UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Jádro animace synchronizované s metodou snímání aktivit

Jiří Adam

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jiří Adam**
Osobní číslo:	**I08005**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Jádro animace synchronizované s metodou snímání aktivit**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

1. Primárním cílem bakalářské práce je realizace vzorové implementace jádra animace spolupracující s jádrem počítačové simulace založené na metode snímání aktivit.
2. V rámci případové studie je implementované jádro využito pro animace měnících se poloh mobilních prostředků v rámci vybraného typu simulovaného systému.
3. Pro implementaci je zvolen programovací jazyk Java nebo C (případně C++).

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. KRIVÝ, I., KINDLER, E. Simulace a modelování, elektronická skripta Ostravské univerzity, 2001. 146 s.
2. BANKS, J. Handbook of simulation. New York: John Wiley & Sons, 1998. 850s. ISBN 0-471-13403-1
3. HUŠEK,R., LAUBER,J. Simulační modely, SNTL/ALFA, Praha 1987, 349 s.
4. KAVICKA,A.,KLIMA,V.,ADAMKO,N.: Agentovo orientovaná simulácia dopravných uzlov, ved. monografie, EDIS, Žilina 2006, 210 s. ISBN 80-8070-477-5

Vedoucí bakalářské práce: **prof. Ing. Antonín Kavička, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 5. 2013 Jiří Adam

Poděkování

Především bych chtěl poděkovat vedoucímu mé bakalářské práce prof. Ing. Antonínu Kavičkovi, Ph.D. za jeho neocenitelné rady. Také bych chtěl poděkovat svým rodičům za jejich podporu a trpělivost během mého studia.

Anotace

V této bakalářské práci byla realizována vzorová implementace jádra animace spolupracující s jádrem počítačové simulace založené na metodě snímání aktivit. V rámci případové studie je implementované jádro využito pro animace měnících se poloh mobilních prostředků v rámci vybraného typu simulovaného systému. Pro implementaci je zvolen programovací jazyk Java.

Klíčová slova

Simulace, animace, aktivita, metoda snímání aktivit

Title

Animation core synchronized with method of activity scanning

Annotation

The bachelor's thesis described paradigmatic implementation of animation core cooperating with computer simulation core based on method of activity scanning. In our case study the implemented core is used for animating changes of vehicle’s position in selected simulated system. Program language Java is chosen for implementation.

Keywords

Simulation, animation, activity, method of activity scanning

Obsah

Seznam zkratek

ADT	Abstraktní datový typ
ADS	Abstraktní datová struktura

Seznam obrázků

Seznam tabulek

Úvod

Tato bakalářská práce bude realizovat vzorovou implementaci jádra počítačové animace spolupracující s jádrem simulace založené na metodě snímání aktivit. V rámci případové studie bude implementované jádro využito pro animace měnících se poloh mobilních prostředků, které se pohybují po krátkém úseku silnice s čerpací stanicí v obou směrech.

Pro dosažení tohoto cíle bude třeba vysvětlit některé pojmy z oboru modelování a simulace. Budou představeny některé synchronizační metody simulačního výpočtu včetně metody snímání aktivit, která bude popsána více do hloubky. Následně bude vysvětlena filozofie implementace metody snímání aktivit, jádra spojité simulace a jádra animace, které bude podporovat on-line animaci.

Popíše se vlastní programová implementace daného simulačního modelu. Zavedou se abstrakce na vymezeném objektu zkoumání, ověří se funkčnost metody snímání aktivit i samotného simulačního modelu, a pak se bude se simulačním modelem experimentovat za různých délek period snímání a simulace. Výsledná data z experimentů budou analyzována a vypočtou se příslušné statistiky.

1 Definice základních pojmů

Pro porozumění této práce je třeba předem definovat význam některých základních termínů oboru modelování a simulace. Pojmy a definice jsou převzaty z publikace (KAVIČKA, 2006, 210s).

1.1 Abstrakce a systém

Pod pojmem **abstrakce** se rozumí převádění nějakého reálného objektu do objektu takového, který obsahuje zjednodušenou verzi objektu reálného. Tento nově vzniklý objekt se v modelování a simulaci nazývá **systém.** Systém tedy musí obsahovat zvládnutelné aspekty objektu reálného a nepotřebné aspekty jsou zanedbány. Tento problém lze také vyjádřit tak, že nad zkoumaným objektem jsou vymezovány systémy. Jako příklad systému může být úsek silnice s čerpací stanicí nebo obchodní dům s několika kasami.

Systém lze rozdělit podle toho, zda uvažuje existenci času, a to na statický a dynamický. **Statický systém** je takový systém, jenž abstrahuje od významu času, to znamená, že zanedbává pojem času. Systém, jehož čas se nezanedbává a zároveň je chápán ve smyslu „Newtonovské“ fyziky se označuje jako **dynamický systém**. Simulace se jinými než dynamickými systémy nezabývají. Množina okamžiků, ve kterých dynamický systém existuje, se nazývá časová existence dynamického systému, zkráceně **existence systému.** Množina takových okamžiků nemusí být intervalem reálných hodnot. Například pro dopravního odborníka, který měří provoz v dopravní špičce křižovatky, jsou důležitá data z těchto období a přechodové fáze mezi nimi jsou pro něj zbytečné. Systém tedy existuje v konečné množině navzájem izolovaných časových úseků.

Nyní je třeba vymezit termín **okolí systému**. Okolí zkoumaného objektu obsahuje objekty, které není třeba zkoumat, ale přesto je třeba jejich existenci brát v úvahu, protože nějakým způsobem ovlivňují zkoumaný objekt. Takovéto okolí nazýváme **okolí systému.** Okolí systému lze také nazvat abstrakcí okolí zkoumaného objektu. Pod tímto okolím si lze představit například proudy zákazníků vstupující do obchodního domu nebo automobily projíždějící úsekem silnice.

Systém se skládá z **entit (prvků).** Entity mohou reprezentovat jak fyzické tak i logické složky zkoumaného objektu. Entity lze dělit na **permanentní** a **temporární**. Permanentní entity zůstávají v systému během celé jeho existence, kdežto temporární jsou v systému jenom určitou dobu. Temporární entity se dají dále rozlišit na **exogenní** a **endogenní.** Exogenní entity vznikají vně systému, tedy v jeho okolí. Endogenní entity se vytváří uvnitř systému. Temporární prvky mohou zaniknout dvěma způsoby, a to v systému nebo přesunutím do okolí systému. Prvky lze také dělit na **mobilní a stabilní**. Mobilní prvky na rozdíl od stabilních mohou měnit polohu, a to samy nebo jinou entitou. Příkladem temporárních exogenních prvků mohou být zákazníci přicházející do obchodu. Smlouva, vyhotovena pracovníkem v obchodě, podepsána zákazníkem a dále archivována, je příkladem temporárního endogenního prvku. Permanentní prvky mohou být stabilní prvky infrastruktury přepážky, koridory pro fronty, apod. Permanentní mobilní prvek je například

obslužný personál. V systémech hromadné obsluhy (obslužných systémech) se prvky dělí na obsluhující (zdroje) a obsluhované (zákazníci). Tyto funkce, však nemusí být stálé a mohou se v průběhu času změnit.

Každá entita obsahuje **atribut** neboli vlastnost. Atributy obsahují hodnoty, jež se mohou v průběhu času měnit. Atribut lze klasifikovat na **standardní** a **referenční**. Pod standardním atributem si lze představit například reálné číslo, text či booleovskou hodnotu. Referenční atributy přiřazují prvku ukazatel na jiný prvek, tedy mezi prvky definují vazby.

Nakonec je ještě potřeba vědět, že stav dynamického systému, zkráceně **stav systému** v čase t, je dán prvky, které existují v systému v čase t a jejich hodnotami atributů.

1.2 Modelování a simulace

V oblasti modelování a simulace je termín **model** použit pro analogii mezi dvěma systémy, a to **modelovaným** (originálním) a **modelujícím**. Vztah mezi těmito modely je dán tím, že každému prvku P modelovaného systému je přiřazen prvek Q modelujícího systému, každému atributu g prvku P je přiřazen atribut h prvku Q a pro hodnoty atributů g a h je dána nějaká relace. Jsou-li modelovaný i modelující systémy statické, lze daný model označit za statický model. V simulaci se však uplatní jen tzv. **simulační model**, jedná se o modely, které splňují následující požadavky (KŘIVÝ, 2001):

1. Jejich modelující i modelované systémy jsou dynamické systémy.
2. Existuje zobrazení τ existence modelovaného systému do existence modelujícího systému. Je-li t_1 časový okamžik, v němž existuje modelovaný systém M_1, je mu přiřazen okamžik $\tau(t_1) = t_2$, v němž existuje modelovaný systém M_2, a tak je zobrazením τ přiřazen i stavu $S(t_1) = \sigma_1$ systému M_1 stav $S(t_2) = \sigma_2$ systému M_2.
3. Mezi stavy σ_1 a σ_2 jsou splněny požadavky na vztahy mezi prvky a jejich atributy, jak již bylo definováno pro modely obecně.
4. Zobrazení τ je neklesající, to znamená, že pokud v originálním systému nastane stav S_1 před stavem S_2, pak v modelujícím systému musí též nastat stav S_1 před stavem S_2 nebo alespoň ve stejnou dobu, pokud se nejedná o kvalitní systém, nikdy však nesmí stav S_2 nastat před stavem S_1.

Běžně v praxi se pod pojmem model rozumí modelující systém, a to i přesto, že tento pojem není přesný a výstižný. U simulačních modelů se před pojmy modelovaný a modelující systém dává přednost pojmům **simulovaný systém** (originální) a **simulující systém**. Analogicky jako u modelujícího systému se místo termínu simulující systém používá méně přesný termín **simulační model** nebo jenom **simulátor**.

Nyní je možné definovat pojem **modelování**. Modelování znamená nahrazování zkoumaného systému jeho modelujícím systémem (jeho modelem), jejímž úkolem je získat pomocí experimentů s modelem informace o původním originálním systému (KŘIVÝ,

2001). S pojmem modelování souvisí i pojem **simulace**. Jedná se o výzkumnou techniku, jejíž podstatou je náhrada zkoumaného modelovaného dynamického systému jeho modelujícím systémem, s nímž se experimentuje za účelem získat informace o originálním systému.

1.3 Aktivity a procesy

Aktivity a procesy představují dynamické vlastnosti systému, tj. běh času a změny stavu systému v čase. Pojem **aktivita** představuje akci reálného světa, která je začleněna do simulujícího systému. Příkladem aktivity může být pohyb automobilu po dopravním úseku nebo obslužná činnost zákazníka u přepážky. Aktivity mají určité časové trvání a mohou měnit stav systému. Běh simulačního programu je realizován vykonáváním těchto aktivit, které se však musí vykonat ve stejném pořadí jako v simulovaném (originálním) systému. Časová existence aktivity je určena množinou reálných čísel (časových okamžiků). V těchto časových okamžicích může aktivita měnit stav systému. Aktivity lze z tohoto hlediska rozdělit na spojité a diskrétní.

