# Pět hodin poprvé

Někdy týden po zadání jsem se rozhodl začít dělat semestrálku z UPG. Začal jsem s načítáním dat ze souborů do pole. Pak jsem se bohužel na 5 týdnů zasekl, protože jsem nevěděl, jak zobrazit bitmapový obrázek a jak s ním pracovat. Až někdy 27. března den po 6. cvičení jsem se zase pustil do práce a trošku se semestrálkou hnul.

## Co jsem udělal?

- Naučení se Gitem – Úspěšné naklonování souborů do mého počítače.
- Načtení dat – pomocí metody nacitaniDat(), metoda načte hodnoty ze souboru do pole.
- Vytvoření obrázku – metoda createPicture() nejdříve ověří že data v poli se dají převést na barvu. Jestliže se hodnoty nedají převést na barvu (jsou větší než 255) provede se metoda upraveniHodnot(), která upraví data, tak aby největší hodnota nebyla větší jak 255 a zároveň se nezměnil poměr mezi barvami. Následně se data pomocí cyklu vkládají do obrázku.
- Vykreslení obrázku – metoda drawPicture() do které se jako parametr vloží velikost okna, zajistí že se pomocí scalingu vykreslí obrázek. Pod obrázkem je černý čtverec, pro případ, že má okno jiný poměr stran než obrázek.
- Zajištění minimální velikosti obrázku – nastaveno při počátečním nastavení velikosti drawingPanelu. Pomocí poměru stran se zjistí, která strana se musí nastavit na minimální velikost, aby měla i druhá strana minimální velikost a zároveň byl zachován poměr stran.

# Pět hodin podruhé

V dalších pěti hodinách jsem implementoval bilineární interpolaci, dosud byla interpolace pouze pomocí nejbližšího souseda. Chtěl jsem si ušetřit práci, a tak jsem projížděl internet, jestli neexistuje nějaká metoda přímo v Javě, kterou bych mohl použít. Naštěstí jsem takovou metodu našel, takže bilineární interpolace nabyla velkým problémem. Dále jsem se pustil do šipek, které už byli mnohem těžší. Šipky nesmějí za žádných okolností v mapy ven a musejí mít stejnou velikost. K vyřešení tohoto problému jsem modifikoval kód ze třetího cvičení. Jako poslední jsem přidělal metody na vypočítávání nevyššího a nejnižšího bodu a maximálního stoupání.

## Co jsem udělal?

- Bilineární interpolace – implementována do metody drawPicture(), kód a porovnání změny níže.

```
g2.setRenderingHint(RenderingHints.KEY_INTERPOLATION,RenderingHints.VALUE_INTERPOLATION_BILINEAR);
```

**Interpolace pomocí nejbližšího souseda**

**Bilineární interpolace**

- Kreslení šipek–vytvoření metody drawArrow()
  - Nevykreslování mimo mapu – když konec šipky směřuje k polovině okna -> šipka se nevykreslí mimo mapu. Začal jsem tedy počítat směrový vektor šipky, který vede z „počátečního“ bodu (kam šipka ukazuje) do středu mapy.
  - Stejná velikost všech šipek – ze směrového vektoru jsem udělal jednotkový směrový vektor a ten jsem vynásobil délkou šipky (v mém případě 50px)
  - Scaling šipky – mapa může být trochu zvětšená, nebo zmenšená. Musím tedy zařídit, aby šipka vždy ukazovala na stejné místo na mapě. Hodnoty bodu, na který šipka ukazuje tedy musím upravit stejně jako obrázek:
```
x1 = (x1*scale) + startX;
y1 = (y1*scale) + startY;
```
- Nalezeni lokálních extrém
  - Nejvyšší bod – metoda getMax(), prochází se celé pole a porovnává jestli dosud největší zjištěná hodnota je větší než aktuální vstupní.
  - Nejnižší bod – metoda getMin(), stejná implementace jako při hledání maxima
  - Maximální stoupání – metoda getMaxStoupani(), nejdříve se porovnává vertikální absolutní rozdíl mezi dvěmi sousedními indexy. Byl zde problém s tím, že se porovnávali i indexy přes „hranice“ obrázku, tj. první a poslední řádku. Toto jsem vyřešil podmínkou, že se tyto dvě hodnoty nesmějí porovnávat. Horizontální je provedeno obdobným způsobem je se porovnávají dvě hodnoty nad sebou.

## Pět hodin potřetí

Práce je skoro hotova, stačí doladit pár detailů. Šipky se vykreslují blízko sebe. Rozhodnout se jakou barvu budou mít šipky. Doplnit do složky bin složku data, která se z nějakého důvodu smazala. Otestovat, jestli aplikace běží, jak má (spuštění před Run.cmd s argumenty). Nakonec jsem našel ještě chyby při zobrazování oken, které jsem následně opravil.

