Szintvonalak meghatározása a marching squares algoritmusssal

Jegyzőkönyv

Beadandó

Írta: Csabai Bence

csabai.bence@gmail.com

Kivonat

A program amelyet készítettem egy domborzati térkép szintvonalainak meghatározására való. Ahhoz, hogy ezt elérjem, az úgynevezett *marching squares* módszert módszert implementáltam a programba. Emellett a program képes megmondani egy pontról, hogy belül van-e egy adott szintvonalon. Ezt a *winding number* nevű technikával állapítom meg. Ez a dokumentum egy jegyzőkönyve a program készülésének folyamatáról.

Tartalomjegyzék

1. Bevezetés **1**

2. Környezet **1**

2.1. Vázlatpontos összefoglaló 1

2.2. Munkamenet I.: Telepítés 1

2.2.1. Visual Studio és VSCode 1

2.2.2. Anaconda, Python és Jupyter Notebook 3

2.2.3. git 3

2.3. Munkamenet II.: Előkészítés 3

3. Program Elvi Működése **4**

3.1. Beolvasás és előkészítés 4

3.1.1. Beolvasás 4

3.1.2. Adatelőkészítés 4

3.2. Marching Squares 5

3.3. Winding Number 6

4. Problémák és megoldásaik **7**

Ábrák jegyzéke

2.2.1.A Visual Studio és a Visual Studio Code böngészőből való letöltése 2
2.2.2.Az 'Extensions' fülből bármilyen bővítmény könnyen letölthető 2
2.2.3.Az Anaconda Navigator-ből egyszerűen csomagokat telepíteni a Pythonhoz 3
2.3.1.Github repo első kinézete 4
2.3.2.A hasznos Quick Start parans 4
3.2.1.Kontúr vonal esetek 5
3.2.2.Kontúrok a Hargita hegységben 6

1. Bevezetés

Ez a jegyzőkönyv leírja a folyamatát annak, hogyan, milyen eszközök használatával készítettem el egy szintvonal meghatározó programot. Ezen kívül leírom a program elvi működését, azt hogy milyen problémákba ütköztem és azokat hogyan oldottam meg.

2. Környezet

2.1. Vázlatpontos összefoglaló

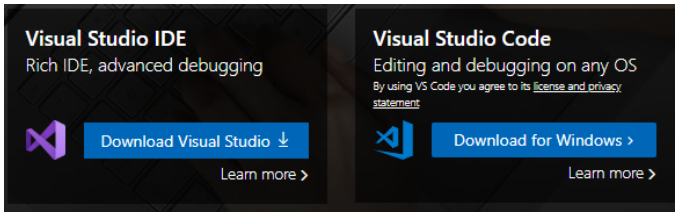
- Microsoft Windows 10.0.17763.0
- Microsoft Visual C++
- Visual Studio Code
  - Microsoft C/C++ extension
  - CMake extension
  - CMake Tools extension
- Anaconda
- Python
- Jupyter Notebook
- git

2.2. Munkamenet I.: Telepítés

2.2.1. Visual Studio és VSCode

A programom elkészítéséhez több mindent telepítenem kellett. A C++ kódom fejlesztéséhez és fordításához a Microsoft Visual Studiot illetve a Visual Studio Codeot használtam. Ezekkel a programokkal már az órákról volt valamilyen gyakorlatom, ezért választottam őket. Ennek telepítéséhez le kellett tölteni a Visual Studiot, illetve a VSCodeot, amelyet a https://visualstudio.microsoft.com/

