Programozói dokumentáció

Tartalomjegyzék

Bevezető

A program 64 bites SDL2 2.28.5-el íródott (nem a forrásmappa része). 64 bites fordítót igényel. Amennyiben optimalizálva akarjuk fordítani, külön fordítsuk le RAM_handler-t anélkül, mert a debugmalloc-ot nem szereti a compiler olyankor. Mellékelve van egy compile.bat fájl, ami gcc-vel lefordítja a programot a fenti módszerrel optimalizálva. A program nem platformhoz kötött, azonban platform specifikus kényelmi korrekciókat tartalmaz (lásd settings.h). C99-es standardnak megfelel. Minden .c fájlhoz tartozik .h fájl, ezek modul fő fájl esetében /inc mappába, almodul forrásfájl esetében modul/inc mappában vannak. 3 forrásfájl nélküli heather fájl van: settings.h, structs.h, debugmalloc.h. Az első finomhangolja a program működését, a második a fő adatstruktúrákat definiálja,míg a harmadik a ramkezelést ellenőrzi. Utóbbi a program kilépésekor az eredményt kiírja debugmalloc_out.txt-be. A standerd error stream kepfeldolgozo_error.txt-be kerül átirányításra, a program első dolgaként. A GUI a nagyfelbontású monitorokhoz dinamikusan igazodik, azonban a softwares DPI scaleléshez nem igazodik, így érdemes 96-os DPI-t (100%-os monitor nagyítás) beállítani a program számára az oprendszerben.

Adatszerkezetek

Globális változók

Itt csak a modulokon átívelő globális változók vannak felsorolva. Eggyes modulok saját forrásfáljai között mások is vannak, ezek az adott modul leírásánál találhatóak.

extern char* input : input ablakba gépelt szöveget tárolja

Deklaráció: input_handler.h, Módosítja: input_handler.c
input_window.c innen olvassa ki a megjelenítendő input szöveget.

extern IMAGE_DATA IMAGE : A beolvasott kép adatait tárolja

Deklaráció: main.h, Módosítja: FILE_handler
FILE_handler függvényei ide mentik el a beolvasott adatokat, a többi modul csak olvassa.

extern bool kep_beolvasva : true ha a kép sikeresen beolvasva, és értelmezve

Deklaráció: main.h, Módosítja: main.c
Információt szolgáltat az adatbeolvasás állapotáról

Segéd szerkezetek

size_16 {w, h} : uint16_t számpárok tárolására

render_pos {x, y}: int32_t számpárok tárolására

IMAGE_DATA IMAGE

structs.h-ban deklarált IMAGE_DATA típusú. main.h-ban deklarált modulokon átívelő globális változó.

char* path : input ablakban bekért fájl útvonalat tárolja

Módosítja: main

size_16 size : kép méreteit tárolja

Módosítja: parameter_beolvas

uint16_t maxval : kép maximális színértéke
Módosítja: parameter_beolvas

void** color_data : kép színcsatornáit tárolja
Módosítja: pixel_beolvas
Dinamikusan kerül lefoglalásra,
[3][IMAGE.size.w*IMAGE.size.h*IMAGE.color_depth] alakú.
a 3 színcsatorna linrátisan tárolja az értékeket. Mivel a képfeldolgozás színcsatornánként történik, azokat külön-külön 1-1 egységes memóriablockba rakni megkönnyíti az esetleges másolást, és a cache-t is jobban kihasználja. A típus void, mert a kép színmélységétől függöen kell egy számhoz 1, vagy 2 byte, így nem kerül sor fölösleges memóriafoglalásra.

uint8_t color_depth : kép egy színértéke ennyi byteot foglal
Módosítja: parameter_beolvas

uint32_t pitch : kép egy sora hány byte-ot foglal el
Módosítja: parameter_beolvas

Modulok

main.c

Ez a program fő egysége. Itt fut az SDL Event loop, innen kerülnek meghívásra a különböző modulok függvényei.

int main(int argc, char *argv[]) : main függvény

A program main függvénye. Első feladata, a modulok INIT függvényeinek meghívása, majd az eventloop futtatása.

Locális változók

SDL_Event event : Az aktuális SDL event tárolását szolgálja

render_pos mouse_pos : Egér aktuális pozícióját tárolja

uint16_t VPwidth : ViewPort szélessége

bool bal_gomb : A bal egérgomb levan-e nyomva

bool csak_VPUpdate : Elég-e csak a ViewPort-ot újra renderelni
Ha ez igaz, csak a controll_texture lesz újrarenderelve, controll_surface nem. Az dinamikusan frissül a slidereket érintő GUI_handler.c függvényeken belül.

bool process_img : A megjelenített képet kell-e újra renderelni

bool first_update : Első-update-je e az ablaknak
Miután megnyílik az ablak hív egy update-t, hogy kirajzolja a dolgokat. Ezután ez false lesz, hogy a továbbiakban fölöslegesen már ne frissítsen mindent.

int Slider_megfogva : Van-e megfogva slide, ha igen melyik
Ha úgy nyomódik le a bal egérgomb, hogy az egér rajta áll egy slideren, az aktív slider sorszámát fogja tárolni (0-SLIDER_COUNT). Ha nem,-1 lesz. Balgomb elengedése értelem szerűen-1-re állítja.

int slider_cursor_diff : Slider széle-mouse_pos.x távolsága

A megfogott slider széle és a kurzor közötti vízszintes távolságot tárolja esztétikai okokból, hogy a megfogás természetesebbnek hasson.

