

Sokoban

Wygenerowano przez Doxygen 1.9.3

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja struktury Animation	7
4.2 Dokumentacja klasy ColliderComponent	7
4.2.1 Dokumentacja funkcji składowych	8
4.2.1.1 draw()	8
4.2.1.2 init()	8
4.2.1.3 update()	8
4.3 Dokumentacja klasy Collision	8
4.4 Dokumentacja klasy Component	9
4.5 Dokumentacja klasy ComponentMove	9
4.6 Dokumentacja klasy Entity	10
4.7 Dokumentacja klasy Game	10
4.8 Dokumentacja klasy KeyboardControl	12
4.8.1 Dokumentacja funkcji składowych	12
4.8.1.1 init()	12
4.8.1.2 update()	12
4.9 Dokumentacja klasy Manager	13
4.10 Dokumentacja klasy Map	13
4.11 Dokumentacja klasy Menu	13
4.12 Dokumentacja klasy ScoreBoard	14
4.13 Dokumentacja klasy SpriteComponent	15
4.13.1 Dokumentacja funkcji składowych	15
4.13.1.1 draw()	15
4.13.1.2 init()	15
4.13.1.3 update()	16
4.14 Dokumentacja klasy Text	16
4.15 Dokumentacja klasy TextureManager	16
4.16 Dokumentacja klasy TileComponent	16
4.16.1 Dokumentacja funkcji składowych	17
4.16.1.1 init()	17
4.17 Dokumentacja klasy Timer	17
4.18 Dokumentacja klasy TransformComponent	18
4.18.1 Dokumentacja funkcji składowych	18
4.18.1.1 init()	18

4.18.1.2 update()	18
4.19 Dokumentacja klasy UI	19
4.19.1 Dokumentacja funkcji składowych	19
4.19.1.1 draw()	19
4.20 Dokumentacja klasy Vector2D	19
5 Dokumentacja plików	21
5.1 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Animation.h	21
5.2 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ColliderComponent.h	21
5.3 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Collision.h	22
5.4 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentMove.h	22
5.5 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Components.h	23
5.6 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.h	23
5.7 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Game.h	25
5.8 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/GameObject.h	27
5.9 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/KeyboardControl.h	27
5.10 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Map.h	28
5.11 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Menu.h	28
5.12 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ScoreBoard.h	29
5.13 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/SpriteComponent.h	30
5.14 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Text.h	31
5.15 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TextureManager.h	31
5.16 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TileComponent.h	32
5.17 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Timer.h	32
5.18 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TransformComponent.h	33
5.19 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/UI.h	34
5.20 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Vector2D.h	35
Indeks	37

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Animation	7
Collision	8
Component	9
ColliderComponent	7
KeyboardControl	12
Menu	13
ScoreBoard	14
SpriteComponent	15
TileComponent	16
TransformComponent	18
UI	19
ComponentMove	9
Entity	10
Game	10
Manager	13
Map	13
Text	16
TextureManager	16
Timer	17
Vector2D	19

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Animation	7
ColliderComponent	7
Collision	8
Component	9
ComponentMove	9
Entity	10
Game	10
KeyboardControl	12
Manager	13
Map	13
Menu	13
ScoreBoard	14
SpriteComponent	15
Text	16
TextureManager	16
TileComponent	16
Timer	17
TransformComponent	18
UI	19
Vector2D	19

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Animation.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	ColliderComponent.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Collision.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	ComponentMove.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Components.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	ComponentSystem.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Game.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	GameObject.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	KeyboardControl.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Map.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Menu.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	ScoreBoard.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	SpriteComponent.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Text.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	TextureManager.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	TileComponent.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Timer.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	TransformComponent.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	UI.h	??
C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/	Vector2D.h	??

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja struktury Animation

Metody publiczne

- **Animation** (int i, int f, int s)

Atrybuty publiczne

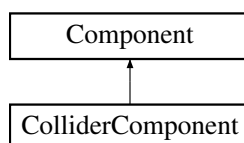
- int **index**
- int **frames**
- int **speed**

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Animation.h

4.2 Dokumentacja klasy ColliderComponent

Diagram dziedziczenia dla ColliderComponent



Metody publiczne

- **ColliderComponent** (std::string t)
- **ColliderComponent** (std::string t, int xpos, int ypos, int size)
- void **init** () override
- void **update** () override
- void **draw** () override

Atrybuty publiczne

- `std::string tag`
- `SDL_Rect collider`
- `SDL_Texture * tex`
- `SDL_Rect srcR`
- `SDL_Rect destR`
- [TransformComponent](#) * `transform`

4.2.1 Dokumentacja funkcji składowych

4.2.1.1 draw()

```
void ColliderComponent::draw ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.2.1.2 init()

