Langton's Ant Programozói dokumentáció

A program a Langton hangyája nevű algoritmust rajzolja ki egy ablakban. A programnak lehet állítani az ablak felbontását, a hangya utasításkészletét és a kirajzoláshoz tartozó értékeket.

A program ezekre a fájlokra van bontva:

- main.c – a program main jét tartalmazza
- menu.c – a menu rajzolását és a méretre szabását végző függvényeket tartalmazza
- ant.c – a hangya algoritmusát és kirajzolását végző függvényeket tartalmazza
- Utilities.c – a programban lévő segédfüggvényeket tartalmazza
- SDLHandler.c – az SDL kezelését és a program leállítását végző függvényeket tartalmazza
- colors.c – a program által használt színeket tartalmazza

- main.h – a main deklarációját tartalmazza
- menu.h – a menu.c függvényeinek a deklarációit tartalmazza
- ant.h – az ant.c függvényeinek a deklarációit tartalmazza
- Utilites.h az utilities.c függvényeinek a deklarációit tartalmazza
- SDLHandler.h az SDLHandler.c függvényeinek a deklarációit tartalmazza
- colors.h – a program által használt színeket tartalmazza
- includes.h – a program által használt c és SDL header öket tartalmazza

includes.h

A program által használt összes c és sdl header include ot tartalmazza.

struct Settings

A program beállításához szükséges változókat tartalmazza

SDL_Rect SCREEN

Az ablak méreteit tartalmazza.

int SCREEN.w

Az ablak szélessége.
Legalább 1

int SCREEN.h

Az ablak magassága.
Legalább 1

int SCALE

A kirajzolt négyzetek mérete pixelben.
Legalább 1

int SPACING
A négyzetrácsozás mérete.
Legalább 0

int ANTMARGIN
A hangya ennyi pixel * 2 ször kisebb, mint a SCALE
Legalább 1

int MSTICK
A szimuláció ennyi milliszekundumonként fut.
Erőforrás mennyiségétől függően legalább 8, inkább 16.

char instructionset[19]
A hangya utasításkészlete amit a színekhez rendel.
Csak R, L, N és U t tartalmazhat, de mindegy hogy kis vagy nagybetű.

main.c/h

int main(int argc, char** argv)
A main-nek egyetlen paramétert lehet adni amivel a konfigurációs fájl nevét és kiterjesztését lehet változtatni. Ha nem adjuk meg neki akkor az alapértelmezett config.ini t tölti be.

char configfilename[52]
A konfig fájl neve és kiterjesztése, maximum 51 karakter.

int lepes
A hangya elvégzett lépéseinek a számát tárolja.

int instructnum
Az utasításkészletben lévő utasítások számát tárolja

Uint32 *pixels
A pixelek színét tárolja abban a formátumban amelyben az SDL kezeli

Uint32 **pixelTex
A pixelek színét tárolja emberek számára könnyebben kezelhető formátumban

int num
A program a menübe való visszatéréseinek a számát tárolja

ant.c/h
A hangya kezelését, mozgatását és rajzolását végző függvényeket tartalmazza

struct Ant
A hangya adatait tartalmazza.

int Ant.x
A hangya pozíciója vízszintesen

int Ant.y
A hangya pozíciója függőlegesen

int Ant.heading
A hangya iránya, merre néz a hangya
0: fel
90: jobbra
180: le
270: balra

int Ant.lasttile
Azon négyzet színe, amin a hangya a mozgása előtt állt.

int Ant.turn [18]
A megadott utasításkészlet átkonvertálva a hangyának forgatási mennyiségekre
90: jobbra
-90: balra
0: előre
180: hátra

bool moveAnt(Uint32*** pixelTex, Ant* ant, int* lepes, Settings settings, int instructnum, FILE* fAntOut)
(a textúra amibe a hangyát és a négyzeteket lehet rajzolni,
a hangya,
a lépésszámláló,
a beállításokat tároló struktúra,
az utasításkészletben lévő utasítások száma,
a fájl amibe írni szeretnénk a hangya elvégzett utasításait)

A hangyát mozgatja és színezi a hangya helyét és az előző négyzet helyét.

if (ant->lasttile == instructnum-1)ant->lasttile = 18;
Csak annyi szín használata ahány kell.

bool turnAnt(Ant* ant, int tile, FILE* fAntOut)
(a hangya,
a négyzet színe,
a fájl amibe írni szeretnénk a hangya elvégzett utasításait)

A hangyát a megfelelő irányba fordítja és ezt kiírja a fájlba.

bool antgorithm(Uint32*** pixelTex, Ant* ant, Settings settings, FILE* fAntOut)
(a textúra amibe a hangyát és a négyzeteket lehet rajzolni,
a hangya,
a beállításokat tároló struktúra,
a fájl amibe írni szeretnénk a hangya elvégzett utasításait)

Meghívja a hangyát irányba forgató függvényt a hangya alatt lévő négyzet színe alapján.

int xpos = (ant->x - settings.SCALE - settings.SPACING + settings.ANTMARGIN);
A hangya alatt lévő négyzet pozíciója

void convertToTurns(char* instructionset, Ant* ant)
(a hangya utasításkészlete,
a hangya)

