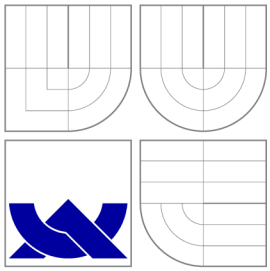
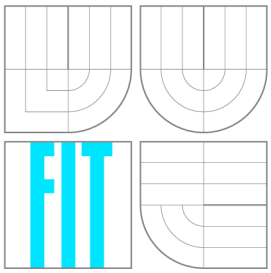


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ŘÍDICÍ SYSTÉM DIVADELNÍ SCÉNY S GRAFICKÝM ROZHRANÍM

THEATRE STAGE CONTROL SYSTEM WITH GRAPHICS INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DAMIÁN MELO
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. PAVEL ZEMČÍK
SUPERVISOR

BRNO 2010

Abstrakt

Tématem této bakalářské práce je řízení technologie scény divadel. V práci jsou rozebrány metody a algoritmy řízení elektrických motorů a pohonů, specifické požadavky na divadelní techniku a postupy pro simulaci. V rámci práce byly implementován simulátor jeviště navazující na uživatelské rozhraní.

Abstract

The topic of the bachelor work is control of theatre stage technology. The methods and algorithms of control of electric motors and other action elements are discussed along with specific requirements for theatre technology and its simulation. Also, a simulator of stage connected to the user interface was implemented.

Klíčová slova

jeviště, divadlo, LexoCad, simulace, regulace, PID

Keywords

stage, theater, LexoCad, simulation, regulations, PID

Citace

Damián Melo: Řídicí systém divadelní scény s grafickým rozhraním, bakalářská práce, Brno, FIT VUT v Brně, 2010

Řídicí systém divadelní scény s grafickým rozhraním

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Pavla Zemčíka.

.....

Damián Melo

16. mája 2010

Poděkování

Rád by som sa poďakoval vedúcemu mojej bakalárskej práce, Doc. Dr. Ing. Pavlovi Zemčíkovi, za cenné rady a podnety pri tvorbe tejto práce a Antonínovi Bučkovi za pomoc s programom LexoCad.

Obsah

Kapitola 1

Úvod

Činnosť, ktorá neustále núti ľudstvo vyvíjať sa, je snaha uľahčiť si svoju prácu. Či už išlo o vytváranie jednoduchých nástrojov a pascí na lov prehistorických zvierat, neskoršie zavádzanie manufaktúr pre zefektívnenie výroby, alebo zavádzanie prvých parných a iných strojov a zariadení od počiatku priemyselnej revolúcie. Práve od tohto okamžiku nastáva otázka, ako nielen prenechať náročnú prácu strojom a zariadeniam, ale dokázať, aby pracovali samostatne bez častého zásahu ľudskej obsluhy. Už v roku 1784 sa podarilo Jamesovi Wattovi zostrojiť parný stroj s automatickou reguláciou jeho rýchlosti.

Táto práca sa venuje využitiu automatických prvkov v divadlách a pri divadelných predstaveniach. Zameriava sa na možnosť jednoduchého, ale presného a spoľahlivého pohybu javiskových komponent. Pod pojmom komponenta sa myslí časť hracej plochy javiska, alebo rôzne predmety a zariadenia zavesené nad ňou (časti kulís, svetlá a pod.). Na ich pohyb sú kladené veľké nároky. Musí byť plynulý, často je vyžadovaný súčasný pohyb niekoľkých komponent súčasne, ale rôznymi smermi a rýchlosťami. Naplánovať takýto pohyb je veľmi náročné. Preto je súčasťou práce aj jednoduchý simulátor umožňujúci naplánovať a otestovať pohyby komponent na počítači a až potom ich aplikovať na reálnom javisku.

Druhá kapitola práce obsahuje jednoduchý teoretický úvod do problematiky javiskovej mechanizácie – popis javiskovej techniky, požiadavky na jej pohyb a metódy používané na jeho reguláciu. Tretia kapitola sa zaoberá teóriou simulácii a modelovania. Štvrtá kapitola, nazvaná Analýza a návrh, zhodnocuje poznatky získané v predchádzajúcich dvoch kapitolách a navrhuje konečné riešenie práce. Piata kapitola popisuje skutočnú implementáciu mojej práce. Šiesta, predposledná kapitola, obsahuje výsledky záverečných testov. Posledná kapitola, Záver, obsahuje celkové zhodnotenie práce, možnosti jej ďalšieho pokračovania a vylepšení.

Kapitola 2

Javisková technika

> *Javiskovou technikou sa v rámci divadelnej prevádzky rozumie súbor technických prostriedkov, ktoré sa podieľajú na tvorbe, stavbe a zmenách divadelných scén.*[4]

Podľa požiadaviek na technické vybavenie a funkčné možnosti ju delíme na[2]:

- Scénickú mechaniku
- Scénické osvetlenie
- Elektroakustické ozvučenie

V práci sa budeme ďalej bližšie zaoberať len scénickou mechanikou.

2.1 Scénická mechanika

Vybavenie javísk, respektíve divadiel scénickou mechanikou je rôzne. Rozsah vybavenia, jeho technologická úroveň ako aj množstvo použitých prvkov je individuálne a záleží na finančných a priestorových možnostiach divadla, ale aj na využiteľnosti daných technológií. Napriek tomu, takmer v každom divadle nájdeme spoločné základné prvky. Scénická mechanika sa zvykne deliť do dvoch základných skupín, kde deliacou líniou je hracia plocha javiska[2]:

- Horná scénická mechanizácia
- Dolná scénická mechanizácia

Horná scénická mechanizácia

Nachádza sa nad hracou plochou javiska a je divákovi väčšinou skrytá. Základnými prvkami sú:

- Opona – Na javisku sa nachádza viacero rôznych typov opôn, ktorých funkcie sú zrejmé z ich názvu[2]:

  - požiarna opona
  - akustická opona
  - slávnostná opona
  - hracia opona

- Ťahy – Využívajú sa k vertikálnemu pohybu nad javiskom. Slúžia na zavesenie a následnú manipuláciu napríklad s časťami kulís alebo osvetlovacích a iných zariadení. Z fyzikálneho hľadiska ide o kladku (kladkostroj) umožňujúci spúšťať časti kúlís z priestoru nad javiskom na javisko a naopak. Delíme ich na bodové a prospektové ťahy.[4]
  - Bodové – Na konci bodového ťahu je upínacie zariadenie (hruška), na ktoré sa zavesí jednotlivé zariadenie prípadne samostatná časť kulís.[2]
  - Prospektové – Prospektový ťah je zakončený ťahovou tyčou rovnako dlhou ako je šírka javiska, na ktorú sa môže zavesiť viacero kulís, poprípade jedna väčšia kulisa upnutá na viacerých miestach (obrázok 2.1).[2]

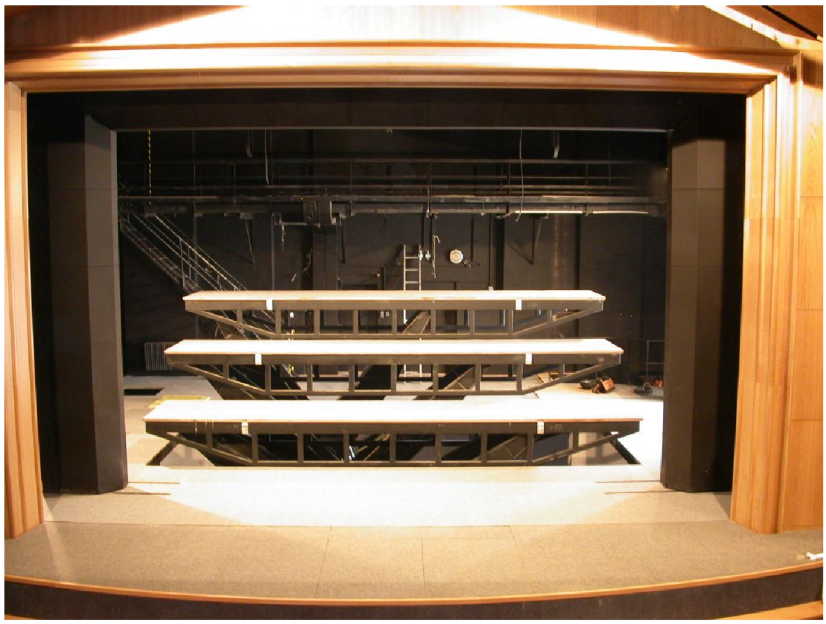
Obrázok 2.1: Prospektové ťahy nad javiskom[4]

Dolná scénická mechanizácia

Tvorí hraciu plochu javiska.[2] Základnými prvkami sú:

- Točňa – Nemusí byť súčasťou každého javiska. Využíva sa na otáčavý pohyb scény, jej rýchlo prestavbu či simuláciu pohybu.[4] Existuje viacero typov: tanierové, valcové, kazetové, skladacie...