```
void ColliderComponent::init ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.2.1.3 update()

```
void ColliderComponent::update ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- `C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ColliderComponent.h`

4.3 Dokumentacja klasy Collision

Statyczne metody publiczne

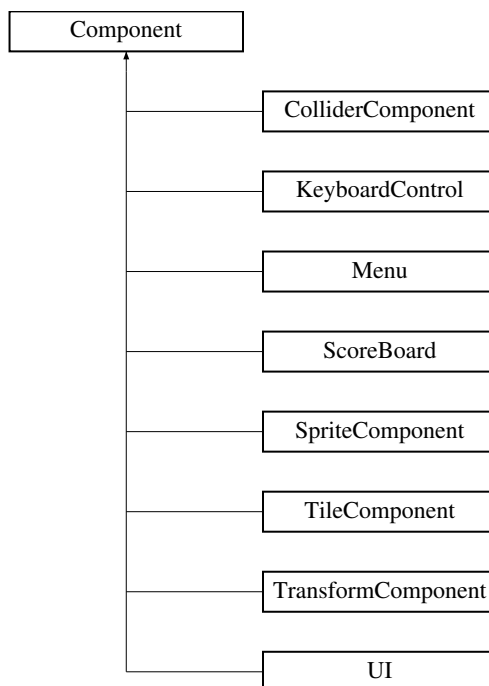
- static bool **AABB** (const `SDL_Rect` &recA, const `SDL_Rect` &recB)
- static bool **AABB** (const [ColliderComponent](#) &colA, const [ColliderComponent](#) &colB)

Dokumentacja dla tej klasy została wygenerowana z plików:

- `C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Collision.h`
- `C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Collision.cpp`

4.4 Dokumentacja klasy Component

Diagram dziedziczenia dla Component



Metody publiczne

- virtual void **init** ()
- virtual void **update** ()
- virtual void **update** ([Entity](#) &Start)
- virtual void **draw** ()

Atrybuty publiczne

- [Entity](#) * entity

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.h

4.5 Dokumentacja klasy ComponentMove

Metody publiczne

- **ComponentMove** ([Game](#) *game, [Timer](#) *mTimer)
- void **update** (int counter, class [Entity](#) &Player, class std::vector< [Entity](#) * > &boxcolliders, class std::vector< [Entity](#) * > &points, class std::vector< [Entity](#) * > &colliders, class [Entity](#) &ui, class [Entity](#) &End_game_0, class [Entity](#) &End_game_1, class [Entity](#) &End_game_2)

Atrybuty publiczne

- `Timer` * `mTimer`
- `Game` * `game`
- `Text` * `textUI`

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentMove.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentMove.cpp

4.6 Dokumentacja klasy Entity

Metody publiczne

- **Entity** (`Manager` &`mManager`)
- void **update** ()
- void **draw** ()
- bool **isActive** () const
- void **destroy** ()
- bool **hasGroup** (Group `mGroup`)
- void **addGroup** (Group `mGroup`)
- void **delGroup** (Group `mGroup`)
- template<typename T >
bool **hasComponent** () const
- template<typename T , typename... TArgs>
T & **addComponent** (TArgs &&... `mArgs`)
- template<typename T >
T & **getComponent** () const

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.cpp

4.7 Dokumentacja klasy Game

Typy publiczne

- enum **groupLabels** : `std::size_t` {
 groupMap , **groupPlayer** , **groupMenu** , **groupBoxColliders** ,
 groupPoints , **groupScoreboard** , **groupColliders** }

Metody publiczne

- void **init** (const char *title, int xpos, int ypos, int width, int height, bool fullscreen)
- void **handleEvents** ()
- void **update** ()
- void **render** ()
- void **clean** ()
- bool **running** ()
- void **Mapstate** (bool mapstate_inside)
- void **Scoreboardstate** (bool Scoreboard_inside)
- void **bool_counter_state** (bool bool_counter_inside)
- void **End_game** ()

Atrybuty publiczne

- bool **mapstate**
- bool **Scoreboard_state**
- bool **bool_counter** = false
- std::string **str_Player_H_S_1**
- std::string **str_Player_H_S_2**
- std::string **str_Player_H_S_3**
- std::string **str_Player_H_S_4**
- std::string **str_Player_H_S_5**
- std::string **str_Player_time_H_S_1**
- std::string **str_Player_time_H_S_2**
- std::string **str_Player_time_H_S_3**
- std::string **str_Player_time_H_S_4**
- std::string **str_Player_time_H_S_5**
- std::string **Timer**
- std::string **Timer_after**
- std::string **Text_End_game_0** = "Brawo twoj czas wynosi: "
- std::string **Text_End_game_1** = "By zapisac sie w tabeli wynikow "
- std::string **Text_End_game_2** = "wpisz swój nick w konsoli"
- std::string **Text_End_game_3** = "i kliknij Enter"
- std::string **Text_End_game_4** = "ESC -> powrot"
- std::string **nick**

Statyczne atrybuty publiczne

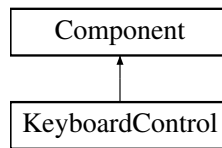
- static SDL_Renderer * **renderer** = nullptr
- static Text * **textUI** = new Text(&manager)
- static SDL_Event **event**
- static SDL_Color **white**

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Game.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Game.cpp

4.8 Dokumentacja klasy KeyboardControl

Diagram dziedziczenia dla KeyboardControl



Metody publiczne

- void [init](#) () override
- void [update](#) () override

Atrybuty publiczne

- [TransformComponent](#) * **transform**
- [SpriteComponent](#) * **sprite**

4.8.1 Dokumentacja funkcji składowych

4.8.1.1 init()

```
void KeyboardControl::init ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.8.1.2 update()

```
void KeyboardControl::update ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/KeyboardControl.h

4.9 Dokumentacja klasy Manager

Metody publiczne

- void **update** ()
- void **draw** ()
- void **refresh** ()
- void **AddToGroup** (Entity *mEntity, Group mGroup)
- std::vector< Entity * > & **getGroup** (Group mGroup)
- Entity & **addEntity** ()

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.h

4.10 Dokumentacja klasy Map

Metody publiczne

- void **LoadMap** (std::string path, int sizeX, int sizeY)
- void **AddTile** (int id, int x, int y)

Atrybuty publiczne

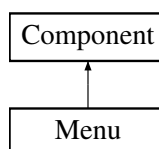
- std::string **map_file**

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Map.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Map.cpp

4.11 Dokumentacja klasy Menu

Diagram dziedziczenia dla Menu



Metody publiczne

- **Menu** (Game *game)
- **Menu** (bool isAnimated)
- void **Options** (Entity &Start, Entity &ScoreB, Entity &Quit)
- void **update** (bool mapstate, class std::vector< Entity * > menu, Entity &Start, Entity &ScoreB, Entity &Quit, Entity &Player)

Atrybuty publiczne

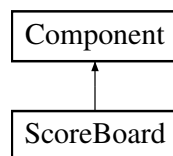
- [SpriteComponent](#) * **sprite**
- [Map](#) * **map**
- [Game](#) * **game**
- [Timer](#) * **mTimer**
- bool **Menuanimation** = false

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Menu.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Menu.cpp

4.12 Dokumentacja klasy ScoreBoard

Diagram dziedziczenia dla ScoreBoard



Metody publiczne

- **ScoreBoard** ([Game](#) *game)
- **ScoreBoard** (bool isAnimated)
- void **Options** ([Entity](#) &H_S, [Entity](#) &Name, [Entity](#) &Time_H_S, [Entity](#) &Back)
- void **update** (class std::vector< [Entity](#) * > Scoreboard_g, [Entity](#) &Back)
- void **loadScoreBoard** (std::string SB_file_players, std::string SB_file_time)

Atrybuty publiczne

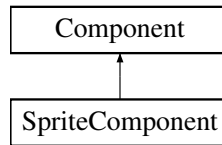
- [SpriteComponent](#) * **sprite**
- [Map](#) * **map**
- [Game](#) * **game**
- [Timer](#) * **mTimer**
- bool **Menuanimation** = false
- std::string **SB_file_players**
- std::string **SB_file_time**

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ScoreBoard.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ScoreBoard.cpp

4.13 Dokumentacja klasy SpriteComponent

Diagram dziedziczenia dla SpriteComponent



Metody publiczne

- **SpriteComponent** (const char *path)
- **SpriteComponent** (const char *path, bool isAnimated)
- void **setTexture** (const char *path)
- void **init** () override
- void **update** () override
- void **draw** () override
- void **play** (const char *animName)

Atrybuty publiczne

- int **animIndex** = 0
- std::map< const char *, [Animation](#) > **animations**
- SDL_RendererFlip **spriteFlip** = SDL_FLIP_NONE

4.13.1 Dokumentacja funkcji składowych

4.13.1.1 draw()

```
void SpriteComponent::draw ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.13.1.2 init()

```
void SpriteComponent::init ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.13.1.3 update()

