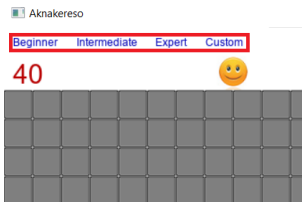
Aknakereső

Programozói dokumentáció

- **valtozotipusok.h:** a programban felhasznált struktúrák definíciójait tartalmazza
  - struct ablak_meretei: egy (SDL) ablak (int) szélesség és (int) magasság változót tartalmaz.
  - struct koordinatak: egy (Sint32): x és egy (Sint32): y koordinátát tartalmaz.
  - struct palya_tombok: A pálya minden mezőjét egy ilyen struktúrában lehet tárolni és így a pálya tárolásához csak egy ilyen struktúrákból álló tömbre van szükség.
    i. (int) palya: az adott mezőn milyen szám van (hány bomba van körülötte).
    ii. (bool) bombak_helyei: az adott mező bomba-e.
    iii. (bool) lathato_e: az adott mezőt felderítette-e már a játékos.
    iv. (bool) zaszlo: be van e zászlózva az adott mező.
  - struct jatek_adatai: Az éppen folyamatban lévő játékhoz szükséges összes adatot tudja tárolni.
    i. (int) palya_szelessege: Mezők száma egy sorban.
    ii. (int) palya_magassaga: Mezők száma egy oszlopban.
    iii. (int) bombak_szama: Hány bomba van a pályán.
    iv. (palya_tombok*) lefoglal: Ezzel foglalja le a program a pálya adatait tároló tömböt
    v. (palya_tombok**) palya: Az elemei a lefoglalt tömb megfelelő helyeire mutatnak, így egy 2 dimenziós tömbként kezelhetjük.
    vi. (int) nehezsegi_szint: A játék nehézségi szintjét tárolja.
    vii. (int) idomero: A kezdés óta eltelt időt tárolja.
    viii. (int) rekord: Az előre meghatározott nehézségi szinteken ez tárolja az addigi rekordot.
    ix. (int) zaszlozott_mezok_szama: A zászlózott mezők számát tárolja.

- **grafika.c:** minden, a játék grafikus megjelenítéséhez szükséges függvényt ez tartalmaz
  - sdl_init(): Létrehoz egy SDL ablakot, és annak a nevét, szélességét, magasságát, a hozzá tartozó SDL ablakot (SDL_Window**), és SDL renderert (SDL_Renderer**) kéri.
  - nehezsegi_szintek_kiirasa(): A fő ablakhoz tartozó SDL renderert kéri és ez írja ki a nehézségi szinteket az ablak tetejére.
  - mosi_szomi_fely_kirjaz(): Az ablak közepén lévő (a mellékelt képen) mosolygó, vagy szomorú fejet rajzolja ki, és az SDL renderert, a fő ablak méreteit tartalmazó struktúrát és egy boolt kér, ami azt határozzam meg hogy vesztett-e a játékos.



  - bombak_szamanak_kirajzolasa(): az SDL renderert és a játék adatait tartalmazó struktúrát kéri, és kiírja az ablak bal felső sarkába a pályán lévő bombák száma és a zászlózott mezők számának különbségét.
  - ido_kirajzolasa(): az SDL renderert, a játék adatait, és az ablak méreteit kéri és az ablak jobb felső sarkába kirajzolja a játék kezdete óta eltelt időt.
  - kirajzolas(): A játék adatait, SDL renderert és az ablak méreteit kéri. Ez a függvény kirajzolja az egész ablakot és azon belül mindent.
  - ablak_mereteinek_kiszamolasa(): A játék adatait és az ablak adatait tartalmazó struktúrára mutató pointert kér, és kiszámolja, mekkorának kell lennie az ablaknak az adott játékmódhoz

- vesztett_kirajzolas():A játék adatait, SDL renderert és az ablak méreteit kéri és kirajzolja az egész ablakot, amikor a játékos elvesztette a játékot
- custom_kirajz():a custom nehézségi szint beállításához külön ablak nyílik, és ennek a megjelenítését végzi a függvény, egy pálya szélességét, magasságát és a rajta lévő bombák számát tartalmazó int tömböt és az ehhez az ablakoz tartozó SDL renderert kell adni.
- uj_rekord_ablak(): Ha a játékos új rekordot dönt, akkor ez a függvény nyit egy új ablakot és ebbe kiírja neki ezt, a játék adatait kéri.