- Zdvíhací stôl – Časť hracej plochy javiska s možnosťou samostatného pohybu vo vertikálnom smere. Vzájomnou kombináciou rôznych polôh viacerých javiskových stolov sa dajú docieliť rôzne terénne efekty (schody, jamy, a pod.). Niektoré stoly umožňujú ich nakláňanie v jednej alebo dvoch osiach, čím sa dajú vytvoriť napríklad stúpania a klesania javiskovej plochy.[4] Podľa funkcie ich delíme[2]:
  - Stoly hracej plochy – ich nastavením do rôznych polôh sa vytvárajú scénografické aspekty predstavenia (obrázok 2.2)
  - Stoly orchestriska – slúžia na vytvorenie podlahy orchestriska a nastavenie jej výšky vzhľadom k hracej ploche.
- Prepad – výťahové zariadenia umožňujúce rýchle objavenie sa respektíve zmiznutie osôb, prípadne vecí, na hracej ploche[2]



Obrázok 2.2: Javiskové stoly
Zdroj: *http://www.elseremo.com/galerie.php*

2.2 Pohyb javiska

Od pohybu javiskovej techniky, či už stolov alebo ťahov, sa vyžaduje čo najrýchlejší presun požadovanou rýchlosťou na vopred zadanú polohu. Pohyb musí byť čo najplynulejší, aby pri prudkých zmenách rýchlosti nedošlo k prílišnému zaťaženiu ťahových lán, alebo k poškodeniu a deformácii kulís.[4] Graf priebehu dráhy, rýchlosti a zrýchlenia je na obrázku 2.3. Pri výpočte aktuálnej rýchlosti vzhľadom na požadovaný priebeh dráhy sa vychádza

zo základných fyzikálnych vzťahov:[3]

$$v = \frac{ds}{dt} \qquad a = \frac{dv}{dt}$$

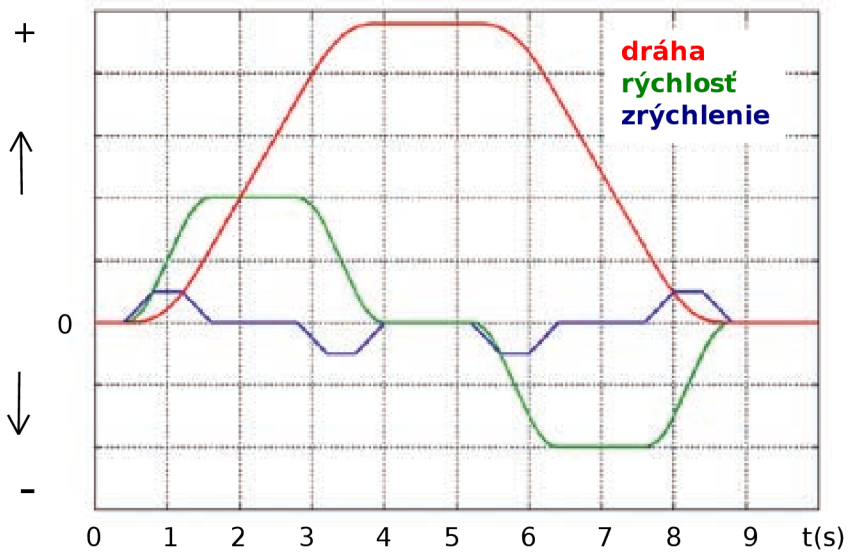
Analogicky na základe vzťahu $\frac{dx}{dt} = \int x dt$ môžme definovať:

$$s = \int v dt \qquad v = \int a dt$$

Pre celkovú dráhu pohybu v čase t platí vzorec:

$$s = s_0 + v_0 t + \frac{1}{2} a t^2$$

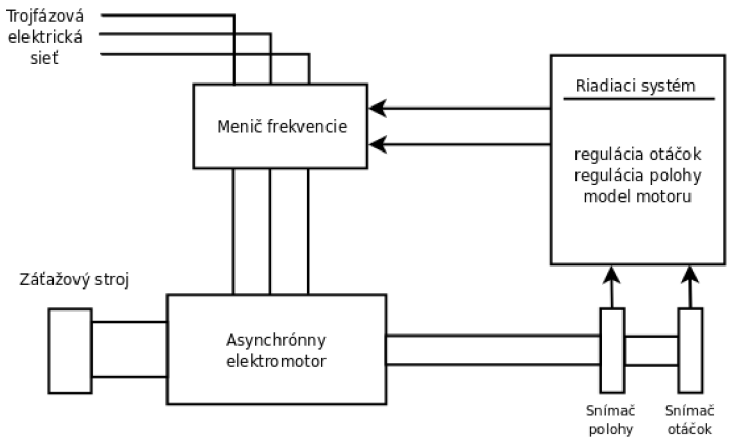
Pre vytvorenie plynulého pohybu, t.j. plynulého rozbehu na požadovanú rýchlosť a následného zotrvania na nej do okamžiku plynulého zastavenia, je potrebné, aby sa zrýchlenie a menilo na základe lineárnej funkcie, viď 2.3. To dosiahneme konštantnou deriváciou zrýchlenia, hodnotou ryvu (trhnutia). Pri zväčšovaní zrýchlenia bude táto konštanta kladná, pri zmenšovaní záporná s rovnakou absolútnou hodnotou a pri nemennom zrýchlení bude hodnota ryvu nulová.



Obrázok 2.3: Graf pohybu[4]

2.3 Akčné členy pohybu

Pre dosiahnutie požadovaných vlastností pohybu, je potrebné, aby sa systém zabezpečujúci tento pohyb skladal prinajmenšom z komponentov, zobrazených na obrázku 2.4. Kde záťažový stroj predstavuje hornú alebo dolnú časť mechanizácie (pohybovú hriadeľ, bubon na navíjanie lana a pod.)



Obrázok 2.4: Diagram pohybových komponent

Asynchrónny elektromotor

Elektromotor je zariadenie, v ktorom sa elektrická energia mení na energiu mechanickú.[5] V súčasnosti všeobecne najpoužívanejší druhom elektromotoru je asynchrónny elektromotor. Vyznačuje sa veľkou variabilitou možných výkonov, jednoduchou konštrukciou a ľahkou údržbou. V javiskovej technológii sa používa na pohyb ťahov, stolov aj prepadlísk. Skladá sa z pohyblivej časti – rotoru a nepohyblivej časti – statoru. Pri priechode striedavého prúdu statorom sa indukuje točivé magnetické pole. To má za následok indukciu napätia na rotore a následný vznik prúdu, ktorý vyvolá otáčavý pohyb rotora. Otáčky rotora sú neustále o niečo menšie ako otáčky točivého magnetického poľa, sú asynchrónne. Rozdiel medzi otáčkami magnetického poľa a skutočnými otáčkami rotoru sa volá sklz.[5] Otáčky sú dané vzťahom:

$$n = \frac{60f}{p}(1 - s) \tag{2.1}$$

kde f je frekvencia napájacieho napätia a s je sklz rotoru. Premenná p je počet pólov statoru a závisí na konkrétnej konštrukcii motoru. Zo vzorca 2.1 je vidieť, že otáčky rotoru sú závislé na frekvencii napájacieho napätia a veľkosti sklzu. Zmenou frekvencie vstupného napätia sa zmení rýchlosť točivého magnetického poľa a tým aj veľkosť otáčok rotoru. Avšak pri následnom zaťažení sa zväčší sklz rotoru a tým klesnú aj jeho otáčky. Na ich vyrovnanie je potrebné zvýšiť výkon motoru. To dosiahneme zmenou (zväčšením) elektrického prúdu. Správnym nastavením frekvencie napájacieho napätia a veľkosťou prúdu môžeme s asynchrónnym motorom dosiahnuť ľubovoľných otáčok. Na ich nastavenie a reguláciu sa v súčasnosti používajú meniče frekvencie.