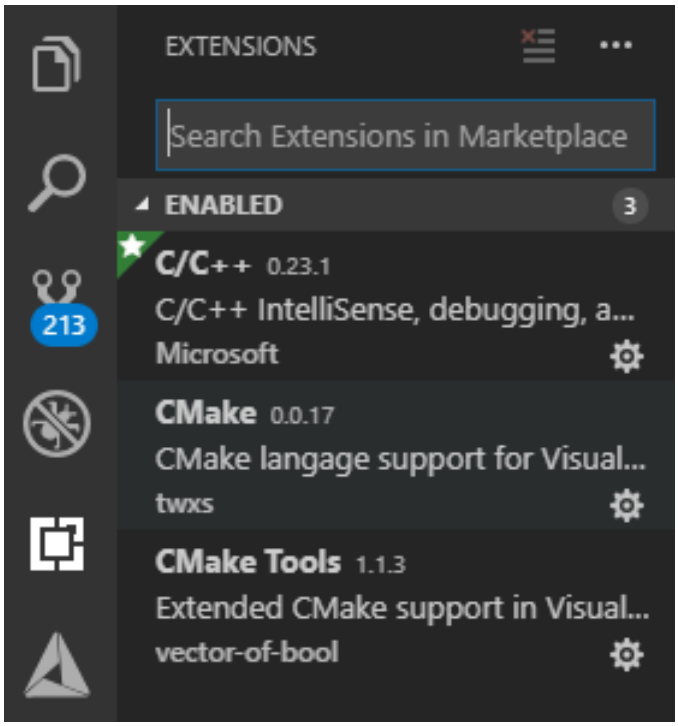
oldalon tudtam megtenni. A Visual Studion belül emellett telepíteni kellett a megfelelő Microsoft Visual C++ fordító csomagot, ami szerencsére a felhasználóbarát grafikus felületnek köszönhetően könnyen ment.



2.2.1. ábra. A Visual Studio és a Visual Studio Code böngészőből való letöltése

Amint ez megvolt, VSCodeon belül, az Extensions fülbe belépve könnyen telepíthető volt a szükséges 3 bővítmény: C/C++, CMake, CMake Tools:

- C/C++ extension: IntelliSense, kód átláthatóság, debugolás
- CMake extension: CMake nyelv támogatás VSCodeba
- CMake Tools extension: Kibővített Cmake támogatás



2.2.2. ábra. Az 'Extensions' fülből bármilyen bővítmény könnyen letölthető

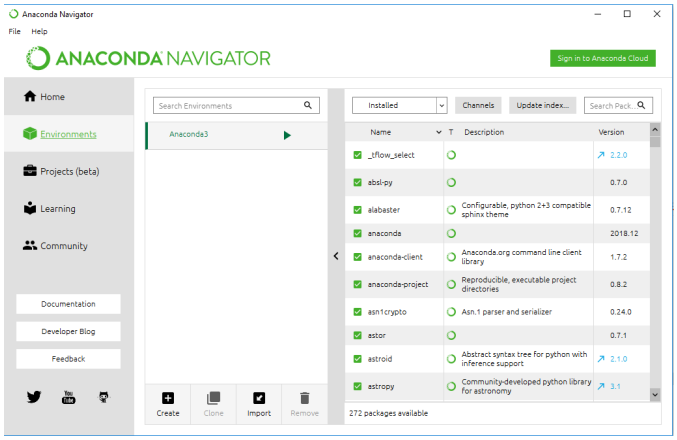
A Visual Studio Code beállításain sokat nem kellett szerencsére változtatni.

2.2.2. Anaconda, Python és Jupyter Notebook

Ahhoz hogy ellenőrizzem, megfelelően futott-e le a program, illetve hogy lássam mik a hibák, Jupyter Notebookban Pythont használva rajzoltam ki bizonyos ábrákat. Ezeket az eszközöket korábban is használtam már, így elérhetőek voltak a számítógépemen.

A Pythont és a Jupyter Notebookot az Anacondán keresztül telepítettem. Rajta keresztül könnyen kezelhető a Python nyelv, beleértve csomagok gyors és egyszerű kezelését is.

Az Anaconda a https://www.anaconda.com/distribution/ címről tölthető le.



2.2.3. ábra. Az Anaconda Navigator-ből egyszerűen csomagokat telepíteni a Pythonhoz

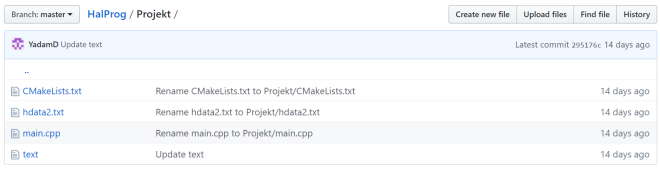
2.2.3. git

A verziókezeléshez gitet illetve GitHubot használtam. Ehhez csak a https://git-scm.com/downloads oldalról le kellett töltenem a gitet és utána már tudtam parancssorból használni a git által kínált featureöket.