Spojitá aktivita, která je definována dvěma reálnými hodnotami, může měnit stav systému během celé její existence. Příkladem spojité aktivity je přesun dopravního prostředku z bodu A do bodu B v případě, že doba přesunu tohoto prostředku není předem známá. Hlavní charakteristikou spojité aktivity tedy je, že předem není možné určit její dobu trvání. **Diskrétní aktivita** může změnit stav systému pouze v okamžiku jejího ukončení. Taková aktivita tedy existuje jenom v okamžiku jejího ukončení a je definována jednoprvkovou množinou reálných čísel. Ukončení diskrétní aktivity a následnou změnu stavu systému označujeme jako **událost** (**U**). Příkladem takové aktivity může být přesun dopravního prostředku z bodu A do bodu B v případě, že je známo za jak dlouho se prostředek dostane do bodu B. Hlavní charakteristikou diskrétní aktivity tedy je, že předem známe její pevnou dobu ukončení. V případě, že aktivity na sebe přirozeně navazují, tj. tvoří jistou posloupnost, pak se tato posloupnost označuje za **proces** (např. průjezd vozidla různými částmi města).

Dle typů aktivit, které simulovaný systém obsahuje, se simulace dají dělit na následující:

- **spojitá simulace**, obsahuje jenom spojité aktivity,
- **diskrétní simulace**, která obsahuje jenom diskrétní aktivity,
- **kombinovaná (diskrétně-spojitá) simulace**, která obsahuje spojité i diskrétní aktivity.

1.4 Simulační experiment

K vysvětlení simulačního experimentu, je třeba definovat **simulační program**. Jedná se o program, který řídí výpočet simulace, přičemž je ho možné spouštět pod různými konfiguracemi simulátoru za účelem provádění simulačních experimentů. Pro spuštění takového programu se nejdříve musí nakonfigurovat scéna, na které se bude program

odehrávat a nadefinovat scénář. **Scéna** označuje množinu všech permanentních prvků v systému včetně jejich atributů. **Scénář** popisuje dynamické chování simulátoru. Scénář je definován třemi parametry:

- scénou,
- pravidly vstupu, výstupu, generování a zánik temporárních prvků a
- rozhodovacími a řídícími algoritmy popisující procesy a jednotlivé aktivity.

Scénář tedy definuje množinu hodnot vstupních parametrů, jejímž výsledkem je množina výstupních parametrů, které popisují chování systému. Nyní lze definovat samotný **simulační experiment (pokus)**, jedná se o běh simulačního programu podle jednoho definovaného scénáře. V rámci simulačního experimentu zkoumáme chování vymezeného systému při použití daného scénáře.

Samotné vstupní parametry nemusí být deterministické, ale mohou být stochastické (o deterministickém a stochastickém přístupu více v kapitole 1.6). V případě, že použijeme stochastický přístup, jsou i příslušné výstupní hodnoty realizacemi náhodných hodnot (závislé na vstupních náhodných hodnotách). Z tohoto důvodu je nutné realizovat celou sérii pokusů se stejným scénářem, ale rozdílnými instancemi vstupních náhodných hodnot. Takové pokusy se nazývají **replikace.**

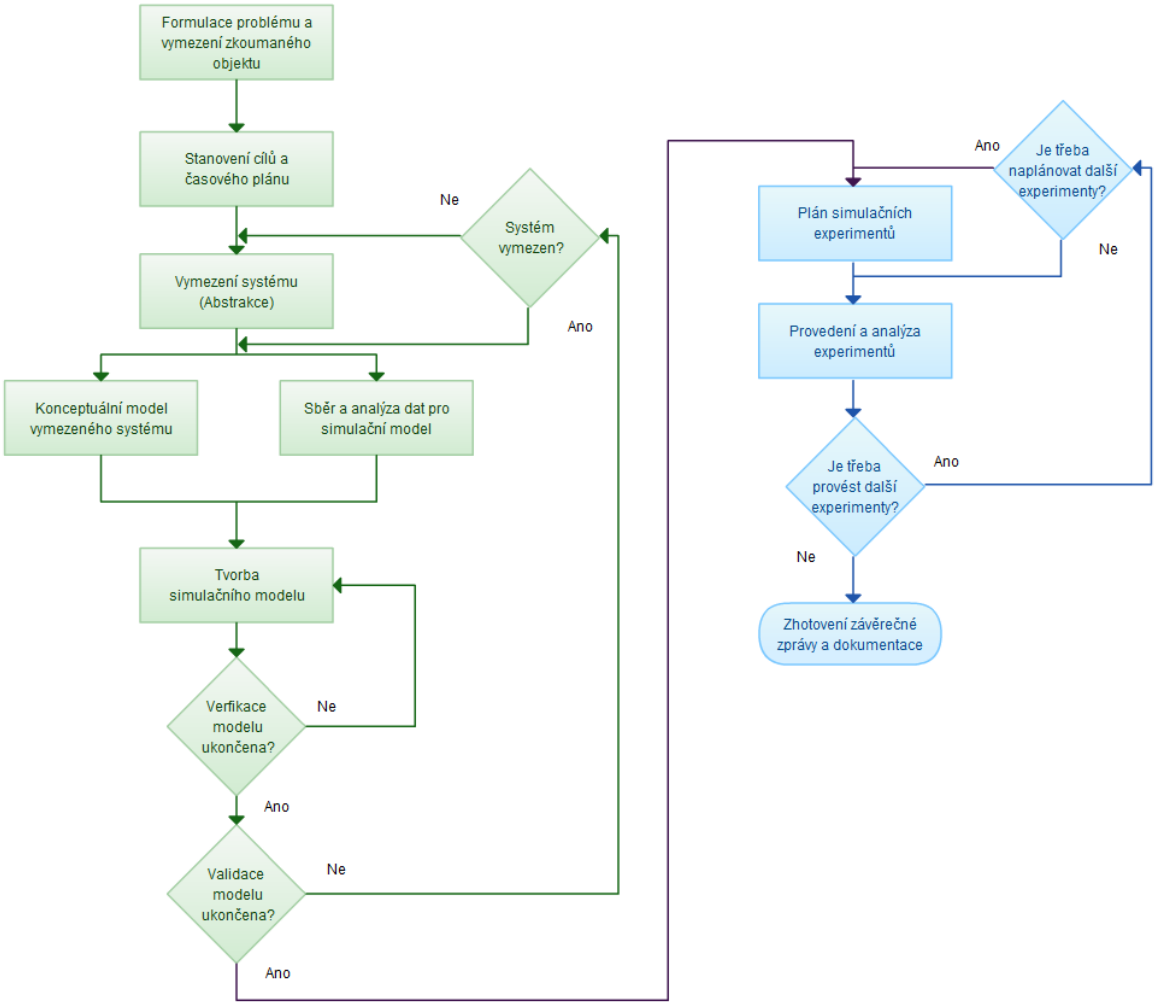
V průběhu simulačního experimentu se zaznamenává čas, který v daném okamžiku simulačního výpočtu odráží čas v originálním systému. Tento čas se nazývá **simulační čas**. Simulační čas obvykle plyne rychleji než reálný čas s tím, že doba provádění jednotlivých aktivit v simulátoru je proporcionální vůči době provádění v originálním systému. Simulační čas nemůže klesat. Okamžik, kdy se simulační čas nemění a dochází k výpočtu, se označuje za **simulační krok**.

1.5 Simulační studie

Série různých simulačních experimentů s různými scénáři, koncipovanými tak, aby co nejlépe zodpověděly zadané otázky, se nazývá **simulační studie** nebo **simulační projekt**. Lze tedy říct, že simulační studie se zaměřuje na studium zkoumaného dynamického systému, vymezeného nad objektem zkoumání, za účelem získání informací či dat o originálním systému, přičemž využíváme výzkumnou metodu – simulaci. Simulační projekt se skládá ze dvou základních etap:

- návrh a tvorba simulačního modelu a
- experimentování s tímto modelem.

Životní cyklus simulačního projektu je znázorněn na následujícím obrázku.



Obrázek 1 - Životní cyklus simulačního projektu[1]

Uvedené etapy lze dále členit do dílčích fází. První etapa (návrh a tvorba simulačního modelu) se dělí následovně:

1. Nejdříve se **zformulují základní problémy** a v případě, že je simulace k řešení tohoto problému vhodná a lze ji aplikovat, se **vymezí objekt zkoumání** po provedení vhodné analýzy.
2. **Stanovení konkrétních cílů** a časového plánu projektu.
3. Na objekt zkoumání je uplatněna **abstrakce**.

[1] Zdroj: [3]

4. Zvolení vhodné **koncepce** pro tvorbu simulačního modelu, v závislosti od zvolené koncepce, se vytvoří **konceptuální model**, který specifikuje základní funkční a řídící části simulačního modelu (zatím se nedefinují detaily). Zákazník by též měl být zahrnut do tvorby tohoto modelu.

5. Paralelně se čtvrtou fází probíhá **sběr a analýza potřebných dat** pro simulační model (můžou být deterministická i stochastická).

6. Samotná **implementace** simulačního modelu.

7. Simulační program je **verifikován**, tj. ověřování správnosti modelu. Pokud průběh simulačního výpočtu odpovídá aktuální představě vyjádřené v konceptuálním modelu, pak je model funkčně správný. Testují se jednotlivé části modelu (např. korektnost interakce procesu, dodržování vztahu kauzalit, korektnost generátoru vstupu apod.). Verifikaci lze také provádět on-line animátorem, který umožňuje sledovat změny stavu simulátoru ve vhodné grafické formě.

8. Simulační program je **validován**, tj. otestování pravdivosti modelu. Zjišťuje se zda, simulátor odráží objekt zkoumání na úrovni přesnosti, která se od něj očekává a která byla určena v cílech projektu. Jinými slovy, zda simulátor představuje takovou substituci reality, která je způsobilá k experimentování. Validaci lze provádět pomocí několika metodik:

    - metoda srovnání s realitou pomocí statistických metod,
    - metoda srovnání s jiným modelem,
    - empirická metoda, kde vnější chování simulátoru je otestováno nezávislým odborníkem (expertem v příslušném oboru), který posoudí, zda je toto chování dostatečně realistické.

Druhá etapa (experimentování) se dělí na tyto fáze:

9. Sestavení **plánu simulačních experimentů.**

10. **Vykonání simulačních experimentů.**

11. Pokud nastaly nové problémy, je třeba se vrátit k realizaci další série experimentů nebo sestavení nového simulačního plánu a ten opět provést. Tudíž nás tato fáze může odkázat na fázi číslo 9. nebo fázi číslo 10.

12. Vyhotovení příslušné **dokumentace.**

1.6 Modelování vstupu simulátoru

Klasifikace modelovacích strategií vstupů simulátoru může být založena na různých kritériích. Kritérium první je rozlišovací úroveň submodelu vstupů a kritérium druhé je náhodnost.

Podle prvního kritéria se modely dělí na dva typy:

1. **Standardní model –** submodel je chápán a realizován jako standardní model, tj. každému prvku P originálu je přiřazen prvek Q simulátoru, každému atribut g a prvku P je přiřazen atribut h prvku Q, kde pro hodnoty atributů g, h je dána nějaká relace.

