Carcassone Clone

Készítette Doxygen 1.9.4

Chapter 1

Adatszerkezet-mutató

1.1 Adatszerkezetek

Az összes adatszerkezet listája rövid leírásokkal:

Chapter 2

Fájlmutató

2.1 Fájllista

Az összes fájl listája rövid leírásokkal:

Chapter 3

Adatszerkezetek dokumentációja

3.1 Button struktúrareferencia

`#include <ui.h>`

Adatmezők

- SDL_Rect label_rect
- SDL_Rect local_rect
- SDL_Rect global_rect
- SDL_Color bg_color
- char * label
- SDL_Texture * label_texture
- TTF_Font * used_font

3.1.1 Részletes leírás

Definíció a(z) ui.h fájl 15. sorában.

3.1.2 Adatmezők dokumentációja

3.1.2.1 bg_color

`SDL_Color Button::bg_color`

Definíció a(z) ui.h fájl 17. sorában.

3.1.2.2 global_rect

```
SDL_Rect Button::global_rect
```

Definíció a(z) ui.h fájl 16. sorában.

3.1.2.3 label

```
char* Button::label
```

Definíció a(z) ui.h fájl 18. sorában.

3.1.2.4 label_rect

```
SDL_Rect Button::label_rect
```

Definíció a(z) ui.h fájl 16. sorában.

3.1.2.5 label_texture

```
SDL_Texture* Button::label_texture
```

Definíció a(z) ui.h fájl 19. sorában.

3.1.2.6 local_rect

```
SDL_Rect Button::local_rect
```

Definíció a(z) ui.h fájl 16. sorában.

3.1.2.7 used_font

```
TTF_Font* Button::used_font
```

Definíció a(z) ui.h fájl 20. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/ui.h

3.2 Carcassone struktúrareferencia

`#include <app.h>`

Adatmezők

- int width
- int height
- bool is_running
- SDL_Window * window
- SDL_Surface * window_icon
- SDL_Texture * splash_title
- SDL_Renderer * renderer
- TTF_Font * default_font
- TTF_Font * small_font
- SDL_mutex * smutex
- AppState state
- MenuScreen * menu_screen
- LeaderboardScreen * lboard_screen
- GameScreen * game_screen

3.2.1 Részletes leírás

Definíció a(z) app.h fájl 34. sorában.

3.2.2 Adatmezők dokumentációja

3.2.2.1 default_font

`TTF_Font* Carcassone::default_font`

Definíció a(z) app.h fájl 42. sorában.

3.2.2.2 game_screen

`GameScreen* Carcassone::game_screen`

Definíció a(z) app.h fájl 49. sorában.

3.2.2.3 height

```
int Carcassone::height
```

Definíció a(z) app.h fájl 36. sorában.

3.2.2.4 is_running

```
bool Carcassone::is_running
```

Definíció a(z) app.h fájl 37. sorában.

3.2.2.5 lboard_screen

```
LeaderboardScreen* Carcassone::lboard_screen
```

Definíció a(z) app.h fájl 48. sorában.

3.2.2.6 menu_screen

```
MenuScreen* Carcassone::menu_screen
```

Definíció a(z) app.h fájl 47. sorában.

3.2.2.7 renderer

```
SDL_Renderer* Carcassone::renderer
```

Definíció a(z) app.h fájl 41. sorában.

3.2.2.8 small_font

```
TTF_Font * Carcassone::small_font
```

Definíció a(z) app.h fájl 42. sorában.

3.2.2.9 smutex

```
SDL_mutex* Carcassone::smutex
```

Definíció a(z) app.h fájl 45. sorában.

3.2.2.10 splash_title

```
SDL_Texture* Carcassone::splash_title
```

Definíció a(z) app.h fájl 40. sorában.

3.2.2.11 state

```
AppState Carcassone::state
```

Definíció a(z) app.h fájl 46. sorában.

3.2.2.12 width

```
int Carcassone::width
```

Definíció a(z) app.h fájl 36. sorában.

3.2.2.13 window

```
SDL_Window* Carcassone::window
```

Definíció a(z) app.h fájl 38. sorában.

3.2.2.14 window_icon

```
SDL_Surface* Carcassone::window_icon
```

Definíció a(z) app.h fájl 39. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/app.h

3.3 CardPile struktúrareferencia

`#include <tile.h>`

Adatmezők

- TileType card
- struct CardPile * next

3.3.1 Részletes leírás

Definíció a(z) tile.h fájl 83. sorában.

3.3.2 Adatmezők dokumentációja

3.3.2.1 card

`TileType CardPile::card`

Definíció a(z) tile.h fájl 84. sorában.

3.3.2.2 next

`struct CardPile* CardPile::next`

Definíció a(z) tile.h fájl 85. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/tile.h

3.4 GameScreen struktúrareferencia

`#include <ui.h>`

Adatmezők

- bool is_ready
- bool is_game_over
- int held_arrow_keys [4]
- SDL_FPoint board_offset
- Tile ** board
- SDL_Texture * board_texture
- Tile * drawn_tile
- CardPile * card_pile
- size_t pile_index
- SDL_Texture * pile_counter [PILE_SIZE]
- SDL_Texture * pile_counter_
- bool update_counter
- Player players [2]
- Player * curr_player
- Player * winner
- SDL_Texture * crown_texture
- SDL_Texture * player_input_labels [2]
- Prompt player_name_inputs [2]
- Prompt * active_input
- Button ready_button
- Button end_turn_button
- Button concede_button
- TilesetWrapper tileset_wrapper

3.4.1 Részletes leírás

Definíció a(z) ui.h fájl 52. sorában.

3.4.2 Adatmezők dokumentációja

3.4.2.1 active_input

```
Prompt* GameScreen::active_input
```

Definíció a(z) ui.h fájl 90. sorában.

3.4.2.2 board

```
Tile** GameScreen::board
```

Definíció a(z) ui.h fájl 66. sorában.

3.4.2.3 board_offset

```
SDL_FPoint GameScreen::board_offset
```

Definíció a(z) ui.h fájl 63. sorában.

3.4.2.4 board_texture

```
SDL_Texture* GameScreen::board_texture
```

Definíció a(z) ui.h fájl 67. sorában.

3.4.2.5 card_pile

```
CardPile* GameScreen::card_pile
```

Definíció a(z) ui.h fájl 73. sorában.

3.4.2.6 concede_button

```
Button GameScreen::concede_button
```

Definíció a(z) ui.h fájl 93. sorában.

3.4.2.7 crown_texture

```
SDL_Texture* GameScreen::crown_texture
```

Definíció a(z) ui.h fájl 85. sorában.

3.4.2.8 curr_player

```
Player* GameScreen::curr_player
```

Definíció a(z) ui.h fájl 83. sorában.

3.4.2.9 drawn_tile

```
Tile* GameScreen::drawn_tile
```

Definíció a(z) ui.h fájl 70. sorában.

3.4.2.10 end_turn_button

```
Button GameScreen::end_turn_button
```

Definíció a(z) ui.h fájl 93. sorában.

3.4.2.11 held_arrow_keys

```
int GameScreen::held_arrow_keys[4]
```

Definíció a(z) ui.h fájl 60. sorában.

3.4.2.12 is_game_over

```
bool GameScreen::is_game_over
```

Definíció a(z) ui.h fájl 57. sorában.

3.4.2.13 is_ready

```
bool GameScreen::is_ready
```

Definíció a(z) ui.h fájl 54. sorában.

3.4.2.14 pile_counter

```
SDL_Texture* GameScreen::pile_counter[PILE_SIZE]
```

Definíció a(z) ui.h fájl 77. sorában.

3.4.2.15 pile_counter_

```
SDL_Texture* GameScreen::pile_counter_
```

Definíció a(z) ui.h fájl 78. sorában.

3.4.2.16 pile_index

```
size_t GameScreen::pile_index
```

Definíció a(z) ui.h fájl 74. sorában.

3.4.2.17 player_input_labels

```
SDL_Texture* GameScreen::player_input_labels[2]
```

Definíció a(z) ui.h fájl 88. sorában.

3.4.2.18 player_name_inputs

```
Prompt GameScreen::player_name_inputs[2]
```

Definíció a(z) ui.h fájl 89. sorában.

3.4.2.19 players

```
Player GameScreen::players[2]
```

Definíció a(z) ui.h fájl 82. sorában.

3.4.2.20 ready_button

```
Button GameScreen::ready_button
```

Definíció a(z) ui.h fájl 93. sorában.

3.4.2.21 tileset_wrapper

```
TilesetWrapper GameScreen::tileset_wrapper
```

Definíció a(z) ui.h fájl 95. sorában.

3.4.2.22 update_counter

```
bool GameScreen::update_counter
```

Definíció a(z) ui.h fájl 79. sorában.

3.4.2.23 winner

```
Player* GameScreen::winner
```

Definíció a(z) ui.h fájl 84. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/ui.h

3.5 Leaderboard struktúrareferencia

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

Adatmezők

- char const * records_file_path
- LeaderboardEntry * entries
- size_t entries_size

3.5.1 Részletes leírás

Definíció a(z) player.h fájl 55. sorában.

3.5.2 Adatmezők dokumentációja

3.5.2.1 entries

```
LeaderboardEntry* Leaderboard::entries
```

Definíció a(z) player.h fájl 60. sorában.

3.5.2.2 entries_size

```
size_t Leaderboard::entries_size
```

Definíció a(z) player.h fájl 61. sorában.

3.5.2.3 records_file_path

```
char const* Leaderboard::records_file_path
```

Definíció a(z) player.h fájl 57. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/player.h

3.6 LeaderboardEntry struktúrareferencia

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

Adatmezők

- char name [(MAX_PLAYER_NAME_LEN+1) ∗sizeof(wchar_t)]
- unsigned int highscore

3.6.1 Részletes leírás

Definíció a(z) player.h fájl 47. sorában.

3.6.2 Adatmezők dokumentációja

3.6.2.1 highscore

```
unsigned int LeaderboardEntry::highscore
```

Definíció a(z) player.h fájl 52. sorában.

3.6.2.2 name

```
char LeaderboardEntry::name[(MAX_PLAYER_NAME_LEN+1) *sizeof(wchar_t)]
```

Definíció a(z) player.h fájl 49. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/player.h

3.7 LeaderboardScreen struktúrareferencia

```
#include <ui.h>
```

Adatmezők

- Leaderboard * leaderboard
- char syntax_error_msg [128+1]
- SDL_Texture * list_texture
- Button back_button

3.7.1 Részletes leírás

Definíció a(z) ui.h fájl 40. sorában.

3.7.2 Adatmezők dokumentációja

3.7.2.1 back_button

```
Button LeaderboardScreen::back_button
```

Definíció a(z) ui.h fájl 48. sorában.

3.7.2.2 leaderboard

```
Leaderboard* LeaderboardScreen::leaderboard
```

Definíció a(z) ui.h fájl 41. sorában.

3.7.2.3 list_texture

```
SDL_Texture* LeaderboardScreen::list_texture
```

Definíció a(z) ui.h fájl 45. sorában.

3.7.2.4 syntax_error_msg

```
char LeaderboardScreen::syntax_error_msg[128+1]
```

Definíció a(z) ui.h fájl 42. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/ui.h

3.8 Meeple struktúrareferencia

```
#include <meeple.h>
```

Adatmezők

- bool is_placed
- int x
- int y
- SDL_Texture * texture

3.8.1 Részletes leírás

Definíció a(z) meeple.h fájl 7. sorában.

3.8.2 Adatmezők dokumentációja

3.8.2.1 is_placed

```
bool Meeple::is_placed
```

Definíció a(z) meeple.h fájl 9. sorában.

3.8.2.2 texture

```
SDL_Texture* Meeple::texture
```

Definíció a(z) meeple.h fájl 15. sorában.

3.8.2.3 x

```
int Meeple::x
```

Definíció a(z) meeple.h fájl 12. sorában.

3.8.2.4 y

```
int Meeple::y
```

Definíció a(z) meeple.h fájl 12. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/meeple.h

3.9 MenuScreen struktúrareferencia

```
#include <ui.h>
```

Adatmezők

- SDL_Texture * background
- SDL_Rect button_container
- Button start_button
- Button lboard_button

3.9.1 Részletes leírás

Definíció a(z) ui.h fájl 29. sorában.

3.9.2 Adatmezők dokumentációja

3.9.2.1 background

```
SDL_Texture* MenuScreen::background
```

Definíció a(z) ui.h fájl 31. sorában.

3.9.2.2 button_container

```
SDL_Rect MenuScreen::button_container
```

Definíció a(z) ui.h fájl 34. sorában.

3.9.2.3 lboard_button

```
Button MenuScreen::lboard_button
```

Definíció a(z) ui.h fájl 36. sorában.

3.9.2.4 start_button

```
Button MenuScreen::start_button
```

Definíció a(z) ui.h fájl 35. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/ui.h

3.10 Player struktúrareferencia

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

Adatmezők

- char name [(MAX_PLAYER_NAME_LEN+1) ∗sizeof(wchar_t)]
- unsigned int score
- bool has_placed_card
- Meeple meeples [MAX_MEEPLES]
- size_t meeples_at_hand
- SDL_Texture ∗ score_counter
- SDL_Texture ∗ handle_texture
- SDL_Texture ∗ stat_panel
- SDL_Texture ∗ own_meeple_texture
- bool update_score

3.10.1 Részletes leírás

Definíció a(z) player.h fájl 17. sorában.

3.10.2 Adatmezők dokumentációja

3.10.2.1 handle_texture

```
SDL_Texture* Player::handle_texture
```

Definíció a(z) player.h fájl 32. sorában.

3.10.2.2 has_placed_card

```
bool Player::has_placed_card
```

Definíció a(z) player.h fájl 24. sorában.

3.10.2.3 meeples

```
Meeple Player::meeples[MAX_MEEPLES]
```

Definíció a(z) player.h fájl 27. sorában.

3.10.2.4 meeples_at_hand

```
size_t Player::meeples_at_hand
```

Definíció a(z) player.h fájl 28. sorában.

3.10.2.5 name

```
char Player::name[(MAX_PLAYER_NAME_LEN+1) *sizeof(wchar_t)]
```

Definíció a(z) player.h fájl 20. sorában.

3.10.2.6 own_meeple_texture

```
SDL_Texture* Player::own_meeple_texture
```

Definíció a(z) player.h fájl 34. sorában.

3.10.2.7 score

```
unsigned int Player::score
```

Definíció a(z) player.h fájl 21. sorában.

3.10.2.8 score_counter

```
SDL_Texture* Player::score_counter
```

Definíció a(z) player.h fájl 31. sorában.

3.10.2.9 stat_panel

```
SDL_Texture* Player::stat_panel
```

Definíció a(z) player.h fájl 33. sorában.

3.10.2.10 update_score

```
bool Player::update_score
```

Definíció a(z) player.h fájl 37. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/player.h

3.11 Prompt struktúrareferencia

```
#include <ui.h>
```

Adatmezők

- Button prompt

3.11.1 Részletes leírás

Definíció a(z) ui.h fájl 24. sorában.

3.11.2 Adatmezők dokumentációja

3.11.2.1 prompt

```
Button Prompt::prompt
```

Definíció a(z) ui.h fájl 25. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/ui.h

3.12 Tile struktúrareferencia

```
#include <tile.h>
```

Adatmezők

- TileType type
- ConnectionType connections [4]
- SDL_FPoint local_coords
- SDL_FPoint global_coords
- SDL_Point board_coords
- unsigned short rotation
- bool rotatable
- bool is_scored
- bool is_expired

3.12.1 Részletes leírás

Definíció a(z) tile.h fájl 54. sorában.

3.12.2 Adatmezők dokumentációja

3.12.2.1 board_coords

```
SDL_Point Tile::board_coords
```

Definíció a(z) tile.h fájl 64. sorában.

3.12.2.2 connections

```
ConnectionType Tile::connections[4]
```

Definíció a(z) tile.h fájl 58. sorában.

3.12.2.3 global_coords

```
SDL_FPoint Tile::global_coords
```

Definíció a(z) tile.h fájl 61. sorában.

3.12.2.4 is_expired

```
bool Tile::is_expired
```

Definíció a(z) tile.h fájl 73. sorában.

3.12.2.5 is_scored

```
bool Tile::is_scored
```

Definíció a(z) tile.h fájl 72. sorában.

3.12.2.6 local_coords

```
SDL_FPoint Tile::local_coords
```

Definíció a(z) tile.h fájl 61. sorában.

3.12.2.7 rotatable

```
bool Tile::rotatable
```

Definíció a(z) tile.h fájl 70. sorában.

3.12.2.8 rotation

```
unsigned short Tile::rotation
```

Definíció a(z) tile.h fájl 67. sorában.

3.12.2.9 type

```
TileType Tile::type
```

Definíció a(z) tile.h fájl 55. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/tile.h

3.13 TilesetWrapper struktúrareferencia

`#include <tile.h>`

Adatmezők

- SDL_Texture * tile_set

3.13.1 Részletes leírás

Definíció a(z) tile.h fájl 92. sorában.

3.13.2 Adatmezők dokumentációja

3.13.2.1 tile_set

```
SDL_Texture* TilesetWrapper::tile_set
```

Definíció a(z) tile.h fájl 93. sorában.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- include/game/tile.h

Chapter 4

Fájlok dokumentációja

4.1 include/app.h fájlreferencia

```
#include <bits/pthreadtypes.h>
#include <stdbool.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "game/tile.h"
#include "game/player.h"
#include "ui.h"
```

Adatszerkezetek

- struct Carcassone

Makródefiníciók

- #define FPS 60
- #define MIN(x, y) ((x < y) ? x : y)
- #define COLOR_BLUE 25, 30, 91, 255
- #define COLOR_LIGHTBLUE 153, 204, 255, 255
- #define COLOR_WHITE 255, 255, 255, 255
- #define COLOR_RED 255, 0, 0, 255
- #define COLOR_SALMON 255, 145, 164, 255
- #define COLOR_BG 102, 102, 153, 255
- #define DBG_LOG(x, ...) SDL_LogDebug(SDL_LOG_CATEGORY_CUSTOM, x, ##__VA_ARGS__)

Enumerációk

- enum AppState { MENU , GAME , LEADERBOARD }

Függvények

- Carcassone * Carcassone__construct (int, int, char const *)

  Initicializálja az összes nézetet, SDL és TTF kontextusokat.
- void Carcassone__destroy (Carcassone *)

  Felszabadítja a megadott Carcassone struktúra által lefoglalt memóriát.
- void Carcassone__switch_state (Carcassone *, AppState)
- void Carcassone__run (Carcassone *)

  A fő programciklus.
- void Carcassone__update (Carcassone *, float)
- void Carcassone__handle_input (Carcassone *)

  Inputok kezelése.
- void Carcassone__render_splash_title (Carcassone *)

  A splash cím renderelése.
- void Carcassone__indicate_possible_placements (Carcassone *)
- bool Carcassone__check_names_valid (Carcassone *)
- bool Carcassone__check_surrounding_tiles (Carcassone *, SDL_Point)

  Helyes pozíció ellenőrzése.
- void Carcassone__check_scorable_constructs (Carcassone *)
- bool Carcassone__check_if_possible (Carcassone *)
- void Carcassone__calculate_scores_for_cloister (Carcassone *, Tile *)
- void Carcassone__calculate_scores_for_road (Carcassone *, Tile **, size_t, unsigned int)
- void Carcassone__calculate_scores_for_castle (Carcassone *, Tile **, size_t, unsigned int)
- void Carcassone__Menu__construct (Carcassone *)

  Létrehozza a menünézetet.
- void Carcassone__Menu__destroy (Carcassone *)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `MenuScreen` által lefoglalt memóriát.
- void Carcassone__Menu__render (Carcassone *)

  Menünézet megjelenítése.
- void Carcassone__Menu__handle_input (SDL_Event)
- void Carcassone__Lboard__construct (Carcassone *)

  Létrehozza a dicsőséglistanézetet.
- void Carcassone__Lboard__destroy (Carcassone *)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `LeaderboardScreen` által lefoglalt memóriát.
- void Carcassone__Lboard__render (Carcassone *)

  Dicsőséglistanézet megjelenítése.
- void Carcassone__Lboard__init_list_texture (Carcassone *)

  Létrehozza a dicsőséglistanézethez a rekordokat.
- void Carcassone__Lboard__handle_input (SDL_Event)
- void Carcassone__Game__construct (Carcassone *)

  Létrehozza a játéknézetet.
- void Carcassone__Game__destroy (Carcassone *)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `GameScreen` által lefoglalt memóriát.
- void Carcassone__Game__handle_input (SDL_Event)
- void Carcassone__Game__init_players (Carcassone *)

  Játékosok létrehozása.
- void Carcassone__Game__init_pile (Carcassone *)

  Kártyapakli létrehozása.
- void Carcassone__Game__init_board (Carcassone *)

  Játéktábla létrehozása.
- void Carcassone__Game__init_counter (Carcassone *)
- void Carcassone__Game__render (Carcassone *)

Játéknézet megjelenítése.

- void Carcassone__Game__render_board (Carcassone ∗)

  Játéktábla renderelése.
- void Carcassone__Game__render_drawn_tile (Carcassone ∗)

  A pakli tetején levő kártya renderelése.
- void Carcassone__Game__render_player_stats (Carcassone ∗)

  A splash cím renderelése.
- void Carcassone__Game__render_meeples (Carcassone ∗)
- void Carcassone__Game__render_game_over (Carcassone ∗)
- void Carcassone__Game__show_finish_screen (Carcassone ∗)
- void Carcassone__Game__move_board (Carcassone ∗, SDL_Scancode)

  Játéktábla mozgatása.
- void Carcassone__Game__draw_new (Carcassone ∗)

  Új kártya húzása a pakli tetejéről.
- Button Carcassone__Button__construct (Carcassone ∗, TTF_Font ∗, char ∗, SDL_Rect, SDL_Color, SDL↩ _Color, bool)
- void Carcassone__Button__destroy (Carcassone ∗, Button ∗)
- void Carcassone__Button__render (Carcassone ∗, Button ∗)
- bool Carcassone__Button__hover (Carcassone ∗, Button ∗, SDL_Point)
- Prompt Carcassone__Prompt__construct (Carcassone ∗, TTF_Font ∗, char ∗, SDL_Rect, SDL_Color, SDL↩ _Color)
- void Carcassone__Prompt__destroy (Carcassone ∗, Prompt ∗)
- void Carcassone__Prompt__render (Carcassone ∗, Prompt ∗)
- void Carcassone__Prompt__edit (Carcassone ∗, Prompt ∗, char ∗, bool)
- size_t get_utf8_length (char ∗)
- void destroy_texture (SDL_Texture ∗)

4.1.1 Makródefiníciók dokumentációja

4.1.1.1 COLOR_BG

```
#define COLOR_BG 102, 102, 153, 255
```

Definíció a(z) app.h fájl 23. sorában.

4.1.1.2 COLOR_BLUE

```
#define COLOR_BLUE 25, 30, 91, 255
```

Definíció a(z) app.h fájl 18. sorában.

4.1.1.3 COLOR_LIGHTBLUE

```
#define COLOR_LIGHTBLUE 153, 204, 255, 255
```

Definíció a(z) app.h fájl 19. sorában.

4.1.1.4 COLOR_RED

```
#define COLOR_RED 255, 0, 0, 255
```

Definíció a(z) app.h fájl 21. sorában.

4.1.1.5 COLOR_SALMON

```
#define COLOR_SALMON 255, 145, 164, 255
```

Definíció a(z) app.h fájl 22. sorában.

