreenToGameRatio ()

  Visszaadja a képernyő és a játék világának arányát.
- static int gameToScreenXPos (const double gameXPos)

  Átváltja a játék világának X koordinátáját képernyő koordinátára.
- static int gameToScreenYPos (const double gameYPos)

  Átváltja a játék világának Y koordinátáját képernyő koordinátára.
- static double screenToGameXPos (const int screenXPos)

  Átváltja a képernyő X koordinátáját a játék világának koordinátájára.
- static double screenToGameYPos (const int screenYPos)

  Átváltja a képernyő Y koordinátáját a játék világának koordinátájára.
- static void setGameHeight (const double gameHeight)

  Beállítja a játék világának magasságát.
- static void setBackgroundColor (const Color color)

  Beállítja a játék háttérszínét.
- static void setCameraOffset (const Vector2 &offset)

  Beállítja a játék kamerájának eltolását.
- static void setCameraScale (const double scale)

  Visszaadja a játék kamerájának méretét.

6.2.1 Detailed Description

Egy egyszínű téglalapot kirajzoló objektum.

A `BoxRenderer` osztály felelős egy téglalap alakú objektum kirajzolásáért a megadott színnel és mérettel. Az osztály a `Renderer` osztályból származik, és annak frissítési mechanizmusát használja a téglalap megjelenítéséhez.

Használható statikus vagy dinamikus objektumok kirajzolására a játék világában.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BoxRenderer()

```
BoxRenderer::BoxRenderer (
            const Transform & transform,
            const Color & color,
            const UpdatePriority priority)
```

Létrehoz egy BoxRenderer objektumot.

Létrehoz egy téglalapot kirajzoló objektumot, amely a megadott helyzetben, méretben és színnel jelenik meg. Az objektum frissítési prioritása meghatározza, hogy milyen sorrendben kerül frissítésre más renderelő objektumokhoz képest.

Parameters

transform	Az objektum helyzete és mérete.
color	Az objektum színe.
priority	Az objektum update prioritása.

6.2.3 Member Function Documentation

6.2.3.1 update()

```
void BoxRenderer::update ()  [override], [virtual]
```

Végrehatja a téglalap frissítését.

A metódus kiszámítja a téglalap képernyőn megjelenő pozícióját és méretét a `Transform` adatai alapján, majd a megadott színnel kirajzolja azt az SDL renderelő segítségével.

Implements Updatable.

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

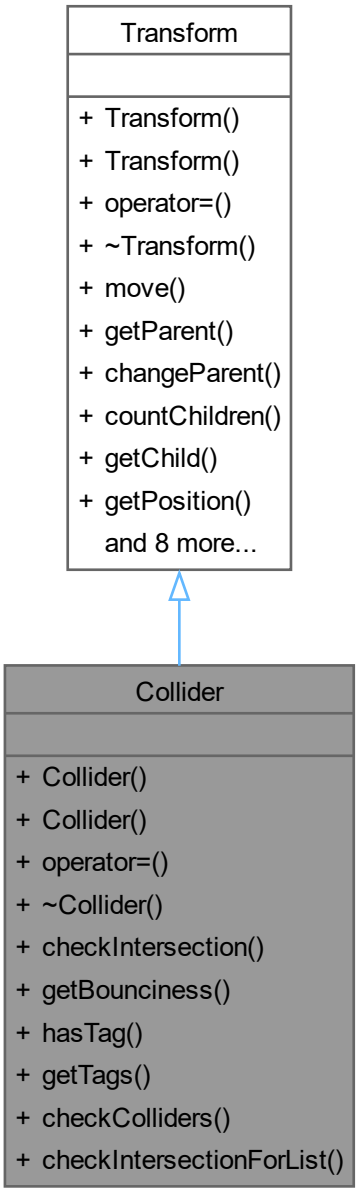
- boxrenderer.h
- boxrenderer.cpp

6.3 Collider Class Reference

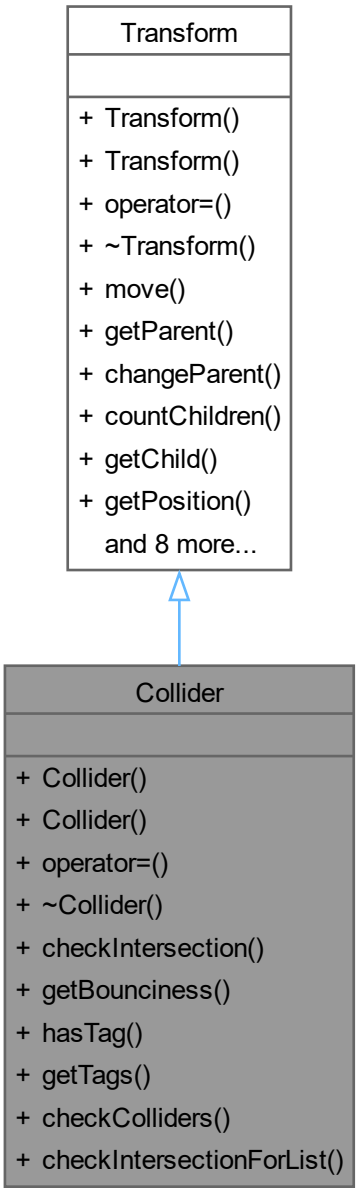
Meghatározza egy objektum fizikai határait.

`#include <collider.h>`

Inheritance diagram for Collider:



Collaboration diagram for Collider:



Public Member Functions

- Collider (const Transform &transform, const ColliderType type=ColliderType::INTERACTIVE, const double bounciness=0, const std::vector< ColliderTag > &tags={})

  Létrehoz egy Collider objektumot.
- Collider (const Collider &collider)

  Másoló konstruktor.
- Collider & operator= (const Collider &collider)

  Értékadás operátor.

- ~Collider ()

  Megsemmisíti a Collider objektumot.
- std::vector< Collider * > checkIntersection () const

  Ellenőrzi, hogy a collider metszi-e a statikus collider listában szereplő collidereket.
- double getBounciness () const

  Visszaadja a collider visszapattanási együtthatóját.
- bool hasTag (const ColliderTag tag) const

  Ellenőrzi, hogy egy adott címke megtalálható-e a collider címkéi között.
- std::vector< ColliderTag > getTags () const

  Visszaadja a collider címkéit.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.
- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

Static Public Member Functions

- static bool checkColliders (const Collider &collider1, const Collider &collider2)

  Ellenőrzi, hogy két collider metszi-e egymást.
- static std::vector< Collider * > checkIntersectionForList (const std::vector< Collider * > &collidersToCheck)

  Ellenőrzi, hogy a megadott colliderek listájában lévő colliderek közül bármelyik metszi-e a statikus collider listában szereplő collidereket.

6.3.1 Detailed Description

Meghatározza egy objektum fizikai határait.

A `Collider` osztály felelős az objektum fizikai határainak meghatározásáért és az ütközések detektálásáért. Az osztály a `Transform` osztályból származik, így rendelkezik pozícióval és mérettel, amelyeket az ütközésdetektálás során használ.

A `Collider` osztály statikus listát tart fenn az összes colliderről, amely lehetővé teszi az ütközések globális ellenőrzését.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Collider() [1/2]

```
Collider::Collider (
            const Transform & transform,
            const ColliderType type = ColliderType::INTERACTIVE,
            const double bounciness = 0,
            const std::vector< ColliderTag > & tags = {})
```

Létrehoz egy Collider objektumot.

Létrehoz egy Collider objektumot a megadott pozícióval, mérettel és tulajdonságokkal. Az objektum automatikusan hozzáadódik a statikus collider listához, így részt vesz az ütközésdetektálásban.

Parameters

transform	Az objektum pozíciója és mérete a játék világában.
type	A collider típusa (interaktív vagy passzív).
bounciness	Az ütközéskor visszapattanás mértéke (0 = nincs visszapattanás, 1 = teljes visszapattanás).
tags	A collider címkék, amelyek extra tulajdonságokat rendelnek a colliderhez.

6.3.2.2 Collider() [2/2]

```
Collider::Collider (
            const Collider & collider)
```

Másoló konstruktor.

Létrehoz egy új Collider objektumot egy meglévő Collider alapján. Az új objektum automatikusan hozzáadódik a statikus collider listához, így részt vesz az ütközésdetektálásban.

Parameters

collider	A másolandó Collider objektum.

6.3.2.3 ~Collider()

```
Collider::~Collider ()
```

Megsemmisíti a Collider objektumot.

Megsemmisíti a Collider objektumot és eltávolítja a satikus collider listából.

6.3.3 Member Function Documentation

6.3.3.1 checkColliders()

```
bool Collider::checkColliders (
            const Collider & collider1,
            const Collider & collider2)  [static]
```

Ellenőrzi, hogy két collider metszi-e egymást.

Ellenőrzi, hogy két collider metszi-e egymást.

Parameters

collider1	Az egyik collider.
collider2	A másik collider.

Returns

true, ha a két collider metszi egymást, egyébként false.

6.3.3.2 checkIntersection()

```
std::vector< Collider * > Collider::checkIntersection () const
```

Ellenőrzi, hogy a collider metszi-e a statikus collider listában szereplő collidereket.

Az ütközésdetektálás során ellenőrzi, hogy az aktuális collider átfedi-e bármelyik collidert a statikus collider listában.

Returns

Azok a colliderek, amelyek metszik az aktuális collidert.

6.3.3.3 checkIntersectionForList()

```
std::vector< Collider * > Collider::checkIntersectionForList (
            const std::vector< Collider * > & collidersToCheck)  [static]
```

Ellenőrzi, hogy a megadott colliderek listájában lévő colliderek közül bármelyik metszi-e a statikus collider listában szereplő collidereket.

Az ütközésdetektálás során minden megadott colliderhez megkeresi azokat a collidereket, amelyekkel ütköznek, és ezeket egy halmazba gyűjti, hogy az eredmény egyedi legyen.

Parameters

collidersToCheck	A colliderek listája, amelyeket ellenőrizni kell.

Returns

Azok a colliderek, amelyek metszik a vizsgált collidereket.

6.3.3.4 getBounciness()

```
double Collider::getBounciness () const
```

Visszaadja a collider visszapattanási együtthatóját.

A visszapattanási együttható határozza meg, hogy az ütközés során az objektum mennyire "rugalmas":

- 0: Nincs visszapattanás
- 1: Teljes visszapattanás

Returns

A visszapattanási együttható értéke.

6.3.3.5 getTags()

```
std::vector< ColliderTag > Collider::getTags () const
```

Visszaadja a collider címkéit.

A címkék extra tulajdonságokat rendelnek a colliderhez.

Returns

A collider címkéinek listája.

6.3.3.6 hasTag()

```
bool Collider::hasTag (
            const ColliderTag tag) const
```

Ellenőrzi, hogy egy adott címke megtalálható-e a collider címkéi között.

A címkék extra tulajdonságokat rendelnek a colliderhez.

Parameters

tag	A keresett címke.

6.3.3.7 operator=()

```
Collider & Collider::operator= (
            const Collider & collider)
```

Értékadás operátor.

Egy meglévő Collider objektum adatait másolja egy másik Collider objektumba. Az értékadás nem módosítja a statikus collider listát.

Parameters

collider	A másolandó Collider objektum.

Returns

Az aktuális Collider objektum referenciája.

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

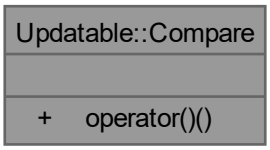
- collider.h
- collider.cpp

6.4 Updatable::Compare Class Reference

Két updatelhető objektum prioritásának összehasonlító osztálya.

```
#include <core.h>
```

Collaboration diagram for Updatable::Compare:



Public Member Functions

- bool **operator()** (const Updatable *updatable1, const Updatable *updatable2) const

6.4.1 Detailed Description

Két updatelhető objektum prioritásának összehasonlító osztálya.

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

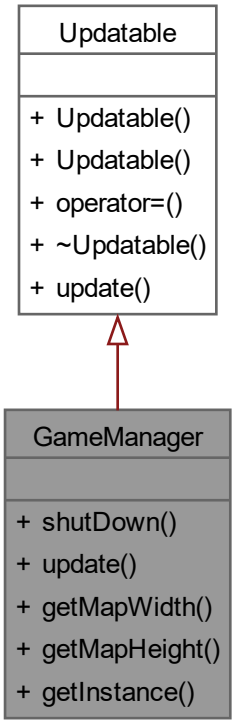
- core.h
- core.cpp

6.5 GameManager Class Reference

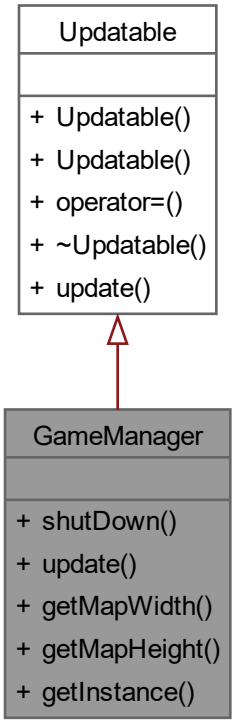
A játékmenetet kezelő osztály.

`#include <gamemanager.h>`

Inheritance diagram for GameManager:



Collaboration diagram for GameManager:



Public Member Functions

- void shutDown ()

  A `GameManager` osztály példánya erőforrásainak felszabadítása.
- void update () override

  A játékmenet frissítése.
- double getMapWidth () const

  Visszaadja a jelenleg betöltött pálya szélességét.
- double getMapHeight () const

  Visszaadja a jelenleg betöltött pálya magasságát.

Static Public Member Functions

- static GameManager & getInstance ()

  Singleton minta megvalósítása.

6.5.1 Detailed Description

A játékmenetet kezelő osztály.

A `GameManager` osztály felelős a játékmenet irányításáért, beleértve a játékosok pontszámának nyilvántartását, a körök kezelését, a pályák váltását és a győzelmi feltételek ellenőrzését. Az osztály kezeli a játékosok állapotát (például halálukat és újraéledésüket), valamint biztosítja a játék folyamatos működését, beleértve a pályák betöltését és a szöveges visszajelzések megjelenítését.

6.5.2 Member Function Documentation

6.5.2.1 getInstance()

```
GameManager & GameManager::getInstance ()  [static]
```

Singleton minta megvalósítása.

A metódus biztosítja, hogy csak egy példány létezzen a GameManager osztályból. Ha a példány még nem létezik, létrehozza azt.

Returns

A GameManager osztály egyetlen példánya.

6.5.2.2 getMapHeight()

```
double GameManager::getMapHeight () const
```

Visszaadja a jelenleg betöltött pálya magasságát.

Returns

A pálya magassága.

6.5.2.3 getMapWidth()

```
double GameManager::getMapWidth () const
```

Visszaadja a jelenleg betöltött pálya szélességét.

Returns

A pálya szélessége.

6.5.2.4 shutDown()

```
void GameManager::shutDown ()
```

A GameManager osztály példánya erőforrásainak felszabadítása.

A metódus biztosítja, hogy a GameManager osztály példánya és az összes kapcsolódó erőforrás felszabaduljon, amikor a játék befejeződik

6.5.2.5 update()

```
void GameManager::update ()  [override], [virtual]
```

A játékmenet frissítése.

A metódus a játék főciklusában kerül meghívásra. Ellenőrzi a játékosok halálát, szükség esetén növeli a győztes pontszámát, és elindítja az újraindítási folyamatot. Ha a győzelemhez szükséges pontszámot elérte valamelyik játékos, a játék új pályára vált.

Implements Updatable.

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

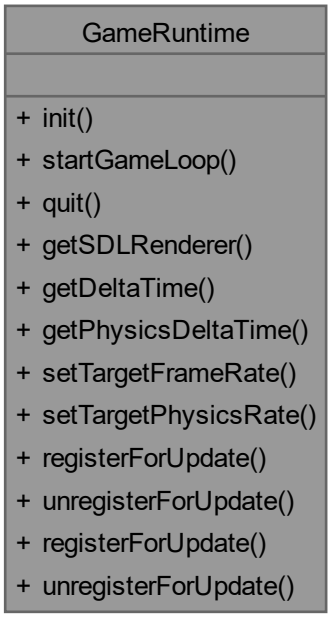
- gamemanager.h
- gamemanager.cpp

6.6 GameRuntime Class Reference

A játék futását kezelő osztály.

```
#include <core.h>
```

Collaboration diagram for GameRuntime:



Static Public Member Functions

- static bool init (const int resolutionX, const int resolutionY)

  A játék futásának inicializálása.
- static void startGameLoop ()

  A játék főciklusának elindítása.
- static void quit ()

  A játék futásának befejezése.
- static SDL_Renderer * getSDLRenderer ()

  Az SDL renderer lekérdezése.
- static double getDeltaTime ()

  A játék lugtóbbi képkockájához kirajzolásához szükséges idő lekérdezése.
- static double getPhysicsDeltaTime ()

  A fizikai szimulációk között eltelt idő lekérdezése.
- static void setTargetFrameRate (const double targetRate)
- static void setTargetPhysicsRate (const double targetRate)
- static void registerForUpdate (Updatable *const updatable)

  Az updatelhető objektum regisztrálása az updateléshez.
- static void unregisterForUpdate (Updatable *const updatable)

  Az updatelhető objektum eltávolítása az updatelésből.
- static void registerForUpdate (PhysicsUpdatable *const updatable)

  A fizikai frissítést végző objektum regisztrálása.
- static void unregisterForUpdate (PhysicsUpdatable *const updatable)

  A fizikai frissítést végző objektum eltávolítása.

6.6.1 Detailed Description

A játék futását kezelő osztály.

A `GameRuntime` osztály felelős a játék futásának kezeléséért. Az osztály inicializálja a játék futásához szükséges komponenseket, például az SDL rendszert, az ablakot és a renderelőt. A főciklus során frissíti az `Updatable` és `PhysicsUpdatable` objektumokat, kezeli a felhasználói eseményeket, és biztosítja a megjelenítést.

Az osztály statikus metódusokon keresztül kezeli a játék futását, például az inicializálást, a főciklus indítását és a játék leállítását.

6.6.2 Member Function Documentation

6.6.2.1 getDeltaTime()

```
double GameRuntime::getDeltaTime ()  [static]
```

A játék lugtóbbi képkockájához kirajzolásához szükséges idő lekérdezése.

Returns

A játék legutóbbi képkockájához kirajzolásához szükséges idő.

6.6.2.2 getPhysicsDeltaTime()

```
double GameRuntime::getPhysicsDeltaTime ()  [static]
```

A fizikai szimulációk között eltelt idő lekérdezése.

Returns

A fizikai szimulációk között eltelt idő.

6.6.2.3 getSDLRenderer()

```
SDL_Renderer * GameRuntime::getSDLRenderer ()  [static]
```

Az SDL renderer lekérdezése.

Returns

Az SDL renderer.

6.6.2.4 init()

```
bool GameRuntime::init (
            const int resolutionX,
            const int resolutionY)  [static]
```

A játék futásának inicializálása.

Az inicializálás során inicializálásra kerül az SDL rendszer, a játék ablaka és a renderelő. Ha bármelyik komponens inicializálása sikertelen, a metódus hamis értékkel tér vissza.

Parameters

resolutionX	Az ablak szélessége pixelben.
resolutionY	Az ablak magassága pixelben.

Returns

true, ha az inicializáció sikeres, különben false.

6.6.2.5 quit()

```
void GameRuntime::quit ()  [static]
```

A játék futásának befejezése.

A játék futásának leállítása során felszabadításra kerülnek a játék futásához szükséges erőforrások, például az SDL ablak és a renderelő.

6.6.2.6 registerForUpdate() [1/2]

```
void GameRuntime::registerForUpdate (
            PhysicsUpdatable *const updatable)  [static]
```

A fizikai frissítést végző objektum regisztrálása.

Parameters

updatable	A fizikai frissítést végző objektum.

6.6.2.7 registerForUpdate() [2/2]

```
void GameRuntime::registerForUpdate (
            Updatable *const updatable)  [static]
```

Az updatelhető objektum regisztrálása az updateléshez.