A program indításkor meghívja a redirect_stderr, GUI_INIT, GUI_OpenInputWindow, INIT_RAM_handler, INIT_Input függvényeket, hogy a dolgok feláljanak. Ezután elkezdi futtatni az SDL Event loop-ot. Amíg a kép nics beolvasva, addig fenntart egy SDL_TextInput-ot, aminek az eredményét Input Ablakon keresztül jeleníti meg. Enter lenyomásakor, megpróbálja beolvasni a képet, ha sikertelen kezeli, ha sikeres kep_beolvasva true-lesz, és meghívja a GUI_EditorraValt függvényt. Ezután az Editor ablakhoz tartozó eventeket fogja kezelni. Görgetés, egérrel mozgatás esetén eldönti, melyik részét érinti az ablaknak az esemény, és ha szükséges, állítja csak_VPUpdate és process_img-et, ezután ugrik update-re. update visszaállítja csak_VPUpdate = true és process_img = false -ra, hogy a következő update a lehető legkevesebb dolgot frissítse.

Ha event.type == SDL_QUIT, kilép az eventloopból.Ezután meghívja Quit-ot.

void Quit(int code) : Kilép a programból a megadott kóddal

Felszabadítja a memóriát, bezárja a dolgokat

FILE_handler

Ez a modul felelős a fájlkezelésért. Itt értelmezi a beolvasott adatokat, amiket eltárol IMAGE-be. A beolvasásnál egymásnak adogatják a fájlt, hibakezelés return-ök formájában main.c-ben történik, ahonnan továbbadja ERROR_handler felé. Az int függvények return értéke (kivéve a seged_fuggvenyek.c elemei) enum FILE_HANDLER része.

FILE_handler.c

enum FILE_HANDLER kep_beolvas(char* path) : fájl megnyitása

Megnyitja a path-ba megadott fájlt. Továbbadja file_dekodol-nak.

static enum FILE_HANDLER file_dekodol(FILE* file) : fájl dekódolása

Beolvassa a file magic number-jét, kezeli, és továbbadja beolvas-nak.

void save_file(void** pixel_data) : Elmenti fájlba a képet

A kiírja pixel_data-t <eredeti_fájlnév>_edited.ppm fájlba.

void redirect_stderr() : Átirányítja stderr streamet a hibafájlba

Kepfeldolgozo_error.txt-be irányítja stderr-t.

beolvas.c

static enum FILE_HANDLER pixel_beolvas(FILE* file) : Beolvassa a pixeleket

Elmenti a beolvasott értékeket IMAGE.color_data-ba

enum FILE_HANDLER beolvas(FILE* file) : Beolvassa a fájlt

Beolvass a paramétereket, ellenőrzi, hogy van-e elég pixel, és beolvassa azokat.

parameter_beolvas.c

enum FILE_HANDLER parameter_beolvas(FILE* file) : Beolvassa a fájl paramétereit

A beolvasott értékeket elmenti IMAGE-be.

seged_fuggvenyek.c

Olyan segéd függvényeket tartalmaz a modulnak, amelyek nem kapcsolódnak közvetlenül a fájlkezeléshez, de bizonyos dolgokra kellenek.

int tizhatvany(int kitevo) : visszaadja tíz^kitevo-t

size_t pixel_byte_count(FILE* file, long pos) : beolvasható byteok számát adja vissza
pos, és a fájl vége közötti byteok számát adja vissza.

uint8_t* parameter_bovit(uint8_t c, uint8_t parameter_meret, uint8_t* parameter_tomb) : paraméter tömböt bővít
Dinamikusan foglalt tömböt ad vissza, nem szabadítja fel.

GUI_handler

Ez a modul kezeli a GUI-t. main.c csak GUI_handler.c függvényeivel kommunikál. Eltérő monitor felbontásokhoz dinamikusan igazodik, azonban ha az oprendszeren belül scaleelve van a monitor, azt a platform specifikus függvények elkerülése végett nem kezeli, így érdemes a programhoz külön 96-os DPI-t beállítani.

Modul szintű globális változók

Az összes változó render.h-ban van deklarálva. Értékeiket a hozzájuk tartozó init függvények során kapják meg.

GUI_RENDER RENDER

structs.h-ban deklarált GUI_RENDER típusú. render.c-ben deklarált GUI_handler-re kiterjedő globális változó. A renderhez általános adatokat tárol.

char* input : aktuális input input ablakhoz
input_handler globális input változója álltal mutatott tömbre mutat.

size_16 WindowSize : Ablak méretei
Módosítja: Window_UpdateSize

double kep_arany : Kép w/h
Módosítja: GUI_OpenEditorWindow

SDL_Renderer* renderer : SDL_Renderer az ablakhoz
Módosítja: Window_INIT, GUI_EditorraValt

double ColorScale : IMAGE.maxval / 255
Módosítja: RENDER_EditorraValt
Update_ImageTexture ezzel igazítja a kép minőségét a megjelenítéshez használt 8 biteshez.