2.3. Munkamenet II.: Előkészítés

Úgy döntöttem, hogy a projektet a már meglévő Haladó Alkalmazott Programozás Github repomba fogom rakni. Ehhez a számítógépemre már leklónozott repoban létrehoztam egy mappát "Projekt" néven.

Ebbe letöltöttem a gyakorlatvezetőm által elküldött szükséges adatfájlokat, valamint a VSCode-ban a CMake Quick Start parancsának segítségével létrehoztam egy CMakeLists.txt-t valamint egy main.cpp-t. Ezután elkezdhettem dolgozni a kódon és a projekten.



2.3.1. ábra. Github repo első kinézete

2.3.2. ábra. A hasznos Quick Start parans

3. Program Elvi Működése

Alábbiakban leírom a program elvi és technikai működését

3.1. Beolvasás és előkészítés

Mielőtt a programunk elkezdheti végezni a fő feladatát szükségünk van jól előkészített adatokra. Ezeket az adatokat megfelelő módon be kell olvasni és preparálnunk kell az adatokat, melyeken a kontúrvonalakra kíváncsiak vagyunk.

3.1.1. Beolvasás

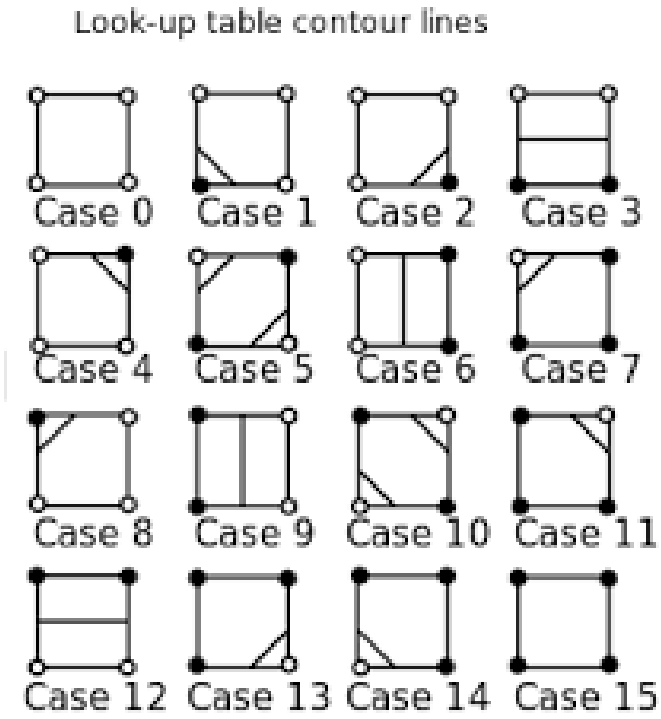
A programot természetesen az adatok beolvasásával kezdjük. Az adatok egy fileban vannak, ennek megyek végig sorain hogy a programom tudja használni a benne lévő adatokat. Fontos, hogy a fájl csak valós számokat tartalmazzon egy n*n-s formátumban, különben a beolvasás lehetetlen lesz. Az adatok tárolására egy korábban létrehozott négyzetes mátrixosztályt használok.

3.1.2. Adatelőkészítés

A 3.2-ben leírt marching squares algoritmus eredetileg nem valós számokra, hanem 0-kra és 1-ekre működik csak, éppen ezért a beolvasott adatokat ilyen formára kell fordítanunk. Ehhez egy függvényt írtam, mely egy adott adathalmazban megvizsgálja, hogy egy elem nagyobb-e mint az előre meghatározott határ. Ha igen, egy új, az eredetivel megegyező méretű táblában erre a helyre 1-est, ha pedig nem, akkor 0-st ír.

3.2. Marching Squares

A marching squares egy számítógépes grafikai algoritmus amu kontúrokat generál egy két dimenziós skalármezőhöz (téglalap alakú tömbje különálló számoknak) [1] Az eljárás lényege, hogy a korábban 0/1-essé alakított adatokon végigmegy, és megvizsgálja az összes pontnégyest, ahogy a 3.2.1 ábrán látható.



