Vízvezetékszerelő

Készítette Doxygen 1.10.0

1. fejezet

Vízvezetékszerelő/Plumber

A vízvezetékszerelő egy játék, amit a Programozás alapjai 1 tárgy nagy házijához készítettem.

GitHubon itt található meg: https://github.com/bribena/Plumber

1.1. Telepítés

Először klónozza a repositoryt:
```
git clone https://github.com/bribena/Plumber.git
```

A program az SDL, az SDL_image, az SDL_mixer, az SDL_ttf, az SDL_gfxPrimitives, az osdialog és a C moduljaira épül.

Linuxon a GCC, Windowson a MinGW segítségével fordítható a program.

1.1.1. SDL telepítése

A különböző SDL modulok forrásainak linkjei:

- SDL: https://github.com/libsdl-org/SDL/releases/tag/release-2.28.5
- SDL_image: https://github.com/libsdl-org/SDL_image/releases
- SDL_ttf: https://github.com/libsdl-org/SDL_ttf/releases
- SDL_mixer: https://github.com/libsdl-org/SDL_mixer/releases
- SDL_gfxPrimitives: https://www.ferzkopp.net/wordpress/2016/01/02/sdl_gfx-sdl2_gfx/

Az SDL telepítési lépései: https://wiki.libsdl.org/SDL2/Installation

1.1.1.1. Linux

Lehetőleg a disztribúció csomagkezelőjével szerezze be ezeket az SDL modulokat. Ha ott nem elérhető, akkor a megfelelő repository telepítési utasításait kövesse.

1.1.1.2. Windows

Windowshoz a megadott oldalakon találhatóak előre buildelt fájlok. Ezeket töltse le (mindegyiknél a MinGW-vel kompatibilis zipfájlt). A zipfájlokat az alábbi módon csomagolja ki:

- A dll fájlok (bin mappa tartalma, kivétel SDL_gfxPrimitives) kerüljenek a projekt gyökérmappájába
- A header fájlok (include mappa tartalma, kivétel SDL_gfxPrimitives) kerüljenek a projekt lib/include mappájába
- A lib mappa pedig mappástul a projekt lib mappájába

Az SDL_gfxPrimitives esetén a megfelelő fájlok (.dll, .h, .a/.o) megtalálhatóak a zipjében. Ezeket a fentiek szerint helyezze el a projekt megfelelő mappáiban.

1.1.2. osdialog telepítése

GitHub: `https://github.com/AndrewBelt/osdialog`

Mindkét rendszeren a repositoryt klónozza a projekt lib mappájába.

Linuxon az osdialog a GTK 2.0-t használja, ezt is szerezze be a disztribúciójának a csomagkezelőjével.

Megjegyzés: az osdialog a debugmalloc miatt hibát dobhat. Erre két javítás van:

1. Az osdialog.h-ban az include-ok alá írja be, hogy `#include "debugmalloc.h"`
2. A projekt fájlaiból ki kell venni az `#include "debugmalloc.h"` sorokat.

1.2. Fordítás

1.2.1. Linux

A `build.sh` futtatásával létrejön a `plumber` bináris fájl, amennyiben az SDL modulok rendszerszinten elérhetőek. Egyéb esetben manuálisan kell fordítani a programot a megfelelő helyek megadásával.

A `build.sh` gcc-vel fordítja le a kódot.

1.2.2. Windows

A `build.bat` futtatásával létrejön a `plumber.exe` bináris fájl, amennyiben megfelelően vannak elhelyezve a külső modulok fájlai. Egyéb esetben manuálisan kell fordítani a programot a megfelelő helyek megadásával.

A `build.bat` x86_64-w64-mingw32-gcc-vel fordítja a kódot.

1.3. A játék menete

A játék induláskor a főmenübe dobja a játékost. Innen 3 opció tárul elé:

- Játék: Létrehoz egy véletlenszerű pályát, amit ki lehet játszani.
- Pálya betöltése: Egy, a program által generált .plm fájlt tölt be. Néhány mintafájl található a projekt testfiles mappájában. Ezután elindítja a játékot.
- Kilépés

A pályán belül 1-1 csőre kattintva el lehet forgatni azt (bal kattintással az óra haladási irányával megegegyezően, jobb kattintással pedig ellentétesen). A játék célja, hogy a bemenettől (hosszú cső a bal oldalon) a kimenetig (hosszú cső a jobb oldalon) a pályán levő csövekkel a játékos elvezesse a vizet. A vizet elindítani csak helyes megoldás esetén lehet a bemeneten található csappal, bal kattintással. Helyes megoldásnak egy a bemenettől a kimenetig vezető csőpálya számít. Az egyes csövekből nincsen "kifolyás", azaz ha már egy irányba tudja tovább tölteni a vizet, akkor a megoldás szempontjából megfelelő.

Egy különleges elem van, az "egymáson átmenő", ami két különböző irányba menő egyenesnek felel meg.

A játék közben bármikor kimenthető a pálya *aktuális* állapota.

Győzelemkor a pálya *kezdésekor előforduló* állapota menthető ki. Azaz, ha a Pálya betöltésével lett elkezdve a játék, akkor a betöltött állapotot lehet kimenteni, játék esetén pedig a legenerált pályát.

A kimentett fájl nevében ne legyen szóköz, ékezet és speciális karakter.

Futás után visszaadott kódok: 0: Nem történt hiba 1: Nem sikerült az SDL inicializálása 2: Nem sikerült az ablakot létrehozni/a fájlokat betölteni 3: A játék futása során hiba történt

1.3.1. A generált fájlok struktúrája (.plm)

A fájl első sora 4 adatot tartalmaz, ebben a sorrendben:

- Az egy sorban található csövek számát
- Az egy oszlopban található csövek számát
- A pálya bemenetének sora
- A pálya kimenetének sora

Ezután 3x3-as blokkokban tartalmazza a pályán szereplő csöveket szöveges formátumban.

Bármilyen fájl, ami ennek a struktúrának nem felel meg, hibás fájlnak számít.

1.4. Hibák

A programmal kapcsolatos hibák a programozói dokumentációban megtalálhatóak (a projekt docs mappájában). Ezeken kívül a hibák nagy részét kiírja a konzolra a program, ha előfordulnak. Ekkor vagy összeomlik a program, vagy az adott funkció nélküll fut tovább.

Készítette: Bihari Bence

2. fejezet

Hiba lista

Fájl game.c

Az osdialog modullal problémája van a debugmalloc.h-nak. Az osdialog dokumentációja tisztán leírja, hogy a visszaadott stringeit fel kell szabadítani használat után. Viszont, ha fel akarnám szabadítani ezt, a debugmalloc.h hibát ad vissza. Ezt úgy lehet kijavítani, hogy az osdialog.h-ban include-oljuk a debugmalloc.h-t vagy innen kivesszük a debugmalloc.h-t.

Fájl game.h

A modul egy osdialog nevű modullal hoz létre rendszerfüggő dialógusablakokat. Ez a GTK2-re épül Linux alatt, ami memóriaszivárgást okoz. Windows alatt az Address Sanitizert nem teszteltem, ott nem tudom, hogy mi a helyzet. Egy lehetséges megoldás, hogy az osdialog-ot mással helyettesítjük, de emiatt a game.c-ben változtatásokat kell hozni.

Fájl SDL_local.h

Az SDL használata miatt memóriaszivárgások előfordulhatnak.

Ha a konzolra valamilyen libGL error kerül akkor textúra hibák léphetnek fel. Nem tudom, hogy mi okkozza ezt, de csak grafikus hibát okoz.

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

4. fejezet

Fájlok dokumentációja

4.1. lib/include/game.h fájlreferencia

A játék headerje.

```
#include "SDL_local.h"
#include "pipe.h"
#include "osdialog.h"
```

Adatszerkezetek

- struct GameStruct

  A játékhoz szükséges struktúrákat fogja össze. Részletek...
- struct EventloopVariables

  A játék eventloopjának a változóit fogja össze. Részletek...
- struct Wheel

  A csap struktúrája. Részletek...
- struct AnimationQueue

  Az animációs sor struktúrája. Részletek...

Makródefiníciók

- #define DEFAULT_WINDOW_WIDTH 1600

  Alapértelmezett ablakszélesség.
- #define DEFAULT_WINDOW_HEIGHT 900

  Alapértelmezett ablakmagasság.
- #define MIN_X_SIZE 18

  A játék pályáján szereplő, egy sorban lévő csövek minimális száma.
- #define MIN_Y_SIZE 14

  A játék pályáján szereplő, egy oszlopban lévő csövek minimális száma.
- #define WHEEL_SIZE_MULTIPLIER 1.75

  A csap csövekhez viszonyított mérete.
- #define WHEEL_ROTATE_SPEED 2

  A csap forgatási animációjának a sebessége.
- #define PIPE_ROTATE_SPEED 3

  A csövek forgatási animációjának a sebessége.
- #define LONG_PIPE_FILL_SPEED 5

  A pálya előtt lévő hosszú cső feltöltési animációjának a sebessége.
- #define PIPE_FILL_SPEED 20

  A pályán lévő csövek feltölési animációjának a sebessége.

Enumerációk

- enum AnimationType { AnimationType_None , AnimationType_Rotate , AnimationType_Fill }

  Az AnimationQueue egy elemének típusa.
- enum AnimationDirection { AnimationDirection_ToRight , AnimationDirection_ToLeft , AnimationDirection_ToUp , AnimationDirection_ToDown }

  Az AnimationQueue-ban szereplő feltöltési animáció iránya.

Függvények

- GameStruct GameStruct_Init (void)

  GameStruct inicializálása (kinullázása)
- bool GameStruct_Create (GameStruct ∗game_struct)

  GameStruct létrehozása.
- void GameStruct_Destroy (GameStruct ∗game_struct)

  Felszabadít egy GameStruct-ot.
- EventloopVariables EventloopVariables_Init (void)

  Az eventloop változóinak inicializálása (kinullázása)
- void Delta_Update (EventloopVariables ∗ev_vars)

  Frissíti a deltaT elemének értékét.
- bool CursorInRect (EventloopVariables ev_vars, SDL_Rect bound)

  Megvizsgálja, hogy a kurzor a megadott SDL_Rect-en belül van-e.
- Wheel Wheel_Init (void)

  A Wheel struktúra inicializálása (kinullázása)
- bool Wheel_Create (Wheel ∗wheel, Window window, SDL_Rect location, SDL_Texture ∗wheel_texture)

  Wheel struktúra létrehozása.
- bool Wheel_Update (Wheel wheel, Window window, SDL_Texture ∗wheel_texture)

  Frissíti egy Wheel Layer-ét.
- void Wheel_Destroy (Wheel ∗wheel)

  Felszabadít egy Wheel struktúrát.
- AnimationQueue ∗ AnimationQueue_Init (void)

  Létrehozza egy AnimationQueue strázsáját.
- bool AnimationQueue_AppendWheelRotate (AnimationQueue ∗start, Wheel ∗wheel, double target_angle, double animation_speed)

  Hozzáfűz az AnimationQueue végéhez egy Wheel forgatási animációját.
- bool AnimationQueue_AppendPipeRotate (AnimationQueue ∗start, Pipe ∗pipe, double target_angle, double animation_speed)

  Hozzáfűz az AnimationQueue végéhez egy Pipe forgatási animációját.
- bool AnimationQueue_AppendLongFill (AnimationQueue ∗start, SDL_Rect render_location, SDL_Texture ∗filled_texture, double animation_speed, AnimationDirection direction)

  Hozzáfűz az AnimationQueue végéhez egy hosszú cső feltöltési animációját.
- bool AnimationQueue_AppendPipeFill (AnimationQueue ∗start, SDL_Renderer ∗renderer, Pipe ∗pipe, SDL_Texture ∗filled_texture, double animation_speed, AnimationDirection direction)

  Hozzáfűz az AnimationQueue végéhez egy Pipe feltöltési animációját.
- AnimationQueue ∗ AnimationQueue_RemoveElement (AnimationQueue ∗start, AnimationQueue ∗element)

  A megadott elemet kitörli az AniamtionQueue-ból és visszaadja az előtte lévő pointerét.
- void AnimationQueue_Destroy (AnimationQueue ∗start)

  Felszabadít egy AnimationQueue láncolt listát.
- bool Menu (GameStruct game_struct)

  A játékmenü függvénye.
- bool Game (GameStruct game_struct, PipeGrid ∗betoltott, int entry_y, int exit_y)

  A játék függvénye.
- bool Win (GameStruct game_struct, Layer state, PipeGrid copy, int entry_y, int exit_y)

  A nyert kép függvénye.

4.1.1. Részletes leírás

A játék headerje.

A játékkal kapcsolatban mindent tartalmaz. Innen már csak a main()-ben kell meghívogatni néhány függvényt. A header tartalmának nagy része csak a sok forrásfájl miatt szerepel itt.

Előfeltétel

Mivel az SDL_local.h-t használja, az SDL inicializálása szükséges a modul megfelelő működéséhez.

4.1.1.1. Struktúrák, enumok:

- `GameStruct`: Az SDL_local.h-ban szereplő struktúrákat fogja össze.
- `EventloopVariables`: A Game() függvény/kép eventloopjának a változóit fogja össze.
- `Wheel`: A csaphoz tartozó változókat fogja össze.
- `AnimationQueue`: Az felhasználó által indított animációkhoz egy strázsás láncolt lista.

Emellett az animáláshoz 2 enumot is tartalmaz a header:

- `AnimationType`: Az animáció típusa.
- `AnimationDirection`: Az animáció iránya.

4.1.1.2. Függvények:

Az itt szereplő függvények vagy a játékmenethez kapcsolódnak vagy a nevükben szereplő struktúrákhoz. A játék 3 állapota (menü, játék, nyerés kép) a saját függvényeikbe vannak szervezve.

4.1.1.3. Példakód:

```
// Ha Windows-on fordítjuk
#ifdef _WIN32
#include <windows.h>
#endif

#include <time.h> // rand() seed-jéhez

#include "game.h"
#include "debugmalloc.h"

// Ha Windows-on fordítjuk, az ablak ezt a main függvényt hívja meg
#ifdef WIN32
int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE FreeProcInstance, PWSTR pCmdLine, int nCmdShow) {
    (void)hInstance, (void)FreeProcInstance, (void)pCmdLine, (void)nCmdShow;
    return main(0, NULL);
}
#endif

// A main függvény
int main(int argc, char *argv[]) {
    (void)argc, (void)argv; // Warning elkerülése
    srand(time(0)); // Random seed

    if (!InitializeSDL()) return 1; // SDL inicializálása

    GameStruct game_struct = GameStruct_Init(); // SDL_local.h változóinak inicializálása
    if (!GameStruct_Create(&game_struct)) { // SDL_local.h változóinak létrehozása
        QuitSDL(); // Hiba esetén kilépés
        return 2;
```

```
    }

    StartMusic(game_struct.assets); // Zene indítása

    int ret = 0;
    if (!Menu(game_struct)) // Menübe lépés
        ret = 3;

    // Kilépés előtti felszabadítás
    Mix_FadeOutMusic(500);
    GameStruct_Destroy(&game_struct);
    QuitSDL();

    return ret;
}
```

4.1.1.4. Megjegyzések:

- A projekt main() függvényéből is látszik, hogy csak a GameStruct-ot kell létrehozni és a Menu() függvényt meghívni. A többi függvény hívást és struktúra létrehozást a Menu() végzi.
- A különböző if-ek a hibakezelés miatt vannak, hogy a program minél kevesebb memóriaszivárgással omoljon össze.

4.1.1.5. Kapcsolódó kódfájlok:

- game.c
- game_struct.c
- ev_vars.c
- wheel.c
- animation.c

Hiba A modul egy osdialog nevű modullal hoz létre rendszerfüggő dialógusablakokat. Ez a GTK2-re épül Linux alatt, ami memóriaszivárgást okoz. Windows alatt az Address Sanitizert nem teszteltem, ott nem tudom, hogy mi a helyzet. Egy lehetséges megoldás, hogy az osdialog-ot mással helyettesítjük, de emiatt a game.c-ben változtatásokat kell hozni.