### Co jsem udělal?

- Šipky – šipky nevypadají tak, jak bych si představoval.
  - Šipky jsou blízko sebe - Šipky se na některých mapách vykreslovali blízko sebe (např. na mapě random), vyřešil jsem to tak, že jsem do jedné podmínky dal > a do druhé >=, takže se jedna šipka vykreslí na začátku okna a druhá až na konci.
  - Barva šipek – zatím byli šipky jen černé což znamenalo, že na některých mapách nebyly vidět. Rozhodl jsem se tedy udělat šipky dvoubarevné, ve spod černé s použitým Gaussovským filtrem a přes to bílé. Myslel jsem si, že to bude vypadat hezky. Bohužel byl opak pravdou a po přibližně dvou a půl hodinách programování jsem se dopracoval k výsledku, který se mi vůbec nelíbil a vzhledem k tomu, že kód se stal celkem nepřehledný, a navíc by se nafoukl skoro na dvojnásobek, a tak jsem se na to vykašlal a změnil barvu šipky na červenou.


**Nepěkné výsledky mého konvolučního filtrování**

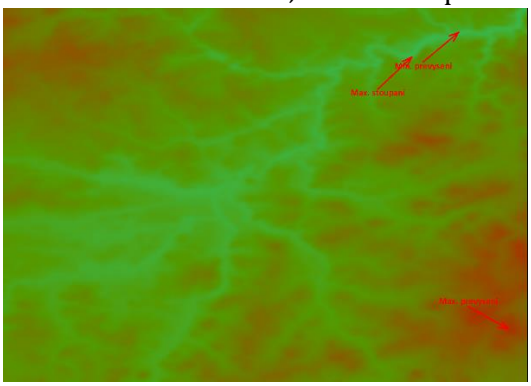
- Zobrazení okna – nějak mi nedošlo, že uživatel může zadat nějaký soubor, který bude mít mnohem větší šířku než výšku a obráceně. Opravil jsem to, tak že jsem přidal mnoho podmínek. Teď by snad nemělo dojít k zobrazení okna o šířce 25000 a výšce 600.
- Ošetření vstupu bez parametru – Původně když se nezadal žádný argument, vyskočil do konzole velký výpis chyb. Snažil jsem se toto ošetřit neboli přidal jsem hlášku a defaultní obrázek který se zobrazí, když není zadán žádný argument při spuštění aplikace.

# Deset hodin poprvé

Po ohodnocení pasivní vizualizace, jsem zjistil že mám špatně popisky, protože se vykreslují mimo obrázek. V následujících 3 hodinách trápení se mi to podařilo opravit. Dále jsem udělal barevnou škálu, tak aby vypadalo jako v mapě. Nakonec jsem začal dělat vrstevnice.

## Co jsem udělal?

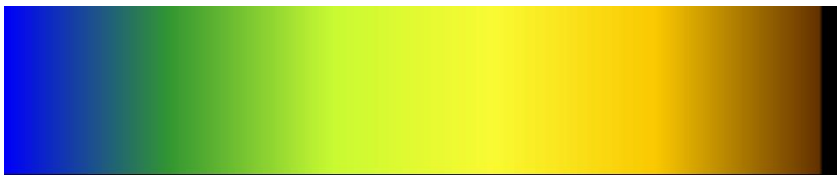
- Popisky – opravení popisků, tak aby se nevykreslovali mimo obrázek. Zajistil jsem, aby se popisky nevykreslovali mimo mapu, dokud má písmo menší šířku/výšku než obrázek. Implementace je pomocí metody textVObraze(), který vrací boolean hodnotu jestli se má souřadnice textu upravovat nebo ne, a podle toho se pak určí místo kde se popisek vykreslí.
- Barevná škála – Barevnou škála se tvoří v metodě makePalete(), cílem bylo udělat z černobílého obrázku barevný, a jelikož děláme mapu, řekl jsem si že udělám barevnou škálu odpovídající mapě. První plán byl takový že nejnižší hodnota bude modrá, která bude přecházet do zelené a dále do hnědé.