Parameters

updatable	Az updatelhető objektum.

6.6.2.8 setTargetFrameRate()

```
void GameRuntime::setTargetFrameRate (
            const double targetRate)  [static]
```

Maximális frissítési ráta beállítása.

Parameters

targetRate	A maximális frissítési ráta.

6.6.2.9 setTargetPhysicsRate()

```
void GameRuntime::setTargetPhysicsRate (
            const double targetRate)  [static]
```

Fizikai szimulációk rátájának beállítása.

Parameters

targetRate	A fizikai szimulációk rátája.

6.6.2.10 startGameLoop()

```
void GameRuntime::startGameLoop ()  [static]
```

A játék főciklusának elindítása.

A főciklus során a játék frissítése, a fizikai frissítés, a felhasználói események kezelése és a megjelenítés történik. A ciklus addig fut, amíg a `running` állapot igaz.

6.6.2.11 unregisterForUpdate() [1/2]

```
void GameRuntime::unregisterForUpdate (
            PhysicsUpdatable *const updatable)  [static]
```

A fizikai frissítést végző objektum eltávolítása.

Parameters

updatable	A fizikai frissítést végző objektum.

6.6.2.12 unregisterForUpdate() [2/2]

```
void GameRuntime::unregisterForUpdate (
            Updatable *const updatable)  [static]
```

Az updatelhető objektum eltávolítása az updatelésből.

Parameters

updatable	Az updatelhető objektum.

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

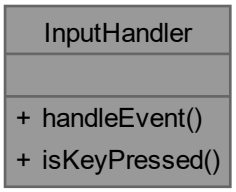
- core.h
- core.cpp

6.7 InputHandler Class Reference

A felhasználói bemenetek kezelésére szolgáló osztály.

`#include <inputhandler.h>`

Collaboration diagram for InputHandler:



Static Public Member Functions

- static void handleEvent (SDL_Event &event)

  Kezeli a felhasználói bemenetekhez tartozó SDL eseményeket.
- static bool isKeyPressed (SDL_Keycode key)

  A billentyűk állapotának lekérdezése.

6.7.1 Detailed Description

A felhasználói bemenetek kezelésére szolgáló osztály.

Az `InputHandler` osztály felelős a felhasználói bemenetek, például billentyűleütések és billentyűfelengedések kezeléséért. Az osztály nyomon követi a lenyomott billentyűket, és lehetővé teszi azok állapotának gyors lekérdezését.

6.7.2 Member Function Documentation

6.7.2.1 handleEvent()

```
void InputHandler::handleEvent (
            SDL_Event & event)  [static]
```

Kezeli a felhasználói bemenetekhez tartozó SDL eseményeket.

A metódus az SDL események alapján frissíti a `pressedKeys` halmazt. Billentyűleütés esetén hozzáadja a billentyű kódját a halmazhoz, míg billentyűfelengedés esetén eltávolítja azt.

Parameters

event	Az SDL esemény, amely tartalmazza a felhasználói bemenetet.

6.7.2.2 isKeyPressed()

```
bool InputHandler::isKeyPressed (
            SDL_Keycode key)  [static]
```

A billentyűk állapotának lekérdezése.

A metódus ellenőrzi, hogy a megadott billentyű kódja szerepel-e a `pressedKeys` halmazban. Ha a billentyű lenyomva van, igaz értéket ad vissza, különben hamisat.

Parameters

key	A billentyű kódja, amelynek állapotát le kell kérdezni.

Returns

true, ha a billentyű lenyomva van, különben false.

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

- inputhandler.h
- inputhandler.cpp

6.8 InputScheme Class Reference

Egy játékoshoz tartozó billentyűkiosztásokat reprezentáló osztály.

`#include <inputscheme.h>`

Collaboration diagram for InputScheme:

Public Member Functions

- InputScheme (const SDL_Keycode jumpKey, const SDL_Keycode leftKey, const SDL_Keycode dashKey, const SDL_Keycode rightKey)

  Konstruktor.
- SDL_Keycode getJumpKey () const

  Visszaadja a játékos ugrásához használt billentyű kódját.
- SDL_Keycode getLeftKey () const

  Visszaadja a játékos balra mozgásához használt billentyű kódját.
- SDL_Keycode getDashKey () const

  Visszaadja a játékos "lefelé vetődés"-hez használt billentyű kódját.
- SDL_Keycode getRightKey () const

  Visszaadja a játékos jobbra mozgásához használt billentyű kódját.

6.8.1 Detailed Description

Egy játékoshoz tartozó billentyűkiosztásokat reprezentáló osztály.

Az `InputScheme` osztály a játékoshoz tartozó billentyűkiosztásokat reprezentálja. Az osztály tárolja a játékos által használt billentyűk kódjait, például az ugrás, mozgás és speciális műveletek vezérléséhez. Lehetővé teszi a billentyűkódok lekérdezését, amelyeket a játék bemeneti kezelője használhat.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 InputScheme()

```
InputScheme::InputScheme (
            const SDL_Keycode jumpKey,
            const SDL_Keycode leftKey,
            const SDL_Keycode dashKey,
            const SDL_Keycode rightKey)
```

Konstruktor.

Inicializálja az `InputScheme` objektumot a megadott billentyűkódokkal. A billentyűkódok az SDL által definiált key code-ok, amelyek a játékos különböző műveleteit vezérlik, például ugrás, mozgás és speciális akciók.

Parameters

jumpKey	A játékos ugrásához használt billentyű kódja (SDL key code).
leftKey	A játékos balra mozgásához használt billentyű kódja (SDL key code).
dashKey	A játékos "lefelé vetődés"-hez használt billentyű kódja (SDL key code).
rightKey	A játékos jobbra mozgásához használt billentyű kódja (SDL key code).

6.8.3 Member Function Documentation

6.8.3.1 getDashKey()

```
SDL_Keycode InputScheme::getDashKey () const
```

Visszaadja a játékos "lefelé vetődés"-hez használt billentyű kódját.

A metódus visszaadja az SDL által definiált key code-ot, amely a játékos "lefelé vetődés" műveletének vezérlésére szolgál.

Returns

A játékos "lefelé vetődés"-hez használt billentyű kódja.

6.8.3.2 getJumpKey()

```
SDL_Keycode InputScheme::getJumpKey () const
```

Visszaadja a játékos ugrásához használt billentyű kódját.

A metódus visszaadja az SDL által definiált key code-ot, amely a játékos ugrásának vezérlésére szolgál.

Returns

A játékos ugrásához használt billentyű kódja.

6.8.3.3 getLeftKey()

```
SDL_Keycode InputScheme::getLeftKey () const
```

Visszaadja a játékos balra mozgásához használt billentyű kódját.

A metódus visszaadja az SDL által definiált key code-ot, amely a játékos balra mozgásának vezérlésére szolgál.

Returns

A játékos balra mozgásához használt billentyű kódja.

6.8.3.4 getRightKey()

```
SDL_Keycode InputScheme::getRightKey () const
```

Visszaadja a játékos jobbra mozgásához használt billentyű kódját.

A metódus visszaadja az SDL által definiált key code-ot, amely a játékos jobbra mozgásának vezérlésére szolgál.

Returns

A játékos jobbra mozgásához használt billentyű kódja.

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

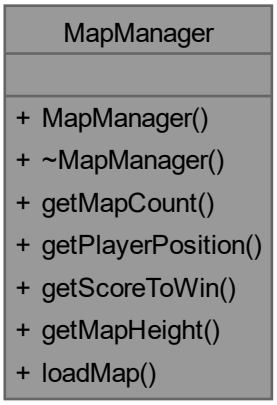
- inputscheme.h
- inputscheme.cpp

6.9 MapManager Class Reference

A játék pályáinak kezelésére szolgáló osztály.

```
#include <mapmanager.h>
```

Collaboration diagram for MapManager:



Public Member Functions

- MapManager ()

  Létrehoz egy `MapManager` objektumot.
- ~MapManager ()

  Felszabadítja a `MapManager` által használt erőforrásokat.
- size_t getMapCount () const

  Visszaadja a pályák számát.
- Vector2 getPlayerPosition (const size_t playerId) const

  Lekérdezi egy játékos kezdőpozícióját az aktuális pályán.
- int getScoreToWin () const

  Visszaadja a győzelemhez szükséges pontszámot az aktuális pályán.
- double getMapHeight () const

  Visszaadja a pálya magasságát.
- void loadMap (const size_t mapId)

  Betölt egy pályát a játék világába.

6.9.1 Detailed Description

A játék pályáinak kezelésére szolgáló osztály.

A `MapManager` osztály felelős a játék pályáinak betöltéséért, eltávolításáért, és az ezekhez kapcsolódó adatok kezeléséért. Kezeli a pályák elemeit, például falakat, játékosok kezdőpozícióit, háttérszíneket, valamint a győzelemhez szükséges pontszámot.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 MapManager()

```
MapManager::MapManager ()
```

Létrehoz egy `MapManager` objektumot.

Inicializálja a pályakezelőt, és előkészíti a pályák kezeléséhez szükséges adatstruktúrákat.

6.9.2.2 ~MapManager()

```
MapManager::~MapManager ()
```

Felszabadítja a `MapManager` által használt erőforrásokat.

A destruktor eltávolítja az összes betöltött pályát, és felszabadítja az erőforrásokat.

6.9.3 Member Function Documentation

6.9.3.1 getMapCount()

```
size_t MapManager::getMapCount () const
```

Visszaadja a pályák számát.

Returns

A pályák száma.

6.9.3.2 getMapHeight()

```
double MapManager::getMapHeight () const
```

Visszaadja a pálya magasságát.

Returns

A pálya magassága.

Exceptions

std::out_of_range	Ha a `mapId` érvénytelen.

6.9.3.3 getPlayerPosition()

```
Vector2 MapManager::getPlayerPosition (
            const size_t playerId) const
```

Lekérdezi egy játékos kezdőpozícióját az aktuális pályán.

A metódus visszaadja az adott pályán lévő játékos kezdőpozícióját.

Parameters

mapId	A pálya azonosítója.
playerId	A játékos azonosítója (0 az első játékos, 1 a második játékos).

Returns

A játékos kezdőpozíciója.

Exceptions

std::out_of_range	Ha a `mapId` vagy a `playerId` érvénytelen.

6.9.3.4 getScoreToWin()

```
int MapManager::getScoreToWin () const
```

Visszaadja a győzelemhez szükséges pontszámot az aktuális pályán.

Returns

A győzelemhez szükséges pontszám.

Exceptions

std::out_of_range	Ha a `mapId` érvénytelen.

6.9.3.5 loadMap()

```
void MapManager::loadMap (
            const size_t mapId)
```

Betölt egy pályát a játék világába.

A metódus eltávolítja az aktuálisan betöltött pályát, majd betölti a megadott azonosítójú pályát, és inicializálja annak elemeit.

Parameters

mapId	A betöltendő pálya azonosítója.

Exceptions

std::out_of_range	Ha a `mapId` érvénytelen.

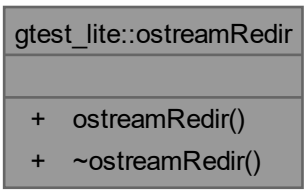
The documentation for this class was generated from the following files:

- mapmanager.h
- mapmanager.cpp

6.10 gtest_lite::ostreamRedir Class Reference

`#include <gtest_lite.h>`

Collaboration diagram for gtest_lite::ostreamRedir:



Public Member Functions

- **ostreamRedir** (std::ostream &src, std::ostream &dst)

6.10.1 Detailed Description

Segédsablon ostream átirányításához A destruktor visszaállít

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

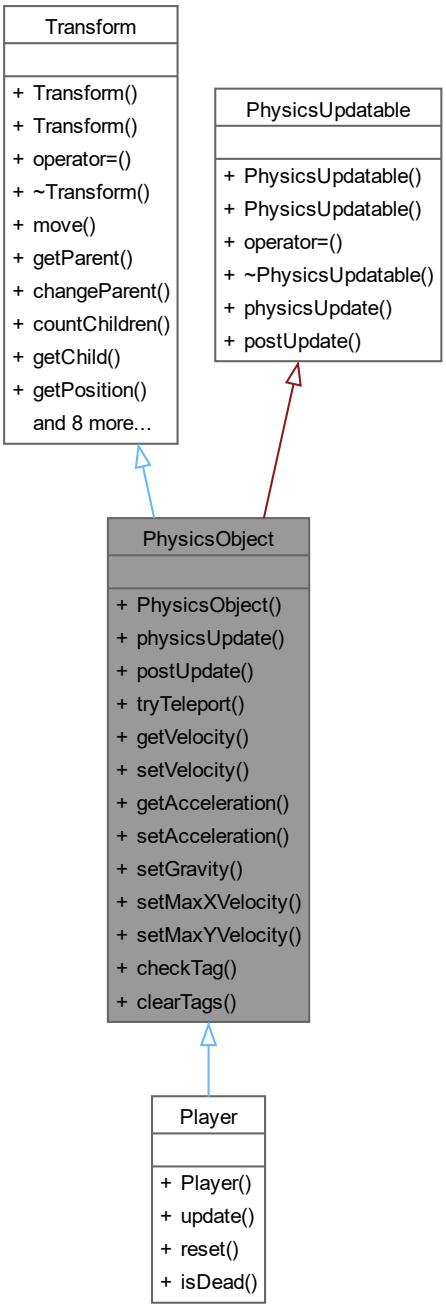
- gtest_lite.h

6.11 PhysicsObject Class Reference

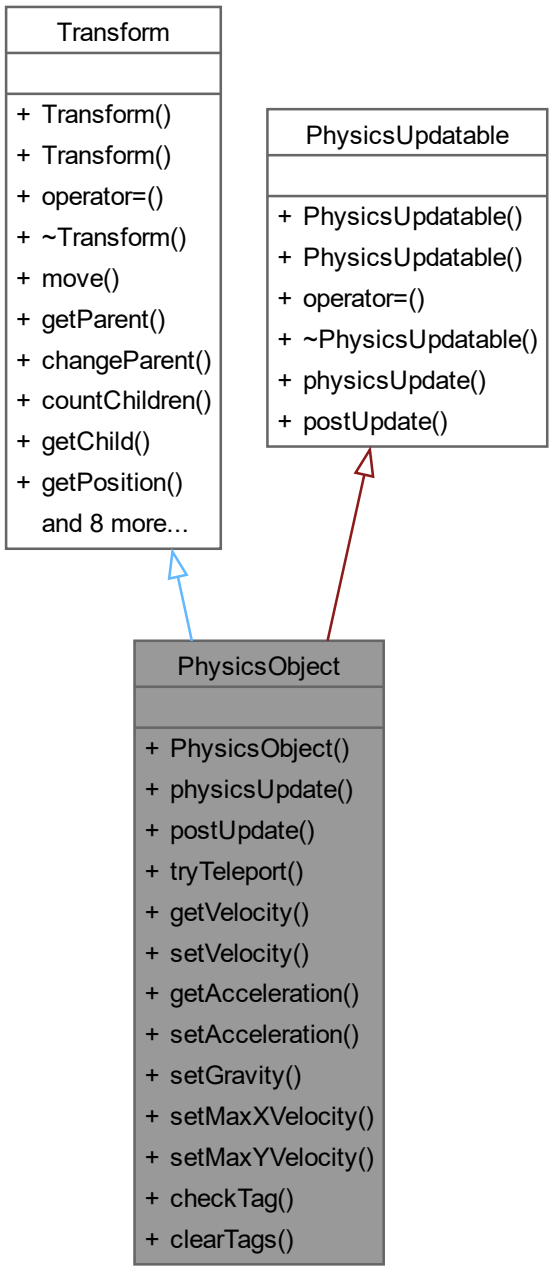
Egy fizikai objektumot reprezentáló osztály.

`#include <physicsObject.h>`

Inheritance diagram for PhysicsObject:



Collaboration diagram for PhysicsObject:



Public Member Functions

- PhysicsObject (const Transform &transform, const std::vector< Collider * > &colliders)

  Létrehoz egy PhysicsObject objektumot.
- void physicsUpdate () override

  A fizikai szimuláció frissítése.
- void postUpdate () override

A címkék törlése, az `update` metódus után.

- bool tryTeleport (const Vector2 &position)

  Az objektum pozíciójának közvetlen módosítása (teleportálás).
- Vector2 getVelocity () const

  Visszaadja az objektum aktuális sebességét.
- void setVelocity (const Vector2 &velocity)

  Beállítja az objektum sebességét.
- Vector2 getAcceleration () const

  Visszaadja az objektum aktuális gyorsulását.
- void setAcceleration (const Vector2 &acceleration)

  Beállítja az objektum gyorsulását.
- void setGravity (const Vector2 &gravity)

  Beállítja az objektumra ható gravitációt.
- void setMaxXVelocity (const std::pair< double, double > &maxXVelocity)

  Beállítja az objektum maximális sebességét az X tengelyen.
- void setMaxYVelocity (const std::pair< double, double > &maxYVelocity)

  Beállítja az objektum maximális sebességét az Y tengelyen.
- bool checkTag (const ColliderTag tag) const

  Visszaadja hogy az objektum érintette-e a megadott collider címkét a legutóbbi `update` óta.
- void clearTags ()

  Törli az összes érintett collider címkét.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.
- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.

- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

6.11.1 Detailed Description

Egy fizikai objektumot reprezentáló osztály.

A `PhysicsObject` osztály felelős az objektum fizikai tulajdonságainak kezeléséért, beleértve a sebességet, gyorsulást, gravitációt és ütközéseket. Az osztály a `Transform` és `PhysicsUpdatable` osztályokból származik, így támogatja a pozíció, méret és ütközésdetektálás kezelését, valamint a fizikai szimuláció frissítését.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 PhysicsObject()

```
PhysicsObject::PhysicsObject (
            const Transform & transform,
            const std::vector< Collider * > & colliders)
```

Létrehoz egy PhysicsObject objektumot.

A konstruktor inicializálja az objektum pozícióját, méretét és a hozzá tartozó collidereket. Az objektum a megadott collidereket használja az ütközésdetektáláshoz.

Parameters

transform	Az objektum pozíciója és mérete a játék világában.
colliders	Az objektumhoz tartozó colliderek listája.

6.11.3 Member Function Documentation

6.11.3.1 checkTag()

```
bool PhysicsObject::checkTag (
            const ColliderTag tag) const
```

Visszaadja hogy az objektum érintette-e a megadott collider címkét a legutóbbi `update` óta.

Returns

true, ha az objektum érintette a megadott collider címkét, különben false.

6.11.3.2 clearTags()

```
void PhysicsObject::clearTags ()
```

Törli az összes érintett collider címkét.

A metódus eltávolítja az összes érintett collider címkét az objektum `touchedTags` halmazából. Ezt minden `update` után meghívja a `postUpdate`.

6.11.3.3 getAcceleration()

```
Vector2 PhysicsObject::getAcceleration () const
```

Visszaadja az objektum aktuális gyorsulását.

Returns

Az objektum aktuális gyorsulása.

6.11.3.4 getVelocity()

```
Vector2 PhysicsObject::getVelocity () const
```

Visszaadja az objektum aktuális sebességét.

Returns

Az objektum aktuális sebessége.

6.11.3.5 physicsUpdate()

```
void PhysicsObject::physicsUpdate ()  [override], [virtual]
```

A fizikai szimuláció frissítése.

A metódus frissíti az objektum pozícióját, sebességét és gyorsulását a fizikai szimuláció szabályai alapján. Kezeli az ütközéseket a colliderekkel, és figyelembe veszi az ütközési visszapattanásokat (bounciness).

Implements PhysicsUpdatable.

6.11.3.6 postUpdate()

```
void PhysicsObject::postUpdate ()  [override], [virtual]
```

A címkék törlése, az `update` metódus után.

Implements PhysicsUpdatable.

6.11.3.7 setAcceleration()

```
void PhysicsObject::setAcceleration (
            const Vector2 & acceleration)
```

Beállítja az objektum gyorsulását.

Parameters

acceleration	Az új gyorsulás vektora.