Átkonvertálja a megadott utasításkészletet a hangyának forgatási mennyiségekre.

menu.c/h

void drawMenu(TTF_Font** StartFont, TTF_Font** MenuFont, TTF_Font** InstructFont, TTF_Font** HelpFont, SDL_Texture** tStrings, SDL_Rect* lStrings, SDL_Window** gWindow, SDL_Renderer** gRenderer, SDL_Texture** tPixelTexture, SDL_Texture** tMainMenu, Settings settings, SDL_Rect HelpButton1, char help[5][14], SDL_Rect StartButton, SDL_Rect StartButtonStroke, SDL_Rect ResButton, SDL_Rect ResUp, SDL_Rect ResDown, SDL_Rect ScaleButton, SDL_Rect InstructButton)
A menüt rajzolja ki.

void refreshMenu(SDL_Window** gWindow, SDL_Renderer** gRenderer, SDL_Texture** tPixelTexture, SDL_Texture** tMainMenu, SDL_Rect SCREEN, int Strokesize,SDL_Rect* StartButton, SDL_Rect* StartButtonStroke, SDL_Rect* ResButton, SDL_Rect* ResUp, SDL_Rect* ResDown, SDL_Rect* ScaleButton, SDL_Rect* InstructButton)
Átméretezi az ablakot az új felbontáshoz és a gombok helyét is frissíti hogy jó helyen legyenek az új ablakban.

void setButtons(SDL_Rect SCREEN, int Strokesize, SDL_Rect* StartButton, SDL_Rect* StartButtonStroke,SDL_Rect* ResButton, SDL_Rect* ResUp, SDL_Rect* ResDown, SDL_Rect* ScaleButton, SDL_Rect* InstructButton)
Frissíti minden gomb helyét a felbontásnak megfelelően

Utilities.c/h

void memset32(void * dest, Uint32 value, uintptr_t size)
memset ami 32 bites unsigned integerrel dolgozik, hogy bármilyen színűre lehessen állítani a hátteret

Uint32 ftick(Uint32 ms, void *param)
A szimuláció alap tickrate-jére generál egy SDL_USEREVENT et, ez alapján fut a szimuláció.

void convertPixels(Uint32** pixels, Uint32*** pixelTex, SDL_Rect SCREEN)
A kezelt Uint32*** által megadott pixelTex 2D s textúrát konvertálja át az SDL által kezelt Uint32** pixels által megadott 1D s tömbbé.

void drawTextintoButton(SDL_Renderer* gRenderer, TTF_Font* font, SDL_Texture** tStrings, SDL_Rect* lStrings, SDL_Rect button, char* text, SDL_Color color)

A char* text által megadott string-et írja be az SDL_Rect button által megadott téglalap közepébe.

void loadintFromConfig(FILE* wDefConf, char* buffer, int* variable, char* variableName)

Egy integert tölt be a char* variableName által megadott név alapján.

void loadcharFromConfig(FILE* wDefConf, char* buffer, char* variable, char* variableName)

Egy string et tölt be a char* variableName által megadott név alapján.

void loadConfig(FILE* file, Settings* settings, int* instructnum)

Betölti az összes értéket a config fájlból.

SDLHandler.c/h

Az SDL és az ahhoz tartozó változók kezelését végző függvényeket tartalmazza

bool initSDL(SDL_Window** gWindow, SDL_Renderer** gRenderer, SDL_Texture** tPixelTexture, SDL_Texture** tMainMenu, SDL_Rect SCREEN)

Az SDLt és az alap változókat, mint az ablak és a renderer inicializálja.

void initTexture(SDL_Renderer** gRenderer, SDL_Texture** tTexture, SDL_Rect SCREEN)

Az SDL_Texture** tTexture által megadott textúrát inicializálja.

void initPixels(Uint32** pixels, Uint32*** pixelTex, SDL_Rect SCREEN)

A hangya és az útvonala kirajzolásához használt tömböket inicializálja.

void close(Uint32** pixels, Uint32*** pixelTex, SDL_Window* gWindow, SDL_Renderer* gRenderer, SDL_Texture* tPixelTexture, SDL_Texture* tMainMenu, SDL_Texture* tStrings)

Mindent felszabadít és bezárja az SDL t.

void save_texture(SDL_Renderer* gRenderer, SDL_Texture* tTexture, const char *filename)

A megadott SDL_Texture* tTexture t menti el egy bmp fájlba.

colors.c/h

A program által használt színeket tartalmazza

enum HEXARGB

A hangya által hagyott négyzetek kirajzolásához használt színek.
Formátum: 0xAARRGGBB
AA: alpha hex számként 0-256
RR: piros hex számként 0-256

GG: zöld hex számként 0-256
BB: kék hex számként 0-256

enum HEXRGBA

A menü kirajzolásához használt színek.
Formátum: 0xRRGGBBAA

extern SDL_Color TextOrange, TextDarkOrange

A szövegek színei.
Formátum: R,G,B
R: piros dec számként 0-256
G: zöld dec számként 0-256
B: kék dec számként 0-256