Obrázok 2.5: Asynchrónny elektromotor
Zdroj: *http://www2.nord.com/cms/cz/product_catalogue/motors/motors_detail_1535.jsp*

Menič frekvencie

Menič frekvencie je zariadenie, umožňujúce pomocou elektronických obvodov meniť sieťové napätie s frekvenciou 50 Hz na napätie s inou frekvenciou a tým nastavovať nižšie alebo vyššie otáčky točivého magnetického poľa elektromotoru. Na dosiahnutie požadovaných hodnôt výstupnej frekvencie sa používajú takzvané PID – regulátory (sekcia 2.4).

Obrázok 2.6: Frekvenčný menič
Zdroj: *http://www.sae.org/mags/sohe/TOOLS/7054*

Snímače

Snímače sú zariadenia, ktoré zisťujú stav alebo hodnotu určitej fyzikálnej veličiny, túto hodnotu transformujú do inej fyzikálnej veličiny a jej pomocou informujú nadriadený proces.[8] Podľa informácii o hodnote meranej veličiny ich môžeme deliť na:

- Relatívne – snímajú relatívny stav meranej veličiny vzhľadom ku okamžiku spustenia merania.
- Absolútne – Snímajú absolútny stav meranej veličiny v rámci jej krajných hodnôt. Snímač si pamätá stav veličiny aj po svojom vypnutí.

Podľa spôsobu snímania ich môžme deliť na:

- Analógové – snímajú stav veličiny (dráha, rýchlosť, napätie) a prevádzajú jej hodnoty na elektrické signály.
- Binárne – majú binárny vstupný signál (spojený/rozpojený kontakt, napätie 0/10 V). Vyhodnocujú, či je snímaná veličina nad, alebo pod nastavenou prahovou úrovňou. Na základe toho vykonajú, alebo nevykonajú určitý akciu.
- Číslicové – majú číslicový výstupný signál. Nameranú analógovú veličinu zdigitalizujú pre využitie v číslicových zariadeniach.

Obrázok 2.7: Rotačný snímač polohy
Zdroj: *http://www.saze.cz/e6c2cwz1x1000/automatizace/371.html*

2.4 Regulácia

Regulátor

Regulátor je zariadenie, ktoré na základe rozdielu medzi aktuálnym stavom regulovanej veličiny $y(t)$ a jej požadovaným stavom $w(t)$ vykonáva zásahy na úpravu tejto odchýlky.

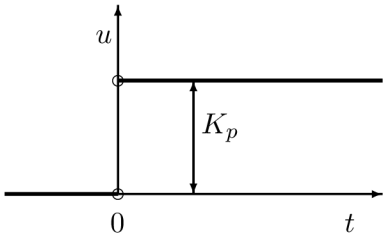
$$e(t) = w(t) - y(t) \tag{2.2}$$

Na základe vypočítanej hodnoty regulačnej odchýlky $e(t)$ je vypočítaná hodnota akčnej veličiny $u(t)$, ktorá má vplyv na zmenu regulovanej veličiny. Cieľom zmeny akčnej veličiny je zmenšiť regulačnú odchýlku alebo ju úplne odstrániť.[1] Hodnota akčnej veličiny je počítaná na základe matematického zákona, ktorý sa môže líšiť podľa regulovanej sústavy. V súčasnosti najpoužívanejší a najrozšírenejší je model PID regulátora. Aktuálna hodnota akčnej veličiny je vypočítaná ako súčet P, I a D zložky.[1]

Proporcionálna (P) zložka

$$u_p(t) = K_p e(t) \tag{2.3}$$

Proporcionálna zložka akčnej veličiny je v každom okamihu úmerná aktuálnej regulačnej odchýlke.[6] Proporcionálna konštanta regulátoru K_p môže pôsobiť ako zosilnenie, pri $K_p > 1$, alebo ako zoslabenie, pri $0 < K_p < 1$, proporcionálnej časti regulátoru. Pri $k_p = 0$ nebude mať proporcionálna zložka vplyv na reguláciu.

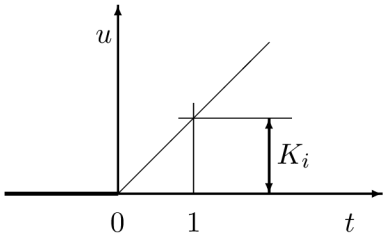


Obrázok 2.8: Prechodová charakteristika P zložky

Integračná (I) zložka

$$u_i(t) = K_i \int_0^t e(\tau) d\tau \tag{2.4}$$

Integračná zložka akčnej veličiny je v každom okamihu úmerná časovému integrálu aktuálnej regulačnej odchýlky.[6] Pri zápornej regulačnej odchýlke (požadovaná hodnota je menšia ako aktuálna) sa bude integračná zložka znižovať, pri kladnej odchýlke zvyšovať. Ak sa regulačná odchýlka ustáli na nulovej hodnote, integračná zložka bude rovná hodnote akčnej veličiny potrebnej na udržanie regulovanej veličiny na požadovanej hodnote. Pri príliš veľkej integračnej konštante K_i bude po dosiahnutí nulovej regulačnej odchýlky integračná zložka príliš veľká a regulovaná hodnota prekročí požadovanú hodnotu. Regulovaná hodnota sa po niekoľkých prekmitoch ustáli na požadovanom stave. Pri správne nastavenej konštante regulovaná hodnota taktiež prekročí požadovanú hodnotu, ale iba jedenkrát. Pri $K_i = 0$ nebude mať integračná zložka vplyv na reguláciu.[9]

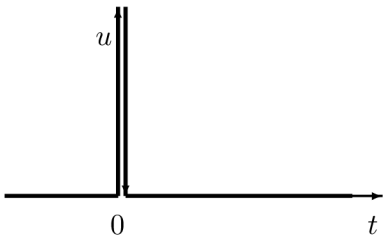


Obrázok 2.9: Prechodová charakteristika I zložky

Derivačná (D) zložka

$$u_d(t) = K_d \frac{de(t)}{dt} \tag{2.5}$$

Derivačná zložka akčnej veličiny je v každom okamihu úmerná derivácii aktuálnej regulačnej odchýlky.[6] Z matematického hľadiska predstavuje derivácia regulačnej odchýlky smernicu jej priebehu. Pri prudkej zmene odchýlky bude derivačná zložka veľká, pri miernej zmene nepatrná, pri nulovej odchýlke nulová. Čím väčšia bude derivačná konštanta K_d,tým rýchlejšia bude odozva derivačnej zložky na zmenu požadovanej hodnoty. Pri $K_d = 0$ nebude mať derivačná zložka vplyv na reguláciu.



Obrázok 2.10: Prechodová charakteristika D zložky

Celková hodnota akčnej veličiny

Celková hodnota aktuálnej akčnej veličiny $u(t)$ PID regulátoru sa rovná súčtu rovníc 2.3, 2.4 a 2.5.[1]

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2.6}$$

Nastavením niektorých konštánt K_p, K_i, K_d na nulovú hodnotu môžeme z výpočtu eliminovať príslušné zložky. Dostaneme tak základné druhy regulátorov: P–regulátor, I–

regulátor. Samostatný D–regulátor sa nepoužíva. Analogicky môžeme dostať aj PI, PD a PID–regulátory.

Diskrétny PID–regulátor

Vyššie uvedené vzťahy platia pre použitie analógového PID–regulátoru. Pre vytvorenie regulačného výpočtu na číslicovom zariadení, je potrebné previesť vzťahy 2.3, 2.4 a 2.5 do diskrétnej podoby.

$$u_p(T) = K_p e(kT) \tag{2.7}$$

$$u_i(T) = K_i T \sum_{i=0}^{k} e(iT) \tag{2.8}$$

$$u_d(T) = K_d \frac{e(kT) - e[(k-1)T]}{T} \tag{2.9}$$

Kde $k = 0 \ldots n$ a T je vzorkovacia perióda. Analogicky dostávame vzorec pre celkovú hodnotu akčnej veličiny:

$$u(T) = u_p(T) = K_p e(kT) + K_i T \sum_{i=0}^{k} e(iT) + K_d \frac{e(kT) - e[(k-1)T]}{T} \tag{2.10}$$

Vzorec v tomto tvare je následne možné jednoducho previesť na algoritmus použitého programovacieho jazyka.

Kapitola 3

Simulovanie a modelovanie

> *Systém je súbor elementárnych častí (prvkov systému), ktoré majú medzi sebou určité väzby (prepojenie prvkov).*[7]

Systémy je možné deliť do rôznych kategórií.[7]

- Podľa existencie:
  - Reálne – reálne existujúce systémy (školská jedáleň).
  - Nereálne – umelo vytvorené, alebo doposiaľ neexistujúce systémy (počítačová hra).
- Podľa zmien stavu:
  - Statické – nemenia svoj stav v čase (konečný automat – jeho stavy nezávisia od času).
  - Dynamické –menia svoj stav v čase (väčšina reálnych systémov okolo nás).