6.11.3.8 setGravity()

```
void PhysicsObject::setGravity (
            const Vector2 & gravity)
```

Beállítja az objektumra ható gravitációt.

Parameters

gravity	Az új gravitációs erő vektora.

6.11.3.9 setMaxXVelocity()

```
void PhysicsObject::setMaxXVelocity (
            const std::pair< double, double > & maxXVelocity)
```

Beállítja az objektum maximális sebességét az X tengelyen.

Parameters

maxXVelocity	Az új maximális sebesség (negatív és pozitív irányban).

6.11.3.10 setMaxYVelocity()

```
void PhysicsObject::setMaxYVelocity (
            const std::pair< double, double > & maxYVelocity)
```

Beállítja az objektum maximális sebességét az Y tengelyen.

Parameters

maxYVelocity	Az új maximális sebesség (negatív és pozitív irányban).

6.11.3.11 setVelocity()

```
void PhysicsObject::setVelocity (
            const Vector2 & velocity)
```

Beállítja az objektum sebességét.

Parameters

velocity	Az új sebesség vektora.

6.11.3.12 tryTeleport()

```
bool PhysicsObject::tryTeleport (
            const Vector2 & position)
```

Az objektum pozíciójának közvetlen módosítása (teleportálás).

A metódus megpróbálja az objektumot a megadott pozícióra teleportálni. Az ütközések elkerülése érdekében ellenőrzi, hogy a célpozíció érvényes-e a colliderekkel való metszés szempontjából.

Fontos: A metódus használata előtt győződj meg arról, hogy az objektumhoz tartozó colliderek megfelelően inicializálva vannak. Ha a `searchChildrenForColliders` beállítás `true`, akkor érdemes legalább egy `physicsUpdate` hívást végrehajtani a teleportálás előtt. Ellenkező esetben a colliderek listáját a konstruktorban kell megfelelően inicializálni.

Parameters

position	A célpozíció, ahová az objektumot teleportálni kell.

Returns

true, ha a teleportálás sikeres, különben false.

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

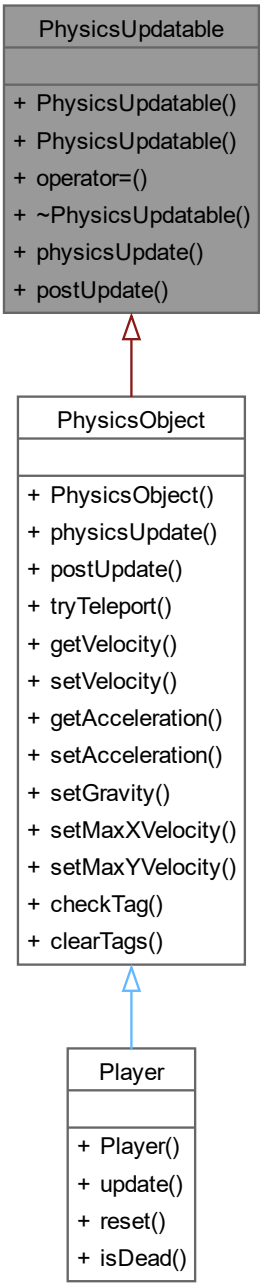
- physicsObject.h
- physicsObject.cpp

6.12 PhysicsUpdatable Class Reference

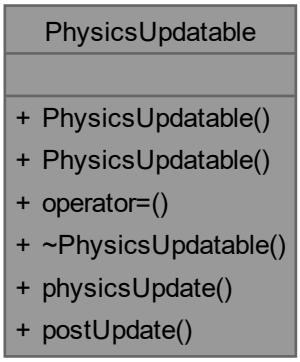
A fizikai frissítést végző objektumokat reprezentáló osztály.

```
#include <core.h>
```

Inheritance diagram for PhysicsUpdatable:



Collaboration diagram for PhysicsUpdatable:



Public Member Functions

- **PhysicsUpdatable** ()

  A PhysicsUpdatable osztály konstruktora.
- **PhysicsUpdatable** (const PhysicsUpdatable &updatable)

  A PhysicsUpdatable osztály másoló konstruktora.
- PhysicsUpdatable & **operator=** (const PhysicsUpdatable &updatable)

  A PhysicsUpdatable osztály értékadó operátora.
- virtual ~**PhysicsUpdatable** ()

  A PhysicsUpdatable osztály destruktora.
- virtual void physicsUpdate ()=0

  A fizikai frissítést végző virtuális metódus.
- virtual void postUpdate ()=0

  A normál frissítést követő metódus.

6.12.1 Detailed Description

A fizikai frissítést végző objektumokat reprezentáló osztály.

A `PhysicsUpdatable` osztály azokat az objektumokat reprezentálja, amelyek fizikai frissítést végeznek a játék főciklusában. Az osztályból származtatott objektumok implementálják a `physicsUpdate()` metódust, amely a fizikai szimulációk frissítését végzi, például mozgás vagy ütközés számításokat.

6.12.2 Member Function Documentation

6.12.2.1 physicsUpdate()

```
virtual void PhysicsUpdatable::physicsUpdate ()  [pure virtual]
```

A fizikai frissítést végző virtuális metódus.

Implemented in PhysicsObject.

6.12.2.2 postUpdate()

```
virtual void PhysicsUpdatable::postUpdate ()  [pure virtual]
```

A normál frissítést követő metódus.

Lehetőséget biztosít a fizikai frissítést végző objektumok számára, hogy a normál frissítést követően végezzenek el további műveleteket, például a címkék eltűntetését.

Implemented in PhysicsObject.

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

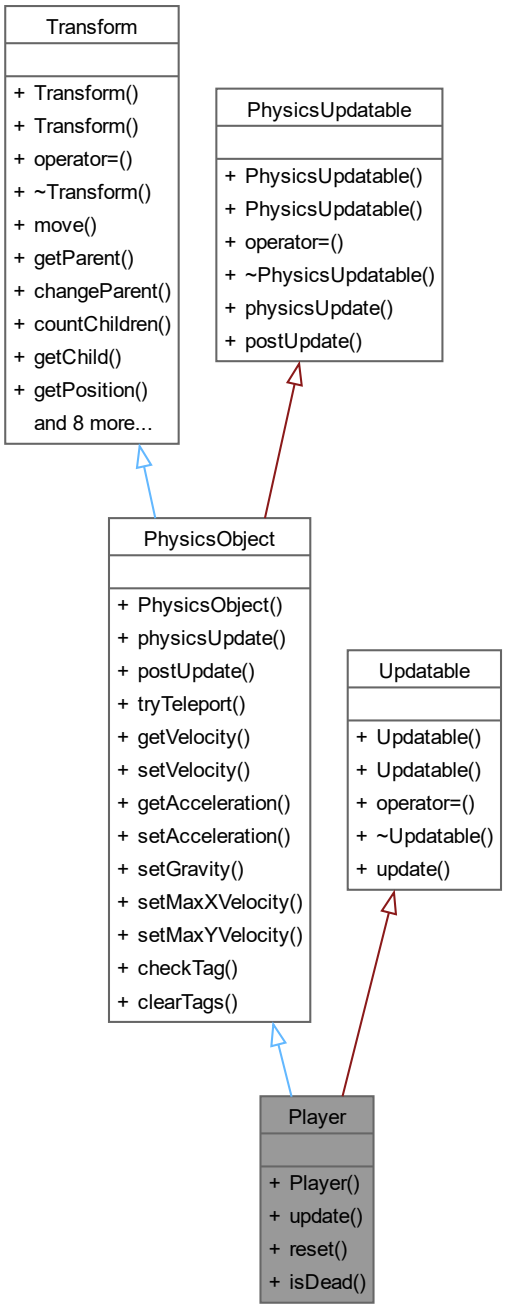
- core.h
- core.cpp

6.13 Player Class Reference

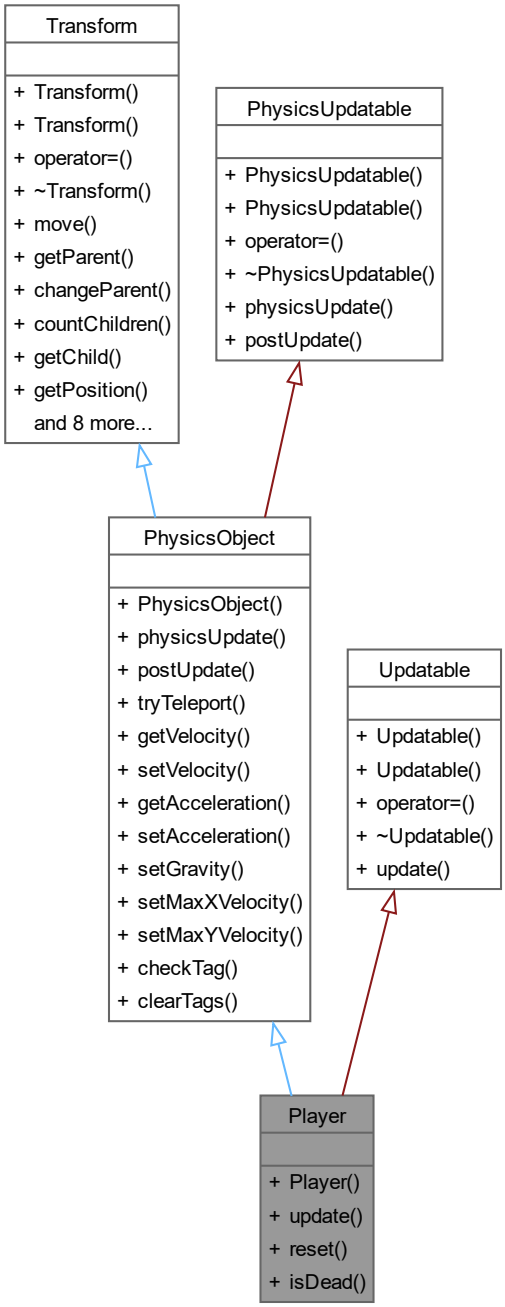
A játékos karaktert reprezentáló osztály.

```
#include <player.h>
```

Inheritance diagram for Player:



Collaboration diagram for Player:



Public Member Functions

- Player (const Color &color, const InputScheme &inputScheme)

  Létrehoz egy `Player` objektumot.
- void update () override

  A játékos frissítése.
- void reset (const Vector2 &resetPosition)

A játékos állapotának visszaállítása.

- bool isDead () const

  Visszaadja, hogy a játékos meghalt-e.

Public Member Functions inherited from PhysicsObject

- PhysicsObject (const Transform &transform, const std::vector< Collider * > &colliders)

  Létrehoz egy PhysicsObject objektumot.
- void physicsUpdate () override

  A fizikai szimuláció frissítése.
- void postUpdate () override

  A címkék törlése, az `update` metódus után.
- bool tryTeleport (const Vector2 &position)

  Az objektum pozíciójának közvetlen módosítása (teleportálás).
- Vector2 getVelocity () const

  Visszaadja az objektum aktuális sebességét.
- void setVelocity (const Vector2 &velocity)

  Beállítja az objektum sebességét.
- Vector2 getAcceleration () const

  Visszaadja az objektum aktuális gyorsulását.
- void setAcceleration (const Vector2 &acceleration)

  Beállítja az objektum gyorsulását.
- void setGravity (const Vector2 &gravity)

  Beállítja az objektumra ható gravitációt.
- void setMaxXVelocity (const std::pair< double, double > &maxXVelocity)

  Beállítja az objektum maximális sebességét az X tengelyen.
- void setMaxYVelocity (const std::pair< double, double > &maxYVelocity)

  Beállítja az objektum maximális sebességét az Y tengelyen.
- bool checkTag (const ColliderTag tag) const

  Visszaadja hogy az objektum érintette-e a megadott collider címkét a legutóbbi `update` óta.
- void clearTags ()

  Törli az összes érintett collider címkét.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.

- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

6.13.1 Detailed Description

A játékos karaktert reprezentáló osztály.

A `Player` osztály felelős a játékos karakter fizikai és logikai működéséért. Kezeli a játékos mozgását, ugrását, vetődését és ütközéseit. Az osztály az `InputScheme` segítségével dolgozza fel a bemeneteket, és a `PhysicsObject` osztályból származik, így támogatja a fizikai szimulációt és az ütközésdetektálást.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Player()

```
Player::Player (
            const Color & color,
            const InputScheme & inputScheme)
```

Létrehoz egy `Player` objektumot.

Inicializálja a játékos színét, bemeneti kiosztását és fizikai tulajdonságait.

Parameters

color	A játékos színe.
inputScheme	A játékoshoz tartozó bemeneti kiosztás.

6.13.3 Member Function Documentation

6.13.3.1 isDead()

```
bool Player::isDead () const
```

Visszaadja, hogy a játékos meghalt-e.

Returns

true, ha a játékos meghalt, különben false.

6.13.3.2 reset()

```
void Player::reset (
            const Vector2 & resetPosition)
```

A játékos állapotának visszaállítása.

A metódus visszaállítja a játékos pozícióját, sebességét és állapotát a megadott kezdőpozícióra. Emellett alaphelyzetbe állítja a halál állapotát és a mozgással kapcsolatos paramétereket.

Parameters

resetPosition	A játékos új pozíciója.

6.13.3.3 update()

```
void Player::update ()  [override], [virtual]
```

A játékos frissítése.

A metódus a játékos mozgását, ütközéseit és állapotát kezeli a játék főciklusában.

Implements Updatable.

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

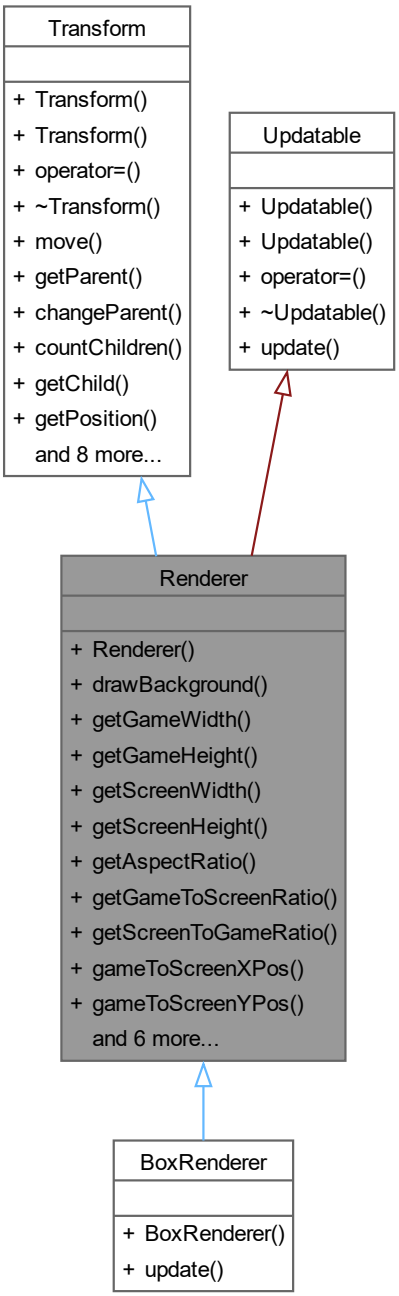
- player.h
- player.cpp

6.14 Renderer Class Reference

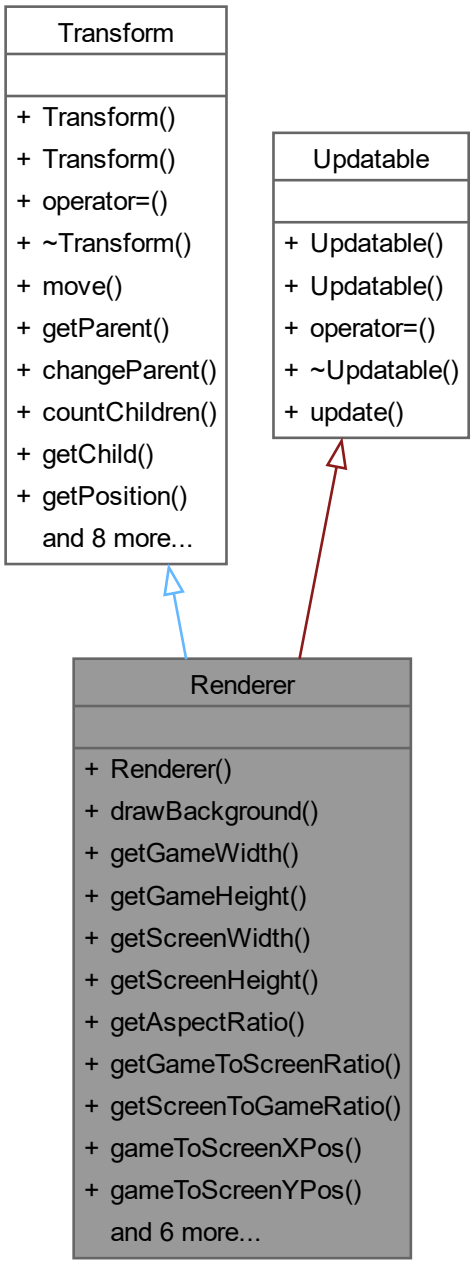
A játék grafikai megjelenítéséért felelős osztály.

`#include <renderer.h>`

Inheritance diagram for Renderer:



Collaboration diagram for Renderer:



Public Member Functions

- Renderer (const Transform &transform, UpdatePriority priority)

  Létrehoz egy `Renderer` objektumot.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

Konstruktor.

- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.
- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

Static Public Member Functions

- static void drawBackground ()

  Kirajzolja a játék hátterét.
- static double getGameWidth ()

  Visszaadja a játék világának szélességét.
- static double getGameHeight ()

  Visszaadja a játék világának magasságát.
- static int getScreenWidth ()

  Visszaadja a képernyő szélességét.
- static int getScreenHeight ()

  Visszaadja a képernyő magasságát.
- static double getAspectRatio ()

Visszaadja a képernyő képarányát.

- static double getGameToScreenRatio ()

  Visszaadja a játék világának és a képernyőnek az arányát.
- static double getScreenToGameRatio ()

  Visszaadja a képernyő és a játék világának arányát.
- static int gameToScreenXPos (const double gameXPos)

  Átváltja a játék világának X koordinátáját képernyő koordinátára.
- static int gameToScreenYPos (const double gameYPos)

  Átváltja a játék világának Y koordinátáját képernyő koordinátára.
- static double screenToGameXPos (const int screenXPos)

  Átváltja a képernyő X koordinátáját a játék világának koordinátájára.
- static double screenToGameYPos (const int screenYPos)

  Átváltja a képernyő Y koordinátáját a játék világának koordinátájára.
- static void setGameHeight (const double gameHeight)

  Beállítja a játék világának magasságát.
- static void setBackgroundColor (const Color color)

  Beállítja a játék háttérszínét.
- static void setCameraOffset (const Vector2 &offset)

  Beállítja a játék kamerájának eltolását.
- static void setCameraScale (const double scale)

  Visszaadja a játék kamerájának méretét.

6.14.1 Detailed Description

A játék grafikai megjelenítéséért felelős osztály.

A `Renderer` osztály felelős a játék grafikai elemeinek megjelenítéséért, beleértve a háttér kirajzolását, a képernyő és a játék világának koordinátái közötti átváltást, valamint a játék világának méretének és arányainak kezelését.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Renderer()

```
Renderer::Renderer (
            const Transform & transform,
            UpdatePriority priority)
```

Létrehoz egy `Renderer` objektumot.

Inicializálja a rendereléshez szükséges transzformációkat és prioritásokat.

Parameters

transform	Az objektum pozíciója és mérete.
priority	Az objektum frissítési prioritása.

6.14.3 Member Function Documentation

6.14.3.1 drawBackground()

```
void Renderer::drawBackground ()  [static]
```

Kirajzolja a játék hátterét.

A metódus a beállított háttérszínt használva kirajzolja a játék hátterét.

6.14.3.2 gameToScreenXPos()

```
int Renderer::gameToScreenXPos (
            const double gameXPos)  [static]
```

Átváltja a játék világának X koordinátáját képernyő koordinátára.