4.1.2. Adatszerkezetek dokumentációja

4.1.2.1. struct GameStruct

A játékhoz szükséges struktúrákat fogja össze.

Az SDL_local.h struktúráit tartalmazza. Könnyebb kezelhetőség a célja, hiszen mindhárom struktúrát létre kell hozni a megfelelő működés érdekében.

Adatmezők

Assets	assets	Assets struktúra. Az assets mappa tartalmát fogja össze.
Fonts	fonts	Fonts struktúra. Az assets/fonts mappában található betűtípusokat tartalmazza.
Window	window	Window struktúra. Az ablakkal kapcsolatos változókat tartalmazza.

4.1.2.2. struct EventloopVariables

A játék eventloopjának a változóit fogja össze.

A Game() függvényben használt.

Adatmezők

bool	clicked	Kattintás állapota. true, amíg nem érkezik SDL_MOUSEBUTTONUP event.
Uint64	currTime	Az SDL_GetPerformanceCounter() függvény értéke.
double	deltaT	A két idő közötti különbség az SDL_GetPerformanceFrequency() értékével elosztva.
SDL_Event	event	A bejövő event. Az SDL_PollEvent() vagy SDL_WaitEvent() függvényeknek lehet átadni.
SDL_Point	prevCursor	Az SDL_MOUSEBUTTONDOWN event érkezésekor kimentett kurzorpozíció.
Uint64	prevTime	Az SDL_GetPerformanceCounter() függvény előző értéke.
bool	running	A játék állapota. Ha true, az eventloop fut, egyébként false.

4.1.2.3. struct Wheel

A csap struktúrája.

A csap megjelenítéséhez és forgatásához szükséges változókat tartalmazza.

Adatmezők

Layer	layer	A csap Layer-e.
double	szog	A csap forgatási szöge (fokban).

4.1.2.4. struct AnimationQueue

Az animációs sor struktúrája.

Az animációs sor maga egy strázsás láncolt lista. Kétféle animációhoz kezeléséhez tárol változókat

Adatmezők

bool	active	Az animáció állapota. true, amíg az animáció nem ért véget.
double *	angle	A forgatandó elem szögének pointere. (Csak AnimationType_Rotate esetén értelmes)
double	animation_speed	Az animáció sebessége.
AnimationDirection	direction	Az animáció iránya. (Csak AnimationType_Fill esetén értelmes)
SDL_Texture *	filled_texture	A renderelendő textúra. (Csak AnimationType_Fill esetén értelmes)
bool	is_pipe	Az animációt Pipe-on kell-e végrehajtani.

Adatmezők

struct AnimationQueue *	next	A következő elem pointere.
Pipe *	pipe	A Pipe pointere. (Csak is_pipe esetén értelmes)
double	ratio	A render_location és a texture_location aránya. (Csak AnimationType_Fill esetén értelmes)
SDL_Rect	render_location	Az elem RenderTarget-en belüli helye. (Csak AnimationType_Fill esetén értelmes)
double	rendered_angle	A struktúra által megtett szög (fokban). (Csak AnimationType_Rotate esetén értelmes)
double	target_angle	Meghatározza a forgatás mértékét (fokban). (Csak AnimationType_Rotate esetén értelmes)
double	target_coor	A cél koordináta. (Csak AnimationType_Fill esetén értelmes)
double	target_length	A renderelendő hossz (pixelben). (Csak AnimationType_Fill esetén értelmes)
SDL_Rect	texture_location	A renderelendő textúra helye a filled_texuter-n belül. (Csak AnimationType_Fill esetén értelmes)
AnimationType	type	Az animáció típusa.
Wheel *	wheel	A Wheel pointere. (Csak !is_pipe és AnimationType_Rotate esetén értelmes)

4.1.3. Makródefiníciók dokumentációja

4.1.3.1. DEFAULT_WINDOW_HEIGHT

```
#define DEFAULT_WINDOW_HEIGHT 900
```

Alapértelmezett ablakmagasság.

Az ablak magasságát határozza meg. A GameStruct_Create() függvény használja a Window struktúra létrehozásakor.

4.1.3.2. DEFAULT_WINDOW_WIDTH

```
#define DEFAULT_WINDOW_WIDTH 1600
```

Alapértelmezett ablakszélesség.

Az ablak szélességét határozza meg. A GameStruct_Create() függvény használja a Window struktúra létrehozásakor.

4.1.3.3. LONG_PIPE_FILL_SPEED

```
#define LONG_PIPE_FILL_SPEED 5
```

A pálya előtt lévő hosszú cső feltöltési animációjának a sebessége.

4.1.3.4. MIN_X_SIZE

```
#define MIN_X_SIZE 18
```

A játék pályáján szereplő, egy sorban lévő csövek minimális száma.

Egy sorban legalább ennyi cső szerepel.

4.1.3.5. MIN_Y_SIZE

```
#define MIN_Y_SIZE 14
```

A játék pályáján szereplő, egy oszlopban lévő csövek minimális száma.

Egy oszlopban legalább ennyi cső szerepel.

4.1.3.6. PIPE_FILL_SPEED

```
#define PIPE_FILL_SPEED 20
```

A pályán lévő csövek feltölési animációjának a sebessége.

4.1.3.7. PIPE_ROTATE_SPEED

```
#define PIPE_ROTATE_SPEED 3
```

A csövek forgatási animációjának a sebessége.

4.1.3.8. WHEEL_ROTATE_SPEED

```
#define WHEEL_ROTATE_SPEED 2
```

A csap forgatási animációjának a sebessége.

Más animációja nincs a csapnak.

4.1.3.9. WHEEL_SIZE_MULTIPLIER

```
#define WHEEL_SIZE_MULTIPLIER 1.75
```

A csap csövekhez viszonyított mérete.

Ennyiszer nagyobb egy csőhöz képest.

4.1.4. Enumerációk dokumentációja

4.1.4.1. AnimationDirection

```
enum AnimationDirection
```

Az AnimationQueue-ban szereplő feltöltési animáció iránya.

Csak AnimationType_Fill esetén értelmezett.

Enumeráció-értékek

AnimationDirection_ToRight	Jobbra tölt.
AnimationDirection_ToLeft	Balra tölt.
AnimationDirection_ToUp	Felfele tölt.
AnimationDirection_ToDown	Lefele tölt.

4.1.4.2. AnimationType

```
enum AnimationType
```

Az AnimationQueue egy elemének típusa.

Esetszétválasztásra szolgál.

Enumeráció-értékek

AnimationType_None	Alapértelmezett érték.
AnimationType_Rotate	Forgatás.
AnimationType_Fill	Feltöltés.

4.1.5. Függvények dokumentációja

4.1.5.1. AnimationQueue_AppendLongFill()

```
bool AnimationQueue_AppendLongFill (
            AnimationQueue * start,
            SDL_Rect render_location,
            SDL_Texture * filled_texture,
            double animation_speed,
            AnimationDirection direction )
```

Hozzáfűz az AnimationQueue végéhez egy hosszú cső feltöltési animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. A hosszú cső lehet más SDL_Texture∗ is, de a game.c-ben csak ehhez van használva. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
render_location	A renderelendő hosszú cső helye
filled_texture	A hosszú cső textúrája
animation_speed	Az animáció sebessége
direction	Az animáció iránya

Visszatérési érték

true, hiba esetén false

4.1.5.2. AnimationQueue_AppendPipeFill()

```
bool AnimationQueue_AppendPipeFill (
            AnimationQueue * start,
            SDL_Renderer * renderer,
            Pipe * pipe,
            SDL_Texture * filled_texture,
            double animation_speed,
            AnimationDirection direction )
```

Hozzáfűz az AnimationQueue végéhez egy Pipe feltöltési animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Hiba esetén megpróbálja a töltést kihagyni. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
renderer	Az ablak rendererje (csak hiba esetén használt)
pipe	A feltöltendő Pipe pointere
filled_texture	A Pipe feltöltött textúrája
animation_speed	Az animáció sebessége
direction	Az animáció iránya

Visszatérési érték

true, hiba esetén false

4.1.5.3. AnimationQueue_AppendPipeRotate()

```
bool AnimationQueue_AppendPipeRotate (
            AnimationQueue * start,
            Pipe * pipe,
            double target_angle,
            double animation_speed )
```

Hozzáfűz az AnimationQueue végéhez egy Pipe forgatási animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
wheel	A forgatandó Pipe pointere
target_angle	A forgatás mértéke (fokban)
animation_speed	Az animáció sebessége

Visszatérési érték

true, hiba esetén false

4.1.5.4. AnimationQueue_AppendWheelRotate()

```
bool AnimationQueue_AppendWheelRotate (
            AnimationQueue * start,
            Wheel * wheel,
            double target_angle,
            double animation_speed )
```

Hozzáfűz az AnimationQueue végéhez egy Wheel forgatási animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
wheel	A forgatandó Wheel pointere
target_angle	A forgatás mértéke (fokban)
animation_speed	Az animáció sebessége

Visszatérési érték

true, hiba esetén false

4.1.5.5. AnimationQueue_Destroy()

```
void AnimationQueue_Destroy (
            AnimationQueue * start )
```

Felszabadít egy AnimationQueue láncolt listát.

A strázsáját is felszabadítja. Használat után KELL használni.

Paraméterek

start	Az AnimationQueue strázsájának pointere

4.1.5.6. AnimationQueue_Init()

```
AnimationQueue * AnimationQueue_Init (
            void  )
```

Létrehozza egy AnimationQueue strázsáját.

malloc()-kal foglal egy AnimationQueue struktúrát és kinullázza. Ehhez a különböző AnimationQueue_Append függvényekkel lehet hozzáfűzni elemeket.

Utófeltétel

Használat után a AnimationQueue_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Visszatérési érték

Egy üres AnimationQueue struktúra

4.1.5.7. AnimationQueue_RemoveElement()

```
AnimationQueue * AnimationQueue_RemoveElement (
            AnimationQueue * start,
            AnimationQueue * element )
```

A megadott elemet kitörli az AniamtionQueue-ból és visszaadja az előtte lévő pointerét.

Figyelmeztetés

TILOS a strázsát átadni.

Paraméterek

start	Az AnimationQueue strázsájának pointere
element	A törlendő elem pointere

Visszatérési érték

A törlendő elem előtti elem pointere

4.1.5.8. CursorInRect()

```
bool CursorInRect (
            EventloopVariables ev_vars,
            SDL_Rect bound )
```

Megvizsgálja, hogy a kurzor a megadott SDL_Rect-en belül van-e.

Egy elemre történő kattintás vizsgálatára alkalmas. Az SDL_MOUSEBUTTONDOWN és az SDL_↩ MOUSEBUTTONUP közötti változást kiszűri, azaz csak akkor ad vissza true-t, ha a kurzor az SDL_Rect-en belül volt mindkét eseménykor.

Paraméterek

ev_vars	Egy EventloopVariables struktúra.
bound	A megvizsgálandó SDL_Rect.

Visszatérési érték

true, ha a kurzor az SDL_Rect-en belül volt SDL_MOUSEBUTTONDOWN és SDL_MOUSEBUTTONUP-kor, egyébként false

4.1.5.9. Delta_Update()

```
void Delta_Update (
            EventloopVariables * ev_vars )
```

Frissíti a deltaT elemének értékét.

Két hívás között eltelt időt számítja ki az SDL_GetPerformanceCounter() és az SDL_GetPerformanceFrequency() segítségével. Az animációk időzítésére szolgál.

Paraméterek

ev_vars	A változtatandó EventloopVariables struktúra pointere.

4.1.5.10. EventloopVariables_Init()

```
EventloopVariables EventloopVariables_Init (
            void  )
```

Az eventloop változóinak inicializálása (kinullázása)

Az visszaadot struktúra minden eleme 0 értéket kap. Mivel nincs dinamikusan foglalt változója, ezért nem kell felszabadítani.

Visszatérési érték

Egy EventloopVariables struktúra

4.1.5.11. Game()

```
bool Game (
            GameStruct game_struct,
            PipeGrid * betoltott,
            int entry_y,
            int exit_y )
```

A játék függvénye.

Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Megjegyzés

A Menu() függvény már képes meghívni, magában ne hívjuk meg, mert nem garantált a működése.

Paraméterek

game_struct	A létrehozott GameStruct.
betoltott	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid.
entry_y	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid bejáratánk sora.
exit_y	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid kijáratának sora.

Visszatérési érték

true, ha a Menu()-be lépünk vissza, false, ha új játék kezdődik, kilépés történt, illetve néhány súlyosabb hiba esetén

4.1.5.12. GameStruct_Create()

```
bool GameStruct_Create (
            GameStruct * game_struct )
```

GameStruct létrehozása.

Meghívja a GameStruct elemeinek létrehozó függvényeit (lásd: SDL_local.h). A Window létrehozásához a DEFAULT_WINDOW_WIDTH és DEFAULT_WINDOW_HEIGHT konstansokat használja. Átadni egy GameStruct_Init()-tel inicializált GameStruct-ot ajánlott. A hibákat a struktok létrehozó függvényei a konzolra kiírják, ennek csak a visszatérési értéke jelzi. Hiba esetén nem ajánlott a struktúra használata.

Utófeltétel

Használat után a GameStruct_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Figyelmeztetés

TILOS egy már létrehozott GameStruct-ot átadni.

Paraméterek

game_struct	Egy inicializált GameStruct pointere.

Visszatérési érték

true, hiba esetén false

4.1.5.13. GameStruct_Destroy()

```
void GameStruct_Destroy (
            GameStruct * game_struct )
```

Felszabadít egy GameStruct-ot.

A GameStruct elemeit szabadítja fel a saját függvényeikkel. (lásd: SDL_local.h)

Paraméterek

game_struct	A felszabadítandó GameStruct pointere.

4.1.5.14. GameStruct_Init()

```
GameStruct GameStruct_Init (
            void  )
```

GameStruct inicializálása (kinullázása)

Visszatérési érték

Egy üres GameStruct

4.1.5.15. Menu()

```
bool Menu (
            GameStruct game_struct )
```

A játékmenü függvénye.

Az SDL és a GameStruct inicializálása/létrehozása után meg lehet hívni. Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Paraméterek

game_struct	A létrehozott GameStruct.

Visszatérési érték

true, súlyos hiba esetén false

4.1.5.16. Wheel_Create()

```
bool Wheel_Create (
            Wheel * wheel,
            Window window,
            SDL_Rect location,
            SDL_Texture * wheel_texture )
```

Wheel struktúra létrehozása.

Véletlenszerűen felvesz egy szöget és lerendereli a Layer-re a csapot. A hibákat a struktok létrehozó függvényei a konzolra kiírják, ennek csak a visszatérési értéke jelzi. Hiba esetén nem ajánlott a struktúra használata.

Utófeltétel

Használat után a Wheel_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Figyelmeztetés

TILOS egy már létrehozott Wheel struktúrát átadni.

Paraméterek

wheel	Egy inicializált Wheel struktúra pointere.
window	A renderert tartalmazó Window struktúra.
location	A csap szülőjéhez relatív helye
wheel_texture	A csap textúrája.

Visszatérési érték

true, hiba esetén false

4.1.5.17. Wheel_Destroy()

```
void Wheel_Destroy (
            Wheel * wheel )
```

Felszabadít egy Wheel struktúrát.

A Wheel Layer-ét szabadítja fel a saját függvényével. (lásd: SDL_local.h)

Paraméterek

wheel	A felszabadítandó Wheel struktúra pointere.

4.1.5.18. Wheel_Init()

```
Wheel Wheel_Init (
            void  )
```

A Wheel struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Wheel struktúra

4.1.5.19. Wheel_Update()

```
bool Wheel_Update (
            Wheel wheel,
            Window window,
            SDL_Texture * wheel_texture )
```

Frissíti egy Wheel Layer-ét.