3.1 Model

Model je napodobenina systému iným systémom. Model systému (napr. počítačový program) musí napodobňovať všetky podstatné vlastnosti modelovaného systému. Modely delíme na[7]:

- Spojité – premenné modelu sa menia spojito v čase.
- Diskrétne – premenné modelu sa menia skokovo v presne definovaných časových okamihoch.
- Kombinované – model obsahuje spojité aj diskrétne prvky.

Simulačný model

Simulačný model vznikne prevedením nejakého reálneho systému do podoby vykonateľného modelu. To znamená, že je možné interpretovať tento model tak, aby vykonával požadované chovanie (napríklad počítačový program interpretovaný na počítači).

3.2 Simulácia

> *Simulácia je proces experimentovania s vhodnou reprezentáciou simulačného modelu*[7]

Úlohou simulácie je analýza chovania modelu na základe jeho počiatočných podmienok a podnetoch z okolia ovplyvňujúcich jeho stav.

Výhody simulácii

- Cena – Experimenty s reálnymi systémami môžu byť príliš nákladné. Napríklad simulácie crash-testov automobilov ušetria veľké množstvo vozidiel. Skutočné zrážky sa využívajú len na overenie týchto experimentov.
- Rýchlosť – Simulačné procesy (experimenty) sa dajú zrýchľovať a spomaľovať podľa potreby (v rámci možností simulačných zariadení). Napríklad model pohybu planét v našej slnečnej sústave, by nebol v reálnom čase použiteľný.
- Bezpečnosť – Niektoré experimenty sú v reálnom svete príliš nebezpečné, alebo dokonca z bezpečnostného hľadiska úplne vylúčené. Napríklad simulácia jadrových výbuchov, morových epidémií a podobne.
- Zložitosť – V simulácii je možné experimentovať s náročnými, zložitými a komplexnými systémami. Reálne experimenty by u niektorých takto zložitých systémov neboli uskutočniteľné, alebo príliš náročné.
- Jediná možnosť – Simulácia môže byť jediná možnosť, ako experimentovať s nejakým systémom. Napríklad zrážky planét, topenie ľadovcov a podobne.

Etapy simulácie

1. Vytvorenie abstraktného modelu – na základe primeranej abstrakcie (zanedbaní niektorých reálnych vlastností reálneho systému) sa vytvorí zjednodušený model skúmaného systému. Abstrakcia odstráni vlastnosti systému, ktoré nemajú podstatný vplyv na simuláciu, alebo pri ktorých je vytvorenie ich modelu príliš náročné alebo neuskutočniteľné.[7]

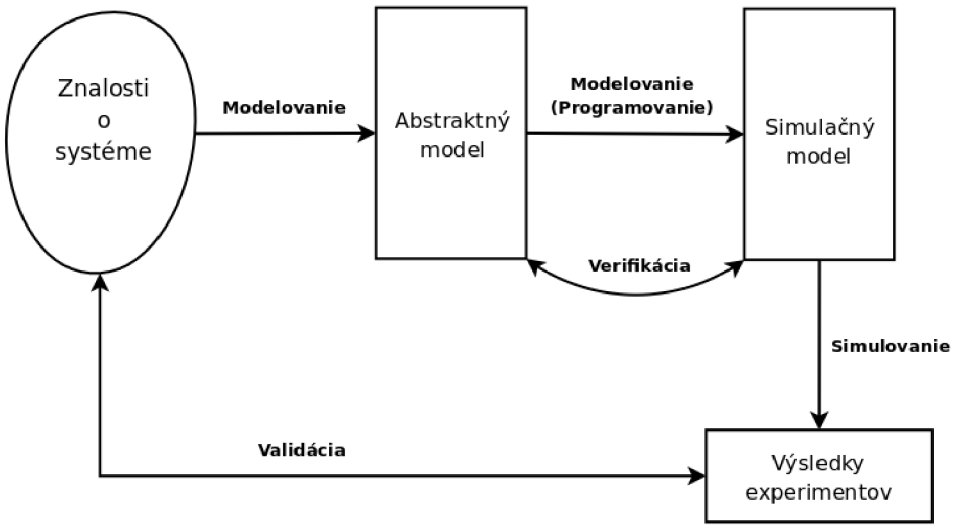
2. Vytvorenie simulačného modelu – na základe abstraktného modelu vytvoríme model simulačný. V súčasnosti najčastejšie vo forme počítačového programu, v minulosti aj formou fyzikálnych modelov.[7]

   Verifikácia – proces overovania totožnosti simulačného a abstraktného modelu. Snaha docieliť totožnosť z hľadiska štruktúry a chovania v pomere 1 : 1.

3. Simulácia – experimentovanie so simulačným modelom. Sledovanie stavu systému v rôznych fázach simulácie, zmena parametrov simulácie a ich vplyv na výsledok experimentu.[7]

   - Príprava experimentu – vytvorenie simulátoru, modelu a ich inicializácia
   - Prevedenie experimentu – spustenie simulácie, záznam jej priebežných výsledkov
   - Ukončenie experimentu – zaznamenanie výsledkov a koncového stavu simulácie, zrušenie modelu a simulátoru

4. Analýza výsledkov – porovnanie výsledného stavu simulácie so skutočným stavom reálneho systému pri rovnakých počiatočných podmienkach. V prípade rozdielnych výsledkov, je potreba upraviť abstraktný a aj simulačný model systému a uskutočniť nové experimenty. Cyklus sa opakuje pokým nedosiahneme výsledky odpovedajúce realite v požadovanej miere. Tento proces analýzy výsledkov sa nazýva **validácia**.[7]



Obrázok 3.1: Proces simulovania

3.3 Reprezentácia výsledkov

Tabuľka

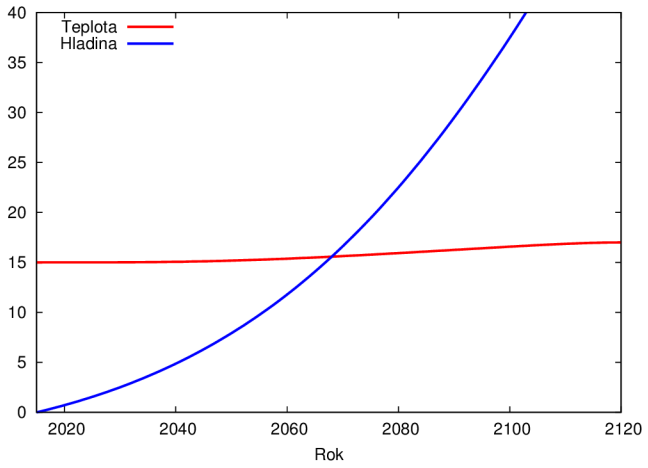
Tabuľka je najjednoduchšia možnosť zobrazenia výsledkov simulácie. Obsahuje hodnoty vybraných parametrov a ich postupný priebeh v čase simulácie. Pri väčšom množstve parametrov nie je na prvý pohľad jasný výsledok simulácie.

Rok	2015	2030	2045	2060	2075	2090	2105	2120
Priemerná teplota [°C]	15,1	15,3	15,6	15,9	16,3	16,7	17,2	17,8
Nárast hladiny oceánov [cm]	0	2	5	10	16	28	41	56

Tabuľka 3.1: Reprezentácia výsledkov simulácie tabuľkou (vymyslené hodnoty)

Graf

Väčšinou vychádza z tabuľkovej reprezentácie simulácie. Na základe hodnôt premenných v čase vytvára ich grafický priebeh. Podľa priebehu a vzájomných vzťahov medzi premennými je možné zvoliť z veľkého množstva rôznych typov grafov. Pre niektoré simulácie je možné použiť aj trojrozmerný graf.



Obrázok 3.2: Reprezentácia výsledkov simulácie grafom

Grafická reprezentácia

Pod pojmom grafická reprezentácia sa myslí vytvorenie reprezentácie simulačného modelu čo najvernejšie odpovedajúcemu realite. Zmeny premenných v čase majú priamy vplyv na tento model, pretvárajú ho na aktuálnu podobu.

Obrázok 3.3: Grafické zobrazenie simulácie havárie automobilu
Zdroj: *http://www.directindustry.com/prod/esi-group/crash-test-simulation-software-8972-131175.html*

Zvolená forma reprezentácie výsledkov závisí od potrieb konkrétnej simulácie, možností jej realizácie a taktiež od formy ich ďalšieho využitia.

