Seznamení s problematikou

Cílem první fáze semestrální práce je vytvořit program, který po spuštění vykreslí v okně o rozměrech 800x600px vodovodní síť s prvky sítě na souřadnicích, které mu byly zadány z třídy WaterNetwork. Jednotlivé prvky navíc musí být do okna převedeny tak, aby zabíraly maximální možný prostor, při tom však nesmí dojít ke zkreslení souřadnic v nějakém směru. Program by dále měl reagovat na změnu velikosti okna a tím pádem vše zachovat. Program musí zajist změnu vizualizace v čase během toho, jak třída WaterNetwork mění stav průtoku vody ve vodovodní síti. Poslední vlastností je umožnit uživateli zadat na vstupu parametr glyphSize, což je maximální možná velikost reprezentace prvků v síti v pixelech. Nicméně je potřeba stále zachovat souřadnice v nějakém směru.

Cílem druhé fáze semestrální práce je udělat vizualizaci mnohem více interaktivnější. Uživatel by tak měl být schopen měnit rychlost simulace, získat grafy zaplnění jednotlivých rezervoarů v závisloti na čase simulace, získat grafy rychlosti toku vody v jednotlivých potrubích v závisloti na času simulace, měnit nastavení ventilů a tím pádem i rychlost toku vody v jednotlivých potrubích a v poslední řadě mít možnost změnit parametr glypSize a tím jednotlivé elementy sítě zvětšovat a zmenšovat.

Konkrétní body zadání:

1.část:

- Výchozí okno o velikosti 800x600px.
- Vodovodní síť s jejími prvky zabírající maximální možný prostor okna s nezkreslenými souřadnicemi v nějakém směru a s maximální velikostí určenou parametrem glyphSize.
- Reagování programu na změny rozměrů velikosti okna (překreslí okno tak, aby bylo vše korektně zachováno).
- Obnovování vodovodní sítě v čase.

2.část:

- Program musí splňovat všechny požadavky z 1.části. (Nebyli fixní glyphy při změně velikosti okna)
- V okně s vizualizací bude ovládací prvek umožňující při běhu programu změnitparametr glyphSize určující velikost grafických reprezentací jednotlivých elementů sítě.
- V okně s vizualizací budou ovládací prvky umožňující „zrychlení“ a „zpomalení“běhu simulace.
- Po kliknutí na rezervoár se otevře okno s vizualizací zaplnění rezervoáru v závislosti na čase od okamžiku spuštění programu. Vizualizace musí plně využívat velikost okna, musí obsahovat popsané osy a musí být především správně. Velikost okna s grafem musí být možné měnit a graf se musí velikosti okna správně přizpůsobit
- Po kliknutí na střed potrubí, kde je zobrazena šipka udávající směr proudění, se otevře okno s vizualizací rychlosti proudění v závislosti na čase od okamžiku spuštění programu. Vizualizace musí plně využívat velikost okna, musí obsahovat popsané osy a musí být především správně. Velikost okna s grafem musí být možné měnit a graf se musí velikosti okna správně přizpůsobit.
- Ventily umožní ovládání myší. Zvolte vhodnou grafickoureprezentaci ovládacího prvku (např. táhlo, popř. vlastní „radiobutton“ možných úrovní otevření)

Volitelná rozšíření:

- Zobrazit rezervoáry tak, aby jejich velikost odpovídala jejich kapacitě. Největší rezervoár bude zobrazen grafickou reprezentací o velikosti odpovídající uživatelskému parametru glyphSize, ostatní budou proporcionálně menší. Plocha grafické reprezentace musí odpovídat objemu příslušného rezervoáru

Popis implementovaného řešení

Celý problém působí při vnějším pohledu velice komplexně a složitě. Proto by bylo dobré si préblém rozdělit na několik menších podúkolů, jejichž postupným splněním a pospojováním ve výsledku vyřešíme problém hlavní. Výsledkem by tak měl být objektově orientovaný program, který pomocí několika tříd, instancí, funkcionalit a metod, bude řešením daného problému.

1.část

1. fáze → načtení scénáře

Třída WaterNetwork disponuje jednotlivými scénáři vodovodních síti a stará se o jejich aktualizaci i u jednotlivých prvků v síti. BasicDrawing je třída, která obsahuje metodu main. V této metodě vytváří okno, nabídne uživateli zadat parametr glyphSize, vytvoří instanci třídy WaterNetwork s konkrétním scénářem a společně s glyphSize předá jako parametry komponentě (DrawingPanel), na kterou se bude vykreslovat. Následuje timer, který se stará o update sítě a překreslení okna 1 za 100 ms (10 x za sekundu).