GUI_CONTROLL CONTROLL

structs.h-ban deklarált GUI_CONTROLL típusú. render.c-ben deklarált GUI_handler-re kiterjedő globális változó. Az ablak kezelőfelületének adatait, állását tárolja.

uint16_t width : Controll szélessége
Módosítja: GUI_OpenEditorWindow

uint16_t SliderBox_height : SliderBox magassága
Módosítja: GUI_OpenEditorWindow

render_pos* SliderPos : SLIDER_COUNT elemű tömb
Módosítja: Update_ControllSurface, Handle_SliderChange
Sliderek jobb felső sarkait tárolja

size_16 SliderCharSize : Slider szélességét, és hosszát tárolja
Módosítja: RENDER_EditorraValt

uint16_t SliderLine_width : Slider vonalának a hosszát tárolja
Módosítja: Update_ControllSurface
Ez az érték határozza meg, hogy a slider mennyi mozgásteret kap.

uint16_t default_slider_pos : Slider alap x pozíciója relatívan a SliderBox-hoz
Módosítja: Update_ControllSurface

uint16_t max_slider_xpos : Maximum slider x pozíció relatívan a SliderBox-hoz
Módosítja: Update_ControllSurface

uint8_t SliderValue : SLIDER_COUNT elemű tömb
Módosítja: Handle_SliderChange, reset_slider_value
A sliderek-hez rendelt értéket tárolja (0-MAX_SLIDER_VALUE)
Az érték számolása: (CONTROLL.SliderPos[slider].x-CONTROLL.default_slider_xpos) / ((double) CONTROLL.SliderLine_width / MAX_SLIDER_VALUE)

GUI_ViewPort ViewPort

structs.h-ban deklarált GUI_ViewPort típusú. render.c-ben deklarált GUI_handler-re kiterjedő globális változó. A ViewPort adatait, állását táolja.

size_16 size : ViewPort méretei
Módosítja: Update_GUISize

render_pos offset : ViewPort beli kép eltolása relatívan a középre igazított álláshoz
Módosítja: Update_ViewPortRENDER, SetViewPortOffset, Update_ViewPortZoom, GUI_OpenEditorWindow

size_16 BaseScaleImgSize : 1-es zoom mellett a kép méretei
Módosítja: Update_GUISize,
A kép ilyenkor torzítás nélkül, a lehető legnagyobb, oly módon, hogy egy pixel se lógjon ki.

render_pos CenteredImgPos : középre igazítás mellett a kép jobb felső sarka
Módosítja: Update_GUISize, Update_ViewPortZoom

double ratio : ViewPort w/h
Módosítja: Update_GUISize

double ZoomScale : Zoomolás mértéke
Módosítja: Update_ViewPortZoom, GUI_EditorraValt
1: alaphelyzet, ilyenkor a kép torzítás nélkül pont kitőlti a ViewPort-ot
Magasabb értékekre ezzel lesz szorozva BaseScaleImgSize, hogy megkapjuk a nagyított kép méreteit

double MaxZoomScale : Maximális ZoomScale
Módosítja: Update_GUISize
Számolása: MAXZOOM x IMAGE.size.w / BaseScaleImgSize.w

GUI_handler.c

Ez a fájl közvetlenül nem kommunikál az SDL-el (kivéve a GUI_INIT, és Update_GUI), segédszámításokat viszont végez hozzá.

Almodulra kiterjedő globális változók

static size_16 min_ablakmeret_editor : Minimum ablakméret azt editornak
Módosítja: GUI_OpenEditorWindow
Alap értéke {.w = 400, .h = 400}, ha nem sikerül lekérni a monitor méreteket, ezt használja.

static size_16 MONITOR_SIZE : Lekért monitor méreteket itt tárolja
Módosítja: GUI_INIT
Alap értéke {.w = 400, .h = 400}, ha nem sikerül lekérni a monitor méreteket, ezt használja.

Függvények

void GUI_INIT() : GUI előkészítése

Ez a függvény készíti elő a GUI modult. SDL elindítása, monitor méretek lekérése, és RENDER_INIT meghívása.

void GUI_OpenInputWindow() : megnyitja az input ablakot

Beállítja az ablak méretét, létrehozza azt, és létrehozza a textúrákat az ablakba.

int Update_GUISize() : Ablak átméretezését kezeli

Módosítja ViewPort értékeit: size, ratio, BaseScaleImgSize, CenteredImgPos, MaxZoomScale. Controll méretét Update_Controll-meghívásával módosítja.
Visszaadja ViewPort szélességét

void GUI_OpenEditorWindow() : Megnyitja az editor ablakot

Kiszámolja az ablak, kezelőfelület méreteit, és létrehozza az ablakot. Hív egy Update_GUISize-ot.

void Update_GUI(bool csak_ViewPort, bool update_processed_image) : Frissíti a GUI-t

Bemenet:
csak_ViewPort: Elég-e csak a ViewPortot frissíteni. Csak a textúrát creálja újra controll_suface-ből, a surface dinamikusan frissül Handle_SliderChange és reset_slider_values-on belül.
update_processed_image: Kell-e a ViewPort-ban megjelített képet újra feldolgozni.