4.1.1.6 COLOR_WHITE

```
#define COLOR_WHITE 255, 255, 255, 255
```

Definíció a(z) app.h fájl 20. sorában.

4.1.1.7 DBG_LOG

```
#define DBG_LOG(
              x,
              ...  ) SDL_LogDebug(SDL_LOG_CATEGORY_CUSTOM, x, ##__VA_ARGS__)
```

Development alatt használt logger.

Definíció a(z) app.h fájl 25. sorában.

4.1.1.8 FPS

```
#define FPS 60
```

Cél FPS.

Definíció a(z) app.h fájl 14. sorában.

4.1.1.9 MIN

```
#define MIN(
              x,
              y ) ((x < y) ?  x :  y)
```

Egyszerű minimum meghatározó.

Definíció a(z) app.h fájl 16. sorában.

4.1.2 Enumerációk dokumentációja

4.1.2.1 AppState

```
enum AppState
```

A program állapotai: menü, játék, dicsőséglista.

Enumeráció-értékek

MENU	
GAME	
LEADERBOARD	

Definíció a(z) app.h fájl 28. sorában.

4.1.3 Függvények dokumentációja

4.1.3.1 Carcassone__Button__construct()

```
Button Carcassone__Button__construct (
             Carcassone * this,
             TTF_Font * font,
             char * label,
             SDL_Rect global_rect,
             SDL_Color bg_color,
             SDL_Color text_color,
             bool center )
```

Definíció a(z) ui.c fájl 16. sorában.

4.1.3.2 Carcassone__Button__destroy()

```
void Carcassone__Button__destroy (
            Carcassone * this,
            Button * button )
```

Definíció a(z) ui.c fájl 51. sorában.

4.1.3.3 Carcassone__Button__hover()

```
bool Carcassone__Button__hover (
            Carcassone * ,
            Button * ,
            SDL_Point  )
```

4.1.3.4 Carcassone__Button__render()

```
void Carcassone__Button__render (
            Carcassone * this,
            Button * button )
```

Definíció a(z) ui.c fájl 40. sorában.

4.1.3.5 Carcassone__calculate_scores_for_castle()

```
void Carcassone__calculate_scores_for_castle (
            Carcassone * this,
            Tile ** castles,
            size_t num,
            unsigned int point )
```

Definíció a(z) app.c fájl 828. sorában.

4.1.3.6 Carcassone__calculate_scores_for_cloister()

```
void Carcassone__calculate_scores_for_cloister (
            Carcassone * this,
            Tile * cloister )
```

Definíció a(z) app.c fájl 792. sorában.

4.1.3.7 Carcassone__calculate_scores_for_road()

```
void Carcassone__calculate_scores_for_road (
            Carcassone * this,
            Tile ** roads,
            size_t num,
            unsigned int point )
```

Definíció a(z) app.c fájl 809. sorában.

4.1.3.8 Carcassone__check_if_possible()

```
bool Carcassone__check_if_possible (
            Carcassone * this )
```

Definíció a(z) app.c fájl 589. sorában.

4.1.3.9 Carcassone__check_names_valid()

```
bool Carcassone__check_names_valid (
            Carcassone * this )
```

Definíció a(z) app.c fájl 324. sorában.

4.1.3.10 Carcassone__check_scorable_constructs()

```
void Carcassone__check_scorable_constructs (
            Carcassone * this )
```

Definíció a(z) app.c fájl 613. sorában.

4.1.3.11 Carcassone__check_surrounding_tiles()

```
bool Carcassone__check_surrounding_tiles (
            Carcassone * this,
            SDL_Point tcoords )
```

Helyes pozíció ellenőrzése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.
tcoords	A kapott kártya potenciális helye a táblán.

Visszatérési érték

Letehető-e a megfelelő pozícióba az adott kártya.

Definíció a(z) app.c fájl 915. sorában.

4.1.3.12 Carcassone__construct()

```
Carcassone * Carcassone__construct (
             int width,
             int height,
             char const * title )
```

Initicializálja az összes nézetet, SDL és TTF kontextusokat.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a Carcassone__destroy(Carcassone∗) függvényt.

Paraméterek

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

Visszatérési érték

Pointer az újonnan létrehozott Carcassone structra.

Definíció a(z) app.c fájl 32. sorában.

4.1.3.13 Carcassone__destroy()

```
void Carcassone__destroy (
             Carcassone * this )
```

Felszabadítja a megadott Carcassone struktúra által lefoglalt memóriát.

Paraméterek

this	A Carcassone struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) app.c fájl 108. sorában.

4.1.3.14 Carcassone__Game__construct()

```
void Carcassone__Game__construct (
            Carcassone * this )
```

Létrehozza a játéknézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Game__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a játéknézetét.

Definíció a(z) screens.c fájl 217. sorában.

4.1.3.15 Carcassone__Game__destroy()

```
void Carcassone__Game__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `GameScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 310. sorában.

4.1.3.16 Carcassone__Game__draw_new()

```
void Carcassone__Game__draw_new (
            Carcassone * this )
```

Új kártya húzása a pakli tetejéről.

Húz egy új kártyát, ha kifogyott akkor kilép a menübe (egyelőre).

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 963. sorában.

4.1.3.17 Carcassone__Game__handle_input()

```
void Carcassone__Game__handle_input (
            SDL_Event  )
```

4.1.3.18 Carcassone__Game__init_board()

```
void Carcassone__Game__init_board (
            Carcassone * this )
```

Játéktábla létrehozása.

Létrehozza a játéktáblát és le is helyezi a kezdőkártyát.

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 419. sorában.

4.1.3.19 Carcassone__Game__init_counter()

```
void Carcassone__Game__init_counter (
            Carcassone *  )
```

4.1.3.20 Carcassone__Game__init_pile()

```
void Carcassone__Game__init_pile (
            Carcassone * this )
```

Kártyapakli létrehozása.

Véletlenszerűen megkeveri a paklit és létrehozza a mezőkártyákat mindegyikhez.

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 359. sorában.

4.1.3.21 Carcassone__Game__init_players()

```
void Carcassone__Game__init_players (
            Carcassone * this )
```

Játékosok létrehozása.

Inicializálja a két játékost előre megadott adatok alapján (ezek a GameScreenben találhatók).

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 342. sorában.

4.1.3.22 Carcassone__Game__move_board()

```
void Carcassone__Game__move_board (
            Carcassone * this,
            SDL_Scancode key )
```

Játéktábla mozgatása.

A nyilak segítségével a játéktábla látható részét mozgatja.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.
key	A beolvasott billentyű (a Carcassone__handle_inputs(Carcassone∗)-tól kapja meg).

Definíció a(z) app.c fájl 882. sorában.

4.1.3.23 Carcassone__Game__render()

```
void Carcassone__Game__render (
            Carcassone * this )
```

Játéknézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a nézet tartozik.

Definíció a(z) screens.c fájl 350. sorában.

4.1.3.24 Carcassone__Game__render_board()

```
void Carcassone__Game__render_board (
            Carcassone * this )
```

Játéktábla renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 450. sorában.

4.1.3.25 Carcassone__Game__render_drawn_tile()

```
void Carcassone__Game__render_drawn_tile (
            Carcassone * this )
```

A pakli tetején levő kártya renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 496. sorában.

4.1.3.26 Carcassone__Game__render_game_over()

```
void Carcassone__Game__render_game_over (
            Carcassone * this )
```

Definíció a(z) app.c fájl 553. sorában.

4.1.3.27 Carcassone__Game__render_meeples()

```
void Carcassone__Game__render_meeples (
            Carcassone * this )
```

Definíció a(z) app.c fájl 525. sorában.

4.1.3.28 Carcassone__Game__render_player_stats()

```
void Carcassone__Game__render_player_stats (
            Carcassone * this )
```

A splash cím renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 851. sorában.

4.1.3.29 Carcassone__Game__show_finish_screen()

```
void Carcassone__Game__show_finish_screen (
            Carcassone * this )
```

Definíció a(z) app.c fájl 1003. sorában.

4.1.3.30 Carcassone__handle_input()

```
void Carcassone__handle_input (
            Carcassone * this )
```

Inputok kezelése.

SDL_Event-öket kezel.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 199. sorában.

4.1.3.31 Carcassone__indicate_possible_placements()

```
void Carcassone__indicate_possible_placements (
            Carcassone * this )
```

Definíció a(z) app.c fájl 569. sorában.

4.1.3.32 Carcassone__Lboard__construct()

```
void Carcassone__Lboard__construct (
            Carcassone * this )
```

Létrehozza a dicsőséglistanézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Lboard__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a dicsőséglistanézetét.

Definíció a(z) screens.c fájl 96. sorában.

4.1.3.33 Carcassone__Lboard__destroy()

```
void Carcassone__Lboard__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `LeaderboardScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 116. sorában.

4.1.3.34 Carcassone__Lboard__handle_input()

```
void Carcassone__Lboard__handle_input (
            SDL_Event  )
```

4.1.3.35 Carcassone__Lboard__init_list_texture()

```
void Carcassone__Lboard__init_list_texture (
            Carcassone * this )
```

Létrehozza a dicsőséglistanézethez a rekordokat.

Akkor kell meghívni, ha frissül a rekordfájl.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a dicsőséglistanézetét.

Definíció a(z) screens.c fájl 131. sorában.

4.1.3.36 Carcassone__Lboard__render()

```
void Carcassone__Lboard__render (
            Carcassone * this )
```

Dicsőséglistanézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a dicsőséglista tartozik.

Definíció a(z) screens.c fájl 196. sorában.

4.1.3.37 Carcassone__Menu__construct()

```
void Carcassone__Menu__construct (
            Carcassone * this )
```

Létrehozza a menünézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Menu__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a menünézetét.

Definíció a(z) screens.c fájl 26. sorában.

4.1.3.38 Carcassone__Menu__destroy()

```
void Carcassone__Menu__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `MenuScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 59. sorában.

4.1.3.39 Carcassone__Menu__handle_input()

```
void Carcassone__Menu__handle_input (
            SDL_Event  )
```

4.1.3.40 Carcassone__Menu__render()

```
void Carcassone__Menu__render (
            Carcassone * this )
```

Menünézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a menü tartozik.

Definíció a(z) screens.c fájl 72. sorában.

4.1.3.41 Carcassone__Prompt__construct()

```
Prompt Carcassone__Prompt__construct (
            Carcassone * this,
            TTF_Font * font,
            char * default_label,
            SDL_Rect global_rect,
            SDL_Color bg_color,
            SDL_Color text_color )
```

Definíció a(z) ui.c fájl 57. sorában.

4.1.3.42 Carcassone__Prompt__destroy()

```
void Carcassone__Prompt__destroy (
            Carcassone * this,
            Prompt * prompt )
```

Definíció a(z) ui.c fájl 96. sorában.

4.1.3.43 Carcassone__Prompt__edit()

```
void Carcassone__Prompt__edit (
            Carcassone * this,
            Prompt * prompt,
            char * new_label,
            bool concat )
```

Definíció a(z) ui.c fájl 66. sorában.

4.1.3.44 Carcassone__Prompt__render()

```
void Carcassone__Prompt__render (
            Carcassone * this,
            Prompt * prompt )
```

Definíció a(z) ui.c fájl 91. sorában.

4.1.3.45 Carcassone__render_splash_title()

```
void Carcassone__render_splash_title (
            Carcassone * this )
```

A splash cím renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 485. sorában.

4.1.3.46 Carcassone__run()

```
void Carcassone__run (
            Carcassone * this )
```

A fő programciklus.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 1028. sorában.

4.1.3.47 Carcassone__switch_state()

```
void Carcassone__switch_state (
            Carcassone * this,
            AppState new_state )
```

Definíció a(z) app.c fájl 130. sorában.

4.1.3.48 Carcassone__update()

```
void Carcassone__update (
            Carcassone * ,
            float  )
```

4.1.3.49 destroy_texture()

```
void destroy_texture (
            SDL_Texture *  )
```

4.1.3.50 get_utf8_length()

```
size_t get_utf8_length (
            char * str )
```

Definíció a(z) app.c fájl 176. sorában.

4.2 app.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_APP_H
00002 #define CRCLONE_APP_H
00003
00004 #include <bits/pthreadtypes.h>
00005 #include <stdbool.h>
00006 #include <SDL2/SDL.h>
00007 #include <SDL2/SDL_ttf.h>
00008
00009 #include "game/tile.h"
00010 #include "game/player.h"
00011 #include "ui.h"
00012
00014 #define FPS 60
00016 #define MIN(x, y) ((x < y) ?  x :  y)
00017 // Színek.
00018 #define COLOR_BLUE 25, 30, 91, 255
00019 #define COLOR_LIGHTBLUE 153, 204, 255, 255
00020 #define COLOR_WHITE 255, 255, 255, 255
```

```
00021 #define COLOR_RED 255, 0, 0, 255
00022 #define COLOR_SALMON 255, 145, 164, 255
00023 #define COLOR_BG 102, 102, 153, 255
00025 #define DBG_LOG(x, ...)  SDL_LogDebug(SDL_LOG_CATEGORY_CUSTOM, x, ##__VA_ARGS__)
00026
00028 typedef enum {
00029     MENU,
00030     GAME,
00031     LEADERBOARD
00032 } AppState;
00033
00034 typedef struct {
00035     // Általános adatok
00036     int width, height;
00037     bool is_running;
00038     SDL_Window* window;
00039     SDL_Surface* window_icon;
00040     SDL_Texture* splash_title;
00041     SDL_Renderer* renderer;
00042     TTF_Font* default_font, * small_font;
00043
00044     // Állapotok
00045     SDL_mutex* smutex;
00046     AppState state;
00047     MenuScreen* menu_screen;
00048     LeaderboardScreen* lboard_screen;
00049     GameScreen* game_screen;
00050 } Carcassone;
00051 Carcassone* Carcassone__construct(int, int, char const*);
00052 void Carcassone__destroy(Carcassone*);
00053 void Carcassone__switch_state(Carcassone*, AppState);
00054 void Carcassone__run(Carcassone*);
00055 void Carcassone__update(Carcassone*, float);
00056 void Carcassone__handle_input(Carcassone*);
00057 void Carcassone__render_splash_title(Carcassone*);
00058 void Carcassone__indicate_possible_placements(Carcassone*);
00059 bool Carcassone__check_names_valid(Carcassone*);
00060 bool Carcassone__check_surrounding_tiles(Carcassone*, SDL_Point);
00061 void Carcassone__check_scorable_constructs(Carcassone*);
00062 bool Carcassone__check_if_possible(Carcassone*);
00063 void Carcassone__calculate_scores_for_cloister(Carcassone*, Tile*);
00064 void Carcassone__calculate_scores_for_road(Carcassone*, Tile**, size_t, unsigned int);
00065 void Carcassone__calculate_scores_for_castle(Carcassone*, Tile**, size_t, unsigned int);
00066
00067 void Carcassone__Menu__construct(Carcassone*);
00068 void Carcassone__Menu__destroy(Carcassone*);
00069 void Carcassone__Menu__render(Carcassone*);
00070 void Carcassone__Menu__handle_input(SDL_Event);
00071
00072 void Carcassone__Lboard__construct(Carcassone*);
00073 void Carcassone__Lboard__destroy(Carcassone*);
00074 void Carcassone__Lboard__render(Carcassone*);
00075 void Carcassone__Lboard__init_list_texture(Carcassone*);
00076 void Carcassone__Lboard__handle_input(SDL_Event);
00077
00078 void Carcassone__Game__construct(Carcassone*);
00079 void Carcassone__Game__destroy(Carcassone*);
00080 void Carcassone__Game__handle_input(SDL_Event);
00081 void Carcassone__Game__init_players(Carcassone*);
00082 void Carcassone__Game__init_pile(Carcassone*);
00083 void Carcassone__Game__init_board(Carcassone*);
00084 void Carcassone__Game__init_counter(Carcassone*);
00085 void Carcassone__Game__render(Carcassone*);
00086 void Carcassone__Game__render_board(Carcassone*);
00087 void Carcassone__Game__render_drawn_tile(Carcassone*);
00088 void Carcassone__Game__render_player_stats(Carcassone*);
00089 void Carcassone__Game__render_meeples(Carcassone*);
00090 void Carcassone__Game__render_game_over(Carcassone*);
00091 void Carcassone__Game__show_finish_screen(Carcassone*);
00092 void Carcassone__Game__move_board(Carcassone*, SDL_Scancode);
00093 void Carcassone__Game__draw_new(Carcassone*);
00094
00095 Button Carcassone__Button__construct(Carcassone*, TTF_Font*, char*, SDL_Rect, SDL_Color, SDL_Color,
      bool);
00096 void Carcassone__Button__destroy(Carcassone*, Button*);
00097 void Carcassone__Button__render(Carcassone*, Button*);
00098 bool Carcassone__Button__hover(Carcassone*, Button*, SDL_Point);
00099
00100 Prompt Carcassone__Prompt__construct(Carcassone*, TTF_Font*, char*, SDL_Rect, SDL_Color, SDL_Color);
00101 void Carcassone__Prompt__destroy(Carcassone*, Prompt*);
00102 void Carcassone__Prompt__render(Carcassone*, Prompt*);
00103 void Carcassone__Prompt__edit(Carcassone*, Prompt*, char*, bool);
00104
00105 size_t get_utf8_length(char*);
00106 void destroy_texture(SDL_Texture*);
00107 //bool render_text(char const*, SDL_Color, );
00108
```

```
00109 #endif
```

4.3 include/game/meeple.h fájlreferencia

```
#include <stdbool.h>
#include <SDL2/SDL.h>
```

Adatszerkezetek

- struct Meeple

Függvények

- Meeple Meeple__construct (SDL_Texture *)

  Létrehoz egy alattvalót.
- void Meeple__render (SDL_Renderer *, unsigned int, SDL_Color)
- void Meeple__destroy (Meeple *)

4.3.1 Függvények dokumentációja

4.3.1.1 Meeple__construct()

```
Meeple Meeple__construct (
            SDL_Texture * texture )
```

Létrehoz egy alattvalót.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Meeple__destroy` függvényt.

Paraméterek

texture	A kinézete.

Visszatérési érték

Az új `Meeple` struktúra.

Definíció a(z) players.c fájl 23. sorában.

4.3.1.2 Meeple__destroy()

```
void Meeple__destroy (
            Meeple *  )
```

4.3.1.3 Meeple__render()

```
void Meeple__render (
            SDL_Renderer * ,
            unsigned int ,
            SDL_Color  )
```

4.4 meeple.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_MEEPLE_H
00002 #define CRCLONE_MEEPLE_H
00003
00004 #include <stdbool.h>
00005 #include <SDL2/SDL.h>
00006
00007 typedef struct {
00008     // Le van e helyezve
00009     bool is_placed;
00010
00011     // Ha !is_placed, akkor akár invalid, "szemét" értéket is tárolhat.
00012     int x, y;
00013
00014     // Játékosoként változtatható kinézet
00015     SDL_Texture* texture;
00016 } Meeple;
00017 Meeple Meeple__construct(SDL_Texture*);
00018 void Meeple__render(SDL_Renderer*, unsigned int, SDL_Color);
00019 void Meeple__destroy(Meeple*);
00020
00021 #endif
```

4.5 include/game/player.h fájlreferencia

```
#include <stdlib.h>
#include <stdbool.h>
#include <wchar.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "game/meeple.h"
```

Adatszerkezetek

- struct Player
- struct LeaderboardEntry
- struct Leaderboard

Makródefiníciók

- #define MAX_MEEPLES 7
- #define MAX_PLAYER_NAME_LEN 24

Függvények

- Player Player__construct (SDL_Renderer ∗, TTF_Font ∗, char ∗, char const ∗)
- void Player__place_meeple (Player ∗, SDL_Point)

  Alattvaló lehelyezése.
- void Player__reclaim_meeple (Player ∗, Meeple ∗)

  Lehelyezett alattvaló visszaszerzése.
- void Player__render (Player ∗, SDL_Renderer ∗, TTF_Font ∗)
- void Player__add_to_score (Player ∗, unsigned int)

  Játékos pontszámának növelése.
- void Player__destroy (Player ∗)

  Felszabadítja a megadott `Player` struktúra által lefoglalt memóriát.
- Leaderboard ∗ Leaderboard__construct (char const ∗)

  Létrehozza a dicsőséglistát.
- void Leaderboard__destroy (Leaderboard ∗)

  Felszabadítja a megadott `Leaderboard` struktúra által lefoglalt memóriát.
- void Leaderboard__sort (Leaderboard ∗)

  A beolvasott rekordok csökkenő sorba rendezése.
- bool Leaderboard__load (Leaderboard ∗)

  Betölti a rekordfájl tartalmát.
- bool Leaderboard__insert_new (Leaderboard ∗, Player ∗)

  Új rekord beszúrása és mentése.

4.5.1 Makródefiníciók dokumentációja

4.5.1.1 MAX_MEEPLES

```
#define MAX_MEEPLES 7
```

Definíció a(z) player.h fájl 13. sorában.

4.5.1.2 MAX_PLAYER_NAME_LEN

```
#define MAX_PLAYER_NAME_LEN 24
```

Definíció a(z) player.h fájl 15. sorában.

4.5.2 Függvények dokumentációja

4.5.2.1 Leaderboard__construct()

```
Leaderboard * Leaderboard__construct (
            char const * records_file_path )
```

Létrehozza a dicsőséglistát.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Leaderboard__destroy` függvényt.

Paraméterek

records_file_path	A fájl elérési útvonala, ahonnan beolvassa a játékosneveket és rekordokat.

Visszatérési érték

Pointer az újonnan létrehozott `Leaderboard`-ra.

Definíció a(z) leaderboard.c fájl 22. sorában.

4.5.2.2 Leaderboard__destroy()

```
void Leaderboard__destroy (
            Leaderboard * this )
```

Felszabadítja a megadott `Leaderboard` struktúra által lefoglalt memóriát.

Paraméterek

this	A `Leaderboard` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) leaderboard.c fájl 41. sorában.

4.5.2.3 Leaderboard__insert_new()

```
bool Leaderboard__insert_new (
            Leaderboard * this,
            Player * new )
```

Új rekord beszúrása és mentése.

Megjegyzés: A visszatérési érték nem feltétlen van kihasználva, elhanyagolható.

Paraméterek

this	A `Leaderboard` struktúra, amihez tartozó rekordfájlba történjen a mentés.
new	A potenciálisan új, rekordot elérő játékos.

Visszatérési érték

Már tartalmazta e a dicsőséglista a játékost.

Definíció a(z) leaderboard.c fájl 129. sorában.

4.5.2.4 Leaderboard__load()

```
bool Leaderboard__load (
            Leaderboard * this )
```

Betölti a rekordfájl tartalmát.

Érdemes meghívni a `Leaderboard__sort`-ot ezután.

Paraméterek

this	A `Leaderboard` struktúra, amelyhez már tartozik egy rekordfájl.

Visszatérési érték

Hibamentesen lefutott e.

Definíció a(z) leaderboard.c fájl 56. sorában.

4.5.2.5 Leaderboard__sort()

```
void Leaderboard__sort (
            Leaderboard * this )
```

A beolvasott rekordok csökkenő sorba rendezése.

Paraméterek

this	A `Leaderboard` struktúra, amely a beolvasott rekordokat tárolja.

Definíció a(z) leaderboard.c fájl 115. sorában.

4.5.2.6 Player__add_to_score()

```
void Player__add_to_score (
            Player * this,
            unsigned int add )
```

Játékos pontszámának növelése.

Beállítja az `update_score`-t, a következő render esetén frissíti a számlálót.

Paraméterek

this	A `Player`, amelynek a pontszámát növeljük.
add	A hozzáadott mennyiség.

Definíció a(z) players.c fájl 199. sorában.

4.5.2.7 Player__construct()