**První výsledek přidání barevné škály**

Výsledek sice nebyl úplně špatný, ale mě osobně tam něco scházelo, a tak jsem šel na internet stáhl jsem si mapu barevnou mapu České Republiky a pokusil se o napodobení barevné škály následovně:

- 0% Modrá
- 20% Tmavě zelená
- 40% Světle zelená
- 60% Žlutá
- 80% Oranžová
- 99% Hnědá
- 100% Černá

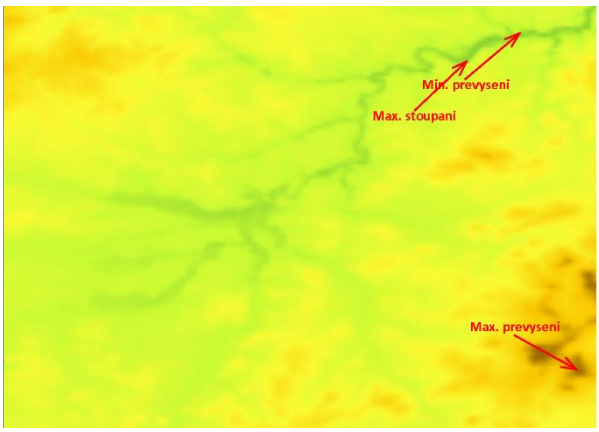


**Výsledná barevná škála**

Vytváření barevné škály tedy vypadalo tak, že jsem si otevřel program s nastavováním barvy RGB a zjistil jsem hodnoty r, g a b. Poté jsem si vypočítal, o jaká je tato změna mezi těmito hodnotami mezi jednotlivými barvami, aby nedocházelo ke skokům mezi barvami.

Př.: Modrá má hodnotu R = 0, G = 0, B = 250; Tmavě zelená má hodnotu R = 50, G =150, B = 50.
To znamená, že R se zvětší o 50, G o 150 a B se zmenší o 200. Dále tedy musím implementovat funkci postupného převodu. Pro nás to znamená, že když se nadmořská výška zvětší o 1 m (v případě, že maximální nadmořská výška je maximálně 255) R se zdvíhne o 1, G o 3 a B se zmenší o 4.
V případě, že je nadmořská výška větší jak 255, nejdříve se data upraví metodou, tak aby se poměr mezi hodnotami nezměnil. Ještě to bude chtít vylepši v hlavě se mi rýsuje tak, uvidím, jak se mi podaří implementovat.

Osobně si myslím, že se mi paleta barev povedla, barvy vypadají podobně jako v mapách, což byl také můj cíl. V mapě „hory“ jsem objevil menší easter egg ve formě textu „Toto je správně“.




**Druhý výsledek barevné škály**


**Mapa hory**

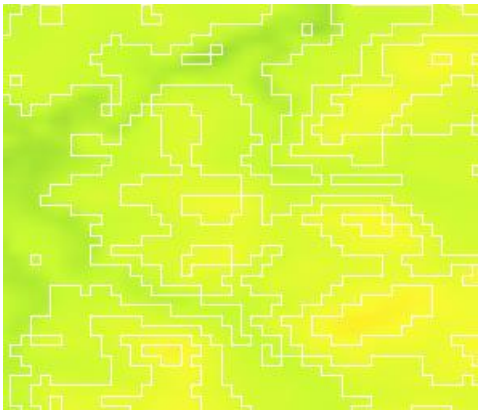
Vrstevnice – Přidělal jsem metodu vetsiNez(), která zjišťuje jestli je nadmořská výška pod danou výškou, vrací pole booleanů, takže teď vím, teď můžu zjistit, v jakých indexech se nadmořská výška přehoupne před nějakou hodnotu.

## Deset hodin podruhé

Minulý večer jsem začal dělat vrstevnice, celou noc jsem přemýšlel, jak to vykreslit čáry. Nedalo mi to spát, a tak jsem vstával už v půl sedmé ráno (spát jsem šel ve dvě :-D) a začal zkoušet různé metody vykreslení. Nakonec se mi povedlo vykreslit alespoň hranaté vrstevnice a já se mohl další ráno v klidu dospat. Nakonec jsem zvolil něco odpočinkovějšího, první rozšíření – Tisk, který jsme brali na posledním cvičení.

### Co jsem udělal?

- Vykreslení vrstevnic – Přidání metody drawVrstevnice(), věděl jsem že chci využít vytvořeného pravdivostního pole. Metoda tedy funguje tak, že se porovnávají hodnoty v řádku a když se změní pravdivostní hodnota, tak se nakreslí svislá bílá čára. To samé jenom v bledě modrém platí pro svislou osu, kde se nakreslí pro změnu čára vodorovná.



**První verze vrstevnic**

Jak je vidět vrstevnice se opravdu vykreslili, ale nevypadají moc dobře, pokusím se je nejspíš ještě vylepšit, uvidím, jak se mi to povede.