2. **Metamodel –** submodel vstupů nemá charakter standardního modelu, a tudíž se chová jako černá skříňka, která produkuje požadovaný typ dat.

Podle druhého kritéria (náhodnost) se přístupy k vstupním datům dělí na tyto:

1. **Deterministický přístup** - vyjadřuje proud vstupních dat, která odpovídají nějakým reálným historickým datům. Může se jednat o nějaký sesbíraný soubor reálných historických dat, ale dále nezpracovaných.

2. **Stochastický přístup** – znamená, že vstupní data jsou generována generátorem (realizace umělých náhodných pokusů), který byl vytvořen na základě statistického zpracování reálných dat.

2 Algoritmizace simulačního modelu

Tato kapitola čerpá z publikace (KAVIČKA, 2006, 210s).

2.1 Hlavní úkoly metodiky simulačního modelu

Metodika algoritmizace simulačního modelu se dělí na následující kroky:

1. **Navržení datových struktur** pro reprezentaci stavových hodnot simulátoru a operací (funkcí), které změny těchto hodnot provádějí. Dobře navržené datové struktury (s ohledem na paměť a výpočet) efektivně provádí výpočty během realizace simulačního experimentu. Pro tvorbu těchto struktur se nejčastěji využívá implementační prostředí, které umožňuje aplikovat objektově orientované programování.

2. **Realizaci plynutí času simulátoru**. Čas v simulačním výpočtu se nazývá simulační čas, jak již bylo vysvětleno v kapitole Simulační experiment.

3. **Zajištění synchronizace stavových změn** v simulátoru tak, aby tyto změny probíhaly v určitém pořadí (dodržení vztahu kauzality v simulovaném systému, jak již bylo popsáno v kapitole 1.2).

V prvním bodě metodiky simulačního modelu je zmíněno objektové programování. Mezi jeho hlavní tři vlastnosti patří:

1. **Zapouzdření** zaručuje, že objekt nemůže přímo přistupovat k jeho atributům (i některým metodám), což by mohlo vést k následné nekonzistenci. Objekt navenek zpřístupňuje rozhraní, se kterým se pak s objektem pracuje.

2. **Dědičnost** organizuje objekty hierarchicky (do stromu), kde objekt třídy A může dědit atributy a metody třídy B, které následně může používat. Výsledný potomek třídy A tedy disponuje atributy a metodami obou tříd.

3. **Polymorfismus** zajišťuje, že objekt se chová podle toho jaké třídy, je instance. Pokud několik objektů poskytuje stejné rozhraní, pracuje se s nimi stejným způsobem, ale jejich konkrétní chování se liší podle implementace.

Jelikož při tvorbě simulačního modelu se využívají datové struktury a objektově orientované programovaní, je vhodné co nejvíce struktur naprogramovat abstraktně. To nám umožní znovu využívat již naprogramované struktury pro jiné (nelze ji však využít na všechny) simulační modely. Výsledná abstraktní část simulačního modelu se nazývá **simulační jádro**.

2.2 Metody synchronizace simulačního výpočtu

Způsoby synchronizace simulačního výpočtu můžou být různé. Tato práce se bude soustředit na metodu snímání aktivit.

2.2.1 Metoda snímání aktivit

Metoda snímání aktivit je metodou univerzální, která je použitelná pro realizaci jak spojité tak i diskrétní simulace. Tato metoda se též někdy označuje za dvojfázovou metodu. Přístup metody snímání aktivit se označuje jako synchronní. Její princip spočívá ve snímání všech právě běžících (registrovaných) aktivit v simulátoru. Snímání probíhá v obvykle pevně daných přírůstcích simulačního času. Časové přírůstky τ jsou označovány jako **snímací perioda.** Během snímání každé aktivity dochází k jejímu vyhodnocování, tj. testuje se, zda v simulačním čase $t_S = n * \tau$ (kde n vyjadřuje počet snímacích period od počátku simulace) došlo:

- v případě diskrétní simulace ke splnění výskytu její koncové události nebo
- v případě spojité simulace ke změně hodnot příslušných dynamických atributů a případně ke splnění aktivační podmínky (například k dosažení dané prahové hodnoty atributy nebo výskyt interakce mezi aktivitami) pro vykonání specifické operace.

Jelikož je již definována snímací perioda τ, je zřejmé, že přesnost této metody bude záviset na zvolené délce snímací periody τ. Čím menší snímací periodu nastavíme, tím přesnější bude simulační výpočet. Pro spojitou simulaci je typické, že aktivita je formalizována analytickým popisem (např. soustavou diferenciálních rovnic), přičemž lze principiálně zjistit hodnotu požadovaných atributů v libovolném simulačním čase. Běh simulačního programu, který je založen na metodě snímání aktivit, funguje jako periodické testování aktivační podmínky každé aktivity. Pokud je aktivační podmínka splněna, přirozeně dochází k výkonu příslušné operace.

Pokud by bylo třeba metodu snímání aktivit realizovat, musela by se zavést příslušná ADS, která by obsahovala veškeré registrované aktivity. Dále by bylo třeba zavést datovou strukturu Aktivita. Datová struktura Aktivita by se dala představit takto (za předpokladu, že její aktivační podmínka je dosažení určitého času a terminační podmínka je dosažení nějakých souřadnic):

```
Aktivita = class
  private
    aktivacniPodminka : real;
    ukoncovaciPodminka : point;
  public
    constructor Vytvor;
    destructor  Zrus; virtual;
    procedure   Zpracuj; virtual;
end;
```

Kde metoda „Zpracuj“ vykoná potřebné operace pro danou iteraci cyklu snímání. Pro každý simulační krok by bylo třeba testovat veškeré registrované aktivity, zda splňují aktivační podmínky a následně nad nimi provedly příslušné operace. Následně by se přičetla délka hodnoty periody snímání k celkovému simulačnímu času.

Tabulka 1 - Řídící algoritmus metody snímaní aktivit[2]

Krok	Činnost	Vykonána za podmínek
0	Inicializace simulačního času t_S($t_S = 0$)	
1	Ukončení běhu simulačního programu	Je vyčerpán čas vymezený pro běh simulačního programu
2	**1. fáze** Vyhodnocení všech běžících aktivit vzhledem k času t_S a výkon příslušných akcí v případě, že jsou splněny aktivační podmínky	
3	**2. fáze** Aktualizace simulačního času ($t_S = t_S + \tau$)	
4	Návrat na **krok 1**	

2.2.2 Ostatní metody

Mezi další metody synchronizace výpočtu patří:

a) metoda plánování událostí,

b) metoda interakce procesů

c) třífázová metoda.

Metodou interakce procesů a třífázovou se tato práce zabývat nebude.

Metoda plánování událostí je jednou z nejrozšířenějších metod pro realizaci diskrétní simulace. Přístup metody plánování událostí se označuje jako asynchronní. Jak již bylo dříve definováno, ukončení diskrétní aktivity se nazývá událost. Princip této metody spočívá v plánování událostí do budoucnosti. Jelikož u diskrétní simulace je charakteristické, že doba trvání každé aktivity je předem známa, určíme pro každou aktivitu (diskrétní), při jejím vytvoření, událost, která bude představovat její konec. Tato událost může vytvořit či odstartovat jinou aktivitu. Informace o příslušných událostech je nutné uchovávat v **kalendáři událostí** (nebo časové ose), a to od počátku vytvoření až do jejího výskytu. Kalendář událostí je typicky realizován ADT prioritní fronta.

[2] Zdroj: [3]

Tabulka 2 - Řídící algoritmus metody plánování událostí[3]

Krok	Činnost	Vykonána za podmínek
0	Inicializace simulačního času t_S($t_S = 0$)	
1	Ukončení běhu simulačního programu	Je vyčerpán čas vymezený pro běh simulačního programu nebo kalendář neobsahuje žádné události
2	Odebírání první události z kalendáře (tj. událost s nejmenší hodnotou plánovaného času výskytu t_{u})	
3	Aktualizace simulačního času ($t_S = t_u$)	
4	Výkon akce spojené s výskytem události (akce provádí stavové změny a případné naplánování dalších událostí)	
5	Návrat na **krok 1**	

2.3 Simulátor synchronizovaný s animátorem

Pro následující výklad je nutné zavést pojem animace. Uživatel často požaduje, aby mohl průběh vykonávání aktivit (spojitých i diskrétních) plynule sledovat na obrazovce počítače (např. v čase se měnící polohu mobilního objektu). Z tohoto důvodu je nutné zabezpečit vhodný mechanismus synchronizovaného grafického zobrazování jejich průběhu neboli animaci, jejímž stavovým prostorem je obrazovka počítače.

On-line animace je animace, která zobrazuje vždy aktuální stav simulačního jádra, tj. simulační jádro nemůže provádět svůj obvyklý cyklus, dokud jádro on-line animace nezobrazí stav na obrazovku počítače. Vznikne-li nutnost vytvořit podporu pro on-line animaci (ať už jako požadavek od uživatele nebo z důvodu verifikování modelu), je nutné ji zakomponovat do výsledného simulačního jádra

Pokud simulační model bude obsahovat diskrétní a spojité aktivity, je možné ho synchronizovat několika způsoby, dva takové způsoby jsou:

1. Využít diskrétního jádra simulace, které je založeno na metodě plánování událostí, spolupracující s jádrem spojité simulace, které je založené na metodě snímání aktivit.

2. Využít diskrétního jádra simulace, které je založeno na metodě snímání aktivit.

V rámci této práce je implementována druhá možnost. Jádro simulace, založené na metodě snímání aktivit, je schopné zpracovávat jak aktivity diskrétní, tak aktivity spojité.

[3] Zdroj: [3]

Výslednou strukturu simulátoru, který navíc bude obsahovat volitelnou možnost výkonu animace, lze rozdělit do několika modulů:

- modul simulace,
- modul animace a
- modul rozhraní.

Kde modul rozhraní obstarává komunikaci mezi jednotlivými moduly a modul animace, též založený na metodě snímání aktivit, bude obstarávat animaci aktivit.

Pro další výklad je nutné vědět, jak probíhá zahajování jednotlivých procesů. Zahájení každého procesu je spojeno s výskytem události typu **Start** a jejím následným zpracováním. Pro libovolnou aktivitu procesu (jak spojitou, tak diskrétní) platí, že její ukončení se provede jako událost typu **Konec**, kde při zpracování dojde k zahájení následující aktivity procesu. Zahájení procesů se liší podle typu procesu na dvě varianty:

1. Zahájení diskrétní aktivity je chápáno jako naplánování okamžiku jejího budoucího ukončení, neboli vložení události typu Konec do kalendáře událostí. Časové trvání této aktivity je definováno jako časový interval mezi naplánováním události a okamžikem výskytu této události, kdy jsou provedeny všechny operace (stavové změny této aktivity).

2. Zahájení spojité aktivity je chápáno jako její registrace u modulu spojitého simulátoru. V intervalech simulačního času, kdy je spojitému simulátoru přiřazen čas z diskrétního simulátoru, se spojitý simulátor postará o běh této aktivity, která své ukončení ohlásí prostřednictvím naplánování příslušné události typu Konec do kalendáře událostí v modulu diskrétní simulace.

V následujících podkapitolách bude popsána funkce a činnost těchto tří modulů.