❖ **palya_elokeszitese.c:** egy új pálya létrehozását és előkészítését végző függvényeket tartalmazza
- custom_beker(): a játék adataira mutató pointert kéri, és a custom nehézségi szint adatait kéri be, ehhez nyit egy új ablakot, majd ezeket beleírja a kapott struktúrába.
- palya_adatainak_bekerese() A játék adataira mutató pointert kéri és az új nehézségi szint adatait kéri be: vagy txt-ből, vagy a játékostól custom nehézségi szint esetén a custom_beker() függvénnyel. A játék adataira mutató pointert kéri.
- uj_rekord(): A játék adataira mutató pointert kéri, és elmenti az új rekordot az előre meghatározott nehézségi szintekhez. A rekord ugyan abban a txt-ben van elmentve, mint a pálya összes többi adata.
- palya_tombjeinek_lefoglalasa(): A játék adataira mutató pointert kéri, és lefoglalja a pálya tárolásához szükséges tömböket.
- palya_tombjeinek_falszabaditasa(): A játék adataira mutató pointert kéri és felszabadítja pálya tárolásához szükséges tömböket.
- ures_palya_feltoltese(): A játék adataira mutató pointert kéri, és a pályát feltölti alap értékekkel: üres pályát hoz létre.
- bombak_generalasa(): A játék adataira mutató pointert kéri és lerakja a pályára a megfelelő mennyiségű bombát.
- bombak_szama_korulotte():A játék adataira mutató pointert kéri és egy adott mező koordinátáit a tároló tömbben. és visszaadja, hogy hány bomba van körülötte.
- szamok_helyrerakasa():A játék adataira mutató pointert kéri és a pálya mezőin lévő számokat határozza meg és menti el.

❖ **lathatosag_terjedese.c**
- lathatosag_terjedese(): Egy mező koordinátáit (az azt tároló tömbben), és a játék adataira mutató pointert kéri. Ha egy mező láthatóvá vállik megnézi, hogy a körülötte lévőknek is azzá kell-e válnia.

❖ **jatekmenet.c:** a közvetlenül a játék menetéhez kapcsolódó függvényeket tartalmazza
- jatek_ujrakezdese(): A játék adataira mutató pointert, az ablak méreteirte mutató pointert és egy boolt ami azt dönti el, hogy a játékot csak újrakezdte a játkos, vagy úgy nehézségi szintet is választott. Új játék kezdéséhez szükséges függvényeket gyűjti össze, hívja meg.
- Uint32 idozit(): Egy időzítő, ami paraméterként kapott időközönként generál egy SDL_USEREVENT-et, az idő méréséhez kell.
- tomb_koordinatakra(): Koordinnáta struktúrára mutató pointert kér, és az ablakon az egér koordinátáiból csinál a pálya tömbjéhez indexet.

- menure_kattintott(): A játék adataira mutató pointert, a felhasználó által kattintott koordinátákat, az ablak méreteit, és az SDL ablakra mutató pointert kéri. Megváltoztatja a nehézségi szintet, ha a felhasználó arra kattintott.
- mezore_kattintott():A kattintott koordinátákat, a játék adataira mutató pointert, az SDL rendererre mutató pointert és az ablak méreteit kéri. Kezeli, ha a felhasználó egy mezőre kattintott.
- nyert_e(): A játék adatait kéri, és bool értékben visszaadja, hogy a játékos megnyerte-e a játékot.
- ha_nyert(): A játék adataira mutató pointert, az SDL renderer pointerét, és az ablak méreteit kéri, és kezeli, ha a játékos nyert.
- zaszlozott():a kattintott koordinátákat, és a játék adataira mutató pointert kéri. Kezeli, ha a játékos be, vagy ki zászlózott egy mezőt.