```
Player Player__construct (
            SDL_Renderer * renderer,
            TTF_Font * font,
            char * name,
            char const * meeple_outfit )
```

Definíció a(z) players.c fájl 45. sorában.

4.5.2.8 Player__destroy()

```
void Player__destroy (
            Player * this )
```

Felszabadítja a megadott `Player` struktúra által lefoglalt memóriát.

Paraméterek

this	A `Player` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) players.c fájl 94. sorában.

4.5.2.9 Player__place_meeple()

```
void Player__place_meeple (
            Player * this,
            SDL_Point tile_index )
```

Alattvaló lehelyezése.

Nem viszgálja meg, hogy a mezőkártya valóban megfelelő e.

Paraméterek

this	Az a `Player`, amelynek egy alattvalóját le kell helyezni.
tile_index	A mezőkártya koordinátája a játéktáblán, amire az alattvaló le lesz téve.

Definíció a(z) players.c fájl 155. sorában.

4.5.2.10 Player__reclaim_meeple()

```
void Player__reclaim_meeple (
            Player * this,
            Meeple * to_reclaim )
```

Lehelyezett alattvaló visszaszerzése.

Paraméterek

this	A `Player` struktúra, amelynek egy alattvalóját vissza kell szerezni.
to_reclaim	A visszaszerzendő `Meeple`.

Definíció a(z) players.c fájl 183. sorában.

4.5.2.11 Player__render()

```
void Player__render (
            Player * this,
            SDL_Renderer * renderer,
            TTF_Font * font )
```

Definíció a(z) players.c fájl 102. sorában.

4.6 player.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_PLAYER_H
00002 #define CRCLONE_PLAYER_H
00003
00004 #include <stdlib.h>
00005 #include <stdbool.h>
00006 #include <wchar.h>
00007 #include <SDL2/SDL.h>
00008 #include <SDL2/SDL_ttf.h>
00009
00010 #include "game/meeple.h"
```

```
00011
00012 // Maximum alattvalók száma per játékos
00013 #define MAX_MEEPLES 7
00014 // A játékos nevének max hossza (nullterminálló nélkül)
00015 #define MAX_PLAYER_NAME_LEN 24
00016
00017 typedef struct {
00018     // Általános adatok
00019     // UTF-8 support:  sajnos a TTF_RenderUTF8_Blended valójában char*-t fogad el, ezért nem
      tárolhatom wchar_t*-ben.
00020     char name[(MAX_PLAYER_NAME_LEN + 1) * sizeof(wchar_t)];
00021     unsigned int score;
00022
00023     // Körre vonatkozó
00024     bool has_placed_card;
00025
00026     // Alattvalók
00027     Meeple meeples[MAX_MEEPLES];
00028     size_t meeples_at_hand;
00029
00030     // Textúrák
00031     SDL_Texture* score_counter;
00032     SDL_Texture* handle_texture;
00033     SDL_Texture* stat_panel;
00034     SDL_Texture* own_meeple_texture;
00035
00036     // Kell e frissíteni a pontot
00037     bool update_score;
00038 } Player;
00039 Player Player__construct(SDL_Renderer*, TTF_Font*, char*, char const*);
00040 void Player__place_meeple(Player*, SDL_Point);
00041 void Player__reclaim_meeple(Player*, Meeple*);
00042 void Player__render(Player*, SDL_Renderer*, TTF_Font*);
00043 void Player__add_to_score(Player*, unsigned int);
00044 void Player__destroy(Player*);
00045
00046 // Dicsőséglista rekord
00047 typedef struct {
00048     // Játékos neve
00049     char name[(MAX_PLAYER_NAME_LEN + 1) * sizeof(wchar_t)];
00050
00051     // Játékos rekordja
00052     unsigned int highscore;
00053 } LeaderboardEntry;
00054
00055 typedef struct {
00056     // A rekordfájl elérési útvonala
00057     char const* records_file_path;
00058
00059     // A rekordok
00060     LeaderboardEntry* entries;
00061     size_t entries_size;
00062 } Leaderboard;
00063 Leaderboard* Leaderboard__construct(char const*);
00064 void Leaderboard__destroy(Leaderboard*);
00065 void Leaderboard__sort(Leaderboard*);
00066 bool Leaderboard__load(Leaderboard*);
00067 bool Leaderboard__insert_new(Leaderboard*, Player*);
00068
00069 #endif
```

4.7 include/game/tile.h fájlreferencia

```
#include <stdbool.h>
#include <SDL2/SDL.h>
```

Adatszerkezetek

- struct Tile
- struct CardPile
- struct TilesetWrapper

Makródefiníciók

- #define TILE_SIZE 100
- #define TILE_SIZE_SRC 64
- #define PILE_SIZE 71

Típusdefiníciók

- typedef struct CardPile CardPile

Enumerációk

- enum ConnectionType { NONE = 0U , FIELD , ROAD , CASTLE }
- enum ConnectionDirection { NORTH = 0U , EAST = 1U , SOUTH = 2U , WEST = 3U }
- enum TileType {
  EMPTY = -1 , FIELD_CLOISTER_ROAD_S = 0 , FIELD_CLOISTER_ROAD_NS , FIELD_VILLAGE_ROAD_S
  ,
  FIELD_VILLAGE_ROAD_NS , ROAD_NS , ROAD_NW , ROAD_NWE ,
  ROAD_NSWE , CASTLE_TOWN , CASTLE_PANTHEON , CASTLE_TUNNEL ,
  CASTLE_CORNER_WALL , CASTLE_CORNER_WALL_ROAD_BY , CASTLE_CAP_WALL , CASTLE_CAP_WALL_ROAD_TO
  ,
  CASTLE_CAP_WALL_ROAD_BY , CASTLE_SHIRT_WALL , CASTLE_SHIRT_WALL_ROAD_TO ,
  TILETYPE_SIZE__ }

Függvények

- void Tile__construct (Tile *, TileType, SDL_Point, SDL_FPoint)
- bool Tile__point_in_tile (Tile *, SDL_FPoint)
- void Tile__move_by (Tile *, float, float)
- void Tile__rotate (Tile *)
- void Tile__set_rotation (Tile *, unsigned short)
- void Tile__set_type (Tile *, TileType, unsigned short)
- CardPile * CardPile__construct (void)
- CardPile * CardPile__pop (CardPile *, TileType *)
- CardPile * CardPile__push (CardPile *, TileType)
- void CardPile__destroy (CardPile *)
- TilesetWrapper TilesetWrapper__construct (SDL_Renderer *)
- void TilesetWrapper__destroy (TilesetWrapper *)
- SDL_Rect get_texture_rect_for (TileType)

4.7.1 Makródefiníciók dokumentációja

4.7.1.1 PILE_SIZE

```
#define PILE_SIZE 71
```

Definíció a(z) tile.h fájl 12. sorában.

4.7.1.2 TILE_SIZE

```
#define TILE_SIZE 100
```

Definíció a(z) tile.h fájl 8. sorában.

4.7.1.3 TILE_SIZE_SRC

```
#define TILE_SIZE_SRC 64
```

Definíció a(z) tile.h fájl 10. sorában.

4.7.2 Típusdefiníciók dokumentációja

4.7.2.1 CardPile

```
typedef struct CardPile CardPile
```

4.7.3 Enumerációk dokumentációja

4.7.3.1 ConnectionDirection

```
enum ConnectionDirection
```

Enumeráció-értékek

NORTH	
EAST	
SOUTH	
WEST	

Definíció a(z) tile.h fájl 23. sorában.

4.7.3.2 ConnectionType

```
enum ConnectionType
```

Enumeráció-értékek

NONE	
FIELD	
ROAD	
CASTLE	

Definíció a(z) tile.h fájl 15. sorában.

4.7.3.3 TileType

```
enum TileType
```

Enumeráció-értékek

EMPTY	
FIELD_CLOISTER_ROAD_S	
FIELD_CLOISTER_ROAD_NS	
FIELD_VILLAGE_ROAD_S	
FIELD_VILLAGE_ROAD_NS	
ROAD_NS	
ROAD_NW	
ROAD_NWE	
ROAD_NSWE	
CASTLE_TOWN	
CASTLE_PANTHEON	
CASTLE_TUNNEL	
CASTLE_CORNER_WALL	
CASTLE_CORNER_WALL_ROAD_BY	
CASTLE_CAP_WALL	
CASTLE_CAP_WALL_ROAD_TO	
CASTLE_CAP_WALL_ROAD_BY	
CASTLE_SHIRT_WALL	
CASTLE_SHIRT_WALL_ROAD_TO	
TILETYPE_SIZE__	

Definíció a(z) tile.h fájl 31. sorában.

4.7.4 Függvények dokumentációja

4.7.4.1 CardPile__construct()

```
CardPile * CardPile__construct (
            void  )
```

Definíció a(z) tile.c fájl 11. sorában.

4.7.4.2 CardPile__destroy()

```
void CardPile__destroy (
            CardPile * this )
```

Definíció a(z) tile.c fájl 21. sorában.

4.7.4.3 CardPile__pop()

```
CardPile * CardPile__pop (
            CardPile * this,
            TileType * popped )
```

Definíció a(z) tile.c fájl 28. sorában.

4.7.4.4 CardPile__push()

```
CardPile * CardPile__push (
            CardPile * this,
            TileType new_type )
```

Definíció a(z) tile.c fájl 41. sorában.

4.7.4.5 get_texture_rect_for()

```
SDL_Rect get_texture_rect_for (
            TileType type )
```

Definíció a(z) textures.c fájl 38. sorában.

4.7.4.6 Tile__construct()

```
void Tile__construct (
            Tile * this,
            TileType type,
            SDL_Point board_coords,
            SDL_FPoint offset )
```

Definíció a(z) tile.c fájl 53. sorában.

4.7.4.7 Tile__move_by()

```
void Tile__move_by (
            Tile * this,
            float mvx,
            float mvy )
```

Definíció a(z) tile.c fájl 77. sorában.

4.7.4.8 Tile__point_in_tile()

```
bool Tile__point_in_tile (
            Tile * this,
            SDL_FPoint pt )
```

Definíció a(z) tile.c fájl 68. sorában.

4.7.4.9 Tile__rotate()

```
void Tile__rotate (
            Tile * this )
```

Definíció a(z) tile.c fájl 87. sorában.

4.7.4.10 Tile__set_rotation()

```
void Tile__set_rotation (
            Tile * this,
            unsigned short new_rotation )
```

Definíció a(z) tile.c fájl 94. sorában.

4.7.4.11 Tile__set_type()

```
void Tile__set_type (
            Tile * this,
            TileType new_type,
            unsigned short new_rotation )
```

Definíció a(z) tile.c fájl 119. sorában.

4.7.4.12 TilesetWrapper__construct()

```
TilesetWrapper TilesetWrapper__construct (
            SDL_Renderer * renderer )
```

Definíció a(z) textures.c fájl 11. sorában.

4.7.4.13 TilesetWrapper__destroy()

```
void TilesetWrapper__destroy (
            TilesetWrapper * this )
```

Definíció a(z) textures.c fájl 31. sorában.

4.8 tile.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_TILE_H
00002 #define CRCLONE_TILE_H
00003
00004 #include <stdbool.h>
00005 #include <SDL2/SDL.h>
00006
00007 // A megjelenő mezőkártya mérete
00008 #define TILE_SIZE 100
00009 // A texture atlas-ban egy mezőkártya mérete
00010 #define TILE_SIZE_SRC 64
00011 // A pakli maximális mérete
00012 #define PILE_SIZE 71
00013
00014 // Kártyaoldal típusok
00015 typedef enum {
00016     NONE = 0U,
00017     FIELD,
00018     ROAD,
00019     CASTLE
00020 } ConnectionType;
00021
00022 // Alias-ok a kártyaoldalaknak
00023 typedef enum {
00024     NORTH = 0U,
00025     EAST = 1U,
00026     SOUTH = 2U,
00027     WEST = 3U
00028 } ConnectionDirection;
00029
00030 // Kártyatípusok
00031 typedef enum {
00032     EMPTY = -1,
00033     FIELD_CLOISTER_ROAD_S = 0,
00034     FIELD_CLOISTER_ROAD_NS,
00035     FIELD_VILLAGE_ROAD_S,
00036     FIELD_VILLAGE_ROAD_NS,
00037     ROAD_NS,
00038     ROAD_NW,
00039     ROAD_NWE,
00040     ROAD_NSWE,
00041     CASTLE_TOWN,
00042     CASTLE_PANTHEON,
00043     CASTLE_TUNNEL,
00044     CASTLE_CORNER_WALL,
00045     CASTLE_CORNER_WALL_ROAD_BY,
00046     CASTLE_CAP_WALL,
00047     CASTLE_CAP_WALL_ROAD_TO,
00048     CASTLE_CAP_WALL_ROAD_BY,
00049     CASTLE_SHIRT_WALL,
00050     CASTLE_SHIRT_WALL_ROAD_TO,
00051     TILETYPE_SIZE__ // = 19, trick
00052 } TileType;
```

```
00053
00054 typedef struct {
00055     TileType type;
00056
00057     // A négy oldal:  north, east, south, west (ehhez van a ConnectionDirection).
00058     ConnectionType connections[4];
00059
00060     // Bal felső sarok koordinátái.
00061     SDL_FPoint local_coords, global_coords;
00062
00063     // Táblán a koordináta
00064     SDL_Point board_coords;
00065
00066     // 'unsigned int', mert csak egész foknyi forgatás van engedve (sőt, gyakorlatban az is csak 90
      fok, de legalább van lehetőség egyébre).
00067     unsigned short rotation;
00068
00069     // Egyes kártyákat, amelyeknek minden oldalkapcsolata azonos, nem lehet forgatni.
00070     bool rotatable;
00071
00072     bool is_scored;
00073     bool is_expired;
00074 } Tile;
00075 void Tile__construct(Tile*, TileType, SDL_Point, SDL_FPoint);
00076 bool Tile__point_in_tile(Tile*, SDL_FPoint);
00077 void Tile__move_by(Tile*, float, float);
00078 void Tile__rotate(Tile*);
00079 void Tile__set_rotation(Tile*, unsigned short);
00080 void Tile__set_type(Tile*, TileType, unsigned short);
00081
00082 // Kártyapakli stackként implementálva (nem feltétlen a legokosabb megoldás)
00083 typedef struct CardPile {
00084     TileType card;
00085     struct CardPile* next;
00086 } CardPile;
00087 CardPile* CardPile__construct(void);
00088 CardPile* CardPile__pop(CardPile*, TileType*);
00089 CardPile* CardPile__push(CardPile*, TileType);
00090 void CardPile__destroy(CardPile*);
00091
00092 typedef struct {
00093     SDL_Texture* tile_set;
00094 } TilesetWrapper;
00095 TilesetWrapper TilesetWrapper__construct(SDL_Renderer*);
00096 void TilesetWrapper__destroy(TilesetWrapper*);
00097 SDL_Rect get_texture_rect_for(TileType);
00098
00099 #endif
```

4.9 include/ui.h fájlreferencia

```
#include <stdbool.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "game/tile.h"
#include "game/player.h"
```

Adatszerkezetek

- struct Button
- struct Prompt
- struct MenuScreen
- struct LeaderboardScreen
- struct GameScreen

Makródefiníciók

- #define BOARD_SIZE 71

4.9.1 Makródefiníciók dokumentációja

4.9.1.1 BOARD_SIZE

```
#define BOARD_SIZE 71
```

Definíció a(z) ui.h fájl 12. sorában.

4.10 ui.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_UI_H
00002 #define CRCLONE_UI_H
00003
00004 #include <stdbool.h>
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL_ttf.h>
00007
00008 #include "game/tile.h"
00009 #include "game/player.h"
00010
00011 // A játéktábla mérete (BOARD_SIZE * BOARD_SIZE)
00012 #define BOARD_SIZE 71
00013
00014 // Gomb
00015 typedef struct {
00016     SDL_Rect label_rect, local_rect, global_rect;
00017     SDL_Color bg_color;
00018     char* label;
00019     SDL_Texture* label_texture;
00020     TTF_Font* used_font;
00021 } Button;
00022
00023 // Szöveginput (régi verzió miatt van külön a gombtól, s azzal is lehet implementálni, csak talán így
      szebb)
00024 typedef struct {
00025     Button prompt;
00026 } Prompt;
00027
00028 // Menü állapot
00029 typedef struct {
00030     // Háttérkép
00031     SDL_Texture* background;
00032
00033     // A "menü menü"
00034     SDL_Rect button_container;
00035     Button start_button;
00036     Button lboard_button;
00037 } MenuScreen;
00038
00039 // Dicsőséglista állapot
00040 typedef struct {
00041     Leaderboard* leaderboard;
00042     char syntax_error_msg[128+1];
00043
00044     // A rekordok egy textúrán
00045     SDL_Texture* list_texture;
00046
00047     // "Vissza" gomb
00048     Button back_button;
00049 } LeaderboardScreen;
00050
00051 // Játék állapot
00052 typedef struct {
00053     // A ready_button meg lett e nyomva
00054     bool is_ready;
00055
00056     // Vége van e játéknak
00057     bool is_game_over;
00058
00059     // A tábla smooth mozgatása miatt van egy "global state" a lenyomott nyilakhoz.
00060     int held_arrow_keys[4];
```

```
00061
00062     // A tábla felsősarkának koordinátái
00063     SDL_FPoint board_offset;
00064
00065     // Tábla
00066     Tile** board;
00067     SDL_Texture* board_texture;
00068
00069     // Húzott kártya
00070     Tile* drawn_tile;
00071
00072     // Pakli
00073     CardPile* card_pile;
00074     size_t pile_index;
00075
00076     // TODO: Az első implementálásakor így lehetett normálisan megoldani (TEMP).
00077     SDL_Texture* pile_counter[PILE_SIZE];
00078     SDL_Texture* pile_counter_;
00079     bool update_counter;
00080
00081     // Játékosok, referenciák
00082     Player players[2];
00083     Player* curr_player;
00084     Player* winner;
00085     SDL_Texture* crown_texture;
00086
00087     // Játékosneveknek a szöveginputjai
00088     SDL_Texture* player_input_labels[2];
00089     Prompt player_name_inputs[2];
00090     Prompt* active_input;
00091
00092     // Startgomb, "Kör vége" gomb, "Felad" gomb
00093     Button ready_button, end_turn_button, concede_button;
00094
00095     TilesetWrapper tileset_wrapper;
00096 } GameScreen;
00097
00098 #endif
```

4.11 include/utils.h fájlreferencia

```
#include <stdbool.h>
#include <SDL2/SDL_render.h>
```

Függvények

- char * strdup_ (char const *, bool)
- char * strcatdyn (char *, char const *, bool)
- void remove_last_utf8_char_dyn (char *)
- void destroy_SDL_Texture (SDL_Texture *)
- SDL_Texture * create_SDL_texture_from_BMP (SDL_Renderer *, char const *)

4.11.1 Függvények dokumentációja

4.11.1.1 create_SDL_texture_from_BMP()

```
SDL_Texture * create_SDL_texture_from_BMP (
            SDL_Renderer * renderer,
            char const * source_path )
```

Definíció a(z) utils.c fájl 56. sorában.

4.11.1.2 destroy_SDL_Texture()

```
void destroy_SDL_Texture (
            SDL_Texture * texture )
```

Definíció a(z) utils.c fájl 48. sorában.

4.11.1.3 remove_last_utf8_char_dyn()

```
void remove_last_utf8_char_dyn (
            char * str )
```

Definíció a(z) utils.c fájl 43. sorában.

4.11.1.4 strcatdyn()

```
char * strcatdyn (
            char * original,
            char const * to_cat,
            bool is_utf8 )
```

Definíció a(z) utils.c fájl 25. sorában.

4.11.1.5 strdup_()

```
char * strdup_ (
            char const * str,
            bool is_utf8 )
```

Definíció a(z) utils.c fájl 14. sorában.

4.12 utils.h

Ugrás a fájl dokumentációjához.

```
00001 #ifndef CRCLONE_UTILS_H
00002 #define CRCLONE_UTILS_H
00003
00004 #include <stdbool.h>
00005 #include <SDL2/SDL_render.h>
00006
00007 char* strdup_(char const*, bool);
00008 char* strcatdyn(char*, char const*, bool);
00009 void remove_last_utf8_char_dyn(char*);
00010 void destroy_SDL_Texture(SDL_Texture*);
00011 SDL_Texture* create_SDL_texture_from_BMP(SDL_Renderer*, char const*);
00012
00013 #endif
```

4.13 src/app.c fájlreferencia

```
#include <SDL2/SDL_events.h>
#include <SDL2/SDL_mutex.h>
#include <SDL2/SDL_rect.h>
#include <SDL2/SDL_render.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "game/player.h"
#include "game/tile.h"
#include "ui.h"
#include "app.h"
```

Függvények

- Carcassone * Carcassone__construct (int width, int height, char const *title)

  Initicializálja az összes nézetet, SDL és TTF kontextusokat.
- void Carcassone__destroy (Carcassone *this)

  Felszabadítja a megadott Carcassone struktúra által lefoglalt memóriát.
- void Carcassone__switch_state (Carcassone *this, AppState new_state)
- size_t get_utf8_length (char *str)
- void Carcassone__handle_input (Carcassone *this)

  Inputok kezelése.
- bool Carcassone__check_names_valid (Carcassone *this)
- void Carcassone__Game__init_players (Carcassone *this)

  Játékosok létrehozása.
- void Carcassone__Game__init_pile (Carcassone *this)

  Kártyapakli létrehozása.
- void Carcassone__Game__init_board (Carcassone *this)

  Játéktábla létrehozása.
- void Carcassone__Game__render_board (Carcassone *this)

  Játéktábla renderelése.
- void Carcassone__render_splash_title (Carcassone *this)

  A splash cím renderelése.
- void Carcassone__Game__render_drawn_tile (Carcassone *this)

  A pakli tetején levő kártya renderelése.
- void Carcassone__Game__render_meeples (Carcassone *this)
- void Carcassone__Game__render_game_over (Carcassone *this)
- void Carcassone__indicate_possible_placements (Carcassone *this)
- bool Carcassone__check_if_possible (Carcassone *this)
- void Carcassone__check_scorable_constructs (Carcassone *this)
- void Carcassone__calculate_scores_for_cloister (Carcassone *this, Tile *cloister)
- void Carcassone__calculate_scores_for_road (Carcassone *this, Tile **roads, size_t num, unsigned int point)
- void Carcassone__calculate_scores_for_castle (Carcassone *this, Tile **castles, size_t num, unsigned int point)
- void Carcassone__Game__render_player_stats (Carcassone *this)

A splash cím renderelése.

- void Carcassone__Game__move_board (Carcassone *this, SDL_Scancode key)

  Játéktábla mozgatása.
- bool Carcassone__check_surrounding_tiles (Carcassone *this, SDL_Point tcoords)

  Helyes pozíció ellenőrzése.
- void Carcassone__Game__draw_new (Carcassone *this)

  Új kártya húzása a pakli tetejéről.
- void Carcassone__Game__show_finish_screen (Carcassone *this)
- void Carcassone__run (Carcassone *this)

  A fő programciklus.

4.13.1 Függvények dokumentációja

4.13.1.1 Carcassone__calculate_scores_for_castle()

```
void Carcassone__calculate_scores_for_castle (
            Carcassone * this,
            Tile ** castles,
            size_t num,
            unsigned int point )
```

Definíció a(z) app.c fájl 828. sorában.