2.3.1 Modul rozhraní

Modul rozhraní zajišťuje vazbu mezi ostatními moduly. Tento modul by měl zajišťovat komunikaci, tj. předávání aktivit jednotlivými jádry. Měl by také obsahovat vyrovnávací paměť, která obsahuje informace o procesech, a to jak od aktivit spojitých, tak diskrétních. Aktivity, u nichž je požadována animace, se ukládají chronologicky v čase do paměti, která obsahuje potřebné informace pro samotné grafické zobrazení. Vyrovnávací paměť je využívána jádrem animace, které z ní průběžně čerpá informace pro animaci aktivit.

2.3.2 Modul simulace

Modul simulace, jak již bylo řečeno, je zodpovědný za simulaci spojitých a diskrétních aktivit. Tento modul musí obsahovat registrované aktivity a jádro snímání aktivit, které obstarává simulační cyklus. Registrace probíhá tak, že při požadavku o registraci modul simulace vloží danou aktivitu do své evidence právě běžících aktivit. Na všechny registrované aktivity je poté aplikována metoda snímání aktivit.

Po spuštění jádra simulace je aktuální čas simulace $\boldsymbol{t_S}$ inicializován na $\boldsymbol{t_S} = 0$ a spustí se snímací cyklus. Je testováno, zda se simulace má ukončit, tj. zda, je vyčerpán vymezený čas pro běh simulace. Aplikace metody snímání aktivit s hodnotou periody snímání $\boldsymbol{\tau^C}$ na všechny registrované aktivity. Po vykonání příslušných operací jsou informace o aktivitách, určených k animování, předány do modulu rozhraní a kontrola nad simulací je předána jádru animace s vymezeným časovým kvantem o velikosti periody snímání. $\boldsymbol{\Delta t = \tau^D}$. V momentě, kdy jádro animace vrátí kontrolu jádru simulace, je vyprázdněna vyrovnávací paměť. Dále probíhá navýšení simulačního času o hodnotu periody snímání $\boldsymbol{t_S = t_S + \tau^D}$ a celý cyklus se opakuje

2.3.3 Modul animace

Modul animace slouží v prezentování průběhu simulačního výpočtu, tj. jednotlivých aktivit na obrazovce. Pro další popis modulu animace je důležité definovat animační aktivitu. **Animační aktivita** je programová rutina (procedura), která provádí animační zobrazování daného objektu na obrazovce. Jelikož je modul animace založený na metodě snímání aktivit (animačních aktivit), musí podobně jako modul simulace obsahovat jádro snímání aktivit, tzv. **jádro animace** a registrované aktivity (animační), které postupně vykonávají dílčí animační úkony s animovanými objekty.

Po aktivaci jádra animace je čas animace $\boldsymbol{t_A}$ inicializován na $\boldsymbol{t_A} = 0$ a spustí se animační cyklus, který bude provádět tzv. „ex-post“ animaci po dobu přiděleného časového kvanta $\boldsymbol{\Delta t}$. Proběhne registrace příslušných animačních aktivit u modulu animace a dále je aplikována metoda snímání aktivit s hodnotou periody snímání $\boldsymbol{\tau^A}$ na všechny registrované animační aktivity. Hodnota $\boldsymbol{\tau^A}$ je typicky menší než hodnota $\boldsymbol{\tau^D}$. Při testování jednotlivých aktivit je provedena odpovídající operace pro danou animační aktivitu. Po vyčerpání časového kvanta $\boldsymbol{\Delta t}$ modul animace předává kontrolu modulu simulace.

Vizuální interaktivní simulace – jedná se o simulaci umožňující experimentátorovi přepínat mezi dvěma režimy:

- **automatickým** režimem simulačního výpočtu, kde výpočet probíhá předdefinovaným algoritmem a
- **interaktivním** režimem simulačního výpočtu, který může sloužit pro navrhování alternativních řešení problémů identifikovaných v průběhu simulačního výpočtu samotným experimentátorem a nikoliv pouze předdefinovaným automatickým algoritmem.

Použití on-line animace je zcela nezbytné z důvodu zobrazení aktuálního stavového prostoru simulátoru na obrazovce počítače.

3 Implementace simulačního modelu

K implementaci simulačního (simulujícího) modelu byl použit programovací jazyk Java.

3.1 Filozofie implementace simulátoru a animátoru

Tato práce se zaměřuje na realizaci vzorové implementace jádra animace spolupracující s jádrem simulace založené na metodě snímání aktivit. Implementovaný simulátor je založen na diskrétní simulaci využívající metody snímání aktivit. Výsledné jádro je tedy schopné zpracovávat diskrétní i spojité aktivity. Z tohoto důvodu odpadá nutnost modulu diskrétní simulace a z modulu spojité simulace se stává „modul simulace“. Celková struktura simulátoru potom vypadá následovně:

- **modul simulace**, který zajišťuje simulaci všech aktivit,
- **modul animace**, který zajišťuje animování aktivit a
- **modul rozhraní**, který obsahuje vyrovnávací paměť a obstarává komunikaci mezi všemi moduly.

Zpracování diskrétní aktivity probíhá podobně jako zpracování spojité aktivity. Po dobu, kdy je aktivita registrována, je na ní aplikována metoda snímání aktivit, zda je v dané periodě splněna její ukončovací podmínka.

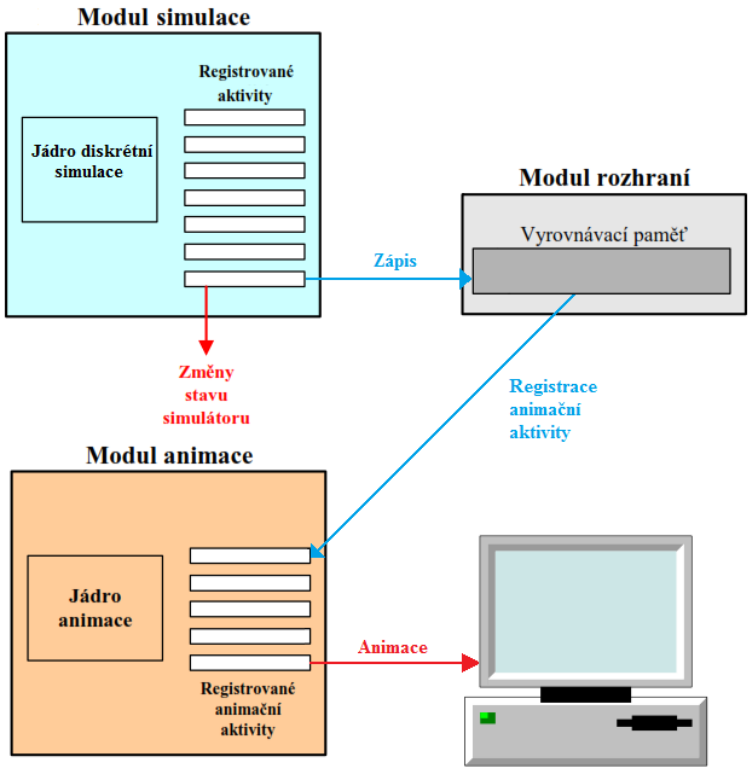
Vozidlo, které přijede k obsluze a naplánuje se jeho čas odjezdu, se dá klasifikovat jako diskrétní aktivita, tedy událost odjezdu z fronty. Pokud by se tato aktivita zpracovávala spojitě, tak by se časová hodnota výskytu události (odjezd auta z obsluhy) nastavila jako prahová hodnota ukončovací podmínky. Tato aktivita se zařadí k registrovaným aktivitám v jádře simulace a každou iteraci cyklu snímání aktivit je testována tato podmínka, zda je aktivita ukončena tedy, zda má uvolnit místo v obsluze a zda má naplánovat další aktivitu odjezdu z obsluhy.

Obvyklá vzorová implementace, jádra animace spolupracující s jádrem simulace založené na metodě snímání aktivit, provádí každou iteraci cyklu snímání tyto operace:

1. Testuje se skutečnost, zda nová entita vstoupí do systému, v případě, že ano, zaregistruje se k právě běžícím aktivitám.
2. Na všechny právě běžící aktivity je následně použita metoda snímání, která nad nimi následně provádí operace (např. výpočet nové polohy jedoucího vozidla).
3. Všechny aktivity, které jsou určeny k animaci, předají své informace modulu rozhraní ve formě animačních aktivit, ze kterého si je odebírá modul animace.
4. Testování, zda je jejich ukončující podmínka splněna (např. dojezd na určité místo). Pokud ano, jsou aktivity deregistrovány (odebrány z právě běžících aktivit a kompletně odstraněny).

5. Navýší simulační čas o časovou hodnotu periody snímání a vrací se na Krok 1.

Modul animace realizovaný též metodou snímání aktivit by měl podobný algoritmus jako jádro simulace.



Obrázek 2 - Implementovaná struktura simulátoru s animátorem[4]

Vlastní simulační program, zajišťuje **on-line animaci**. To znamená, že uživatel musí „vidět“ na obrazovce stav jádra simulace (např. polohy aut, počet entit ve frontě, apod.) za poslední periodu a jádro simulace nesmí počítat dál, aby uživatel neviděl stav simulace před několika snímacími periodami, ale aktuální stav simulace. Z tohoto důvodu jádro simulace, které implementuje metodu snímání aktivit, provede jednu periodu snímání. Informace aktivit určených pro animaci (z této periody) předá do modulu rozhraní jako animační aktivity a odevzdá řízení nad simulací jádru animace s přiděleným kvantem Δt. Velikost tohoto kvanta musí být rovna délce jedné periody snímání tj., $\Delta t = \tau^D$. Následně jádro simulace čeká. Jádro animace tedy vždy vybere výsledky jedné periody z vyrovnávací paměti a dále implementuje metodu snímání aktivit na všechny tyto převzaté animační aktivity a následně nad nimi provádí operace, tedy jejich animování. Poté jádro animace předá kontrolu jádru simulace.

[4] Zdroj: [3]

Tabulka 3 - Řídící algoritmus implementovaného řešení simulátoru a animátoru[5]

Krok	Vykonává	Činnost	Vykonána za podmínek
0	Jádro simulace	Inicializace simulačního času $\boldsymbol{t_S}$($\boldsymbol{t_S = 0}$)	
1		Ukončení běhu simulačního programu	Vymezený čas pro běh simulace je vyčerpán
2		Aplikace metody snímání aktivit se snímací periodou $\boldsymbol{\tau^C}$ na všechny zaregistrované spojité aktivity	
3		Vymezení časového kvanta $\boldsymbol{\Delta t = \tau^D}$	
4	Jádro animace	Provedení registrace animačních aktivit převzatých z vyrovnávací paměti	$\boldsymbol{\Delta t \neq 0}$
5		Aplikace metody snímání aktivit se snímací periodou $\boldsymbol{\tau^A}$ na všechny zaregistrované animační aktivity	$\boldsymbol{\Delta t \neq 0}$ a počet zaregistrovaných aktivit $n_A > 0$
6	Jádro simulace	Vydání pokynu k vyprázdnění vyrovnávací paměť	Vyrovnávací paměť je neprázdná
7		Aktualizace simulačního času ($\boldsymbol{t_S = t_S + \tau^D}$)	
8		Návrat na **krok 1**	

3.2 Implementace metody snímaní aktivit

Metoda snímání aktivit je implementována v jádře simulace i v jádře animace.