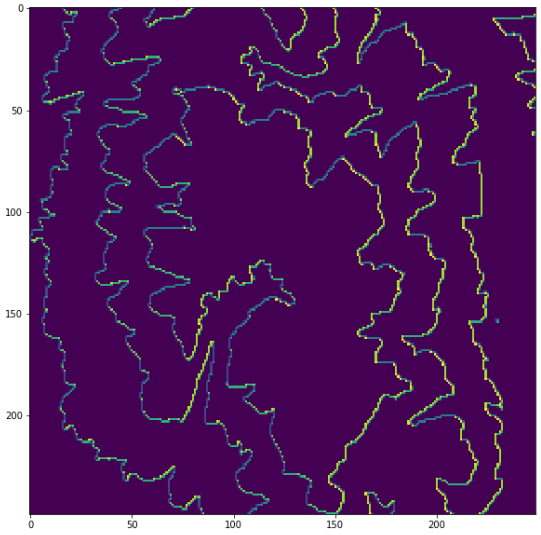
3.2.1. ábra. Kontúr vonal esetek

Ezeknek 16 különböző kombinációjuk van, így ezt megfigyelve tudja eldönteni a módszer, hogy melyik élet kell berajzolnia. Egyszerűen ez úgy volt leprogramozható, hogy vesszük sorban a sarkokat, és mindegyiknek értékét beszorozzuk egy kettőhatvánnyal, majd ezeket összeadjuk:

$$val = 8a_1 + 4 \cdot a_2 + 2 \cdot a_4 + 1 \cdot a_3 \tag{3.1}$$

Így minden kombinációra egyedi érték jön ki. Ha erre írunk egy függvényt, kapunk egy $(n-1) \times (k-1)$ méretű mátrixot, ahol már az adott szintvonal be van jelölve az elemek értéke által.

Ezután a 15-ösöket 0-sra cserélve (lényegük ugyan az, vonal nem rajzolódik) és több szintvonal mátrixot összeadva, kaphatunk egy kész térképet, amin már több szintvonal is be van jelölve. A 3.2.2 ábrán a különböző árnyalatú pixelek a 15 esetet reprezentálják.



3.2.2. ábra. Kontúrok a Hargita hegységben

3.3. Winding Number

Végső featureként a program a winding number nevű technikát használva meg tudja állapítani egy adott pontról, hogy egy adott kontúron belül van-e. A metódus lényege, hogy összekötjük a pontunk a kontúr egy pontjával, körbehaladunk a kontúrvonalon és mérjük a szöget. Amennyiben körbeérünk és a pont a kontúron belül van, a szög 2π lesz.

4. Problémák és megoldásaik

Mind az előkészületek mind a kódolás közben akadtak problémák, melyeket meg kellett oldani.

Az első akadály a Visual Studio Code-al adódott, amikor is nem akart működni a CMake, se a C++ fordító. Miután nagyon sok mindent kipróbáltam, az elérési útvonalak manuális átírásaitól kezdve az újratelepítésig, észrevettem, hogy egy másik felhasználó a számítógépre már korábban feltelepítette a Visual Studiot és a VSCode-ot nem megfelelő beállításokkal és a két telepítés összeakadt. Így a probléma megoldása végül az lett, hogy letöröltem az összes verziót és amikor már csak egy maradt a gépen, akkor végre működött.

Egy másik gond ami sok fejfájást okozott, már a kódban volt. Megírtam a programot, úgy éreztem elméletben mindent jól megcsináltam de nem akart működni a program. Ez után gondoltam úgy, hogy kiplotolom Pythonban az eredeti és a beolvasott mátrixot is. Kiderült, hogy csupán a beolvasásnál fel volt cserélve két index ezért a beolvasott térkép 90 fokkal el volt forgatva. A vizuális visszajelzés nagyban segítette innentől az ellenőrzést.

Eredetileg a mátrixokat és vektorok bevezetéséhez pointereket használtam. Gyakorlatvezetőm felhívta a figyelmemet, hogy a korábban megírt mátrix osztályt használhatnám ehelyett. Miután beiktattam ezt a változást, a programom futásideje jelentősen gyorsabb lett (6-7 mpről lecsökkent kevesebb mint 1-re)

Hivatkozások

[1] Wikipedia *Marching squares*.
`https://en.wikipedia.org/wiki/Marching_squares`