```
void SpriteComponent::update ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/SpriteComponent.h

4.14 Dokumentacja klasy Text

Metody publiczne

- **Text** ([Manager](#) *man)
- void **AddFonts** (std::string id, std::string path, int fontSize)
- TTF_Font * **GetFont** (std::string id)

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Text.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Text.cpp

4.15 Dokumentacja klasy TextureManager

Statyczne metody publiczne

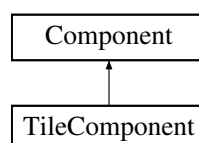
- static SDL_Texture * **LoadTexture** (const char *fileName)
- static SDL_Texture * **toTexture** (SDL_Surface *surface, int destroySurface)
- static void **Draw** (SDL_Texture *tex, SDL_Rect src, SDL_Rect dest, SDL_RendererFlip flip)

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TextureManager.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TextureManager.cpp

4.16 Dokumentacja klasy TileComponent

Diagram dziedziczenia dla TileComponent



Metody publiczne

- **TileComponent** (int x, int y, int w, int h, int id)
- void **init** () override

Atrybuty publiczne

- **TransformComponent** * **transform**
- **SpriteComponent** * **sprite**
- SDL_Rect **tileRect**
- int **tileID**
- const char * **path**

4.16.1 Dokumentacja funkcji składowych

4.16.1.1 init()

```
void TileComponent::init ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TileComponent.h

4.17 Dokumentacja klasy Timer

Metody publiczne

- void **Reset** ()
- void **ResetMin** ()
- int **deltaTime** ()
- void **TimeScale** (float t)
- int **TimeScale** ()
- void **Update** ()
- unsigned int **Hour** ()
- unsigned int **Minute** ()
- void **AddHour** ()
- void **AddMinute** ()

Statyczne metody publiczne

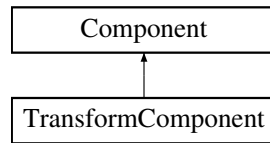
- static **Timer** * **Instance** ()
- static void **Release** ()

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Timer.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Timer.cpp

4.18 Dokumentacja klasy TransformComponent

Diagram dziedziczenia dla TransformComponent



Metody publiczne

- **TransformComponent** (std::string t)
- **TransformComponent** (int sc)
- **TransformComponent** (float x, float y)
- **TransformComponent** (int x, int y, int h, int w, int sc)
- void **init** () override
- void **update** () override

Atrybuty publiczne

- **Vector2D** **position**
- **Vector2D** **velocity**
- int **height** = 100
- int **width** = 100
- int **scale** = 1
- std::string **tag**
- int **speed** = 3

4.18.1 Dokumentacja funkcji składowych

4.18.1.1 init()

```
void TransformComponent::init ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

4.18.1.2 update()

```
void TransformComponent::update ( ) [inline], [override], [virtual]
```

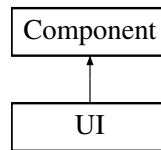
Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TransformComponent.h

4.19 Dokumentacja klasy UI

Diagram dziedziczenia dla UI



Metody publiczne

- **UI** (int xpos, int ypos, std::string text, std::string font, SDL_Color &color)
- void **SetText** (std::string text, std::string font)
- void **draw** () override
- void **Free_All** ()

Dodatkowe Dziedziczone Składowe

4.19.1 Dokumentacja funkcji składowych

4.19.1.1 draw()

```
void UI::draw ( ) [inline], [override], [virtual]
```

Reimplementowana z [Component](#).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/UI.h

4.20 Dokumentacja klasy Vector2D

Metody publiczne

- **Vector2D** (float x, float y)
- **Vector2D** & **Zero** ()
- **Vector2D** & **Add** (const **Vector2D** &vec)
- **Vector2D** & **Subtract** (const **Vector2D** &vec)
- **Vector2D** & **Multiply** (const **Vector2D** &vec)
- **Vector2D** & **operator*** (const int &i)
- **Vector2D** & **operator+=** (const **Vector2D** &vec)
- **Vector2D** & **operator-=** (const **Vector2D** &vec)

Atrybuty publiczne

- float **x**
- float **y**

Przyjaciele

- [Vector2D](#) & **operator+** ([Vector2D](#) &v1, const [Vector2D](#) &v2)
- [Vector2D](#) & **operator-** ([Vector2D](#) &v1, const [Vector2D](#) &v2)
- std::ostream & **operator<<** (std::ostream &stream, const [Vector2D](#) &vec)

Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Vector2D.h
- C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Vector2D.cpp

Rozdział 5

Dokumentacja plików

5.1 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/↵ Sokoban/Animation.h

```
1 #pragma once
2
3 /** Struktura zawierajaca dane dot. Animacji
4  /** index przypisany numer
5  /** frames ilosc klatek
6  /** speed szybkość wyświetlania
7  /** konstruktor zawierajacy wszystkie te zmienne
8 struct Animation
9 {
10     int index;
11     int frames;
12     int speed;
13
14     Animation() {
15         Animation::index = 0;
16         Animation::frames = 0;
17         Animation::speed = 0;
18     }
19     Animation(int i, int f, int s) {
20         index = i;
21         frames = f;
22         speed = s;
23     }
24 };
```

5.2 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/↵ Sokoban/ColliderComponent.h

```
1 #pragma once
2 #include "string"
3 #include "SDL.h"
4 #include "ComponentSystem.h"
5 #include "TransformComponent.h"
6 #include "TextureManager.h"
7
8 /** klasa dzięki której możliwa jest kolizja obiektów
9 class ColliderComponent : public Component {
10 public:
11     /** zienna dot. nazwy
12     std::string tag;
13
14     /** przywołanie zmiennych z biblioteki graficznej dot. tekstur
15     SDL_Rect collider;
16     SDL_Texture* tex;
17     SDL_Rect srcR, destR;
18
19     /** wskaznik do klasy TransformComponent
20     TransformComponent* transform;
21
22     /** konstruktor z zawartoscia
```

```

23     /** nazwa
24     ColliderComponent(std::string t) {
25         tag = t;
26     }
27
28     /** konstruktow z zawartoscia
29     /** nazwa
30     /** pozycja w osi x
31     /** pozycja w osi y
32     /** wielkosc
33     ColliderComponent(std::string t, int xpos, int ypos, int size)
34     {
35         tag = t;
36         collider.x = xpos;
37         collider.y = ypos;
38         collider.h = collider.w = size;
39     }
40
41     /** funkcja inicjalizacyjna, odwołująca się do klasy virtualnej bazowej
42     void init() override {
43         if (!entity->hasComponent<TransformComponent>()) {
44             entity->addComponent<TransformComponent>();
45         }
46         transform = &entity->getComponent<TransformComponent>();
47     }
48
49
50     /** funkcja odświeżająca kolizje, odwołująca się do klasy virtualnej bazowej
51     void update() override {
52         if (tag != "wall" && tag != "box") {
53             collider.x = static_cast<int>(transform->position.x);
54             collider.y = static_cast<int>(transform->position.y);
55             collider.w = transform->width * transform->scale;
56             collider.h = transform->height * transform->scale;
57         }
58     }
59
60
61     /** funkcja wyświetlająca teksturę, odwołująca się do klasy virtualnej bazowej
62     void draw() override
63     {
64         TextureManager::Draw(tex, srcR, destR, SDL_FLIP_NONE);
65     }
66
67 };

