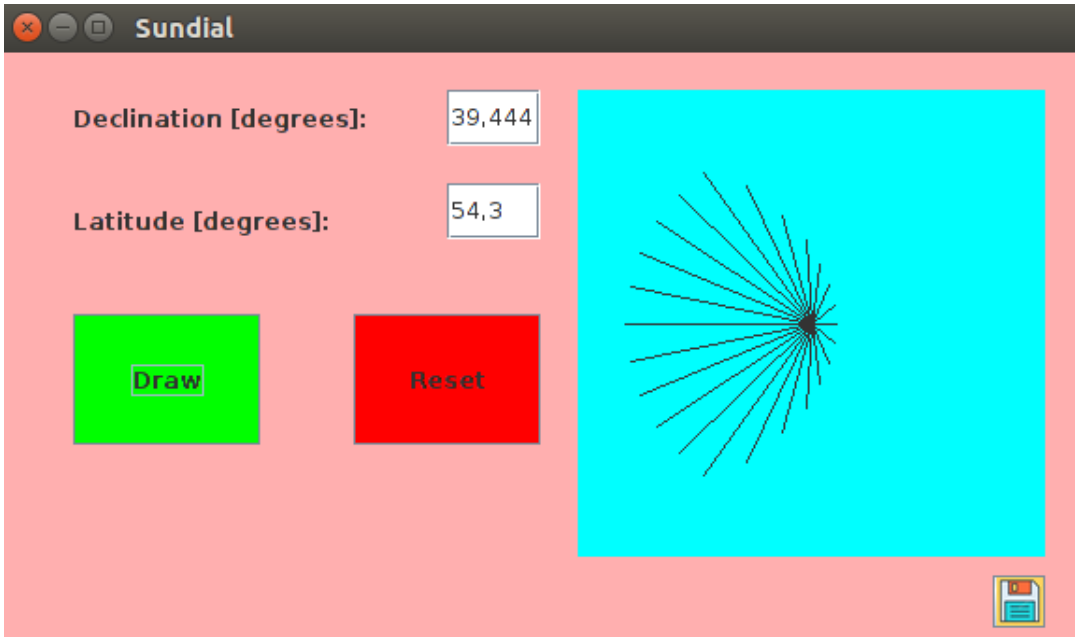
# A kutatómunka információs eszközei
# Git, CMake

Horváth Anna

May 2018



# 1 Sundial

## 1.1 A program ismertetése

A Sundial program méretarányosan kirajzolja egy a képernyőre merőleges bot árnyékát napkeltétől napnyugtáig. A bemeneti paraméterek a deklináció, mellyel azt adhatjuk meg, hogy az év mely napja van, illetve a földrajzi szélesség, ahol a megfigyelő tartózkodik. A program két csillágaszati koordináta-rendszer közti átjárást biztosít. Az ekvatoriális koordinátarendszer koordinátái:

- deklináció: a Nap égi egyenlítővel (az éggömbre vetített egyenlítővel) bezárt szöge, melyből meghatározhatók az év napjai.
- óraszög: megmutatja, hogy mennyi ideje delelt a Nap

A horizontális koordináta-rendszer koordinátái:

- magasság: A Nap milyen magasan jár a horizonthoz képest
- azimut szög: A Nap milyen szélességben van a horizonttal párhuzamos síkban

A program a deklinációt és a szélességi fokot bekérve rajzolja ki az árnyék járását. Meggondolható, hogy például -23,5 fok deklináció esetén (a téli napforduló idején), az északi sarkon állva (90 fok) nem látjuk felkelni a Napot. Ugyanígy 23,5 fok (nyári napforduló) esetén sem a déli sarkról (-90 fok). Ezt kiírja a program. Egyéb esetben 24 vonalat rajzol ki, a 24 óraszögnek megfelelően. (Ezek közt lehet 0 hosszúságú is). Fordított esetben, például 23,5 fok deklináció és 90 fok földrajzi szélesség esetén nem nyugszik le a Nap, ez látható az ábrán.

A program *Java* nyelven íródott, két forrásfile tartozik hozzá, a *Sundial.java* tartalmazza a grafikus komponenseket és a program belépési pontját (*main*), a *SunEvent.java* pedig az eseményeket kezeli. A koordináta-rendszerek közti áttérést az átváltási szabályoknak megfelelően végzi a *shadow* függvény. Ezután a magasságból kiszámítja, hogy milyen hosszú lesz az árnyék, az azimut szög pedig megadja az árnyék irányát. Ezt a két adatot az egyes óraszögekre egy kétdimenziós tömbben (*arr*) tárolja. Az első dimenzió az árnyék hossza, a második annak iránya (mely az azimut szöggel egyenlő).

