=-=-= infoClicker =-=-=
felhasználói dokumentáció

A játék használata szinte kizárólag az egérrel, és annak nyomógombjával történik.

Indításkor a program kísérletet tesz mentett játékállás betöltésére, sikertelenség esetén kezdőállapotba inicializálja a játékot.

A képernyő három jól elkülöníthető részre szeparálható:

- Bal oldalt az infoC-s bagoly látható, erre kattintva szerezhetünk krediteket - ez ebben a játékban a pénznem. Fölötte két számláló látható, egyik a kátékos jelenlegi összes krediteinek számát, másik a másodpercenként automatikusan beáramló kreditek számát (későbbiekben kps, kredit-per-second). (Kezdetben mindkét érték nulla) A bagoly alatt egy manuális mentésre és egy a játékállás törlésére szolgáló nyomógomb található.
- Középen a megvásárolt épületek limitált vizuális reprezentációja látható. Az első néhány megvásárolt épület mindegyik típusból megjelenik, lassan feltöltve a helyet.
- Jobb oldalt található két oszlopnyi gomb, az, hogy ezekből mennyi látható, a játékállástól függ. A szélesebb gombok olyan "épületek" megvásárlását teszik lehetővé, melyek növelik a kps-t. A keskenyebb gombok - a fejlesztések - a velük egy sorban lévő épületekhez kapcsolódnak. A fejlesztések megvásárlásával az ahhoz tartozó épülettípus összes épülete megkétszerezi a hatékonyságát, megduplázza az adott épület által generált kps-t.

Mindkét oszlop gombjaira igaz továbbá az, hogy a kurzort fölöttük tartva részletesebb információt kapunk a tartalmukról, funkcionalitásukról, valamint a jelenlegi játékra gyakorolt hatásukról. Ilyen információ lehet az adott fejlesztés neve, leírása, hatása, vagy az épületek esetén az általuk generált kps külön-külön, összesen, valamint a teljes kps bevételhez viszonyított aránya százalékban.

A játékot az escape billentyű lenyomásával bármikor bezárhatjuk, ekkor a program kilépés előtt elmenti a játékállást. Mentés egyébként percenként automatikusan is történik, váratlanul leállított program esetén sem veszik el a haladás teljes egésze, legfeljebb az utolsó hatvan másodperc. Az s gomb megnyomásával egyébként bármikor manuálisan is elmenthetjük a játék pillanatnyi állását.

=-=-= infoClicker =-=-=
programozói dokumentáció

A program C nyelven SDL grafikával készült, szükséges környezete a GCC (*GNU Compiler Collection*), a futásához SDL könyvtár szükséges.

A **main.c** 3 részre különíthető el.

1) **Inicializálás**: Fájlok olvasása, adatok változókba töltése, képernyőre rajzolandó helyek kijelölése, forrás képfájlok szétszabdalása és elmentése.

2) **Futás**: A kurzor helyének, lenyomott gomboknak és a játékállásnak megfelelően rajzolás, adatok módosítása, megfelelő függvények hívása.

3) **Zárás**: Fájlok bezárása, esetleges vissza nem adott memória felszabadítása, textúrák, betűtípusok bezárása.

A program a **main.c** modulon kívül 4 másik modulra bomlik.

- **fajlok.c/fajlok.h**: A fájlkezelést végző függvények, itt találhatók. Ezek feladata többek között a játékállás mentése, indításkor betöltése, konstans adatok fájlból olvasása, rendszerezése.

```
void fajlnev (int i, char * fajlnev);
```

  - Ez a függvény egy X számból "upgrades/upgX.txt" stringet állít elő, amit elment a pointerként kapott stringbe.

```
void vonaltalanit (char * szoveg);
```

  - Egy pointerként átvett stringben az alsóvonás karaktereket szóközökre cseréli.

```
unsigned long int mostaniar (epuletek * epulet, int x);
```

  - Egy epuletek típusú tömb a struktúrában tárolt indexébe beolvassa a megfelelő nevű fájlból az X-edik fejlesztés adatait.

```
void jatekallas_mentese (double bgkredit, int gombdb, epuletek * epulet);
```

  - Egy "save.txt" fájlba írja a jelenlegi játékállást. Az adatokat, amiből az állás egyértelműen rekonstruálható a három paraméterként átvett változó elégségesen tárolja.