Parameters

gameXPos	A játék világának X koordinátája.

Returns

Az X koordináta pixelben.

6.14.3.3 gameToScreenYPos()

```
int Renderer::gameToScreenYPos (
            const double gameYPos)  [static]
```

Átváltja a játék világának Y koordinátáját képernyő koordinátára.

Parameters

gameYPos	A játék világának Y koordinátája.

Returns

Az Y koordináta pixelben.

6.14.3.4 getAspectRatio()

```
double Renderer::getAspectRatio ()  [static]
```

Visszaadja a képernyő képarányát.

Az arány a képernyő szélessége és magassága közötti hányados.

Returns

A képernyő képaránya.

6.14.3.5 getGameHeight()

```
double Renderer::getGameHeight ()  [static]
```

Visszaadja a játék világának magasságát.

Returns

A játék világának magassága.

6.14.3.6 getGameToScreenRatio()

```
double Renderer::getGameToScreenRatio ()  [static]
```

Visszaadja a játék világának és a képernyőnek az arányát.

Az arány megmutatja, hogy a játék világának egy egysége hány pixelnek felel meg a képernyőn.

Returns

A játék világának és a képernyőnek az aránya.

6.14.3.7 getGameWidth()

```
double Renderer::getGameWidth ()  [static]
```

Visszaadja a játék világának szélességét.

A szélesség a játék világának magassága és a képarány alapján kerül kiszámításra.

Returns

A játék világának szélessége játékegységekben.

6.14.3.8 getScreenHeight()

```
int Renderer::getScreenHeight ()  [static]
```

Visszaadja a képernyő magasságát.

Returns

A képernyő magassága pixelben.

6.14.3.9 getScreenToGameRatio()

```
double Renderer::getScreenToGameRatio ()  [static]
```

Visszaadja a képernyő és a játék világának arányát.

Az arány megmutatja, hogy a képernyő egy pixelje hány egységnek felel meg a játék világában.

Returns

A képernyő és a játék világának aránya.

6.14.3.10 getScreenWidth()

```
int Renderer::getScreenWidth ()  [static]
```

Visszaadja a képernyő szélességét.

Returns

A képernyő szélessége pixelben.

6.14.3.11 screenToGameXPos()

```
double Renderer::screenToGameXPos (
            const int screenXPos)  [static]
```

Átváltja a képernyő X koordinátáját a játék világának koordinátájára.

Parameters

screenXPos	A képernyő X koordinátája pixelben.

Returns

Az X koordináta a játék világában.

6.14.3.12 screenToGameYPos()

```
double Renderer::screenToGameYPos (
            const int screenYPos)  [static]
```

Átváltja a képernyő Y koordinátáját a játék világának koordinátájára.

Parameters

screenYPos	A képernyő Y koordinátája pixelben.

Returns

Az Y koordináta a játék világában.

6.14.3.13 setBackgroundColor()

```
void Renderer::setBackgroundColor (
            const Color color)  [static]
```

Beállítja a játék háttérszínét.

Parameters

color	Az új háttérszín.

6.14.3.14 setCameraOffset()

```
void Renderer::setCameraOffset (
            const Vector2 & offset)  [static]
```

Beálítja a játék kamerájának eltolását.

Parameters

offset	Az új eltolás vektora.

6.14.3.15 setCameraScale()

```
void Renderer::setCameraScale (
            const double scale)  [static]
```

Visszaadja a játék kamerájának méretét.

Parameters

scale	A kamera mérete.

6.14.3.16 setGameHeight()

```
void Renderer::setGameHeight (
            const double gameHeight)  [static]
```

Beállítja a játék világának magasságát.

A magasság beállítása után a szélesség automatikusan kiszámításra kerül az arány alapján.

Parameters

gameHeight	A játék világának új magassága.

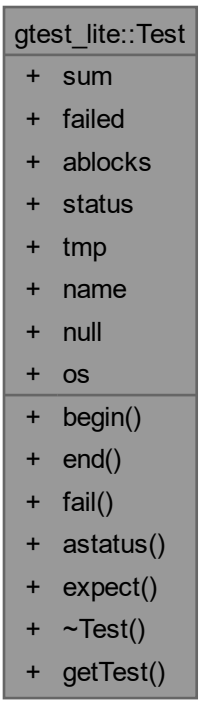
The documentation for this class was generated from the following files:

- renderer.h
- renderer.cpp

6.15 gtest_lite::Test Struct Reference

`#include <gtest_lite.h>`

Collaboration diagram for gtest_lite::Test:



Public Member Functions

- void **begin** (const char *n)

  Teszt kezdete.
- std::ostream & **end** (bool memchk=false)

  Teszt vége.
- bool **fail** ()
- bool **astatus** ()
- std::ostream & **expect** (bool st, const char *file, int line, const char *expr, bool pr=false)

  Eredményt adminisztráló tagfüggvény True a jó eset.
- ~**Test** ()

  Destruktor.

Static Public Member Functions

- static Test & getTest ()

Public Attributes

- int **sum**

  tesztek számlálója
- int **failed**

  hibás tesztek
- int **ablocks**

  allokált blokkok száma
- bool **status**

  éppen futó teszt státusza
- bool **tmp**

  temp a kivételkezeléshez;
- std::string **name**

  éppen futó teszt neve
- std::fstream **null**

  nyelő, ha nem kell kiírni semmit
- std::ostream & **os**

  ide írunk

6.15.1 Detailed Description

Tesztek állapotát tároló osztály. Egyetlen egy statikus példány keletkezik, aminek a destruktora a futás végén hívódik meg.

6.15.2 Member Function Documentation

6.15.2.1 getTest()

```
static Test & gtest_lite::Test::getTest ()  [inline], [static]
```

< egyedüli (singleton) példány

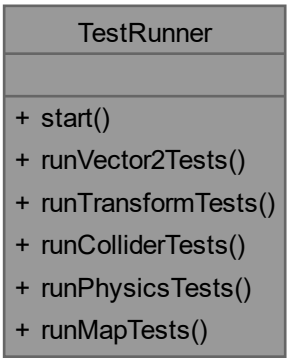
The documentation for this struct was generated from the following file:

- gtest_lite.h

6.16 TestRunner Class Reference

```
#include <test.h>
```

Collaboration diagram for TestRunner:



Static Public Member Functions

- static void **start** ()
- static void **runVector2Tests** ()
- static void **runTransformTests** ()
- static void **runColliderTests** ()
- static void **runPhysicsTests** ()
- static void **runMapTests** ()

6.16.1 Detailed Description

Tesztek futtatására szolgáló osztály.

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

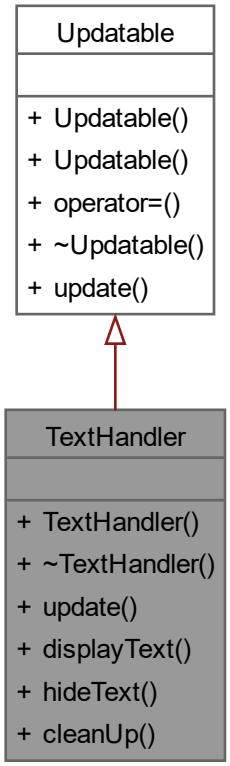
- test.h

6.17 TextHandler Class Reference

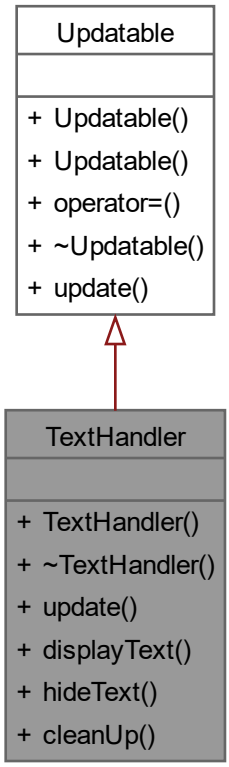
Szövegek kezelésére és megjelenítésére szolgáló osztály.

`#include <texthandler.h>`

Inheritance diagram for TextHandler:



Collaboration diagram for TextHandler:



Public Member Functions

- TextHandler ()

  Létrehoz egy `TextHandler` objektumot.
- ~TextHandler ()

  Felszabadítja a `TextHandler` által használt erőforrásokat.
- void update () override

  A szövegkezelő frissítése a játék főciklusában.
- void displayText (const std::string &text, const Color &color)

  Szöveg megjelenítése a képernyőn.
- void hideText ()

  Elrejti a képernyőn megjelenített szöveget.
- void cleanUp ()

  A szövegkezelő erőforrásainak felszabadítása.

6.17.1 Detailed Description

Szövegek kezelésére és megjelenítésére szolgáló osztály.

A `TextHandler` osztály felelős a játékban megjelenített szövegek kezeléséért, beleértve a szövegek megjelenítését, elrejtését, valamint a szín és méret dinamikus beállítását. Az osztály az SDL_ttf könyvtárat használja a szövegek rendereléséhez, és automatikusan alkalmazkodik a képernyő méretéhez.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 TextHandler()

```
TextHandler::TextHandler ()
```

Létrehoz egy `TextHandler` objektumot.

Inicializálja az osztály tagváltozóit, és előkészíti a szövegkezelést.

6.17.2.2 ~TextHandler()

```
TextHandler::~TextHandler ()
```

Felszabadítja a `TextHandler` által használt erőforrásokat.

A destruktor felszabadítja a betöltött betűtípushoz tartozó erőforrásokat, és biztosítja, hogy ne maradjanak memória-szivárgások.

6.17.3 Member Function Documentation

6.17.3.1 cleanUp()

```
void TextHandler::cleanUp ()
```

A szövegkezelő erőforrásainak felszabadítása.

A metódus felszabadítja a `TextHandler` által használt erőforrásokat, ezalatt a betűtípust értve.

6.17.3.2 displayText()

```
void TextHandler::displayText (
            const std::string & text,
            const Color & color)
```

Szöveg megjelenítése a képernyőn.

A metódus beállítja a megjelenítendő szöveget és annak színét, majd megjeleníti azt a képernyőn.

Parameters

text	A megjelenítendő szöveg.
color	A szöveg színe.

6.17.3.3 hideText()

```
void TextHandler::hideText ()
```

Elrejti a képernyőn megjelenített szöveget.

A metódus eltünteti a jelenleg megjelenített szöveget a képernyőről.

6.17.3.4 update()

```
void TextHandler::update ()  [override], [virtual]
```

A szövegkezelő frissítése a játék főciklusában.

A metódus ellenőrzi a képernyő méretének változását, és ha szükséges, újratölti a betűtípust és újrarendereli a szöveget.

Implements Updatable.

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

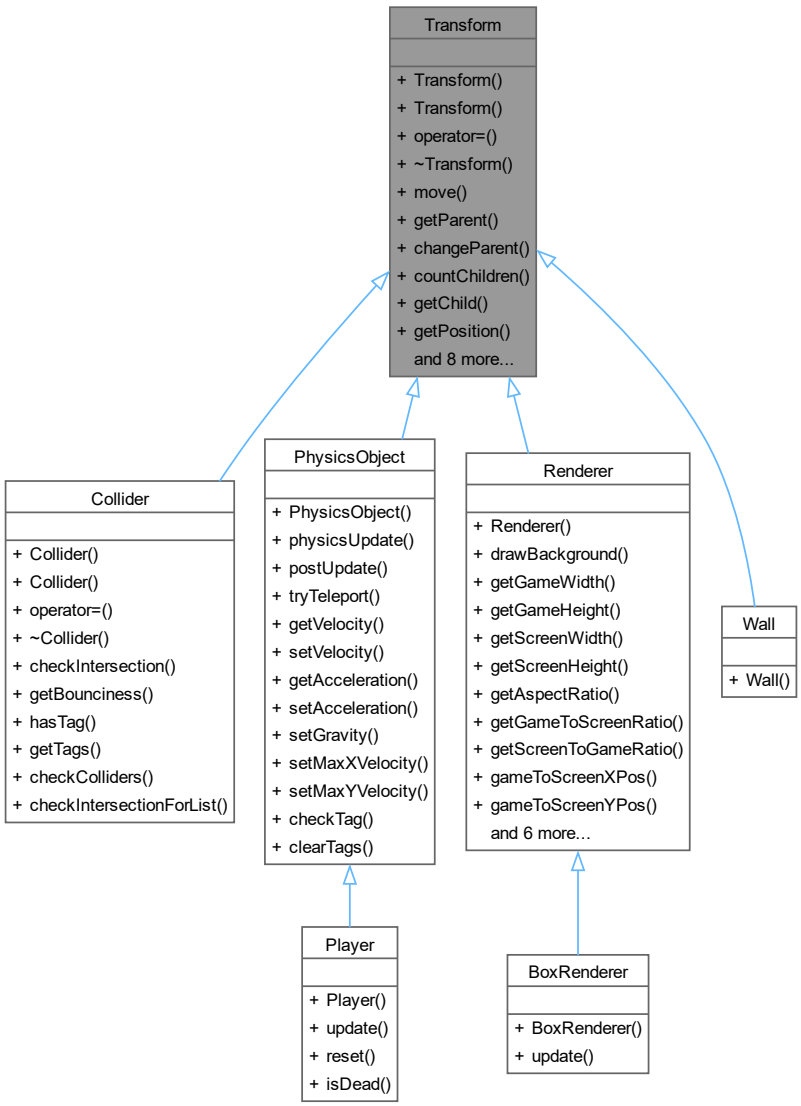
- texthandler.h
- texthandler.cpp

6.18 Transform Class Reference

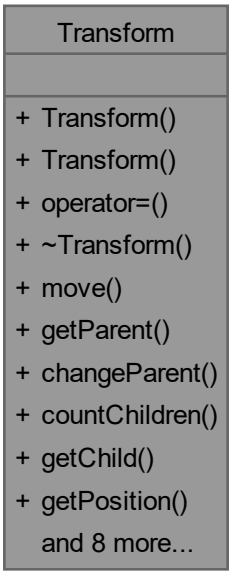
Hierarchikus transzformációkat kezelő osztály.

```
#include <transform.h>
```

Inheritance diagram for Transform:



Collaboration diagram for Transform:



Public Member Functions

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.
- Transform * getParent () const

  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)

  Az objektum szülőjének módosítása.
- size_t countChildren () const

  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const

  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const

  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const

  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)

Beállítja az objektum globális pozícióját.

- void setScale (const Vector2 &scale)

  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const

  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const

  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)

  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)

  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()

  Keres egy adott típusú objektumot a gyermekek között.

6.18.1 Detailed Description

Hierarchikus transzformációkat kezelő osztály.

A `Transform` osztály egy objektum pozícióját és méretét kezeli, valamint támogatja a szülő-gyermek hierarchiát. Lehetővé teszi a globális és lokális transzformációk számítását a játék világában.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Transform() [1/2]

```
Transform::Transform (
            Transform *const parent = nullptr,
            const Vector2 & position = { 0, 0 },
            const Vector2 & scale = { 1, 1 })
```

Konstruktor.

Inicializálja az objektumot a megadott szülővel, pozícióval és mérettel.

Parameters

parent	Az objektum szülője (alapértelmezés szerint nincs szülő).
position	Az objektum pozíciója (alapértelmezés szerint {0, 0}).
scale	Az objektum mérete (alapértelmezés szerint {1, 1}).

6.18.2.2 Transform() [2/2]

```
Transform::Transform (
            const Transform & transform)
```

Másoló konstruktor.

Létrehoz egy új `Transform` objektumot egy meglévő másolataként.

A másoló konstruktor átmásolja az eredeti objektum pozícióját, méretét és szülőjét. Ha az eredeti objektumnak van szülője, az új objektumot automatikusan hozzáadja a szülő gyermekei közé. A gyermekek nem kerülnek másolásra, mivel a másoló konstruktor csak az aktuális objektumot másolja.

Parameters

transform	A másolandó `Transform` objektum.

6.18.2.3 ~Transform()

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

Destruktor.

A destruktor felszabadítja az objektum által használt erőforrásokat, és eltávolítja az objektumot a szülő hierarchiájából. Az alábbi lépéseket hajtja végre:

1. **Gyermekek szülőjének eltávolítása**: Az összes gyermek `Transform` objektum szülőjét `nullptr`-ra állítja, így megszakítja a kapcsolatot a gyermekekkel.
2. **Eltávolítás a szülő gyermekei közül**: Ha az objektumnak van szülője, eltávolítja magát a szülő gyermekei közül.

Ez biztosítja, hogy a hierarchia megfelelően frissüljön, és ne maradjanak érvénytelen hivatkozások.

6.18.3 Member Function Documentation

6.18.3.1 changeParent()

```
void Transform::changeParent (
            Transform *const parent)
```

Az objektum szülőjének módosítása.

Az objektumot áthelyezi egy új szülő alá a hierarchiában. A globális pozíciót és méretet nem változtatja meg. A lokális pozíciót és méretet igazítja az új szülőhöz.

Parameters

parent	Az új szülő objektum.

6.18.3.2 countChildren()

```
size_t Transform::countChildren () const
```

Visszaadja az objektum gyermekei számát.

Returns

Az objektum gyermekei száma.

6.18.3.3 findTypeInChildren()

```
template<typename T>
std::vector< T * > Transform::findTypeInChildren ()
```

Keres egy adott típusú objektumot a gyermekek között.

Template Parameters

T	A keresett objektum típusa.

Returns

Az adott típusú objektumok listája.

6.18.3.4 getChild()

```
Transform * Transform::getChild (
            const size_t idx) const
```

Visszaadja az objektum egy adott gyermekét.

Parameters

idx	A gyermek indexe.

Returns

Az adott gyermek objektum.

6.18.3.5 getLocalPosition()

```
Vector2 Transform::getLocalPosition () const
```

Visszaadja az objektum lokális pozícióját.

Returns

Az objektum lokális pozíciója.

6.18.3.6 getLocalScale()

```
Vector2 Transform::getLocalScale () const
```

Visszaadja az objektum lokális méretét.

Returns

Az objektum lokális mérete.

6.18.3.7 getParent()

```
Transform * Transform::getParent () const
```

Visszaadja az objektum szülőjét.

Returns

Az objektum szülője, vagy nullptr, ha nincs szülő.

6.18.3.8 getPosition()

```
Vector2 Transform::getPosition () const
```

Az objektum globális pozíciójának lekérdezése.

A globális pozíció a szülő hierarchiájának figyelembevételével kerül kiszámításra.

Returns

Az objektum globális pozíciója.

6.18.3.9 getScale()

```
Vector2 Transform::getScale () const
```

Az objektum globális méretének lekérdezése.

A globális méret a szülő hierarchiájának figyelembevételével kerül kiszámításra.

Returns

Az objektum globális mérete.

6.18.3.10 move()

```
void Transform::move (
            const Vector2 & offset)
```

Az objektum elmozdítása.

Az objektum pozícióját a megadott eltolással módosítja.

Parameters

offset	Az elmozdulás vektora.

6.18.3.11 operator=()

```
Transform & Transform::operator= (
            const Transform & transform)
```

Értékadás operátor.

Egy meglévő `Transform` objektum adatait másolja az aktuális objektumba.

Az értékadás operátor átmásolja az eredeti objektum pozícióját, méretét és szülőjét. Az aktuális objektumot eltávolítja a korábbi szülőjéből, és hozzáadja az új szülő gyermekei közé. A gyermekek nem kerülnek másolásra, mivel az értékadás csak az aktuális objektumra vonatkozik.

Parameters

transform	A másolandó `Transform` objektum.

Returns

Az aktuális objektum referenciája.

6.18.3.12 setLocalPosition()

```
void Transform::setLocalPosition (
            const Vector2 & position)
```

Beállítja az objektum lokális pozícióját.

Parameters

position	Az új lokális pozíció.

6.18.3.13 setLocalScale()

```
void Transform::setLocalScale (
            const Vector2 & scale)
```

Beállítja az objektum lokális méretét.

Parameters

scale	Az új lokális méret.

6.18.3.14 setPosition()

```
void Transform::setPosition (
            const Vector2 & position)
```

Beállítja az objektum globális pozícióját.