Kapitola 4

Analýza a návrh

Simulácia modelu divadelného javiska, ktorý by obsahoval aj samotné elektrické pohony s ich charakteristikami, by bola príliš náročná. Musel by sa modelovať aj samotný priebeh elektrického napätia a prúdu a ich následný vplyv na model. Preto v rámci vytvorenia abstraktného modelu tieto prvky zanedbáme. Pokúsim sa vytvoriť simulačný model, ktorý bude čo najvernejšie odpovedať pohybu javiskových stolov a ťahov podľa 2.3. V každom výpočetnom kroku simulácie sa vypočíta aktuálna rýchlosť zariadenia podľa požiadaviek na pohyb. V reálnom modeli by bola táto rýchlosť vstupom pre reguláciu otáčok elektrického pohonu. Na základe aktuálnej rýchlosti sa dopočíta aj nová aktuálna pozícia zariadenia.

4.1 Programy simulácie

Simulácia ako celok bude pozostávať z dvoch samostatných častí – riadiaceho systému a samotnej simulácie (výpočtu pohybu javiska).

Riadiaci systém

Hlavnou funkciou riadiaceho systému bude prijímať požiadavky na pohyb z riadiaceho pultu a následne ich odosielať jednotlivým zariadeniam. Riadiaci systém bude mať prehľad o všetkých potrebných parametroch jednotlivých zariadení a bude ich pravidelne zasielať riadiacemu pultu ako aktualizácie. Taktiež bude podľa aktuálneho stavu jednotlivých zariadení generovať a odosielať vhodné reťazce reprezentujúce príkazy na zmenu modelu v programe LexoCad.

Simulácia

Simulácia pohybu jednotlivých zariadení bude jeden samostatný program. Výpočet aktuálnych stavov zariadení bude riadený jedným centrálnym časom. To znamená, že sa postupne vypočítajú nové hodnoty týchto parametrov pre všetky zariadenia, potom sa výpočet pozastaví na vopred stanovený čas. Po jeho uplynutí sa výpočetný cyklus znovu opakuje.

Je potrebné, aby výpočet parametrov jednotlivých zariadení nebol spolu väčší ako čas, po ktorý je cyklus pozastavený. To by viedlo k spomaleniu simulácie oproti reálnemu času. Simulačný program bude nielen vykonávať tento výpočet, ale bude prijímať príkazy zaslané riadiacim systémom pre všetky zariadenia.

4.2 Zobrazenie výsledkov simulácie

Ako najvhodnejší pre grafický výstup simulácie považujem program LexoCad od firmy Cadwork Informatik. Ide o jednoduchý CAD aplikáciu, v ktorej je možné veľmi rýchlo zo vstavaných komponent vytvoriť jednoduchý model javiska. Skúsenejší užívateľ by bol istotne schopný vytvoriť model verne napodobňujúci konkrétne javisko. Ja však vytvorím len jednoduchý model, na ktorom budú jasne zreteľné možnosti simulácie (bez okrasných efektov). Program LexoCad som zvolil hlavne preto, že ako jediný z voľne dostupných podobných aplikácii poskytuje jednoduché užívateľské rozhranie pre zmenu parametrov vytvoreného modelu (simulácia pohybu). Program má zabudovaný interprét programovacieho jazyka Python, ktorý umožňuje pomocou jednoduchých príkazov a funkcií meniť parametre daného modelu. Príkazy je možné zadávať do konzolového okna zabudovaného priamo v programe, alebo zasielať prostredníctvom TPC/IP rozhrania programu. Program prijíma TCP/IP packety na porte 4000 a následne sa ich pokúsi vykonať, ako keby boli zadané do konzolového okna. To znamená, že pri správne zostavenom reťazci zaslanom na toto rozhranie je taktiež možné meniť parametre modelu. Tento prístup použijem aj pri svojej práci.

4.3 Komunikačný protokol

Na zabezpečenie komunikácie medzi riadiacim systémom a zariadeniami, respektíve na prijímanie požiadaviek na pohyb od ovládacieho pultu, bol vytvorený jednotný komunikačný protokol. Obsahuje definície riadiacich správ na vytvorenie komunikácie, zadávanie povelov na pohyb, na získanie aktuálnych údajov o zariadeniach. Protokol využíva TCP/IP komunikáciu.

Ovlácací pult bude schopný pomocou protokolu zasielať požiadavky na pohyb jednotlivých zariadení, na vytvorenie a editovanie skupín zariadení a bude môcť získať aktuálne informácie od riadiaceho systému. Ten na základe správ od ovládacieho pultu bude riadiť pohyb zariadení zasielaním konkrétnych pohybových príkazov pre jednotlivé zariadenia. Zariadenia budú tieto požiadavky prijímať, na ich základe vykonávať určitý pohyb a pravidelne informovať riadiaci systém o ich aktuálnom stave.

V rámci bakalárskej práce implementujeme podporu základných správ protokolu. Bližšie vysvetlenie použitých správ bude uvedené v príslušných častiach implementácie. Celkový popis štruktúry komunikačného protokolu sa nachádza v Dodatku A.

Kapitola 5

Implementácia

Na vytvorenie programov bol použitý jazyk C++. Avšak programy nie sú písané čisto objektovým spôsobom. Využívajú taktiež jazyk C a niektoré jeho knižnice.

Ako vývojové prostredie bol použitý program Visual Studio 2008 od firmy Microsoft. Oba programy sú vytvorené ako konzolové aplikácie.

5.1 Riadiaci systém

Riadiaci systém musí byť schopný súčasne prijímať požiadavky na pohyb, odosielať požiadavky zariadeniam, prijímať aktualizácie od zariadení a následne ich odosielať ovládaciemu pultu. Na vytvorenie samostatných vlákien programu, v ktorých tieto funkcie prebiehajú nezávisle na ostatných, som využil funkciu `CreateThread()` z knižnice `Windows.h`. Po zavolaní táto funkcia vytvorí nové vlákno, v ktorom sa vykoná funkcia, ktorej názov bol zadaný ako jeden z parametrov funkcie `CreateThread()`. V rámci riadiaceho systému sú to funkcie `Feedback()`, `Komunikacia()`, `Pult()`, `Pohony()`.

Trieda zariadenie

Objekty tejto triedy slúžia na uloženie informácií o zariadeniach použitých v simulácii. Údaje slúžia na kontrolu limitných podmienok pri zadávaní pohybu (napr. maximálna možná rýchlosť alebo poloha) a na uloženie aktuálnych stavov zariadení kôli aktualizácii pre ovládací pult.

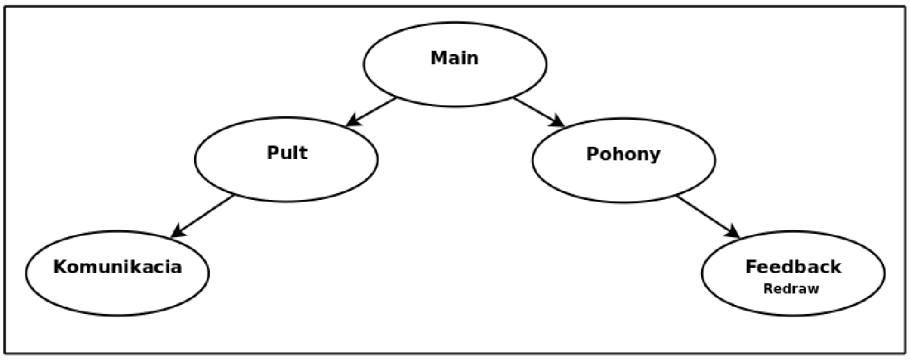
Trieda definuje metódy na nastavenie a získanie hodnôt premenných. Metóda `sent_feed()` vráti presne definovaný reťazec vytvorený z hodnôt premenných, potrebný na zaslanie aktualizácie pre ovládací pult. Metóda `sent_init()` reťazec potrebný na inicializáciu ovládacieho pultu pri zahájení komunikácie.

Trieda cas

Trieda definuje jedinú metódu – `wait()`. Slúži na pozastavenie programu na presne definovanú dobu. Doba zastavenia sa udáva v sekundách ako parameter metódy, napríklad

`wait(0.5)` pozastaví beh vlákna, kde bola zavolaná, na 0,5 sekundy. Na pozastavenie programu sa využívajú funkcie definované v knižnici `time.h`.

Trieda sa využíva hlavne pri chybových hláseniach programu, po ktorých nemá zmysel pokračovať vo vykonávaní programu a ten sa ukončí. Pozastavenie ukončenia programu po výpise chybového hlásenia umožní jeho prečítanie skôr, než sa konzolové okno uzavrie.