- **main.c:** a program lebonyolítását végzi
  - int main():
    i. Létrehozza a játék menetéhez szükséges változókat.
    ii. Létrehozza az SDL ablakot, amiben a játék fog zajlani.
    iii. Egy kilépésig egy while ciklusban vár egy SDL event-et, amiről egy switch-ben meghatározza mit jelent
      1. ha kattint:
        - bal egérgomb:
          - A menüben kattintott-e valahova: nehézségi szintet váltott-e.
          - A mosolygós arcra kattintott-e: új játék kezdése.
          - A pályán kattintott-e valahova: A játék játszása.
        - jobb egérgomb:
          - A pályán zászlózott-e valamit.
      2. generált userevent:
          - Egy másodperc eltelt.
      3. kilépett-e:
          - Ciklusfeltétel hamissá tétele.

      Ezek után kirajzolja a pályát.

    iv. A program végén bezárja az ablakot, felszabadítja a dinamikus tömböket.
- A program ezeken kívül felhasználja a(z):
  - SDL2_gfx.dll
  - SDL2.dll
  - libpng16-16.dll
  - libfreetype-6.dll
  - SDL2_ttf.dll
  - zlib1.dll
  - SDL2_mixer.dll

  függvénykönyvtárakat, és ezekhez a sdl2-config file-t, továbbá a standard c headereket.
- A programban ezen kívül a grafikus megjelenítéshez használt 0-8, default, mosifely, szomifely, mine és flag.png és a pályák adatainak tárolásához Beginner, Intermediate, Expert.txt file-okat használja fel.
- Ezen kívül az arial.ttf betűtípust használja fel a szövegek megjelenítéséhez.

Aknakereső

felhasználói dokumentáció

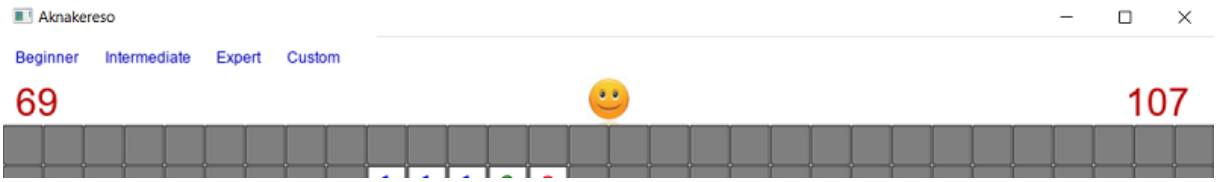
Az aknakereső egy játék, aminek a célja, hogy a pálya összes olyan mezőjét felfedezzük, amin nincsen akna.

Az összes mező alapvetően lefedett állapotban van.

Ha egy mezőre a jobb egérgombbal rákattintunk, akkor azt bezászlózzuk: ezzel azt jelezve, hogy bombának véljük.

Ha a bal egérgombbal kattintunk egy mezőre, akkor feltárjuk az adott mezőt: ha a feltárt mező bomba: felrobbanunk, elvesztettük a játékot, ha nem bomba a feltárt mező, akkor a rajta lévő szám azt jelzi, hogy hány darab bomba van a körülötte lévő mezőkön, ha üres az értelem szerűen 0 bombát jelez, de a játék magától felfedezi az üres mezők körülötti területet.

A játékot akkor nyertük meg, ha az összes, nem bomba, mezőt felfedtük.



Az ablak bal felső sarkától kezdődően ki vannak írva a különböző nehézségi szintek:

- Beginner: 9 x 9-es pálya 10 bombával.
- Intermediate: 16x16-os pálya 40 bombával.
- Expert: 16x30-as pálya 99 bombával.
- Custom: A játékos beállíthatja a pálya méretét, és a rajta lévő bombamennyiséget.

A feliratokra kattintva változtathatjuk meg a nehézségi szintet és kezdhetünk új játékot.
Ezek alatt a még nem felfedezett bombák számát találjuk, vele egy sorban, középen egy mosolygós arcot, amire kattintva új játékot kezdhetünk, és jobb szélen a játék kezdete óta eltelt időt találjuk (másodpercekben).