Ha az objektumnak van szülője, a megadott globális pozíciót lokális pozícióvá alakítja a szülő globális pozíciója és mérete alapján. Ha nincs szülője, a pozíció közvetlenül kerül beállításra.

Parameters

position	Az új pozíció.

6.18.3.15 setScale()

```
void Transform::setScale (
            const Vector2 & scale)
```

Beállítja az objektum globális méretét.

Ha az objektumnak van szülője, a megadott globális méretet lokális méretté alakítja a szülő globális mérete alapján. Ha nincs szülője, a méret közvetlenül kerül beállításra.

Parameters

scale	Az új méret.

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

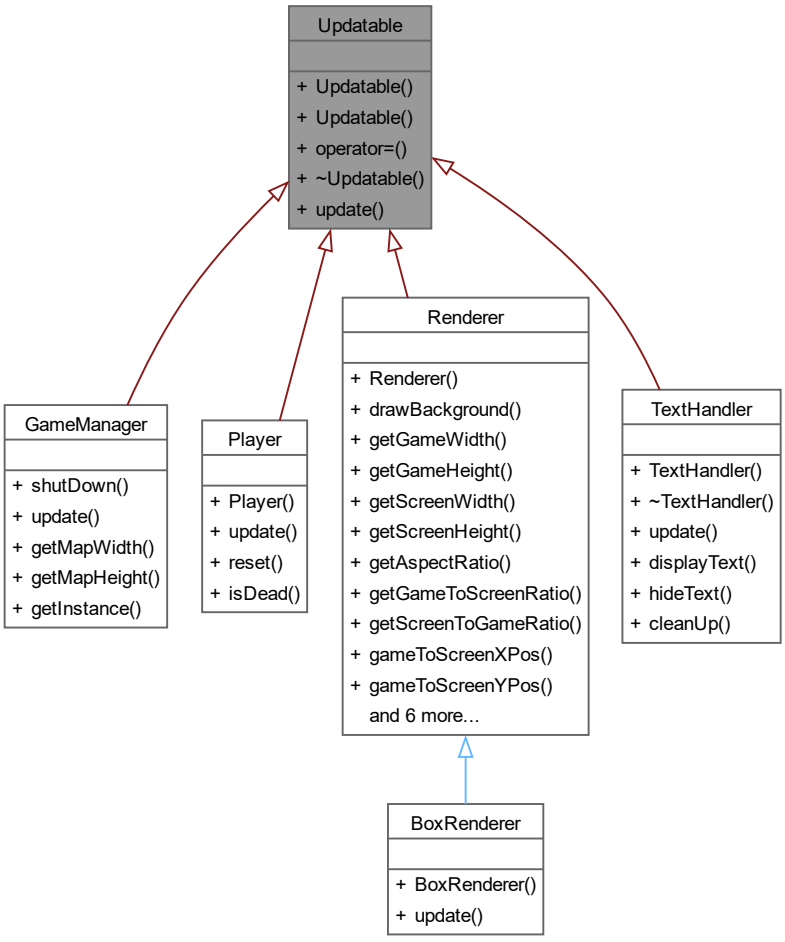
- transform.h
- transform.cpp
- transform.inl

6.19 Updatable Class Reference

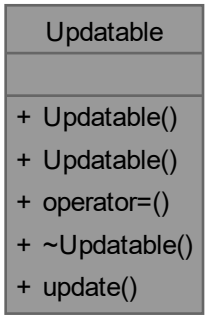
Az updatelhető objektumokat reprezentáló osztály.

`#include <core.h>`

Inheritance diagram for Updatable:



Collaboration diagram for Updatable:



Classes

- class Compare

  Két updatelhető objektum prioritásának összehasonlító osztálya.

Public Member Functions

- Updatable (const UpdatePriority priority)

  Az Updatable osztály konstruktora.
- **Updatable** (const Updatable &updatable)

  Az Updatable osztály másoló konstruktora.
- Updatable & **operator=** (const Updatable &updatable)

  Az Updatable osztály értékadó operátora.
- virtual ~**Updatable** ()

  Az Updatable osztály destruktora.
- virtual void update ()=0

  A frissítést végző virtuális metódus.

6.19.1 Detailed Description

Az updatelhető objektumokat reprezentáló osztály.

Az `Updatable` osztály azokat az objektumokat reprezentálja, amelyek frissíthetőek a játék főciklusában. Az osztályból származtatott objektumok implementálják az `update()` metódust, amely a frissítést végzi.

Az osztály tárolja az objektum frissítési prioritását, amely alapján a `GameRuntime` osztály rendezni tudja az `Updatable` objektumokat.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 Updatable()

```
Updatable::Updatable (
            const UpdatePriority priority)
```

Az Updatable osztály konstruktora.

Parameters

priority	Az updatelés prioritása.

6.19.3 Member Function Documentation

6.19.3.1 update()

```
virtual void Updatable::update ()  [pure virtual]
```

A frissítést végző virtuális metódus.

Implemented in BoxRenderer, GameManager, Player, and TextHandler.

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

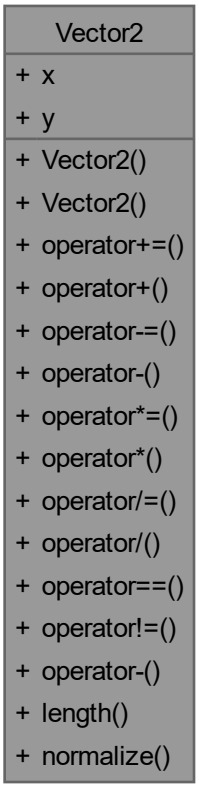
- core.h
- core.cpp

6.20 Vector2 Struct Reference

Két dimenziós vektort reprezentáló struktúra.

`#include <transform.h>`

Collaboration diagram for Vector2:



Public Member Functions

- **Vector2** ()

  Alapértelmezett konstruktor, amely (0, 0) értékekkel inicializálja a vektort.
- Vector2 (const double x, const double y)

  Konstruktor, amely a megadott X és Y értékekkel inicializálja a vektort.
- Vector2 & operator+= (const Vector2 &other)

  Hozzáad egy másik vektort ehhez a vektorhoz.
- Vector2 operator+ (const Vector2 &other) const

  Két vektor összeadása.
- Vector2 & operator-= (const Vector2 &other)

  Kivon egy másik vektort ebből a vektorból.
- Vector2 operator- (const Vector2 &other) const

  Két vektor kivonása.
- Vector2 & operator*= (const double scalar)

  Megszorozza a vektort egy skalárral.
- Vector2 operator* (const double scalar) const

  Egy vektor és egy skalár szorzása.
- Vector2 & operator/= (const double scalar)

  Elosztja a vektort egy skalárral.
- Vector2 operator/ (const double scalar) const

  Egy vektor és egy skalár osztása.
- bool operator== (const Vector2 &other) const

  Összehasonlítja ezt a vektort egy másikkal.
- bool operator!= (const Vector2 &other) const

  Összehasonlítja ezt a vektort egy másikkal.
- Vector2 operator- () const

  Egy vektor megfordítása.
- double length () const

  Kiszámolja a vektor hosszát.
- Vector2 normalize () const

  Normalizálja a vektort.

Public Attributes

- double **x**

  A vektor X komponense.
- double **y**

  A vektor Y komponense.

6.20.1 Detailed Description

Két dimenziós vektort reprezentáló struktúra.

A `Vector2` struktúra X és Y komponensekkel rendelkező vektorokat reprezentál, amelyeket pozíciók, méretek vagy sebességek tárolására használhatunk. Támogatja az alapvető matematikai műveleteket, például összeadást, kivonást, szorzást és osztást.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 Vector2()

```
Vector2::Vector2 (
            const double x,
            const double y)
```

Konstruktor, amely a megadott X és Y értékekkel inicializálja a vektort.

Parameters

x	A vektor X komponense.
y	A vektor Y komponense.

6.20.3 Member Function Documentation

6.20.3.1 length()

```
double Vector2::length () const
```

Kiszámolja a vektor hosszát.

Returns

A vektor hossza.

6.20.3.2 normalize()

```
Vector2 Vector2::normalize () const
```

Normalizálja a vektort.

A normalizálás során a vektor hosszát 1-re állítja, miközben megőrzi az irányát. Ha a vektor hossza 0, akkor a normalizálás nem hajtható végre.

Returns

A normalizált vektor.

6.20.3.3 operator"!=()

```
bool Vector2::operator!= (
            const Vector2 & other) const
```

Összehasonlítja ezt a vektort egy másikkal.

Parameters

other	A másik vektor, amellyel összehasonlítjuk.

Returns

true, ha a két vektor nem egyenlő, különben false.

6.20.3.4 operator∗()

```
Vector2 Vector2::operator* (
            const double scalar) const
```

Egy vektor és egy skalár szorzása.

Parameters

scalar	A szorzó értéke.

Returns

Az új vektor, amely a szorzás eredménye.

6.20.3.5 operator∗=()

```
Vector2 & Vector2::operator*= (
            const double scalar)
```

Megszorozza a vektort egy skalárral.

Parameters

scalar	A szorzó értéke.

Returns

Az aktuális vektor referenciája.

6.20.3.6 operator+()

```
Vector2 Vector2::operator+ (
            const Vector2 & other) const
```

Két vektor összeadása.

Parameters

other	A hozzáadandó vektor.

Returns

Az összeadás eredményeként kapott új vektor.

6.20.3.7 operator+=()

```
Vector2 & Vector2::operator+= (
            const Vector2 & other)
```

Hozzáad egy másik vektort ehhez a vektorhoz.

Parameters

other	A hozzáadandó vektor.

Returns

Az aktuális vektor referenciája.

6.20.3.8 operator-() [1/2]

```
Vector2 Vector2::operator- () const
```

Egy vektor megfordítása.

Returns

A megfordított vektor.

6.20.3.9 operator-() [2/2]

```
Vector2 Vector2::operator- (
            const Vector2 & other) const
```

Két vektor kivonása.

Parameters

other	A kivonandó vektor.

Returns

A kivonás eredményeként kapott új vektor.

6.20.3.10 operator-=()

```
Vector2 & Vector2::operator-= (
            const Vector2 & other)
```

Kivon egy másik vektort ebből a vektorból.

Parameters

other	A kivonandó vektor.

Returns

Az aktuális vektor referenciája.

6.20.3.11 operator/()

```
Vector2 Vector2::operator/ (
            const double scalar) const
```

Egy vektor és egy skalár osztása.

Parameters

scalar	Az osztó értéke.

Returns

Az új vektor, amely az osztás eredménye.

6.20.3.12 operator/=()

```
Vector2 & Vector2::operator/= (
            const double scalar)
```

Elosztja a vektort egy skalárral.

Parameters

scalar	Az osztó értéke.

Returns

Az aktuális vektor referenciája.

6.20.3.13 operator==()

```
bool Vector2::operator== (
            const Vector2 & other) const
```

Összehasonlítja ezt a vektort egy másikkal.

Parameters

other	A másik vektor, amellyel összehasonlítjuk.

Returns

true, ha a két vektor egyenlő, különben false.

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

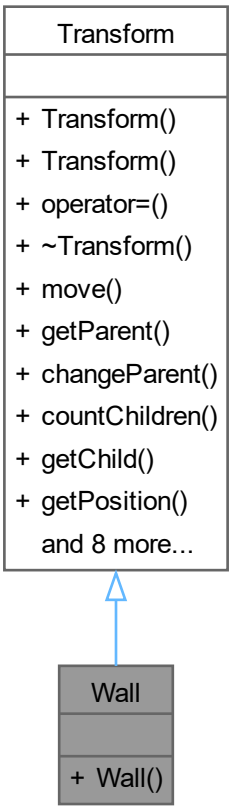
- transform.h
- transform.cpp

6.21 Wall Class Reference

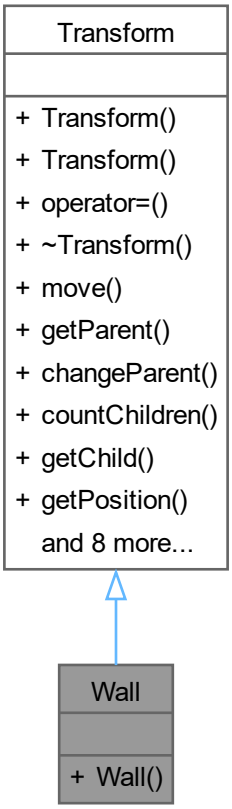
Statikus falat reprezentáló osztály a játék világában.

`#include <wall.h>`

Inheritance diagram for Wall:



Collaboration diagram for Wall:



Public Member Functions

- Wall (const Transform &transform, const Color &color, const double bounciness=0, const Vector2 &collider↩ Ratio={1, 1}, const std::vector< ColliderTag > colliderTags={})

  Létrehoz egy statikus fal objektumot.

Public Member Functions inherited from Transform

- Transform (Transform *const parent=nullptr, const Vector2 &position={ 0, 0 }, const Vector2 &scale={ 1, 1 })

  Konstruktor.
- Transform (const Transform &transform)

  Másoló konstruktor.
- Transform & operator= (const Transform &transform)

  Értékadás operátor.
- virtual ~Transform ()

  Destruktor.
- void move (const Vector2 &offset)

  Az objektum elmozdítása.

- Transform * getParent () const
  Visszaadja az objektum szülőjét.
- void changeParent (Transform *const parent)
  Az objektum szülőjének módosítása.
- size_t countChildren () const
  Visszaadja az objektum gyermekei számát.
- Transform * getChild (const size_t idx) const
  Visszaadja az objektum egy adott gyermekét.
- Vector2 getPosition () const
  Az objektum globális pozíciójának lekérdezése.
- Vector2 getScale () const
  Az objektum globális méretének lekérdezése.
- void setPosition (const Vector2 &position)
  Beállítja az objektum globális pozícióját.
- void setScale (const Vector2 &scale)
  Beállítja az objektum globális méretét.
- Vector2 getLocalPosition () const
  Visszaadja az objektum lokális pozícióját.
- Vector2 getLocalScale () const
  Visszaadja az objektum lokális méretét.
- void setLocalPosition (const Vector2 &position)
  Beállítja az objektum lokális pozícióját.
- void setLocalScale (const Vector2 &scale)
  Beállítja az objektum lokális méretét.
- template<typename T>
  std::vector< T * > findTypeInChildren ()
  Keres egy adott típusú objektumot a gyermekek között.

6.21.1 Detailed Description

Statikus falat reprezentáló osztály a játék világában.

A `Wall` osztály egy statikus falat valósít meg, amely vizuális megjelenítéssel (`BoxRenderer`) rendelkezik, és egy collider segítségével lehetővé teszi, hogy más objektumok nekiütközzenek. A `Transform` osztályból származik, így kezeli a pozíciót és a méretet.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Wall()

```
Wall::Wall (
            const Transform & transform,
            const Color & color,
            const double bounciness = 0,
            const Vector2 & colliderRatio = {1, 1},
            const std::vector< ColliderTag > colliderTags = {})
```

Létrehoz egy statikus fal objektumot.

Inicializálja a fal pozícióját, méretét, színét és ütközési tulajdonságait. A fal vizuálisan megjelenik a játék világában, és részt vesz az ütközésdetektálásban.

Parameters

transform	A fal pozíciója és mérete a játék világában.
color	A fal színe.
bounciness	Az ütközés visszapattanási együtthatója (0 = nincs visszapattanás, 1 = teljes visszapattanás).
colliderRatio	Az ütközési arány, amely meghatározza a collider méretét a fal méretéhez képest.
colliderTags	A collider címkék, amelyek extra tulajdonságokat adnak a colliderhez.

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

- wall.h
- wall.cpp

Chapter 7

File Documentation

7.1 boxrenderer.h

```
00001 #pragma once
00002
00003 #include "renderer.h"
00004
00014 class BoxRenderer : public Renderer
00015 {
00016     private:
00017     Color color;
00018
00019     public:
00032     BoxRenderer(const Transform& transform, const Color& color, const UpdatePriority priority);
00033
00041     void update() override;
00042 };
```

7.2 collider.h

```
00001 #pragma once
00002
00003 #include "transform.h"
00004
00005 #include <vector>
00006
00014 enum class ColliderType
00015 {
00016     INTERACTIVE,
00017     PASSIVE
00018 };
00019
00027 enum class ColliderTag
00028 {
00029     DEADLY,
00030     PLAYER
00031 };
00032
00043 class Collider : public Transform
00044 {
00045     private:
00046     static std::vector<Collider*> colliders;
00047
00048     ColliderType type;
00049     double bounciness;
00050     std::vector<ColliderTag> tags;
00051
00055     void registerCollider();
00056
00060     void unregisterCollider();
00061
00062     public:
00075     Collider(const Transform& transform, const ColliderType type = ColliderType::INTERACTIVE, const
      double bounciness = 0, const std::vector<ColliderTag>& tags = {});
00076
00086     Collider(const Collider& collider);
```

```
00087
00097     Collider& operator=(const Collider& collider);
00098
00104     ~Collider();
00105
00116     static bool checkColliders(const Collider& collider1, const Collider& collider2);
00117
00126     std::vector<Collider*> checkIntersection() const;
00127
00139     static std::vector<Collider*> checkIntersectionForList(const std::vector<Collider*>&
      collidersToCheck);
00140
00151     double getBounciness() const;
00152
00160     bool hasTag(const ColliderTag tag) const;
00161
00169     std::vector<ColliderTag> getTags() const;
00170 };
```

7.3 colors.h

```
00001 #pragma once
00002
00003 #ifndef CPORTA
00004 #include <SDL3/SDL_pixels.h>
00005 #else
00006 struct SDL_Color
00007 {
00008     unsigned char r;
00009     unsigned char g;
00010     unsigned char b;
00011     unsigned char a;
00012 };
00013 #endif
00014
00021 using Color = SDL_Color;
00022
00035 Color makeColor(const unsigned char r, const unsigned char g, const unsigned char b);
```

7.4 core.h

```
00001 #pragma once
00002
00003 #include <set>
00004 #include <unordered_set>
00005
00024 enum class UpdatePriority : int
00025 {
00026     GAME_LOGIC,
00027     PLAYER_RENDERER,
00028     WALL_RENDERER,
00029     UI_RENDERER,
00030     OTHER,
00031 };
00032
00043 class Updatable
00044 {
00045     private:
00046     UpdatePriority priority;
00047
00048     public:
00053     Updatable(const UpdatePriority priority);
00057     Updatable(const Updatable& updatable);
00061     Updatable& operator=(const Updatable& updatable);
00065     virtual ~Updatable();
00066
00070     virtual void update() = 0;
00071
00075     class Compare
00076     {
00077         public:
00078         bool operator()(const Updatable* updatable1, const Updatable* updatable2) const;
00079     };
00080 };
00081
00090 class PhysicsUpdatable
00091 {
00092     public:
```

```
00096     PhysicsUpdatable();
00100     PhysicsUpdatable(const PhysicsUpdatable& updatable);
00104     PhysicsUpdatable& operator=(const PhysicsUpdatable& updatable);
00108     virtual ~PhysicsUpdatable();
00109
00113     virtual void physicsUpdate() = 0;
00114
00122     virtual void postUpdate() = 0;
00123 };
00124
00125 #ifndef CPORTA
00126 struct SDL_Window;
00127 struct SDL_Renderer;
00128
00141 class GameRuntime
00142 {
00143     private:
00144     static SDL_Window* SDLWindow;
00145     static SDL_Renderer* SDLRenderer;
00146
00147     static bool running;
00148
00149     static unsigned long long lastFrameCounter;
00150     static unsigned long long currentFrameCounter;
00151
00152     static double deltaTime;
00153     static double physicsSimTime;
00154
00155     static double targetFrameRate;
00156     static double targetPhysicsRate;
00157
00158     static std::set<Updatable*, Updatable::Compare> updatables;
00159     static std::unordered_set<PhysicsUpdatable*> physicsUpdatables;
00160
00167     static void loop();
00168
00175     static void callUpdates();
00176
00182     static void schedulePhysicsUpdates();
00183
00184     public:
00196     static bool init(const int resolutionX, const int resolutionY);
00197
00205     static void startGameLoop();
00206
00213     static void quit();
00214
00219     static SDL_Renderer* getSDLRenderer();
00220
00225     static double getDeltaTime();
00226
00231     static double getPhysicsDeltaTime();
00232
00237     static void setTargetFrameRate(const double targetRate);
00238
00243     static void setTargetPhysicsRate(const double targetRate);
00244
00249     static void registerForUpdate(Updatable* const updatable);
00250
00255     static void unregisterForUpdate(Updatable* const updatable);
00256
00261     static void registerForUpdate(PhysicsUpdatable* const updatable);
00262
00267     static void unregisterForUpdate(PhysicsUpdatable* const updatable);
00268 };
00269 #endif // CPORTA
00270
00271 #ifdef CPORTA
00272
00276 class GameRuntime
00277 {
00278     static std::set<Updatable*, Updatable::Compare> updatables;
00279     static std::unordered_set<PhysicsUpdatable*> physicsUpdatables;
00280
00281     static double deltaTime;
00282
00283     static double targetFrameRate;
00284     static double targetPhysicsRate;
00285
00286     public:
00290     static void configureMock(const double targetFrameRate, const double targetPhysicsRate);
00291
00297     static void mockUpdate(const size_t calls);
00298
00304     static void mockPhysicsUpdate(const size_t calls);
00305
00310     static double getDeltaTime();
```

```
00311
00316     static double getPhysicsDeltaTime();
00317
00322     static void setTargetFrameRate(const double targetRate);
00323
00328     static void setTargetPhysicsRate(const double targetRate);
00329
00334     static void registerForUpdate(Updatable* const updatable);
00335
00340     static void unregisterForUpdate(Updatable* const updatable);
00341
00346     static void registerForUpdate(PhysicsUpdatable* const updatable);
00347
00352     static void unregisterForUpdate(PhysicsUpdatable* const updatable);
00353 };
00354
00355 #endif // CPORTA
```