```

5.3 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/↵ Sokoban/Collision.h

```

1 #pragma once
2 #include "SDL.h"
3 class ColliderComponent;
4
5 /** klasa dot. kolizji i jej obliczania
6 class Collision
7 {
8 public:
9     /** funkcja z wzorcem kolizji w oknie
10    /** zawiera odwołania do zmiennych biblioteki graficznej dot. okna
11    static bool AABB(const SDL_Rect& recA, const SDL_Rect& recB);
12
13    /** funkcja z wzorcem kolizji pomiędzy obiektami
14    /** zawiera odwołania do klasy definiującej obiekty kolizyjne
15    static bool AABB(const ColliderComponent& colA, const ColliderComponent& colB);
16 };

```

5.4 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/↵ Sokoban/ComponentMove.h

```

1 #pragma once
2 #include "Collision.h"
3 #include "ComponentSystem.h"
4 #include "Timer.h"
5 #include "UI.h"
6 #include "Game.h"
7

```

```

8 extern Manager manager;
9
10 /** klasa dot. poruszania sie obiektow
11 class ComponentMove{
12 public:
13     Timer* mTimer;
14     Game* game;
15     Text* textUI;
16
17     ComponentMove(Game* game, Timer* mTimer);
18
19     ~ComponentMove();
20
21     /** funkcja odswierzajaca
22     /** zawierajaca odwołania do obiektow oraz grup z obiektami
23     void update(int counter, class Entity& Player, class std::vector<Entity*> &boxcolliders, class
24         std::vector<Entity*> &points, class std::vector<Entity*> &colliders, class Entity& ui, class Entity&
25         End_game_0, class Entity& End_game_1, class Entity& End_game_2);
26
27 private:
28 };

```

5.5 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Components.h

```

1 #pragma once
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <string.h>
5 #include <math.h>
6
7 #include "ComponentSystem.h"
8 #include "SpriteComponent.h"
9 #include "TransformComponent.h"
10 #include "KeyboardControl.h"
11 #include "ColliderComponent.h"

```

5.6 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ComponentSystem.h

```

1 #pragma once
2 #include <iostream>
3 #include <vector>
4 #include <memory>
5 #include <algorithm>
6 #include <bitset>
7 #include <array>
8
9 /** Tworzenie klas bez definicji
10 class Component;
11 class Entity;
12 class Manager;
13
14 /** ID obiektu
15 using ComponentID = std::size_t;
16 using Group = std::size_t;
17
18 /** Wstawianie ID obiektu w wybrane miejsce
19 inline ComponentID getNewComponentTypeID() {
20     static ComponentID lastID = 0u;
21     return lastID++;
22 }
23
24 /** Szablon do pobierania ID obiektu
25 template <typename T> inline ComponentID GetComponentTypeID() noexcept {
26     static_assert (std::is_base_of<Component, T>::value, "");
27     static ComponentID typeID = getNewComponentTypeID();
28     return typeID;
29 }
30
31 /** Maksymalna liczba przetrzymywania Komponentow oraz Grup
32 constexpr std::size_t maxComponents = 64;
33 constexpr std::size_t maxGroups = 64;
34
35 using ComponentBitSet = std::bitset<maxComponents>;

```

```

36 using GroupBitset = std::bitset<maxGroups>;
37
38 using ComponentArray = std::array<Component*, maxComponents>;
39
40 /** Definicja klasy wirtualnej do tworzenia komponentow - polimorfizm
41 class Component {
42 public:
43     Entity* entity;
44
45     virtual void init() {}
46     virtual void update() {}
47     virtual void update(Entity& Start) {}
48     virtual void draw() {}
49     virtual ~Component() {}
50 };
51
52 /** Definicja klasy
53 class Entity {
54 private:
55     Manager& manager;
56     /** okreslanie stanu
57     bool active = true;
58     /**wektor komponentow z dynamiczna wielkoscia
59     std::vector<std::unique_ptr<Component>> components;
60
61     ComponentArray componentArray;
62     ComponentBitSet componentBitset;
63     GroupBitset groupBitset;
64
65 public:
66     Entity(Manager& mManager) : manager(mManager) {}
67
68     /** funkcja odswierzajaca
69     void update() {
70         for (auto& c : components) c->update();
71     }
72
73     /** funkcja renderujaca
74     void draw() {
75         for (auto& c : components) c->draw();
76     }
77
78     /** funkcja sprawdzajaca aktywnosc
79     bool isActive() const { return active; }
80
81     /** funkcja niszczaca/wylaczajaca obiekt
82     void destroy() { active = false; }
83
84     /** funkcja sprawdzajaca grupe obiketu
85     bool hasGroup(Group mGroup) {
86         return groupBitset[mGroup];
87     }
88
89     /** funkcja dodajaca obiekt do grupy
90     void addGroup(Group mGroup);
91
92     /** funkcja usuwajaca obiekt z grupy
93     void delGroup(Group mGroup) {
94         groupBitset[mGroup] = false;
95     }
96
97     /** szablon do sprawdzania czy dany obiekt posiada dany komponent
98     template <typename T> bool hasComponent() const {
99         return componentBitset[getComponentTypeID<T>()];
100     }
101
102     /** szablon dodawnia do obiektu dany komponent
103     template <typename T, typename... TArgs>
104     T& addComponent(TArgs&&... mArgs) {
105         T* c(new T(std::forward<TArgs>(mArgs)...));
106         c->entity = this;
107         std::unique_ptr<Component>uPtr{ c };
108         components.emplace_back(std::move(uPtr));
109
110         componentArray[getComponentTypeID<T>()] = c;
111         componentBitset[getComponentTypeID<T>()] = true;
112
113         c->init();
114         return *c;
115     }
116
117     /** szablon pobierajacy z obiektu dany komponent
118     template<typename T> T& getComponent() const {
119         auto ptr(componentArray[getComponentTypeID<T>()]);
120         return *static_cast<T*>(ptr);
121     }
122