4.13.1.2 Carcassone__calculate_scores_for_cloister()

```
void Carcassone__calculate_scores_for_cloister (
            Carcassone * this,
            Tile * cloister )
```

Definíció a(z) app.c fájl 792. sorában.

4.13.1.3 Carcassone__calculate_scores_for_road()

```
void Carcassone__calculate_scores_for_road (
            Carcassone * this,
            Tile ** roads,
            size_t num,
            unsigned int point )
```

Definíció a(z) app.c fájl 809. sorában.

4.13.1.4 Carcassone__check_if_possible()

```
bool Carcassone__check_if_possible (
            Carcassone * this )
```

Definíció a(z) app.c fájl 589. sorában.

4.13.1.5 Carcassone__check_names_valid()

```
bool Carcassone__check_names_valid (
            Carcassone * this )
```

Definíció a(z) app.c fájl 324. sorában.

4.13.1.6 Carcassone__check_scorable_constructs()

```
void Carcassone__check_scorable_constructs (
            Carcassone * this )
```

Definíció a(z) app.c fájl 613. sorában.

4.13.1.7 Carcassone__check_surrounding_tiles()

```
bool Carcassone__check_surrounding_tiles (
            Carcassone * this,
            SDL_Point tcoords )
```

Helyes pozíció ellenőrzése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.
tcoords	A kapott kártya potenciális helye a táblán.

Visszatérési érték

Letehető-e a megfelelő pozícióba az adott kártya.

Definíció a(z) app.c fájl 915. sorában.

4.13.1.8 Carcassone__construct()

```
Carcassone * Carcassone__construct (
             int width,
             int height,
             char const * title )
```

Initicializálja az összes nézetet, SDL és TTF kontextusokat.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a Carcassone__destroy(Carcassone*) függvényt.

Paraméterek

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

Visszatérési érték

Pointer az újonnan létrehozott Carcassone structra.

Definíció a(z) app.c fájl 32. sorában.

4.13.1.9 Carcassone__destroy()

```
void Carcassone__destroy (
             Carcassone * this )
```

Felszabadítja a megadott Carcassone struktúra által lefoglalt memóriát.

Paraméterek

this	A Carcassone struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) app.c fájl 108. sorában.

4.13.1.10 Carcassone__Game__draw_new()

```
void Carcassone__Game__draw_new (
             Carcassone * this )
```

Új kártya húzása a pakli tetejéről.

Húz egy új kártyát, ha kifogyott akkor kilép a menübe (egyelőre).

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 963. sorában.

4.13.1.11 Carcassone__Game__init_board()

```
void Carcassone__Game__init_board (
            Carcassone * this )
```

Játéktábla létrehozása.

Létrehozza a játéktáblát és le is helyezi a kezdőkártyát.

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 419. sorában.

4.13.1.12 Carcassone__Game__init_pile()

```
void Carcassone__Game__init_pile (
            Carcassone * this )
```

Kártyapakli létrehozása.

Véletlenszerűen megkeveri a paklit és létrehozza a mezőkártyákat mindegyikhez.

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 359. sorában.

4.13.1.13 Carcassone__Game__init_players()

```
void Carcassone__Game__init_players (
            Carcassone * this )
```

Játékosok létrehozása.

Inicializálja a két játékost előre megadott adatok alapján (ezek a GameScreenben találhatók).

Paraméterek

this	A Carcassone struktúra.

Definíció a(z) app.c fájl 342. sorában.

4.13.1.14 Carcassone__Game__move_board()

```
void Carcassone__Game__move_board (
            Carcassone * this,
            SDL_Scancode key )
```

Játéktábla mozgatása.

A nyilak segítségével a játéktábla látható részét mozgatja.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.
key	A beolvasott billentyű (a Carcassone__handle_inputs(Carcassone∗)-tól kapja meg).

Definíció a(z) app.c fájl 882. sorában.

4.13.1.15 Carcassone__Game__render_board()

```
void Carcassone__Game__render_board (
            Carcassone * this )
```

Játéktábla renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 450. sorában.

4.13.1.16 Carcassone__Game__render_drawn_tile()

```
void Carcassone__Game__render_drawn_tile (
            Carcassone * this )
```

A pakli tetején levő kártya renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 496. sorában.

4.13.1.17 Carcassone__Game__render_game_over()

```
void Carcassone__Game__render_game_over (
            Carcassone * this )
```

Definíció a(z) app.c fájl 553. sorában.

4.13.1.18 Carcassone__Game__render_meeples()

```
void Carcassone__Game__render_meeples (
            Carcassone * this )
```

Definíció a(z) app.c fájl 525. sorában.

4.13.1.19 Carcassone__Game__render_player_stats()

```
void Carcassone__Game__render_player_stats (
            Carcassone * this )
```

A splash cím renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 851. sorában.

4.13.1.20 Carcassone__Game__show_finish_screen()

```
void Carcassone__Game__show_finish_screen (
            Carcassone * this )
```

Definíció a(z) app.c fájl 1003. sorában.

4.13.1.21 Carcassone__handle_input()

```
void Carcassone__handle_input (
            Carcassone * this )
```

Inputok kezelése.

SDL_Event-öket kezel.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 199. sorában.

4.13.1.22 Carcassone__indicate_possible_placements()

```
void Carcassone__indicate_possible_placements (
            Carcassone * this )
```

Definíció a(z) app.c fájl 569. sorában.

4.13.1.23 Carcassone__render_splash_title()

```
void Carcassone__render_splash_title (
            Carcassone * this )
```

A splash cím renderelése.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 485. sorában.

4.13.1.24 Carcassone__run()

```
void Carcassone__run (
            Carcassone * this )
```

A fő programciklus.

Paraméterek

this	A Carcassone struktúra, ami tartalmazza az SDL kontextust.

Definíció a(z) app.c fájl 1028. sorában.

4.13.1.25 Carcassone__switch_state()

```
void Carcassone__switch_state (
            Carcassone * this,
            AppState new_state )
```

Definíció a(z) app.c fájl 130. sorában.

4.13.1.26 get_utf8_length()

```
size_t get_utf8_length (
            char * str )
```

Definíció a(z) app.c fájl 176. sorában.