- **gombok.c/gombok.h**: Ebben a fájlban vannak a gombok működéséhez szükséges függvények. Ezek a függvények akkor hívódnak meg, amikor a gombok rajzolásával, és azok interakciójával akarunk foglalkozni. (pl: a gomb, amin a kurzor van legyen másmilyen, mint a többi, gombokra kattintás detektálása)

```
typedef struct epuletek {...} epuletek;
```

- Az épületek és a hozzájuk tartozó fejlesztések adatait tároló struktúra.

```
double dist (SDL_Point a, SDL_Point b);
```

- Két képernyőkoordináta távolságát számítja ki. Visszatérési értéke valós.

```
bool baglyon (SDL_Point kurzor);
```

- Azzal a logikai értékkel tér vissza, amely kifejezi, hogy a kurzor a baglyon van-e.

```
bool gombon (SDL_Point kurzor, SDL_Rect gomb);
```

- Azzal a logikai értékkel tér vissza, amely kifejezi, hogy a kurzor egy paraméterként megadott téglalapon belül van-e.

```
bool gomboknal (SDL_Point kurzor, SDL_Rect gombok, int gombdb);
```

- Azzal a logikai értékkel tér vissza, amely kifejezi, hogy a kurzor a feloldott gombok valamelyikén van-e.

```
double sumkps(epuletek * epulet);
```

- A megvásárolt épületek számának és fejlesztéseknek számából kiszámolja a másodpercenkénti kredittermelést.

- **rajzol.c/rajzol.h**: Ez a fájl az képernyőre rajzoláshoz használt segédfüggvényeket tárolja. (pl: képfelület létrehozása, USEREVENT-ek generálása, nem statikus megjelenítés kezelése)

```
void sdl_init (char const *felirat, int szeles, int magas, SDL_Window **pwindow, SDL_Renderer **prenderer);
```

- Létrehozza a játéktérként funkcionáló ablakot.

```
Uint32 idozit (Uint32 ms, void *param);
```

- Létrehoz egy időzítőt.

```
void forgat (double *i, int speed);
```

- A játék tickspeedjéhez viszonyított sebességgel egy 0 és 360 közötti értékre állítja az i változót.

```
void fenyrender (SDL_Renderer *renderer, SDL_Texture *gui, SDL_Rect feny, SDL_Rect fenyhely, int speed);
```

- Két világító effektet rajzol a bagoly köré, adott szögben elforgatva azt.

```
int minimum (int a, int b);
```

- Visszaadja két egész szám közül a kisebbet.

- **szovegek.c/szovegek.h**: Ebben a fájlban vannak azok a függvények, melyek a szövegek képernyõre írását teszik lehetõvé, valamint az ehhez tartozó struktúrák, láncolt lista definíciója. Az itt található függvények elengedhetetlenek a program szövegkiírásaihoz. (pl a kiírandó szám hármas tagolása, nagyságrendekre bontása, kiírandó szövegek igazítása ées a képernyõre renderelése mind itt van definiálva)

```
typedef enum align {center, left, right} align;
```

- A szovegki függvény utolsó paramétere, ettől függően a szöveg középre, balra vagy jobbra igazítva jelenik meg.

```
typedef struct sz {...} sz;
```

- Struktúra, mely egy szöveg színét, betűtípusát, tartalmát, helyét és méretét tárolja.

```
typedef struct Lebeges {...} lebeges;
```

- Láncolt lista, meyben egy szöveg adatai, egy számláló, amely a szöveg élettartamának eltárolására szolgál és a következő elemre mutató pointer van.

```
void szovegki (SDL_Renderer *renderer, SDL_Surface *felirat, SDL_Texture *felirat_t, sz szoveg, align igazitas);
```

- Egy kiírandó szöveg összes paramétere tudatában megjeleníti azt a képernyőn.

```
sz feltolt (int red, int green, int blue, TTF_Font * betutipus, int x, int y, int h, int w);
```

- Egy sz típusú struktúrát inicializál. (szín, betűtípus, hely, méret) A tartalom külön adandó hozzá a szovegki függvény meghívása előtt!

```
void csoportosit (unsigned long int szam, char * tomb);
```

- Egy egész számot hármas tagolásúra bont, majd a pointerként átvett stringbe írja szövegként.

– **szovegek.c/szovegek.h** (folytatás):

```
char * hogyirki (unsigned long long int kredit, double bgkredit,
char * szoveg, char * prefix, char * suffix, bool tortkent, bool*
nocount);
```

- A csoportosít függvény segítségével véglegesíti a kiírandó szám + esetleges elő- vagy utótagként megadott szöveg kombinációt.

```
lebeges* klikklista (lebeges ** elso);
```

- Létrehoz az átvett lista végén egy új elemet, amely címét visszatérési értékként visszaadja.

```
void elsoelemtorol (lebeges ** elso);
```

- Törli egy láncolt lista első elemét.