```

```

123
124 };
125
126 /** Definicja klasy
127 class Manager {
128 private:
129     std::vector<std::unique_ptr<Entity>> entities;
130     std::array<std::vector<Entity*>, maxGroups> groupedEntities;
131 public:
132
133     /** funkcja odswierzajaca
134     void update() {
135         for (auto& e : entities) e->update();
136     }
137
138     /** funkcja renderujaca
139     void draw() {
140         for (auto& e : entities) e->draw();
141     }
142
143     /** funkcja odswierzajaca
144     void refresh() {
145         for (auto i(0u); i < maxGroups; i++)
146         {
147             auto& v(groupedEntities[i]);
148             v.erase(
149                 std::remove_if(std::begin(v), std::end(v),
150                     [i](Entity* mEntity)
151                     {
152                         return !mEntity->isActive() || !mEntity->hasGroup(i);
153                     })),
154                 std::end(v));
155         }
156
157         entities.erase(std::remove_if(std::begin(entities), std::end(entities),
158             [](const std::unique_ptr<Entity>& mEntity)
159             {
160                 return !mEntity->isActive();
161             })),
162             std::end(entities));
163     }
164
165     /** funkcja dodawania do grupy
166     void AddToGroup(Entity* mEntity, Group mGroup) {
167         groupedEntities[mGroup].emplace_back(mEntity);
168     }
169
170     std::vector<Entity*>& getGroup(Group mGroup) {
171         return groupedEntities[mGroup];
172     }
173
174     Entity& addEntity() {
175         Entity* e = new Entity(*this);
176         std::unique_ptr<Entity> uPtr{ e };
177         entities.emplace_back(std::move(uPtr));
178         return *e;
179     }
180 };

```

5.7 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Game.h

```

1 #pragma once
2 #include "SDL.h"
3 #include "SDL_ttf.h"
4 #include "SDL_image.h"
5 #include <iostream>
6 #include <fstream>
7 #include <vector>
8 #include "Text.h"
9
10 /** klasa podstawa silnika gry
11 class Game {
12 public:
13     static SDL_Renderer* renderer;
14     static Text* textUI;
15     Game();
16
17     /**Inicjalizacja funkcji czyszczacej pamieci po zakonczeniu programu
18     ~Game();
19
20     /** funkcja inicjalizujaca okno programu

```

```

21     /** title - nazwa okna
22     /** xpos - pozycja w osi x
23     /** ypos - pozycja w osi y
24     /** width - szerokosc
25     /** height - wysokosc
26     /** fullscreen - definiowanie okna pelnoekranowego
27     void init(const char* title, int xpos, int ypos, int width, int height, bool fullscreen);
28
29     /** funkcja wykrycia wydarzenia zwiazanym z oknem
30     void handleEvents();
31
32     /** funkcja odswierzanie okna
33     void update();
34
35     /** funkcja renderujaca obiekty
36     void render();
37
38     /** funkcja czyszczaca pamiec po zakonczeniu programu
39     void clean();
40
41     /** funkcja wskazujaceja na dzialanie programu
42     bool running() { return isRunning; }
43
44     /** przywołanie zmiennych z biblioteki graficznej dot. tekstur oraz wydarzen
45
46     static SDL_Event event;
47     static SDL_Color white;
48
49     /** obecny stan generowania mapy
50     bool mapstate;
51     bool Scoreboard_state;
52     bool bool_counter = false;
53
54     /** zmienne do zapisu rekordu (nazwy oraz czasu gracza) tablicy wynikow
55     std::string str_Player_H_S_1;
56     std::string str_Player_H_S_2;
57     std::string str_Player_H_S_3;
58     std::string str_Player_H_S_4;
59     std::string str_Player_H_S_5;
60     std::string str_Player_time_H_S_1;
61     std::string str_Player_time_H_S_2;
62     std::string str_Player_time_H_S_3;
63     std::string str_Player_time_H_S_4;
64     std::string str_Player_time_H_S_5;
65
66     /**zapis zegara
67     std::string Timer;
68     std::string Timer_after;
69
70     /** zmienne do zapisu tekstu dla konca gry
71     std::string Text_End_game_0 = "Brawo twoj czas wynosi: ";
72     std::string Text_End_game_1 = "By zapisac sie w tabeli wynikow ";
73     std::string Text_End_game_2 = "wpisz swoj nick w konsoli";
74     std::string Text_End_game_3 = "i kliknij Enter";
75     std::string Text_End_game_4 = "ESC -> powrot";
76
77     /** zmienna do zapisu nazwy gracza
78     std::string nick;
79
80     /** funkcja zmiany stanu generowania mapy, licznika oraz tablicy wynikow
81     void Mapstate(bool mapstate_inside);
82     void Scoreboardstate(bool Scoreboard_inside);
83     void bool_counter_state(bool bool_counter_inside);
84
85     /** funkcja wywolujaca koniec gry
86     void End_game();
87
88     /** inicjalizacja grup dla pozniej tworzonych obiektow
89     enum groupLabels : std::size_t
90     {
91         groupMap,
92         groupPlayer,
93         groupMenu,
94         groupBoxColliders,
95         groupPoints,
96         groupScoreboard,
97         groupColliders
98     };
99
100 private:
101     SDL_Event ev;
102
103     /** zmienna dot. dzialania okna
104     bool isRunning = true;
105
106     /**licznik
107     int counter = 2;

```

```

108
109     /* zmienna okna
110     SDL_Window* window;
111 };

```

5.8 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/GameObject.h

```

1 // #pragma once
2 // #include "Game.h"
3 //
4 // class GameObject {
5 // public:
6 //     GameObject(const char* texturesheet, int x, int y);
7 //     ~GameObject();
8 //
9 //     void Update();
10 //     void Render();
11 // private:
12 //     int xpos;
13 //     int ypos;
14 //
15 //     SDL_Texture* ObjTexture;
16 //     SDL_Rect srcRect, destRect;
17 //     SDL_Renderer* renderer;
18 //
19 //
20 // };

```

5.9 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/KeyboardControl.h

```

1 #pragma once
2 #include "Game.h"
3 #include "Components.h"
4 #include "TextureManager.h"
5 #include "Map.h"
6 #include "TileComponent.h"
7 #include "ComponentMove.h"
8
9 /* klasa pozwalajaca na kontrole postaci za pomoca klawiatury
10 /* z klasa bazowa Component
11 class KeyboardControl : public Component {
12 public:
13     /* wskazniki do klas
14     TransformComponent* transform;
15     SpriteComponent* sprite;
16
17     /* funkcja inicjalizacyjna, odwołująca się do klasy virtualnej bazowej
18     void init() override {
19         transform = &entity->GetComponent<TransformComponent>();
20         sprite = &entity->GetComponent<SpriteComponent>();
21     }
22
23     /* funkcja odświeżająca kliknięcie, odwołująca się do klasy virtualnej bazowej
24     void update() override {
25         if (Game::event.type == SDL_KEYDOWN) {
26             switch (Game::event.key.keysym.sym)
27             {
28                 case SDLK_w:
29                     transform->velocity.y = -1;
30                     sprite->play("Walk");
31                     break;
32                 case SDLK_s:
33                     transform->velocity.y = 1;
34                     sprite->play("Walk");
35                     break;
36                 case SDLK_a:
37                     transform->velocity.x = -1;
38                     sprite->play("Walk");
39                     sprite->spriteFlip = SDL_FLIP_HORIZONTAL;
40                     break;
41                 case SDLK_d:
42                     transform->velocity.x = 1;
43                     sprite->play("Walk");
44                     sprite->spriteFlip = SDL_FLIP_NONE;
45                     break;