A Wheel Layer-ét frissíti (újrarendereli) az aktuális értékei szerint. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

wheel	A frissítendő Wheel
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture∗

Visszatérési érték

true, hiba esetén false

4.1.5.20. Win()

```
bool Win (
            GameStruct game_struct,
            Layer state,
            PipeGrid copy,
            int entry_y,
            int exit_y )
```

A nyert kép függvénye.

Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Megjegyzés

A Game() függvény már képes meghívni, magában ne hívjuk meg, mert nem garantált a működése.

Paraméterek

game_struct	A létrehozott GameStruct.
state	A Game() függvény aktuális renderelt képe.
copy	A PipeGrid pálya betöltéskori másolata.
entry_y	A PipeGrid bejáratának sora.
exit_y	A PipeGrid kijáratának sora.

Visszatérési érték

true, ha a Menu()-be lépünk vissza, false, ha új játék kezdődik, kilépés történt, illetve néhány súlyosabb hiba esetén

4.2. game.h

Ugrás a fájl dokumentációjához.

```
00001
00051 /* game.h eleje */
00052 #ifndef GAME_H
00053 #define GAME_H
00054
00055 #include "SDL_local.h"
00056 #include "pipe.h"
00057 #include "osdialog.h"
00058
00063 #define DEFAULT_WINDOW_WIDTH 1600
00068 #define DEFAULT_WINDOW_HEIGHT 900
00069
00074 #define MIN_X_SIZE 18
00079 #define MIN_Y_SIZE 14
00080
00085 #define WHEEL_SIZE_MULTIPLIER 1.75
00086
00091 #define WHEEL_ROTATE_SPEED 2
00095 #define PIPE_ROTATE_SPEED 3
00099 #define LONG_PIPE_FILL_SPEED 5
00103 #define PIPE_FILL_SPEED 20
00104
00110 typedef enum AnimationType {
00114     AnimationType_None,
00118     AnimationType_Rotate,
00122     AnimationType_Fill
00123 } AnimationType;
00124
00130 typedef enum AnimationDirection {
00134     AnimationDirection_ToRight,
00138     AnimationDirection_ToLeft,
00142     AnimationDirection_ToUp,
00146     AnimationDirection_ToDown
00147 } AnimationDirection;
00148
00155 typedef struct GameStruct {
00160     Window window;
00165     Assets assets;
00170     Fonts fonts;
00171 } GameStruct;
00172
00178 typedef struct EventloopVariables {
00183     SDL_Event event;
00188     bool running;
00193     bool clicked;
00197     SDL_Point prevCursor;
00201     Uint64 currTime;
00205     Uint64 prevTime;
00209     double deltaT;
00210 } EventloopVariables;
00211
00217 typedef struct Wheel {
00221     Layer layer;
00225     double szog;
00226 } Wheel;
00227
00233 typedef struct AnimationQueue {
00238     bool active;
00242     AnimationType type;
00246     bool is_pipe;
00250     double animation_speed;
00251
00255     Pipe * pipe;
00259     Wheel * wheel;
00260
00264     double target_angle;
00268     double rendered_angle;
00272     double * angle;
00273
00277     AnimationDirection direction;
00281     double target_length;
00285     double target_coor;
00289     double ratio;
```

```
00293     SDL_Rect texture_location;
00297     SDL_Rect render_location;
00301     SDL_Texture * filled_texture;
00302
00306     struct AnimationQueue * next;
00307 } AnimationQueue;
00308
00313 GameStruct GameStruct_Init(void);
00326 bool GameStruct_Create(GameStruct * game_struct);
00332 void GameStruct_Destroy(GameStruct * game_struct);
00333
00340 EventloopVariables EventloopVariables_Init(void);
00347 void Delta_Update(EventloopVariables * ev_vars);
00356 bool CursorInRect(EventloopVariables ev_vars, SDL_Rect bound);
00357
00362 Wheel Wheel_Init(void);
00376 bool Wheel_Create(Wheel * wheel, Window window, SDL_Rect location, SDL_Texture * wheel_texture);
00386 bool Wheel_Update(Wheel wheel, Window window, SDL_Texture * wheel_texture);
00392 void Wheel_Destroy(Wheel * wheel);
00393
00401 AnimationQueue * AnimationQueue_Init(void);
00412 bool AnimationQueue_AppendWheelRotate(AnimationQueue * start, Wheel * wheel, double target_angle,
      double animation_speed);
00423 bool AnimationQueue_AppendPipeRotate(AnimationQueue * start, Pipe * pipe, double target_angle, double
      animation_speed);
00436 bool AnimationQueue_AppendLongFill(AnimationQueue * start, SDL_Rect render_location, SDL_Texture *
      filled_texture, double animation_speed, AnimationDirection direction);
00450 bool AnimationQueue_AppendPipeFill(AnimationQueue * start, SDL_Renderer * renderer, Pipe * pipe,
      SDL_Texture * filled_texture, double animation_speed, AnimationDirection direction);
00458 AnimationQueue * AnimationQueue_RemoveElement(AnimationQueue * start, AnimationQueue * element);
00464 void AnimationQueue_Destroy(AnimationQueue * start);
00465
00474 bool Menu(GameStruct game_struct);
00486 bool Game(GameStruct game_struct, PipeGrid * betoltott, int entry_y, int exit_y);
00499 bool Win(GameStruct game_struct, Layer state, PipeGrid copy, int entry_y, int exit_y);
00500
00501 #endif
00502 /* game.h vége */
```

4.3. lib/include/pipe.h fájlreferencia

A játékban használt csövekhez kapcsolódó függvényeket és adatszerkezeteket tartalmazza a fájl.

```
#include "SDL_local.h"
```

Adatszerkezetek

- struct Pipe

  Egy cső adatainak tárolására szolgáló struktúra. Részletek...
- struct PipeGrid

  A Pipe-ok mátrixát tároló struktúra. Részletek...
- struct SolvedPipeGrid

  Egy PipeGrid megoldását tároló láncolt lista. Részletek...

Makródefiníciók

- #define DEFAULT_PIPE_SIZE 50

  A csövek alapértelmezett magassága és szélessége (pixelben)

Típusdefiníciók

- typedef ElemiResz Elem[3][3]

  Egy Elem-et reprezentáló 3x3-as mátrix.

Enumerációk

- enum ElemiResz { Levego = 'X' , Cso = 'O' , Ismeretlen = '?' , Rossz = '.' , Jo = '∗' }

  Egy-egy Elem részeinek lehetséges állapotai.
- enum PipeType { PipeType_None , PipeType_Straight , PipeType_Curved , PipeType_Tshaped , PipeType_Crossing }

  A Pipe típusának enumja.

Függvények

- void Elem_Init (Elem elem)

  Egy üres (csupa Levego-s) Elem-et álltít elő.
- void Elem_Copy (Elem hova, Elem honnan)

  Átmásol egy Elem tartalmát egy másikba.
- void Elem_RotatePos (Elem elem)

  Elforgat egy Elem-et az óramutató járásával megegyező irányba.
- void Elem_RotateNeg (Elem elem)

  Elforgat egy Elem-et az óramutató járásával ellentétes irányba.
- Pipe Pipe_Init (void)

  Pipe struktúra inicializálása (kinullázása/alapértelmezett értékek beállítása)
- bool Pipe_CreateFromElem (Pipe ∗pipe, Window window, SDL_Texture ∗pipe_texture, SDL_Rect base_↩ location)

  Pipe struktúra létrehozása a benne található Elem alapján.
- void Pipe_Rotate (Pipe ∗pipe, int irany)

  Elforgatja egy Pipe Elem-ét.
- bool Pipe_Render (Pipe pipe, Window window)

  Renderel egy Pipe-ot a jelenlegi RenderTarget-re.
- bool Pipe_UpdateLayer (Pipe pipe, Window window, SDL_Texture ∗pipe_texture)

  Frissíti egy Pipe Layer-ét.
- void Pipe_Destroy (Pipe ∗pipe)

  Felszabadít egy Pipe struktúrát.
- PipeGrid PipeGrid_Init (void)

  PipeGrid struktúra inicializálása (kinullázása)
- bool PipeGrid_Create (PipeGrid ∗pipegrid, int x, int y)

  PipeGrid struktúra létrehozása.
- PipeGrid PipeGrid_CreateCopy (PipeGrid source)

  Lemásol egy PipeGrid-et és visszaadja a másolatot.
- void PipeGrid_GenerateGrid (PipeGrid ∗pipegrid, SDL_Point entry, SDL_Point exit)

  Legenerál egy PipeGrid-et.
- void PipeGrid_Shuffle (PipeGrid pipegrid)

  Összekever egy PipeGrid-et.
- bool PipeGrid_Feldolgoz (PipeGrid pipegrid, Window window, SDL_Texture ∗pipe_texture, SDL_Rect base↩ _location)

  Egy PipeGrid Pipe-jainak hiányzó értékeit feltölti.
- bool PipeGrid_Render (PipeGrid pipegrid, Window window)

  Lerenderel egy feldolgozott PipeGrid-et az aktuális RenderTarget-re.
- void PipeGrid_Destroy (PipeGrid ∗pipegrid)

  Felszabadít egy PipeGrid-et.
- SolvedPipeGrid ∗ PipeGrid_Solve (PipeGrid pipegrid, SDL_Point entry, SDL_Point exit)

  Megold egy PipeGrid-et és a megoldását egy SolvedPipeGrid-ben adja vissza.
- SolvedPipeGrid ∗ SolvedPipeGrid_RemoveFirst (SolvedPipeGrid ∗start)

  A SolvedPipeGrid láncolt lista első elemét felszabadítja és visszaadja a következő elem pointerét.
- void SolvedPipeGrid_Destroy (SolvedPipeGrid ∗start)

  Felszabadít egy SolvedPipeGrid láncolt listát.

4.3.1. Részletes leírás

A játékban használt csövekhez kapcsolódó függvényeket és adatszerkezeteket tartalmazza a fájl.

Előfeltétel

Mivel az SDL_local.h-t használja, az SDL inicializálása szükséges a modul megfelelő működéséhez.

4.3.1.1. Struktúrák, enumok:

3 fő struktúrát tartalmaz a fájl:

- `Pipe`: Egy cső adatait tartalmazza.
- `PipeGrid`: Csövek mátrixát és a mátrix szélességét és magasságát tartalmazza.
- `SolvedPipeGrid`: Egy PipeGrid megoldását tartalmazza. A megfelelő csöveket és indexeiket strázsa nélküli láncolt listában tartalmazza.

A struktúrák használata többnyire az alábbi módon történik:

```
Struktúra név = Struktúra_Init(); // Az init egy üres elemet ad vissza, nem foglal magának memóriát
Struktúra_LétrehozóFüggvény(&struktúra, paraméterek); // Lásd lentebb

// Használat

Struktúra_Destroy(&struktúra); // A struktúrákat, amik tartalmaznak dinamikusan foglalt elemeket, fel kell
      szabadítani
```

Ez alól van kivétel, lásd lentebb.

Maga a cső az `Elem` típusú 3x3-as mátrixban van leképezve, ami `ElemiResz` enumokból épül fel. Ebben van nyilvántartva maga a cső és a megoldás során az "áramlás" útvonalának helyessége.

Ezen kívül a `PipeType` enum egy Pipe típusát határozza meg.

4.3.1.2. Függvények:

A modul a struktúrákhoz kapcsolódó függvényeket tartalmazza.

4.3.1.3. Példakód:

```
#include "SDL_local.h" // Explicit include
#include "pipe.h"

int main(int argc, char * argv[]) {
    (void)argc, (void)argv; // Warning elkerülése
    InitializeSDL(); // SDL inicializálása

    PipeGrid pipegrid = PipeGrid_Init(); // PipeGrid inicializálása
    PipeGrid_Create(&pipegrid, 10, 10); // 10x10-es PipeGrid létrehozása
    PipeGrid_GenerateGrid(&pipegrid, (SDL_Point){0, 0}, (SDL_Point){9, 9}); // PipeGrid feltöltése

    SolvedPipeGrid * solved = PipeGrid_Solve(pipegrid, (SDL_Point){0, 0}, (SDL_Point){9, 9}); // PipeGrid
      megoldása

    // Megoldás koordinátáinak kiírása
    for (SolvedPipeGrid * i = solved; i != NULL; i = i->next)
        printf("%d, %d\n", i->x, i->y);

    SolvedPipeGrid_Destroy(solved); // SolvedPipeGrid felszabadítása
    PipeGrid_Destroy(&pipegrid); // PipeGrid felszabadítása
    QuitSDL(); // SDL leállítása
    return 0;
}
```

A fenti példakód létrehoz egy 10x10-es PipeGrid-et, majd generál ebből egy pályát. Ezután megoldja a pályát, majd kiírja stdout-on a megfelelő csövek koordinátáit (indexeit).

4.3.1.4. Megjegyzések:

- Minden innen származó struktúrának a "legtetejét" kell felszabadítani. Ezalatt azt kell érteni, hogy például a PipeGrid tartalmazza a szintén dinamikusan foglalt Pipe-ot, de a PipeGrid_Destroy() a tárolt Pipe-ot is felszabadítja. Erről bővebben az adott függvény dokumentációja ír.

4.3.1.5. Kapcsolódó kódfájlok:

- elem.c
- pipe.c
- pipegrid.c
- solve.c

4.3.2. Adatszerkezetek dokumentációja

4.3.2.1. struct Pipe

Egy cső adatainak tárolására szolgáló struktúra.

Egy cső feldolgozásához, illetve egy cső rendereléséhez szükséges összes adatot tárolja.

Adatmezők

Layer	rendered	A cső rétege. Azért kell, hogy csakis egyetlen csőt meg lehessen változtatni.
double	szog	A cső szöge (fokban). Rendereléskor (animáláskor) használt.
Elem	tenyleges	A cső PipeGrid_GenerateGrid()-ben és PipeGrid_Solve()-ban használt alakja.
SDL_Rect	texture_loc	A cső textúrájának helye az assets/img/sprites_empty.png-n belül (ez az Assets struktúra pipes_empty eleme).
PipeType	type	A cső típusa.

4.3.2.2. struct PipeGrid

A Pipe-ok mátrixát tároló struktúra.

Egy y∗x-es mátrix, amiben Pipe-ok vannak. Indexeléskor oda kell figyelni, hogy először a sor (y), majd az oszlop (x) indexét kell megadni.

Adatmezők

Pipe ∗∗	matrix	A Pipe mátrix.
int	x	Sorszélesség ("x tengely")
int	y	Oszlopmagasság ("y tengely")

4.3.2.3. struct SolvedPipeGrid

Egy PipeGrid megoldását tároló láncolt lista.

Egy PipeGrid megoldását tárolja. A megfelelő Pipe-okat és indexeiket strázsa nélküli láncolt listában tárolja. Nincsen Init() függvénye, helyette egy PipeGrid-ből hozzza létre magát.

Adatmezők

struct SolvedPipeGrid *	next	A láncolt lista következő elemére mutató pointer.
Pipe *	pipe	Egy Pipe pointere (a PipeGrid-ben).
int	x	A Pipe oszlop (x) indexe a PipeGrid-ben.
int	y	A Pipe sor (y) indexe a PipeGrid-ben.

4.3.3. Makródefiníciók dokumentációja

4.3.3.1. DEFAULT_PIPE_SIZE

```
#define DEFAULT_PIPE_SIZE 50
```

A csövek alapértelmezett magassága és szélessége (pixelben)

4.3.4. Típusdefiníciók dokumentációja

4.3.4.1. Elem

```
Elem
```

Egy Elem-et reprezentáló 3x3-as mátrix.

Egy-egy Pipe egyszerűbb alakja. A PipeGrid_GenerateGrid() és a PipeGrid_Solve() függvények egyszerűbb működéséhez szükséges.

4.3.5. Enumerációk dokumentációja

4.3.5.1. ElemiResz

```
enum ElemiResz
```

Egy-egy Elem részeinek lehetséges állapotai.

Egy Elem generálás után Levego és Cso ElemiResz-ekből áll. A másik három állapot a PipeGrid_Solve() függvénynél jön elő.