4.14 app.c

Ugrás a fájl dokumentációjához.

```
00001 #include <SDL2/SDL_events.h>
00002 #include <SDL2/SDL_mutex.h>
00003 #include <SDL2/SDL_rect.h>
00004 #include <SDL2/SDL_render.h>
00005 #include <stdio.h>
00006 #include <stdlib.h>
00007 #include <string.h>
00008 #include <time.h>
00009 #include <limits.h>
00010 #include <SDL2/SDL.h>
00011 #include <SDL2/SDL_ttf.h>
00012
00013 #include "game/player.h"
00014 #include "game/tile.h"
00015 #include "ui.h"
00016 #include "app.h"
00017
00018 #ifdef _CRCLONE_DEBUG
00019 #include "debug/debugmalloc.h"
00020 #endif
00021
00032 Carcassone* Carcassone__construct(int width, int height, char const* title)
00033 {
00034     Carcassone* new_app = malloc(sizeof(Carcassone));
00035     new_app->width = width;
00036     new_app->height = height;
00037     new_app->is_running = false;
00038     new_app->window = NULL;
00039     new_app->window_icon = NULL;
00040     new_app->splash_title = NULL;
00041     new_app->renderer = NULL;
00042     new_app->state = MENU;
00043
00044     // Összes dolog betöltése
00045     if(SDL_Init(SDL_INIT_VIDEO) != 0 || TTF_Init() != 0) {
```

```
00046         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "Nem sikerült inicializálni az SDL2-t!");
00047         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "%s", SDL_GetError());
00048         return NULL;
00049     }
00050 
00051     new_app->window = SDL_CreateWindow(title,
00052         SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, width, height, SDL_WINDOW_SHOWN);
00053     if(new_app->window == NULL) {
00054         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "Nem sikerült létrehozni az ablakot!");
00055         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "%s", SDL_GetError());
00056         return NULL;
00057     }
00058 
00059     new_app->renderer = SDL_CreateRenderer(new_app->window, -1, SDL_RENDERER_ACCELERATED);
00060     if(new_app->renderer == NULL) {
00061         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "Nem sikerült létrehozni a renderert!");
00062         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "%s", SDL_GetError());
00063         return NULL;
00064     }
00065 
00066     new_app->default_font = TTF_OpenFont("res/fonts/Sedan_SC/sedan_sc.ttf", 48);
00067     new_app->small_font = TTF_OpenFont("res/fonts/Sedan_SC/sedan_sc.ttf", 36);
00068     if(new_app->default_font == NULL || new_app->small_font == NULL) {
00069         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "Nem sikerült betölteni a fontot!");
00070         SDL_LogCritical(SDL_LOG_CATEGORY_APPLICATION, "%s", TTF_GetError());
00071         return NULL;
00072     }
00073 
00074     // Nem gond, ha nincs (bár úgy jobban néz ki)
00075     new_app->splash_title = SDL_CreateTextureFromSurface(new_app->renderer,
      SDL_LoadBMP("res/splash_title.bmp"));
00076     if(new_app->splash_title == NULL) {
00077         SDL_LogError(SDL_LOG_CATEGORY_RENDER, "Nem sikerült betölteni a címképet!");
00078         SDL_LogError(SDL_LOG_CATEGORY_RENDER, "%s", SDL_GetError());
00079     }
00080 
00081     // Nem gond, ha nincs
00082     new_app->window_icon = SDL_LoadBMP("res/crc_icon.bmp");
00083     if(new_app->window_icon != NULL) {
00084         SDL_SetWindowIcon(new_app->window, new_app->window_icon);
00085     } else {
00086         SDL_LogWarn(SDL_LOG_CATEGORY_APPLICATION, "Nem sikerült betölteni az appikont!");
00087         SDL_LogWarn(SDL_LOG_CATEGORY_APPLICATION, "%s", SDL_GetError());
00088     }
00089 
00090     srand((unsigned int)(time(NULL) % SHRT_MAX));
00091 
00092     Carcassone__Menu__construct(new_app);
00093     Carcassone__Game__construct(new_app);
00094     Carcassone__Lboard__construct(new_app);
00095 
00096     SDL_StartTextInput();
00097 
00098     new_app->smutex = SDL_CreateMutex();
00099 
00100     return new_app;
00101 }
00102 
00108 void Carcassone__destroy(Carcassone* this)
00109 {
00110     if(SDL_LockMutex(this->smutex) == -1) SDL_UnlockMutex(this->smutex);
00111     SDL_DestroyMutex(this->smutex);
00112     this->is_running = false;
00113     if(SDL_IsTextInputActive()) SDL_StopTextInput();
00114 
00115     Carcassone__Menu__destroy(this);
00116     Carcassone__Game__destroy(this);
00117     Carcassone__Lboard__destroy(this);
00118 
00119     if(this->window_icon != NULL)   SDL_FreeSurface(this->window_icon);
00120     if(this->splash_title != NULL)  SDL_DestroyTexture(this->splash_title);
00121     if(this->window != NULL)        SDL_DestroyWindow(this->window);
00122     if(this->renderer != NULL)      SDL_DestroyRenderer(this->renderer);
00123     if(this->default_font != NULL)  TTF_CloseFont(this->default_font);
00124     if(TTF_WasInit() != 0)          TTF_Quit();
00125     if(SDL_WasInit(0) != 0)   SDL_Quit();
00126 
00127     free(this);
00128 }
00129 
00130 void Carcassone__switch_state(Carcassone* this, AppState new_state)
00131 {
00132     if(this->state == new_state) return;
00133 
00134     this->state = new_state;
00135     switch(new_state) {
00136         case MENU:
```

```
00137             this->game_screen->is_ready = false;
00138             this->game_screen->is_game_over = false;
00139             break;
00140         case GAME:
00141             this->game_screen->is_ready = false;
00142             this->game_screen->is_game_over = false;
00143             Carcassone__Game__init_board(this);
00144             Carcassone__Game__init_pile(this);
00145             break;
00146         case LEADERBOARD:
00147             Carcassone__Lboard__init_list_texture(this);
00148             break;
00149     }
00150 }
00151
00152
00153 static void remove_last_utf8_char(char* str) {
00154     if(str == NULL) return;
00155     size_t len = strlen(str);
00156     if (len == 0) return;
00157
00158     char* p = (char*)str + len - 1;
00159
00160     // UTF-8 enkódolási szabályok:
00161     // - 1-byte karakter:  0xxxxxxx
00162     // - 2-byte karakter:  110xxxxx 10xxxxxx
00163     // - 3-byte karakter:  1110xxxx 10xxxxxx 10xxxxxx
00164     // - 4-byte karakter:  11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
00165
00166     while (p >= (char*)str) {
00167         // Megkeressük az első, nem "continuation" bitet; odatesszük a \0-t.
00168         if ((*p & 0b11000000) != 0b10000000) {
00169             *p = '\0';
00170             break;
00171         }
00172         --p;
00173     }
00174 }
00175
00176 size_t get_utf8_length(char* str)
00177 {
00178     if(str == NULL) return 0U;
00179
00180     char* p = str;
00181     size_t len = 0U;
00182     while(*p != '\0') {
00183         if ((*p & 0b11000000) != 0b10000000) {
00184             ++len;
00185         }
00186         ++p;
00187     }
00188
00189     return len;
00190 }
00191
00199 void Carcassone__handle_input(Carcassone* this)
00200 {
00201     SDL_Event event;
00202     SDL_PollEvent(&event);
00203
00204     switch(event.type) {
00205         case SDL_QUIT:
00206             this->is_running = false;
00207             break;
00208         case SDL_TEXTINPUT:
00209             if(!(this->state == GAME && !this->game_screen->is_ready)) break;
00210             if(event.text.text[0] == ' ') break;
00211
00212             if(get_utf8_length(this->game_screen->active_input->prompt.label) < 24) {
00213                 Carcassone__Prompt__edit(this, this->game_screen->active_input, event.text.text,
      true);
00214             }
00215             break;
00216         case SDL_TEXTEDITING:
00217             break;
00218         case SDL_KEYUP:
00219             if(!(SDL_SCANCODE_RIGHT <= event.key.keysym.scancode && event.key.keysym.scancode <=
      SDL_SCANCODE_UP)) return;
00220
00221             unsigned int key = event.key.keysym.scancode - SDL_SCANCODE_RIGHT;
00222             this->game_screen->held_arrow_keys[key] = false;
00223
00224             break;
00225         case SDL_KEYDOWN:
00226             switch(event.key.keysym.sym) {
00227                 case SDLK_ESCAPE:
00228                     this->is_running = false;
```

```
00229                     break;
00230                 case SDLK_BACKSPACE:
00231                     if(get_utf8_length(this->game_screen->active_input->prompt.label) > 0) {
00232                         remove_last_utf8_char(this->game_screen->active_input->prompt.label);
00233                         Carcassone__Prompt__edit(this, this->game_screen->active_input,
      this->game_screen->active_input->prompt.label, false);
00234                     }
00235                     break;
00236                 case SDLK_r:
00237                     if(this->game_screen->is_ready) Tile__rotate(this->game_screen->drawn_tile);
00238                     break;
00239                 case SDLK_d:  // TODO: temp
00240                     if(this->game_screen->is_game_over) break;
00241                     Carcassone__Game__draw_new(this);
00242                     Carcassone__check_scorable_constructs(this);
00243                     break;
00244                 default:
00245                     if(!(SDL_SCANCODE_RIGHT <= event.key.keysym.scancode && event.key.keysym.scancode
      <= SDL_SCANCODE_UP)) break;
00246 
00247                     unsigned int key = event.key.keysym.scancode - SDL_SCANCODE_RIGHT;
00248                     this->game_screen->held_arrow_keys[key] = true;
00249 
00250                     break;
00251             }
00252             break;
00253         case SDL_MOUSEBUTTONDOWN:
00254             if(this->state == MENU) {
00255                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->menu_screen->start_button.global_rect)) {
00256                     Carcassone__switch_state(this, GAME);
00257                 }
00258                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->menu_screen->lboard_button.global_rect)) {
00259                     Carcassone__switch_state(this, LEADERBOARD);
00260                 }
00261             }
00262 
00263             if(this->state == LEADERBOARD) {
00264                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->lboard_screen->back_button.global_rect)) {
00265                     Carcassone__switch_state(this, MENU);
00266                 }
00267             }
00268 
00269             if(this->state != GAME) break;
00270 
00271             if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->game_screen->player_name_inputs[0].prompt.global_rect)) {
00272                 this->game_screen->active_input = &this->game_screen->player_name_inputs[0];
00273             } else if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->game_screen->player_name_inputs[1].prompt.global_rect)) {
00274                 this->game_screen->active_input = &this->game_screen->player_name_inputs[1];
00275             }
00276 
00277             if(!this->game_screen->is_ready) {
00278                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->game_screen->ready_button.global_rect)) {
00279                     if(!Carcassone__check_names_valid(this)) break;
00280                     this->game_screen->is_ready = true;
00281                     Carcassone__Game__init_players(this);
00282                 }
00283             } else {
00284                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->game_screen->end_turn_button.global_rect)) {
00285                     if(this->game_screen->curr_player->has_placed_card) {
00286                         Carcassone__Game__draw_new(this);
00287                         Carcassone__check_scorable_constructs(this);
00288                     }
00289                 }
00290                 if(SDL_PointInRect(&(SDL_Point){event.button.x, event.button.y},
      &this->game_screen->concede_button.global_rect))
00291                     Carcassone__Game__show_finish_screen(this);
00292 
00293                 for(int y = 0; y < BOARD_SIZE; ++y) {
00294                     for(int x = 0; x < BOARD_SIZE; ++x) {
00295                         Tile* curr_tile = &this->game_screen->board[x][y];
00296                         if(!this->game_screen->curr_player->has_placed_card) {
00297                             if(Tile__point_in_tile(curr_tile, (SDL_FPoint){event.button.x,
      event.button.y})
00298                                     && curr_tile->type == EMPTY
00299                                     && Carcassone__check_surrounding_tiles(this, (SDL_Point){x, y})
00300                             ) {
00301                                 Tile__set_type(curr_tile, this->game_screen->drawn_tile->type,
      this->game_screen->drawn_tile->rotation);
00302                                 // Tile__set_rotation(curr_tile,
      this->game_screen->drawn_tile->rotation);
```

```
00303
00304                                 this->game_screen->curr_player->has_placed_card = true;
00305                             }
00306                         } else {
00307                             if(Tile__point_in_tile(curr_tile, (SDL_FPoint){event.button.x,
      event.button.y})
00308                                     && curr_tile->type != EMPTY && !curr_tile->is_scored &&
      !curr_tile->is_expired) {
00309                                 Player__place_meeple(this->game_screen->curr_player, (SDL_Point){x,
      y});
00310                             }
00311                         }
00312                     }
00313                 }
00314             }
00315
00316             break;
00317         default:
00318             break;
00319     }
00320
00321     if(this->game_screen->is_ready) Carcassone__Game__move_board(this, 0U);
00322 }
00323
00324 bool Carcassone__check_names_valid(Carcassone* this)
00325 {
00326     for(size_t p = 0U; p < 2; ++p) {
00327         if(strlen(this->game_screen->player_name_inputs[p].prompt.label) == 0) return false;
00328     }
00329
00330     if(strcmp(this->game_screen->player_name_inputs[0].prompt.label,
      this->game_screen->player_name_inputs[1].prompt.label) == 0) return false;
00331
00332     return true;
00333 }
00334
00342 void Carcassone__Game__init_players(Carcassone* this) // TODO
00343 {
00344     for(size_t p = 0U; p < 2; ++p) {
00345         this->game_screen->players[p] =
00346             Player__construct(this->renderer, this->small_font,
      this->game_screen->player_name_inputs[p].prompt.label, "./res/meeple_base.bmp");
00347     }
00348
00349     this->game_screen->curr_player = &this->game_screen->players[0];
00350 }
00351
00359 void Carcassone__Game__init_pile(Carcassone* this)
00360 {
00361     // ?  TODO: put this in a config file
00362     TileType pile[PILE_SIZE] = {
00363         FIELD_CLOISTER_ROAD_S, FIELD_CLOISTER_ROAD_S,
00364         FIELD_CLOISTER_ROAD_S, FIELD_CLOISTER_ROAD_S,
00365         FIELD_CLOISTER_ROAD_NS, FIELD_CLOISTER_ROAD_NS,
00366         FIELD_VILLAGE_ROAD_S, FIELD_VILLAGE_ROAD_S,
00367         FIELD_VILLAGE_ROAD_NS, FIELD_VILLAGE_ROAD_NS,
00368         ROAD_NS, ROAD_NS, ROAD_NS, ROAD_NS,
00369         ROAD_NS, ROAD_NS, ROAD_NS, ROAD_NS,
00370         ROAD_NW, ROAD_NW, ROAD_NW, ROAD_NW,
00371         ROAD_NW, ROAD_NW, ROAD_NW, ROAD_NW,
00372         ROAD_NW,
00373         ROAD_NWE, ROAD_NWE, ROAD_NWE, ROAD_NWE,
00374         ROAD_NSWE,
00375         CASTLE_CAP_WALL, CASTLE_CAP_WALL,
00376         CASTLE_CAP_WALL, CASTLE_CAP_WALL,
00377         CASTLE_CAP_WALL,
00378         CASTLE_CAP_WALL_ROAD_BY, CASTLE_CAP_WALL_ROAD_BY,
00379         CASTLE_CAP_WALL_ROAD_BY, CASTLE_CAP_WALL_ROAD_BY,
00380         CASTLE_CAP_WALL_ROAD_BY, CASTLE_CAP_WALL_ROAD_BY,
00381         CASTLE_CAP_WALL_ROAD_BY, CASTLE_CAP_WALL_ROAD_BY,
00382         CASTLE_CAP_WALL_ROAD_TO, CASTLE_CAP_WALL_ROAD_TO,
00383         CASTLE_CAP_WALL_ROAD_TO,
00384         CASTLE_CORNER_WALL_ROAD_BY, CASTLE_CORNER_WALL_ROAD_BY,
00385         CASTLE_CORNER_WALL_ROAD_BY, CASTLE_CORNER_WALL_ROAD_BY,
00386         CASTLE_CORNER_WALL, CASTLE_CORNER_WALL,
00387         CASTLE_CORNER_WALL, CASTLE_CORNER_WALL,
00388         CASTLE_TOWN, CASTLE_TOWN, CASTLE_TOWN,
00389         CASTLE_PANTHEON, CASTLE_PANTHEON,
00390         CASTLE_TUNNEL, CASTLE_TUNNEL,
00391         CASTLE_TUNNEL, CASTLE_TUNNEL,
00392         CASTLE_SHIRT_WALL, CASTLE_SHIRT_WALL,
00393         CASTLE_SHIRT_WALL, CASTLE_SHIRT_WALL,
00394         CASTLE_SHIRT_WALL_ROAD_TO, CASTLE_SHIRT_WALL_ROAD_TO
00395     };
00396
00397     for(size_t n = 0U; n < PILE_SIZE; ++n) {
00398         size_t rand_index = (size_t) (rand() % PILE_SIZE); // SIZE_MAX > INT_MAX (legtöbb esetben, de
```

```
    ha nem akkor meg wrappel)
00399         TileType temp = pile[n];
00400
00401         pile[n] = pile[rand_index];
00402         pile[rand_index] = temp;
00403     }
00404
00405     for(size_t n = 0U; n < PILE_SIZE; ++n) {
00406         this->game_screen->card_pile = CardPile__push(this->game_screen->card_pile, pile[n]);
00407     }
00408
00409     this->game_screen->pile_index = 0U;
00410 }
00411
00419 void Carcassone__Game__init_board(Carcassone* this)
00420 {
00421     // !  debugmalloc falsely claims the code below produces a memory leak
00422     // !  it does not, free happens inside Carcassone__Game__destroy() without any issues
00423     this->game_screen->board = malloc(BOARD_SIZE * sizeof(Tile*));
00424     for(size_t n = 0U; n < BOARD_SIZE; ++n) {
00425         this->game_screen->board[n] = malloc(BOARD_SIZE * sizeof(Tile));
00426     }
00427
00428     for(int y = 0; y < BOARD_SIZE; ++y) {
00429         for(int x = 0; x < BOARD_SIZE; ++x) {
00430             Tile__construct(&this->game_screen->board[x][y], EMPTY, (SDL_Point){x, y},
    this->game_screen->board_offset);
00431             Tile__move_by(&this->game_screen->board[x][y],
00432                 -BOARD_SIZE / 2.0f + 3.0f,
00433                 -BOARD_SIZE / 2.0f + 4.0f);
00434         }
00435     }
00436
00437     // Kezdőkártya
00438     Tile__construct(&this->game_screen->board[BOARD_SIZE / 2][BOARD_SIZE / 2],
    CASTLE_CAP_WALL_ROAD_BY,
00439         (SDL_Point){BOARD_SIZE / 2, BOARD_SIZE / 2}, this->game_screen->board_offset);
00440     Tile__move_by(&this->game_screen->board[BOARD_SIZE / 2][BOARD_SIZE / 2],
00441                 -BOARD_SIZE / 2.0f + 3.0f,
00442                 -BOARD_SIZE / 2.0f + 4.0f);
00443 }
00444
00450 void Carcassone__Game__render_board(Carcassone* this)
00451 {
00452     SDL_Rect viewport_rect =
00453         {this->game_screen->board_offset.x, this->game_screen->board_offset.y, 600, this->height -
    this->game_screen->board_offset.y - 10};
00454     SDL_SetRenderTarget(this->renderer, this->game_screen->board_texture);
00455     SDL_RenderClear(this->renderer);
00456
00457     for(int y = 0U; y < BOARD_SIZE; ++y) {
00458         for(int x = 0U; x < BOARD_SIZE; ++x) {
00459             Tile* curr_tile = &this->game_screen->board[x][y];
00460             SDL_Rect tile_rect = {
00461                 curr_tile->local_coords.x,
00462                 curr_tile->local_coords.y,
00463                 TILE_SIZE, TILE_SIZE
00464             };
00465
00466             SDL_Rect ts_rect = get_texture_rect_for(curr_tile->type);
00467             SDL_RenderCopyEx(this->renderer, this->game_screen->tileset_wrapper.tile_set, &ts_rect,
    &tile_rect,
00468                 curr_tile->rotation, NULL, SDL_FLIP_NONE);
00469
00470             // Fehér keret a celláknak
00471             SDL_SetRenderDrawColor(this->renderer, 255, 255, 255, 255);
00472             SDL_RenderDrawRect(this->renderer, &tile_rect);
00473         }
00474     }
00475
00476     SDL_SetRenderTarget(this->renderer, NULL);
00477     SDL_RenderCopy(this->renderer, this->game_screen->board_texture, NULL, &viewport_rect);
00478 }
00479
00485 void Carcassone__render_splash_title(Carcassone* this)
00486 {
00487     SDL_RenderCopy(this->renderer, this->splash_title, NULL,
00488         &(SDL_Rect){this->game_screen->board_offset.x, 0, 400, 120});
00489 }
00490
00496 void Carcassone__Game__render_drawn_tile(Carcassone* this)
00497 {
00498     SDL_Rect ts_rect = get_texture_rect_for(this->game_screen->drawn_tile->type);
00499
00500     SDL_RenderCopyEx(this->renderer, this->game_screen->tileset_wrapper.tile_set, &ts_rect,
00501         &(SDL_Rect){this->game_screen->board_offset.x + 450, this->game_screen->board_offset.y -
    TILE_SIZE, TILE_SIZE-5, TILE_SIZE-5},
```

```
00502         this->game_screen->drawn_tile->rotation, NULL, SDL_FLIP_NONE);
00503
00504     SDL_Rect pile_index_rect = {this->game_screen->board_offset.x + 555,
      this->game_screen->board_offset.y - TILE_SIZE/2.0f, 50, 50};
00505     if(PILE_SIZE - this->game_screen->pile_index < 10) {
00506         pile_index_rect.w /= 2;
00507         pile_index_rect.x += pile_index_rect.w/2;
00508     }
00509
00510     SDL_RenderCopy(this->renderer, this->game_screen->pile_counter[this->game_screen->pile_index],
      NULL, &pile_index_rect);
00511
00512     // TODO: temp
00513     int mx, my;
00514     SDL_GetMouseState(&mx, &my);
00515     if(!this->game_screen->curr_player->has_placed_card) {
00516         SDL_RenderCopyEx(this->renderer, this->game_screen->tileset_wrapper.tile_set, &ts_rect,
00517             &(SDL_Rect){mx - TILE_SIZE / 2, my - TILE_SIZE / 2, TILE_SIZE, TILE_SIZE},
00518             this->game_screen->drawn_tile->rotation, NULL, SDL_FLIP_NONE);
00519     } else if(this->game_screen->curr_player->meeples_at_hand > 0) {
00520         SDL_RenderCopy(this->renderer, this->game_screen->curr_player->meeples[0].texture, NULL,
00521             &(SDL_Rect){mx - 64 / 2, my - 64 / 2, 64, 64});
00522     }
00523 }
00524
00525 void Carcassone__Game__render_meeples(Carcassone* this)
00526 {
00527     SDL_Rect rect;
00528     for(int y = 0; y < BOARD_SIZE; ++y) {
00529         for(int x = 0; x < BOARD_SIZE; ++x) {
00530             Tile* placed_on = &this->game_screen->board[x][y];
00531             unsigned int offset_f = 0U;
00532             unsigned int offset_b = 0U;
00533             for(size_t p = 0U; p < 2; ++p) {
00534                 for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00535                     Meeple* curr_m = &this->game_screen->players[p].meeples[m];
00536                     if(!curr_m->is_placed) continue;
00537                     if(curr_m->x != placed_on->board_coords.x || curr_m->y !=
      placed_on->board_coords.y) continue;
00538
00539                     rect = (SDL_Rect){
00540                         p == 0 ?  placed_on->global_coords.x + 10*offset_f :
      placed_on->global_coords.x + TILE_SIZE - 32 - 10*offset_b,
00541                         p == 0 ?  placed_on->global_coords.y :  placed_on->global_coords.y + TILE_SIZE
      - 32,
00542                         32, 32
00543                     };
00544                     SDL_RenderCopy(this->renderer, curr_m->texture, NULL, &rect);
00545                     if(p == 0) ++offset_f;
00546                     else ++offset_b;
00547                 }
00548             }
00549         }
00550     }
00551 }
00552
00553 void Carcassone__Game__render_game_over(Carcassone* this)
00554 {
00555     if(this->game_screen->winner == NULL) return;
00556
00557     SDL_SetRenderDrawColor(this->renderer, COLOR_RED);
00558     if(this->game_screen->winner == &this->game_screen->players[0]) {
00559         SDL_RenderDrawRect(this->renderer, &(SDL_Rect){10, this->height - 710, 300, 700});
00560         SDL_RenderCopy(this->renderer, this->game_screen->crown_texture, NULL,
00561             &(SDL_Rect){150 - 64, this->height - 710 - 128, 128, 128});
00562     } else {
00563         SDL_RenderDrawRect(this->renderer, &(SDL_Rect){this->width - 310, this->height - 710, 300,
      700});
00564         SDL_RenderCopy(this->renderer, this->game_screen->crown_texture, NULL,
00565             &(SDL_Rect){this->width - 310/2 - 64, this->height - 710 - 128, 128, 128});
00566     }
00567 }
00568
00569 void Carcassone__indicate_possible_placements(Carcassone* this)
00570 {
00571     if(this->game_screen->curr_player->has_placed_card) return;
00572
00573     for(int y = 0; y < BOARD_SIZE; ++y) {
00574         for(int x = 0; x < BOARD_SIZE; ++x) {
00575             Tile* curr_tile = &this->game_screen->board[x][y];
00576             if(curr_tile->type != EMPTY) continue;
00577
00578             if(Carcassone__check_surrounding_tiles(this, (SDL_Point){x, y})) {
00579                 SDL_SetRenderDrawColor(this->renderer, 20, 240, 100, 255);
00580                 SDL_RenderFillRect(this->renderer,
00581                 &(SDL_Rect){
00582                         curr_tile->global_coords.x, curr_tile->global_coords.y, TILE_SIZE, TILE_SIZE}
```

```
00583                 );
00584             }
00585         }
00586     }
00587 }
00588
00589 bool Carcassone__check_if_possible(Carcassone* this)
00590 {
00591     if(this->game_screen->drawn_tile == NULL) return false;
00592
00593     for(int y = 0; y < BOARD_SIZE; ++y) {
00594         for(int x = 0; x < BOARD_SIZE; ++x) {
00595             Tile* curr_tile = &this->game_screen->board[x][y];
00596             if(curr_tile->type != EMPTY) continue;
00597
00598             for(unsigned int rot = 0U; rot < 4; ++rot) {
00599                 if(Carcassone__check_surrounding_tiles(this, (SDL_Point){x, y})) {
00600                     return true;
00601                 }
00602
00603                 if(!this->game_screen->drawn_tile->rotatable) break;
00604                 else Tile__rotate(this->game_screen->drawn_tile);
00605             }
00606
00607         }
00608     }
00609
00610     return false;
00611 }
00612
00613 void Carcassone__check_scorable_constructs(Carcassone* this)
00614 {
00615     for(int y = 0; y < BOARD_SIZE; ++y) {
00616         for(int x = 0; x < BOARD_SIZE; ++x) {
00617             Tile* curr_tile = &this->game_screen->board[x][y];
00618             if(curr_tile->type == EMPTY || curr_tile->is_scored) continue;
00619
00620             if(!this->game_screen->is_game_over) {
00621                 if(curr_tile->type == FIELD_CLOISTER_ROAD_S || curr_tile->type ==
    FIELD_CLOISTER_ROAD_NS) {
00622                     curr_tile->is_expired = true;
00623                     bool finished = true;
00624                     for(int yrel = y-1; yrel <= y+1; ++yrel) {
00625                         for(int xrel = x-1; xrel <= x+1; ++xrel) {
00626                             if(yrel < 0 || yrel >= BOARD_SIZE || xrel < 0 || xrel >= BOARD_SIZE)
    continue;
00627                             if(this->game_screen->board[xrel][yrel].type == EMPTY) {
00628                                 finished = false;
00629                                 break;
00630                             }
00631                         }
00632
00633                         if(!finished) break;
00634                     }
00635
00636                     if(finished) {
00637                         Carcassone__calculate_scores_for_cloister(this, curr_tile);
00638                     }
00639                 }
00640             }
00641
00642             // WIP
00643             bool has_road_out = false;
00644             for(unsigned int dir = NORTH; dir <= WEST; ++dir) {
00645                 if(curr_tile->connections[dir] == ROAD) {
00646                     has_road_out = true;
00647                     break;
00648                 }
00649             }
00650
00651             if(has_road_out) {
00652                 bool finished = true;
00653                 unsigned int num_of_connected_roads = 0U;
00654
00655                 SDL_Point dir_rel_coords[4] = {
00656                     {0, -1}, {1, 0}, {0, 1}, {-1, 0}
00657                 };
00658
00659                 // TODO: NO
00660                 size_t visited_idx = 0U;
00661                 Tile* visited[PILE_SIZE] = {0};
00662                 size_t stack_idx = 0U;
00663                 Tile* stack[PILE_SIZE] = {0};
00664
00665                 stack[stack_idx] = curr_tile;
00666                 ++stack_idx;
00667                 while(stack_idx != 0U) {
```

```
00668                     --stack_idx;
00669                     Tile* popped = stack[stack_idx];
00670                     if(!this->game_screen->is_game_over && popped->type == EMPTY) {
00671                         finished = false;
00672                         break;
00673                     }
00674 
00675                     bool is_visited = false;
00676                     for(size_t vi = 0U; vi < visited_idx; ++vi) {
00677                         if(popped->board_coords.x == visited[vi]->board_coords.x &&
    popped->board_coords.y == visited[vi]->board_coords.y) {
00678                             is_visited = true;
00679                             break;
00680                         }
00681                     }
00682                     if(is_visited) {
00683                         continue;
00684                     } else {
00685                         if(popped->type != EMPTY) ++num_of_connected_roads;
00686                         if(0 <= popped->type && popped->type <= 3) ++num_of_connected_roads;
00687                         visited[visited_idx] = popped;
00688                         ++visited_idx;
00689                     }
00690 
00691                     for(unsigned int dir = NORTH; dir <= WEST; ++dir) {
00692                         int new_x = popped->board_coords.x + dir_rel_coords[dir].x;
00693                         int new_y = popped->board_coords.y + dir_rel_coords[dir].y;
00694                         if(new_x < 0 || new_x >= BOARD_SIZE || new_y < 0 || new_y >= BOARD_SIZE)
    continue;
00695 
00696                         if(popped->connections[dir] == ROAD) {
00697                             stack[stack_idx] = &this->game_screen->board[new_x][new_y];
00698                             ++stack_idx;
00699                         }
00700 
00701                     }
00702 
00703                     if(!finished) break;
00704                 }
00705 
00706                 if(finished) {
00707                     Carcassone__calculate_scores_for_road(this, visited, visited_idx,
    num_of_connected_roads);
00708                     for(size_t i = 0U; i < visited_idx; ++i) {
00709                         visited[i]->is_scored = true;
00710                     }
00711                 }
00712             }
00713             // WIP
00714 
00715             // WIP
00716             bool has_castle_out = false;
00717             for(unsigned int dir = NORTH; dir <= WEST; ++dir) {
00718                 if(curr_tile->connections[dir] == CASTLE) {
00719                     has_castle_out = true;
00720                     break;
00721                 }
00722             }
00723 
00724             if(has_castle_out) {
00725                 bool finished = true;
00726                 unsigned int num_of_connected_castles = 0U;
00727 
00728                 SDL_Point dir_rel_coords[4] = {
00729                     {0, -1}, {1, 0}, {0, 1}, {-1, 0}
00730                 };
00731 
00732                 // TODO: NO
00733                 size_t visited_idx = 0U;
00734                 Tile* visited[PILE_SIZE] = {0};
00735                 size_t stack_idx = 0U;
00736                 Tile* stack[PILE_SIZE] = {0};
00737 
00738                 stack[stack_idx] = curr_tile;
00739                 ++stack_idx;
00740                 while(stack_idx != 0U) {
00741                     --stack_idx;
00742                     Tile* popped = stack[stack_idx];
00743                     if(!this->game_screen->is_game_over && popped->type == EMPTY) {
00744                         finished = false;
00745                         break;
00746                     }
00747 
00748                     bool is_visited = false;
00749                     for(size_t vi = 0U; vi < visited_idx; ++vi) {
00750                         if(popped->board_coords.x == visited[vi]->board_coords.x &&
    popped->board_coords.y == visited[vi]->board_coords.y) {
```

```
00751                             is_visited = true;
00752                             break;
00753                         }
00754                     }
00755                     if(is_visited) {
00756                         continue;
00757                     } else {
00758                         if(popped->type != EMPTY) ++num_of_connected_castles;
00759                         if(8 <= popped->type && popped->type <= 9) ++num_of_connected_castles;
00760                         visited[visited_idx] = popped;
00761                         ++visited_idx;
00762                     }
00763 
00764                     for(unsigned int dir = NORTH; dir <= WEST; ++dir) {
00765                         int new_x = popped->board_coords.x + dir_rel_coords[dir].x;
00766                         int new_y = popped->board_coords.y + dir_rel_coords[dir].y;
00767                         if(new_x < 0 || new_x >= BOARD_SIZE || new_y < 0 || new_y >= BOARD_SIZE)
      continue;
00768 
00769                         if(popped->connections[dir] == CASTLE) {
00770                             stack[stack_idx] = &this->game_screen->board[new_x][new_y];
00771                             ++stack_idx;
00772                         }
00773 
00774                     }
00775 
00776                     if(!finished) break;
00777                 }
00778 
00779                 if(finished) {
00780                     Carcassone__calculate_scores_for_castle(this, visited, visited_idx,
      num_of_connected_castles);
00781                     for(size_t i = 0U; i < visited_idx; ++i) {
00782                         visited[i]->is_scored = true;
00783                     }
00784                 }
00785             }
00786             // WIP
00787         }
00788     }
00789 }
00790 
00791 // TODO: NO
00792 void Carcassone__calculate_scores_for_cloister(Carcassone* this, Tile* cloister)
00793 {
00794     for(size_t p = 0U; p < 2; ++p) {
00795         for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00796             Meeple* curr_m = &this->game_screen->players[p].meeples[m];
00797             if(!curr_m->is_placed) continue;
00798             if(curr_m->x != cloister->board_coords.x || curr_m->y != cloister->board_coords.y)
      continue;
00799 
00800             Player__add_to_score(&this->game_screen->players[p], 3);
00801             Player__reclaim_meeple(&this->game_screen->players[p], curr_m);
00802             cloister->is_scored = true;
00803             DBG_LOG("Added score (cloister) to:  %s", this->game_screen->players[p].name);
00804         }
00805     }
00806 }
00807 
00808 // TODO
00809 void Carcassone__calculate_scores_for_road(Carcassone* this, Tile** roads, size_t num, unsigned int
      point)
00810 {
00811     for(size_t i = 0U; i < num; ++i) {
00812         for(size_t p = 0U; p < 2; ++p) {
00813             for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00814                 Meeple* curr_m = &this->game_screen->players[p].meeples[m];
00815                 if(!curr_m->is_placed) continue;
00816                 if(curr_m->x != roads[i]->board_coords.x || curr_m->y != roads[i]->board_coords.y)
      continue;
00817 
00818                 Player__add_to_score(&this->game_screen->players[p], point);
00819                 Player__reclaim_meeple(&this->game_screen->players[p], curr_m);
00820                 roads[i]->is_scored = true;
00821                 DBG_LOG("Added score (road) to:  %s", this->game_screen->players[p].name);
00822             }
00823         }
00824     }
00825 }
00826 
00827 // TODO
00828 void Carcassone__calculate_scores_for_castle(Carcassone* this, Tile** castles, size_t num, unsigned
      int point)
00829 {
00830     for(size_t i = 0U; i < num; ++i) {
00831         for(size_t p = 0U; p < 2; ++p) {
```

```
00832             for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00833                 Meeple* curr_m = &this->game_screen->players[p].meeples[m];
00834                 if(!curr_m->is_placed) continue;
00835                 if(curr_m->x != castles[i]->board_coords.x || curr_m->y != castles[i]->board_coords.y)
      continue;
00836
00837                 Player__add_to_score(&this->game_screen->players[p], point);
00838                 Player__reclaim_meeple(&this->game_screen->players[p], curr_m);
00839                 castles[i]->is_scored = true;
00840                 DBG_LOG("Added score (castle) to:  %s", this->game_screen->players[p].name);
00841             }
00842         }
00843     }
00844 }
00845
00851 void Carcassone__Game__render_player_stats(Carcassone* this)
00852 {
00853     if(this->game_screen->players[0].stat_panel != NULL) {
00854         Player__render(&this->game_screen->players[0], this->renderer, this->small_font);
00855         SDL_RenderCopy(this->renderer, this->game_screen->players[0].stat_panel, NULL,
00856             &(SDL_Rect){10, this->height - 710, 300, 700});
00857         if(this->game_screen->curr_player == &this->game_screen->players[0]) {
00858             SDL_SetRenderDrawColor(this->renderer, COLOR_SALMON);
00859             SDL_RenderDrawRect(this->renderer, &(SDL_Rect){10, this->height - 710, 300, 700});
00860         }
00861     }
00862
00863     if(this->game_screen->players[1].stat_panel != NULL) {
00864         Player__render(&this->game_screen->players[1], this->renderer, this->small_font);
00865         SDL_RenderCopy(this->renderer, this->game_screen->players[1].stat_panel, NULL,
00866             &(SDL_Rect){this->width - 310, this->height - 710, 300, 700});
00867         if(this->game_screen->curr_player == &this->game_screen->players[1]) {
00868             SDL_SetRenderDrawColor(this->renderer, COLOR_SALMON);
00869             SDL_RenderDrawRect(this->renderer, &(SDL_Rect){this->width - 310, this->height - 710, 300,
      700});
00870         }
00871     }
00872 }
00873
00882 void Carcassone__Game__move_board(Carcassone* this, SDL_Scancode key)
00883 {
00884     float mvx = 0.0f;
00885     float mvy = 0.0f;
00886
00887     // TODO: should depend on dt
00888     if(this->game_screen->held_arrow_keys[0]) {
00889         mvx -= 0.02f;
00890     }
00891     if(this->game_screen->held_arrow_keys[1]) {
00892         mvx += 0.02f;
00893     }
00894     if(this->game_screen->held_arrow_keys[2]) {
00895         mvy -= 0.02f;
00896     }
00897     if(this->game_screen->held_arrow_keys[3]) {
00898         mvy += 0.02f;
00899     }
00900
00901     for(int y = 0U; y < BOARD_SIZE; ++y) {
00902         for(int x = 0U; x < BOARD_SIZE; ++x) {
00903             Tile__move_by(&this->game_screen->board[x][y], mvx, mvy);
00904         }
00905     }
00906 }
00907
00915 bool Carcassone__check_surrounding_tiles(Carcassone* this, SDL_Point tcoords)
00916 {
00917     int x = tcoords.x;
00918     int y = tcoords.y;
00919
00920     // TODO: for loop
00921     // még egy korábbi refaktor előtti állapotból maradt így, amikor nem lehetett volna ciklussal
      megoldani
00922     bool next_to_placed = false;
00923     if(y - 1 >= 0) {
00924         if(this->game_screen->board[x][y-1].type != EMPTY) next_to_placed = true;
00925         if(this->game_screen->board[x][y-1].connections[SOUTH] !=
      this->game_screen->drawn_tile->connections[NORTH]
00926             && this->game_screen->board[x][y-1].connections[SOUTH] != NONE) {
00927             return false;
00928         }
00929     }
00930     if(y + 1 < BOARD_SIZE) {
00931         if(this->game_screen->board[x][y+1].type != EMPTY) next_to_placed = true;
00932         if(this->game_screen->board[x][y+1].connections[NORTH] !=
      this->game_screen->drawn_tile->connections[SOUTH]
00933             && this->game_screen->board[x][y+1].connections[NORTH] != NONE) {
```

```
00934             return false;
00935         }
00936     }
00937     if(x - 1 >= 0) {
00938         if(this->game_screen->board[x-1][y].type != EMPTY) next_to_placed = true;
00939         if(this->game_screen->board[x-1][y].connections[EAST] !=
      this->game_screen->drawn_tile->connections[WEST]
00940             && this->game_screen->board[x-1][y].connections[EAST] != NONE) {
00941             return false;
00942         }
00943     }
00944     if(x + 1 < BOARD_SIZE) {
00945         if(this->game_screen->board[x+1][y].type != EMPTY) next_to_placed = true;
00946         if(this->game_screen->board[x+1][y].connections[WEST] !=
      this->game_screen->drawn_tile->connections[EAST]
00947             && this->game_screen->board[x+1][y].connections[WEST] != NONE) {
00948             return false;
00949         }
00950     }
00951
00952     return next_to_placed;
00953
00954 }
00955
00963 void Carcassone__Game__draw_new(Carcassone* this) // TODO
00964 {
00965     TileType next_type;
00966     this->game_screen->card_pile = CardPile__pop(this->game_screen->card_pile, &next_type);
00967     if(this->game_screen->card_pile != NULL) {
00968         Tile__construct(this->game_screen->drawn_tile, next_type, (SDL_Point){-1, -1}, (SDL_FPoint){0,
      0});
00969         ++this->game_screen->pile_index;
00970
00971         if(this->game_screen->curr_player != NULL) {
00972             this->game_screen->curr_player->has_placed_card = false;
00973         }
00974
00975         // TODO: NO
00976         if(this->game_screen->curr_player->name == this->game_screen->players[0].name) {
00977             this->game_screen->curr_player = &this->game_screen->players[1];
00978         } else {
00979             this->game_screen->curr_player = &this->game_screen->players[0];
00980         }
00981     } else {
00982
00983     }
00984
00985     if(this->game_screen->pile_index >= PILE_SIZE) {
00986         Carcassone__Game__show_finish_screen(this);
00987     } else if(!this->game_screen->curr_player->has_placed_card &&
      !Carcassone__check_if_possible(this)) {
00988         Carcassone__Game__draw_new(this);
00989         DBG_LOG("Skipped a card.");
00990     }
00991
00992 }
00993
00994 static unsigned int return_to_menu(unsigned int interval, void* this)
00995 {
00996     Carcassone* thiz = (Carcassone*) this;
00997     SDL_LockMutex(thiz->smutex);
00998     Carcassone__switch_state(thiz, MENU);
00999     SDL_UnlockMutex(thiz->smutex);
01000     return 0;
01001 }
01002
01003 void Carcassone__Game__show_finish_screen(Carcassone* this) // TODO: rename to "wrap up" or some shi
01004 {
01005     this->game_screen->is_game_over = true;
01006     Carcassone__check_scorable_constructs(this);
01007
01008     if(this->game_screen->players[0].score == this->game_screen->players[1].score) {
01009         this->game_screen->winner = NULL;
01010     } else {
01011         this->game_screen->winner =
01012             this->game_screen->players[0].score > this->game_screen->players[1].score ?
      &this->game_screen->players[0] :  &this->game_screen->players[1];
01013     }
01014     if(this->lboard_screen->leaderboard != NULL) {
01015         Leaderboard__insert_new(this->lboard_screen->leaderboard, this->game_screen->winner);
01016         Leaderboard__load(this->lboard_screen->leaderboard);
01017     }
01018
01019     DBG_LOG("The winner is:  %s", this->game_screen->winner == NULL ? "tie" :
      this->game_screen->winner->name);
01020     SDL_AddTimer(5000, return_to_menu, this);
01021 }
```

```
01022
01028 void Carcassone__run(Carcassone* this)
01029 {
01030     this->is_running = true;
01031
01032     // Ez egy másik projektemből van átmásolva.
01033     Uint64 now;
01034     Uint64 last = 0U;
01035     float accumulator = 0U;
01036     while(this->is_running) {
01037         // Calculate delta time
01038         now = SDL_GetTicks64();
01039         accumulator += MIN((float)(now - last), 1000.0f / FPS);
01040         last = now;
01041         for(float delta; accumulator >= (delta = 1.0f / FPS); accumulator -= delta) {
01042             SDL_LockMutex(this->smutex);
01043             switch(this->state) {
01044             case MENU:
01045                 Carcassone__Menu__render(this);
01046                 break;
01047             case LEADERBOARD:
01048                 Carcassone__Lboard__render(this);
01049                 break;
01050             case GAME:
01051                 Carcassone__Game__render(this);
01052                 break;
01053             }
01054             SDL_UnlockMutex(this->smutex);
01055
01056             SDL_RenderPresent(this->renderer);
01057             Carcassone__handle_input(this);
01058         }
01059
01060     }
01061 }
```