Obrázok 5.1: Diagram vetvenia programu na vlákna

Funkcia main

Funkcia vytvorí objekty pre uloženie informácií o zariadeniach (trieda `zariadenie`), vytvorí komunikačné kanály pre príjem a odosielanie správ v rámci simulácie a taktiež nadviaže spojenie s programom LexoCad. Na vytvorenie dostatočne počtu inštancií pre uloženie informácii o zariadeniach je potrebné, aby sa v zložke s programom nachádzal súbor `info.txt`, v ktorom je definovaný počet zariadení.

Po úspešnom načítaní dát zo súboru, vytvorení potrebných dátových štruktúr a nadviazaní komunikácie funkcia vytvorí vlákna `Pult()` a `Pohony()`. Funkcia potom už len čaká na prípadne zadaný reťazec stop, ktorý ukončí činnosť celého programu.

Funkcia Pult

Funkcia v nekonečnom cykle čaká na pakety zaslané od ovládacieho pultu. Ak obdrží takýto paket, vytvorí nové vlákno na jeho spracovanie, vlákno `Komunikacia()`, a sama ďalej čaká na ďalšie pakety. Reťazec prijatý v pakete je predaný vláknu ako jeho parameter.

Funkcia Pohony

Funkcia slúži v prvom rade na nadviazanie komunikácie s programom pre simuláciu zariadení. Funkcia odošle inicializačný paket na rozhranie, na ktorom očakáva komunikáciu so simuláciou. Pri obdržaní následného paketu obsahujúceho počiatočné informácie o zariadeniach, skontroluje, či počet zariadení odpovedá požadovanému počtu zariadení zo strany

simulácie. Ak údaje nesúhlasia, funkcia opäť odošle inicializačný paket. Pri korektných údajoch odošle potvrdzujúci paket, ktorý signalizuje simulačnému programu, že riadiaci systém je pripravený prijímať aktualizácie. Po odoslaní paketu funkcia vytvorí vlákno `Feedback()`.

Na získanie údajov o zariadeniach z inicializačného paketu slúži funkcia `set_init()` definovaná v hlavičkovom súbore `funkcie.h`.

Funkcia Komunikacia

Funkcia zo zadaného reťazca (paket od ovládacieho pultu) zistí, o aký typ správy ide a na základe toho zavolá príslušnú funkciu z hlavičkového súboru `funkcie.h`:

- sent_motion() – pri `MOTION` pakete. Funkcia rozparsuje (oddelí) príkazy na pohyb pre jednotlivé zariadenia a odošle ich postupne simulačnému programu. Pred samotným odoslaním sa ešte skontrolujú zadané parametre pohybu. Pri hodnotách väčších ako sú limity pre dané zariadenie, sa tieto hodnoty orežú na maximálnu povolenú veľkosť.

- set_desk() – pri `CHOOSE_ON_DESK` pakete. Funkcia podľa obsahu správy buď priradí zariadenie danému ovládaciemu pultu, alebo toto zariadenie uvoľní. Priradenie a zrušenie sa vykoná zmenou hodnoty premennej `desk_id`.

Pri obdržaní paketu `INIT_REQUEST` sa zvýši počítadlo pripojených ovládacích pultov. Následne sa vygeneruje správa potvrdzujúca inicializáciu. Správa obsahuje identifikačné číslo pultu (poradové číslo pripojenia) a následne zoznam potrebných parametrov jednotlivých zariadení. Ten sa vytvorí zavolaním metódy `sent_init()`.

Funkcia Feedback

Funkcia v nekonečnom cykle čaká na pakety od simulačného programu. Pri obdržaní paketu `NEW_FEEDBACK` sa zavolá funkcia `set_feedback()` z hlavičkového súboru `funkcie.h`. Tá rozparsuje prijatý reťazec na jednotlivé zariadenia a aktualizuje ich parametre (premenné). Po aktualizácii sa nové údaje zašlú riadiacemu pultu. Pomocou metódy `sent_feed()` sa vygeneruje požadovaný reťazec.

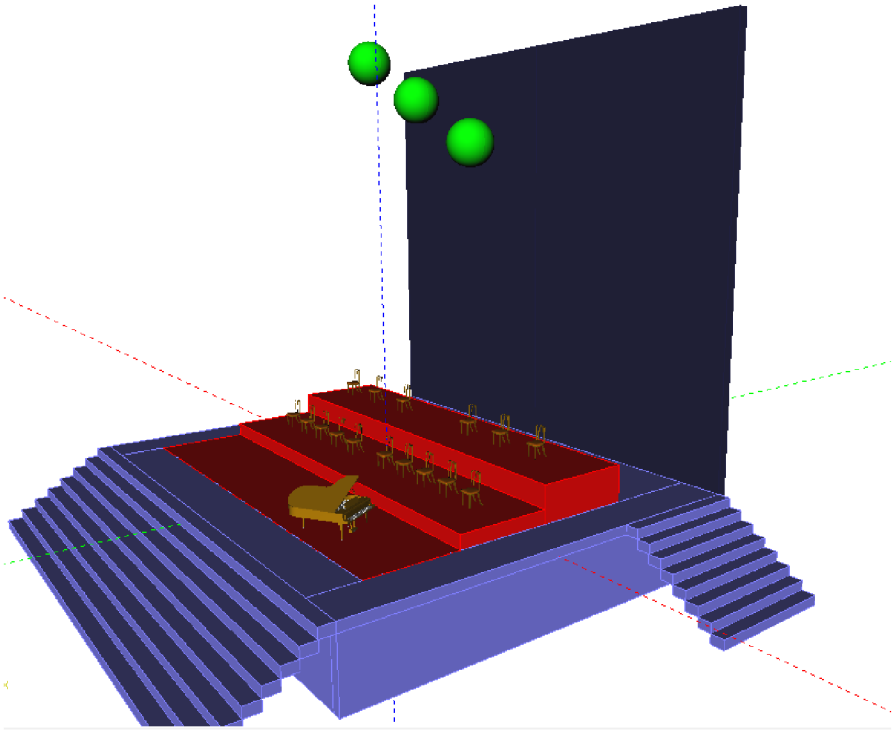
Na konci sa zavolá funkcia `Redraw()`, v ktorej je implementované generovanie vykresľovacích reťazcov pre program LexoCad. Podrobný popis vykresľovania aj funkcie `Redraw()` nájdete v nasledujúcej sekcii 5.2.

5.2 Vykresľovanie

V rámci grafického výstupu simulátoru som zostrojil jednoduchý model javiska v programe LexoCad. Pozostáva len z 6 pohyblivých komponent, z ktorých 3 predstavujú posuvné stoly javiska a 3 bodové ťahy. Model nie je prepracovaný do detailov, skôr v ňom ide o vystihnutie možnosti grafického znázornenia pohybu v reálnom čase. Model sa mi podarilo vytvoriť

len za použitia už vstavaných komponent programu, nebolo treba kresliť ani modelovať komplikovanejšie časti. Model javiska by sa dal rozdeliť na 2 základné časti5.2:

- Statické komponenty - predstavujú všetko, čo užívateľ chce, aby jeho model obsahoval, ale v skutočnom javisku to nemá žiadne dynamické vlastnosti. V modeli počet týchto komponentov predstavuje akúsi mierku detailnosti a realistickosti vytvoreného modelu. Záleží len na užívateľovi, ako detailne a realisticky chce mať svoj model prepracovaný. V mojom jednoduchom modeli som použil minimum týchto prvkov, pretože z funkčného hľadiska simuláciu nijako neovplyvňujú. Obrázok 5.2 – odtiene modrej, klavír, stoličky.
- Pohyblivé komponenty - predstavujú pohyblivé časti skutočného javiska, stoly, prepady a ťahy. Vo svojom modeli som vytvoril trojicu posuvných stolov a trojicu bodových ťahov, na ktorých demonštrujem možnosti pohybu simulácie. Pohyblivé komponenty sú taktiež vytvorené zo vstavaných komponent v programe. Obrázok 5.2 – červenou farbou (stoly), zelenou farbou (ťahy).