Enumeráció-értékek

Levego	Egy Elem-en belül a levegőt jelöli (azaz nem Cso).
Cso	Egy Elem-en belül a csövet jelöli. Ebben mehet a víz.
Ismeretlen	A PipeGrid_Solve() során egy eldöntetlen útvonalnál használt.
Rossz	A PipeGrid_Solve() során egy rossz útvonalnál használt.
Jo	A PipeGrid_Solve() során egy jó útvonalnál használt.

4.3.5.2. PipeType

```
enum PipeType
```

A Pipe típusának enumja.

Nincs túl sok helyen használva, viszont hasznos információt mond el egy Pipe-ról. Emellett ezek a lehetséges típusok.

Enumeráció-értékek

PipeType_None	Az feldolgozás előtti alapértelmezett érték. Egyébként nem lehetséges.
PipeType_Straight	Egyenes cső.
PipeType_Curved	Kanyarodó cső.
PipeType_Tshaped	T alakú cső.
PipeType_Crossing	Kereszteződő/Egymáson átmenő cső.

4.3.6. Függvények dokumentációja

4.3.6.1. Elem_Copy()

```
void Elem_Copy (
            Elem hova,
            Elem honnan )
```

Átmásol egy Elem tartalmát egy másikba.

A honnan Elem tartalmát másolja át a hova Elem-be. A hova Elem tartalmát felülírja. A honnan elem tartalmát nem változtatja, viszont a fordító warningot ad, ha const a paraméter.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

hova	A cél Elem
honnan	A forrás Elem

4.3.6.2. Elem_Init()

```
void Elem_Init (
            Elem elem )
```

Egy üres (csupa Levego-s) Elem-et álltít elő.

Az átadott Elem-et feltölti Levego-vel. A C nyelv sajátossága miatt tér el a többi Init függvénytől.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	Az inicializálandó Elem

4.3.6.3. Elem_RotateNeg()

```
void Elem_RotateNeg (
            Elem elem )
```

Elforgat egy Elem-et az óramutató járásával ellentétes irányba.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	A változtatandó Elem

4.3.6.4. Elem_RotatePos()

```
void Elem_RotatePos (
            Elem elem )
```

Elforgat egy Elem-et az óramutató járásával megegyező irányba.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	A változtatandó Elem

4.3.6.5. Pipe_CreateFromElem()

```
bool Pipe_CreateFromElem (
            Pipe * pipe,
            Window window,
            SDL_Texture * pipe_texture,
            SDL_Rect base_location )
```

Pipe struktúra létrehozása a benne található Elem alapján.

A függvény létrehozza a Pipe Layer-ét, illetve beállítja a szülőjéhez relatív helyét (lásd: Layer), felvesz egy szöget és beállítja a PipeType-ot, illetve a textúrájának helyét. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi. Hiba esetén nem ajánlott a létrejött struktúrát használni.

Utófeltétel

Mivel létrehoz egy Layer-t, használat után a Pipe-ot fel kell szabadítani a Pipe_Destroy() függvénnyel.

Megjegyzés

A projekten belül található Create függvényektől eltérően inkább feldolgoz, mintsem létrehoz. Emiatt is lehetőleg csak a PipeGrid_Feldolgoz() függvényen belül maradjon használva.

Paraméterek

pipe	A feldolgozandó Pipe pointere
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture*
base_location	A Pipe relatív helye

Visszatérési érték

true, hiba esetén false

4.3.6.6. Pipe_Destroy()

```
void Pipe_Destroy (
            Pipe * pipe )
```

Felszabadít egy Pipe struktúrát.

Az átadott Pipe_CreateFromElem() függvénnyel létrehozott Pipe-ot felszabadítja. Lényegében a struktúra Layer-ét szabadítja fel.

Paraméterek

pipe	A felaszabadítandó Pipe struktúra

4.3.6.7. Pipe_Init()

```
Pipe Pipe_Init (
            void  )
```

Pipe struktúra inicializálása (kinullázása/alapértelmezett értékek beállítása)

Visszatérési érték

Egy üres Pipe struktúra

4.3.6.8. Pipe_Render()

```
bool Pipe_Render (
            Pipe pipe,
            Window window )
```

Renderel egy Pipe-ot a jelenlegi RenderTarget-re.

A Pipe Layer-ét rendereli a jelenlegi RenderTarget-re. Fontos, hogy a RenderTarget a "szülője" legyen, azaz az, amihez a Pipe Layer-ének helye relatív. Egyéb esetben nem garantált a helyes működés. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

pipe	A renderelendő Pipe
window	A renderert tartalmazó Window struktúra

Visszatérési érték

true, hiba esetén false

4.3.6.9. Pipe_Rotate()

```
void Pipe_Rotate (
            Pipe * pipe,
            int irany )
```

Elforgatja egy Pipe Elem-ét.

Az iránynak megfelelően elforgatja a Pipe Elem-ét. Ha az irány 1, az óramutató járásával megegyező irányba, ha -1, akkor az óramutató járásával ellentétes irányba. Ha a Pipe típusa PipeType_Crossing, akkor nem történik semmi.

Paraméterek

pipe	A változtatandó Pipe
irany	A forgatás iránya (1: óramutató járásával megegyező, -1: óramutató járásával ellentétes)

4.3.6.10. Pipe_UpdateLayer()

```
bool Pipe_UpdateLayer (
            Pipe pipe,
            Window window,
            SDL_Texture * pipe_texture )
```

Frissíti egy Pipe Layer-ét.

A Pipe Layer-ét frissíti (újrarendereli) az aktuális értékei szerint. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

pipe	A frissítendő Pipe
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture*

Visszatérési érték

true, hiba esetén false

4.3.6.11. PipeGrid_Create()

```
bool PipeGrid_Create (
            PipeGrid * pipegrid,
            int x,
            int y )
```

PipeGrid struktúra létrehozása.

Létrehoz egy PipeGrid-et a megadott szélességgel és magassággal. A létrejövő mátrix elemeit a Pipe_Init()-tel hozza létre. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi. Hiba esetén a struktúra mátrixa NULL.

Utófeltétel

Mivel létrehoz egy Pipe mátrixot, használat után a PipeGrid-ot fel kell szabadítani a PipeGrid_Destroy() függvénnyel.

Figyelmeztetés

TILOS egy, már létrehozott, PipeGrid struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

pipegrid	A létrehozandó PipeGrid pointere
x	A PipeGrid szélessége
y	A PipeGrid magassága

Visszatérési érték

true, hiba esetén false

4.3.6.12. PipeGrid_CreateCopy()

```
PipeGrid PipeGrid_CreateCopy (
            PipeGrid source )
```

Lemásol egy PipeGrid-et és visszaadja a másolatot.

Csak a Pipe-ok Elem-ét másolja le, ezért ha használni akarjuk a másolatot, akkor először fel kell dolgozni a PipeGrid_Feldolgoz() függvénnyel.

Utófeltétel

A másolatot a PipeGrid_Create()-tel hozza létre, így fel kell szabadítani használat után a PipeGrid_Destroy()-el.

Paraméterek

source	A forrás PipeGrid

Visszatérési érték

Egy új PipeGrid, hiba esetén ennek a mátrixa NULL

4.3.6.13. PipeGrid_Destroy()

```
void PipeGrid_Destroy (
            PipeGrid * pipegrid )
```

Felszabadít egy PipeGrid-et.

A PipeGrid-ben lévő Pipe-okat is felszabadítja.

Paraméterek

pipegrid	A felaszabadítandó PipeGrid pointere

4.3.6.14. PipeGrid_Feldolgoz()

```
bool PipeGrid_Feldolgoz (
            PipeGrid pipegrid,
            Window window,
            SDL_Texture * pipe_texture,
            SDL_Rect base_location )
```

Egy PipeGrid Pipe-jainak hiányzó értékeit feltölti.

A PipeGrid-ben található Pipe-ok Elem-ei alapján feltölti a Pipe-ok hiányzó értékeit a Pipe_CreateFromElem()-el, azaz létrehozza a Layer-ét, beállítja a szülőjéhez relatív helyét, felvesz egy szöget és beállítja a PipeType-ot, illetve a textúrájának helyét. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott a struktúra használata.

Előfeltétel

A Pipe-ok Elem-jeinek már be kell legyen állítva az értéke. Emiatt csak a "végleges" állapotban ajánlott használni, azaz a PipeGrid_GenerateGrid() és a PipeGrid_Shuffle() függvények után.

Figyelmeztetés

TILOS egy, már feldolgozott, PipeGrid struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

pipegrid	A feldolgozandó PipeGrid
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture∗
base_location	A PipeGrid megjelenítési helye

Visszatérési érték

true, hiba esetén false

4.3.6.15. PipeGrid_GenerateGrid()

```
void PipeGrid_GenerateGrid (
            PipeGrid * pipegrid,
            SDL_Point entry,
            SDL_Point exit )
```

Legenerál egy PipeGrid-et.

A játékhoz az átadott PipeGrid-ben generál egy pályát. Ez egy rekurzív labirintusgeneráló algoritmussal történik, aminek a segítségével egy véletlenszerű, megoldható pályát generál. Alapul az infoc-n található algoritmus szolgált (link: https://infoc.eet.bme.hu/labirintus/), ami módosítva lett, hogy az általam használt adatszerkezetekkel működjön (Pipe-on belüli Elem).

Figyelmeztetés

A belépési és kilépési pontok szabadon megadhatóak, DE az entry lehetőleg az első oszlopban, az exit pedig az utolsóban legyen. Egyéb esetben nem garantált a helyes működés.

Paraméterek

pipegrid	A változtatandó PipeGrid pointere
entry	A belépési pont (x: oszlopindex, y: sorindex)
exit	A kilépési pont (x: oszlopindex, y: sorindex)

4.3.6.16. PipeGrid_Init()

```
PipeGrid PipeGrid_Init (
            void  )
```

PipeGrid struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres PipeGrid struktúra

4.3.6.17. PipeGrid_Render()

```
bool PipeGrid_Render (
            PipeGrid pipegrid,
            Window window )
```

Lerenderel egy feldolgozott PipeGrid-et az aktuális RenderTarget-re.

Az előforduló hibákat a függvény kiírja a konzolra.

Paraméterek

pipegrid	Egy feldolgozott PipeGrid
window	A renderert tartalmazó Window struktúra

Visszatérési érték

true, hiba esetén false

4.3.6.18. PipeGrid_Shuffle()

```
void PipeGrid_Shuffle (
            PipeGrid pipegrid )
```

Összekever egy PipeGrid-et.

A PipeGrid-en belül lévő Pipe-ok Elem-eit véletlenszerűen elforgatja.

Paraméterek

pipegrid	Az összekeverendő PipeGrid

4.3.6.19. PipeGrid_Solve()

```
SolvedPipeGrid * PipeGrid_Solve (
            PipeGrid pipegrid,
            SDL_Point entry,
            SDL_Point exit )
```

Megold egy PipeGrid-et és a megoldását egy SolvedPipeGrid-ben adja vissza.

Át kell adni a PipeGrid_GenerateGrid()-ben használt belépési és kilépési pontokat, egyébként nem fogja jól megoldani.

Utófeltétel

Mivel egy láncolt listát hoz létre a függvény, használat után a SolvedPipeGrid_Destroy()-al kell felszabadítani.

Paraméterek

pipegrid	A megoldandó PipeGrid
entry	A belépési pont (x: oszlopindex, y: sorindex)
exit	A kilépési pont (x: oszlopindex, y: sorindex)

Visszatérési érték

A megoldás láncolt listájának első eleme, hiba vagy megoldás hiánya esetén NULL.

4.3.6.20. SolvedPipeGrid_Destroy()

```
void SolvedPipeGrid_Destroy (
            SolvedPipeGrid * start )
```

Felszabadít egy SolvedPipeGrid láncolt listát.

Paraméterek

start	A láncolt lista első elemére mutató pointer

4.3.6.21. SolvedPipeGrid_RemoveFirst()

```
SolvedPipeGrid * SolvedPipeGrid_RemoveFirst (
            SolvedPipeGrid * start )
```

A SolvedPipeGrid láncolt lista első elemét felszabadítja és visszaadja a következő elem pointerét.

Paraméterek

start	A láncolt lista első elemére mutató pointer

Visszatérési érték

A következő elemre mutató pointer (NULL, ha nincs több elem)

4.4. pipe.h

Ugrás a fájl dokumentációjához.

```
00001
00081 /* pipe.h eleje */
00082 #ifndef PIPE_H
00083 #define PIPE_H
00084
00085 #include "SDL_local.h"
00086
00090 #define DEFAULT_PIPE_SIZE 50
00091
00098 typedef enum ElemiResz {
00102     Levego = 'X',
```

```
00106     Cso = 'O',
00110     Ismeretlen = '?',
00114     Rossz = '.',
00118     Jo = '*'
00119 } ElemiResz;
00120
00126 typedef enum PipeType {
00130     PipeType_None,
00134     PipeType_Straight,
00138     PipeType_Curved,
00142     PipeType_Tshaped,
00146     PipeType_Crossing
00147 } PipeType;
00148
00155 typedef ElemiResz Elem[3][3];
00156
00162 typedef struct Pipe {
00166     Elem tenyleges;
00170     SDL_Rect texture_loc;
00174     Layer rendered;
00178     PipeType type;
00182     double szog;
00183 } Pipe;
00184
00191 typedef struct PipeGrid {
00195     Pipe **matrix;
00204     int x, y;
00205 } PipeGrid;
00206
00213 typedef struct SolvedPipeGrid {
00217     Pipe * pipe;
00226     int x, y;
00230     struct SolvedPipeGrid * next;
00231 } SolvedPipeGrid;
00232
00239 void Elem_Init(Elem elem);
00248 void Elem_Copy(Elem hova, Elem honnan);
00254 void Elem_RotatePos(Elem elem);
00260 void Elem_RotateNeg(Elem elem);
00261
00266 Pipe Pipe_Init(void);
00282 bool Pipe_CreateFromElem(Pipe * pipe, Window window, SDL_Texture * pipe_texture, SDL_Rect
      base_location);
00290 void Pipe_Rotate(Pipe * pipe, int irany);
00300 bool Pipe_Render(Pipe pipe, Window window);
00310 bool Pipe_UpdateLayer(Pipe pipe, Window window, SDL_Texture * pipe_texture);
00317 void Pipe_Destroy(Pipe * pipe);
00318
00323 PipeGrid PipeGrid_Init(void);
00337 bool PipeGrid_Create(PipeGrid * pipegrid, int x, int y);
00346 PipeGrid PipeGrid_CreateCopy(PipeGrid source);
00359 void PipeGrid_GenerateGrid(PipeGrid * pipegrid, SDL_Point entry, SDL_Point exit);
00365 void PipeGrid_Shuffle(PipeGrid pipegrid);
00382 bool PipeGrid_Feldolgoz(PipeGrid pipegrid, Window window, SDL_Texture * pipe_texture, SDL_Rect
      base_location);
00390 bool PipeGrid_Render(PipeGrid pipegrid, Window window);
00396 void PipeGrid_Destroy(PipeGrid * pipegrid);
00397
00407 SolvedPipeGrid * PipeGrid_Solve(PipeGrid pipegrid, SDL_Point entry, SDL_Point exit);
00413 SolvedPipeGrid * SolvedPipeGrid_RemoveFirst(SolvedPipeGrid * start);
00418 void SolvedPipeGrid_Destroy(SolvedPipeGrid * start);
00419
00420 #endif
00421 /* pipe.h vége */
```

4.5. lib/include/SDL_local.h fájlreferencia

Ez a modul tartalmazza az SDL-lel kapcsolatos függvényeket, és a hozzájuk tartozó adatszerkezeteket.

```
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_ttf.h>
#include <SDL_mixer.h>
#include <SDL2_gfxPrimitives.h>
#include <stdbool.h>
```

Adatszerkezetek

- struct Window

  Window struktúra. Részletek...
- struct Assets

  Assets mappa fájlaihoz struktúra. Részletek...
- struct Fonts

  Betűtípusokhoz struktúra. Részletek...
- struct Layer

  Réteg struktúra. Részletek...