Törli a rendert, és a fentiek függvényében frissíti a dolgokat (Update_EditorWindow, Update_ImageTexture, Update_inputWindow), majd presentálja az új rendert.

void Update_ViewPortZoom(Sint32 scroll) : Kezeli a zoomolást

Bement:
scroll: A görgetés mértéke (event.wheel.y*SCROLL_CORRECTION-t kapja meg)

Változók:
double scale_inc : scroll*ZOOM, ZoomScale-hez az előző ZoomScale ennyiszeresét fogja hozzáadni a program, az új érték megkapásához. Ez 0-MAXZOOMSCALE-ig terjedhet.

Átállítja ZoomScale-t, és CenteredImgPos-t. offset-et úgy módosítja, hogy a zoomolás a ViewPort közepe felé közelítsen.

int GUI_EditorraValt() : Átvált editor ablakra

Visszatérés: ViewPort szélességét adja vissza

Hív egy sor függvényt, amik eltakarítják InputWindow-t, átváltja RENDER-t, és létrehozza EditorWindow-t.

void SetViewPortOffset(int xrel, int yrel) : Állítja ViewPort.offset-et

Bemenet: Az egér kiindulóponthoz vett relatív elmozdulása

A megadott elmozdulást berakja ViewPort.offset-be

int CheckForSliderContact(render_pos mouse_pos) : Ellenőrzi, hogy mouse_pos rajta van-e egy slideren

Bemenet: Az egér aktuális koordinátája

Ha mouse_pos érint egy slider-t, visszaadja a sorszámát. Ha nem, -1-et ad vissza. A függvény egész osztással szerzi meg az érintett SliderBox sorszámát, és unsigned int overflow-t használ ki, hogy egy tesztben ellenőrizze, hogy se előtte, se mögötte ne legyen az egér a slider-nek a gyors működés érdekében.

void Handle_SliderChange(int mouse_xpos, int slider, int slider_cursor_diff) : Kezeli a megfogott slider huzogatását

Bemenet:

mouse_xpos: az egér aktuális x pozícója

slider: A megfogott slider

slider_cursor_diff: mouse_xpos és SliderPos[slider].x különbsége

újraszámolja az adott sliderhez tartozó értéket, és vizuálisan hozzáigazítja a slider pozícióját.

int SliderCursorDiff(int mouse_xpos, int slider) : Az egér-slider x különbségét adja vissza

Bemenet:

mosue_xpos: aktuális egér x pozíció relatívan az ablakhoz

slider: A kérdéses slider

Visszatérés:

Visszaküldi mouse_xpos (controll-hoz relatívan igazítva) és SliderPos[slider].x különbségét.

void reset_slider_values() : Alapheylzetbe állítja a slidereket

void GUI_Quit() : Bezárja a rendert, és az SDL-t

window_handler.c

Ez az almodul kezeli közvetlenül az ablakot. Egyszerre csak 1 ablakot képes kezelni.

Almodul szintű globális változók

static SDL_Window* window : Az aktuális ablak

Függvények

void Window_UpdateSize() : Frissíti az ablak méreteit RENDER-be

void Window_INIT(int init_w, int init_h, int min_w, int min_h, Uint32 flags) : Létrehozza az ablakot

Bemenet:

init_w, init_h, min_w, min_h : Az ablak kezdeti, illetve minimum méretei

flags: SDL flag az ablakhoz

létrehoz egy középre pozícionált ablakot a paraméterek alapján, Kepszerkeszto néven. Az ablakhoz tartozó renderer-t berakja RENDER.renderer-be.

void Window_destroy() : Törli az ablakot.

render.c

Ez a fájl tartalmazza az általános renderrel kapcsolatos függvényeket.

Függvények

void RENDER_INIT() : Előkészíti a dolgokat

beállítja az ascii surface-t seged_fuggvenyek.c-nek

void RENDER_EditorraValt() : Átállítja a rendert editorwindow-ra

Törli InputWindow textúráit, beállítja ColorSacale-t, initializálja Controll-t ,és img_processing-et.

void RENDER_Quit() : Felszabadítja a meglévő textúrákat, renderhez tartozó tömböket

input_window.c

Ez a fájl tartalmazza az InputWindow-t kezelő függvényeket.

Almodulra kiterjedő globális változók

static SDL_Texture* TEXTURES[4] : Kiirandó szövegek textúráit tárolja

static char* error_str : Az esetleges hibaüzenetet tárolja

Függvények

void Update_inputWindow() : Frissíti az input ablakot

frissíti a szövegek textúráit, majd kimásolja őket settings.h-ban definiált ypos-nak és scale-nek megfelelően.

void INIT_InputWindow() : Létrehozza a szövegek textúráit, renderre másolja azokat

void SetInputGUIError(char*str) : Bállítja error_str értékét

Ez a fügvény frissíti a megjelendő szöveget, amit majd Update_inputWindow fog textúrává alakítani TEXTURES[3]-ba.