4.15 src/main.c fájlreferencia

```
#include <SDL2/SDL_log.h>
#include "app.h"
```

Függvények

- int main (void)

  A program entry pointja.

4.15.1 Függvények dokumentációja

4.15.1.1 main()

```
int main (
            void  )
```

A program entry pointja.

Létrehozza és inicializálja a programciklust, illetve biztosítja a lefoglalt memória helyes felszabadulását.

Visszatérési érték

0

Definíció a(z) main.c fájl 11. sorában.

4.16 main.c

Ugrás a fájl dokumentációjához.

```
00001 #include <SDL2/SDL_log.h>
00002 #include "app.h"
00003
00011 int main(void)
00012 {
00013 #ifdef _CRCLONE_DEBUG
00014         SDL_LogSetAllPriority(SDL_LOG_PRIORITY_DEBUG);
00015 #else
00016         SDL_LogSetAllPriority(SDL_LOG_PRIORITY_CRITICAL);
00017 #endif
00018
00019     Carcassone* game = Carcassone__construct(1280, 960, "Carcassone másolat - Prog1 NHF");
00020     if(game != NULL) {
00021         Carcassone__run(game);
00022         Carcassone__destroy(game);
00023     }
00024
00025     return 0;
00026 }
```

4.17 src/player/leaderboard.c fájlreferencia

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include "utils.h"
#include "app.h"
#include "game/player.h"
```

Függvények

- Leaderboard * Leaderboard__construct (char const *records_file_path)

  Létrehozza a dicsőséglistát.
- void Leaderboard__destroy (Leaderboard *this)

  Felszabadítja a megadott `Leaderboard` struktúra által lefoglalt memóriát.
- bool Leaderboard__load (Leaderboard *this)

  Betölti a rekordfájl tartalmát.
- void Leaderboard__sort (Leaderboard *this)

  A beolvasott rekordok csökkenő sorba rendezése.
- bool Leaderboard__insert_new (Leaderboard *this, Player *new)

  Új rekord beszúrása és mentése.

4.17.1 Függvények dokumentációja

4.17.1.1 Leaderboard__construct()

```
Leaderboard * Leaderboard__construct (
            char const * records_file_path )
```

Létrehozza a dicsőséglistát.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Leaderboard__destroy` függvényt.

Paraméterek

records_file_path	A fájl elérési útvonala, ahonnan beolvassa a játékosneveket és rekordokat.

Visszatérési érték

Pointer az újonnan létrehozott `Leaderboard`-ra.

Definíció a(z) leaderboard.c fájl 22. sorában.

4.17.1.2 Leaderboard__destroy()

```
void Leaderboard__destroy (
            Leaderboard * this )
```

Felszabadítja a megadott `Leaderboard` struktúra által lefoglalt memóriát.

Paraméterek

this	A `Leaderboard` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) leaderboard.c fájl 41. sorában.

4.17.1.3 Leaderboard__insert_new()

```
bool Leaderboard__insert_new (
            Leaderboard * this,
            Player * new )
```

Új rekord beszúrása és mentése.

Megjegyzés: A visszatérési érték nem feltétlen van kihasználva, elhanyagolható.

Paraméterek

this	A `Leaderboard` struktúra, amihez tartozó rekordfájlba történjen a mentés.
new	A potenciálisan új, rekordot elérő játékos.

Visszatérési érték

Már tartalmazta e a dicsőséglista a játékost.

Definíció a(z) leaderboard.c fájl 129. sorában.

4.17.1.4 Leaderboard__load()

```
bool Leaderboard__load (
            Leaderboard * this )
```

Betölti a rekordfájl tartalmát.

Érdemes meghívni a `Leaderboard__sort`-ot ezután.

Paraméterek

this	A `Leaderboard` struktúra, amelyhez már tartozik egy rekordfájl.

Visszatérési érték

Hibamentesen lefutott e.

Definíció a(z) leaderboard.c fájl 56. sorában.

4.17.1.5 Leaderboard__sort()

```
void Leaderboard__sort (
            Leaderboard * this )
```

A beolvasott rekordok csökkenő sorba rendezése.

Paraméterek

this	A `Leaderboard` struktúra, amely a beolvasott rekordokat tárolja.

Definíció a(z) leaderboard.c fájl 115. sorában.

4.18 leaderboard.c

Ugrás a fájl dokumentációjához.

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <wchar.h>
00005
00006 #include "utils.h"
00007 #include "app.h"
00008 #include "game/player.h"
00009
00010 #ifdef _CRCLONE_DEBUG
00011 #include "debug/debugmalloc.h"
00012 #endif
00013
00022 Leaderboard* Leaderboard__construct(char const* records_file_path)
00023 {
00024     Leaderboard* new_lboard = malloc(sizeof(Leaderboard));
00025     new_lboard->records_file_path = strdup_(records_file_path, true);
00026     new_lboard->entries = NULL;
```

```
00027     new_lboard->entries_size = 0U;
00028
00029     if(!Leaderboard__load(new_lboard)) {
00030         return NULL;
00031     } else {
00032         return new_lboard;
00033     }
00034 }
00035
00041 void Leaderboard__destroy(Leaderboard* this)
00042 {
00043     free((char*)this->records_file_path);
00044     free(this->entries);
00045     free(this);
00046 }
00047
00056 bool Leaderboard__load(Leaderboard* this)
00057 {
00058     if(this->records_file_path == NULL) return false;
00059
00060     FILE* records_f = fopen(this->records_file_path, "r");
00061     if(records_f == NULL) return false;
00062
00063     // Beolvasandó adatok.
00064     char name[(MAX_PLAYER_NAME_LEN + 1) * sizeof(wchar_t)];
00065     unsigned int record;
00066
00067     size_t entries_size = 1U;
00068     this->entries = malloc(entries_size * sizeof(LeaderboardEntry));
00069     size_t entry_idx = 0U;
00070
00071     // Addig olvas amíg nem kap EOF-ot vagy nem sikerült 2 adatot beolvasnia és eltárolnia a megfelelő
      adattípusban
00072     int read_status;
00073     while((read_status = fscanf(records_f, "%s %u", name, &record)) != EOF) {
00074         if(read_status != 2 && read_status != EOF) {
00075             // Hibás fájlformátum.
00076             return false;
00077         }
00078
00079         // Az `entries` méretének növelése amíg még van adat.
00080         if(entry_idx >= entries_size) {
00081             entries_size *= 2;
00082             this->entries = realloc(this->entries, entries_size * sizeof(LeaderboardEntry));
00083         }
00084
00085         // Rekord beszúrása.
00086         strcpy(this->entries[entry_idx].name, name);
00087         this->entries[entry_idx].highscore = record;
00088
00089         ++entry_idx;
00090     }
00091     this->entries_size = entry_idx;
00092
00093     fclose(records_f);
00094
00095     return true;
00096 }
00097
00101 static int entry_cmp(void const* e1, void const* e2)
00102 {
00103     LeaderboardEntry s1 = *(LeaderboardEntry const*)e1;
00104     LeaderboardEntry s2 = *(LeaderboardEntry const*)e2;
00105
00106     if (s1.highscore > s2.highscore) return -1;
00107     if (s1.highscore < s2.highscore) return 1;
00108     return 0;
00109 }
00115 void Leaderboard__sort(Leaderboard* this)
00116 {
00117     qsort(this->entries, this->entries_size, sizeof(LeaderboardEntry), entry_cmp);
00118 }
00119
00129 bool Leaderboard__insert_new(Leaderboard* this, Player* new)
00130 {
00131     if(new == NULL) return false;
00132
00133     // Rekord potenciális frissítése.
00134     bool found = false;
00135     for(size_t i = 0U; i < this->entries_size; ++i) {
00136         if(strcmp(this->entries[i].name, new->name) == 0) {
00137             found = true;
00138             if(this->entries[i].highscore < new->score) {
00139                 this->entries[i].highscore = new->score;
00140             }
00141             break;
00142         }
```

```
00143     }
00144
00145     // Ha nem volt eddig benne, akkor biztosan új rekord.
00146     if(!found) {
00147         ++this->entries_size;
00148         this->entries = realloc(this->entries, this->entries_size * sizeof(LeaderboardEntry));
00149
00150         strcpy(this->entries[this->entries_size - 1].name, new->name);
00151         this->entries[this->entries_size - 1].highscore = new->score;
00152     }
00153
00154     // A rekordfájl frissítése.
00155     FILE* records_f = fopen(this->records_file_path, "w");
00156     if(records_f == NULL) return false;
00157
00158     for(size_t i = 0U; i < this->entries_size; ++i) {
00159         fprintf(records_f, "%s %u\n", this->entries[i].name, this->entries[i].highscore);
00160     }
00161
00162     fclose(records_f);
00163
00164     return found;
00165 }
00166
```

4.19 src/player/players.c fájlreferencia

```
#include <stdio.h>
#include <string.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "utils.h"
#include "app.h"
#include "game/player.h"
#include "game/meeple.h"
```

Függvények

- Meeple Meeple__construct (SDL_Texture *texture)

  Létrehoz egy alattvalót.
- Player Player__construct (SDL_Renderer *renderer, TTF_Font *font, char *name, char const *meeple_↩ outfit)
- void Player__destroy (Player *this)

  Felszabadítja a megadott `Player` struktúra által lefoglalt memóriát.
- void Player__render (Player *this, SDL_Renderer *renderer, TTF_Font *font)
- void Player__place_meeple (Player *this, SDL_Point tile_index)

  Alattvaló lehelyezése.
- void Player__reclaim_meeple (Player *this, Meeple *to_reclaim)

  Lehelyezett alattvaló visszaszerzése.
- void Player__add_to_score (Player *this, unsigned int add)

  Játékos pontszámának növelése.

4.19.1 Függvények dokumentációja

4.19.1.1 Meeple__construct()

```
Meeple Meeple__construct (
            SDL_Texture * texture )
```

Létrehoz egy alattvalót.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Meeple__destroy` függvényt.

Paraméterek

texture	A kinézete.

Visszatérési érték

Az új `Meeple` struktúra.

Definíció a(z) players.c fájl 23. sorában.

4.19.1.2 Player__add_to_score()

```
void Player__add_to_score (
            Player * this,
            unsigned int add )
```

Játékos pontszámának növelése.

Beállítja az `update_score`-t, a következő render esetén frissíti a számlálót.

Paraméterek

this	A `Player`, amelynek a pontszámát növeljük.
add	A hozzáadott mennyiség.

Definíció a(z) players.c fájl 199. sorában.

4.19.1.3 Player__construct()

```
Player Player__construct (
            SDL_Renderer * renderer,
            TTF_Font * font,
            char * name,
            char const * meeple_outfit )
```

Definíció a(z) players.c fájl 45. sorában.

4.19.1.4 Player__destroy()

```
void Player__destroy (
            Player * this )
```

Felszabadítja a megadott `Player` struktúra által lefoglalt memóriát.

Paraméterek

this	A `Player` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) players.c fájl 94. sorában.

4.19.1.5 Player__place_meeple()

```
void Player__place_meeple (
            Player * this,
            SDL_Point tile_index )
```

Alattvaló lehelyezése.

Nem viszgálja meg, hogy a mezőkártya valóban megfelelő e.

Paraméterek

this	Az a `Player`, amelynek egy alattvalóját le kell helyezni.
tile_index	A mezőkártya koordinátája a játéktáblán, amire az alattvaló le lesz téve.

Definíció a(z) players.c fájl 155. sorában.

4.19.1.6 Player__reclaim_meeple()

```
void Player__reclaim_meeple (
            Player * this,
            Meeple * to_reclaim )
```

Lehelyezett alattvaló visszaszerzése.

Paraméterek

this	A `Player` struktúra, amelynek egy alattvalóját vissza kell szerezni.
to_reclaim	A visszaszerzendő `Meeple`.

Definíció a(z) players.c fájl 183. sorában.

4.19.1.7 Player__render()

```
void Player__render (
            Player * this,
            SDL_Renderer * renderer,
            TTF_Font * font )
```

Definíció a(z) players.c fájl 102. sorában.

4.20 players.c

Ugrás a fájl dokumentációjához.

```
00001 #include <stdio.h>
00002 #include <string.h>
00003 #include <SDL2/SDL.h>
00004 #include <SDL2/SDL_ttf.h>
00005
00006 #include "utils.h"
00007 #include "app.h"
00008 #include "game/player.h"
00009 #include "game/meeple.h"
00010
00011 #ifdef _CRCLONE_DEBUG
00012 #include "debug/debugmalloc.h"
00013 #endif
00014
00023 Meeple Meeple__construct(SDL_Texture* texture)
00024 {
00025     Meeple new_meeple = {
00026         .x = -1,
00027         .y = -1,
00028         .is_placed = false,
00029         .texture = texture
00030     };
00031
00032     return new_meeple;
00033 }
00034
00035 // /**
00036 //  * @brief Felszabadítja a megadott `Meeple` struktúra által lefoglalt memóriát.
00037 //  *
00038 //  * @param this A `Meeple` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.
00039 //  */
00040 // void Meeple__destroy(Meeple* this)
00041 // {
00042 //     destroy_SDL_Texture(this->texture);
00043 // }
00044
00045 Player Player__construct(SDL_Renderer* renderer, TTF_Font* font, char* name, char const*
      meeple_outfit)
00046 {
00047     Player new_player = {
00048         .score = 0U,
00049         .has_placed_card = false,
00050         .meeples_at_hand = MAX_MEEPLES,
00051         .update_score = true
00052     };
00053     strcpy(new_player.name, get_utf8_length(name) <= 24 ?  name :  "INVALID");
00054
00055     // Alattvalók létrehozása a játékos számára.
00056     new_player.own_meeple_texture = create_SDL_texture_from_BMP(renderer, meeple_outfit);
00057     if(new_player.own_meeple_texture != NULL) {
00058         for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00059             new_player.meeples[m] = Meeple__construct(new_player.own_meeple_texture);
00060         }
00061     }
00062
00063
00064     // Stat panel létrehozása
00065     // Név handle létrehozása
00066     SDL_Surface* handle_surface = TTF_RenderUTF8_Blended_Wrapped(font, new_player.name, (SDL_Color){0,
      0, 0, 255}, 300);
00067     if(handle_surface != NULL) {
00068         new_player.handle_texture = SDL_CreateTextureFromSurface(renderer, handle_surface);
00069         SDL_FreeSurface(handle_surface);
00070     }
00071
```

```
00072     // Pontszámoló létrehozása
00073     char score_string[2] = "0";
00074     sprintf(score_string, "%u", new_player.score);
00075
00076     SDL_Surface* score_surface = TTF_RenderUTF8_Blended(font, score_string, (SDL_Color){0, 0, 0,
      255});
00077     new_player.score_counter = NULL;
00078     if(score_surface != NULL) {
00079         new_player.score_counter = SDL_CreateTextureFromSurface(renderer, score_surface);
00080         SDL_FreeSurface(score_surface);
00081     }
00082
00083     // Magának a panelnek a létrehozása
00084     new_player.stat_panel = SDL_CreateTexture(renderer, SDL_PIXELFORMAT_RGBA8888,
      SDL_TEXTUREACCESS_TARGET, 300, 700);
00085
00086     return new_player;
00087 }
00088
00094 void Player__destroy(Player* this)
00095 {
00096     destroy_SDL_Texture(this->stat_panel); // TODO: ENNÉL VAN A HIBA
00097     destroy_SDL_Texture(this->handle_texture);
00098     destroy_SDL_Texture(this->score_counter);
00099     destroy_SDL_Texture(this->own_meeple_texture);
00100 }
00101
00102 void Player__render(Player* this, SDL_Renderer* renderer, TTF_Font* font)
00103 {
00104     if(this->stat_panel != NULL) {
00105         SDL_SetRenderTarget(renderer, this->stat_panel);
00106         SDL_SetRenderDrawBlendMode(renderer, SDL_BLENDMODE_BLEND);
00107
00108         SDL_SetRenderDrawColor(renderer, COLOR_WHITE);
00109         SDL_RenderFillRect(renderer, NULL);
00110
00111         char score_str[5+1];
00112         snprintf(score_str, 6, "%u", this->score);
00113         if(this->update_score) {
00114             SDL_Surface* score_surface = TTF_RenderUTF8_Blended(font, score_str, (SDL_Color){0, 0, 0,
      255});
00115             if(score_surface != NULL) {
00116                 SDL_DestroyTexture(this->score_counter);
00117                 this->score_counter = SDL_CreateTextureFromSurface(renderer, score_surface);
00118                 SDL_FreeSurface(score_surface);
00119             }
00120
00121             this->update_score = false;
00122         }
00123
00124         int w, h;
00125         TTF_SizeUTF8(font, this->name, &w, &h);
00126         if(this->handle_texture != NULL)
00127             SDL_RenderCopy(renderer, this->handle_texture, NULL, &(SDL_Rect){0, 0, MIN(300, w), h});
00128         TTF_SizeUTF8(font, score_str, &w, &h);
00129         if(this->score_counter != NULL) {
00130             SDL_RenderCopy(renderer, this->score_counter, NULL,
00131                 &(SDL_Rect){15, 95, w, h});
00132         }
00133
00134         unsigned int offset = 0U;
00135         for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00136             if(this->meeples[m].is_placed) continue;
00137
00138             SDL_RenderCopy(renderer, this->meeples[m].texture, NULL,
00139                 &(SDL_Rect){10 + 30*offset, 200, 64, 64});
00140             ++offset;
00141         }
00142
00143         SDL_SetRenderTarget(renderer, NULL);
00144     }
00145 }
00146
00155 void Player__place_meeple(Player* this, SDL_Point tile_index)
00156 {
00157     if(this->meeples_at_hand <= 0) return;
00158
00159     // Az első nem-lehelyezett alattvaló kiválasztása.
00160     Meeple* to_be_placed = NULL;
00161     for(size_t m = 0U; m < MAX_MEEPLES; ++m) {
00162         if(!this->meeples[m].is_placed) {
00163             to_be_placed = &this->meeples[m];
00164             break;
00165         }
00166     }
00167
00168     // Lehelyezés
```

```
00169     if(to_be_placed != NULL) {
00170         to_be_placed->is_placed = true;
00171         to_be_placed->x = tile_index.x;
00172         to_be_placed->y = tile_index.y;
00173         --this->meeples_at_hand;
00174     }
00175 }
00176
00183 void Player__reclaim_meeple(Player* this, Meeple* to_reclaim)
00184 {
00185     if(to_reclaim == NULL || !to_reclaim->is_placed || this->meeples_at_hand >= MAX_MEEPLES) return;
00186
00187     ++this->meeples_at_hand;
00188     to_reclaim->is_placed = false;
00189 }
00190
00199 void Player__add_to_score(Player* this, unsigned int add)
00200 {
00201     this->score += add;
00202     this->update_score = true;
00203 }
```

4.21 src/textures.c fájlreferencia

```
#include <SDL2/SDL.h>
#include "game/tile.h"
#include "app.h"
#include "utils.h"
```

Függvények

- TilesetWrapper TilesetWrapper__construct (SDL_Renderer *renderer)
- void TilesetWrapper__destroy (TilesetWrapper *this)
- SDL_Rect get_texture_rect_for (TileType type)

4.21.1 Függvények dokumentációja

4.21.1.1 get_texture_rect_for()

```
SDL_Rect get_texture_rect_for (
            TileType type )
```

Definíció a(z) textures.c fájl 38. sorában.

4.21.1.2 TilesetWrapper__construct()

```
TilesetWrapper TilesetWrapper__construct (
            SDL_Renderer * renderer )
```

Definíció a(z) textures.c fájl 11. sorában.

4.21.1.3 TilesetWrapper__destroy()

```
void TilesetWrapper__destroy (
            TilesetWrapper * this )
```

Definíció a(z) textures.c fájl 31. sorában.