2.fáze → Vykreslení sítě

Vykresluje se na komponentu (JPanel), která je definována v třídě DrawingPanel. Tato třída obsahuje hned několik důležitých metod, které zajistí vykreslení sítě. Z nichž nejhlavnější jsou metody naciScenar(), initScale(), drawTrubky(), drawRezervoary()

naciScenar() → Načte síťově uzly, potrubí a ty uloží do třídních atributů, abychom si je zachovali pro případnou další práci.

InitScale() → Metoda nastaví třídní atribut scale používající k přenastovavání souřadnic. V principu si metoda zjistí velikost světa vodovodní sítě a tu napasuje do výsledného okna na základě jeho rozměrů. Podělí šířku, výšku okna s šířkou, výškou světa a vybere minimum z nich (abychom nedeformovali souřadnice v obou směrech) a to určí jako scale. Následně si uchová maximální a minimální body světa pro případ, že budou elementy sítě na extremních souřadnicích.

drawTrubky() → Vykreslí potrubí s šipkami popisující směr průtoku vody i s popiskem rychlosti průtoku a kohouty, jejichž otevřenost určuje rychlost průtoku. U šipek je zajištěno, aby se vykreslily kolem středu o velikosti 1/3 délky potrubí. Šířka potrubí je odvozena z glyphSize * scale.

DrawRezervoary() → Vykreslí nádrže o velikost glyphSize * scale s aktuálním stavem vody v nádrži, která je získána z atributů třídy Reservoir (content/capacity).

Tyto metody obsahují další pomocné metody, které se starají o konkrétní výpočty a nastavení.

3.fáze → metoda paint v DrawingPanel

Ve třídě Drawing panel se výše popsané metody nachází v metodě paint, která grafickým kontextem umožňuje kreslit do okna. V metodě paint se poté metodám předají potřebné parametry a za pomocí metod z ostatních tříd se vykreslí do okna vodovodní síť s aktuálním časem.

2.část

1.fáze → Oprava, aby byly fixní glyphy při změně velikosti okna dle parametru glyphSize.

V metodách drawReservoar() drawPipe() při škálování souřadnic jsem přidal funkcionalitu (pokud je scale > 1, tzn. došlo by k přesažení velikosti elementů dle glyphSize), která jednotlivé rezervoáry zmenší do velikosti určenou glyphSize tak, aby zachovala poměry.

2.fáze → vytvoření hitboxů pro vytvoření a vyobrazení grafů a ovládání stavů kohoutů.

Ve DrawingPanel si v paměti (třídních atributech „listech“) udržuji objekty představující rezervoáry, středy potrubí (hitbox o velikosti prurezu potrubí), kohouty a následně jsem k nim vytvořil metody kliklUzivatelNaStredPotrubi(), kliklUzivatelNaRezervoar(), jeUzivatelNaKohoutu(), které testují, jestli uživatel trefil hitboxy. Během celé simulace si udržuji v paměti k jednotlivým elementům sítě požadované hodnoty a v jakém času byly naměřeny. V hlavní třídě WNVis_SP2024 pak vytvářím grafy, tak že si vytáhnu z paměti listy k požadovaným elementům sítě, které uživatel vybral *(u rychlosti toku vody v potrubí záporné hodnoty převádím na absolutní hodnoty, protože znaménko pouze nese informaci o směru toku vody, která je pro reprezentaci v grafu irelevantní).* K reprezentaci obu grafů jsem se rozhodl použít spojnicový graf, protože mi přijde, že lze vhodně posuzovat měnící se trendy mezi daty tak, jak plyne čas simulace. Jednotlivé grafy a radio buttony pro nastavení kohoutů vyobrazuji v nových oknech (jedno okno pro grafy rezervorů, jedno okno pro rychlosti toků vody v potrubí, jedno okno pro nastavení kohoutů).

3.fáze → Vytvoření tlačítek pro zmenšování/zvětšovaní glyphSize a rychlosti/pomalosti simulace.

Pro umístění tlačítek jsem vytvořil toolBar, který jsem umístil na South pozici okna. Tlačítka pak vytvořím a přidám do panelu buttonPanel a ten vložím do toolbaru. Následně přídám actionListenry, kde nastavuji akce *(pomocí setrů na atribut glyphSize a metod runFaster() a runNormal() z třídy WaterNetwork)* k příslušným tlačítkům. Implementace se nachází ve třídě WNVis_SP2024 v metodě vytvorARozmistiTlacitka()

4.fáze → výpočet proporciálních rezervoárů

O výpočet se stará metoda vypoctiProporcionalniRozmeryRezervoaru(), která si nejdříve najde rezervoár s nejvetší kapacitou a kapacitu ostatních rezervoárů podělí právě touto kapacitou, tím získá poměr, kterým můžeme přepočítat objem nejvetšího na požadovaný (meritko * maxObsah „glyphSize * glyphSize“). Nálsedně vypočíta z obsahu stranu a.