Obrázok 5.2: Jednoduchý model divadelného javiska

Prístup ku komponentom

Na to, aby bol pohyb jednotlivých komponent možný, musí byť každá jedna pohyblivá časť modelu jednoznačne identifikovateľná. A jej označenie musí byť rovnaké ako v modeli tak aj

```
import App, Base, Par
doc = App.Application.instance().getActiveDocument()
p=[0,0,0,0,0,0]
h=[0,0,0,0,0,0]
allElems = doc.getVisibleElements()
i=0
for i in range(6):
  for e in allElems:
    if e.userName.getValue() == "p"+str(i+1):
      p[i]=e.getGeometry()
      h[i]=e
      h[i]=App.cast2_Element(h[i])
      p[i]=App.cast2_Geometry(p[i])
doc.recompute()
```

Tabuľka 5.1: Inicializačný paket

v riadiacom systéme. Na označenie pohyblivých častí používam ich premenovanie v kolónke "name"z defaultného na označenie p1, p2, ..., pn. Podľa počtu zariadení.

Inicializácia

Po nadviazaní komunikácie s programom LexoCad, je potrebné poslať prvý, takzvaný inicializačný packet. Ten zabezpečí neskoršie jednoduché zmeny parametrov prekresľovaných objektov. Jeho štruktúra je nasledujúca:

Inicializačný reťazec je z časti statický a z časti sa generuje na základe počtu použitých zariadení. Preto je dôležité, aby počet zariadení v riadiacom systéme odpovedal počtu pohyblivých komponent v modeli.

Popis reťazca: Prvý riadok obsahuje importovanie dôležitých knižníc na prácu s modelom v prostredí LexoCad. Potom si uložíme ukazovateľ na aktuálne aktívny dokument, náš model. Vytvoríme zoznamy p,h, ktoré pri vytvorení naplníme nulovými hodnotami. Zoznam musí mať veľkosť zhodnú z počtom pohyblivých komponent. Napĺňame ich pri vytvorení preto, aby sme vytvorili dostatočne veľké zoznamy a nemuseli neskôr ich veľkosť zväčšovať vkladaním nových prvkov. Potom v dvoch cyklusoch prejdeme mená všetkých objektov v dokumente. Pri zhode mien s nami zadanými menami p1, p2,...,pn uložíme ukazovateľ na geometrické vlastnosti tohto objektu do príslušného prvku zoznamu p. Do zoznamu h uložíme ukazovateľ na objekt ako celok.

Je potrebné, aby zaslaný reťazec mal presnú formu odsadzovania, ako vidieť na 5.1. Je to kvôli vlastnostiam jazyka Python, ktorý na oddelenie príkazov používa znak nového riadku a na oddelenie jednotlivých blokov programu odsadzovanie pomocou tabulátorov.

Pri nedodržaní presného počtu tabulátorov na začiatku každého riadku, by mohol mať inicializačný reťazec v programe LexoCad nesprávnu funkciu, prípadne spôsobiť chybu pri jeho interpretácii.

Funkcia Redraw

Funkcia vytvorí reťazec pozostávajúci z presne definovaných riadkov, pomocou ktorých sa prekreslí model v programe LexoCad podľa aktuálneho stavu simulácie. Každý riadok reťazca odpovedá jednému zariadeniu simulácie. Štruktúra reťazca sa líši podľa toho, či ide o zariadenie dolnej, alebo hornej mechanizácie:

- Dolná mechanizácia – `ok=p[i].setProperty("height",x)`

  Pohyb dolnej mechanizácie predstavuje zmenu výšky pohyblivej komponenty. Pre zariadenie `i` bude premenná `x` nastavená na hodnotu jeho aktuálnej polohy. Po vykreslení sa zmení parameter `height`, reprezentujúci výšku komponenty v programe LexoCad, na zadanú hodnotu.

- Horná mechanizácia – `h[i].placement.translate(Base.Vec(0, 0,x),Base.CA.WCS)`

  Pohyb hornej mechanizácie predstavuje zmenu súradnice polohy komponenty v ose `z`. Pre zariadenie `i` bude premenná `x` nastavená na hodnotu rozdielu jeho aktuálnej polohy a jeho minulej polohy. Preto je potrebné si u zariadení hornej mechanizácie pamätať polohu z predchádzajúceho kroku vykreslenia. Po vykreslení sa hodnota `z` súradnice zadanej komponenty zväčší (zmenší) podľa zadanej hodnoty. Po vytvorení prekresľovacieho reťazca sa hodnota minulej polohy zariadenia prepíše na aktuálu hodnotu polohy. Pri nedoručení paketu programu LexoCad sa riadiaci algoritmus ukončí. Stratou prekresľovacích dát hornej mechanizácie vznikne rozdiel medzi skutočnou polohou zariadení a vykreslenou polohou.

Na poslednom riadku reťazca sa musí nachádzať `doc.recompute()`. Tento príkaz signalizuje programu LexoCad, že je potrebné znova vykresliť model. Bez tohto príkazu by sa parametre komponent menili, ale nijako by sa to neprejavilo na zobrazenom modele.

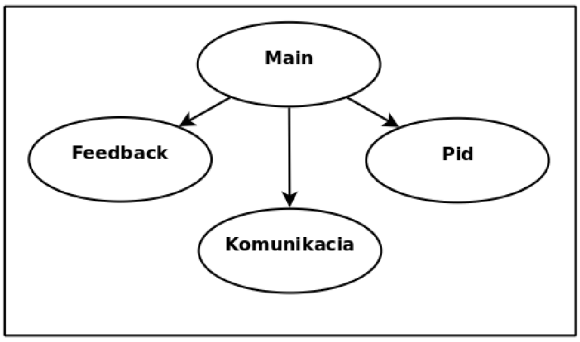
5.3 Simulácia zariadení

Simulačný program, podobne ako riadiaci systém, musí byť schopný súčasne prijímať požiadavky na pohyb jednotlivých zariadení a periodicky odosielať ich aktuálny stav. Navyše je potrebné, aby neustále na pozadí týchto komunikácii bežal samotný výpočet aktuálneho stavu zariadenia (jeho poloha, rýchlosť, zrýchlenie...). Na vytvorenie nezávislých vlákien programu som opäť použil funkciu `CreateThread()`. V rámci simulačného programu sú to funkcie `Feedback()`, `Komunikacia()` a `Pid()`.

Trieda pohon

Objekty tejto triedy slúžia na uloženie informácii o zariadeniach, potrebných na výpočet ich pohybu. Trieda definuje metódy na nastavenie a získanie hodnôt premenných. Metóda `set_init()` nastaví počiatočný stav zariadenia pri spustení simulácie, metóda `sent_init()` naopak odošle počiatočné informácie pre riadiaci systém pri vytvorení komunikácie. Metóda `fet_feedback()` vytvorí presne definovaný reťazec z hodnôt premenných, potrebný pri zasielaní aktualizácie pre riadiaci systém.

Pre výpočet aktuálneho stavu zariadenia slúži metóda `pohyb()`. Podrobnejší popis jej implementácie sa nachádza v sekcii 5.4.



Obrázok 5.3: Diagram vetvenia programu na vlákna

Funkcia main

Funkcia vytvorí objekty potrebné na uloženie informácii o zariadeniach v simulácii. Počet zariadení a ich inicializačné hodnoty sa nachádzajú v súbore `info.txt`. Preto je potrebné, aby sa tento súbor nachádzal v rovnakej zložke ako simulačný program.

Po úspešnej inicializácii vytvorí komunikačné kanály pre príjem a odosielanie paketov od a pre riadiaci systém. Následne vytvorí vlákna `Feedback()` a `Pid()`. Funkcia potom v nekonečnom cykle čaká na pakety od riadiaceho systému. Pri obdržaní paketu vytvorí vlákno `Komunikácia()`, v ktorom sa prijatý paket spracuje. Samotná funkcia pokračuje ďalej v čakaní na ďalšie prichádzajúce pakety.

Funkcia Komunikacia

Funkcia zo zadaného reťazca (paket od riadiaceho systému) zistí, o aký typ správy ide a na základe toho vykoná príslušnú akciu. Simulačný program môže obdržať len 2 typy paketov, buď paket obsahujúci požiadavku na pohyb, alebo inicializačný paket od riadiaceho systému pri zahajovaní komunikácie:

- `SET_MOTION` paket – zavolá sa funkcia `set_pohyb()` z hlavičkového súboru `funkcie.h`. Tá zo zadaného reťazca zistí, pre ktoré zariadenie bol zaslaný a aké sú požadova-

né parametre pohybu (konečná poloha, rýchlosť). Tieto hodnoty sa následne uložia do odpovedajúcich premenných zariadenia.

- `GET_INFO` paket – pomocou metódy `get_init()` sa vytvorí presne definovaný reťazec obsahujúci inicializačné údaje o všetkých zariadeniach simulácie. Ten sa následne odošle riadiacemu systému. Pri obdržaní potvrdzujúceho paketu funkcia povolí odosielanie aktualizácie z vlákna `Feedback`.