- Přidání timeru – s přidáním vrstevnic se objevil problém, že se vrstevnice překreslují jen když se hýbe s oknem, takže když se pohnulo s oknem rychle, tak se vrstevnice vykreslili špatně, což zrovna nefungovalo u velikých souborů jako např. hory. Problém jsem vyřešil tak, že jsem přidal timer.
- Tisk – Přidání menubaru do okna, funkce tisk je v menu export -> tisk.

# Deset hodin potřetí

V dalších deseti hodinovce jsem se zaměřil na dodělání základních a dalších požadavků interaktivní vizualizace. Nejdříve jsem vytvořil reakci na kliknutí do mapy, kde se po kliknutí zobrazí tečka a nadmořská výška v bodě. Dále jsem pokračoval (i přesto že nemám vyladěné vrstevnice) doplněním legendy a grafů. Po doplnění legendy mi přišlo, že je okno celkem přeplácané, a tak jsem usoudil, že grafy budu vykreslovat v novém okně. Legendu jsem dal do spodní částí okna v novém JPanelu. V tomto bloku jsem vzdal snahu o neopakování se kódu, takže v panelu, která vykresluje legendu je pár úseků skoro stejných jako v panelu, co kreslí mapu.

## Co jsem udělal?

- Kliknutí do mapy
  - Vykreslení bodu s nadmořskou výškou – Pomocí Mouse Listeneru se nejdříve zjišťuje, pomocí dvou podmínek, zda bylo kliknuto na mapu (v případě, že se klikne na okraj, nebo legendu nic se neděje). Dáje se muselo ošetřit, aby bod s číslem neměnil polohu, když se bude okno zmenšovat nebo zvětšovat. Souřadnice X a Y jsou tedy upraveny scalem a zmenšeny o velikost Okraje.
  - Změna barvy nejbližší vrstevnice – Abych změnil barvu správné vrstevnice, musím nejdříve zjistit, která vrstevnice má změnit barvu. O toto se stará metoda najdiNejblizsi(). Metoda prochází seřazené pole, ve kterém jsou hodnoty vrstevnic, a když je nějaká hodnota blíže nebo přesně 25 výškových metrů od nadmořské výšky bodu, bude tato vrstevnice nejblíže. Výjimky jsou krajní body, kde bod může být vzdálen více jak 25 výškových metrů. Je zde tedy speciální podmínka pro první prvek, tedy jestliže je menší, než první prvek bude nejblíže právě k prvnímu prvku. A zároveň když se projde celé pole a podmínka není splněna a bod je výše než nejvýše vykreslená vrstevnice, bude nadmořská výška nejblíže právě k největší nadmořské výšce. Nadmořská výška této vrstevnice se ukládá do proměnné zvýrazněné vrstevnice, která se pak vykresluje růžově, když tato proměnná se nerovná nule.

V této funkcionalitě se mi nějak nedařilo získávat správné hodnoty nadmořské výšky v bodě kliknutí při používání jednorozměrného pole, implementoval jsem tedy metodu, kde nahrávám všechna data do dvourozměrného pole, pomocí kterého spouštím metodu najdiNejblizsi().

- Legenda – Jelikož mám v DrawingPanelu vyřešenou logiky (jako např.: správné vykreslení šipek a textů), které nechci přepracovávat, rozhodl jsem se pro nový JPanel, který je ve spodní části obrazovky.
  - Hodnoty v legendě – Metoda createHodnoty() zajišťuje aby byly v legendě rozumné hodnoty v rozumném počtu (tj. např. aby se nezobrazilo 10 000 hodnot pod mapou se svým čtverečkem, to by bylo naprosto nečitelné, dále jsem šlo o to aby hodnoty nebyly nějaké nerozumné např.: 273, 313, 348, …).
    - Běžný obrázek - Jelikož bodu často zobrazovány obrázky, kde je škála nadmořských výšek 0 až 255, rozhodl jsem se udělat pro ně připravit předdefinované hodnoty (0, 50, 100, 150, 200, 250, 255).
    - Mapa, která má kontrast větší než 50 - Pro tyto případy jsem udělal výpočet počtu, tak aby bylo na obrazovce maximálně 20 hodnot legendy (ve většině případů méně než 10). Výpočet zjistí rozdíl mezi nejmenší a největší nadmořskou výškou, poté se zjistí do které „kategorie“ rozdíl patří a podle tohoto se určuje, po jakých skocích se budou generovat hodnoty v legendě. Začíná se číslem, které je dělitelné pod minimem a pokračuje se až hodnotou pod maximem.
    - Mapa, která má kontrast menší než 50 – U takové mapy by se do legendy nezobrazilo nic. Udělal jsem tedy pro takové hodnoty výjimku. Můžou nastat 2 stavy:
      - V mapě je jen 1 hodnota -> v legendě bude jenom 1 hodnota.
      - V mapě je jiné minimum a maximum -> v legendě budou 2 hodnoty min. a max.
  - Vykreslování legendy – Zde přišla na řadu matematika, konkrétně počítání, jak mají být čtverečky ukázky legendy veliké, aby i při zmenšeném okně šlo legendě porozumět. Počítám tedy velikost čtverce. Ta se určí podle počtu čtverců a aktuální velikosti okna. Mezi čtvercemi je mezera veliká jako velikost 2 čtverců. Čtverce tedy mění velikost společně s velikostí okna, pokud se text nevejde do okna (pod čtvercemi není dostatečné místo), začne se vykreslovat do čtverců.