A jobb alsó sarokban lévő ikonra kattintva ezeket tudjuk kimenteni egy *ShadowLength_Azimuth.dat* nevű file-ba. Ezen kívűl még két gombot tartalmaz a program, az egyik a kirajzolásra, a másik a bevitt és kirajzolt adatok törlésére szolgál.

## 1.2 Git és fejlesztés

A feladatot a git megismerésével kezdtem. Angol és magyar nyelvű online források alapján gyakorló feladatokat végeztem. Alább csak a fontosabb parancsokat listáztam. A git egy verziókezelő rendszer, a gyökérkönyvtár (*working directory*) tartalmazza a *repository*-t (.git mappa), melyben a különböző ágak (*branch*-ek) találhatók. A kitüntetett főág a *master branch*. A szerveren található a

*remote repo*, saját gépen a *local repo*. A feladat megkezdéséhez létrehoztam a *repository*-t a *github.com* felületen. A *local repo* létrehozásához ezt klónoztam:

$$\$\ git\ clone\ https://github.com/t536ee/kutinfo$$

mely parancs az aktuális mappában létrehozta a *local repo*t (egy kutinfo nevű mappát). A *remote repo*-ra való feltöltéshez a *push*, onnan letöltéshez a *pull* parancsot használjuk. A *master*hez *pull request*tel kell ellátni a *branch*et.

$$\$git\ branch\ --list$$

listázza a brancheket, *-gal jelöli, melyiken vagyunk éppen,

$$\$git\ checkout\ -b\ dev$$

létrehozza dev branchet és odalép.($*git checkout dev*: csak odalép.) Az első elkészített programegység a $Ma.java$, mely tartalmazza az algoritmust. Egy darab $JFrame$-et tartalmaz, melyre kirajzolódik az ábra. Ezen kívül nincsenek grafikus elemek benne. Beleteszem a Ma.java-t a kutinfo mappába, majd a

$$\$git\ add\ Ma.java$$

paranccsal hozzáadom a verziókezeléshez (*git add.* mindent hozzáad), majd commitolok.

$$\$git\ commit\ -m\ 'commit\ szovege'$$

$$\$git\ push\ origin\ dev$$

aktuális branchet pusholja a remote (origin) dev branchébe, felhasználónevet (t536ee) és jelszót kér. Eztán a remote repon megjelenik a dev branch is, és benne a Ma.java is.

A fejlesztést ezután a grafikus felület létrehozásával folytattam. A $Sundial.java$ tartalmaz egy $JFrame$-et gombokkal, szövegdobozokkal és címkékkel. Ezen kívül tartalmazza a program belépési pontját, viszont nem tartalmazza az algoritmust, és nem reagál felhasználói interakcióra.

Ezután töltöttem fel a *Sundial* actionListener-rel ellátott verzióját. A fájl frissítéséhez a szerkesztett fájlt addoltam, commitoltam, pusholtam. Feltöltöttem a $SunEvent.java$ első verzióját, mely a draw gomb nyomására reagálva kiírja, ha a nap nem kel fel.

Egyéb parancsok, melyeket használtam a feladat során:
git push origin – delete branchName: remote-on lévő branch törlése
git branch r: listázza a remote-on lévő brancheket
git init: könyvtár inicializálása repo-ként

# 2 CMake

A CMake használatával nem jártam sikerrel, dokumentálom a próbálkozásaimat. Installáltam a CMake-et, és létrehoztam a $CMakeLists.txt$ fájlt, és a *test.cpp*

forrásfile-t, mely egy egyszerű Hello World program c++ nyelven. A file-ok tartalma:

```
//test.cpp
#include <iostream>
using namespace std;
int main(void)
{
     cout << "Hello World" << endl;
     return(0);
}
```

```
# CMakeLists.txt
cmake_minimum_required(VERSION 2.8)

project(hello)

set(CMAKE_BINARY_DIR $/home/anna/cm/bin)
set(EXECUTABLE_OUTPUT_PATH $/home/anna/cm)
set(LIBRARY_OUTPUT_PATH $/home/anna/cm)

include_directories("$/home/anna/cm")

add_executable(hello $/home/anna/cm/test.cpp)
```

A fordítást a következő parancsokkal próbáltam elvégezni:

$$\$cmake - H. - Bbuild$$

$$\$cmake - -buildbuild - - - j3$$

melyekre a következő hibákat kaptam:

```
CMake Error: CMake can not determine linker language for target: hello
CMake Error: Cannot determine link language for target "hello".
```

illetve

```
make[2]: *** No rule to make target 'CMakeFiles/hello.dir/build'.
Stop.
CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/hello.dir/all' failed
make[1]: *** [CMakeFiles/hello.dir/all] Error 2
Makefile:83: recipe for target 'all' failed
make: *** [all] Error 2
```