```

```

46         default:
47             break;
48     }
49 }
50 if (Game::event.type == SDL_KEYUP) {
51     switch (Game::event.key.keysym.sym)
52     {
53     case SDLK_w:
54         transform->velocity.y = 0;
55         sprite->play("Idle");
56         break;
57     case SDLK_s:
58         transform->velocity.y = 0;
59         sprite->play("Idle");
60         break;
61     case SDLK_a:
62         transform->velocity.x = 0;
63         sprite->play("Idle");
64         break;
65     case SDLK_d:
66         transform->velocity.x = 0;
67         sprite->play("Idle");
68         break;
69     default:
70         break;
71     }
72 }
73 }
74 };

```

5.10 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Map.h

```

1 #pragma once
2 #include <string>
3
4 /** klasa generujaca mape wyswietlana
5 class Map {
6 public:
7     Map();
8     ~Map();
9
10     /** sciezka wczytywanego pliku z tablica
11     std::string map_file;
12
13     /** funkcja wczytujaca plik i rozpoznajaca elementy wyswietlane
14     /** path - sciezka pliku
15     /** sizeX - definiowanie wilkosci w osi x
16     /** sizeY - definiowanie wilkosci w osi y
17     void LoadMap(std::string path, int sizeX, int sizeY);
18
19     /** funkcja tworząca kafelek wyswietlany
20     /** id - przypisany numer
21     /** x - pozycja osi x
22     /** y - pozycja osi y
23     void AddTile(int id, int x, int y);
24
25 private:
26 };

```

5.11 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Menu.h

```

1 #pragma once
2 #include <string>
3 #include "ComponentSystem.h"
4 #include "Components.h"
5 #include "SpriteComponent.h"
6
7 /** rozszerzenie dzialania szablonu Manager
8 extern Manager manager;
9
10 /** klasa zawierajaca dzialanie menu
11 /** z klasa bazowa Component
12 class Menu : public Component {
13 public:

```



```

14
15     /** wskazniki do klas
16     SpriteComponent* sprite;
17     Map* map;
18     Game* game;
19     Timer* mTimer;
20
21     /** zmienan wskazujaca stan animacji obiektu
22     bool Menuanimation = false;
23
24     Menu();
25     Menu( Game* game);
26
27     /** kolstruktor przypisujacy stan do zmiennej
28     /** isAnimated - wskazanie stanu
29     Menu(bool isAnimated) {
30         Menuanimation = isAnimated;
31     }
32     ~Menu();
33
34     /** funkcja z zdefiniowanymi opcjami menu
35     /** odwołania do obiektów
36     void Options(Entity& Start, Entity& ScoreB, Entity& Quit);
37
38
39     /** funkcja odswierzajaca stan opcji wybranej
40     /** odwołania do obiektów oraz ich grupy
41     void update(bool mapstate, class std::vector<Entity*> menu, Entity& Start, Entity& ScoreB, Entity&
42         Quit, Entity& Player);
43 };

```

5.12 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/ScoreBoard.h

```

1 #pragma once
2 #include <string>
3 #include <fstream>
4 #include "Menu.h"
5 #include "Components.h"
6 #include "SpriteComponent.h"
7
8 /** rozszerzenie dzialania szablonu Manager
9 extern Manager manager;
10
11 /** klasa zawierajaca dzialanie menu
12 /** z klasa bazowa Component
13 class ScoreBoard : public Component {
14 public:
15
16     /** wskazniki do klas
17     SpriteComponent* sprite;
18     Map* map;
19     Game* game;
20     Timer* mTimer;
21
22     /** zmienna okreslajaca stan animacji obiektu
23     bool Menuanimation = false;
24
25     /** zmienne do zapisu sciezki plikow
26     std::string SB_file_players;
27     std::string SB_file_time;
28
29     ScoreBoard();
30     ScoreBoard(Game* game);
31
32     /** kolstruktor przypisujacy stan do zmiennej
33     /** isAnimated - wskazanie stanu
34     ScoreBoard(bool isAnimated) {
35         Menuanimation = isAnimated;
36     }
37     ~ScoreBoard();
38
39     /** funkcja z zdefiniowanymi opcjami menu
40     /** odwołania do obiektów
41     void Options(Entity& H_S, Entity& Name, Entity& Time_H_S, Entity& Back);
42
43
44     /** funkcja odswierzajaca stan opcji wybranej
45     /** odwołania do obiektów oraz ich grupy
46     void update(class std::vector<Entity*> Scoreboard_g, Entity& Back);
47
48     /** funkcja ladujaca z plikow dane dot. tabeli wyników

```

```

49     /** SB_file_players -> sciezka pliku z nazwami graczy
50     /** SB_file_time -> sciezka pliku z czasami graczy
51     void loadScoreBoard(std::string SB_file_players, std::string SB_file_time);
52 };

```

5.13 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/SpriteComponent.h

```

1 #pragma once
2 #include "TransformComponent.h"
3 #include "TextureManager.h"
4 #include "Animation.h"
5 #include "SDL.h"
6 #include <map>
7
8 /** klasa pozwalajaca wyswietlac postac wraz z animacja ruchu
9 class SpriteComponent : public Component {
10 private:
11     /** wskaznik do klasy
12     TransformComponent* transform;
13
14     /** odwołania do zmiennych dot. tekstur z biblioteki graficznej
15     SDL_Texture* texture;
16     SDL_Rect srcRect, destRect;
17
18     /** zmienna wskazujaca na stan animacji
19     bool animated = false;
20
21     /** zmienne wskazujace na ilosc klatek oraz ich predkosc wyswietlania
22     int frames = 1;
23     int speed = 100;
24
25 public:
26     /** zmienna indeksu
27     int animIndex = 0;
28
29     /** mapa zbierajaca stany animacji
30     std::map<const char*, Animation> animations;
31     SDL_RendererFlip spriteFlip = SDL_FLIP_NONE;
32
33     SpriteComponent();
34
35     /** konstruktor ustawiajacy tekstone
36     /** path - sciezka pliku
37     SpriteComponent(const char* path) {
38         setTexture(path);
39     }
40
41     /** konstruktor tworzacy stany animacji wraz z dodaniem do mapy
42     /** path - sciezka pliku
43     /** isAnimated - wskazanie stanu animacji
44     SpriteComponent(const char* path, bool isAnimated) {
45         animated = isAnimated;
46         Animation idle = Animation(0, 1, 100);
47         Animation walk = Animation(1, 8, 100);
48         Animation change = Animation(0, 2, 250);
49         animations.emplace("Idle", idle);
50         animations.emplace("Walk", walk);
51         animations.emplace("Change", change);
52
53         play("Idle");
54
55         setTexture(path);
56     }
57
58     /** dekonstruktor niszczacy tekstone
59     ~SpriteComponent() {}
60
61     /** funkcja ustawiajaca tekstone
62     /** path - sciezka pliku
63     void setTexture(const char* path) {
64         texture = TextureManager::LoadTexture(path);
65     }
66
67     /** funkcja inicjalizacyjna, odwołująca się do klasy virtualnej bazowej
68     void init() override {
69
70         transform = &entity->GetComponent<TransformComponent>();
71
72         srcRect.x = srcRect.y = 0;
73         srcRect.w = transform->width;
74         srcRect.h = transform->height;