void inputWindow_Quit() : Törli a textúrákat TEXTURES-ből

CONTROLL.c

Ez a fájl kezeli a controll felületet. CONTROLL_SURFACE (almodul szingű globális változó) tárolja az aktuális surface-t, amit külön függvény (Update_ControllTexture) alakít textúrává, és egy harmadik (Draw_Controll) másolja rá az ablakrenderre. Ez a megoldást erőforrást takarít meg, hiszen utóbbi minden ablak frissítésnél szükséges, a textúrát frissíteni viszont csak akkor kell, ha változik, mit akarunk megjeleníteni. Az pedig, hogy külön megtartjuk a surface-t, lehetővé teszi, hogy blitelve, csak azt a részét frissítsük a controll-nak, amit kell, így gyorsítva a folyamatot. A slider sorszáma megeggyezik a hozzá kötött, IMG_processing-be tartozó edit_type-al. Ha egy slider változik, annak a sorszáma lesz a változó edit_type, vagyis a végrehajtandó képfeldolgozó lépés.

Almodul szintű globális változók

static SDL_Surface* CONTROLL_SURFACE : A megjelenítendő surface

static SDL_Surface* SliderChar_surface : A slider textúrája

static SDL_Texture* TEXTURE : A megjelenítendő textúra

static size_16 ControllSize : Controll méretei

static uint16_t SliderBoxHeight : Egy sliderBox magassága

static SDL_Rect CharDST : SliderChar érkezése
w, h paramétere tárolja az érkezési méretet, x, y koordinátája az adott update-hez igazodik

static SDL_Rect LineDST : Slider line érkezését tárolja
w, h paramétere tárolja az érkezési méretet, x, y koordinátája az adott update-hez igazodik

static SDL_Rect ValDST : Slider értékét tartalmazó textúra érkezése
Minden paramétere dinamikusan igazodik a frissítéshez

Függvények

static void CreateEmptySliderBox(uint16_t slider) : Slider nélküli SliderBoxot kreál

Bemenet: A sliderbox sorszáma, amit kiürítsen

Kirajzolja egy SliderBox-ba a vonalat, és a hozzá tartozó feliratot. (AKA törli az ott lévő dinamikusan változó dolgokat). Csak CONTROLL_SURFACE-t frissíti.

void Update_SliderBox(uint16_t slider) : Frissíti az adott slider-t vizuálisan

Bemenet: Az érintett slider sorszáma

Újragenerálja az adott slider-boxot, az aktuális(friss) értékekkel. Csak CONTROLL_SURFACE-t frissíti.

void Update_ControllSurface() : A teljes CONTROLL_SURFACE-t legenerálja

Az egész CONTROLL_SURFACE-t létrehozza a 0-ról,az aktuális értékek alapján.

void INIT_Controll() : Legenerálja a Controll-t

Ha már létezik CONTROLL_SURFACE, nem csinál mindent a 0-ról, csak átméretezi a surface-t, és legenerálja. Ha nem, Az egész controll-t initializálja, CONTROLL dinamikusan foglalt tömb paramétereit is beleértve. Frissíti a surface-t.

void Update_ControllTexture() : Újragenerálja TEXTURE-t CONTROLL_SURFACE-ből

void Draw_Controll() : Az ablak renderre másolja TEXTURE-t

void controll_Quit() : Törli a textúrákat, memóriafoglalásokat Controll-hoz

ViewPort.c

Ez a modul kezeli a ViewPort-ot.

Almodul szintű globális változók

static SDL_Texture* TEXTURE : A megjelenítendő kép SDL_Texture-ként

Függvények

void Update_ImageTexture() : Frissíti TEXTURE-t

Ez a függvény indítja el a képfeldolgozást. A feldolgozott kép-et convertálja SDL_Texture-be. A szímélységet úgy skálázza, hogy 8 bites legyen a végeredmény

void Update_ViewPortREMDER() : Frissíti a ViewPort-ot az ablakrenderen.

A függvény kiszámolja, a zoomolás és offset alapján, hogy az eredeti kép mely részei lesznek láthatóak, és annak megfelelően másolja csak az érintett részt a render-re, ezzel erőforrást megtakarítva.

void ViewPort_Quit() : Törli TEXTURE-t

seged_fuggvenyek.c

Ez a fájl tartalmaz általánosabb függvényeket a modulhoz.

Almodul szintű globális változók

static SDL_Surface*ASCII : Surface az ascii táblázatból

Függvények

SDL_Surface* ReSize_Surface(SDL_Surface* surface, size_16 NewSize) : Újraméretez egy surface-t

Bemenet:

surface: A módosítani kívánt surface

NewSize: A surface új mérete

Visszatérés: Az új surface

A függvény törli az eredeti surface-t, helyette csinál egy újjat az új méretekkel. A régi tartalmát torzítva átmásolja.

SDL_Surface* CreateSurface_TextLine(char* szoveg) : Surface-t csinál egy szövegből egy vonalba

Bemenet: A szöveg

NULL esetén a viselkedés határozatlan

Visszatérés: A surface, ami tartalmazza a szöveget

Ha üres karakterláncot kap, 1 széles, 17 magas fekete surface-t küld vissza

SDL_Texture* CreateTexture_TextLine(SDL_Renderer* renderer, char* szoveg) : Texture-t cinál a szövegből egy vonalba

Bemenet:

rebderer: SDL renderer

szoveg: A szöveg. NULL esetén a viselkedés határozatlan

Visszatérés: A surface, ami tartalmazza a szöveget

Ha üres karakterláncot kap, 1 széles, 17 magas fekete textúrát-t küld vissza

void SetAscii(SDL_Surface* ascii) : Beállítja ASCII-t.

void FreeAscii() : törli ascii-t.