- Změnění barev – Pro lepší vzhled jsem, změnil pozadí pod mapou, když má okno jiný poměr než obrázek, na bílou. A maximální hodnotu v mapové škále na šedou (jako barva skály).


**Dva příklady vzhledu legendy**

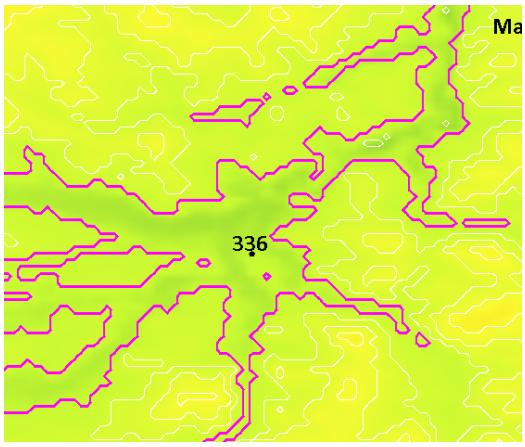
- Grafy – Pro grafy je použita knihovna jfreechart-1.5.3, oba typy grafů se spouští přes menu bar podobně jako při exportování souborů.
  - Histogram Převýšení – Použita před připravená třída ChartPanel pro histogram. Histogram ukazuje četnost nadmořských výšek po 5 nadmořských metrech. Je zařízeno, že na ose x budou jen ty hodnoty, které jsou v mapě (tj. bude osa x začíná minimální výškou v mapě a končí maximální výškou).
  - TukeyBox – Také použita předpřipravená třída. Graf pro zobrazení mediánu, minima, maxima a kvartilů. Z nějakého důvodu se mi nad grafem zobrazují 2 kružnice a nikde jsem nenašel, jak tyto kružnice zneviditelnit. Dále se mi nedaří zviditelnit minimum a maximum i přes to, že v kódu nastavuji jejich viditelnost na true. Na Stack Overflow jsem bohužel řešení nenašel, možná jsem hledal špatně, ještě se s tím pokusím něco udělat.

## Deset hodin počtvrté

V dalších deseti hodinách jsem se konečně doprokrastinoval k vylepšení vrstevnic, upravil jsem vzhled aplikace i kódu, tak aby aplikace vypadala co nejlépe kód se co nejméně opakoval. Začal jsem dělat další volitelná rozšíření, a to exporty dat do png, svg dále jsem upravil tisk mapy. Dále se zveřejnily další soubory na debuging, které mi neudělali radost, protože mi většina z nich nefungovala správně, kvůli velkému počtu vykreslovaných vrstevnic. Na konci desetihodinovky jsem se neúspěšně snažil o rozšíření Vyznačení maximálního převýšení.

### Co jsem udělal?

- Vylepšení vrstevnic – Snažil jsem se vylepšit vzhled podobným algoritmem, jaký byl implementován doposud, akorát u toho používat desetinná čísla. Můj nápad, ale vůbec nefungoval, takže jsem musel hledat jinde. Nakonec jsem se inspiroval algoritmem jednoho neindického youtubera, který zmínil, že existuje jen 16 možností vykreslení v pixelu. Algoritmus tedy funguje tak, že se nezaměřuji na pixely ale spíše na jejich hrany. Poté porovnávám 4 hrany a podle toho kreslím variantu vrstevnice. Starou metodu jsem v programu nechal bylo mi jí líto mazat.