4.22 textures.c

Ugrás a fájl dokumentációjához.

```
00001 #include <SDL2/SDL.h>
00002
00003 #include "game/tile.h"
00004 #include "app.h" // TODO: create a font manager
00005 #include "utils.h"
00006
00007 #ifdef _CRCLONE_DEBUG
00008 #include "debug/debugmalloc.h"
00009 #endif
00010
00011 TilesetWrapper TilesetWrapper__construct(SDL_Renderer* renderer)
00012 {
00013     TilesetWrapper new_tswrapper = {
00014         .tile_set = NULL
00015     };
00016
00017     SDL_Surface* tileset_img = SDL_LoadBMP("res/tileset.bmp");
00018     if(tileset_img == NULL) {
00019         SDL_LogCritical(SDL_LOG_CATEGORY_ASSERT, "Nem sikerült a res/tileset.bmp beolvasása!\n%s",
      SDL_GetError());
00020         return new_tswrapper;
00021     }
00022
00023     if(renderer == NULL) return new_tswrapper;
00024
00025     new_tswrapper.tile_set = SDL_CreateTextureFromSurface(renderer, tileset_img);
00026     SDL_FreeSurface(tileset_img);
00027
00028     return new_tswrapper;
00029 }
00030
00031 void TilesetWrapper__destroy(TilesetWrapper* this)
00032 {
00033     if(this != NULL) {
00034         destroy_SDL_Texture(this->tile_set);
00035     }
00036 }
00037
00038 SDL_Rect get_texture_rect_for(TileType type)
00039 {
00040     TileType type_index = type;
00041
00042     SDL_Rect rect;
00043     rect.x = (type_index % (TILETYPE_SIZE__ / 2)) * TILE_SIZE_SRC;
00044     rect.y = (type_index / (TILETYPE_SIZE__ / 2)) * TILE_SIZE_SRC;
00045     rect.w = TILE_SIZE_SRC;
00046     rect.h = TILE_SIZE_SRC;
00047
00048     return rect;
00049 }
```

4.23 src/tile.c fájlreferencia

```
#include <math.h>
#include <SDL2/SDL.h>
#include "app.h"
#include "game/tile.h"
```

Függvények

- CardPile * CardPile__construct (void)
- void CardPile__destroy (CardPile *this)
- CardPile * CardPile__pop (CardPile *this, TileType *popped)
- CardPile * CardPile__push (CardPile *this, TileType new_type)
- void Tile__construct (Tile *this, TileType type, SDL_Point board_coords, SDL_FPoint offset)
- bool Tile__point_in_tile (Tile *this, SDL_FPoint pt)
- void Tile__move_by (Tile *this, float mvx, float mvy)
- void Tile__rotate (Tile *this)
- void Tile__set_rotation (Tile *this, unsigned short new_rotation)
- void Tile__set_type (Tile *this, TileType new_type, unsigned short new_rotation)

4.23.1 Függvények dokumentációja

4.23.1.1 CardPile__construct()

```
CardPile * CardPile__construct (
            void  )
```

Definíció a(z) tile.c fájl 11. sorában.

4.23.1.2 CardPile__destroy()

```
void CardPile__destroy (
            CardPile * this )
```

Definíció a(z) tile.c fájl 21. sorában.

4.23.1.3 CardPile__pop()

```
CardPile * CardPile__pop (
            CardPile * this,
            TileType * popped )
```

Definíció a(z) tile.c fájl 28. sorában.

4.23.1.4 CardPile__push()

```
CardPile * CardPile__push (
            CardPile * this,
            TileType new_type )
```

Definíció a(z) tile.c fájl 41. sorában.

4.23.1.5 Tile__construct()

```
void Tile__construct (
            Tile * this,
            TileType type,
            SDL_Point board_coords,
            SDL_FPoint offset )
```

Definíció a(z) tile.c fájl 53. sorában.

4.23.1.6 Tile__move_by()

```
void Tile__move_by (
            Tile * this,
            float mvx,
            float mvy )
```

Definíció a(z) tile.c fájl 77. sorában.

4.23.1.7 Tile__point_in_tile()

```
bool Tile__point_in_tile (
            Tile * this,
            SDL_FPoint pt )
```

Definíció a(z) tile.c fájl 68. sorában.

4.23.1.8 Tile__rotate()

```
void Tile__rotate (
            Tile * this )
```

Definíció a(z) tile.c fájl 87. sorában.

4.23.1.9 Tile__set_rotation()

```
void Tile__set_rotation (
            Tile * this,
            unsigned short new_rotation )
```

Definíció a(z) tile.c fájl 94. sorában.

4.23.1.10 Tile__set_type()

```
void Tile__set_type (
            Tile * this,
            TileType new_type,
            unsigned short new_rotation )
```

Definíció a(z) tile.c fájl 119. sorában.

4.24 tile.c

Ugrás a fájl dokumentációjához.

```
00001 #include <math.h>
00002 #include <SDL2/SDL.h>
00003
00004 #include "app.h"
00005 #include "game/tile.h"
00006
00007 #ifdef _CRCLONE_DEBUG
00008 #include "debug/debugmalloc.h"
00009 #endif
00010
00011 CardPile* CardPile__construct(void)
00012 {
00013     CardPile* new_cardpile = malloc(sizeof(CardPile));
00014     if(new_cardpile == NULL) return NULL;
00015
00016     new_cardpile->card = EMPTY;
00017     new_cardpile->next = NULL;
00018
00019     return new_cardpile;
00020 }
00021 void CardPile__destroy(CardPile* this)
00022 {
00023     if(this == NULL) return;
00024
00025     CardPile__destroy(this->next);
00026     free(this);
00027 }
00028 CardPile* CardPile__pop(CardPile* this, TileType* popped)
00029 {
00030     if(this == NULL) {
00031         *popped = EMPTY;
00032         return NULL;
00033     }
00034
00035     *popped = this->card;
00036     CardPile* new_top = this->next;
00037     free(this);
00038
00039     return new_top;
00040 }
00041 CardPile* CardPile__push(CardPile* this, TileType new_type)
00042 {
00043     CardPile* new_top = CardPile__construct();
00044     if(new_top == NULL) return NULL;
00045
00046     new_top->card = new_type;
00047     new_top->next = this;
```

```
00048
00049     return new_top;
00050 }
00051
00052
00053 void Tile__construct(Tile* this, TileType type, SDL_Point board_coords, SDL_FPoint offset)
00054 {
00055     if(this == NULL) return;
00056
00057     this->board_coords = board_coords;
00058     this->local_coords = (SDL_FPoint){board_coords.x * TILE_SIZE, board_coords.y * TILE_SIZE};
00059     this->global_coords = (SDL_FPoint){this->local_coords.x + offset.x, this->local_coords.y +
      offset.y};
00060     this->is_scored = false;
00061     this->is_expired = false;
00062
00063     this->rotatable = true;
00064     this->rotation = 0;
00065     Tile__set_type(this, type, this->rotation);
00066 }
00067
00068 bool Tile__point_in_tile(Tile* this, SDL_FPoint pt)
00069 {
00070     if(this == NULL) return false;
00071
00072     // ?  Lehetne kis módosítással `SDL_PointInRect`-et is használni.
00073     return this->global_coords.x < pt.x && pt.x <= this->global_coords.x + TILE_SIZE
00074         && this->global_coords.y < pt.y && pt.y <= this->global_coords.y + TILE_SIZE;
00075 }
00076
00077 void Tile__move_by(Tile* this, float mvx, float mvy)
00078 {
00079     if(this == NULL) return;
00080
00081     this->local_coords = (SDL_FPoint){this->local_coords.x + mvx * TILE_SIZE, this->local_coords.y +
      mvy * TILE_SIZE};
00082     this->global_coords = (SDL_FPoint){this->global_coords.x + mvx * TILE_SIZE, this->global_coords.y
      + mvy * TILE_SIZE};
00083
00084     //if(this->type == CASTLE_CAP_WALL_ROAD_BY) DBG_LOG("(%f, %d)", this->local_coords.x + mvy *
      TILE_SIZE, (int)(this->local_coords.x + mvy * TILE_SIZE));
00085 }
00086
00087 void Tile__rotate(Tile* this)
00088 {
00089     if(this == NULL) return;
00090
00091     Tile__set_rotation(this, this->rotation + 90);
00092 }
00093
00094 void Tile__set_rotation(Tile* this, unsigned short new_rotation)
00095 {
00096     if(this == NULL || !this->rotatable) return;
00097
00098     ConnectionType prev_connections[4];
00099
00100     // TODO: no
00101     new_rotation %= 360;
00102     unsigned int r = this->rotation;
00103     while(r != new_rotation) {
00104         r += 90;
00105         r %= 360;
00106
00107         for(unsigned int dir = NORTH; dir <= WEST; ++dir) {
00108             prev_connections[dir] = this->connections[dir];
00109         }
00110         for(int dir = NORTH; dir <= WEST; ++dir) {
00111             this->connections[dir] = (dir - 1) < 0 ?  prev_connections[3] :  prev_connections[dir -
      1];
00112         }
00113
00114     }
00115
00116     this->rotation = new_rotation;
00117 }
00118
00119 void Tile__set_type(Tile* this, TileType new_type, unsigned short new_rotation)
00120 {
00121     if(this == NULL) return;
00122
00123     this->type = new_type;
00124     // ?  Esetleg egy config fájlt létre lehetne hozni ennek.
00125     switch(new_type) {
00126         case FIELD_CLOISTER_ROAD_S:
00127             this->connections[NORTH] = FIELD;
00128             this->connections[EAST] = FIELD;
00129             this->connections[SOUTH] = ROAD;
```

```
00130             this->connections[WEST] = FIELD;
00131             break;
00132         case FIELD_CLOISTER_ROAD_NS:
00133             this->connections[NORTH] = ROAD;
00134             this->connections[EAST] = FIELD;
00135             this->connections[SOUTH] = ROAD;
00136             this->connections[WEST] = FIELD;
00137             break;
00138         case FIELD_VILLAGE_ROAD_S:
00139             this->connections[NORTH] = FIELD;
00140             this->connections[EAST] = FIELD;
00141             this->connections[SOUTH] = ROAD;
00142             this->connections[WEST] = FIELD;
00143             break;
00144         case FIELD_VILLAGE_ROAD_NS:
00145             this->connections[NORTH] = ROAD;
00146             this->connections[EAST] = FIELD;
00147             this->connections[SOUTH] = ROAD;
00148             this->connections[WEST] = FIELD;
00149             break;
00150         case ROAD_NS:
00151             this->connections[NORTH] = ROAD;
00152             this->connections[EAST] = FIELD;
00153             this->connections[SOUTH] = ROAD;
00154             this->connections[WEST] = FIELD;
00155             break;
00156         case ROAD_NW:
00157             this->connections[NORTH] = ROAD;
00158             this->connections[EAST] = FIELD;
00159             this->connections[SOUTH] = FIELD;
00160             this->connections[WEST] = ROAD;
00161             break;
00162         case ROAD_NWE:
00163             this->connections[NORTH] = ROAD;
00164             this->connections[EAST] = ROAD;
00165             this->connections[SOUTH] = FIELD;
00166             this->connections[WEST] = ROAD;
00167             break;
00168         case ROAD_NSWE:
00169             this->connections[NORTH] = ROAD;
00170             this->connections[EAST] = ROAD;
00171             this->connections[SOUTH] = ROAD;
00172             this->connections[WEST] = ROAD;
00173             this->rotatable = false;
00174             break;
00175         case CASTLE_PANTHEON:
00176         case CASTLE_TOWN:
00177             this->connections[NORTH] = CASTLE;
00178             this->connections[EAST] = CASTLE;
00179             this->connections[SOUTH] = CASTLE;
00180             this->connections[WEST] = CASTLE;
00181             this->rotatable = false;
00182             break;
00183         case CASTLE_TUNNEL:
00184             this->connections[NORTH] = CASTLE;
00185             this->connections[EAST] = FIELD;
00186             this->connections[SOUTH] = CASTLE;
00187             this->connections[WEST] = FIELD;
00188             break;
00189         case CASTLE_CORNER_WALL:
00190             this->connections[NORTH] = CASTLE;
00191             this->connections[EAST] = FIELD;
00192             this->connections[SOUTH] = FIELD;
00193             this->connections[WEST] = CASTLE;
00194             break;
00195         case CASTLE_CORNER_WALL_ROAD_BY:
00196             this->connections[NORTH] = CASTLE;
00197             this->connections[EAST] = ROAD;
00198             this->connections[SOUTH] = ROAD;
00199             this->connections[WEST] = CASTLE;
00200             break;
00201         case CASTLE_CAP_WALL:
00202             this->connections[NORTH] = CASTLE;
00203             this->connections[EAST] = FIELD;
00204             this->connections[SOUTH] = FIELD;
00205             this->connections[WEST] = FIELD;
00206             break;
00207         case CASTLE_CAP_WALL_ROAD_TO:
00208             this->connections[NORTH] = CASTLE;
00209             this->connections[EAST] = FIELD;
00210             this->connections[SOUTH] = ROAD;
00211             this->connections[WEST] = FIELD;
00212             break;
00213         case CASTLE_CAP_WALL_ROAD_BY:
00214             this->connections[NORTH] = CASTLE;
00215             this->connections[EAST] = ROAD;
00216             this->connections[SOUTH] = FIELD;
```

```
00217             this->connections[WEST] = ROAD;
00218             break;
00219         case CASTLE_SHIRT_WALL:
00220             this->connections[NORTH] = CASTLE;
00221             this->connections[EAST] = CASTLE;
00222             this->connections[SOUTH] = FIELD;
00223             this->connections[WEST] = CASTLE;
00224             break;
00225         case CASTLE_SHIRT_WALL_ROAD_TO:
00226             this->connections[NORTH] = CASTLE;
00227             this->connections[EAST] = CASTLE;
00228             this->connections[SOUTH] = ROAD;
00229             this->connections[WEST] = CASTLE;
00230             break;
00231         case EMPTY:
00232         default:
00233             this->connections[NORTH] = NONE;
00234             this->connections[EAST] = NONE;
00235             this->connections[SOUTH] = NONE;
00236             this->connections[WEST] = NONE;
00237             break;
00238     }
00239
00240     Tile__set_rotation(this, new_rotation);
00241 }
```

4.25 src/ui/screens.c fájlreferencia

```
#include <SDL2/SDL_surface.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "utils.h"
#include "game/player.h"
#include "game/tile.h"
#include "ui.h"
#include "app.h"
```

Függvények

- void Carcassone__Menu__construct (Carcassone *this)

  Létrehozza a menünézetet.
- void Carcassone__Menu__destroy (Carcassone *this)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `MenuScreen` által lefoglalt memóriát.
- void Carcassone__Menu__render (Carcassone *this)

  Menünézet megjelenítése.
- void Carcassone__Lboard__construct (Carcassone *this)

  Létrehozza a dicsőséglistanézetet.
- void Carcassone__Lboard__destroy (Carcassone *this)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `LeaderboardScreen` által lefoglalt memóriát.
- void Carcassone__Lboard__init_list_texture (Carcassone *this)

  Létrehozza a dicsőséglistanézethez a rekordokat.
- void Carcassone__Lboard__render (Carcassone *this)

  Dicsőséglistanézet megjelenítése.
- void Carcassone__Game__construct (Carcassone *this)

  Létrehozza a játéknézetet.

- void Carcassone__Game__destroy (Carcassone *this)

  Felszabadítja a megadott `Carcassone` struktúrához tartozó `GameScreen` által lefoglalt memóriát.
- void Carcassone__Game__render (Carcassone *this)

  Játéknézet megjelenítése.

4.25.1 Függvények dokumentációja

4.25.1.1 Carcassone__Game__construct()

```
void Carcassone__Game__construct (
            Carcassone * this )
```

Létrehozza a játéknézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Game__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a játéknézetét.

Definíció a(z) screens.c fájl 217. sorában.

4.25.1.2 Carcassone__Game__destroy()

```
void Carcassone__Game__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `GameScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 310. sorában.

4.25.1.3 Carcassone__Game__render()

```
void Carcassone__Game__render (
            Carcassone * this )
```

Játéknézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a nézet tartozik.

Definíció a(z) screens.c fájl 350. sorában.

4.25.1.4 Carcassone__Lboard__construct()

```
void Carcassone__Lboard__construct (
            Carcassone * this )
```

Létrehozza a dicsőséglistanézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Lboard__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a dicsőséglistanézetét.

Definíció a(z) screens.c fájl 96. sorában.

4.25.1.5 Carcassone__Lboard__destroy()

```
void Carcassone__Lboard__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `LeaderboardScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 116. sorában.

4.25.1.6 Carcassone__Lboard__init_list_texture()

```
void Carcassone__Lboard__init_list_texture (
            Carcassone * this )
```

Létrehozza a dicsőséglistanézethez a rekordokat.

Akkor kell meghívni, ha frissül a rekordfájl.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a dicsőséglistanézetét.

Definíció a(z) screens.c fájl 131. sorában.

4.25.1.7 Carcassone__Lboard__render()

```
void Carcassone__Lboard__render (
            Carcassone * this )
```

Dicsőséglistanézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a dicsőséglista tartozik.

Definíció a(z) screens.c fájl 196. sorában.

4.25.1.8 Carcassone__Menu__construct()

```
void Carcassone__Menu__construct (
            Carcassone * this )
```

Létrehozza a menünézetet.

Megjegyzés: A lefoglalt memória megfelelő felszabadításához meg kell hívni a `Carcassone__Menu__destroy` függvényt.

Paraméterek

this	A `Carcassone` struktúra, amelynek létrehozza a menünézetét.

Definíció a(z) screens.c fájl 26. sorában.

4.25.1.9 Carcassone__Menu__destroy()

```
void Carcassone__Menu__destroy (
            Carcassone * this )
```

Felszabadítja a megadott `Carcassone` struktúrához tartozó `MenuScreen` által lefoglalt memóriát.

Paraméterek

this	A `Carcassone` struktúra, aminek a lefoglalt memóriáját fel kell szabadítani.

Definíció a(z) screens.c fájl 59. sorában.

4.25.1.10 Carcassone__Menu__render()

```
void Carcassone__Menu__render (
            Carcassone * this )
```

Menünézet megjelenítése.

Paraméterek

this	A `Carcassone` struktúra, amihez a menü tartozik.

Definíció a(z) screens.c fájl 72. sorában.