IMG_processing

Ez a modul végzi a képfeldolgozást. Az eggyes feldolgozó modulok processing_units mappában találhatóak külön forrásfájlokban, azonban egy közös processing_units.h heather fájl fogja össze őket. Eggyes unit-okhoz tartoznak free függvények, melyek a dinamikusan foglalt kernelt szabadítják fel. A program használ nem visszafordítható képfeldolgozási folyamatokat is,ezért minden filter érték változásnál az összes korábbit is újraszámolja. Ennek felgyorsítására a kerneleket nem számolja újra, ha a hozzá tartozó érték nem változott. (ez csak blur-re igaz, sharpen esetében fix méretű

3x3-as kernelről beszélünk, teljesítmény nyereségről értelmetlen lenne beszélni abban az esetben. A többi kernelt használú unit (edge_detection) konstans kernelt használ.

img_processing.c

Ez a fájl tartalmazza a képfeldolgozást irányító függvényeket. A hozzáadott filtereket egy láncolt listában rögzíti (edit_list), amit bővít, ha új filter kerül hozzáadásra. Egy feldolgozás során ezen a listán megy sorban végig, és hívja meg az érintett unit-okat. Az eredeti képet IMAGE.color_data-ból PIXEL_DATA-ba másolja, és azt módosítja.

edit_list szerkezet:

enum edit_types :
BLUR,
SHARPEN,
EDGE_DETECTION,
DARKEN,
BRIGHTEN,
CONTRAST_INC

typedef struct edit :
enum edit_types action : A filter típusa
uint8_t value : A filter értéke
struct edit* kov : A láncolt lista következő eleme

Almodul szintű globális változók:

void** PIXEL_DATA : Az eredeti kép másolata, ezt módosítják sorban a unit-ok
size_t ColorChannel_size : Egy színcsatorna méretét tárolja byteban

void handle_ValueChange(enum edit_types action, uint8_t value) : Kezeli az értékváltozást

Bemenet:
action: a módosítás típusa (módisított slider sorszáma)
value: a hozzárendelt érték (0-MAX_SLIDER_VALUE)

Ha már benne van a listában, módosítja a hozzátartozó értéket, ha nincs, hozzáfűzi a listához az új action-t, és értéket ad neki.

void init_processing() : Lefoglalja PIXEL_DATA-t

void** process() : Feldolgozza a képet

visszatérés: A feldolgozott kép

Feldolgozza a képet, edit_list alapján sorban haladva. Mindig az eredeti képpel kezd.

void Save_image() : Elmenti az aktuálisan megjelenített képet fájlba.

<eredeti fájlnév>_edited.ppm néven menti el. (lásd: save_file)

void edit_list_free() : Törli edit_list-et

void quit_processing() : törli a listát, törli PIXEL_DATA-t, és felszabadítja a kerneleket.

Unitok

5 darab unit van: blur (Guassian), sharpen, edge_detection, darken, brighten, contrast_inc. Mindeggyik a nevének megeggyező módosítást végzi el a képen. Van még egy generic.c fájl, ez általános függvényeket tartalmaz (pontosan 1 darabot, ami általános mátrix konvolúciót hajt végre). Alább csak az említésre méltóbb unit-ok szerepelnek részletesebben.

generic.c

void convolve(uint8_t* src, double* kernel, size_16 src_size, size_t kernel_size, uint8_t depth, uint16_t maxval) : Általános mátrix konvolúció

Bemenet:

src: A forrásmátrix
kernel: A kernel, amiről feltételezi, hogy nxn-es.
src_size: A forrás mátrix méretei
kernel_size: kernel oldalhossza
depth: A forrásmátrix színmélysége (1 vagy 2 byte tárol 1 számot)
maxval: A forrásmátrix max színértéke

A függvény a részeredményeket egy ideiglenes mátrixban tárolja, majd ha végzett, akkor másolja az eredményeket a forrásmátrixba. A kernel indexeket előre kiszámolja, ezzel erőforrást takarít meg. Ezt statikus változóban eltárolja, csak akkor frissíti, ha változik a kernel mérete. 50% körüli időmegtakarítást eredményez, ha a legnagyobb optimalizálással fordítjuk. (ezt a debugmalloc nem szereti, de ezt majd RAM_handler-nél) A kép határait tükörként tekinti, vagyis ha a kernel lelóg a képről, a lelógó rész vissza lesz tükrözve a kép közepe felé.

blur.c

Guassian blur-t használ. A kernel-hez használt Guass-függvény:

$$\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2}$$

Sigmát a következő alapján becsüli meg:

$$\frac{size}{2\sqrt{2\ln size}}$$

ezzel elérve, hogy a Guass-görbe pont jól „kitöltse” a kernel-t.