7.5 gamemanager.h

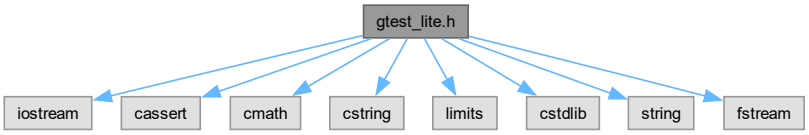
```
00001 #pragma once
00002
00003 #include "texthandler.h"
00004 #include "mapmanager.h"
00005 #include "player.h"
00006
00017 class GameManager : Updatable
00018 {
00019     TextHandler textHandler;
00020     MapManager mapManager;
00021
00022     std::string player1Name = "Blue";
00023     Color player1Color = makeColor(0, 0, 200);
00024     std::string player2Name = "Red";
00025     Color player2Color = makeColor(200, 0, 0);
00026
00027     double resetTime = 0.5;
00028
00029     Player player1;
00030     Player player2;
00031
00032     int player1Score;
00033     int player2Score;
00034     int round;
00035
00036     double timeUntilReset;
00037     bool shouldReset;
00038
00046     GameManager();
00047
00054     GameManager(const GameManager& gameManager);
00055
00062     GameManager& operator=(const GameManager& gameManager);
00063
00072     void checkForWin();
00073
00080     void countReset();
00081
00089     void resetGame();
00090
00100     size_t getMapId() const;
00101
00109     void nextMap();
00110
00111     public:
00120     static GameManager& getInstance();
00121
00129     void shutDown();
00130
00139     void update() override;
00140
00146     double getMapWidth() const;
00147
00153     double getMapHeight() const;
00154 };
```

7.6 gtest_lite.h File Reference

```
#include <iostream>
```

```
#include <cassert>
#include <cmath>
#include <cstring>
#include <limits>
#include <cstdlib>
#include <string>
#include <fstream>
```

Include dependency graph for gtest_lite.h:



Classes

- struct _Is_Types< F, T >

  Segédsablon típuskonverzió futás közbeni ellenőrzésere.
- struct gtest_lite::Test
- class gtest_lite::ostreamRedir

Namespaces

- namespace gtest_lite

  gtest_lite: a keretrendszer függvényinek és objektumainak névtere

Macros

- #define TEST(C, N)
- #define **END** gtest_lite::test.end(); } while (false);

  Teszteset vége.
- #define ENDM gtest_lite::test.end(true); } while (false);
- #define ENDMsg(t)
- #define SUCCEED()

  Sikeres teszt makrója.
- #define FAIL()

  Sikertelen teszt fatális hiba makrója.
- #define ADD_FAILURE()

  Sikertelen teszt makrója.
- #define EXPECT_EQ(expected, actual)

  Azonosságot elváró makró
- #define EXPECT_NE(expected, actual)

  Eltérést elváró makró
- #define EXPECT_LE(expected, actual)

  Kisebb, vagy egyenlő relációt elváró makró
- #define EXPECT_LT(expected, actual)

Kisebb, mint relációt elváró makró

- #define EXPECT_GE(expected, actual)

  Nagyobb, vagy egyenlő relációt elváró makró
- #define EXPECT_GT(expected, actual)

  Nagyobb, mint relációt elváró makró
- #define EXPECT_TRUE(actual)

  Igaz értéket elváró makró
- #define EXPECT_FALSE(actual)

  Hamis értéket elváró makró
- #define EXPECT_FLOAT_EQ(expected, actual)

  Valós számok azonosságát elváró makró
- #define EXPECT_DOUBLE_EQ(expected, actual)

  Valós számok azonosságát elváró makró
- #define EXPECT_STREQ(expected, actual)

  C stringek (const char ∗) azonosságát tesztelő makró
- #define EXPECT_STRNE(expected, actual)

  C stringek (const char ∗) eltéréset tesztelő makró
- #define EXPECT_STRCASEEQ(expected, actual)

  C stringek (const char ∗) azonosságát tesztelő makró (kisbetű/nagybetű azonos)
- #define EXPECT_STRCASENE(expected, actual)

  C stringek (const char ∗) eltéréset tesztelő makró (kisbetű/nagybetű azonos)
- #define EXPECT_THROW(statement, exception_type)

  Kivételt várunk.
- #define EXPECT_ANY_THROW(statement)

  Kivételt várunk.
- #define EXPECT_NO_THROW(statement)

  Nem várunk kivételt.
- #define ASSERT_NO_THROW(statement)

  Nem várunk kivételt.
- #define EXPECT_THROW_THROW(statement, exception_type)

  Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.
- #define EXPECT_ENVEQ(expected, actual)

  Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.
- #define EXPECT_ENVCASEEQ(expected, actual)

  Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)
- #define ASSERT_EQ(expected, actual)

  Azonosságot elváró makró
- #define ASSERT_NO_THROW(statement)

  Nem várunk kivételt.
- #define CREATE_Has_(X)
- #define CREATE_Has_fn_(X, S)
- #define EXPECTTHROW(statement, exp, act)

  EXPECTTHROW: kivételkezelés.
- #define ASSERTTHROW(statement, exp, act)
- #define ASSERT_(expected, actual, fn, op)
- #define **GTINIT**(IS)
- #define **GTEND**(os)

Functions

- void hasMember (...)
- template<typename T1, typename T2>
  std::ostream & **gtest_lite::EXPECT_** (T1 exp, T2 act, bool(∗pred)(T1, T1), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  általános sablon a várt értékhez.
- template<typename T1, typename T2>
  std::ostream & **gtest_lite::EXPECT_** (T1 ∗exp, T2 ∗act, bool(∗pred)(T1 ∗, T1 ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")

  pointerre specializált sablon a várt értékhez.
- std::ostream & gtest_lite::EXPECTSTR (const char ∗exp, const char ∗act, bool(∗pred)(const char ∗, const char ∗), const char ∗file, int line, const char ∗expr, const char ∗lhs="elvart", const char ∗rhs="aktual")
- template<typename T>
  bool gtest_lite::eq (T a, T b)
- bool **gtest_lite::eqstr** (const char ∗a, const char ∗b)
- bool **gtest_lite::eqstrcase** (const char ∗a, const char ∗b)
- template<typename T>
  bool **gtest_lite::ne** (T a, T b)
- bool **gtest_lite::nestr** (const char ∗a, const char ∗b)
- template<typename T>
  bool **gtest_lite::le** (T a, T b)
- template<typename T>
  bool **gtest_lite::lt** (T a, T b)
- template<typename T>
  bool **gtest_lite::ge** (T a, T b)
- template<typename T>
  bool **gtest_lite::gt** (T a, T b)
- template<typename T>
  bool gtest_lite::almostEQ (T a, T b)

7.6.1 Detailed Description

(v4/2022)

Google gtest keretrendszerhez hasonló rendszer. Sz.I. 2015., 2016., 2017. (*Has_X) Sz.I. 2018 (template), ENDM, ENDMsg, nullptr_t Sz.I. 2019 singleton Sz.I. 2021 ASSERT.., STRCASE... Sz.I. 2021 EXPEXT_REGEXP, CREATE_Has_fn*, cmp w. NULL, EXPECT_ param fix V.B., Sz.I. 2022 almostEQ fix, Sz.I. 2022. EXPECT_THROW fix

A tesztelés legalapvetőbb funkcióit támogató függvények és makrók. Nem szálbiztos megvalósítás.

Szabadon felhasználható, bővíthető.

Használati példa: Teszteljük az f(x)=2∗x függvényt: int f(int x) { return 2∗x; }

int main() { TEST(TeszEsetNeve, TesztNeve) EXPECT_EQ(0, f(0)); EXPECT_EQ(4, f(2)) << "A függvény hibás eredményt adott" << std::endl; ... END ... // Fatális hiba esetén a teszteset nem fut tovább. Ezek az ASSERT... makrók. // Nem lehet a kiírásukhoz további üzenetet fűzni. PL: TEST(TeszEsetNeve, TesztNeve) ASSERT_NO_THROW(f(0)); // itt nem lehet << "duma" EXPECT_EQ(4, f(2)) << "A függvény hibás eredményt adott" << std::endl; ... END ...

A működés részleteinek megértése szorgalmi feladat.

7.6.2 Macro Definition Documentation

7.6.2.1 ADD_FAILURE

```
#define ADD_FAILURE()
```

Value:

```
gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()", true)
```

Sikertelen teszt makrója.

7.6.2.2 ASSERT_

```
#define ASSERT_(
                expected,
                actual,
                fn,
                op)
```

Value:

```
    EXPECT_(expected, actual, fn, __FILE__, __LINE__, #op "(" #expected ", " #actual ")" ); \
    if (!gtest_lite::test.status) { gtest_lite::test.end(); break; }
```

7.6.2.3 ASSERT_EQ

```
#define ASSERT_EQ(
                expected,
                actual)
```

Value:

```
gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASSER_EQ")
```

Azonosságot elváró makró
ASSERT típusú ellenőrzések. CSak 1-2 van megvalósítva. Nem ostream& -val térnek vissza !!! Kivételt várunk

7.6.2.4 ASSERT_NO_THROW [1/2]

```
#define ASSERT_NO_THROW(
                statement)
```

Value:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.6.2.5 ASSERT_NO_THROW [2/2]

```
#define ASSERT_NO_THROW(
                statement)
```

Value:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.6.2.6 ASSERTTHROW

```
#define ASSERTTHROW(
                statement,
                exp,
                act)
```

Value:

```
    gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
    « "** Az utasitas " « (act) \
    « "\n** Azt vartuk, hogy " « (exp) « std::endl; if (!gtest_lite::test.status) { gtest_lite::test.end();
      break; }
```

7.6.2.7 CREATE_Has_

```
#define CREATE_Has_(
                X)
```

Value:

```
template<typename T> struct _Has_##X {  \
    struct Fallback { int X; };         \
    struct Derived : T, Fallback {};    \
    template<typename C, C> struct ChT; \
    template<typename D> static char (&f(ChT<int Fallback::*, &D::X>*))[1]; \
    template<typename D> static char (&f(...))[2]; \
    static bool const member = sizeof(f<Derived>(0)) == 2; \
};
```

Segédmakró egy adattag, vagy tagfüggvény létezésének tesztelésére futási időben Ötlet: https://cpptalk.wordpress.com/2009/09/12/substitution-failure-is-not-an-error-2
Használat: CREATE_Has_(size) ... if (_Has_size<std::string>::member)...

7.6.2.8 CREATE_Has_fn_

```
#define CREATE_Has_fn_(
                X,
                S)
```

Value:

```
template<typename R, typename T> struct _Has_fn_##X##_##S {  \
    template<typename C, R (C::*f)() S> struct ChT; \
    template<typename D> static char (&f(ChT<D, &D::X>*))[1]; \
    template<typename D> static char (&f(...))[2]; \
    static bool const fn = sizeof(f<T>(0)) == 1; \
};
```

7.6.2.9 ENDM

```
#define ENDM gtest_lite::test.end(true); } while (false);
```

Teszteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos.

7.6.2.10 ENDMsg

```
#define ENDMsg(
                t)
```

Value:

```
gtest_lite::test.end(true) « t « std::endl; } while (false);
```

Teszteset vége allokált blokkok számának összehasonlításával Ez az ellenőrzés nem bomba biztos. Ha hiba van kiírja az üzenetet.

7.6.2.11 EXPECT_ANY_THROW

```
#define EXPECT_ANY_THROW(
                statement)
```

Value:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (...) { gtest_lite::test.tmp = true; } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott kivetelt.")
```

Kivételt várunk.

7.6.2.12 EXPECT_DOUBLE_EQ

```
#define EXPECT_DOUBLE_EQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_DOUBLE_EQ("
      #expected ", " #actual ")" )
```

Valós számok azonosságát elváró makró

7.6.2.13 EXPECT_ENVCASEEQ

```
#define EXPECT_ENVCASEEQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstrcase, __FILE__, __LINE__,
      "EXPECT_ENVCASEEQ(" #expected ", " #actual ")" )
```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben (kisbetű/nagybetű azonos)

7.6.2.14 EXPECT_ENVEQ

```
#define EXPECT_ENVEQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(std::getenv(expected), actual, gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_ENVEQ("
      #expected ", " #actual ")" )
```

Környezeti változóhoz hasonlít – ilyen nincs a gtest-ben.

7.6.2.15 EXPECT_EQ

```
#define EXPECT_EQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__, __LINE__, "EXPECT_EQ(" #expected ", "
      #actual ")" )
```

Azonosságot elváró makró

7.6.2.16 EXPECT_FALSE

```
#define EXPECT_FALSE(
                actual)
```

Value:

```
gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__, "EXPECT_FALSE(" #actual ")" )
```

Hamis értéket elváró makró

7.6.2.17 EXPECT_FLOAT_EQ

```
#define EXPECT_FLOAT_EQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ, __FILE__, __LINE__, "EXPECT_FLOAT_EQ(" #expected
      ", " #actual ")" )
```

Valós számok azonosságát elváró makró

7.6.2.18 EXPECT_GE

```
#define EXPECT_GE(
```

```
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__, __LINE__, "EXPECT_GE(" #expected ", "
      #actual ")", "etalon" )
```

Nagyobb, vagy egyenlő relációt elváró makró

7.6.2.19 EXPECT_GT

```
#define EXPECT_GT(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__, __LINE__, "EXPECT_GT(" #expected ", "
      #actual ")", "etalon" )
```

Nagyobb, mint relációt elváró makró

7.6.2.20 EXPECT_LE

```
#define EXPECT_LE(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__, __LINE__, "EXPECT_LE(" #expected ", "
      #actual ")", "etalon" )
```

Kisebb, vagy egyenlő relációt elváró makró

7.6.2.21 EXPECT_LT

```
#define EXPECT_LT(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__, __LINE__, "EXPECT_LT(" #expected ", "
      #actual ")", "etalon" )
```

Kisebb, mint relációt elváró makró

7.6.2.22 EXPECT_NE

```
#define EXPECT_NE(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__, __LINE__, "EXPECT_NE(" #expected ", "
      #actual ")", "etalon" )
```

Eltérést elváró makró

7.6.2.23 EXPECT_NO_THROW

```
#define EXPECT_NO_THROW(
                statement)
```

Value:

```
    try { gtest_lite::test.tmp = true; statement; } \
    catch (...) { gtest_lite::test.tmp = false; }\
    EXPECTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
```

Nem várunk kivételt.

7.6.2.24 EXPECT_STRCASEEQ

```
#define EXPECT_STRCASEEQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_STRCASEEQ("
      #expected ", " #actual ")" )
```

C stringek (const char ∗) azonosságát tesztelő makró (kisbetű/nagybetű azonos)

7.6.2.25 EXPECT_STRCASENE

```
#define EXPECT_STRCASENE(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestrcase, __FILE__, __LINE__, "EXPECT_STRCASENE("
      #expected ", " #actual ")", "etalon" )
```

C stringek (const char ∗) eltéréset tesztelő makró (kisbetű/nagybetű azonos)

7.6.2.26 EXPECT_STREQ

```
#define EXPECT_STREQ(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_STREQ(" #expected ",
      " #actual ")" )
```

C stringek (const char ∗) azonosságát tesztelő makró

7.6.2.27 EXPECT_STRNE

```
#define EXPECT_STRNE(
                expected,
                actual)
```

Value:

```
gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr, __FILE__, __LINE__, "EXPECT_STRNE(" #expected ",
      " #actual ")", "etalon" )
```

C stringek (const char ∗) eltéréset tesztelő makró

7.6.2.28 EXPECT_THROW

```
#define EXPECT_THROW(
                statement,
                exception_type)
```

Value:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (exception_type &e) { gtest_lite::test.tmp = true; } \
    catch (...) { } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
```

Kivételt várunk.

7.6.2.29 EXPECT_THROW_THROW

```
#define EXPECT_THROW_THROW(
                statement,
                exception_type)
```

Value:

```
    try { gtest_lite::test.tmp = false; statement; } \
    catch (exception_type &e) { gtest_lite::test.tmp = true; throw; } \
    EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
```

Kivételt várunk és továbbdobjuk – ilyen nincs a gtest-ben.

7.6.2.30 EXPECT_TRUE

```
#define EXPECT_TRUE(
                actual)
```

Value:

```
gtest_lite::EXPECT_(true, actual,  gtest_lite::eq, __FILE__, __LINE__, "EXPECT_TRUE(" #actual ")" )
```

Igaz értéket elváró makró

7.6.2.31 EXPECTTHROW

```
#define EXPECTTHROW(
                statement,
                exp,
                act)
```

Value:

```
    gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__, __LINE__, #statement) \
    « "** Az utasitas " « (act) \
    « "\n** Azt vartuk, hogy " « (exp) « std::endl
```

EXPECTTHROW; kivételkezelés.

Belső megvalósításhoz tartozó makrók, és osztályok.

7.6.2.32 Nem célszerű közvetlenül használni, vagy módosítani

7.6.2.33 FAIL

```
#define FAIL()
```

Value:

```
gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)
```

Sikertelen teszt fatális hiba makrója.

7.6.2.34 SUCCEED

```
#define SUCCEED()
```

Value:

```
gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)
```

Sikeres teszt makrója.

7.6.2.35 TEST

```
#define TEST(
                C,
                N)
```

Value:

```
do { gtest_lite::test.begin(#C"."#N);
```

Teszt kezdete. A makró paraméterezése hasonlít a gtest paraméterezéséhez. Így az itt elkészített testek könnyen átemelhetők a gtest keretrendszerbe.