4.26 screens.c

Ugrás a fájl dokumentációjához.

```
00001 #include <SDL2/SDL_surface.h>
00002 #include <stdio.h>
00003 #include <stdlib.h>
00004 #include <string.h>
00005 #include <time.h>
00006 #include <SDL2/SDL.h>
00007 #include <SDL2/SDL_ttf.h>
00008
00009 #include "utils.h"
00010 #include "game/player.h"
00011 #include "game/tile.h"
00012 #include "ui.h"
00013 #include "app.h"
00014
00015 #ifdef _CRCLONE_DEBUG
00016 #include "debug/debugmalloc.h"
00017 #endif
00018
00026 void Carcassone__Menu__construct(Carcassone* this)
00027 {
00028     this->menu_screen = malloc(sizeof(MenuScreen));
00029     this->menu_screen->background = create_SDL_texture_from_BMP(this->renderer, "./res/menu_bg.bmp");
00030
00031     // A háttérkép transzparenciája miatt.
00032     SDL_SetRenderDrawBlendMode(this->renderer, SDL_BLENDMODE_BLEND);
00033
00034     this->menu_screen->button_container = (SDL_Rect){this->width/2 - 400, 250, 800, 600};
00035
00036     // Gombok létrehozása
00037     SDL_Rect start_button_rect = {
00038         this->menu_screen->button_container.x + this->menu_screen->button_container.w / 2 - 100,
00039         this->menu_screen->button_container.y + 100,
00040         200, 60
00041     };
00042     this->menu_screen->start_button =
00043         Carcassone__Button__construct(this, this->default_font, "START", start_button_rect,
      (SDL_Color){COLOR_BLUE}, (SDL_Color){COLOR_WHITE}, true);
00044
00045     SDL_Rect lboard_button_rect = {
00046         this->menu_screen->button_container.x + this->menu_screen->button_container.w / 2 - 200,
```

```
00047         this->menu_screen->button_container.y + 200,
00048         400, 60
00049     };
00050     this->menu_screen->lboard_button =
00051         Carcassone__Button__construct(this, this->default_font, "DICSŐSÉGLISTA", lboard_button_rect,
      (SDL_Color){COLOR_BLUE}, (SDL_Color){COLOR_WHITE}, true);
00052 }
00053
00059 void Carcassone__Menu__destroy(Carcassone* this)
00060 {
00061     destroy_SDL_Texture(this->menu_screen->background);
00062     Carcassone__Button__destroy(this, &this->menu_screen->start_button);
00063     Carcassone__Button__destroy(this, &this->menu_screen->lboard_button);
00064     free(this->menu_screen);
00065 }
00066
00072 void Carcassone__Menu__render(Carcassone* this)
00073 {
00074     SDL_RenderClear(this->renderer);
00075
00076     // Háttér.
00077     SDL_RenderCopy(this->renderer, this->menu_screen->background, NULL, NULL);
00078     SDL_SetRenderDrawColor(this->renderer, 235, 235, 225, 100);
00079     SDL_RenderFillRect(this->renderer, NULL);
00080
00081     Carcassone__Button__render(this, &this->menu_screen->start_button);
00082     Carcassone__Button__render(this, &this->menu_screen->lboard_button);
00083
00084     // TODO: a splash_title minden nézetben ugyanott van
00085     SDL_RenderCopy(this->renderer, this->splash_title, NULL,
00086         &(SDL_Rect){this->width/2 - 400, 0, 800, 240});
00087 }
00088
00096 void Carcassone__Lboard__construct(Carcassone* this)
00097 {
00098     this->lboard_screen = malloc(sizeof(LeaderboardScreen));
00099     this->lboard_screen->leaderboard = Leaderboard__construct("res/data/records.dat");
00100     this->lboard_screen->list_texture = NULL;
00101     strcpy(this->lboard_screen->syntax_error_msg, "Hibás fájlformátum!");
00102
00103     // "Vissza" gomb.
00104     this->lboard_screen->back_button =
00105         Carcassone__Button__construct(this, this->small_font, "VISSZA", (SDL_Rect){this->width - 150,
      10, 140, 50},
00106         (SDL_Color){COLOR_WHITE}, (SDL_Color){0, 0, 0, 255}, true);
00107
00108     Carcassone__Lboard__init_list_texture(this);
00109 }
00110
00116 void Carcassone__Lboard__destroy(Carcassone* this)
00117 {
00118     Leaderboard__destroy(this->lboard_screen->leaderboard);
00119     destroy_SDL_Texture(this->lboard_screen->list_texture);
00120     Carcassone__Button__destroy(this, &this->lboard_screen->back_button);
00121     free(this->lboard_screen);
00122 }
00123
00131 void Carcassone__Lboard__init_list_texture(Carcassone* this)
00132 {
00133     // A rekordok.
00134     destroy_SDL_Texture(this->lboard_screen->list_texture);
00135     this->lboard_screen->list_texture = SDL_CreateTexture(this->renderer, SDL_PIXELFORMAT_RGBA8888,
      SDL_TEXTUREACCESS_TARGET,
00136         1000, this->height - 200);
00137     SDL_SetRenderTarget(this->renderer, this->lboard_screen->list_texture);
00138
00139     if(this->lboard_screen->leaderboard != NULL) {
00140         Leaderboard__sort(this->lboard_screen->leaderboard);
00141
00142         char score_string[10+1]; // safe bet
00143         int w, h;
00144         // A top 5 rekord megjelenítése.
00145         for(size_t place = 0U; place < MIN(5, this->lboard_screen->leaderboard->entries_size);
      ++place) {
00146             // A rekordhoz tartozó játékos neve.
00147             char* curr_name = this->lboard_screen->leaderboard->entries[place].name;
00148             TTF_SizeUTF8(this->small_font, curr_name, &w, &h);
00149             SDL_Surface* name_surface = TTF_RenderUTF8_Blended(this->small_font,
      this->lboard_screen->leaderboard->entries[place].name,
00150                 (SDL_Color){COLOR_WHITE});
00151             if(name_surface != NULL) {
00152                 SDL_Texture* name = SDL_CreateTextureFromSurface(this->renderer, name_surface);
00153                 SDL_RenderCopy(this->renderer, name,
00154                     NULL, &(SDL_Rect){0, 160 * place, w, h});
00155                 SDL_FreeSurface(name_surface);
00156                 destroy_SDL_Texture(name);
00157             }
```

```
00158
00159             // A rekord.
00160             sprintf(score_string, "%u", this->lboard_screen->leaderboard->entries[place].highscore);
00161             TTF_SizeUTF8(this->small_font, score_string, &w, &h);
00162             SDL_Surface* score_surface = TTF_RenderUTF8_Blended(this->default_font, score_string,
00163                 (SDL_Color){COLOR_WHITE});
00164             if(score_surface != NULL) {
00165                 SDL_Texture* score = SDL_CreateTextureFromSurface(this->renderer, score_surface);
00166                 SDL_RenderCopy(this->renderer, score,
00167                     NULL, &(SDL_Rect){0, 160 * place + h, w, h});
00168                 SDL_FreeSurface(score_surface);
00169                 destroy_SDL_Texture(score);
00170             }
00171         }
00172     } else {
00173         // Hibás formátum esetén.
00174         int w, h;
00175         TTF_SizeUTF8(this->small_font, this->lboard_screen->syntax_error_msg, &w, &h);
00176         SDL_Surface* msg_surface = TTF_RenderUTF8_Blended(this->default_font,
    this->lboard_screen->syntax_error_msg,
00177             (SDL_Color){COLOR_WHITE});
00178         if(msg_surface != NULL) {
00179             SDL_Texture* error_msg = SDL_CreateTextureFromSurface(this->renderer, msg_surface);
00180             SDL_RenderCopy(this->renderer, error_msg, NULL, &(SDL_Rect){100, 100, w, h});
00181             SDL_FreeSurface(msg_surface);
00182             destroy_SDL_Texture(error_msg);
00183         }
00184     }
00185
00186     SDL_SetRenderTarget(this->renderer, NULL);
00187     // Nem néz ki sajnos a legjobban
00188     SDL_SetTextureBlendMode(this->lboard_screen->list_texture, SDL_BLENDMODE_BLEND);
00189 }
00190
00196 void Carcassone__Lboard__render(Carcassone* this)
00197 {
00198     SDL_SetRenderDrawColor(this->renderer, COLOR_BG);
00199     SDL_RenderClear(this->renderer);
00200
00201     Carcassone__Button__render(this, &this->lboard_screen->back_button);
00202
00203     SDL_RenderCopy(this->renderer, this->splash_title, NULL,
00204         &(SDL_Rect){this->width/2 - 400, 0, 800, 240});
00205
00206     SDL_RenderCopy(this->renderer, this->lboard_screen->list_texture, NULL,
00207         &(SDL_Rect){this->width/2 - 500, 250, 1000, 700});
00208 }
00209
00217 void Carcassone__Game__construct(Carcassone* this)
00218 {
00219     this->game_screen = malloc(sizeof(GameScreen));
00220     this->game_screen->board_offset = (SDL_FPoint){this->width/2.0f-300, 120};
00221     this->game_screen->board = NULL;
00222     this->game_screen->active_input = NULL;
00223     this->game_screen->curr_player = NULL;
00224     this->game_screen->winner = NULL;
00225     this->game_screen->pile_index = 0U;
00226     this->game_screen->is_ready = false;
00227     this->game_screen->is_game_over = false;
00228     this->game_screen->drawn_tile = malloc(sizeof(Tile));
00229     this->game_screen->card_pile = CardPile__construct();
00230     for(size_t k = 0U; k < 4; ++k) {
00231         this->game_screen->held_arrow_keys[k] = 0;
00232     }
00233     for(size_t p = 0U; p < 2; ++p) {
00234         this->game_screen->players[p] = Player__construct(NULL, NULL, "", "");
00235     }
00236
00237     // TODO: ezt ne így
00238     char counter_string[2] = "0";
00239     SDL_Surface* curr_index_surface = TTF_RenderText_Blended(this->default_font, counter_string,
    (SDL_Color){COLOR_WHITE});
00240     for(size_t ti = 0U; ti < PILE_SIZE; ++ti) {
00241         sprintf(counter_string, "%zu", PILE_SIZE - ti);
00242         curr_index_surface = TTF_RenderText_Blended(this->default_font, counter_string,
    (SDL_Color){COLOR_WHITE});
00243         if(curr_index_surface != NULL) {
00244             this->game_screen->pile_counter[ti] = SDL_CreateTextureFromSurface(this->renderer,
    curr_index_surface);
00245             SDL_FreeSurface(curr_index_surface);
00246         }
00247
00248         if(this->game_screen->pile_counter[ti] == NULL)
00249             break;
00250     }
00251
00252     this->game_screen->board_texture = SDL_CreateTexture(this->renderer, SDL_PIXELFORMAT_RGBA8888,
```

```
00253         SDL_TEXTUREACCESS_TARGET, 600, this->height - this->game_screen->board_offset.y - 10);
00254     this->game_screen->tileset_wrapper = TilesetWrapper__construct(this->renderer);
00255     if(this->game_screen->tileset_wrapper.tile_set == NULL) {
00256         SDL_LogError(SDL_LOG_CATEGORY_RENDER, "Nem lehetett betölteni az atlaszt!");
00257         SDL_LogError(SDL_LOG_CATEGORY_RENDER, "%s", SDL_GetError());
00258     }
00259     Carcassone__Game__init_board(this);
00260     Carcassone__Game__init_pile(this);
00261     Carcassone__Game__draw_new(this);
00262 
00263     int max_width, height;
00264     TTF_SizeUTF8(this->small_font, "WWWWWWWWWWWWWWWWWWWWWWWW", &max_width, &height);
00265 
00266     SDL_Rect input1_max_rect = {
00267         this->width / 2 - max_width / 2, this->height / 4 + 150, max_width, height + 10
00268     };
00269     SDL_Rect input2_max_rect = {
00270         this->width / 2 - max_width / 2, this->height / 4 - 100, max_width, height + 10
00271     };
00272 
00273     SDL_Surface* plabel1_surface = TTF_RenderUTF8_Blended(this->small_font, "Játékos 1",
    (SDL_Color){COLOR_LIGHTBLUE});
00274     if(plabel1_surface != NULL) {
00275         this->game_screen->player_input_labels[0] =
00276             SDL_CreateTextureFromSurface(this->renderer, plabel1_surface);
00277         SDL_FreeSurface(plabel1_surface);
00278     }
00279     SDL_Surface* plabel2_surface = TTF_RenderUTF8_Blended(this->small_font, "Játékos 2",
    (SDL_Color){COLOR_SALMON});
00280     if(plabel1_surface != NULL) {
00281         this->game_screen->player_input_labels[1] =
00282             SDL_CreateTextureFromSurface(this->renderer, plabel2_surface);
00283         SDL_FreeSurface(plabel2_surface);
00284     }
00285 
00286     this->game_screen->player_name_inputs[0] =
00287         Carcassone__Prompt__construct(this, this->small_font, "", input1_max_rect,
    (SDL_Color){COLOR_WHITE}, (SDL_Color){0, 0, 0, 255});
00288     this->game_screen->player_name_inputs[1] =
00289         Carcassone__Prompt__construct(this, this->small_font, "", input2_max_rect,
    (SDL_Color){COLOR_WHITE}, (SDL_Color){0, 0, 0, 255});
00290     this->game_screen->ready_button =
00291         Carcassone__Button__construct(this, this->small_font, "OK", (SDL_Rect){10, this->height - 10 -
    60, 150, 60}, (SDL_Color){COLOR_WHITE},
00292             (SDL_Color){0, 0, 0, 255}, true);
00293     this->game_screen->concede_button =
00294         Carcassone__Button__construct(this, this->small_font, "FELAD", (SDL_Rect){this->width - 10 -
    150, 10, 150, 60}, (SDL_Color){COLOR_WHITE},
00295             (SDL_Color){140, 0, 0, 255}, true);
00296     this->game_screen->end_turn_button =
00297         Carcassone__Button__construct(this, this->small_font, "KÖR VÉGE", (SDL_Rect){10, 10, 200, 60},
    (SDL_Color){COLOR_WHITE},
00298             (SDL_Color){0, 0, 140, 255}, true);
00299 
00300     this->game_screen->active_input = &this->game_screen->player_name_inputs[1];
00301 
00302     this->game_screen->crown_texture = create_SDL_texture_from_BMP(this->renderer,
    "./res/winners_crown.bmp");
00303 }
00304 
00310 void Carcassone__Game__destroy(Carcassone* this)
00311 {
00312     destroy_SDL_Texture(this->game_screen->board_texture);
00313     for(size_t n = 0U; n < PILE_SIZE; ++n) {
00314         destroy_SDL_Texture(this->game_screen->pile_counter[n]);
00315     }
00316 
00317     for(size_t p = 0U; p < 2; ++p) {
00318         Player__destroy(&this->game_screen->players[p]);
00319     }
00320 
00321     if(this->game_screen->board != NULL) {
00322         for(size_t n = 0U; n < BOARD_SIZE; ++n) {
00323             free(this->game_screen->board[n]);
00324         }
00325     }
00326     free(this->game_screen->board);
00327     CardPile__destroy(this->game_screen->card_pile);
00328     free(this->game_screen->drawn_tile);
00329 
00330     Carcassone__Button__destroy(this, &this->game_screen->ready_button);
00331     Carcassone__Button__destroy(this, &this->game_screen->concede_button);
00332     Carcassone__Button__destroy(this, &this->game_screen->end_turn_button);
00333     Carcassone__Prompt__destroy(this, &this->game_screen->player_name_inputs[0]);
00334     Carcassone__Prompt__destroy(this, &this->game_screen->player_name_inputs[1]);
00335     for(size_t i = 0U; i < 2; ++i) {
00336         destroy_SDL_Texture(this->game_screen->player_input_labels[i]);
```

```
00337     }
00338
00339     destroy_SDL_Texture(this->game_screen->crown_texture);
00340
00341     TilesetWrapper__destroy(&this->game_screen->tileset_wrapper);
00342     free(this->game_screen);
00343 }
00344
00350 void Carcassone__Game__render(Carcassone* this)
00351 {
00352     SDL_SetRenderDrawColor(this->renderer, COLOR_BG);
00353     SDL_RenderClear(this->renderer);
00354
00355     if(!this->game_screen->is_ready) {
00356         // Carcassone__Prompt__render(this, &this->game_screen->player_name_inputs[0]);
00357         // Carcassone__Prompt__render(this, &this->game_screen->player_name_inputs[1]);
00358         Carcassone__Button__render(this, &this->game_screen->ready_button);
00359
00360         SDL_RenderCopy(this->renderer, this->game_screen->player_input_labels[0], NULL, &(SDL_Rect){
00361             this->width / 2 - 75, this->height / 4 - 170, 130, 50
00362         });
00363         SDL_RenderCopy(this->renderer, this->game_screen->player_input_labels[1], NULL, &(SDL_Rect){
00364             this->width / 2 - 75, this->height / 4 + 80, 130, 50
00365         });
00366
00367         if(this->game_screen->active_input != NULL) {
00368             SDL_SetRenderDrawColor(this->renderer, COLOR_RED);
00369             SDL_RenderDrawRect(this->renderer, &this->game_screen->active_input->prompt.global_rect);
00370         }
00371     } else {
00372         Carcassone__Game__render_board(this);
00373         Carcassone__Game__render_meeples(this);
00374         if(!this->game_screen->is_game_over) {
00375             Carcassone__indicate_possible_placements(this);
00376             Carcassone__Game__render_drawn_tile(this);
00377         }
00378         Carcassone__render_splash_title(this);
00379         Carcassone__Game__render_player_stats(this);
00380         Carcassone__Button__render(this, &this->game_screen->concede_button);
00381         Carcassone__Button__render(this, &this->game_screen->end_turn_button);
00382         if(this->game_screen->is_game_over) {
00383             Carcassone__Game__render_game_over(this);
00384         }
00385     }
00386 }
```

4.27 src/ui/ui.c fájlreferencia

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include "ui.h"
#include "app.h"
#include "utils.h"
```

Függvények

- Button Carcassone__Button__construct (Carcassone *this, TTF_Font *font, char *label, SDL_Rect global↩ _rect, SDL_Color bg_color, SDL_Color text_color, bool center)
- void Carcassone__Button__render (Carcassone *this, Button *button)
- void Carcassone__Button__destroy (Carcassone *this, Button *button)
- Prompt Carcassone__Prompt__construct (Carcassone *this, TTF_Font *font, char *default_label, SDL_Rect global_rect, SDL_Color bg_color, SDL_Color text_color)
- void Carcassone__Prompt__edit (Carcassone *this, Prompt *prompt, char *new_label, bool concat)
- void Carcassone__Prompt__render (Carcassone *this, Prompt *prompt)
- void Carcassone__Prompt__destroy (Carcassone *this, Prompt *prompt)

4.27.1 Függvények dokumentációja

4.27.1.1 Carcassone__Button__construct()

```
Button Carcassone__Button__construct (
            Carcassone * this,
            TTF_Font * font,
            char * label,
            SDL_Rect global_rect,
            SDL_Color bg_color,
            SDL_Color text_color,
            bool center )
```

Definíció a(z) ui.c fájl 16. sorában.

4.27.1.2 Carcassone__Button__destroy()

```
void Carcassone__Button__destroy (
            Carcassone * this,
            Button * button )
```

Definíció a(z) ui.c fájl 51. sorában.

4.27.1.3 Carcassone__Button__render()

```
void Carcassone__Button__render (
            Carcassone * this,
            Button * button )
```

Definíció a(z) ui.c fájl 40. sorában.

4.27.1.4 Carcassone__Prompt__construct()

```
Prompt Carcassone__Prompt__construct (
            Carcassone * this,
            TTF_Font * font,
            char * default_label,
            SDL_Rect global_rect,
            SDL_Color bg_color,
            SDL_Color text_color )
```

Definíció a(z) ui.c fájl 57. sorában.

4.27.1.5 Carcassone__Prompt__destroy()

```
void Carcassone__Prompt__destroy (
            Carcassone * this,
            Prompt * prompt )
```

Definíció a(z) ui.c fájl 96. sorában.

4.27.1.6 Carcassone__Prompt__edit()

```
void Carcassone__Prompt__edit (
            Carcassone * this,
            Prompt * prompt,
            char * new_label,
            bool concat )
```

Definíció a(z) ui.c fájl 66. sorában.

4.27.1.7 Carcassone__Prompt__render()

```
void Carcassone__Prompt__render (
            Carcassone * this,
            Prompt * prompt )
```

Definíció a(z) ui.c fájl 91. sorában.

4.28 ui.c

Ugrás a fájl dokumentációjához.

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <string.h>
00004 #include <time.h>
00005 #include <SDL2/SDL.h>
00006 #include <SDL2/SDL_ttf.h>
00007
00008 #include "ui.h"
00009 #include "app.h"
00010 #include "utils.h"
00011
00012 #ifdef _CRCLONE_DEBUG
00013 #include "debug/debugmalloc.h"
00014 #endif
00015
00016 Button Carcassone__Button__construct(Carcassone* this, TTF_Font* font, char* label, SDL_Rect
      global_rect, SDL_Color bg_color, SDL_Color text_color, bool center)
00017 {
00018     Button new_button = {.label = strdup_(label, true), .bg_color = bg_color, .global_rect =
      global_rect, .used_font = font};
00019     DBG_LOG("LabeL: %s", new_button.label);
00020
00021     int rect_width, rect_height;
00022     TTF_SizeUTF8(font, label, &rect_width, &rect_height);
00023     new_button.label_rect = (SDL_Rect){
00024         0, global_rect.h/2 - rect_height/2,
00025         rect_width, rect_height
00026     };
```

```
00027     if(center) {
00028         new_button.label_rect.x = global_rect.w/2 - rect_width/2;
00029     }
00030
00031     SDL_Surface* label_surface = TTF_RenderUTF8_Blended(font, label, text_color);
00032     if(label_surface != NULL) {
00033         new_button.label_texture = SDL_CreateTextureFromSurface(this->renderer, label_surface);
00034         SDL_FreeSurface(label_surface);
00035     }
00036
00037     return new_button;
00038 }
00039
00040 void Carcassone__Button__render(Carcassone* this, Button* button)
00041 {
00042     SDL_RenderSetViewport(this->renderer, &button->global_rect);
00043
00044     SDL_SetRenderDrawColor(this->renderer, button->bg_color.r, button->bg_color.g, button->bg_color.b,
      button->bg_color.a);
00045     SDL_RenderFillRect(this->renderer, NULL);
00046     SDL_RenderCopy(this->renderer,button->label_texture, NULL, &button->label_rect);
00047
00048     SDL_RenderSetViewport(this->renderer, NULL);
00049 }
00050
00051 void Carcassone__Button__destroy(Carcassone* this, Button* button)
00052 {
00053     free(button->label);
00054     if(button->label_texture != NULL) SDL_DestroyTexture(button->label_texture);
00055 }
00056
00057 Prompt Carcassone__Prompt__construct(Carcassone* this, TTF_Font* font, char* default_label, SDL_Rect
      global_rect, SDL_Color bg_color, SDL_Color text_color)
00058 {
00059     Prompt new_prompt = {
00060         .prompt = Carcassone__Button__construct(this, font, default_label, global_rect, bg_color,
      text_color, false),
00061     };
00062
00063     return new_prompt;
00064 }
00065
00066 void Carcassone__Prompt__edit(Carcassone* this, Prompt* prompt, char* new_label, bool concat)
00067 {
00068     if(concat) {
00069         prompt->prompt.label = strcatdyn(prompt->prompt.label, new_label, true);
00070     } else {
00071         DBG_LOG("Before:  %s", prompt->prompt.label);
00072         free(prompt->prompt.label);
00073         prompt->prompt.label = strdup_(new_label, true);
00074         DBG_LOG("After:  %s", prompt->prompt.label);
00075     }
00076
00077     destroy_SDL_Texture(prompt->prompt.label_texture);
00078
00079     SDL_Surface* updated_surface = TTF_RenderUTF8_Blended(prompt->prompt.used_font,
      prompt->prompt.label, (SDL_Color){0, 0, 0, 255});
00080     if(updated_surface != NULL) {
00081         prompt->prompt.label_texture = SDL_CreateTextureFromSurface(this->renderer, updated_surface);
00082
00083         int new_width;
00084         TTF_SizeUTF8(prompt->prompt.used_font, prompt->prompt.label, &new_width, NULL);
00085         prompt->prompt.label_rect.w = new_width;
00086
00087         SDL_FreeSurface(updated_surface);
00088     }
00089 }
00090
00091 void Carcassone__Prompt__render(Carcassone* this, Prompt* prompt)
00092 {
00093     Carcassone__Button__render(this, &prompt->prompt);
00094 }
00095
00096 void Carcassone__Prompt__destroy(Carcassone* this, Prompt* prompt)
00097 {
00098     Carcassone__Button__destroy(this, &prompt->prompt);
00099 }
```

4.29 src/utils.c fájlreferencia

```
#include <stdlib.h>
#include <string.h>
```

```
#include <stdbool.h>
#include <wchar.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_surface.h>
#include "utils.h"
```

Függvények

- char * strdup_ (char const *str, bool is_utf8)
- char * strcatdyn (char *original, char const *to_cat, bool is_utf8)
- void remove_last_utf8_char_dyn (char *str)
- void destroy_SDL_Texture (SDL_Texture *texture)
- SDL_Texture * create_SDL_texture_from_BMP (SDL_Renderer *renderer, char const *source_path)

4.29.1 Függvények dokumentációja

4.29.1.1 create_SDL_texture_from_BMP()

```
SDL_Texture * create_SDL_texture_from_BMP (
            SDL_Renderer * renderer,
            char const * source_path )
```

Definíció a(z) utils.c fájl 56. sorában.

4.29.1.2 destroy_SDL_Texture()

```
void destroy_SDL_Texture (
            SDL_Texture * texture )
```

Definíció a(z) utils.c fájl 48. sorában.

4.29.1.3 remove_last_utf8_char_dyn()

```
void remove_last_utf8_char_dyn (
            char * str )
```

Definíció a(z) utils.c fájl 43. sorában.

4.29.1.4 strcatdyn()

```
char * strcatdyn (
            char * original,
            char const * to_cat,
            bool is_utf8 )
```

Definíció a(z) utils.c fájl 25. sorában.

4.29.1.5 strdup_()

```
char * strdup_ (
            char const * str,
            bool is_utf8 )
```

Definíció a(z) utils.c fájl 14. sorában.

4.30 utils.c

Ugrás a fájl dokumentációjához.

```
00001 #include <stdlib.h>
00002 #include <string.h>
00003 #include <stdbool.h>
00004 #include <wchar.h>
00005 #include <SDL2/SDL_render.h>
00006 #include <SDL2/SDL_surface.h>
00007
00008 #include "utils.h"
00009
00010 #ifdef _CRCLONE_DEBUG
00011 #include "debug/debugmalloc.h"
00012 #endif
00013
00014 char* strdup_(char const* str, bool is_utf8)
00015 {
00016     size_t len = strlen(str) * (is_utf8 ?  sizeof(wchar_t) :  sizeof(char));
00017     char* dyn_str = malloc(len + 1);
00018     if(dyn_str == NULL) return NULL;
00019
00020     strcpy(dyn_str, str);
00021
00022     return dyn_str;
00023 }
00024
00025 char* strcatdyn(char* original, char const* to_cat, bool is_utf8)
00026 {
00027     if(original == NULL) return NULL;
00028
00029     size_t len = (strlen(original) + strlen(to_cat)) * (is_utf8 ?  sizeof(wchar_t) :  sizeof(char));
00030     size_t cat_from = strlen(original);
00031
00032     char* new_str = malloc(len + 1);
00033     if(new_str == NULL) return NULL;
00034
00035     strcpy(new_str, original);
00036     strcpy(new_str + cat_from, to_cat);
00037     free(original);
00038     original = NULL;
00039
00040     return new_str;
00041 }
00042
00043 void remove_last_utf8_char_dyn(char* str)
00044 {
00045     //
00046 }
00047
```

```
00048 void destroy_SDL_Texture(SDL_Texture* texture)
00049 {
00050     if(texture == NULL) return;
00051
00052     SDL_DestroyTexture(texture);
00053     texture = NULL;
00054 }
00055
00056 SDL_Texture* create_SDL_texture_from_BMP(SDL_Renderer* renderer, char const* source_path)
00057 {
00058     SDL_Surface* src_surface = SDL_LoadBMP(source_path);
00059     if(src_surface == NULL) return NULL;
00060
00061     SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, src_surface);
00062     SDL_FreeSurface(src_surface);
00063
00064     return texture;
00065 }
```