A kernelt természetesen normalizálja, hogy fényerő változás ne legyen jelen mint mellékhatás. A mátrixot statikus változóban eltárolja, csak akkor generálja újra, ha változik a mérete. Az elmosás mértékét a mátrix méretével határozza meg, ami 3-tól a kép domináns dimenziójának méretének az 1/5-éig terjedhet.

edge_detection.c

Sobel operációt használ, a küszöbérték mindig a megadott value 7-szerese. A képet először fekete-fehérré alakítja, majd arra futtattja az algoritmust. A túlbonyolítás elkerülése végett nem optimalizálja az adott képre sem a kernmelt, sem a küszöbértéket. Ez a unit nem használja a generic.c-t, mert gyorsabb az összeg kiszámolását követően egyből a küszöbértékhez viszonyítani, és azt elmenteni. Illetve a kernel túllógása sem érinti, hiszen konstans 3x3-as kernelt használ, és ha a kernel lelóg, egyépként is érelmét veszti az élkeresés, így itt csak azokara a pixelekre fut le, ahol nem lóg túl a kernel.

A használt kernelek:

Horizontális élek detektálása

$$\begin{matrix} -4 & 0 & 4 \\ -5 & 0 & 5 \\ -4 & 0 & 4 \end{matrix}$$

Vertikális élek detekátálása

$$\begin{matrix} -4 & -5 & -4 \\ 0 & 0 & 0 \\ 4 & 5 & 4 \end{matrix}$$

sharpen.c

A megkapott value-t $\frac{value}{MAX\_SLIDER\_VALUE}$ alapján normalizálja 0-1 közé, majd a következő kernelt hozza létre:

Az élesítéshez használt filter: $\begin{matrix} 0 & -strength & 0 \\ -strength & 5 * strength & -strength \\ 0 & -strength & 0 \end{matrix}$

ahol $strength = \frac{value}{MAX\_SLIDER\_VALUE}$

ERROR_handler.c

Ez a modul felel a hibakezelésért. Ez Check fügvényeken keresztül történik, amik paraméterként bekérik az ellenőrizendő értéket, majd ellenőrzik. A hibát a standard error streamre írja, amit a program elején kepfeldolgozo_errot.txt-be irányít.

Enumok

enum FILE_HANDLER: FILE_handler lehetséges visszatérési értékei

- SUCCESS : Beolvasás sikeres
- FILE_NOT_FOUND: A fájl megnyitása sikertelen (nem létezik)
- FILE_NOT_PPM: A fájl nem PPM magicnumberrel van ellátva
- FILE_NOT_VALID: Valami egyébb hiba/sérülés van a fájlban
- FILE_TOO_BIG: Akép túl nagy (>LONG_MAX*3 byte)

enum SDL_CHECK: Milyen tipusu értéket ellenőrizzen SDL-től

- FUNC_RES: Függvény visszatérési értéke
- SDL_PTR: SDL struktúrára mutató pointer

Függvények

bool Check_FILE_HANDLER(enum FILE_HANDLER res) : File beolvasás eredményét kezeli

Bemenet: FILE_HANDLER enum eleme (kep_beolvas visszatérése)

Visszatérés: Beolvasás sikeres-e

Ha sikertelen, beállít egy InputGUIError-t róla.

bool Check_SDL(intptr_t obj, int type, char* label, bool kritikus) : SDL visszatérését ellenőrzi

Bemenet:

obj: Az ellenőrizendő érték

type: SDL_CHECK eleme, az érték típusát jelzi

label: Egy rövid leírás, hogy mi ez (hiba logolásához)

kritikus: ha igaz, és hibás, bezárja a programot

Visszatérés: Hibás-e az érték.

void Check_DyArraymalloc(void* arr, size_t block_count) : Dinamikusan foglalt tömböt ellenőrzi

Bemenet:

arr: a tömb

block_count: A memóriablockock száma (2D-s tömböket is ellenőriz, ha nagyobb mint 1);

Bezárja a programot ha hibás. Minden memóriafoglalás kritikus hibaként van kezelve.

generic_fuggvenyek.c

Ez a modul az összes modul számára hasznos általános függvényeket (1 darabot) tartalmaz.

Függvények

char* StringAssembler(char* str, ...) : Összerakja a stringet

Bemenet: String, és a hozzá tartozó formatáló változók

Visszatérés: Az összerakott string

Egyszer használatosra tervezett, remember_dmalloc-al foglal. Ennek következménye, hogy ha 2x használjuk, az első foglalás-t felszabadítja.

input_handler.c

Ez a modul kezeli az InputWindow inputjának tömbjét.

Globális változók

char* input : Az aktuális input szövege

Ez modolukon átívelő változó. Mindig minimum egy '\0' karaktert tartalmaz.

static size_t input_meret : input tömb elemeinek a száma (min. 1)

Függvények

void INIT_Input() : Lefoglalja inputot, feltölti '\0'-val.

void Input_Bovit(char* str) : Bővíti str-el input-ot

void Input_Torol() : Törli az utolsó betűt input-ból

void Input_RESET() : Reseteli input-ot

RAM_handler.c

Ez a modul felel a ramkezelésért. debugmalloc.h ide van includeolva. A program többi részének jót tesz (főleg convolve-nak) ha optimalizálva fordítjuk le, viszont debugmalloc-ot nem szokta engedni, hibát dob rá. Ennek megoldása, ha ezt a modult külön lefordítjuk optimalizáció nélkül .o fájlba, majd azzal fordítjuk a program többi részét optimalizálva. Emiatt ez a modul teljesen független a többi modultól. A függetlenség hátránya, hogy ERROR_handler ram ellenőrző függvényét minden alkalommal, ha ennek a modulnak a függvényeivel foglalunk, meg kell külön hívni. Debugmalloc kimenetét debugmalloc_out.txt-be irányítja.