Parameters

C	- teszteset neve (csak a gtest kompatibilitás miatt van külön neve az eseteknek)
N	- teszt neve

7.6.3 Function Documentation

7.6.3.1 hasMember()

```
void hasMember (
                ...)  [inline]
```

Segédfüggvény egy publikus adattag, vagy tagfüggvény létezésének tesztelésére fordítási időben

7.7 gtest_lite.h

Go to the documentation of this file.

```
00001 #ifndef GTEST_LITE_H
00002 #define GTEST_LITE_H
00003
00044
00045 #include <iostream>
00046 #include <cassert>
00047 #include <cmath>
00048 #include <cstring>
00049 #include <limits>
00050 #include <cstdlib>
00051 #include <string>
00052 #include <fstream>
00053 #if __cplusplus >= 201103L
00054 # include <iterator>
00055 # include <regex>
00056 #endif
00057 #ifdef MEMTRACE
00058 # include "memtrace.h"
00059 #endif
00060
00061 // Két makró az egyes tesztek elé és mögé:
00062 // A két makró a kapcsos zárójelekkel egy új blokkot hoz létre, amiben
00063 // a nevek lokálisak, így elkerülhető a névütközés.
00064
00070 #define TEST(C, N) do { gtest_lite::test.begin(#C"."#N);
00071
00073 #define END gtest_lite::test.end(); } while (false);
00074
00077 #define ENDM gtest_lite::test.end(true); } while (false);
00078
00082 #define ENDMsg(t) gtest_lite::test.end(true) « t « std::endl; } while (false);
00083
00084 // Eredmények vizsgálatát segítő makrók.
00085 // A paraméterek és a funkciók a gtest keretrendszerrel megegyeznek.
00086
00088 #define SUCCEED() gtest_lite::test.expect(true, __FILE__, __LINE__, "SUCCEED()", true)
00089
00091 #define FAIL() gtest_lite::test.expect(false, __FILE__, __LINE__, "FAIL()", true)
00092
00094 #define ADD_FAILURE() gtest_lite::test.expect(false, __FILE__, __LINE__, "ADD_FAILURE()", true)
00095
00097 #define EXPECT_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::eq, __FILE__,
      __LINE__, "EXPECT_EQ(" #expected ", " #actual ")" )
00098
00100 #define EXPECT_NE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ne, __FILE__,
      __LINE__, "EXPECT_NE(" #expected ", " #actual ")", "etalon" )
00101
00103 #define EXPECT_LE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::le, __FILE__,
      __LINE__, "EXPECT_LE(" #expected ", " #actual ")", "etalon" )
00104
00106 #define EXPECT_LT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::lt, __FILE__,
      __LINE__, "EXPECT_LT(" #expected ", " #actual ")", "etalon" )
00107
00109 #define EXPECT_GE(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::ge, __FILE__,
      __LINE__, "EXPECT_GE(" #expected ", " #actual ")", "etalon" )
00110
00112 #define EXPECT_GT(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::gt, __FILE__,
      __LINE__, "EXPECT_GT(" #expected ", " #actual ")", "etalon" )
00113
00115 #define EXPECT_TRUE(actual)  gtest_lite::EXPECT_(true, actual,  gtest_lite::eq, __FILE__, __LINE__,
      "EXPECT_TRUE(" #actual ")" )
00116
00118 #define EXPECT_FALSE(actual) gtest_lite::EXPECT_(false, actual, gtest_lite::eq, __FILE__, __LINE__,
      "EXPECT_FALSE(" #actual ")" )
00119
00121 #define EXPECT_FLOAT_EQ(expected, actual)  gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ,
      __FILE__, __LINE__, "EXPECT_FLOAT_EQ(" #expected ", " #actual ")" )
00122
00124 #define EXPECT_DOUBLE_EQ(expected, actual) gtest_lite::EXPECT_(expected, actual, gtest_lite::almostEQ,
      __FILE__, __LINE__, "EXPECT_DOUBLE_EQ(" #expected ", " #actual ")" )
00125
00127 #define EXPECT_STREQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::eqstr,
      __FILE__, __LINE__, "EXPECT_STREQ(" #expected ", " #actual ")" )
00128
00130 #define EXPECT_STRNE(expected, actual) gtest_lite::EXPECTSTR(expected, actual, gtest_lite::nestr,
      __FILE__, __LINE__, "EXPECT_STRNE(" #expected ", " #actual ")", "etalon" )
00131
00133 #define EXPECT_STRCASEEQ(expected, actual) gtest_lite::EXPECTSTR(expected, actual,
      gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_STRCASEEQ(" #expected ", " #actual ")" )
00134
00136 #define EXPECT_STRCASENE(expected, actual) gtest_lite::EXPECTSTR(expected, actual,
      gtest_lite::nestrcase, __FILE__, __LINE__, "EXPECT_STRCASENE(" #expected ", " #actual ")", "etalon" )
00137
00139 #define EXPECT_THROW(statement, exception_type) try { gtest_lite::test.tmp = false; statement; } \
00140     catch (exception_type &e) { gtest_lite::test.tmp = true; } \
00141     catch (...) { } \
00142     EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
```

```
00143
00145 #define EXPECT_ANY_THROW(statement) try { gtest_lite::test.tmp = false; statement; } \
00146     catch (...) { gtest_lite::test.tmp = true; } \
00147     EXPECTTHROW(statement, "kivetelt dob.", "nem dobott kivetelt.")
00148
00150 #define EXPECT_NO_THROW(statement) try { gtest_lite::test.tmp = true; statement; } \
00151     catch (...) { gtest_lite::test.tmp = false; }\
00152     EXPECTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
00153
00155 #define ASSERT_NO_THROW(statement) try { gtest_lite::test.tmp = true; statement; } \
00156     catch (...) { gtest_lite::test.tmp = false; }\
00157     ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
00158
00160 #define EXPECT_THROW_THROW(statement, exception_type) try { gtest_lite::test.tmp = false; statement; }
    \
00161     catch (exception_type &e) { gtest_lite::test.tmp = true; throw; } \
00162     EXPECTTHROW(statement, "kivetelt dob.", "nem dobott '"#exception_type"' kivetelt.")
00163
00165 #define EXPECT_ENVEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual,
    gtest_lite::eqstr, __FILE__, __LINE__, "EXPECT_ENVEQ(" #expected ", " #actual ")" )
00166
00168 #define EXPECT_ENVCASEEQ(expected, actual) gtest_lite::EXPECTSTR(std::getenv(expected), actual,
    gtest_lite::eqstrcase, __FILE__, __LINE__, "EXPECT_ENVCASEEQ(" #expected ", " #actual ")" )
00169
00170 #if __cplusplus >= 201103L
00172 # define EXPECT_REGEXP(expected, actual, match, err) gtest_lite::EXPECTREGEXP(expected, actual, match,
    err, __FILE__, __LINE__, "EXPECT_REGEXP(" #expected ", " #actual ", " #match ")" )
00173 #endif
00177
00179 #define ASSERT_EQ(expected, actual) gtest_lite::ASSERT_(expected, actual, gtest_lite::eq, "ASSER_EQ")
00180
00182 #define ASSERT_NO_THROW(statement) try { gtest_lite::test.tmp = true; statement; } \
00183     catch (...) { gtest_lite::test.tmp = false; }\
00184     ASSERTTHROW(statement, "nem dob kivetelt.", "kivetelt dobott.")
00185
00186
00193 #define CREATE_Has_(X) \
00194 template<typename T> struct _Has_##X {  \
00195     struct Fallback { int X; };         \
00196     struct Derived : T, Fallback {};    \
00197     template<typename C, C> struct ChT; \
00198     template<typename D> static char (&f(ChT<int Fallback::*, &D::X>*))[1]; \
00199     template<typename D> static char (&f(...))[2]; \
00200     static bool const member = sizeof(f<Derived>(0)) == 2; \
00201 };
00202
00203 #define CREATE_Has_fn_(X, S) \
00204 template<typename R, typename T> struct _Has_fn_##X##_##S {  \
00205     template<typename C, R (C::*f)() S> struct ChT; \
00206     template<typename D> static char (&f(ChT<D, &D::X>*))[1]; \
00207     template<typename D> static char (&f(...))[2]; \
00208     static bool const fn = sizeof(f<T>(0)) == 1; \
00209 };
00210
00213 inline void hasMember(...) {}
00214
00216 template <typename F, typename T>
00217 struct _Is_Types {
00218     template<typename D> static char (&f(D))[1];
00219     template<typename D> static char (&f(...))[2];
00220     static bool const convertable = sizeof(f<T>(F())) == 1;
00221 };
00222
00227
00229 #define EXPECTTHROW(statement, exp, act) gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__,
    __LINE__, #statement) \
00230     « "** Az utasitas " « (act) \
00231     « "\n** Azt vartuk, hogy " « (exp) « std::endl
00232
00233 #define ASSERTTHROW(statement, exp, act) gtest_lite::test.expect(gtest_lite::test.tmp, __FILE__,
    __LINE__, #statement) \
00234     « "** Az utasitas " « (act) \
00235     « "\n** Azt vartuk, hogy " « (exp) « std::endl; if (!gtest_lite::test.status) {
    gtest_lite::test.end(); break; }
00236
00237 #define ASSERT_(expected, actual, fn, op) EXPECT_(expected, actual, fn, __FILE__, __LINE__, #op "("
    #expected ", " #actual ")" ); \
00238     if (!gtest_lite::test.status) { gtest_lite::test.end(); break; }
00239
00240 #ifdef CPORTA
00241 #define GTINIT(is)  \
00242     int magic;      \
00243     is » magic;
00244 #else
00245 #define GTINIT(IS)
00246 #endif // CPORTA
00247
```

```
00248 #ifdef CPORTA
00249 #define GTEND(os)  \
00250     os « magic « (gtest_lite::test.fail() ? " NO" : " OK?") « std::endl;
00251 #else
00252 #define GTEND(os)
00253 #endif // CPORTA
00254
00256 namespace gtest_lite {
00257
00261 struct Test {
00262     int sum;
00263     int failed;
00264     int ablocks;
00265     bool status;
00266     bool tmp;
00267     std::string name;
00268     std::fstream null;
00269     std::ostream& os;
00270     static Test& getTest() {
00271         static Test instance;
00272         return instance;
00273     }
00274 private:
00275     Test() :sum(0), failed(0), status(false), null("/dev/null"), os(std::cout) {}
00276     Test(const Test&);
00277     void operator=(const Test&);
00278 public:
00280     void begin(const char *n) {
00281         name = n; status = true;
00282 #ifdef MEMTRACE
00283         ablocks = memtrace::allocated_blocks();
00284 #endif
00285         os « "\n---> " « name « std::endl;
00286         ++sum;
00287     }
00289     std::ostream& end(bool memchk = false) {
00290 #ifdef MEMTRACE
00291         if (memchk && ablocks != memtrace::allocated_blocks()) {
00292             status = false;
00293             return os « "** Lehet, hogy nem szabaditott fel minden memoriat! **" « std::endl;
00294         }
00295 #endif
00296         os « (status ? "     SIKERES" : "** HIBAS ****") « "\t" « name « " <---" « std::endl;
00297 #ifdef CPORTA
00298         if (!status)
00299             std::cerr « (status ? "     SIKERES" : "** HIBAS ****") « "\t" « name « " <---" «
      std::endl;
00300 #endif // CPORTA
00301         if (!status)
00302             return os;
00303         else
00304             return null;
00305     }
00306
00307     bool fail() { return failed; }
00308
00309     bool astatus() { return status; }
00310
00312     std::ostream& expect(bool st, const char *file, int line, const char *expr, bool pr = false) {
00313         if (!st) {
00314             ++failed;
00315             status = false;
00316         }
00317         if (!st || pr) {
00318             std::string str(file);
00319             size_t i = str.rfind("\\");
00320             if (i == std::string::npos) i = str.rfind("/");
00321             if (i == std::string::npos) i = 0; else i++;
00322             return os « "\n**** " « &file[i] « "(" « line « "): " « expr « " ****" « std::endl;
00323         }
00324         return null;
00325     }
00326
00328     ~Test() {
00329         if (sum != 0) {
00330             os « "\n==== TESZT VEGE ==== HIBAS/OSSZES: " « failed « "/" « sum « std::endl;
00331 #ifdef CPORTA
00332             if (failed)
00333                 std::cerr « "\n==== TESZT VEGE ==== HIBAS/OSSZES: " « failed « "/" « sum « std::endl;
00334 #endif // CPORTA
00335         }
00336     }
00337 };
00338
00341 static Test& test = Test::getTest();
00342
00344 template <typename T1, typename T2>
```

```
00345 std::ostream& EXPECT_(T1 exp, T2 act, bool (*pred)(T1, T1), const char *file, int line,
00346                       const char *expr, const char *lhs = "elvart", const char *rhs = "aktual") {
00347     return test.expect(pred(exp, act), file, line, expr)
00348         « "** " « lhs « ": " « std::boolalpha « exp
00349         « "\n** " « rhs « ": " « std::boolalpha « act « std::endl;
00350 }
00351
00353 template <typename T1, typename T2>
00354 std::ostream& EXPECT_(T1* exp, T2* act, bool (*pred)(T1*, T1*), const char *file, int line,
00355                       const char *expr, const char *lhs = "elvart", const char *rhs = "aktual") {
00356     return test.expect(pred(exp, act), file, line, expr)
00357         « "** " « lhs « ": " « (void*) exp
00358         « "\n** " « rhs « ": " « (void*) act « std::endl;
00359 }
00360
00361 #if __cplusplus >= 201103L
00363 template <typename T>
00364 std::ostream& EXPECT_(std::nullptr_t exp, T* act, bool (*pred)(T*, T*), const char *file, int line,
00365                       const char *expr, const char *lhs = "elvart", const char *rhs = "aktual") {
00366     return test.expect(pred(exp, act), file, line, expr)
00367         « "** " « lhs « ": " « (void*) exp
00368         « "\n** " « rhs « ": " « (void*) act « std::endl;
00369 }
00370
00371 template <typename T>
00372 std::ostream& EXPECT_(T* exp, std::nullptr_t act, bool (*pred)(T*, T*), const char *file, int line,
00373                       const char *expr, const char *lhs = "elvart", const char *rhs = "aktual") {
00374     return test.expect(pred(exp, act), file, line, expr)
00375         « "** " « lhs « ": " « (void*) exp
00376         « "\n** " « rhs « ": " « (void*) act « std::endl;
00377 }
00378 #endif
00379
00382 inline
00383 std::ostream& EXPECTSTR(const char *exp, const char *act, bool (*pred)(const char*, const char*),
    const char *file, int line,
00384                       const char *expr, const char *lhs = "elvart", const char *rhs = "aktual") {
00385     return test.expect(pred(exp, act), file, line, expr)
00386         « "** " « lhs « ": " « (exp == NULL ? "NULL pointer" : std::string("\"") + exp +
    std::string("\""))
00387         « "\n** " « rhs « ": " « (act == NULL ? "NULL pointer" : std::string("\"") + act +
    std::string("\"")) « std::endl;
00388 }
00389
00390 #if __cplusplus >= 201103L
00392 template <typename E, typename S>
00393 int count_regexp(E exp, S str) {
00394     std::regex rexp(exp);
00395     auto w_beg = std::sregex_iterator(str.begin(), str.end(), rexp);
00396     auto w_end = std::sregex_iterator();
00397     return std::distance(w_beg, w_end);
00398 }
00399
00400 template <typename E, typename S>
00401 std::ostream& EXPECTREGEXP(E exp, S str, int match, const char *err,  const char *file, int line,
00402                     const char *expr, const char *lhs = "regexp", const char *rhs = "string",
00403                     const char *m = "elvart/illeszkedik") {
00404     int cnt = count_regexp(exp, str);
00405     if (match < 0) match = cnt;
00406     return test.expect(cnt == match, file, line, expr)
00407         « "** " « lhs « ": " « std::string("\"") + exp + std::string("\"")
00408         « "\n** " « rhs « ": " « (err == NULL ? std::string("\"") + str + std::string("\"") : err)
00409         « "\n** " « m « ": " « match « "/" « cnt « std::endl;
00410 }
00411 #endif
00412
00415 template <typename T>
00416 bool eq(T a, T b) { return a == b; }
00417
00418 inline
00419 bool eqstr(const char *a, const char *b) {
00420     if (a != NULL && b != NULL)
00421         return strcmp(a, b) == 0;
00422     return false;
00423 }
00424
00425 inline
00426 bool eqstrcase(const char *a, const char *b) {
00427     if (a != NULL && b != NULL) {
00428         while (toupper(*a) == toupper(*b) && *a != '\0') {
00429             a++;
00430             b++;
00431         }
00432         return *a == *b;
00433     }
00434     return false;
00435
```

```
00436 }
00437
00438 template <typename T>
00439 bool ne(T a, T b) { return a != b; }
00440
00441 inline
00442 bool nestr(const char *a, const char *b) {
00443     if (a != NULL && b != NULL)
00444         return strcmp(a, b) != 0;
00445     return false;
00446 }
00447
00448 template <typename T>
00449 bool le(T a, T b) { return a <= b; }
00450
00451 template <typename T>
00452 bool lt(T a, T b) { return a < b; }
00453
00454 template <typename T>
00455 bool ge(T a, T b) { return a >= b; }
00456
00457 template <typename T>
00458 bool gt(T a, T b) { return a > b; }
00459
00464 template <typename T>
00465 bool almostEQ(T a, T  b) {
00466     // eps: ha a relatív, vagy abszolút hiba ettől kisebb, akkor elfogadjuk
00467     T eps = 10 * std::numeric_limits<T>::epsilon(); // 10-szer a legkisebb érték
00468     T diff = fabs(a - b);
00469     if (diff < eps)
00470        return true;
00471     T aa = fabs(a);
00472     T ba = fabs(b);
00473     if (aa < ba) {
00474         aa = ba;
00475         ba = fabs(a);
00476     }
00477     return diff < aa * eps;
00478 }
00479
00482 class ostreamRedir {
00483     std::ostream& src;
00484     std::streambuf *const save;
00485 public:
00486     ostreamRedir(std::ostream& src, std::ostream& dst)
00487             : src(src), save(src.rdbuf(dst.rdbuf())) {}
00488     ~ostreamRedir() { src.rdbuf(save); }
00489 };
00490
00491 } // namespace gtest_lite
00492
00493 #endif // GTEST_LITE_H
```

7.8 inputhandler.h

```
00001 #pragma once
00002
00003 #include <SDL3/SDL.h>
00004 #include <unordered_set>
00005
00014 class InputHandler
00015 {
00016     private:
00024     static std::unordered_set<SDL_Keycode> pressedKeys;
00025
00026     public:
00036     static void handleEvent(SDL_Event& event);
00037
00048     static bool isKeyPressed(SDL_Keycode key);
00049 };
```

7.9 inputscheme.h

```
00001 #pragma once
00002
00003 #include <SDL3/SDL.h>
00004
00014 class InputScheme
00015 {
00016     private:
00017     SDL_Keycode jumpKey;
```

```
00018     SDL_Keycode leftKey;
00019     SDL_Keycode dashKey;
00020     SDL_Keycode rightKey;
00021
00022     public:
00035     InputScheme(const SDL_Keycode jumpKey, const SDL_Keycode leftKey, const SDL_Keycode dashKey, const
      SDL_Keycode rightKey);
00036
00045     SDL_Keycode getJumpKey() const;
00046
00055     SDL_Keycode getLeftKey() const;
00056
00065     SDL_Keycode getDashKey() const;
00066
00075     SDL_Keycode getRightKey() const;
00076 };
```