Makródefiníciók

- #define DEFAULT_FONT_SIZE 200

  Alapértelmezett betűméret.
- #define MIX_VOLUME 70

  Alapértelmezett hangerő

Függvények

- bool InitializeSDL (void)

  Az SDL inicializáló függvénye.
- void StartMusic (Assets assets)

  Elindítja a háttérzenét és végtelenül ismétli.
- void QuitSDL (void)

  Leállítja az SDL-lel kapcsolatos modulokat.
- Window Window_Init (void)

  Window struktúra inicializálása (kinullázása)
- bool Window_Create (Window ∗window, const char ∗title, int width, int height)

  Feltölt egy inicializált Window struktúrát.
- void Window_Destroy (Window ∗window)

  Felszabadít egy Window struktúrát.
- Assets Assets_Init (void)

  Assets struktúra inicializálása (kinullázása)
- bool Assets_Load (Assets ∗assets, Window window)

  Feltölt egy inicializált Assets struktúrát.
- void Assets_Destroy (Assets ∗assets)

  Felszabadít egy Assets struktúrát.
- Fonts Fonts_Init (void)

  Fonts struktúra inicializálása (kinullázása)
- bool Fonts_Load (Fonts ∗fonts)

  Feltölt egy inicializált Fonts struktúrát.
- void Fonts_Destroy (Fonts ∗fonts)

  Felszabadít egy Fonts struktúrát.
- Layer Layer_Init (void)

  Layer struktúra inicializálása (kinullázása)
- bool Layer_Create (Layer ∗layer, Window window, const SDL_Rect ∗location)

  Feltölt egy inicializált Layer struktúrát.
- bool Layer_RenderFont (Layer ∗layer, Window window, TTF_Font ∗font, const char ∗text, SDL_Color foreground)

  Létrehoz egy szöveget tartalmazó Layer-t.
- void Layer_Destroy (Layer ∗layer)

  Felszabadít egy Layer struktúrát.

4.5.1. Részletes leírás

Ez a modul tartalmazza az SDL-lel kapcsolatos függvényeket, és a hozzájuk tartozó adatszerkezeteket.

4.5.1.1. Struktúrák:

4 struktúra szerepel benne:

- `Window`: Az ablakot és a renderert tartalmazza, valamint betöltéskor a méreteket is eltárolja.
- `Assets`: Az assets mappa tartalmát tárolja. (Csak erre a projektre működik, egyébként meg kell változtatni)
- `Fonts`: A betűtípusokat tárolja. (Jelenleg csak a bold van használatban)
- `Layer`: Egy renderelési réteg. A renderelt kép tartalmának csoportosítására szolgál, hogy ne kelljen mindent külön-külön kirajzolni.

A struktúrák használata többnyire az alábbi módon történik:

```
Struktúra név = Struktúra_Init(); // Az init egy üres elemet ad vissza, nem foglal magának memóriát
Struktúra_LétrehozóFüggvény(&struktúra, paraméterek); // Lásd lentebb

// Használat

Struktúra_Destroy(&struktúra); // Mindegyik struktúra tartalmaz dinamikusan foglalt elemet, ezért ezeket fel
      kell szabadítani
```

4.5.1.2. Függvények:

A modul tartalmazza a struktúrákhoz kapcsolódó függvényeket. Ezek mellett tartalmazza az SDL inicializációjához és leállításához szükséges függvényeket.

Az SDL modult a main() függvényben kell inicializálni, majd használat után lezárni.

4.5.1.3. Példakód:

```
#include "SDL_local.h"

int main(int argc, char * argv[]) {
    (void)argc, (void)argv; // Csak a fordító ne kapjon warningot, az SDL-nek kell ez a két paraméter
    InitializeSDL(); // SDL Inicializálása

    Window ablak = Window_Init(); // Ablak inicializálása
    Window_Create(&ablak, "Példa", 800, 600); // Ablak létrehozása
    Assets assets = Assets_Init(); // Assets inicializálása
    Assets_Load(&assets, ablak); // Assets betöltése
    Fonts fonts = Fonts_Init(); // Betűtípusok inicializálása
    Fonts_Load(&fonts); // Betűtípusok betöltése

    Layer background = Layer_Init(); // Háttér réteg inicializálása
    Layer_Create(&background, ablak, NULL); // Háttér réteg létrehozása
    SDL_SetRenderTarget(ablak.renderer, background.layer); // A háttér rétegét változtatjuk
    SDL_RenderCopy(ablak.renderer, assets.background, NULL, NULL); // Háttérkép renderelése

    Layer text = Layer_Init(); // Szöveg réteg inicializálása
    Layer_Create(&text, ablak, (SDL_Rect){200, 200, 100, 400}); // Szöveg réteg létrehozása
    Layer_RenderFont(&text, ablak, fonts.bold, "Hello World!", (SDL_Color){255, 255, 255, 255}); // Szöveg
      létrehozása
    SDL_RenderCopy(ablak.renderer, text.layer, NULL, &text.location); // Szöveg renderelése
    Layer_Destroy(&text); // Használat utáni felszabadítás

    SDL_SetRenderTarget(ablak.renderer, NULL); // Renderer target visszaállítása
    SDL_RenderCopy(ablak.renderer, background.layer, NULL, NULL); // A háttér réteg renderelése az ablakra
    SDL_RenderPresent(ablak.renderer); // Renderelés megjelenítése
    SDL_Delay(2000); // 2 másodpercig megjelenítés

    // ...
```

```
    // Használat utáni felszabadítás és kilépés
    Layer_Destroy(&background);
    Fonts_Destroy(&fonts);
    Assets_Destroy(&assets);
    Window_Destroy(&ablak);
    QuitSDL();
    return 0;
}
```

A fenti példakód megnyit egy ablakot 2 másodpercig, betölti a háttérképet, majd ráírja a "Hello world!"-öt.

4.5.1.4. Megjegyzések:

- Minden innen származó struktúrát használat után fel kell szabadítani.
- A Layer, mivel SDL_Texture∗-t tárol, kompatibilis az SDL függvényekkel.
- A Layer-ek az SDL_SetRenderTarget() használatához lett kitalálva, ennek a "layer" elemét kell átadni, hogy változtatgassuk bármilyen SDL_Render-es fügvénnyel. Ha az ablakra akarunk renderelni, mindenképpen vissza kell váltani az SDL_SetRenderTarget(renderer, NULL) hívásával.
- A szövegek kiírása lehetséges, viszont "mókolós". Itt egy tetszőleges szélesség lett megadva a text-hez, viszont ez teljesen elronthatja az arányokat. Sajnos trial and error-ral lehet csak megadni egy-egy szöveg méreteit.
- A zenelejátszást nem foglaltam bele a példába, ez már csak egy "bónusz" a modulban, ez is lehetséges.

4.5.1.5. Kapcsolódó kódfájlok:

- SDL_local.c
- window.c
- layer.c
- assets.c
- fonts.c

Előfeltétel

Az InitializeSDL() használata szükséges a modul használatához.

Utófeltétel

A QuitSDL() használata szükséges a modul használata után.

Hiba Az SDL használata miatt memóriaszivárgások előfordulhatnak.

Hiba Ha a konzolra valamilyen libGL error kerül akkor textúra hibák léphetnek fel. Nem tudom, hogy mi okkozza ezt, de csak grafikus hibát okoz.

4.5.2. Adatszerkezetek dokumentációja

4.5.2.1. struct Window

Window struktúra.

Az ablakot és a renderert tartalmazza, valamint betöltéskor a méreteket is eltárolja.

Adatmezők

int	height	Ablakmagasság (pixelben)
SDL_Renderer *	renderer	A renderer pointere.
int	width	Ablakszélesség (pixelben)
SDL_Window *	window	Az ablak pointere.

4.5.2.2. struct Assets

Assets mappa fájlaihoz struktúra.

Az assets mappa tartalmát megfelelő változókban tárolja. (Csak erre a projektre működik, egyébként meg kell változtatni.) Maga az Assets kevés helyen használt, inkább csoportosításra szolgál.

Adatmezők

SDL_Texture *	background	A háttérkép textúrája.
Mix_Music *	bgm	A háttérzene pointere.
SDL_Texture *	pipes_empty	A csövek textúrája.
SDL_Texture *	pipes_full	Szintén csövek textúrája, de egy TextureMod és egy AlphaMod van rárakva, hogy "vízzel teli" hatást adjon neki.
SDL_Texture *	wheel	A csap textúrája.

4.5.2.3. struct Fonts

Betűtípusokhoz struktúra.

A betűtípusokat tárolja. (Jelenleg csak a bold van használatban.) Szövegeket a Layer_RenderFont()-tal lehet létrehozni, ahol a Fonts struktúra egyik elemét lehet átadni.

Adatmezők

TTF_Font *	bold	A félkövér betűtípus pointere. (Jelenleg csak ez van használatban)
TTF_Font *	bold_italic	A félkövér dőlt betűtípus pointere.
TTF_Font *	italic	A dőlt betűtípus pointere.
TTF_Font *	normal	A normál betűtípus pointere.

4.5.2.4. struct Layer

Réteg struktúra.

Egy renderelési réteg. A renderelt elemek csoportosítására szolgál, hogy ne kelljen mindent külön-külön kirajzolni. A layer eleme SDL_SetRenderTarget()-tel kiválasztható, ezután az összes renderelés erre hajtódik végre. A location a szülejében elfoglalt helyét és méretét tárolja. Szülőnek a létrehozáskor használt render target-et kell tekinteni.

Adatmezők

SDL_Texture *	layer	A réteg textúrája. SDL_SetRenderTarget() által változtatható.
SDL_Rect	location	A réteg helye és mérete a "szülejében".

4.5.3. Makródefiníciók dokumentációja

4.5.3.1. DEFAULT_FONT_SIZE

```
#define DEFAULT_FONT_SIZE 200
```

Alapértelmezett betűméret.

Az alapértelmezett "betűméret". A jelenlegi szöveghasználat mellett inkább a betűk minősége állítható vele. Ezen az értéken megfelelő minőségűek.

4.5.3.2. MIX_VOLUME

```
#define MIX_VOLUME 70
```

Alapértelmezett hangerő

A zene hangereje (0-128). A játékon belül nem állítható, viszont, hogy ne legyen túl hangos a zene, muszáj volt lentebb állítani a hangerejét. Tetszés szerint lehet állítani.

4.5.4. Függvények dokumentációja

4.5.4.1. Assets_Destroy()

```
void Assets_Destroy (
            Assets * assets )
```

Felszabadít egy Assets struktúrát.

Az átadott Assets_Load() által létrehozott struktúrát szabadítja fel.

Paraméterek

assets	A felszabadítandó Assets struktúra pointere.

4.5.4.2. Assets_Init()

```
Assets Assets_Init (
            void  )
```

Assets struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Assets struktúra.

4.5.4.3. Assets_Load()

```
bool Assets_Load (
            Assets * assets,
            Window window )
```

Feltölt egy inicializált Assets struktúrát.

Betölti az assets mappa tartalmát a megfelelő változókba. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

Az Assets_Load() által létrehozott Assets struktúrákat használat után az Assets_Destroy()-al kell felszabadítani.

Figyelmeztetés

Az Assets struktúrát csak egyszer kell létrehozni, de ha valamiért mégegyszer kell az Assets_Load(), TILOS egy, már létrehozott, Assets struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

assets	Egy inicializált Assets struktúra pointere.
window	Az renderert tartalmazó Window struktúra.

Visszatérési érték

true, hiba esetén false

4.5.4.4. Fonts_Destroy()

```
void Fonts_Destroy (
            Fonts * fonts )
```

Felszabadít egy Fonts struktúrát.

Az átadott Fonts_Load() által létrehozott struktúrát szabadítja fel.

Paraméterek

fonts	A felszabadítandó Fonts struktúra pointere.

4.5.4.5. Fonts_Init()

```
Fonts Fonts_Init (
            void  )
```

Fonts struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Fonts struktúra.

4.5.4.6. Fonts_Load()

```
bool Fonts_Load (
            Fonts * fonts )
```

Feltölt egy inicializált Fonts struktúrát.

Betölti a betűtípusokat az assets/fonts mappából a megfelelő változókba. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát. A betöltött betűtípusokat a Layer_RenderFont() függvénnyel lehet használni.

Utófeltétel

Az Fonts_Load() által létrehozott Fonts struktúrákat használat után az Fonts_Destroy()-al kell felszabadítani.

Figyelmeztetés

A Fonts struktúrát csak egyszer kell létrohozni, de ha valamiért mégegyszer kell az Fonts_Load(), TILOS egy, már létrehozott, Fonts struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

fonts	Egy inicializált Fonts struktúra pointere.

Visszatérési érték

true, hiba esetén false

4.5.4.7. InitializeSDL()

```
bool InitializeSDL (
            void  )
```

Az SDL inicializáló függvénye.

Az SDL-t és a hozzá kapcsolódó modulokat inicializálja. Ha használni akarjuk az SDL-t, akkor ezt a függvényt meg kell hívni. Az előforduló hibákat a függvény kiírja a konzolra.

Utófeltétel

A használat után a QuitSDL() függvényt kell meghívni, hogy leállítsuk a különböző modulokat.

Figyelmeztetés

Többszöri hívás kilépés nélkül összeomlást okozhat.

Visszatérési érték

true, hiba esetén false

4.5.4.8. Layer_Create()

```
bool Layer_Create (
            Layer * layer,
            Window window,
            const SDL_Rect * location )
```

Feltölt egy inicializált Layer struktúrát.

Létrehoz egy SDL_Texture∗-t SDL_TEXTUREACCESS_TARGET-tel, majd a Layer struktúra layer elemében eltárolja ezt. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

A Layer_Create() által létrehozott Layer struktúrákat használat után a Layer_Destroy()-al kell felszabadítani.

Figyelmeztetés

TILOS egy, már létrehozott, Layer struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

layer	Egy inicializált Layer struktúra pointere.
window	A renderert tartalmazó Window struktúra.
location	A réteg helye és mérete a "szülőjéhez" relatívan. Ha NULL, akkor a Window méretét veszi fel!

Visszatérési érték

true, hiba esetén false

4.5.4.9. Layer_Destroy()

```
void Layer_Destroy (
            Layer * layer )
```

Felszabadít egy Layer struktúrát.

Az átadott Layer_Create() által létrehozott struktúrát szabadítja fel.

Paraméterek

layer	A felszabadítandó Layer struktúra pointere.

4.5.4.10. Layer_Init()

```
Layer Layer_Init (
            void  )
```

Layer struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Layer struktúra.

4.5.4.11. Layer_RenderFont()

```
bool Layer_RenderFont (
            Layer * layer,
            Window window,
            TTF_Font * font,
            const char * text,
            SDL_Color foreground )
```

Létrehoz egy szöveget tartalmazó Layer-t.

Egy SDL_Surface∗-en létrehoz egy szöveget, majd SDL_CreateTextureFromSurface()-el átalakítja SDL_Texture∗-ré. Egy Layer_Create() által létrehozott Layer-t kell átadni, viszont valójában ebből csak a location kell. Emiatt technikailag bármilyen Layer-re lehet használni, de a Layer_Create()-el létrehozott Layer-eket célszerű használni. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

Használat után a Layer_Destroy()-al kell felszabadítani a léterhozott struktúrát.

Figyelmeztetés

A függvény nagyon egyszerű, ezért a szöveg szélességét nem számítja ki, ezért előre meg kell adni. Ebből adódik, hogy próbálgatással lehet csak viszonylag megfelelő arányokat beállítani.

Paraméterek

layer	Egy Layer_Create()-el létrehozott Layer struktúra pointere.
window	A renderert tartalmazó Window struktúra.
font	A használni kívánt betűtípus.
text	A renderelendő szöveg.
foreground	A betűk színe.

Visszatérési érték

true, hiba esetén false

4.5.4.12. QuitSDL()

```
void QuitSDL (
            void  )
```

Leállítja az SDL-lel kapcsolatos modulokat.