**Finální verze vrstevnic**

- Export do PNG – Přidal jsem do menu export další menuItem jménem PNG, po stisknutí tlačítka se zobrazí tázací dialog s dvěma textFieldama do kterých se zadá šířka a výška obrázku, který se bude exportovat. Vstupy jsou ošetřeny, takže by při zadání písmen do textFiledů, neměla aplikace spadnout a měl by se zobrazit varovný dialog, který uživatele informuje o nesprávnosti vstupních dat.
- Export do SVG – Přidal jsem do menu export další menuItem jménem SVG, po stisknutí tlačítka se spustí metoda exportSVG(), která se stará společně s knihovnou org.jfree.svg-4.2 o export do svg. Jelikož jsou vrstevnice bílé a nebyly by na bílém pozadí vidět, změnil jsem jejich barvu na světle šedou.
- Snaha o optimalizaci aplikace – Jelikož se aplikace při spuštění souborů random2, horizont a čárový kód načítala strašně dlouho a vůbec se nedala ovládat (aplikace nereaguje). Pokusil jsem se aplikaci nějak urychlit.
    - Smazání timeru – timer byl zatím v aplikaci kvůli rychlé změně okna, kdy se špatně vykreslovali vrstevnice, rozhodl jsem se ho zabít. Po zabití timeru jsem musel nějak reagovat na změny okna. Přidal jsem tedy do metody paint() podmínku, která zaznamená změnu velikosti okna a okno bleskurychle překreslí.
    - Nastrkání všeho, co jde do konstruktoru – Všechny metody, které stačí provést jen při vytváření okna jsem narval brutální silou do konstruktoru.

Bohužel tyto dvě úpravy nepomohly a aplikace byla skoro stejně pomalá. Zapojil jsem tedy všech 100 jednotek pozornosti na řešení problému a uvědomil jsem si, že se musí vykreslit něco před 1000 vrstevnic, v metodě random2 (kde byl problém největší), dokonce většinou obkreslujeme skoro každý bod. Toto mě vedlo k jednomu provizornímu řešení.

    - Optimalizované vykreslování – Rozhodl jsem se k něčemu podobnému jako u kreslení legendy, a to že vrstevnice nebudu kreslit po 50 m, ale budu počítat po kolika metrech se mají vykreslovat, aby se aplikace co nejrychleji spustila a byly vidět alespoň nějaké vrstevnice. Později chci implementovat metodu, ve které se bude moc nastavit po kolika metrech se budou vrstevnice vykreslovat. Takže nyní se pro data do 1000 m vykreslují vrstevnice po 50 m, do 10 000 m po 500 m a pro hodnoty větší po 5000 m. Je tedy zajištěno, že se aplikace spustí v přijatelné době a uživatel si pak bude (snad) moc zvolit, jestli si chce počkat na vykreslení vrstevnic po 50 m (u souboru random2 je to něco mezi 15 a 20 minutami).
- Mód největšího převýšení – řekl jsem si, že to bude relativně lehké rozšíření. Vymyslel jsem algoritmus pro vypočítání převýšení, tj. počítal jsem průměrné stoupání hran pixelu, to jsem pak průměroval a nahrával do pole. V menuBaru jsem založil novou záložku mode, do kterého jsem vložil dva radioButtony jeden pro vizualizaci převýšení, druhý pro vizualizaci stoupání. Bohužel vše nefungovalo skvěle po přepnutí vizualizace sice byly vidět tmavší části mapy -> hrany (pro testování souboru UPG a lenna), šlo poznat obrazec, bohužel se mi nepřekreslili šipky, celá mapa byla modrá (hodnoty pod 50) a při kliknutí se ukazovaly hodnoty pro mapu převýšení. Po uvědomění, co všechno budu muset předělat jsem toto rozšíření horem pádem vzdal.

## Deset hodin popáté

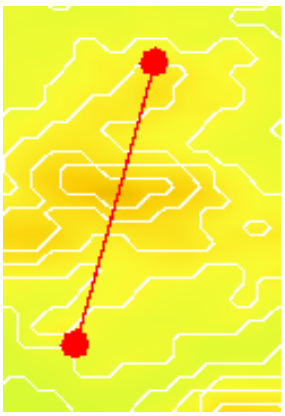
V posledních deseti hodinách jsem dokončil a optimalizoval aplikaci. Implementoval jsem metodu na nastavování vykreslení vrstevnic, rozšíření grafu převýšení a ASCII artu.