3.2.1 V modulu simulace

Tato podkapitola se bude věnovat modulu spojité simulace, tedy modulu jádra simulace. Metoda snímání aktivit je implementována ve třídě **JadroSimulace**, která obsahuje informace (atributy) o stavu simulace, čas jádra simulace, kolekci (ArrayList[6]) právě probíhajících aktivit, hodnotu snímací periody a časový limit simulace a všechny operace s nimi spojené. Hlavní metoda třídy JadroSimulace je metoda „snímej“, která obstarává snímací cyklus simulátoru. A ten realizuje plynutí simulačního času a volá jednotlivé kroky algoritmu metody snímání aktivit. Zjednodušená struktura třídy vypadá takto:

```
public class JadroSimulace<T extends IAktivita>{
  private ArrayList<T> aktivityProbihajici = null;
  private int snimaciPerioda=1;
  private double casJadraSnimani = 0;
```

[5] Zdroj: Vlastní

[6] ArrayList – uspořádaná kolekce typu seznam, která slouží k uchování proměnného počtu objektů. ArrayList obsahuje příslušné metody pro manipulaci s jeho prvky (implementuje rozhraní List<E>)

```
  private double casovyLimit=0;
  private StavoveHodnoty stav=null;
  …
  public void snimej(){
      do{
            testovaniAktivacniPodminkyAktivit ();
            testovaniUkoncovaciPodminkyAktivit();
            spocitejRegistrovaneAktivity();
            vlozAnimacniAktivityDoVyrovnavaciPameti();
            predejKontroluNadSimulaciProCasoveKvantum(snimaciPerioda);
            vydaniPrikazuKVymazaniVyrovnavaciPameti();
            casJadraSnimani=casJadraSnimani+snimaciPerioda;
      } while (casJadraSnimani < casovyLimit);
  }
…
}
```

Třída JadroSimulace je naprogramována genericky, tzn. je zavedena vrstva abstrakce, která umožňuje vyhnout se psaní opakujícího se kódu, a to zavedením parametrizace, kde parametr T je určitý datový typ. V případě této implementace simulačního jádra to znamená, že za parametr T v deklaraci třídy JadroSimulace může být jakýkoliv typ entity, tj. třída, která implementuje rozhraní **IAktivita** a zaručuje tím, že jádro simulace s ní bude umět pracovat, a to například voláním metody „zpracuj“ jednotlivých aktivit, které jsou typu T.

```
private void spocitejRegistrovaneAktivity(){
      Iterator it = aktivityProbihajici.iterator();
      T aktivita=null;
      while(it.hasNext()){
      aktivita = (T)it.next();
      aktivita.zpracuj();}
…
}
```

Modul simulace mimo třídu JadroSimulace obsahuje třídu StavoveHodnoty, která uchovává hodnoty o simulaci (např. obsazenost fronty u obsluhy).

Obrázek 3 - UML diagram tříd modulu simulace[7]

Samotný běh modulu simulace je iniciován uživatelem z inicializačního okna. V momentě, kdy se tak stane a konstruktor inicializoval potřebné atributy (délku simulace, snímací periodu), je spuštěna metoda „snímej“, která následně prochází kolekci vstupních aktivit, zda je splněna jejich aktivační podmínka (obvykle dosažení prahové hodnoty času). Pokud ano, zaregistruje příslušné aktivity k právě běžícím aktivitám. Dále testuje registrované aktivity, zda nemají být ukončeny (obvykle dosažením určitých souřadnic), pokud ano,

[7] Zdroj: Vlastní

deregistruje příslušné aktivity z evidence právě běžících aktivit, zapíše jejich potřebné statistiky (čas ve frontě a čas v systému) a následně je úplně odstraní.

Zpracování aktivity určené k animaci vygeneruje příslušnou animační aktivitu a uloží do kolekce animačních aktivit určených k předání do vyrovnávací paměti. Po dokončení snímání je celá kolekce animačních aktivit předána do vyrovnávací paměti modulu rozhraní a jádro simulace se vzdá kontroly nad simulací (kterou přebere jádro animace) po dobu hodnoty časového kvanta Δt a dále čeká. Aby byla zajištěna **on-line animace**, je hodnota časové kvanta Δt pokaždé rovna délce uživatelem zvolené snímací periody. Jinými slovy „to co se zpracuje, se hned zanimuje“.

Po dokončení příslušné animace (která bude popsána v další kapitole) jádro simulace opět přebírá kontrolu, provede inkrementaci času jádra snímání o hodnotu periody snímání, vydá příkaz k vyprázdnění vyrovnávací paměti v modulu rozhraní, zkontroluje, zda nemá být simulace pozastavena (změnou booleovské proměnné) nebo ukončena (čas jádra snímání musí být roven určenému časovému limitu) a celý cyklus se opakuje.

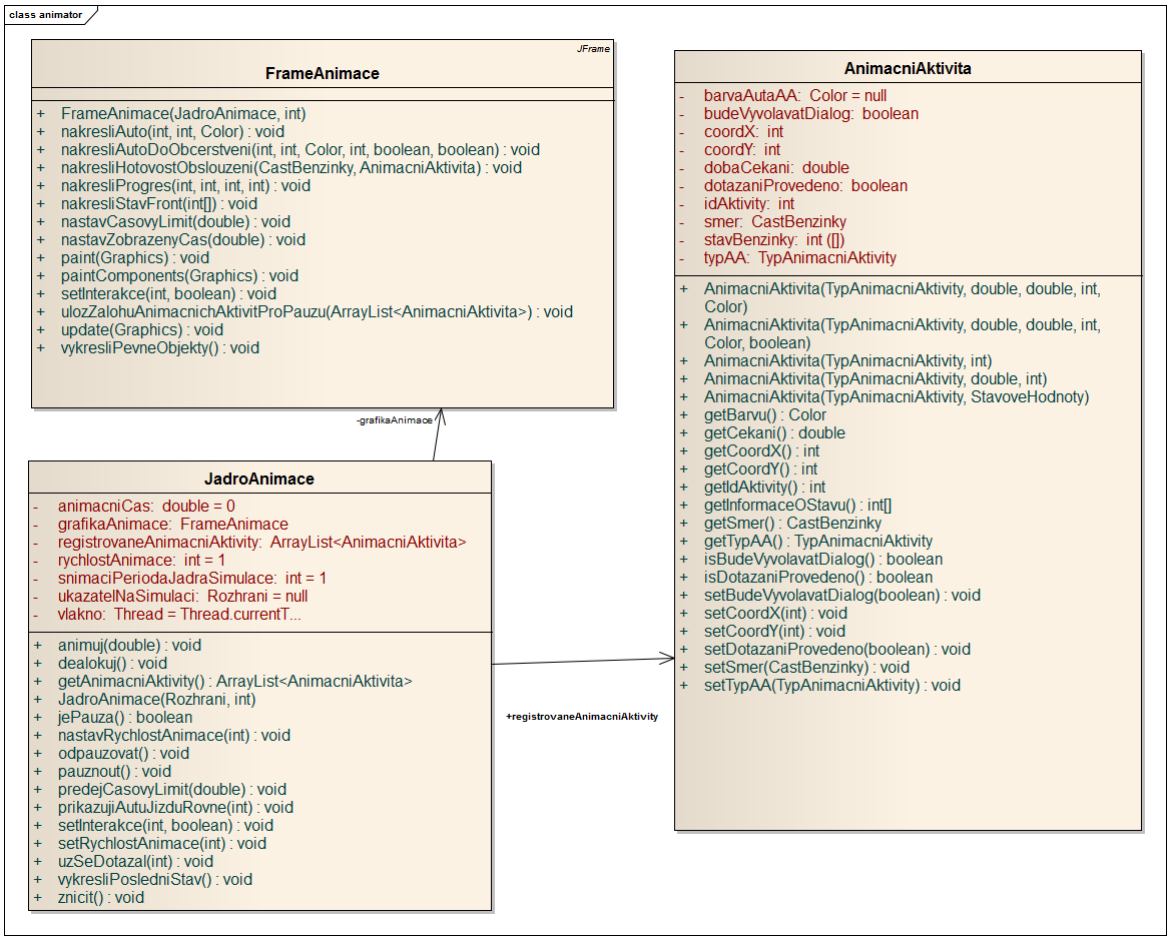
3.2.2 V modulu animace

V případě modulu animace je metoda snímání aktivit implementována ve třídě **JadroAnimace**, která obsahuje informace (atributy) o rychlosti animace, snímací periodě (animování), kolekci registrovaných animačních aktivit a animační čas. Dále je také třeba zmínit, že tato třída obsahuje asociaci na třídu **FrameAnimace**. Třída FrameAnimace je potomek třídy JFrame a obsahuje grafickou část programu, tj. obsahuje metody pro vykreslení entit, zobrazení stavů obsluh apod. Hlavní metoda třídy JadroAnimace je metoda „animuj“, která spouští metodu snímaní aktivit nad všemi registrovanými animačními aktivitami po dobu časového kvanta přiděleného z jádra simulace (obvykle o délce jedné snímací periody). Zjednodušená struktura třídy by se dala napsat takto:

```
public class JadroAnimace{
  private double animacniCas = 0;
  private FrameAnimace grafikaAnimace = null;
  private int snimaciPeriodaJadraSimulace=1;
  private int rychlostAnimace=1;
  private ArrayList<AnimacniAktivita> registrovaneAnimacniAktivity =null;
  …
  public void animuj (double prideleneCasoveKvantum){
        double cas=0;
        do{
          uspaniVlaknaNaDobuUrcenouZRychlostiAnimace();
          registraceAnimacnichAktivitZVyrovnavaciPameti(animacniCas);
          animujAnimacniAktivityPomociMetodySnimaniAktivit();
          cas=cas+snimaciPeriodaJadraSimulace;
          animacniCas=animacniCas+snimaciPeriodaJadraSimulace;
        }while(cas < prideleneCasoveKvantum);

  }
  …
}
```

V předchozí podkapitole byl vysvětlen snímací cyklus modulu simulace. Bylo řečeno, že modul simulace se vzdá kontroly nad řízením simulačního programu po dobu určeného časového kvanta Δt (o hodnotě jedné snímací periody).



Obrázek 4 - UML diagram tříd modulu animace[8]

V momentě, kdy jádro animace přebírá kontrolu, se děje následující. Nejprve je vlákno simulačního programu uspáno na určitou dobu odvozenou z atributu rychlosti animace, tím se realizuje rychlost animování. Dále jsou animační aktivity pro danou periodu vybrány z vyrovnávací paměti rozhraní a registrovány do kolekce (typu ArrayList) registrovaneAnimacniAktivity. Další část obstarává samotný JFrame, kterému je přidělen animační čas k zobrazení a je zavolána metoda „paint“, která vypadá následovně:

```
@Override
public void paint(Graphics g) {
   inicializujDoubleBufferedPlochu();
   animujScenar();
   g.drawImage(offscreen,0,0,this);
}
```

[8] Zdroj: Vlastní

Tato metoda je zodpovědná za vykreslení animačních aktivit. Pro tento účel využívá tzv. techniky **DoubleBuffering** tzn., vytvoří prázdnou plochu (obrázek) typu Image. Do této plochy následně vykreslí všechny animační aktivity zvlášť. Jakmile jsou vykresleny, překreslí tímto obrázkem plochu, kde má probíhat animace. Takto probíhá zpracování animačních aktivit každé periody a výsledkem je on-line animace.