Ha az InitializeSDL() meg volt hívva, ezt is meg kell hívni a kilépés előtt.

4.5.4.13. StartMusic()

```
void StartMusic (
            Assets assets )
```

Elindítja a háttérzenét és végtelenül ismétli.

SDL_mixer függvényekkel indítja el a háttérzenét. Ezután az SDL_mixer függvényekkel kezelhető az Assets bgm eleme. Az előforduló hibákat a függvény kiírja a konzolra.

Utófeltétel

Bár a QuitSDL() leállítja a zenét, mégis érdemes manuálisan leállítani, kilépés előtt.

Paraméterek

assets	Az assets mappa tartalmát tároló struktúra. A bgm eleme kell.

4.5.4.14. Window_Create()

```
bool Window_Create (
            Window * window,
            const char * title,
            int width,
            int height )
```

Feltölt egy inicializált Window struktúrát.

Létrehoz egy SDL_Window∗-ot és egy SDL_Renderer∗-t, majd a window paraméterben eltárolja ezeket. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

A Window_Create() által létrehozott Window struktúrákat használat után a Window_Destroy()-al kell felszabadítani.

Figyelmeztetés

TILOS egy, már létrehozott, Window struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

window	Egy inicializált Window struktúra pointere.
title	Az ablak neve
width	Az ablak szélessége (pixelben)
height	Az ablak magassága (pixelben)

Visszatérési érték

true, hiba esetén false

4.5.4.15. Window_Destroy()

```
void Window_Destroy (
              Window * window )
```

Felszabadít egy Window struktúrát.

Az átadott Window_Create() által létrehozott struktúrát szabadítja fel.

Paraméterek

window	A felszabadítandó Window struktúra pointere.

4.5.4.16. Window_Init()

```
Window Window_Init (
              void  )
```

Window struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Window struktúra.

4.6. SDL_local.h

Ugrás a fájl dokumentációjához.

```
00001
00106 /* SDL_local.h eleje */
00107 #ifndef SDL_LOCAL_H
00108 #define SDL_LOCAL_H
00109
00110 #include <SDL.h>
00111 #include <SDL_image.h>
00112 #include <SDL_ttf.h>
00113 #include <SDL_mixer.h>
00114 #include <SDL2_gfxPrimitives.h>
00115 #include <stdbool.h>
00116
00117
00123 #define DEFAULT_FONT_SIZE 200
```

```
00129 #define MIX_VOLUME 70
00130
00136 typedef struct Window {
00140     SDL_Window * window;
00144     SDL_Renderer * renderer;
00153     int width, height;
00154 } Window;
00155
00162 typedef struct Assets {
00166     SDL_Texture * background;
00170     SDL_Texture * pipes_empty;
00174     SDL_Texture * pipes_full;
00178     SDL_Texture * wheel;
00182     Mix_Music * bgm;
00183 } Assets;
00184
00191 typedef struct Fonts {
00195     TTF_Font * normal;
00199     TTF_Font * italic;
00203     TTF_Font * bold;
00207     TTF_Font * bold_italic;
00208 } Fonts;
00209
00218 typedef struct Layer {
00222     SDL_Texture * layer;
00226     SDL_Rect location;
00227 } Layer;
00228
00238 bool InitializeSDL(void);
00247 void StartMusic(Assets assets);
00252 void QuitSDL(void);
00253
00258 Window Window_Init(void);
00272 bool Window_Create(Window * window, const char * title, int width, int height);
00278 void Window_Destroy(Window * window);
00279
00284 Assets Assets_Init(void);
00297 bool Assets_Load(Assets * assets, Window window);
00303 void Assets_Destroy(Assets * assets);
00304
00309 Fonts Fonts_Init(void);
00322 bool Fonts_Load(Fonts * fonts);
00328 void Fonts_Destroy(Fonts * fonts);
00329
00334 Layer Layer_Init(void);
00347 bool Layer_Create(Layer * layer, Window window, const SDL_Rect * location);
00365 bool Layer_RenderFont(Layer * layer, Window window, TTF_Font * font, const char * text, SDL_Color
      foreground);
00371 void Layer_Destroy(Layer * layer);
00372
00373 #endif
00374 /* SDL_local.h vége */
```

4.7. lib/src/animation.c fájlreferencia

A game.h-ban deklarált AnimationQueue struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "game.h"
#include "debugmalloc.h"
```

Függvények

- AnimationQueue ∗ AnimationQueue_Init (void)

  Létrehozza egy AnimationQueue strázsáját.
- bool AnimationQueue_AppendWheelRotate (AnimationQueue ∗start, Wheel ∗wheel, double target_angle, double animation_speed)

  Hozzáfűz az AnimationQueue végéhez egy Wheel forgatási animációját.
- bool AnimationQueue_AppendPipeRotate (AnimationQueue ∗start, Pipe ∗pipe, double target_angle, double animation_speed)

  Hozzáfűz az AnimationQueue végéhez egy Pipe forgatási animációját.

- bool AnimationQueue_AppendLongFill (AnimationQueue ∗start, SDL_Rect render_location, SDL_Texture ∗filled_texture, double animation_speed, AnimationDirection direction)

  Hozzáfűz az AnimationQueue végéhez egy hosszú cső feltöltési animációját.
- bool AnimationQueue_AppendPipeFill (AnimationQueue ∗start, SDL_Renderer ∗renderer, Pipe ∗pipe, SDL_Texture ∗filled_texture, double animation_speed, AnimationDirection direction)

  Hozzáfűz az AnimationQueue végéhez egy Pipe feltöltési animációját.
- AnimationQueue ∗ AnimationQueue_RemoveElement (AnimationQueue ∗start, AnimationQueue ∗element)

  A megadott elemet kitörli az AniamtionQueue-ból és visszaadja az előtte lévő pointerét.
- void AnimationQueue_Destroy (AnimationQueue ∗start)

  Felszabadít egy AnimationQueue láncolt listát.

4.7.1. Részletes leírás

A game.h-ban deklarált AnimationQueue struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.7.2. Függvények dokumentációja

4.7.2.1. AnimationQueue_AppendLongFill()

```
bool AnimationQueue_AppendLongFill (
            AnimationQueue * start,
            SDL_Rect render_location,
            SDL_Texture * filled_texture,
            double animation_speed,
            AnimationDirection direction )
```

Hozzáfűz az AnimationQueue végéhez egy hosszú cső feltöltési animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. A hosszú cső lehet más SDL_Texture∗ is, de a game.c-ben csak ehhez van használva. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
render_location	A renderelendő hosszú cső helye
filled_texture	A hosszú cső textúrája
animation_speed	Az animáció sebessége
direction	Az animáció iránya

Visszatérési érték

true, hiba esetén false

4.7.2.2. AnimationQueue_AppendPipeFill()

```
bool AnimationQueue_AppendPipeFill (
            AnimationQueue * start,
```

```
            SDL_Renderer * renderer,
            Pipe * pipe,
            SDL_Texture * filled_texture,
            double animation_speed,
            AnimationDirection direction )
```

Hozzáfűz az AnimationQueue végéhez egy Pipe feltöltési animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Hiba esetén megpróbálja a töltést kihagyni. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
renderer	Az ablak rendererje (csak hiba esetén használt)
pipe	A feltöltendő Pipe pointere
filled_texture	A Pipe feltöltött textúrája
animation_speed	Az animáció sebessége
direction	Az animáció iránya

Visszatérési érték

true, hiba esetén false

4.7.2.3. AnimationQueue_AppendPipeRotate()

```
bool AnimationQueue_AppendPipeRotate (
            AnimationQueue * start,
            Pipe * pipe,
            double target_angle,
            double animation_speed )
```

Hozzáfűz az AnimationQueue végéhez egy Pipe forgatási animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
wheel	A forgatandó Pipe pointere
target_angle	A forgatás mértéke (fokban)
animation_speed	Az animáció sebessége

Visszatérési érték

true, hiba esetén false

4.7.2.4. AnimationQueue_AppendWheelRotate()

```
bool AnimationQueue_AppendWheelRotate (
```

```
            AnimationQueue * start,
            Wheel * wheel,
            double target_angle,
            double animation_speed )
```

Hozzáfűz az AnimationQueue végéhez egy Wheel forgatási animációját.

A szükséges változókat beállítja és hozzáfűzi az új elemet az AnimationQueue végéhez. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

start	Az AnimationQueue strázsájának pointere
wheel	A forgatandó Wheel pointere
target_angle	A forgatás mértéke (fokban)
animation_speed	Az animáció sebessége

Visszatérési érték

true, hiba esetén false

4.7.2.5. AnimationQueue_Destroy()

```
void AnimationQueue_Destroy (
            AnimationQueue * start )
```

Felszabadít egy AnimationQueue láncolt listát.

A strázsáját is felszabadítja. Használat után KELL használni.

Paraméterek

start	Az AnimationQueue strázsájának pointere

4.7.2.6. AnimationQueue_Init()

```
AnimationQueue * AnimationQueue_Init (
            void  )
```

Létrehozza egy AnimationQueue strázsáját.

malloc()-kal foglal egy AnimationQueue struktúrát és kinullázza. Ehhez a különböző AnimationQueue_Append függvényekkel lehet hozzáfűzni elemeket.

Utófeltétel

Használat után a AnimationQueue_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Visszatérési érték

Egy üres AnimationQueue struktúra

4.7.2.7. AnimationQueue_RemoveElement()

```
AnimationQueue * AnimationQueue_RemoveElement (
            AnimationQueue * start,
            AnimationQueue * element )
```

A megadott elemet kitörli az AniamtionQueue-ból és visszaadja az előtte lévő pointerét.

Figyelmeztetés

TILOS a strázsát átadni.

Paraméterek

start	Az AnimationQueue strázsájának pointere
element	A törlendő elem pointere

Visszatérési érték

A törlendő elem előtti elem pointere

4.8. lib/src/assets.c fájlreferencia

Az SDL_local.h-ban deklarált Assets struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "SDL_local.h"
#include "debugmalloc.h"
```

Függvények

- Assets Assets_Init (void)
  Assets struktúra inicializálása (kinullázása)
- bool Assets_Load (Assets *assets, Window window)
  Feltölt egy inicializált Assets struktúrát.
- void Assets_Destroy (Assets *assets)
  Felszabadít egy Assets struktúrát.

4.8.1. Részletes leírás

Az SDL_local.h-ban deklarált Assets struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.8.2. Függvények dokumentációja

4.8.2.1. Assets_Destroy()

```
void Assets_Destroy (
            Assets * assets )
```

Felszabadít egy Assets struktúrát.

Az átadott Assets_Load() által létrehozott struktúrát szabadítja fel.

Paraméterek

assets	A felszabadítandó Assets struktúra pointere.

4.8.2.2. Assets_Init()

```
Assets Assets_Init (
            void  )
```

Assets struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Assets struktúra.

4.8.2.3. Assets_Load()

```
bool Assets_Load (
            Assets * assets,
            Window window )
```

Feltölt egy inicializált Assets struktúrát.

Betölti az assets mappa tartalmát a megfelelő változókba. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

Az Assets_Load() által létrehozott Assets struktúrákat használat után az Assets_Destroy()-al kell felszabadítani.

Figyelmeztetés

Az Assets struktúrát csak egyszer kell létrehozni, de ha valamiért mégegyszer kell az Assets_Load(), TILOS egy, már létrehozott, Assets struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

assets	Egy inicializált Assets struktúra pointere.
window	Az renderert tartalmazó Window struktúra.

Visszatérési érték

true, hiba esetén false

4.9. lib/src/elem.c fájlreferencia

A pipe.h-ban deklarált Elem struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "pipe.h"
#include "debugmalloc.h"
```

Függvények

- void Elem_Init (Elem elem)

  Egy üres (csupa Levego-s) Elem-et álltít elő.
- void Elem_Copy (Elem hova, Elem honnan)

  Átmásol egy Elem tartalmát egy másikba.
- void Elem_RotatePos (Elem elem)

  Elforgat egy Elem-et az óramutató járásával megegyező irányba.
- void Elem_RotateNeg (Elem elem)

  Elforgat egy Elem-et az óramutató járásával ellentétes irányba.

4.9.1. Részletes leírás

A pipe.h-ban deklarált Elem struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.9.2. Függvények dokumentációja

4.9.2.1. Elem_Copy()

```
void Elem_Copy (
            Elem hova,
            Elem honnan )
```

Átmásol egy Elem tartalmát egy másikba.

A honnan Elem tartalmát másolja át a hova Elem-be. A hova Elem tartalmát felülírja. A honnan elem tartalmát nem változtatja, viszont a fordító warningot ad, ha const a paraméter.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

hova	A cél Elem
honnan	A forrás Elem

4.9.2.2. Elem_Init()

```
void Elem_Init (
            Elem elem )
```

Egy üres (csupa Levego-s) Elem-et álltít elő.

Az átadott Elem-et feltölti Levego-vel. A C nyelv sajátossága miatt tér el a többi Init függvénytől.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	Az inicializálandó Elem

4.9.2.3. Elem_RotateNeg()

```
void Elem_RotateNeg (
            Elem elem )
```

Elforgat egy Elem-et az óramutató járásával ellentétes irányba.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	A változtatandó Elem

4.9.2.4. Elem_RotatePos()

```
void Elem_RotatePos (
            Elem elem )
```

Elforgat egy Elem-et az óramutató járásával megegyező irányba.

Megjegyzés

Ne használd. Csak azért van a header-ben, mert máshol van definiálva, mint ahol használva.

Paraméterek

elem	A változtatandó Elem

4.10. lib/src/ev_vars.c fájlreferencia

A game.h-ban deklarált EventloopVariables struktúrához és az eventloophoz kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "game.h"
#include "debugmalloc.h"
```

Függvények

- EventloopVariables EventloopVariables_Init (void)

  Az eventloop változóinak inicializálása (kinullázása)
- void Delta_Update (EventloopVariables ∗ev_vars)

  Frissíti a deltaT elemének értékét.
- bool CursorInRect (EventloopVariables ev_vars, SDL_Rect bound)

  Megvizsgálja, hogy a kurzor a megadott SDL_Rect-en belül van-e.

4.10.1. Részletes leírás

A game.h-ban deklarált EventloopVariables struktúrához és az eventloophoz kapcsolódó függvények kódjait tartalmazza a fájl.

4.10.2. Függvények dokumentációja

4.10.2.1. CursorInRect()

```
bool CursorInRect (
            EventloopVariables ev_vars,
            SDL_Rect bound )
```

Megvizsgálja, hogy a kurzor a megadott SDL_Rect-en belül van-e.

Egy elemre történő kattintás vizsgálatára alkalmas. Az SDL_MOUSEBUTTONDOWN és az SDL_↩ MOUSEBUTTONUP közötti változást kiszűri, azaz csak akkor ad vissza true-t, ha a kurzor az SDL_Rect-en belül volt mindkét eseménykor.

Paraméterek

ev_vars	Egy EventloopVariables struktúra.
bound	A megvizsgálandó SDL_Rect.

Visszatérési érték

true, ha a kurzor az SDL_Rect-en belül volt SDL_MOUSEBUTTONDOWN és SDL_MOUSEBUTTONUP-kor, egyébként false

4.10.2.2. Delta_Update()

```
void Delta_Update (
            EventloopVariables * ev_vars )
```

Frissíti a deltaT elemének értékét.

Két hívás között eltelt időt számítja ki az SDL_GetPerformanceCounter() és az SDL_GetPerformanceFrequency() segítségével. Az animációk időzítésére szolgál.

Paraméterek

ev_vars	A változtatandó EventloopVariables struktúra pointere.

4.10.2.3. EventloopVariables_Init()

```
EventloopVariables EventloopVariables_Init (
            void  )
```

Az eventloop változóinak inicializálása (kinullázása)

Az visszaadot struktúra minden eleme 0 értéket kap. Mivel nincs dinamikusan foglalt változója, ezért nem kell felszabadítani.