Popis vytvoření, instalace a spuštění aplikace

Uživatel si stáhne zazipovaný soubor, který posléze rozbalí ve svém souborovém systému. Spouštěcí soubor Run.sh se nachází v ROOT složce. Uživatel si pak v místě, kde se nachází spouštěcí soubor Run.sh, otevře terminál a zadá příkaz: bash Run.sh → tímto se spustí celá aplikace.

Soubor Run.sh obsahuje následující příkaz:

- #!/bin/bash java -cp ./bin:./Jama-1.0.3.jar WNVis_SP2024 $@→ pomocí příkazu se přesuneme do adresáře bin, kde se nachází Class soubory (zakódované v UTF-8), a následně se vybere soubor WNVis_SP2024.class, kde se nachází Main metoda a tím se celá aplikace spustí.

Popis ovládání aplikace

V první části semestrální práce je ovládání velice prosté. Uživatel spustí aplikaci zadá parametr glyphSize, poté už se aplikace sama stará o simulaci. Uživatel může simulaci vodovodní sítě kdykoliv ukončit stisknutím křížku v právém horním rohu okna.

V druhé části semestrální práce přibylo pro uživatele více možností, jak s vizualizací interagovat. V dolní částí okna, kde se vizualizace vykresluje, nalezne tyto ovladácí prvky:

- zmensit Glyph size o 1 px → tlačítko zmenší glyphSize (velikost elementů) o 1 pixel
- zmensit Glyph size o 10 px → tlačítko zmenší glyphSize (velikost elementů) o 10 pixelů
- zrychlit → tlačítko zrychlí běh simulace (využije metodu runFast())
- normální běh → tlačítko vrátí rychlost běhu simulace do defaultního stavu
- zvetsit Glyph size o 1 px → tlačítko zvetší glyphSize (velikost elementů) o 1 pixel
- zvetsit Glyph size o 10 px → tlačítko zvetší glyphSize (velikost elementů) o 10 pixelů

Interakce s prvky sítě:

- kliknutí na rezervoár → po tom, co uživatel klikne na rezervoár, se uživateli v novém okně zobrazí graf reprezetnující obsah vody (v kubických jednotkách) v rezervoáru v závislosti na čase simulace. V případě, že uživatel klikne na jiný rezervoár, tak se graf v okně překreslí na graf související s aktuálním rezervoárem. Okno uživatel zavře kliknutím na křížek v pravém horním rohu.
- kliknutí na střed potrubí → po tom, co uživatel klikne na střed potrubí, se uživateli v novém okně zobrazí graf reprezetnující rychlost toku vody v potrubí v závislosti na čase simulace. V případě, že uživatel klikne na jiný střed potrubí, tak se graf v okně překreslí na graf související s aktuálním středem potrubí. Okno uživatel zavře kliknutím na křížek v pravém horním rohu. Hitbox pro uživatele odpovídá čtverci o velikosti průřezu potrubí, který je umístěn ve středu potrubí.
- kliknutí na kohout → po tom, co uživatel klikne na kohout, se uživateli v novém okně zobrazí radio buttony umožnující vybrat mezi různými stavy nastavení kohoutu, čím vetší je otevřenost, tím více a rychleji voda potrubím poteče, tzn. 100% znamená, že kohout je úplně otevřen a naopak 0% znamená, že kohout je zavřen. Uživatel zavře okno kliknutím na křížek v pravém horním rohu nebo kliknutím kamkoliv jinam v okně vizualizace.