Modul szintű globális változók:

static void* remember : remember_dmalloc-al foglalt utolsó cím

Függvények

void* dmalloc(size_t size) : Lefoglal size méretű memóriát

Visszatérés: A lefoglat terület címe

2147483647 byteot nem haladhatja meg ez a méret.

void dfree(void* addr) : Felszavadítja az adott címet

void* tomb_atmeretezo(void* tomb, size_t meret, size_t ujmeret, size_t elem_meret) : Átméretez egy tömböt

Bemenet:

tomb: az átméretezendő tömb

meret: jelenleg hány elemet tartalmaz

ujmeret: új mérete hány elemet fog tartalmazni

elem_meret: Egy elem hány byte

Visszatérés: Az új tömb címe

Az elemeket átmásolja az új tömbbe (ha kisebb, akkor csak az új tömb végéig megy), és felszabadítja a régi tömböt

void** tomb2D_malloc(size_t block_count, size_t block_size) : 2D-s tömböt foglal le

Bemenet:

block_count: Hány memóriablockból álljon a pointertömb

block_size: Egy block hány byteos legyen

Visszatérés: A pointertömb kezdetének a címe

void tomb2D_free(void** tomb, size_t block_count) : 2D-s tömböt szabadít fel

void* remember_dmalloc(size_t size) : Megjegyzi az utoljára foglalt területet

Olyan malloc függvény, ami megjegyzi az utoljára foglalt terület címét. Ha újra használjuk az előzőt felszabadítja. Egyszer használatos foglalásokhoz (pl. StringAssembler) hasznos, így egy helyen van a memória foglalás/felszabadítás.

void release_remembered_addr() : Felszabadítja az utoljára megjegyzett területet

void copy_2Dtomb(void** dest, void** src, size_t block_count, size_t block_size) : Átmásol egy 2D-s tömböt

Bemenet:

dest: a cím, ahova másolja

src: a forrás tömb

block_count: hány elemű a pointertömb

block_size: Egy memóriablock hány byte

void INIT_RAM_handler() : Beállítja debugmallocot

A program indításakor egyszer fut le. A foglalható max ramméretet a maximum értékre (LONG_MAX), a kimenetet pedig debugmalloc_out.txt-re állítja.

settings.h

Ez a fájl konstansokat tartalmaz, amik finomhangolják a felhasználói élményt, illetve egyes funkciók működését hangolják.

Általános

ASSET_PATH : Az asseteket tartalmazó mappa útvonala

Default: ”assets/”

ViewPort

MAXZOOM : A kép 1 pixele maximum ennyi szélességű lesz a monitoron

Default: 5

ZOOM: Zoomolási együttható

Default: 0.04

A Zoomolás sebességét állítja

SLIDER_COUNT : Hány darab slider van

Defaul : 6

Controll

MAX_SLIDER_VALUE : Egy slider maximum értéke

Default: 100

0-256 között kell legyen

CONTROLL_WIDTH : 1920p széles monitoron a controll szélessége

Default: 300

Monitor felbontásához dinamikusan igazodik azányokat megtartva

CONTROLL_HEIGHT : 1080p magas monitoron a controll min. magassága

Default: 600

Monitor felbontásához dinamikusan igazodik azányokat megtartva

uint8_t CONTROLL_BGND[3] : Controll hátterének 8 bites RGB kódja

Default: {100, 100, 100}

uint8_t SLIDER_COLOR[3] : Egy slider vonal 8 bites RGB kódja

Default: {0, 0, 0}

char* Slider_nevek[] : A sliderek feliratait tartalmazza

Default: {”Elmosas”, ”Elesites”, ”Elkereses”, ”Sotetites”, ”Vilagositas ”Kontraszt”}

Felülről lefele, a slider sorszáma megeggyezik a hozzá tartozó név indexével.

char SLIDER_CHAR[2] : A slider csúszkájának ábraként szólgáló ascii karakter kódja

Default: {10, ’\0’}

Kell egy lezáró karakter is mögé, mert a kirajzoló függvény teljes stringekre specializálódott

Input

double scales[4] : input ablakban a feliratok méretei

Default: {5, 3, 2, 3}

A szöveg felülről lefele sorszáma megeggyezik a tömb beli hozzá rendelt szám sorszámával

int ypos[4] : input ablakban a szövegek vertikális pozíói

Default: {0, 100, 200, 300}

A szöveg felülről lefele sorszáma megeggyezik a tömb beli hozzá rendelt szám sorszámával. A szám 1920p magas monitoron a pixelek száma, monitor méretéhez dinamikusan igazodik.

Korrekciók

SCROLL_CORRECTION : Együttható az egérgörgetés részére

Default: Mac: -1 Egyébb: 1

Mac-en megfordítja a görgetést