Visszatérési érték

Egy EventloopVariables struktúra

4.11. lib/src/fonts.c fájlreferencia

Az SDL_local.h-ban deklarált Fonts struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "SDL_local.h"
#include "debugmalloc.h"
```

Függvények

- Fonts Fonts_Init (void)
  Fonts struktúra inicializálása (kinullázása)
- bool Fonts_Load (Fonts *fonts)
  Feltölt egy inicializált Fonts struktúrát.
- void Fonts_Destroy (Fonts *fonts)
  Felszabadít egy Fonts struktúrát.

4.11.1. Részletes leírás

Az SDL_local.h-ban deklarált Fonts struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.11.2. Függvények dokumentációja

4.11.2.1. Fonts_Destroy()

```
void Fonts_Destroy (
            Fonts * fonts )
```

Felszabadít egy Fonts struktúrát.

Az átadott Fonts_Load() által létrehozott struktúrát szabadítja fel.

Paraméterek

fonts	A felszabadítandó Fonts struktúra pointere.

4.11.2.2. Fonts_Init()

```
Fonts Fonts_Init (
            void  )
```

Fonts struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Fonts struktúra.

4.11.2.3. Fonts_Load()

```
bool Fonts_Load (
            Fonts * fonts )
```

Feltölt egy inicializált Fonts struktúrát.

Betölti a betűtípusokat az assets/fonts mappából a megfelelő változókba. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát. A betöltött betűtípusokat a Layer_RenderFont() függvénnyel lehet használni.

Utófeltétel

Az Fonts_Load() által létrehozott Fonts struktúrákat használat után az Fonts_Destroy()-al kell felszabadítani.

Figyelmeztetés

A Fonts struktúrát csak egyszer kell létrohozni, de ha valamiért mégegyszer kell az Fonts_Load(), TILOS egy, már létrehozott, Fonts struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

fonts	Egy inicializált Fonts struktúra pointere.

Visszatérési érték

true, hiba esetén false

4.12. lib/src/game.c fájlreferencia

A game.h-ban deklarált, a játék állapotához tartozó függvények kódjait tartalmazó fájl.

```
#include "game.h"
#include <errno.h>
#include "debugmalloc.h"
```

Függvények

- bool Menu (GameStruct game_struct)

  A játékmenü függvénye.
- bool Game (GameStruct game_struct, PipeGrid *betoltott, int entry_y, int exit_y)

  A játék függvénye.
- bool Win (GameStruct game_struct, Layer state, PipeGrid copy, int entry_y, int exit_y)

  A nyert kép függvénye.

4.12.1. Részletes leírás

A game.h-ban deklarált, a játék állapotához tartozó függvények kódjait tartalmazó fájl.

Hiba Az osdialog modullal problémája van a debugmalloc.h-nak. Az osdialog dokumentációja tisztán leírja, hogy a visszaadott stringeit fel kell szabadítani használat után. Viszont, ha fel akarnám szabadítani ezt, a debugmalloc.h hibát ad vissza. Ezt úgy lehet kijavítani, hogy az osdialog.h-ban include-oljuk a debugmalloc.h-t vagy innen kivesszük a debugmalloc.h-t.

4.12.2. Függvények dokumentációja

4.12.2.1. Game()

```
bool Game (
            GameStruct game_struct,
            PipeGrid * betoltott,
            int entry_y,
            int exit_y )
```

A játék függvénye.

Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Megjegyzés

A Menu() függvény már képes meghívni, magában ne hívjuk meg, mert nem garantált a működése.

Paraméterek

game_struct	A létrehozott GameStruct.
betoltott	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid.
entry_y	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid bejáratánk sora.
exit_y	Ha betöltés volt a Menu()-ben választva, akkor a betöltött PipeGrid kijáratának sora.

Visszatérési érték

true, ha a Menu()-be lépünk vissza, false, ha új játék kezdődik, kilépés történt, illetve néhány súlyosabb hiba esetén

4.12.2.2. Menu()

```
bool Menu (
            GameStruct game_struct )
```

A játékmenü függvénye.

Az SDL és a GameStruct inicializálása/létrehozása után meg lehet hívni. Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Paraméterek

game_struct	A létrehozott GameStruct.

Visszatérési érték

true, súlyos hiba esetén false

4.12.2.3. Win()

```
bool Win (
            GameStruct game_struct,
            Layer state,
            PipeGrid copy,
            int entry_y,
            int exit_y )
```

A nyert kép függvénye.

Az előforduló hibákat a függvény kiírja a konzolra. Súlyosabb hiba esetén a függvény false értékkel leáll.

Megjegyzés

A Game() függvény már képes meghívni, magában ne hívjuk meg, mert nem garantált a működése.

Paraméterek

game_struct	A létrehozott GameStruct.
state	A Game() függvény aktuális renderelt képe.
copy	A PipeGrid pálya betöltéskori másolata.
entry_y	A PipeGrid bejáratának sora.
exit_y	A PipeGrid kijáratának sora.

Visszatérési érték

true, ha a Menu()-be lépünk vissza, false, ha új játék kezdődik, kilépés történt, illetve néhány súlyosabb hiba esetén

4.13. lib/src/game_struct.c fájlreferencia

A game.h-ban deklarált GameStruct struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "game.h"
#include "debugmalloc.h"
```

Függvények

- GameStruct GameStruct_Init (void)

  GameStruct inicializálása (kinullázása)
- bool GameStruct_Create (GameStruct *game_struct)

  GameStruct létrehozása.
- void GameStruct_Destroy (GameStruct *game_struct)

  Felszabadít egy GameStruct-ot.

4.13.1. Részletes leírás

A game.h-ban deklarált GameStruct struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.13.2. Függvények dokumentációja

4.13.2.1. GameStruct_Create()

```
bool GameStruct_Create (
            GameStruct * game_struct )
```

GameStruct létrehozása.

Meghívja a GameStruct elemeinek létrehozó függvényeit (lásd: SDL_local.h). A Window létrehozásához a DEFAULT_WINDOW_WIDTH és DEFAULT_WINDOW_HEIGHT konstansokat használja. Átadni egy GameStruct_Init()-tel inicializált GameStruct-ot ajánlott. A hibákat a struktok létrehozó függvényei a konzolra kiírják, ennek csak a visszatérési értéke jelzi. Hiba esetén nem ajánlott a struktúra használata.

Utófeltétel

Használat után a GameStruct_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Figyelmeztetés

TILOS egy már létrehozott GameStruct-ot átadni.

Paraméterek

game_struct	Egy inicializált GameStruct pointere.

Visszatérési érték

true, hiba esetén false

4.13.2.2. GameStruct_Destroy()

```
void GameStruct_Destroy (
            GameStruct * game_struct )
```

Felszabadít egy GameStruct-ot.

A GameStruct elemeit szabadítja fel a saját függvényeikkel. (lásd: SDL_local.h)

Paraméterek

game_struct	A felszabadítandó GameStruct pointere.

4.13.2.3. GameStruct_Init()

```
GameStruct GameStruct_Init (
            void  )
```

GameStruct inicializálása (kinullázása)

Visszatérési érték

Egy üres GameStruct

4.14. lib/src/layer.c fájlreferencia

Az SDL_local.h-ban deklarált Layer struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "SDL_local.h"
#include "debugmalloc.h"
```

Függvények

- Layer Layer_Init (void)

  Layer struktúra inicializálása (kinullázása)
- bool Layer_Create (Layer *layer, Window window, const SDL_Rect *location)

  Feltölt egy inicializált Layer struktúrát.
- bool Layer_RenderFont (Layer *layer, Window window, TTF_Font *font, const char *text, SDL_Color foreground)

  Létrehoz egy szöveget tartalmazó Layer-t.
- void Layer_Destroy (Layer *layer)

  Felszabadít egy Layer struktúrát.

4.14.1. Részletes leírás

Az SDL_local.h-ban deklarált Layer struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.14.2. Függvények dokumentációja

4.14.2.1. Layer_Create()

```
bool Layer_Create (
            Layer * layer,
            Window window,
            const SDL_Rect * location )
```

Feltölt egy inicializált Layer struktúrát.

Létrehoz egy SDL_Texture∗-t SDL_TEXTUREACCESS_TARGET-tel, majd a Layer struktúra layer elemében eltárolja ezt. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

A Layer_Create() által létrehozott Layer struktúrákat használat után a Layer_Destroy()-al kell felszabadítani.

Figyelmeztetés

TILOS egy, már létrehozott, Layer struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

layer	Egy inicializált Layer struktúra pointere.
window	A renderert tartalmazó Window struktúra.
location	A réteg helye és mérete a "szülőjéhez" relatívan. Ha NULL, akkor a Window méretét veszi fel!

Visszatérési érték

true, hiba esetén false

4.14.2.2. Layer_Destroy()

```
void Layer_Destroy (
            Layer * layer )
```

Felszabadít egy Layer struktúrát.

Az átadott Layer_Create() által létrehozott struktúrát szabadítja fel.

Paraméterek

layer	A felszabadítandó Layer struktúra pointere.

4.14.2.3. Layer_Init()

```
Layer Layer_Init (
            void  )
```

Layer struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Layer struktúra.

4.14.2.4. Layer_RenderFont()

```
bool Layer_RenderFont (
            Layer * layer,
            Window window,
            TTF_Font * font,
            const char * text,
            SDL_Color foreground )
```

Létrehoz egy szöveget tartalmazó Layer-t.

Egy SDL_Surface∗-en létrehoz egy szöveget, majd SDL_CreateTextureFromSurface()-el átalakítja SDL_Texture∗-ré. Egy Layer_Create() által létrehozott Layer-t kell átadni, viszont valójában ebből csak a location kell. Emiatt technikailag bármilyen Layer-re lehet használni, de a Layer_Create()-el létrehozott Layer-eket célszerű használni. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

Használat után a Layer_Destroy()-al kell felszabadítani a léterhozott struktúrát.

Figyelmeztetés

A függvény nagyon egyszerű, ezért a szöveg szélességét nem számítja ki, ezért előre meg kell adni. Ebből adódik, hogy próbálgatással lehet csak viszonylag megfelelő arányokat beállítani.

Paraméterek

layer	Egy Layer_Create()-el létrehozott Layer struktúra pointere.
window	A renderert tartalmazó Window struktúra.
font	A használni kívánt betűtípus.
text	A renderelendő szöveg.
foreground	A betűk színe.

Visszatérési érték

true, hiba esetén false

4.15. lib/src/pipe.c fájlreferencia

A pipe.h-ban deklarált Pipe struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "pipe.h"
#include "debugmalloc.h"
```

Függvények

- Pipe Pipe_Init (void)

  Pipe struktúra inicializálása (kinullázása/alapértelmezett értékek beállítása)
- bool Pipe_CreateFromElem (Pipe ∗pipe, Window window, SDL_Texture ∗pipe_texture, SDL_Rect location)

  Pipe struktúra létrehozása a benne található Elem alapján.
- void Pipe_Rotate (Pipe ∗pipe, int irany)

  Elforgatja egy Pipe Elem-ét.
- void Pipe_Destroy (Pipe ∗pipe)

  Felszabadít egy Pipe struktúrát.
- bool Pipe_Render (Pipe pipe, Window window)

  Renderel egy Pipe-ot a jelenlegi RenderTarget-re.
- bool Pipe_UpdateLayer (Pipe pipe, Window window, SDL_Texture ∗pipe_texture)

  Frissíti egy Pipe Layer-ét.

4.15.1. Részletes leírás

A pipe.h-ban deklarált Pipe struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.15.2. Függvények dokumentációja

4.15.2.1. Pipe_CreateFromElem()

```
bool Pipe_CreateFromElem (
            Pipe * pipe,
            Window window,
            SDL_Texture * pipe_texture,
            SDL_Rect base_location )
```

Pipe struktúra létrehozása a benne található Elem alapján.

A függvény létrehozza a Pipe Layer-ét, illetve beállítja a szülőjéhez relatív helyét (lásd: Layer), felvesz egy szöget és beállítja a PipeType-ot, illetve a textúrájának helyét. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi. Hiba esetén nem ajánlott a létrejött struktúrát használni.

Utófeltétel

Mivel létrehoz egy Layer-t, használat után a Pipe-ot fel kell szabadítani a Pipe_Destroy() függvénnyel.

Megjegyzés

A projekten belül található Create függvényektől eltérően inkább feldolgoz, mintsem létrehoz. Emiatt is lehetőleg csak a PipeGrid_Feldolgoz() függvényen belül maradjon használva.

Paraméterek

pipe	A feldolgozandó Pipe pointere
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture*
base_location	A Pipe relatív helye

Visszatérési érték

true, hiba esetén false

4.15.2.2. Pipe_Destroy()

```
void Pipe_Destroy (
            Pipe * pipe )
```

Felszabadít egy Pipe struktúrát.

Az átadott Pipe_CreateFromElem() függvénnyel létrehozott Pipe-ot felszabadítja. Lényegében a struktúra Layer-ét szabadítja fel.

Paraméterek

pipe	A felaszabadítandó Pipe struktúra

4.15.2.3. Pipe_Init()

```
Pipe Pipe_Init (
            void  )
```

Pipe struktúra inicializálása (kinullázása/alapértelmezett értékek beállítása)

Visszatérési érték

Egy üres Pipe struktúra

4.15.2.4. Pipe_Render()

```
bool Pipe_Render (
            Pipe pipe,
            Window window )
```

Renderel egy Pipe-ot a jelenlegi RenderTarget-re.

A Pipe Layer-ét rendereli a jelenlegi RenderTarget-re. Fontos, hogy a RenderTarget a "szülője" legyen, azaz az, amihez a Pipe Layer-ének helye relatív. Egyéb esetben nem garantált a helyes működés. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

pipe	A renderelendő Pipe
window	A renderert tartalmazó Window struktúra

Visszatérési érték

true, hiba esetén false

4.15.2.5. Pipe_Rotate()

```
void Pipe_Rotate (
            Pipe * pipe,
            int irany )
```

Elforgatja egy Pipe Elem-ét.

Az iránynak megfelelően elforgatja a Pipe Elem-ét. Ha az irány 1, az óramutató járásával megegyező irányba, ha -1, akkor az óramutató járásával ellentétes irányba. Ha a Pipe típusa PipeType_Crossing, akkor nem történik semmi.

Paraméterek

pipe	A változtatandó Pipe
irany	A forgatás iránya (1: óramutató járásával megegyező, -1: óramutató járásával ellentétes)

4.15.2.6. Pipe_UpdateLayer()

```
bool Pipe_UpdateLayer (
            Pipe pipe,
            Window window,
            SDL_Texture * pipe_texture )
```

Frissíti egy Pipe Layer-ét.