Po vykreslení animačních aktivit dané periody, proběhne inkrementace animačního času o hodnotu periody snímání. Dále je zkontrolováno, zda je vyčerpáno přidělené časové kvantum Δt, v případě, že není, opakuje se celý cyklus a pokud ano, ukončí svůj cyklus a kontrola se vrací jádru simulace.

Implementovaný animátor podporuje **situační interakci**, tzn., v případě, že je fronta obsluhy příslušného směru, ve kterém se entita pohybuje, příliš dlouhá, je uživateli nabídnuta možnost změnit chování této entity, tj. místo jízdy do obsluhy je této entitě rozkázáno, aby pokračovala v jízdě rovně.

3.3 Implementace simulačního modelu

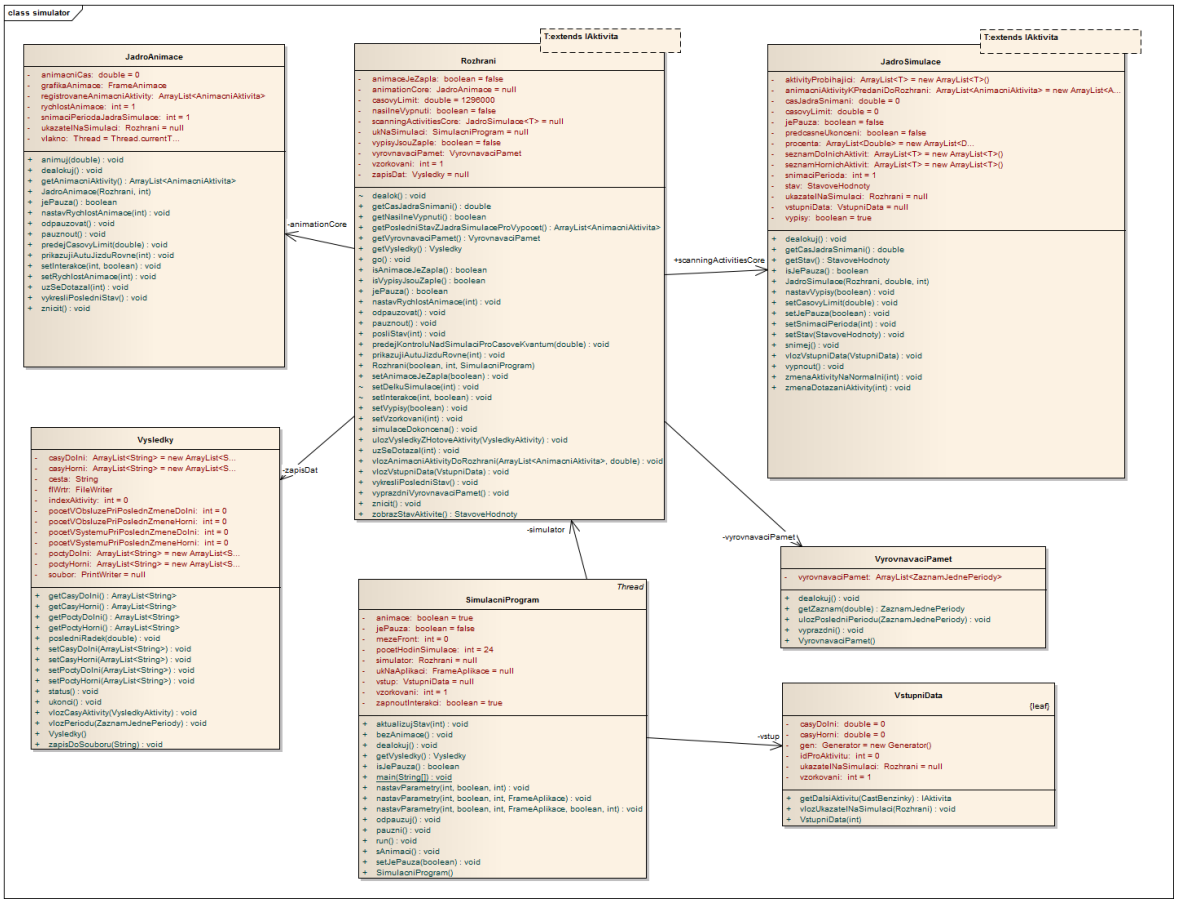
Tato kapitola se bude věnovat implementaci simulačního modelu. Pro implementovaný simulátor je elementární aktivita, definována dvěma body, a to vjezd do systému a odjezd ze systému. Její jednotlivé úseky se počítají jinak, podle toho na jakých souřadnicích se nacházejí. Třída **EntitaAuto** obsahuje atribut typu aktivita (kde aktivita může být auto, které pojede jenom rovně a auto, které pojede do čerpací stanice), tj. atribut typAktivity typu enum[9]. Třída **VstupniData** obsahuje všechny aktivity, které budou vstupovat do simulátoru, které jsou chronologicky seřazené. Pro vlastní implementaci to znamená, že metoda snímání aktivit nemusí procházet celou kolekci aktivit, ale kontroluje pouze jedinou aktivitu v režii, zda je splněna aktivační podmínka pro danou periodu a dále nic neprochází (uvažuje se, že aktivity mají mezi sebou časové okno). V případě, že aktivita splňuje aktivační podmínku, je aktivita zaregistrována k právě běžícím aktivitám a do režie (ve třídě VstupniData) se vygeneruje příjezd nové entity. Výsledkem je značné urychlení výpočtu simulace.

Implementovaný generátor pseudonáhodných čísel, který generuje intervaly mezi příjezdy, využívá **exponenciální rozdělení pravděpodobnosti**. Zjednodušená struktura třídy **Generator** a metody pro výpočet intervalu vypadá následovně:

```
public class Generator {
  private Random sadaHodnot;
  …
    private double vypocetExpoRozdeleniPrijezdy(double a) {
      double U = sadaHodnot.nextDouble();
      double x  = (-a)*Math.log (1-U);
     return x;
    }
}
```

[9] Enum (enumrration) je výčtový typ, který umožňuje vytvářet vlastní datové typy, které mohou nabývat pouze určitého omezeného množství hodnot.

Hodnoty vygenerovány implementovaným generátorem byly ověřeny v Arena Input Analyzer (verze 13.5.0) od společnosti Rockwell Automation, Inc. Po vygenerování sto tisíc hodnot z vlastního generátoru byl proveden příslušný test a vygenerované hodnoty se přibližují k exponenciálnímu rozdělení pravděpodobnosti s velikostí disperze (rozptyl[10]) $\sigma^2 = 0{,}000008$ a po vygenerování miliónu hodnot se hodnoty skutečně přibližují k exponenciálnímu rozdělení pravděpodobnosti s velikostí disperze $\sigma^2 = 0{,}000001$.



Obrázek 5 - UML diagram tříd simulačního modelu[11]

Aktivita typu EntitaAuto, která musí implementovat rozhraní IAktivita, obsahuje metodu „zpracuj“, při jejímž spuštění dochází k příslušnému spočítání souřadnic. Každý typ aktivity (která jede do benzínky a která jede rovně) je zpracován svým způsobem. V případě aktivity „normální“ výpočet probíhá jednoduše a to tak, že při každé iteraci cyklu snímání aktivit jsou aktualizovány souřadnice pozice auta. V případě aktivity, která jede do benzínky, je výpočet rozdělen na tři části (podle toho na jaké souřadnici osy x se nachází):

[10] Rozptyl je definován jako střední hodnota kvadrátů odchylek od střední hodnoty.
[11] Zdroj: Vlastní

1. Nejprve entita jede do benzínky, tedy pouze probíhá výpočet nových souřadnic.
2. V momentě, kdy dojede entita do benzínky, je zkontrolováno, jestli je obslužná jednotka volná a v případě že ano, tak tato entita obsadí obsluhu a bude zpracovávána. V případě, že tomu tak není, je tato entita zařazena do fronty. Dále je testováno, jestli je obsluhování dané aktivity dokončeno. Pokud ano, vypočtou se nové souřadnice a entita uvolní obsluhu. Každou iteraci je prováděno stejné vyhodnocování nad aktivitami, které se nachází v benzínce.
3. Entita odjíždí z benzínky a jako u prvního bodu pouze probíhá výpočet nových souřadnic. V momentě, kdy dojede na koncovou souřadnici je ukončovací atribut přepnut na ukončený (jádro simulace, bude vědět, že tuto aktivitu má odstranit z evidence právě běžících aktivit).

Po dokončení výpočtu dané aktivity je vytvořena příslušná animační aktivita (např. aktivita „vykresli auto na daných souřadnicích“), která se po dokončení iterace cyklu snímaní aktivit vloží do vyrovnávací paměti.

3.4 Ověření funkčnosti simulátoru

Funkčnost implementovaného simulátoru, tj. především metody snímání aktivit byla ověřena porovnáním vlastních výsledků s výsledky systému hromadné obsluhy M/M/1. Jedná se o triviální simulační model, který obsahuje pouze jednu obslužnou jednotku. Tou procházejí všechny aktivity, které do systému vstoupí. Při nakonfigurování intenzity vstupního proudu na 1 transakci za jednotku času a intenzitu obsluhy na 10/9 transakcí za jednotku času musí systém M/M/1 podávat tyto hodnoty:

Tabulka 4 - Exaktní řešení systému M/M/1[12]

Zkoumaná statistika	Vzorec	Požadovaná hodnota	Jednotky
Intenzita vstupního proudu:	$\boldsymbol{\lambda}$	1	[zákazníků/čas. jednotek]
Intenzita obsluhy:	$\boldsymbol{\mu}$	10/9	[zákazníků/čas. jednotek]
Využití linky obsluhy:	$\boldsymbol{\rho = (\lambda/\mu) * 100}$	90%	[%]
Průměrný počet zákazníků v systému:	$\boldsymbol{L = \lambda/(\mu - \lambda)}$	9	[zákazníků]
Průměrný pobyt zákazníka v systému:	$\boldsymbol{W = 1/(\mu - \lambda)}$	9	[čas. jednotek]
Průměrný pobyt zákazníka ve frontě:	$\boldsymbol{W_Q = \lambda/[\mu(\mu - \lambda)]}$	**8,1**	[čas. jednotek]

[12] Zdroj: [3]

Implementovaný simulátor byl nakonfigurován na odpovídající intenzitu vstupního proudu a obsluhy. Poté byla provedena dostatečně dlouhá simulace (75 600 000 jednotek jádra simulace, to odpovídá 1400 hodinám) a výsledky vypadaly následovně:

Tabulka 5 - Výstupní hodnoty M/M/1 z implementovaného simulátoru[13]

Vzorec	Hodnota	Jednotky
$\boldsymbol{\rho_E} \cong$	90,05%	[%]
$\boldsymbol{L_E} \cong$	9,019	[zákazníků]
$\boldsymbol{W_E} \cong$	9,027	[minut]
$\boldsymbol{W_{Q_E}} \cong$	**8,103**	[minut]

Implementovaný simulátor byl také ověřen, zda animátor ovlivňuje výpočet simulátoru. Byly provedeny dvě simulace se stejnou násadou generátoru pseudonáhodných čísel, tj. generátor v obou případech generoval stejná čísla, kdy jedna simulace se provedla bez spuštění animátoru a druhá simulace se provedla se spuštěným animátorem. Simulace v obou případech běžela po dobu 540 000 jednotek simulačního času (to odpovídá 10 hodinám). Výsledky, při stejných vstupních hodnotách, byly identické. Animátor tedy nemá vliv na vlastní výpočet simulátoru.

