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

# main.c

## 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.

### 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.

### 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

## 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