7.10 mapmanager.h

```
00001 #pragma once
00002
00003 #include "transform.h"
00004 #include "colors.h"
00005 #include <vector>
00006
00015 class MapManager
00016 {
00017     private:
00024     struct MapElement
00025     {
00026         Vector2 position;
00027         Vector2 scale;
00028         Color color;
00029         bool deadly;
00030         double bounciness;
00031         Vector2 colliderRatio;
00032
00045         MapElement(const Vector2& position, const Vector2& scale, const Color& color, const bool
      deadly, const double bounciness, const Vector2& colliderRatio);
00046     };
00047
00055     struct Map
00056     {
00057         double mapHeight;
00058         Color backgroundColor;
00059         int scoreToWin;
00060         Vector2 player1Position;
00061         Vector2 player2Position;
00062         std::vector<MapElement> elements;
00063
00074         void addElement(const Vector2& position, const Vector2& scale, const Color& color, const bool
      deadly, const double bounciness, const Vector2& colliderRatio);
00075     };
00076
00077     std::vector<Map> mapCache;
00078     std::vector<Transform*> mapInstance;
00079     size_t loadedMapId;
00080
00081     #ifdef CPORTA
00087     std::vector<std::string> getGamemapFiles() const;
00088     #endif
00089
00090     #ifndef COPRTA
00099     void initializeMap(const Map& map);
00100     #endif
00101
00108     void discardMap();
00109
00110     public:
00117     MapManager();
00118
00119     private:
00126     MapManager(const MapManager& mapManager);
00127
00134     MapManager& operator=(const MapManager& mapManager);
00135
00136     public:
00143     ~MapManager();
00144
00150     size_t getMapCount() const;
00151
00162     Vector2 getPlayerPosition(const size_t playerId) const;
00163
00170     int getScoreToWin() const;
```

```
00171
00178     double getMapHeight() const;
00179
00180     #ifndef CPORTA
00190     void loadMap(const size_t mapId);
00191     #endif
00192
00193     #ifdef CPORTA
00201     std::string getSerializedMapInfo(const size_t mapId);
00202
00210     std::string getSerializedMapElement(const size_t mapId, const size_t elementId);
00211     #endif
00212 };
```

7.11 memtrace.h

```
00001 /*********************************
00002 Memoriaszivargas-detektor
00003 Keszitette: Peregi Tamas, BME IIT, 2011
00004             petamas@iit.bme.hu
00005 Kanari:     Szeberenyi Imre, 2013.,
00006 VS 2012:    Szeberényi Imre, 2015.,
00007 mem_dump:   2016.
00008 inclue-ok:  2017., 2018., 2019., 2021., 2022.
00009 *********************************/
00010
00011 #ifndef MEMTRACE_H
00012 #define MEMTRACE_H
00013
00014 #if defined(MEMTRACE)
00015
00016 /*ha definiálva van, akkor a hibakat ebbe a fajlba írja, egyébkent stderr-re*/
00017 /*#define MEMTRACE_ERRFILE MEMTRACE.ERR*/
00018
00019 /*ha definialva van, akkor futas kozben lancolt listat epit. Javasolt a hasznalata*/
00020 #define MEMTRACE_TO_MEMORY
00021
00022 /*ha definialva van, akkor futas kozben fajlba irja a foglalasokat*/
00023 /*ekkor nincs ellenorzes, csak naplozas*/
00024 /*#define MEMTRACE_TO_FILE*/
00025
00026 /*ha definialva van, akkor a megallaskor automatikus riport keszul */
00027 #define MEMTRACE_AUTO
00028
00029 /*ha definialva van, akkor malloc()/calloc()/realloc()/free() kovetve lesz*/
00030 #define MEMTRACE_C
00031
00032 #ifdef MEMTRACE_C
00033     /*ha definialva van, akkor free(NULL) nem okoz hibat*/
00034     #define ALLOW_FREE_NULL
00035 #endif
00036
00037 #ifdef __cplusplus
00038     /*ha definialva van, akkor new/delete/new[]/delete[] kovetve lesz*/
00039     #define MEMTRACE_CPP
00040 #endif
00041
00042 #if defined(__cplusplus) && defined(MEMTRACE_TO_MEMORY)
00043     /*ha definialva van, akkor atexit helyett objektumot hasznal*/
00044     /*ajanlott bekapcsolni*/
00045     #define USE_ATEXIT_OBJECT
00046 #endif
00047
00048 /*****************************************/
00049 /* INNEN NE MODOSITSD                    */
00050 /*****************************************/
00051 #ifdef NO_MEMTRACE_TO_FILE
00052     #undef MEMTRACE_TO_FILE
00053 #endif
00054
00055 #ifdef NO_MEMTRACE_TO_MEMORY
00056     #undef MEMTRACE_TO_MEMORY
00057 #endif
00058
00059 #ifndef MEMTRACE_AUTO
00060     #undef USE_ATEXIT_OBJECT
00061 #endif
00062
00063 #ifdef __cplusplus
00064     #define START_NAMESPACE namespace memtrace {
00065     #define END_NAMESPACE } /*namespace*/
00066     #define TRACEC(func) memtrace::func
00067     #include <new>
00068 #else
```

```
00069     #define START_NAMESPACE
00070     #define END_NAMESPACE
00071     #define TRACEC(func) func
00072 #endif
00073
00074 // THROW deklaráció változatai
00075 #if defined(_MSC_VER)
00076   // VS rosszul kezeli az __cplusplus makrot
00077   #if _MSC_VER < 1900
00078     // * nem biztos, hogy jó így *
00079     #define THROW_BADALLOC
00080     #define THROW_NOTHING
00081   #else
00082     // C++11 vagy újabb
00083     #define THROW_BADALLOC noexcept(false)
00084     #define THROW_NOTHING noexcept
00085   #endif
00086 #else
00087   #if __cplusplus < 201103L
00088     // C++2003 vagy régebbi
00089     #define THROW_BADALLOC throw (std::bad_alloc)
00090     #define THROW_NOTHING throw ()
00091   #else
00092     // C++11 vagy újabb
00093     #define THROW_BADALLOC noexcept(false)
00094     #define THROW_NOTHING noexcept
00095   #endif
00096 #endif
00097
00098 START_NAMESPACE
00099     int allocated_blocks();
00100 END_NAMESPACE
00101
00102 #if defined(MEMTRACE_TO_MEMORY)
00103 START_NAMESPACE
00104     int mem_check(void);
00105     int poi_check(void*);
00106 END_NAMESPACE
00107 #endif
00108
00109 #if defined(MEMTRACE_TO_MEMORY) && defined(USE_ATEXIT_OBJECT)
00110 #include <cstdio>
00111 START_NAMESPACE
00112     class atexit_class {
00113         private:
00114             static int counter;
00115             static int err;
00116         public:
00117             atexit_class() {
00118 #if defined(CPORTA) && !defined(CPORTA_NOSETBUF)
00119                 if (counter == 0) {
00120                         setbuf(stdout, 0);
00121                         setbuf(stderr, 0);
00122                 }
00123 #endif
00124             counter++;
00125             }
00126
00127             int check() {
00128                 if(--counter == 0)
00129                     err = mem_check();
00130                 return err;
00131             }
00132
00133             ~atexit_class() {
00134                 check();
00135             }
00136     };
00137
00138 static atexit_class atexit_obj;
00139
00140 END_NAMESPACE
00141 #endif/*MEMTRACE_TO_MEMORY && USE_ATEXIT_OBJECT*/
00142
00143 /*Innentol csak a "normal" include eseten kell, kulonben osszezavarja a mukodest*/
00144 #ifndef FROM_MEMTRACE_CPP
00145 #include <stdlib.h>
00146 #ifdef __cplusplus
00147     #include <iostream>
00148 /* ide gyűjtjük a nemtrace-vel összeakadó headereket, hogy előbb legyenek */
00149
00150     #include <fstream>  // VS 2013 headerjében van deleted definició
00151     #include <sstream>
00152     #include <vector>
00153     #include <list>
00154     #include <map>
00155     #include <algorithm>
```

```
00156     #include <functional>
00157     #include <memory>
00158     #include <iomanip>
00159     #include <locale>
00160     #include <typeinfo>
00161     #include <ostream>
00162     #include <stdexcept>
00163     #include <ctime>
00164     #include <random>
00165     #if __cplusplus >= 201103L
00166         #include <iterator>
00167         #include <regex>
00168     #endif
00169 #endif
00170 #ifdef MEMTRACE_CPP
00171     namespace std {
00172         typedef void (*new_handler)();
00173 }
00174 #endif
00175
00176 #ifdef MEMTRACE_C
00177 START_NAMESPACE
00178     #undef malloc
00179     #define malloc(size) TRACEC(traced_malloc)(size,#size,__LINE__,__FILE__)
00180     void * traced_malloc(size_t size, const char *size_txt, int line, const char * file);
00181
00182     #undef calloc
00183     #define calloc(count,size) TRACEC(traced_calloc)(count, size, #count","#size,__LINE__,__FILE__)
00184     void * traced_calloc(size_t count, size_t size, const char *size_txt, int line, const char *
    file);
00185
00186     #undef free
00187     #define free(p) TRACEC(traced_free)(p, #p,__LINE__,__FILE__)
00188     void traced_free(void * p, const char *size_txt, int line, const char * file);
00189
00190     #undef realloc
00191     #define realloc(old,size) TRACEC(traced_realloc)(old,size,#size,__LINE__,__FILE__)
00192     void * traced_realloc(void * old, size_t size, const char *size_txt, int line, const char * file);
00193
00194     void mem_dump(void const *mem, size_t size, FILE* fp = stdout);
00195
00196 END_NAMESPACE
00197 #endif/*MEMTRACE_C*/
00198
00199 #ifdef MEMTRACE_CPP
00200 START_NAMESPACE
00201     #undef set_new_handler
00202     #define set_new_handler(f) TRACEC(_set_new_handler)(f)
00203     void _set_new_handler(std::new_handler h);
00204
00205     void set_delete_call(int line, const char * file);
00206 END_NAMESPACE
00207
00208 void * operator new(size_t size, int line, const char * file) THROW_BADALLOC;
00209 void * operator new[](size_t size, int line, const char * file) THROW_BADALLOC;
00210 void * operator new(size_t size) THROW_BADALLOC;
00211 void * operator new[](size_t size) THROW_BADALLOC;
00212 void operator delete(void * p)  THROW_NOTHING;
00213 void operator delete[](void * p) THROW_NOTHING;
00214
00215 #if __cplusplus >= 201402L
00216 // sized delete miatt: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2013/n3536.html
00217 void operator delete(void * p, size_t)  THROW_NOTHING;
00218 void operator delete[](void * p, size_t) THROW_NOTHING;
00219 #endif
00220
00221 /* Visual C++ 2012 miatt kell, mert háklis, hogy nincs megfelelő delete, bár senki sem használja */
00222 void operator delete(void *p, int, const char *) THROW_NOTHING;
00223 void operator delete[](void *p, int, const char *) THROW_NOTHING;
00224
00225
00226 #define new new(__LINE__, __FILE__)
00227 #define delete memtrace::set_delete_call(__LINE__, __FILE__),delete
00228
00229 #ifdef CPORTA
00230 #define system(...)  // system(__VA_ARGS__)
00231 #endif
00232
00233 #endif /*MEMTRACE_CPP*/
00234
00235 #endif /*FROM_MEMTRACE_CPP*/
00236 #else
00237 #pragma message ( "MEMTRACE NOT DEFINED" )
00238 #endif /*MEMTRACE*/
00239
00240 #endif /*MEMTRACE_H*/
```

7.12 physicsObject.h

```
00001 #pragma once
00002
00003 #include "core.h"
00004 #include "collider.h"
00005
00006 #include <unordered_set>
00007
00017 class PhysicsObject : public Transform, PhysicsUpdatable
00018 {
00019     private:
00020     static const double maxIntersectionResolveDistance;
00021     static const size_t intersectionResolvePasses;
00022     static const size_t physicsPasses;
00023     static const bool searchChildrenForColliders;
00024
00025     Vector2 velocity;
00026     Vector2 acceleration;
00027     Vector2 gravity;
00028
00029     std::pair<double, double> maxXVelocity;
00030     std::pair<double, double> maxYVelocity;
00031
00032     std::vector<Collider*> colliders;
00033
00034     std::unordered_set<ColliderTag> touchedTags;
00035
00044     void addTagsFromCollider(const Collider* collider);
00045
00052     void tryResolveIntersections();
00053
00065     Vector2 tryResolveIntersectionsInDirection(const Vector2& offset);
00066
00083     void tryCheck(const Vector2& offset, bool& didIntersect, double& maxBounciness,
      std::vector<Collider*>& intersects);
00084
00085     public:
00096     PhysicsObject(const Transform& transform, const std::vector<Collider*>& colliders);
00097
00105     void physicsUpdate() override;
00106
00110     void postUpdate() override;
00111
00128     bool tryTeleport(const Vector2& position);
00129
00135     Vector2 getVelocity() const;
00136
00142     void setVelocity(const Vector2& velocity);
00143
00149     Vector2 getAcceleration() const;
00150
00156     void setAcceleration(const Vector2& acceleration);
00157
00163     void setGravity(const Vector2& gravity);
00164
00170     void setMaxXVelocity(const std::pair<double, double>& maxXVelocity);
00171
00177     void setMaxYVelocity(const std::pair<double, double>& maxYVelocity);
00178
00185     bool checkTag(const ColliderTag tag) const;
00186
00193     void clearTags();
00194 };
```

7.13 player.h

```
00001 #pragma once
00002
00003 #include "boxrenderer.h"
00004 #include "physicsObject.h"
00005 #include "inputscheme.h"
00006
00007 #include <string>
00008
00017 class Player : public PhysicsObject, Updatable
00018 {
00019     private:
00020     BoxRenderer renderer;
00021     Collider collider;
00022     Collider groundCheck;
00023     Collider headCheck;
00024
```

```
00025     const InputScheme inputScheme;
00026
00027     double jumpSpeed = 10.5;
00028     double dashSpeed = 20;
00029     double moveAcceleration = 20;
00030     double turnAcceleration = 55;
00031     double maxSpeed = 8;
00032     double maxFallSpeed = 30;
00033     double floatGravity = 1.2 * 9.81;
00034     double fallGravity = 2.2 * 9.81;
00035     double groundDrag = 30;
00036     double airDrag = 10;
00037
00038     bool hasDied;
00039
00046     void controlPlayer();
00047
00054     void boundsCheck();
00055
00065     bool checkDeath() const;
00066
00080     bool isGrounded() const;
00081
00082     public:
00091     Player(const Color& color, const InputScheme& inputScheme);
00092
00099     void update() override;
00100
00110     void reset(const Vector2& resetPosition);
00111
00117     bool isDead() const;
00118 };
```

7.14 renderer.h

```
00001 #pragma once
00002
00003 #include "core.h"
00004 #include "transform.h"
00005 #include "colors.h"
00006
00014 class Renderer : public Transform, Updatable
00015 {
00016     private:
00017     static double gameHeight;
00018     static Color backgroundColor; //< A játék világának háttérszíne (RGBA formátumban).
00019
00020     static Vector2 cameraOffset;
00021     static double cameraScale;
00022
00023     public:
00032     Renderer(const Transform& transform, UpdatePriority priority);
00033
00039     static void drawBackground();
00040
00048     static double getGameWidth();
00049
00055     static double getGameHeight();
00056
00062     static int getScreenWidth();
00063
00069     static int getScreenHeight();
00070
00078     static double getAspectRatio();
00079
00087     static double getGameToScreenRatio();
00088
00096     static double getScreenToGameRatio();
00097
00104     static int gameToScreenXPos(const double gameXPos);
00105
00112     static int gameToScreenYPos(const double gameYPos);
00113
00120     static double screenToGameXPos(const int screenXPos);
00121
00128     static double screenToGameYPos(const int screenYPos);
00129
00137     static void setGameHeight(const double gameHeight);
00138
00144     static void setBackgroundColor(const Color color);
00145
00151     static void setCameraOffset(const Vector2& offset);
00152
00158     static void setCameraScale(const double scale);
```

```
00159 };
```

7.15 test.h

```
00001
00004 class TestRunner
00005 {
00006     public:
00007     static void start();
00008
00009     static void runVector2Tests();
00010
00011     static void runTransformTests();
00012
00013     static void runColliderTests();
00014
00015     static void runPhysicsTests();
00016
00017     static void runMapTests();
00018 };
```

7.16 texthandler.h

```
00001 #pragma once
00002 #include "memtrace.h"
00003
00004 #include "core.h"
00005 #include "colors.h"
00006
00007 #include <SDL3_ttf/SDL_ttf.h>
00008 #include <string>
00009
00018 class TextHandler : Updatable
00019 {
00020     std::string text;
00021     Color color;
00022     double textScale = 0.2;
00023     bool shouldDisplay;
00024     TTF_Font* font;
00025
00026     int lastScreenHeight;
00027
00035     void renderText();
00036
00046     void loadFont(const double fontSize);
00047
00054     void unloadFont();
00055
00056     public:
00062     TextHandler();
00063
00070     ~TextHandler();
00071
00078     void update() override;
00079
00089     void displayText(const std::string& text, const Color& color);
00090
00096     void hideText();
00097
00104     void cleanUp();
00105 };
```

7.17 transform.h

```
00001 #pragma once
00002 #include "memtrace.h"
00003
00004 #include <vector>
00005
00014 struct Vector2
00015 {
00016     double x;
00017     double y;
00018
00022     Vector2();
00023
00030     Vector2(const double x, const double y);
00031
```

```
00038     Vector2& operator+=(const Vector2& other);
00039
00046     Vector2 operator+(const Vector2& other) const;
00047
00054     Vector2& operator-=(const Vector2& other);
00055
00062     Vector2 operator-(const Vector2& other) const;
00063
00070     Vector2& operator*=(const double scalar);
00071
00078     Vector2 operator*(const double scalar) const;
00079
00086     Vector2& operator/=(const double scalar);
00087
00094     Vector2 operator/(const double scalar) const;
00095
00102     bool operator==(const Vector2& other) const;
00103
00110     bool operator!=(const Vector2& other) const;
00111
00117     Vector2 operator-() const;
00118
00124     double length() const;
00125
00134     Vector2 normalize() const;
00135 };
00136
00144 class Transform
00145 {
00146     private:
00147     Vector2 position;
00148     Vector2 scale;
00149
00150     Transform* parent;
00151     std::vector<Transform*> children;
00152
00162     void addChild(Transform* const child);
00163
00173     void removeChild(Transform* const child);
00174
00175     public:
00185     Transform(Transform* const parent = nullptr, const Vector2& position = { 0, 0 }, const Vector2&
      scale = { 1, 1 });
00186
00199     Transform(const Transform& transform);
00200
00214     Transform& operator=(const Transform& transform);
00215
00231     virtual ~Transform();
00232
00240     void move(const Vector2& offset);
00241
00247     Transform* getParent() const;
00248
00258     void changeParent(Transform* const parent);
00259
00265     size_t countChildren() const;
00266
00273     Transform* getChild(const size_t idx) const;
00274
00282     Vector2 getPosition() const;
00283
00291     Vector2 getScale() const;
00292
00302     void setPosition(const Vector2& position);
00303
00313     void setScale(const Vector2& scale);
00314
00320     Vector2 getLocalPosition() const;
00321
00327     Vector2 getLocalScale() const;
00328
00334     void setLocalPosition(const Vector2& position);
00335
00341     void setLocalScale(const Vector2& scale);
00342
00349     template <typename T>
00350     std::vector<T*> findTypeInChildren();
00351 };
00352
00353 #include "transform.inl"
```

7.18 transform.inl

```
00001 #pragma once
00002
00003 #include <queue>
00004
00005 template <typename T>
00006 std::vector<T*> Transform::findTypeInChildren()
00007 {
00008     std::queue<Transform*> toCheck = std::queue<Transform*>();
00009     std::vector<T*> found = std::vector<T*>();
00010
00011     toCheck.push(this);
00012     while (!toCheck.empty())
00013     {
00014         T* target = dynamic_cast<T*>(toCheck.front());
00015         if (target != nullptr)
00016             found.push_back(target);
00017
00018         for (size_t i = 0; i < toCheck.front()->countChildren(); i++)
00019         {
00020             toCheck.push(toCheck.front()->getChild(i));
00021         }
00022
00023         toCheck.pop();
00024     }
00025
00026     return found;
00027 }
```

7.19 wall.h

```
00001 #pragma once
00002 #include "memtrace.h"
00003
00004 #include "boxrenderer.h"
00005 #include "collider.h"
00006
00015 class Wall : public Transform
00016 {
00017     private:
00018     BoxRenderer renderer;
00019     Collider collider;
00020
00021     public:
00034     Wall(const Transform& transform, const Color& color, const double bounciness = 0, const Vector2&
      colliderRatio = {1, 1}, const std::vector<ColliderTag> colliderTags = {});
00035 };
```

Index