[13] Zdroj: Vlastní

4 Vymezení objektu zkoumání

Vymezení objektu zkoumání – tato práce zkoumá měnící se polohy mobilních prostředků na malém úseku silnice, která obsahuje benzínku s tzv. drive-in obslužným systémem v obou směrech.

Vymezení systému – k simulaci tohoto systému je třeba zavést několik abstrakcí:

- každá mobilní entita se ve směru osy x pohybuje konstantní rychlostí,
- čas přejezdu entity z fronty obsluhy k obslužné jednotce je nulový,
- příjezdy entit jsou řízeny generátorem pseudonáhodných čísel založeným na exponenciálním rozdělení pravděpodobnosti, to znamená, že se neuvažují různé vstupní hodnoty v různou denní dobu,
- vygenerované časy příjezdů zajišťují, že nemůže nastat srážka, tj. vygenerovaná hodnota času příjezdu je inkrementována o jednu sekundu,
- doba obsluhy každé entity je taktéž generována generátorem pseudonáhodných čísel založeným na exponenciálním rozdělení pravděpodobnosti.

Jsou sledovány čtyři hodnoty:

- průměrný počet entit v systému,
- průměrný počet entit ve frontě,
- průměrný pobyt entity v systému,
- využití linky obsluhy.

5 Experimenty se simulátorem

Generátor pseudonáhodných čísel založený na exponenciálním rozdělení byl nastaven tak, aby střední délka mezi příjezdy entit byla 11,3 sekund (170 jednotek jádra simulace) a střední délka obsluhy entity byla 40 sekund (600 jednotek jádra simulace). Generátor dále generuje hodnoty pro určení, zda se jedná o auto, které odbočí do obsluhy anebo auto, které bude pokračovat v jízdě. Pravděpodobnost, že auto odbočí, je rovna 25%. V tomto případě se řídí rovnoměrným rozdělením.

Bylo provedeno několik experimentů se stejnou násadou pro generátor pseudonáhodných čísel s různými hodnotami periody snímání a různou délkou simulace. Na následujících tabulkách je ukázán vývoj stabilizování středních hodnot sledovaných statistik.

Tabulka 6 - Výsledky simulace při snímací periodě 10 s[14]

Délka simulace [hodin]	Průměrný pobyt v systému [min]	Průměrný pobyt ve frontě [min]	Průměrný počet entit v systému [entit]	Využití linky obsluhy [%]
0,1667	0,78	0,18	3	99,94
1	0,64	0,15	3,83	82,78
12	0,81	0,31	4,33	76,69
72	0,85	0,34	4,55	79,05
168	0,86	0,35	4,61	79,79
720	0,9	0,39	4,82	81,04
2000	0,92	0,4	4,89	81,33

Tabulka 7 - Výsledky simulace při snímací periodě 1 s[15]

Délka simulace [hodin]	Průměrný pobyt v systému [min]	Průměrný pobyt ve frontě [min]	Průměrný počet entit v systému [entit]	Využití linky obsluhy [%]
0,1667	0,8	0,30	2,01	99,92
1	0,68	0,23	3,44	77,22
12	0,96	0,52	4,61	73,66
72	0,99	0,54	4,75	77,33
168	1	0,55	4,83	78,26
720	1,09	0,63	5,24	80,11
2000	1,12	0,67	5,40	80,39

[14] Zdroj: Vlastní
[15] Zdroj: Vlastní

Tabulka 8 - Výsledky simulace při snímací periodě 1/15 s[16]

Délka simulace [hodin]	Průměrný pobyt v systému [min]	Průměrný pobyt ve frontě [min]	Průměrný počet entit v systému [entit]	Využití linky obsluhy [%]
0,1667	0,8	0,31	2,01	99,28
1	0,69	0,24	3,40	77,87
12	0,98	0,55	4,68	74,05
72	1,01	0,57	4,82	77,82
168	1,03	0,58	4,90	78,77
720	1,12	0,67	5,37	80,62
2000	1,16	0,71	5,54	80,9

Přesnost výsledků simulace závisí na dvou faktorech:

- délce zvolené periody snímání a
- dostatečně dlouhé trvání simulace.

[16] Zdroj: Vlastní

Závěr

Cílem této bakalářské práce bylo vzorově realizovat implementaci jádra počítačové animace spolupracující s jádrem simulace založené na metodě snímání aktivit.

K dosažení tohoto cíle byly vysvětleny některé pojmy z oboru modelování a simulace potřebné k následujícímu popsání a porozumění filozofie implementace jádra spojité simulace a jádra animace.

Byla vysvětlena filozofie metody snímání aktivit, včetně její implementace na realizovaném simulátoru a animátoru. Vlastní implementace simulátoru a animátoru, který zvládá on-line animaci a zaručuje tím situační interakce s jádrem simulace, byla také vysvětlena.

V poslední řadě se s realizovaným simulačním programem experimentovalo za různých délek period snímání a simulace. Z výsledků je zřejmé, že metoda snímání aktivit je přesnější při menší snímací periodě, tím se značně zvyšuje výpočetní doba. Pro vstupní hodnoty byl využit generátor pseudonáhodných čísel, který byl založen na exponenciálním rozdělení pravděpodobnosti. Po určité době plynutí simulace se střední hodnoty sledovaných statistik začaly stabilizovat ve všech pokusech pod různými snímacími periodami.

Implementované jádro animace spolupracující s jádrem simulace bylo úspěšně realizováno.

Literatura

[1] **BANKS, Jerry. 1998, 850s.** *Handbook of simulation.* New York : John Wiley & Sons, 1998, 850s. ISBN 0-471-13403-1.

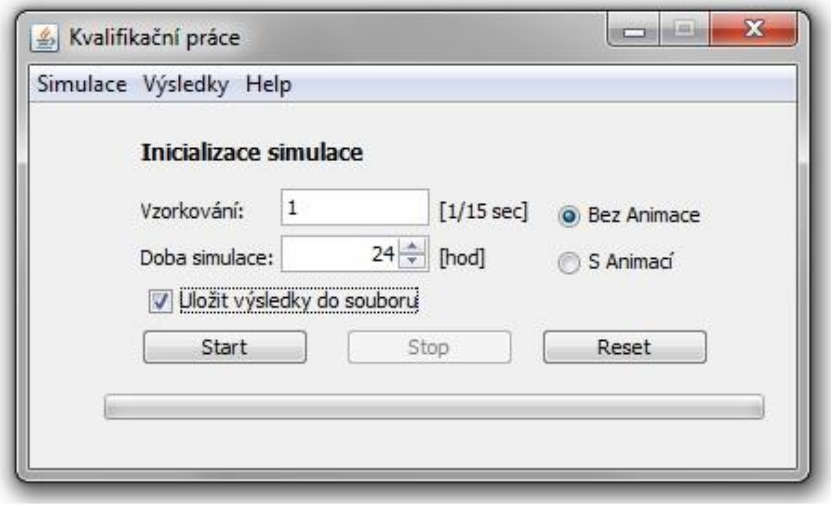
[2] **HUŠEK, R., LAUBER, J. 1987, 349s.** *Simulační modely.* Praha : STNTL/ALFA, 1987, 349s.

[3] **KAVIČKA, A., KLIMA, V., ADAMKO, N. 2006, 210s.** *Agentovo orientovaná simulácia dopravných uzlov.* Žilina : EDIS, 2006, 210s. ISBN 80-8070-477-5.

[4] **KŘIVÝ, Ivan a Evžen KINDLER. 2001.** Simulace a modelování. [Online] 2001. [Citace: 12. 4. 2013.] Dostupné z: prf.osu.cz/kip/dokumenty/Msm.pdf.

Příloha A – Manuál k aplikaci

Ovládání celé aplikace je velice intuitivní. Po spuštění Bakalarka_Jiri_Adam_2013.jar se spustí následující okno.

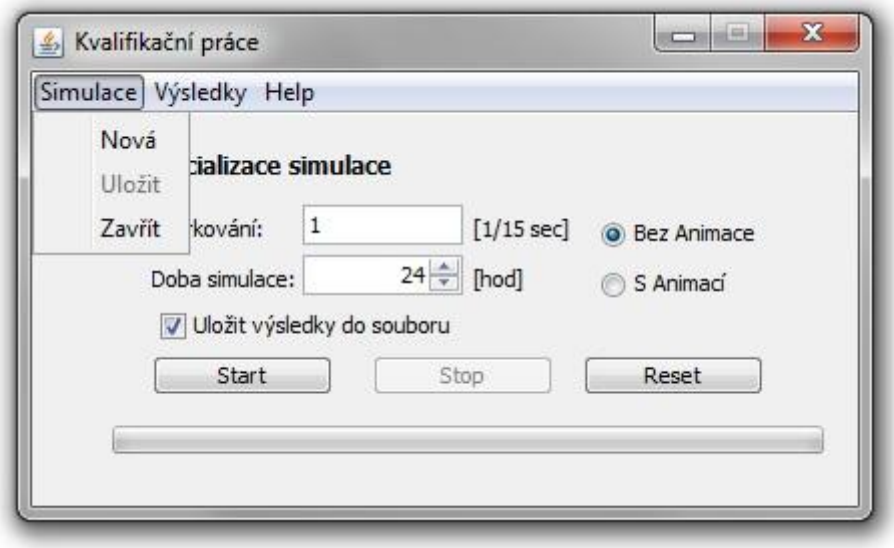


Jedná se o inicializační okno pro simulace. Zde se může nastavit vzorkování neboli snímací perioda, která je implicitně nastavena na 1/15 sekundy (minimální hodnota). Vzorkování nastaveno na 15 se tedy rovná jedné sekundě. Nastavení doby simulace je v hodinách.

Dále je možnost simulaci spustit s animací nebo bez a také na uložení výsledků do souboru po dokončení simulace. Po dokončení každé simulace jsou výsledky ještě uchovávány v paměti a dají se dodatečně také uložit do souboru.

Tlačítko Start spustí simulaci podle nastavených parametrů. Tlačítko Stop pozastaví simulaci (dokončí právě probíhanou periodu). Tlačítko Reset zruší poslední simulaci a dealokuje údaje o poslední simulaci z paměti. Dole je lišta, která představuje průběh simulace.

Na dalších snímcích je popsáno menu.

