Dokumentace k semestrální práci

Jan Pelikán

Obsah

1. Odevzdání

Úvod

Než jsem začal vypracovávat zadání samostatné práce, popral jsem se s Gitlabem o stažení „kostry“ programu. Potom co jsem vyhrál bitvu, prostudoval jsem si Java dokumentaci k tomu, co již bylo naprogramované, abych věděl, co a jak dělá celý program. Když jsem nevěděl, jak dále pokračovat v seminární práci, dost mi pomáhalo povídat si o tom s někým, kdo tomu vůbec nerozumí. To mi pomohlo abych si věci ujasnil a napadlo mě řešení problému.

Fungování programu

Spuštění programu

Program se spouští souborem Run.cmd. Při spuštění programu můžete navíc zadat jako parametr číslo od 0 do 3, podle toho, která chcete spustit scénář. 0 spustí první scénář. Vstup je ošetřen tak, aby při zadání jiného vstupu, než jsou čísla od 0-3 se spustil první scénář a vypsala se hláška do konzole. Vstup je ověřen pomocí if-else bloku.

Velikost okna se nastaví na velikost potřebnou pro přesné vykreslení scénáře.

Třídy a jejich metody

Program má pouze dvě třídy: „L01_SpusteniSimulatoru“ a „DrawingPanel“. První třída má v sobě Main, stará se pouze o spuštění DrawingPanel a nastavuje správnou velikost okna. Zajišťuje také aby se vykreslení obnovovalo každých 100ms. Třída DrawingPanel je zajímavější. Má spoustu metod.

První metoda ve třídě DrawingPanel je „*paint*“. V této metodě se neděje nic zajímavého, vykresluje celý program a stará se o správné škálování, které ovšem nepočítá. První zajímavá metoda je až *drawWaterLayer*.

V této metodě pomocí dvou cyklů „for“ vykresluji vodu. První cyklus je y-souřadnice a druhý cyklus počítá x-souřadnici. Celé to stojí na metodě *isDry*, která vrací jestli daný bod je mokrý nebo suchý. Pokud jsou buňky vedle sebe mokré, vykreslí se voda. Na konci metody se ptám, jestli zde mám i nějaké vodní zdroje, které případně vykreslím.

V předchozí metodě jsem zmiňoval vykreslení vodních zdrojů, o které se stará další metoda *drawWaterSources*. Tato metoda využívá metodu *drawWaterFlowLabel*, která má na starost vykreslit jméno řeky a směr toku řeky. Informace o pozicích mi dává *WaterSourceUpdater*, pozici šipky zjistím přes getIndex() % maxX a getIndex()/maxX. V prvním případě mám operátor „modulo“, aby mi vrátil x-ovou souřadnici v řádce, řádku zjistím poté pomocí dělení, které mi vrátí řádek, na kterém se bod nachází.

Metoda *drawWaterFlowLabel* používá a přesouvá pozici přijmutou z parametru (aby šipky a jména nevedla z řeky, ale byla vedle řeky). Otáčí jména, aby byla vodorovně se šipkou a vykresluje jméno řeky a šipku.

Metoda na vykreslení šipek je hodně inspirována šipkami ze cvičení 3.

Další zajímavou metodou je *computeModel2Windowtansformation*. Tato metoda počítá škálování, aby se vykreslování nedeformovalo se změnou velikosti okna. A potom také počítá posun souřadnic. Tato metoda je hodně inspirována cvičením 6.

Prostor ke zlepšení, případné nedostatky

Šipky a jména

Vykreslování šipek -> šipky stále mění směr a drobet i velikost, vzhledově to není úplně hezké, to samé je i se jmény. Ve výsledku bych toto chování rád odstranil, ovšem mě teď nenapadá žádný způsob jak na to.

Vykreslování

Všiml jsem si, že při spuštění vykreslování se více než dvojnásobně zvedne práce CPU. Dost mě překvapilo, když jsem si všiml že na mém PC, běžel procesor na 95% a z toho 65% bylo vykreslování programu. Mám Intel Core i5 osmé generace 3,2 GHZ. Ovšem s tímto chováním také nevím, co udělat.

Závěr

První část semestrální práce byla pro mě přínosná hlavně v tom ohledu, že jsem si vyzkoušel vše, co jsme se učili na cvičení a přednáškách. Dal jsem to všechno dohromady a myslím si, že všemu co dělám už celkem rozumím. Obtížnost první části mi osobně přišla docela vysoká, ale dost možná to bude tím, že nejsem tak zkušený programátor (zatím). Těším se na druhou část!

2. Odevzdání

Úvod

Začátek druhé části se mi zdál jednoduchý a říkal jsem si, že jestli budou druhé dva body stejné těžké, respektive lehké jako první dva, tak udělám i nějaké rozšíření. Opak byl pravdou, třetí a čtvrtý bod byl celkem složitý, hlavně co se týká získávání dat. Největší zvrat byl ovšem, když jsem zjistil, že musím opravit chyby z minulého odevzdání. Na to jsem dlouhou dobu nemohl přijít, protože kdybych věděl jak na to, tak to opravím už při prvním odevzdání.

Fungování programu 2.0

Přidal jsem do programu nové povinné „featury“. Prvním velkým vylepšením byla výšková mapa. Funguje přes metodu *drawTerrain*. Největším oříškem bylo asi vymyslet algoritmus, jak obarvit celou mapu nějak rozumně. Použil jsem podobný přístup jako při vykreslování vody a změnou barev jsem se inspiroval z jednoho cvičení. Použil jsem jiné posuny, aby moje mapa nevypadala jako ostatní, kteří si myslím použijí kombinaci zelené až hnědé. Zrychlení a zpomalení funguje tak jak by mělo přes proměnnou v, kterou zadávám jako parametr do Simulator.nextStep(). Tlačítka jsou dole uprostřed, pro uživatele na očích. Na grafech bylo asi nejsložitější přijít na způsob, jak získat správná data a jak je aktualizovat. Nejvíc jsem měl asi problém se získáním správných souřadnic, na které uživatel kliknul. Toto mi dalo hodně zabrat.

Opravení z minulého odevzdání

Přidal jsem počítání poměru a delt, tak aby se podle velikosti delt správně podle skutečnosti vykreslili scénáře.

Přidal jsem podmínky, tak aby když šipka ukazuje doleva, jméno řeky se otočilo a posunulo na správnou pozici a bylo u šipky.

Prostor ke zlepšení

Jedním místem ke zlepšení je určitě náročnost celého programu, tento problém jsem měl už v prvním odevzdání a teď se to s více funkcemi ještě zhoršilo. Zkoušel jsem předělat vykreslování, ovšem funkční vykreslování bylo ještě víc náročnější než to, co mám teď.

Závěr

Semestrální práce pro mě byla přínosná. Díky ní, jsem si zkusil v praxi, co a jak se v reálu dělá. Určitě to pro mě nebylo nic lehkého. Druhá část semestrální práce mi přišla drobet lehčí než část první, protože už jsme měli na čem stavět a ten program už nějak vypadal. Také už jsem plně chápal, jak funguje simulátor, který jsme obdrželi. Věřím, že celá semestrální práce bude v pořádku.