### Co jsem udělal?

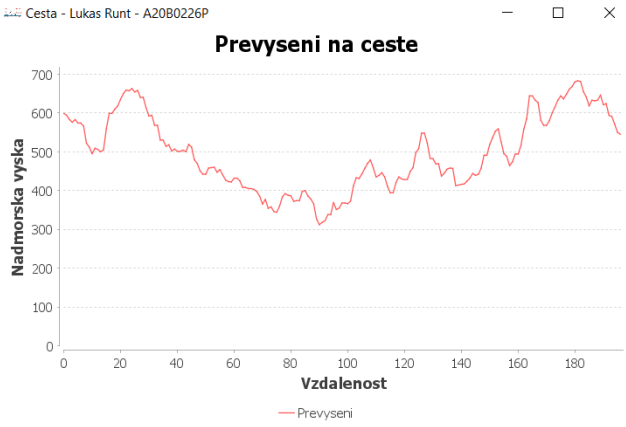
- Klávesa RESET – reakce na zmáčknutí klávesy r, po zmáčknutí zmizí kliknuté body a zvýrazněná vrstevnice přestane být zvýrazněná.
- Nastavení vrstevnic – Do menuBaru jsem přidal možnost nastavení, ve kterém se zatím nastavuje jen po kolika metrech se budou vykreslovat vrstevnice. Po klepnutí na menuItem vrstevnice na nás vyskočí dialogové okénko, které se nás zeptá „Po kolika metrech se budou zobrazovat vrstevnice:“, pod tímto textem se nachází comboBox ve kterém je na výběr po kolika se mají vrstevnice vykreslovat. Možnosti jsou ve formátu {10, 25, 50, 100, atd.}, dokonce je zde možnost nevykreslovat vrstevnice žádné. Jako defaultní možnost jsem vybral vykreslování po 50 m.

- Export do ASCII – Přidal jsem do menu export nový menuItem nazvaný ASCII, který složí k převodu obrázku na ascii obrázek. Jedná se tisknutí 6 druhů znaků do textového souboru. Znaky se určují podle nadmořské výšky. Pro každé dato se vytisknou 2 znaky, protože když se tisknul jen 1 znak vypadal výsledek celkem splácle.
- Graf převýšení – Zde jsem musel nejdříve vymyslet, jak se bude aplikace ovládat. Vymyslel jsem, že levé tlačítko myši bude zobrazovat nadmořskou výšku v bodě a zvýrazňovat nejbližší vrstevnici, pravé tlačítko se bude starat o graf převýšení. Toto vedlo k mírné změně MouseListeneru, který nyní rozlišuje levé a pravé tlačítko.
    - Funkcionalita – Po prvním kliknutí se vykreslí první červený bod, Po druhém kliknutí aplikace ví, že už byl první bod vykreslen a kreslí 2. bod, taktéž spojí oba body čárou, následně na uživatele vyskočí okénko s přímkovým grafem s převýšením na cestě vzdušnou čárou. Po zavření okna s grafem přímka i body zmizí, nevidím důvod, proč je nechat nakreslené v mapě.
    - Provedení – Pro získání hodnot grafu musím z obou bodů počítat směrový vektor, následně vytvořit jednotkový vektor (jako kdyby člověk ušel 1 m v mapě nějakým směrem). Při vytváření datasetu pro graf se data vytvářejí v následující while smyčce, kde se v každém cyklu ukládají data z dvourozměrného pole dat na určitém místě x, y. A toto se opakuje dokud se „nedojde“ k druhému bodu. Ukázka popisovaného kódu, kde u_x a u_y jsou složky směrového vektoru:

```
while(Math.abs(x - bod2X) > 1 || Math.abs(y - bod2Y) > 1) {
      s1.add(vzdalenost,pole2D[(int)x][(int)y]);
      x += u_x;
      y += u_y;
      vzdalenost++;
}
```