Popis dosud neopravených nedostatků a popis možného rozšíření do budoucna

1.část

Můžeme si všimnout, že ve zdrojovém kódu se nachází části kódu, které jsou více méně stejné. Hlavně při přeškálování jednotlivých bodů. Porušil jsem tím základní pravidlo DRY (don’t repeat your self), což zvyšuje riziko přepsání a tím pádem špatné funkčnosti programu. Do budoucna by tak bylo dobré přidat metodu, která by bod vždy přeškálovala.
Dalším velikým a pravděpodobněji závažnějším problémem je, že ve většině metod se vyskytuje velmi časté a poměrně členité větvení programu. To má za výsledek velikou nepřehlednost. Obecně se díky tomu na určitých místech špatně předvídá, jak se program bude chovat. To může mít za následek nevyzpytatelné chování. I po důkladném otestování programu pak může i tak dojít k závažným chybám, jako jsou například výjimky zapříčiňující pády programu, či nezamýšlené chování programu. Toto se hlavně vyskytuje v metodě nastavKrajniBodyPotrubi() a dopoctiZacatekAKonecPotrubi(). Můžeme si všimnout, že tyto metody mají i velmi mnoho parametrů, což opět velice znepřehledňuje kód. Tento závažný problém bych chtěl do budoucna opravit. Zatím jsem však ale nepřišel na jiný způsob, jak dopočítat začáteční a koncový bod grafické reprezentace potrubí. Aplikace se bude také hůře rozšiřovat vzhledem k požadavkům v druhé části. To mě přivádí k tomu, že jsem měl věnovat více úsílí a času rozplánování aplikace a program udělat mnohem více objektově orientovaný. Například vytvořit jednotlivé třídy pro šipky a kohouty. Odkazy na vytvořené instance uchovávat v třídních atributech (poli), takto by pak bylo možné s prvky v budoucnu pracovat a aplikaci rozšiřovat. Můžeme si všimnout, že v aplikaci nevyužívám vůbec žádné abstraktní datové typy, což opět naznačuje, že moje řešení je velice zevrubné a prvoplánové. Posledním velkým nedostatkem je nesplnění bodu glypSize ze zadání, kdy se elementy sítě vykreslují nezávisle na jeho hodnotě. Bohužel i zde jsem zatím nepřišel na způsob, jak tento problém vyřešit. Jediné, co mě napadlo, bylo připravit škálování pouze pro body, kolem kterých se rezervoáry vykreslují, a ne pro cele rezervoáry, jak mám implementováno. Zde jsem ale narazil, jelikož při dopčítávání velikosti elementů (nádrží) pak může dojít k zkreslení poměru vzdáleností mezi jednotlivými nádržemi než poskytuje třída WaterNetwork. Hlavně pokud jsou rezervoáry v krajních bodech okna například (0,0).

2.část
Kód, kterým jsem rozšiřoval program, stavěl na nedostatcích kódu z první části. Snažil jsem se tak aspoň co nejvíce minimalizovat hlavní nedostatky tím, že jsem se snažil kód alespoň co nejvíce rozdělit do metod a snažit se program příliš nevětvit. K mnohem většímu zpřehlednění kódu by pravděpodobně došlo po tom, co bych program více rozdělil do tříd. Například veškeré výpočty souřadnic rezervoáru, škálování, definování hitboxů, definování tvarů bych měl vytvářet a nastavovat ve třídě Reservoir a ve třídě Drawing si pak pouze uložit instance reservoarů do pole a pomocí getrů získávat tyto data a pouze vykreslovat. To samé bych měl udělat pro kohouty, šipky, a potrubí. Myslím si že toto by výrazně zvětšilo čitelnost kódu. Bohužel pro metody nastavKrajniBodyPotrubi() a dopoctiZacateAKonecPotrubi() se mi nepodařilo přijít s novou myšlenkou, která by nevyžadovala tak složité větvení.

Závěr

Na závěr bych dodal, že program má místa, která by šla jednoznačně vyladit. Jsem si také vědom, že tento program nesplňuje „základní vlastnosti dobrého programu“. Kód mi přijde na několika místech zbytečně zdlouhavý a složitý, což zhoršuje jeho přehlednost. V kombinaci s vnořenými a větvícími se podmínkami se násobně zvyšuje jeho Cyclomatic complexity, což je jen dalším důkazem nedokonalosti tohoto programu. Osobně s tím jak jsem aplikaci postavil nejsem vůbec spokojen a v další části se určitě budu snažit chyby neopakovat. To mě přivádí k myšlence, že pro příště bych se měl důkladně zaměřit na to, jestli by na program nešly aplikovat nějaké programátorské struktury (přepravka, rozhraní, abstraktní datová třída atd.), kterými bych dokázal lépe chod programu kontrolovat, zpřehlednit a zefektivnit. Pravděpodobně by mě to přivedlo i k použití abstraktních datových typů (fronta, zásobník, grafy, stromy atd.). Neměl bych se také spokojit s prvním řešením, které mě napadne a spíše problém rozdělit na více podproblémů. Snažit se přijít na různá efektivnější řešení, ba dokonce vyhledávat struktury a snažit se je přímo na problém napasovat a zavrhnout ty, co vedou na příliš hrubé větvení.