A Pipe Layer-ét frissíti (újrarendereli) az aktuális értékei szerint. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

pipe	A frissítendő Pipe
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture*

Visszatérési érték

true, hiba esetén false

4.16. lib/src/pipegrid.c fájlreferencia

A pipe.h-ban deklarált PipeGrid struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "pipe.h"
#include "debugmalloc.h"
```

Függvények

- PipeGrid PipeGrid_Init (void)

  PipeGrid struktúra inicializálása (kinullázása)
- bool PipeGrid_Create (PipeGrid ∗pipegrid, int x, int y)

  PipeGrid struktúra létrehozása.
- void PipeGrid_Destroy (PipeGrid ∗pipegrid)

  Felszabadít egy PipeGrid-et.
- PipeGrid PipeGrid_CreateCopy (PipeGrid source)

  Lemásol egy PipeGrid-et és visszaadja a másolatot.
- void PipeGrid_GenerateGrid (PipeGrid ∗pipegrid, SDL_Point entry, SDL_Point exit)

  Legenerál egy PipeGrid-et.
- void PipeGrid_Shuffle (PipeGrid pipegrid)

  Összekever egy PipeGrid-et.
- bool PipeGrid_Feldolgoz (PipeGrid pipegrid, Window window, SDL_Texture ∗pipe_texture, SDL_Rect base↩ _location)

  Egy PipeGrid Pipe-jainak hiányzó értékeit feltölti.
- bool PipeGrid_Render (PipeGrid pipegrid, Window window)

  Lerenderel egy feldolgozott PipeGrid-et az aktuális RenderTarget-re.

4.16.1. Részletes leírás

A pipe.h-ban deklarált PipeGrid struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.16.2. Függvények dokumentációja

4.16.2.1. PipeGrid_Create()

```
bool PipeGrid_Create (
            PipeGrid * pipegrid,
            int x,
            int y )
```

PipeGrid struktúra létrehozása.

Létrehoz egy PipeGrid-et a megadott szélességgel és magassággal. A létrejövő mátrix elemeit a Pipe_Init()-tel hozza létre. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi. Hiba esetén a struktúra mátrixa NULL.

Utófeltétel

Mivel létrehoz egy Pipe mátrixot, használat után a PipeGrid-ot fel kell szabadítani a PipeGrid_Destroy() függvénnyel.

Figyelmeztetés

TILOS egy, már létrehozott, PipeGrid struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

pipegrid	A létrehozandó PipeGrid pointere
x	A PipeGrid szélessége
y	A PipeGrid magassága

Visszatérési érték

true, hiba esetén false

4.16.2.2. PipeGrid_CreateCopy()

```
PipeGrid PipeGrid_CreateCopy (
            PipeGrid source )
```

Lemásol egy PipeGrid-et és visszaadja a másolatot.

Csak a Pipe-ok Elem-ét másolja le, ezért ha használni akarjuk a másolatot, akkor először fel kell dolgozni a PipeGrid_Feldolgoz() függvénnyel.

Utófeltétel

A másolatot a PipeGrid_Create()-tel hozza létre, így fel kell szabadítani használat után a PipeGrid_Destroy()-el.

Paraméterek

source	A forrás PipeGrid

Visszatérési érték

Egy új PipeGrid, hiba esetén ennek a mátrixa NULL

4.16.2.3. PipeGrid_Destroy()

```
void PipeGrid_Destroy (
            PipeGrid * pipegrid )
```

Felszabadít egy PipeGrid-et.

A PipeGrid-ben lévő Pipe-okat is felszabadítja.

Paraméterek

pipegrid	A felaszabadítandó PipeGrid pointere

4.16.2.4. PipeGrid_Feldolgoz()

```
bool PipeGrid_Feldolgoz (
            PipeGrid pipegrid,
            Window window,
            SDL_Texture * pipe_texture,
            SDL_Rect base_location )
```

Egy PipeGrid Pipe-jainak hiányzó értékeit feltölti.

A PipeGrid-ben található Pipe-ok Elem-ei alapján feltölti a Pipe-ok hiányzó értékeit a Pipe_CreateFromElem()-el, azaz létrehozza a Layer-ét, beállítja a szülőjéhez relatív helyét, felvesz egy szöget és beállítja a PipeType-ot, illetve a textúrájának helyét. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott a struktúra használata.

Előfeltétel

A Pipe-ok Elem-jeinek már be kell legyen állítva az értéke. Emiatt csak a "végleges" állapotban ajánlott használni, azaz a PipeGrid_GenerateGrid() és a PipeGrid_Shuffle() függvények után.

Figyelmeztetés

TILOS egy, már feldolgozott, PipeGrid struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

pipegrid	A feldolgozandó PipeGrid
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture*
base_location	A PipeGrid megjelenítési helye

Visszatérési érték

true, hiba esetén false

4.16.2.5. PipeGrid_GenerateGrid()

```
void PipeGrid_GenerateGrid (
            PipeGrid * pipegrid,
            SDL_Point entry,
            SDL_Point exit )
```

Legenerál egy PipeGrid-et.

A játékhoz az átadott PipeGrid-ben generál egy pályát. Ez egy rekurzív labirintusgeneráló algoritmussal történik, aminek a segítségével egy véletlenszerű, megoldható pályát generál. Alapul az infoc-n található algoritmus szolgált (link: https://infoc.eet.bme.hu/labirintus/), ami módosítva lett, hogy az általam használt adatszerkezetekkel működjön (Pipe-on belüli Elem).

Figyelmeztetés

A belépési és kilépési pontok szabadon megadhatóak, DE az entry lehetőleg az első oszlopban, az exit pedig az utolsóban legyen. Egyéb esetben nem garantált a helyes működés.

Paraméterek

pipegrid	A változtatandó PipeGrid pointere
entry	A belépési pont (x: oszlopindex, y: sorindex)
exit	A kilépési pont (x: oszlopindex, y: sorindex)

4.16.2.6. PipeGrid_Init()

```
PipeGrid PipeGrid_Init (
            void  )
```

PipeGrid struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres PipeGrid struktúra

4.16.2.7. PipeGrid_Render()

```
bool PipeGrid_Render (
            PipeGrid pipegrid,
            Window window )
```

Lerenderel egy feldolgozott PipeGrid-et az aktuális RenderTarget-re.

Az előforduló hibákat a függvény kiírja a konzolra.

Paraméterek

pipegrid	Egy feldolgozott PipeGrid
window	A renderert tartalmazó Window struktúra

Visszatérési érték

true, hiba esetén false

4.16.2.8. PipeGrid_Shuffle()

```
void PipeGrid_Shuffle (
            PipeGrid pipegrid )
```

Összekever egy PipeGrid-et.

A PipeGrid-en belül lévő Pipe-ok Elem-eit véletlenszerűen elforgatja.

Paraméterek

pipegrid	Az összekeverendő PipeGrid

4.17. lib/src/SDL_local.c fájlreferencia

Az SDL_local.h-ban deklarált, struktúrákhoz nem köthető függvények kódjait tartalmazza a fájl.

```
#include "SDL_local.h"
#include "debugmalloc.h"
```

Függvények

- bool InitializeSDL (void)

  Az SDL inicializáló függvénye.
- void StartMusic (Assets assets)

  Elindítja a háttérzenét és végtelenül ismétli.
- void QuitSDL (void)

  Leállítja az SDL-lel kapcsolatos modulokat.

4.17.1. Részletes leírás

Az SDL_local.h-ban deklarált, struktúrákhoz nem köthető függvények kódjait tartalmazza a fájl.

4.17.2. Függvények dokumentációja

4.17.2.1. InitializeSDL()

```
bool InitializeSDL (
             void  )
```

Az SDL inicializáló függvénye.

Az SDL-t és a hozzá kapcsolódó modulokat inicializálja. Ha használni akarjuk az SDL-t, akkor ezt a függvényt meg kell hívni. Az előforduló hibákat a függvény kiírja a konzolra.

Utófeltétel

A használat után a QuitSDL() függvényt kell meghívni, hogy leállítsuk a különböző modulokat.

Figyelmeztetés

Többszöri hívás kilépés nélkül összeomlást okozhat.

Visszatérési érték

true, hiba esetén false

4.17.2.2. QuitSDL()

```
void QuitSDL (
            void  )
```

Leállítja az SDL-lel kapcsolatos modulokat.

Ha az InitializeSDL() meg volt hívva, ezt is meg kell hívni a kilépés előtt.

4.17.2.3. StartMusic()

```
void StartMusic (
            Assets assets )
```

Elindítja a háttérzenét és végtelenül ismétli.

SDL_mixer függvényekkel indítja el a háttérzenét. Ezután az SDL_mixer függvényekkel kezelhető az Assets bgm eleme. Az előforduló hibákat a függvény kiírja a konzolra.

Utófeltétel

Bár a QuitSDL() leállítja a zenét, mégis érdemes manuálisan leállítani, kilépés előtt.

Paraméterek

assets	Az assets mappa tartalmát tároló struktúra. A bgm eleme kell.

4.18. lib/src/solve.c fájlreferencia

A pipe.h-ban deklarált SolvedPipeGrid struktúrához és a PipeGrid megoldásához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "pipe.h"
#include "debugmalloc.h"
```

Függvények

- SolvedPipeGrid ∗ PipeGrid_Solve (PipeGrid pipegrid, SDL_Point entry, SDL_Point exit)

  Megold egy PipeGrid-et és a megoldását egy SolvedPipeGrid-ben adja vissza.
- SolvedPipeGrid ∗ SolvedPipeGrid_RemoveFirst (SolvedPipeGrid ∗start)

  A SolvedPipeGrid láncolt lista első elemét felszabadítja és visszaadja a következő elem pointerét.
- void SolvedPipeGrid_Destroy (SolvedPipeGrid ∗start)

  Felszabadít egy SolvedPipeGrid láncolt listát.

4.18.1. Részletes leírás

A pipe.h-ban deklarált SolvedPipeGrid struktúrához és a PipeGrid megoldásához kapcsolódó függvények kódjait tartalmazza a fájl.

4.18.2. Függvények dokumentációja

4.18.2.1. PipeGrid_Solve()

```
SolvedPipeGrid * PipeGrid_Solve (
            PipeGrid pipegrid,
            SDL_Point entry,
            SDL_Point exit )
```

Megold egy PipeGrid-et és a megoldását egy SolvedPipeGrid-ben adja vissza.

Át kell adni a PipeGrid_GenerateGrid()-ben használt belépési és kilépési pontokat, egyébként nem fogja jól megoldani.

Utófeltétel

Mivel egy láncolt listát hoz létre a függvény, használat után a SolvedPipeGrid_Destroy()-al kell felszabadítani.

Paraméterek

pipegrid	A megoldandó PipeGrid
entry	A belépési pont (x: oszlopindex, y: sorindex)
exit	A kilépési pont (x: oszlopindex, y: sorindex)

Visszatérési érték

A megoldás láncolt listájának első eleme, hiba vagy megoldás hiánya esetén NULL.

4.18.2.2. SolvedPipeGrid_Destroy()

```
void SolvedPipeGrid_Destroy (
            SolvedPipeGrid * start )
```

Felszabadít egy SolvedPipeGrid láncolt listát.

Paraméterek

start	A láncolt lista első elemére mutató pointer

4.18.2.3. SolvedPipeGrid_RemoveFirst()

```
SolvedPipeGrid * SolvedPipeGrid_RemoveFirst (
            SolvedPipeGrid * start )
```

A SolvedPipeGrid láncolt lista első elemét felszabadítja és visszaadja a következő elem pointerét.

Paraméterek

start	A láncolt lista első elemére mutató pointer

Visszatérési érték

A következő elemre mutató pointer (NULL, ha nincs több elem)

4.19. lib/src/wheel.c fájlreferencia

A game.h-ban deklarált Wheel struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "game.h"
#include "debugmalloc.h"
```

Függvények

- Wheel Wheel_Init (void)
  A Wheel struktúra inicializálása (kinullázása)
- bool Wheel_Create (Wheel ∗wheel, Window window, SDL_Rect location, SDL_Texture ∗wheel_texture)
  Wheel struktúra létrehozása.
- bool Wheel_Update (Wheel wheel, Window window, SDL_Texture ∗wheel_texture)
  Frissíti egy Wheel Layer-ét.
- void Wheel_Destroy (Wheel ∗wheel)
  Felszabadít egy Wheel struktúrát.

4.19.1. Részletes leírás

A game.h-ban deklarált Wheel struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.19.2. Függvények dokumentációja

4.19.2.1. Wheel_Create()

```
bool Wheel_Create (
            Wheel * wheel,
            Window window,
            SDL_Rect location,
            SDL_Texture * wheel_texture )
```

Wheel struktúra létrehozása.

Véletlenszerűen felvesz egy szöget és lerendereli a Layer-re a csapot. A hibákat a struktok létrehozó függvényei a konzolra kiírják, ennek csak a visszatérési értéke jelzi. Hiba esetén nem ajánlott a struktúra használata.

Utófeltétel

Használat után a Wheel_Destroy() függvénnyel fel kell szabadítani a struktúrát.

Figyelmeztetés

TILOS egy már létrehozott Wheel struktúrát átadni.

Paraméterek

wheel	Egy inicializált Wheel struktúra pointere.
window	A renderert tartalmazó Window struktúra.
location	A csap szülőjéhez relatív helye
wheel_texture	A csap textúrája.

Visszatérési érték

true, hiba esetén false

4.19.2.2. Wheel_Destroy()

```
void Wheel_Destroy (
            Wheel * wheel )
```

Felszabadít egy Wheel struktúrát.

A Wheel Layer-ét szabadítja fel a saját függvényével. (lásd: SDL_local.h)

Paraméterek

wheel	A felszabadítandó Wheel struktúra pointere.

4.19.2.3. Wheel_Init()

```
Wheel Wheel_Init (
            void  )
```

A Wheel struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Wheel struktúra

4.19.2.4. Wheel_Update()

```
bool Wheel_Update (
            Wheel wheel,
            Window window,
            SDL_Texture * wheel_texture )
```

Frissíti egy Wheel Layer-ét.

A Wheel Layer-ét frissíti (újrarendereli) az aktuális értékei szerint. Az előforduló hibákat a függvény nem írja ki a konzolra, viszont a visszatérési értéke jelzi.

Paraméterek

wheel	A frissítendő Wheel
window	A renderert tartalmazó Window struktúra
pipe_texture	A csöveket tartalmazó SDL_Texture∗

Visszatérési érték

true, hiba esetén false

4.20. lib/src/window.c fájlreferencia

Az SDL_local.h-ban deklarált Window struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

```
#include "SDL_local.h"
#include "debugmalloc.h"
```

Függvények

- Window Window_Init (void)
  Window struktúra inicializálása (kinullázása)
- bool Window_Create (Window ∗window, const char ∗title, int width, int height)
  Feltölt egy inicializált Window struktúrát.
- void Window_Destroy (Window ∗window)
  Felszabadít egy Window struktúrát.

4.20.1. Részletes leírás

Az SDL_local.h-ban deklarált Window struktúrához kapcsolódó függvények kódjait tartalmazza a fájl.

4.20.2. Függvények dokumentációja

4.20.2.1. Window_Create()

```
bool Window_Create (
            Window * window,
            const char * title,
            int width,
            int height )
```

Feltölt egy inicializált Window struktúrát.

Létrehoz egy SDL_Window∗-ot és egy SDL_Renderer∗-t, majd a window paraméterben eltárolja ezeket. Az előforduló hibákat a függvény kiírja a konzolra. Hiba esetén nem ajánlott használni a struktúrát.

Utófeltétel

A Window_Create() által létrehozott Window struktúrákat használat után a Window_Destroy()-al kell felszabadítani.

Figyelmeztetés

TILOS egy, már létrehozott, Window struktúrát átadni, mert memóriaszivárgás lehet belőle.

Paraméterek

window	Egy inicializált Window struktúra pointere.
title	Az ablak neve
width	Az ablak szélessége (pixelben)
height	Az ablak magassága (pixelben)

Visszatérési érték

true, hiba esetén false

4.20.2.2. Window_Destroy()

```
void Window_Destroy (
            Window * window )
```

Felszabadít egy Window struktúrát.

Az átadott Window_Create() által létrehozott struktúrát szabadítja fel.

Paraméterek

window	A felszabadítandó Window struktúra pointere.

4.20.2.3. Window_Init()

```
Window Window_Init (
            void  )
```

Window struktúra inicializálása (kinullázása)

Visszatérési érték

Egy üres Window struktúra.

4.21. main.c fájlreferencia

A main függvényt tartalmazó fájl.

```
#include <time.h>
#include "game.h"
#include "debugmalloc.h"
```

Függvények

- int main (int argc, char *argv[])

  A main függvény.

4.21.1. Részletes leírás

A main függvényt tartalmazó fájl.

4.21.2. Függvények dokumentációja

4.21.2.1. main()

```
int main (
            int argc,
            char * argv[] )
```

A main függvény.

Paraméterek

argc	Parancssori argumentumok száma (nem használt)
argv	Parancsori argumentumok (nem használt)

Visszatérési érték

- 0: Ha sikeresen lefutott a program
- 1: Ha nem sikerült inicializálni az SDL-t
- 2: Ha nem sikerült létrehozni a játékhoz szükséges változókat
- 3: Ha a játék futása során történt hiba

4.22. readme.md fájlreferencia

Tárgymutató