**Dva body po dvou kliknutí**



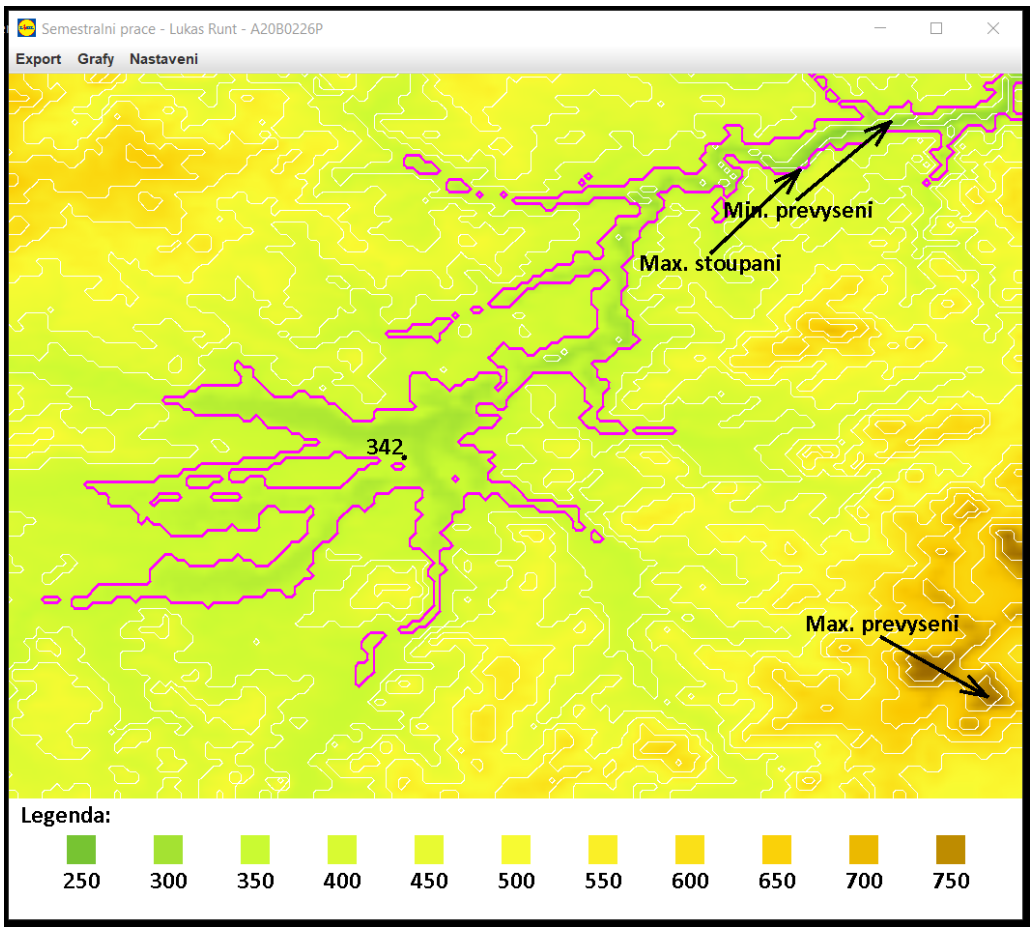

**Ukázka výsledného grafu**

Zbytek času jsem trávil ladění a dolaďováním toho, co je implementováno, aby bylo v aplikaci co nejméně bugů. Myslím, že je důležitější, aby v aplikaci nebyly bug a nedodělané věci, až aby v aplikaci bylo milion nedodělaných zabugovaných věcí. Bugů jsem našel celkem dost a je skoro jisté, že jsem na některé nepřišel.

## Shrnutí

Aplikace zobrazuje barevnou mapu s vrstevnicemi. Při prvotním spuštění se vrstevnice zobrazí po tolika metrech, aby se aplikace načetla rychle a nečekalo se až 20 minut na načtení. Po kolika metrech se budou vrstevnice načítat se dá nastavit v menuBaru -> Nastavení -> Vrstevnice, kde na uživatele vyskočí okno s volbou. Mapa reaguje na kliknutí: levé kliknutí -> zobrazí se nadmořská výška bodu a zvýrazní se nejbližší vrstevnice. Pravé kliknutí 1x – vykreslí se 1.bod, 2x – vykreslí se 2.bod a body se spojí přímkou, zároveň vyskočí graf s převýšením. Klávesa R ruší efekty způsobené myší. Export se najde v levém horním rohu. Aplikace dokáže exportovat do PNG, ASCII, SVG a Tisknout. Grafy nalezneme vedle menu pro export. Vyexportovaná data se ukládají do složky exports.

Volitelná rozšíření: Graf převýšení, Export do PNG, Export do SVG, ASCII art, Tisk




**Výsledný vzhled aplikace**

## Závěr

Aplikace pořád není ve stavu, jaký bych si úplně představoval, ale budu se s tím muset smířit. Myslím, že jsem nad semestrální prací už strávil dost času a vzhledem k tomu, že mám ještě dost práce s ostatními předměty, deadline se nezadržitelně blíží a zároveň už je aplikace v celkem solidním stavu, který mi snad na ten zápočet bude stačit. Trochu mě sice mrzí, že jsem si nestačil zkusit naprogramovat nějaká zajímavá rozšíření, jako například pozvolnou cestu (ADT grafu jsem se naučil pár dní před odevzdáním, btw. vůbec nevím, jestli by se to pomocí grafu dalo řešit nezkoušel jsem to, jen se domnívám), ale i tak jsem se svou prací celkem spokojený. Před začátkem semestru bych ani nevěřil, že něco takového zvládnu. Práce mě bavila, samozřejmě jen když se dařilo, v opačném případě jsem chtěl vyhodit něco jménem počítač z okna. Na závěr bych už snad jenom dodal „Snad mi aplikace nebude vrácena na předělání“.