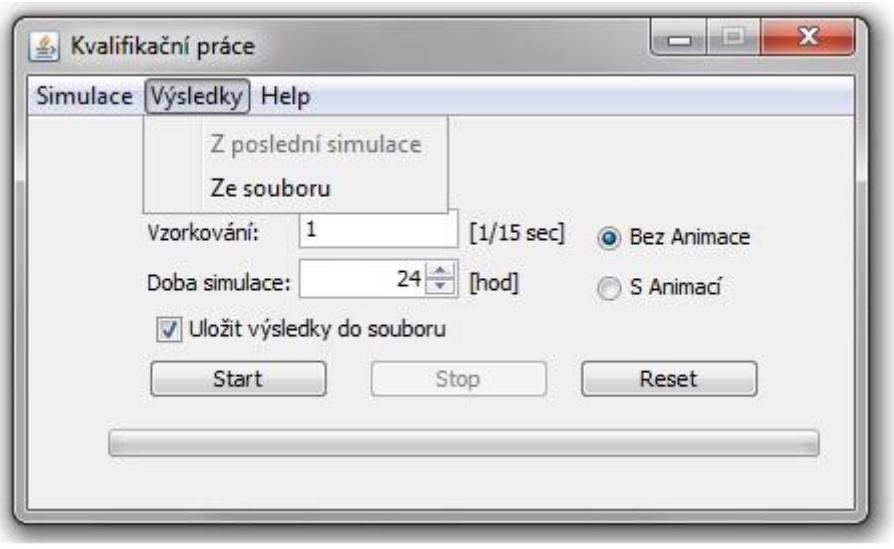


Záložka „Simulace“ obsahuje následující položky:

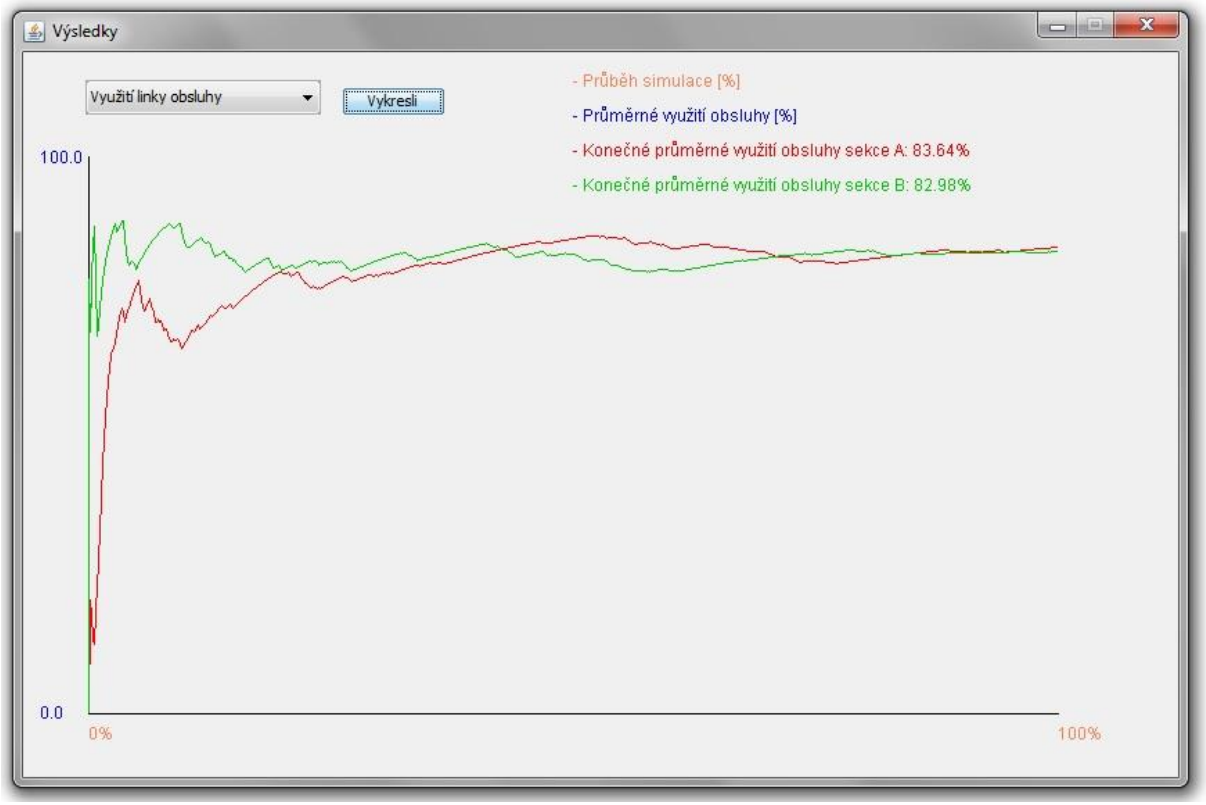
- Nová – je ekvivalent tlačítka Reset
- Uložit – uloží výsledky z poslední hotové simulace do souboru. (uživatelem specifikovaná cestu a název)
- Zavřít – ukončí aplikaci

Záložka „Výsledky“ obsahuje položky:

- Z poslední simulace – otevře okno výsledků s daty z poslední simulace
- Ze souboru – otevře okno výsledků s daty ze souboru (uživatelem zvoleným)



Po zobrazení výsledků se vytvoří následující okno.

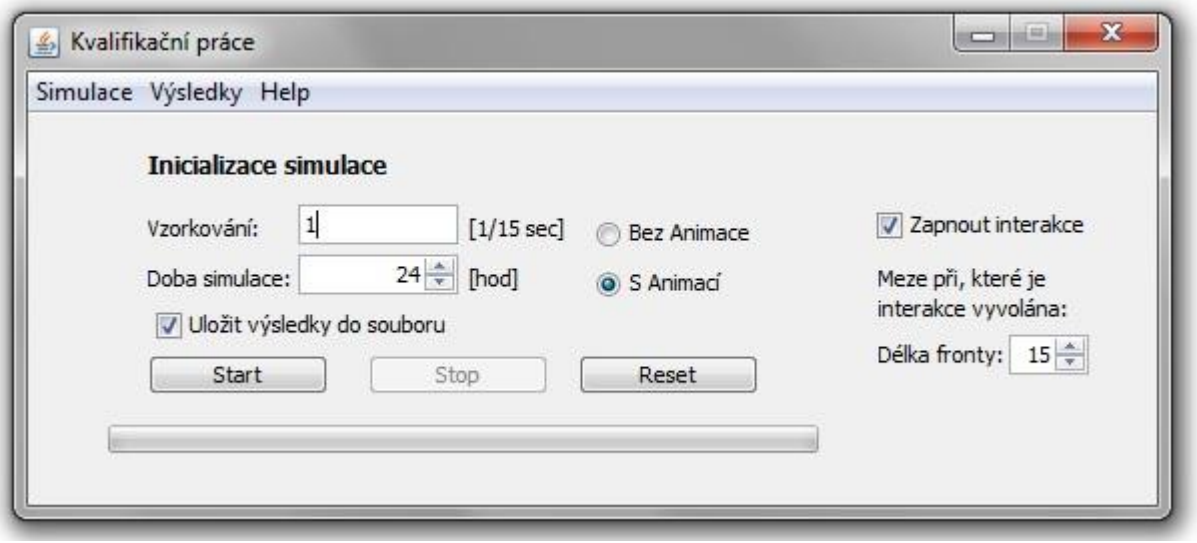


Okno výsledků vykresluje grafy pro každý směr dopravy samostatně, sledují se následující statistiky:

- průměrnou pobyt v systému,
- průměrnou pobyt ve frontě,
- průměrný počet entit v systému,
- průměrné využití linky obsluhy.

Jednotlivé směry dopravy jsou označeny jako „sekce A“ a „sekce B“.

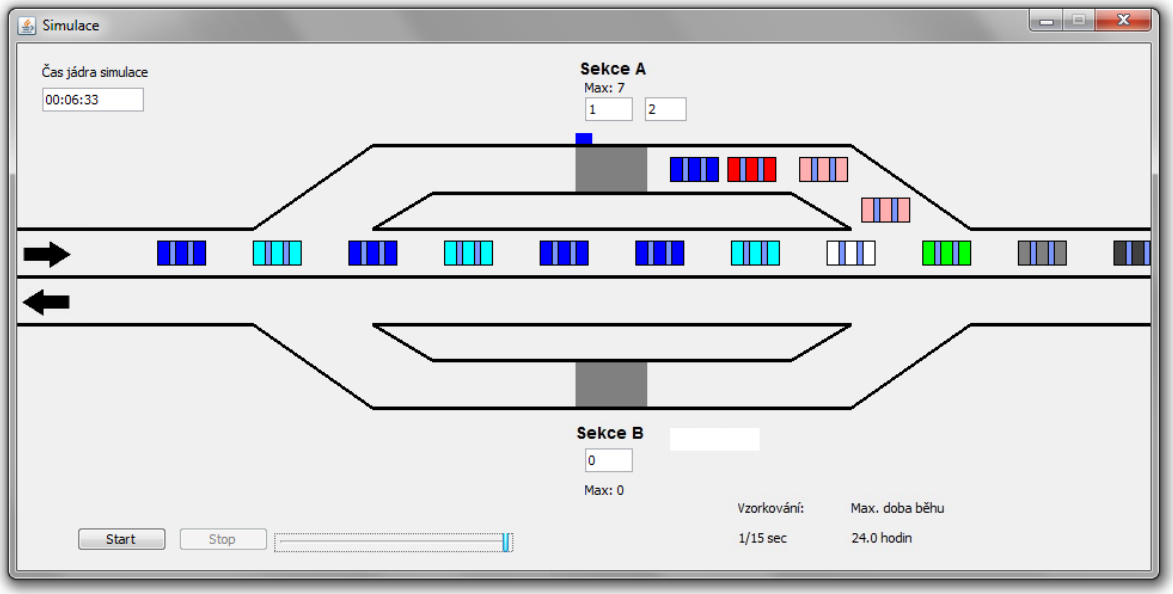
Při zvolení simulace s animací je možnost nastavit meze, pro interakce se simulátorem, tj. pokud je fronta v daném směru příliš dlouhá, bude experimentátorovi nabídnuta možnost změnit chování dané entity. Tyto interakce se dají vypnout.



Po spuštění simulace s animací se spustí okno, kde probíhá samotná animace.

Toto okno umožňuje ovládat rychlost simulace (tím pádem i animace). Také můžeme simulaci pozastavit tlačítkem Stop a opět ji spustit tlačítkem Start. Tlačítka v inicializačním okně zůstávají funkční. Okno zobrazuje frontu obsluhy, vzorkování, čas jádra simulace a maximální dobu běhu simulace.

V případě, že vstupní proud je příliš vysoký, se zviditelní pomocné komponenty na zobrazení fronty při odjíždění (viz obrázek níže). Na tomto snímku je aplikace testována, a proto je spodní část aplikace vypnuta.



Ve výsledku tedy čeká na uvolnění cesty 4 auta na samotné vozovce + 2 auta v „bloku“ obsluhy, které odjedou, jakmile se uvolní místo.

V případě, že jsou interakce spuštěny a dojde k jejich aktivování je vyvolán odpovídající dialog.