```

```

75     }
76
77     /** funkcja odswierzajaca animacje, odwołująca się do klasy virtualnej bazowej
78     void update() override {
79
80         if (animated) {
81             srcRect.x = srcRect.w * static_cast<int>((SDL_GetTicks() / speed) % frames);
82         }
83         srcRect.y = animIndex * transform->height;
84
85         destRect.x = static_cast<int>(transform->position.x);
86         destRect.y = static_cast<int>(transform->position.y);
87         destRect.w = transform->width * transform->scale;
88         destRect.h = transform->height * transform->scale;
89
90     }
91
92     /** funkcja wyswietlajaca tekstone, odwołująca się do klasy virtualnej bazowej
93     void draw() override {
94         TextureManager::Draw(texture, srcRect, destRect, spriteFlip);
95     }
96
97     /** funkcja wprawiajaca w ruch animacje
98     /** animName - nazwa obiektu
99     void play(const char* animName) {
100         frames = animations[animName].frames;
101         animIndex = animations[animName].index;
102         speed = animations[animName].speed;
103     }
104 };

```

5.14 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Text.h

```

1 #pragma once
2 #include "SDL_ttf.h"
3 #include <map>
4 #include <string>
5 #include "ComponentSystem.h"
6
7 /** klasa inicjalizujaca tekt
8 class Text {
9 public:
10     Text(Manager* man);
11     ~Text();
12
13     /** funkcja tworząca tekst, pobierająca czcionkę
14     /** id - przypisany numer
15     /** path - ścieżka pliku
16     /** fontSize - wielkość czcionki
17     void AddFonts(std::string id, std::string path, int fontSize);
18     TTF_Font* GetFont(std::string id);
19
20 private:
21     Manager* manager;
22     /** mapa z czcionkami
23     std::map<std::string, TTF_Font*> fonts;
24 };

```

5.15 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TextureManager.h

```

1 #pragma once
2 #include "Game.h"
3
4 /** klasa zarządzająca generowaniem tekstur
5 class TextureManager {
6 public:
7
8     /** funkcje ładujące tekstury z zewnętrznej biblioteki graficznej
9     static SDL_Texture* LoadTexture(const char* fileName);
10     static SDL_Texture* toTexture(SDL_Surface* surface, int destroySurface);
11
12     /** funkcja rysująca tekturę
13     static void Draw(SDL_Texture* tex, SDL_Rect src, SDL_Rect dest, SDL_RendererFlip flip);
14
15 };

```

5.16 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/↵ Sokoban/Sokoban/TileComponent.h

```

1 #pragma once
2 #include "ComponentSystem.h"
3 #include "TransformComponent.h"
4 #include "SpriteComponent.h"
5 #include "SDL.h"
6
7 class TileComponent : public Component
8 {
9 public:
10     TransformComponent* transform;
11     SpriteComponent* sprite;
12     SDL_Rect tileRect;
13     int tileID;
14     const char* path;
15
16     TileComponent() = default;
17
18     TileComponent(int x, int y, int w, int h, int id) {
19         tileRect.x = x;
20         tileRect.y = y;
21         tileRect.w = w;
22         tileRect.h = h;
23         tileID = id;
24
25         switch (tileID)
26         {
27             case 0:
28                 path = "Assets/floor.png";
29                 break;
30             case 1:
31                 path = "Assets/wall.png";
32                 break;
33             case 2:
34                 break;
35             case 3:
36                 break;
37             default:
38                 break;
39         }
40     }
41
42     void init() override {
43         entity->addComponent<TransformComponent>((float)tileRect.x, (float)tileRect.y, tileRect.w,
44         tileRect.h, 1);
45         transform = &entity->getComponent<TransformComponent>();
46         entity->addComponent<SpriteComponent>(path);
47         sprite = &entity->getComponent<SpriteComponent>();
48     }
49 private:
50
51 };

```

5.17 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/↵ Sokoban/Sokoban/Timer.h

```

1 #pragma once
2 #include "SDL.h"
3
4
5
6
7
8
9
10 /** klasa tworząca zegar
11 class Timer
12 {
13 public:
14
15     /** inicjalizacja zegara
16     static Timer* Instance();
17
18     /** funkcja zerująca inicjalizację
19     static void Release();
20
21     /** funkcja zerująca odmierzenie

```

```

22     void Reset();
23
24     /** funkcja zerujaca minuty
25     void ResetMin();
26
27     /** funkcja zwracajaca tikniecie
28     int deltaTime();
29
30     /** funkcja pozaujaca "skale"
31     void TimeScale(float t);
32     int TimeScale();
33
34     /** funkcja odswierzajaca
35     void Update();
36
37     /** funkcja ustawiajaca godziny
38     unsigned int Hour();
39
40     /** funkcja ustawiajaca minuty
41     unsigned int Minute();
42
43     /** funkcja dodajaca godziny
44     void AddHour();
45
46     /** funkcja dodajaca minuty
47     void AddMinute();
48
49 private:
50
51     /** inicjalizacja zmiennych dot. czasu
52     static Timer* sInstance;
53
54     unsigned int mStartTicks;
55     unsigned int mLastTicks;
56     unsigned int mMinute = 0;
57     unsigned int mHour = 0;
58     int mDeltaTime;
59     float mTimeScale;
60
61     Timer();
62     ~Timer();
63 };

```

5.18 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/TransformComponent.h

```

1 #pragma once
2 #include "Components.h"
3 #include "ComponentSystem.h"
4 #include "Vector2D.h"
5
6 /** klasa pozwalajaca ustawic polozenie obiektu
7 class TransformComponent : public Component
8 {
9 public:
10
11     /** przywołanie zmiennych dot. pozycji z klasy Vector2D
12     Vector2D position;
13     Vector2D velocity;
14
15     /** inicjalizacja zmiennych dot. poruszania
16     int height = 100;
17     int width = 100;
18     int scale = 1;
19     std::string tag;
20     int speed = 3;
21
22     /** konstruktor z startowa pozycja
23     TransformComponent()
24     {
25         position.Zero();
26     }
27
28     /** konstruktor z nazwa obiektu
29     /** t - nazwa
30     TransformComponent(std::string t) {
31         tag = t;
32     }
33
34     /** konstruktor z pozycja i skala
35     /** sc - skala
36     TransformComponent(int sc)

```

```

37     {
38         position.Zero();
39         scale = sc;
40     }
41
42     /** konstruktor z polozeniem
43     /** x - polozenie w osi x
44     /** y - polozenie w osi y
45     TransformComponent(float x, float y)
46     {
47         position.Zero();
48     }
49
50     /** konstruktor z polozeniem, skala, oraz z wielkoscia
51     /** x - polozenie w osi x
52     /** y - polozenie w osi y
53     /** h - wysokosc
54     /** w - szerokosc
55     /** sc - skala
56     TransformComponent(int x, int y, int h, int w, int sc)
57     {
58         position.x = x;
59         position.y = y;
60         height = h;
61         width = w;
62         scale = sc;
63     }
64
65     /** funkcja inicjalizacyjna polozenie, odwołująca się do klasy virtualnej bazowej
66     void init() override {
67         velocity.Zero();
68     }
69
70     /** funkcja odświeżająca poruszanie, odwołująca się do klasy virtualnej bazowej
71     void update() override
72     {
73         position.x += velocity.x * speed;
74         position.y += velocity.y * speed;
75     }
76 };

```

5.19 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/UI.h ↩

```

1 #pragma once
2 #include "ComponentSystem.h"
3 #include "TextureManager.h"
4 #include "Game.h"
5 #include <SDL.h>
6 #include <SDL_ttf.h>
7 #include <string>
8
9
10
11
12
13
14 /** klasa ustawiająca polozenie tekstu w oknie
15 class UI : public Component
16 {
17 public:
18     UI() {};
19     /** konstruktor z polozeniem tekstu
20     /** xpos - polozenie na osi x
21     /** ypos - polozenie na osi y
22     /** text - tekst wyswietlany
23     /** font - czcionka
24     /** color - kolor czcionki
25     UI(int xpos, int ypos, std::string text, std::string font, SDL_Color& color)
26         : Text(text), Font(font), Color(color) {
27         pos.x = xpos;
28         pos.y = ypos;
29
30         SetText(Text, Font);
31     }
32     ~UI() {}
33
34     /** funkcja ustawiająca tekst
35     /** text - tekst
36     /** font - czcionka
37     void SetText(std::string text, std::string font) {
38         Free_All();

```

```

39     surface = TTF_RenderText_Blended(Game::textUI->GetFont(font), text.c_str(), Color);
40     Texture = SDL_CreateTextureFromSurface(Game::renderer, surface);
41     SDL_QueryTexture(Texture, nullptr, nullptr, &pos.w, &pos.h);
42 }
43
44 /** funkcja wyswietlajaca tekstone, odwołująca się do klasy virtualnej bazowej
45 void draw() override {
46     SDL_RenderCopy(Game::renderer, Texture, nullptr, &pos);
47 }
48
49 /** funkcja zwalnijająca pamięć
50 void Free_All()
51 {
52     if (surface == NULL) {
53         //cout << "Text surface already free'd" << endl;
54     }
55     else {
56         SDL_FreeSurface(surface);
57         surface = NULL;
58         //cout << "Free'd surface \n";
59     }
60
61     if (Texture == NULL) {
62         //cout << "Could not free memory for text \"" << text << "\". Error from SDL is: " <<
TTF_GetError() << endl;
63     }
64     else {
65         SDL_DestroyTexture(Texture);
66         Texture = NULL;
67     }
68 }
69
70 private:
71
72 /** inicjalizacja zmiennych dla tekstu
73 std::string Font;
74 std::string Text;
75 SDL_Color Color;
76 SDL_Texture* Texture = NULL;
77 SDL_Surface* surface = NULL;
78 SDL_Rect pos;
79
80 };

```

5.20 C:/Users/slawe/source/repos/93b730f0-gr23-repo/Projekt/Sokoban/Sokoban/Vector2D.h

```

1 #pragma once
2 #include <iostream>
3
4 class Vector2D
5 {
6 public:
7
8
9
10
11
12
13     /** zmienne okreslajace pozycje
14     float x;
15     float y;
16
17     Vector2D();
18     Vector2D(float x, float y);
19
20     /**/ funkcja zerujaca wektor
21     Vector2D& Zero();
22
23     /**/ funkcja dodajaca
24     Vector2D& Add(const Vector2D& vec);
25
26     /**/ funkcja odejmujaca
27     Vector2D& Subtract(const Vector2D& vec);
28
29     /**/ funkcja mnozaca
30     Vector2D& Multiply(const Vector2D& vec);
31
32     /**/ definicja przeciazania operatora +
33     friend Vector2D& operator+(Vector2D& v1, const Vector2D& v2);
34
35     /**/ definicja przeciazania operatora -

```

```
36     friend Vector2D& operator-(Vector2D& v1, const Vector2D& v2);
37
38     /**/ definicja przeciazenia operatora *
39     Vector2D& operator*(const int& i);
40
41     /**/ definicja przeciazenia operatora +=
42     Vector2D& operator+=(const Vector2D& vec);
43
44     /**/ definicja przeciazenia operatora -=
45     Vector2D& operator-=(const Vector2D& vec);
46
47     /**/ definicja przeciazenia operatora «
48     friend std::ostream& operator<<(std::ostream& stream, const Vector2D& vec);
49 };
```


Indeks

Animation, [7](#)

ColliderComponent, [7](#)

draw, [8](#)

init, [8](#)

update, [8](#)

Collision, [8](#)

Component, [9](#)

ComponentMove, [9](#)

draw

ColliderComponent, [8](#)

SpriteComponent, [15](#)

UI, [19](#)

Entity, [10](#)

Game, [10](#)

init

ColliderComponent, [8](#)

KeyboardControl, [12](#)

SpriteComponent, [15](#)

TileComponent, [17](#)

TransformComponent, [18](#)

KeyboardControl, [12](#)

init, [12](#)

update, [12](#)

Manager, [13](#)

Map, [13](#)

Menu, [13](#)

ScoreBoard, [14](#)

SpriteComponent, [15](#)

draw, [15](#)

init, [15](#)

update, [15](#)

Text, [16](#)

TextureManager, [16](#)

TileComponent, [16](#)

init, [17](#)

Timer, [17](#)

TransformComponent, [18](#)

init, [18](#)

update, [18](#)

UI, [19](#)

draw, [19](#)

update

ColliderComponent, [8](#)

KeyboardControl, [12](#)

SpriteComponent, [15](#)

TransformComponent, [18](#)

Vector2D, [19](#)