Funkcia Feedback

Po úspešnom nadviazaní komunikácie s riadiacim systémom funkcia periodicky odosiela aktuálne stavy všetkých zariadení. Na vytvorenie požadovaného reťazca sa využíva metóda `get_feedback()`. odosielanie prebieha v nekonečnom cykle. Pre dosiahnutie určitého intervalu zasielania sa využíva metóda `wait()`, viď trieda `cas` (sekcia 5.1).

5.4 Pohyb

Výpočet aktuálnych hodnôt premenných jednotlivých zariadení (rýchlosť, dráha, zrýchlenie) prebieha na základe vzorcov definovaných v sekcii 2.2. Hodnota derivácie zrýchlenia (ryv) bude v ďalšom texte označovaná ako J. Riadenie pohybu sa dá rozdeliť do dvoch bodov:

- Regulácia rýchlosti
- Regulácia polohy

Regulácia rýchlosti

Regulácia rýchlosti spočíva v dosiahnutí a udržaní maximálnej rýchlosti pohybu alebo zastavení na nulovú rýchlosť. Zrýchlenie a spomalenie sa môžu meniť len s konštantným prírastkom v danom časovom okamihu.

V každom výpočetnom kroku simulácie sa zisťuje, či by pri súčasných hodnotách rýchlosti a zrýchlenia zariadenie prekročilo maximálnu rýchlosť, ak by v tomto okamihu začalo znižovať zrýchlenie na hodnotu 0. Ak túto hodnotu neprekročí, pokračuje v zrýchľovaní, ak ju prekročí, nastaví sa konštantný prírastok zrýchlenia na zápornú hodnotu. Tento postup je platný pre zrýchľovanie, pre spomaľovanie je postup podobný, len s opačnými znamienkami. Keďže zrýchlenie sa mení na základe lineárnej funkcie, hodnotu rýchlosti, o ktorú sa zmení aktuálna rýchlosť pri znížení zrýchlenia na 0 vypočítame pomerne ľahko:

$$v = v_{act} + at$$

Kde v je rýchlosť dosiahnutá pri zrýchlení rovnom nule, v_{act} je aktuálna rýchlosť a at predstavuje veľkosť plochy opísanej zrýchlením pri jeho znížení na nulovú hodnotu. Podľa vzorca na výpočet obsahu trojuholníka, sa veľkosť tejto plochy rovná $\frac{1}{2}a_{act}T$. Kde $T = \frac{a_{act}}{J}$.

Regulácia polohy

Regulácia polohy v každom výpočetnom kroku simulácie vypočíta polohu, ktorú zariadenie prejde pokým nezastaví, ak by v tomto okamžiku začalo zastavovať (t.j. rýchlosť pohybu by bola nastavená na 0). Súčet tejto polohy s polohou aktuálnou sa porovná s maximálnou požadovanou polohou pohybu. Ak je tento súčet väčší, nastaví sa požadovaná rýchlosť na nulovú hodnotu a zariadenie postupne zastaví. Tento postup platí pre pohyb zariadenia v kladných hodnotách rýchlosti (zdola nahor), pre záporné hodnoty je postup podobný, len s obrátenými znamienkami.

V každom výpočetnom kroku simulácie sa zisťuje, akú dráhu by zariadenie urazilo do okamihu zastavenia, ak by pri súčasnej rýchlosti a zrýchlení začalo okamžite zastavovať. Ak je táto hodnota väčšia alebo rovná ako rozdiel konečnej a aktuálnej polohy, nastaví sa rýchlosť pohybu na 0. Zariadenie začne zastavovať.

Poloha do okamžiku zastavenia s sa počíta ako súčet dvoch zložiek. Polohy, ktorú zariadenie urazí do okamžiku nulového zrýchlenia za predpokladu, že ešte zrýchľuje a jeho rýchlosť sa mení. Ak už zariadenie dosiahlo maximálnu rýchlosť pohybu, je táto zložka nulová. Druhou zložkou je poloha, ktorú zariadenie urazí pri zastavení z rýchlosti dosiahnutej pri nulovom zrýchlení.

$$s = s_1 + s_2$$

$$s_1 = v_{act} + at_1$$

Kde a je integrál priebehu zrýchlenia z aktuálnej hodnoty do nulovej hodnoty a $t1$ doba, za ktorú sa táto zmena udeje.

$$s_2 = \frac{1}{2}at_2^2$$

Kde a je opäť integrál zrýchlenia, ktoré zariadenia prejde do okamžiku úplného zastavenia. Na základe vzorcu $v = \int adt$ je hodnota tohto integrálu rovná rýchlosti, ktorú má teleso pri nulovom zrýchlení. Čas t sa rovná dobe, za ktorú je zrýchlenie schopné s konštantným prírastkom zrýchlenia J dosiahnúť túto hodnotu.

Na základe aktuálnej hodnoty zrýchlenia sa vypočíta nová rýchlosť pohybu a pomocou nej aj prírastok dráhy v aktuálnom kroku výpočtu.

Funkcia Pid

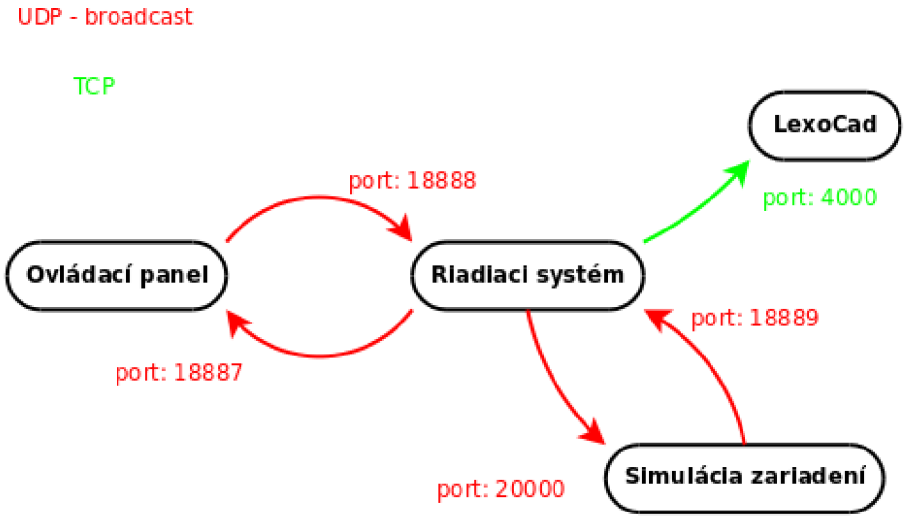
Funkcia opakovane vykonáva výpočet nových hodnôt parametrov jednotlivých zariadení. Funkcia postupne vykoná výpočet pre všetky zariadenia, potom vyčká zadanú dobu a znova opakuje výpočet nových hodnôt. Doba pozastavenia výpočtu predstavuje diskrétny časový skok, ktorého veľkosť je rovnaká času jedného kroku výpočtu aktuálneho stavu zariadenia. Výpočet je implementovaný v metóde `pohyb()` triedy `pohon`.

5.5 Komunikácia

Komunikácia v rámci simulácie aj komunikácia s riadiacim pultom prebieha na TCP/IP protokole. Správy reprezentujúce pokyny od riadiaceho pultu, pokyny na pohyb zariadení, aktualizácie a vykresľovacie reťazce sú dátovou časťou paketov tohto protokolu.

Pakety medzi ovládacím pultom, riadiacim systémom a simuláciou pohybu využívajú na transportnej vrstve protokol UDP a broadcastový spôsob zasielania. Pri tomto type komunikácie nie je potrebné pamätať si IP adresy jednotlivých pultov alebo riadiacich systémov. Pakety sú zaslané všetkým zariadeniam, očakávajúcich komunikáciu na danom porte. Až na základe obsahu správy je neskôr možné usúdiť, či zaslané údaje patria príjemcovi. Pre zamedzenie zbytočného prijímania paketov nepatriacich jednotlivým programom, boli zvolené rôzne komunikačné porty pre rôzne komunikácie medzi zariadeniami (obrázok 5.4). Keďže komunikácia v rámci broadcastu prebieha len v rámci rovnakej siete, je potrebné, aby ovládací pult, riadiaci systém aj simulátor pohybu patrili do rovnakej TCP/IP siete.

Pakety pre vykresľovanie v programe LexoCad využívajú TCP transportnú vrstvu, keďže program bo vytvorený s TCP komunikačným rozhraním. Pre správnu funkčnosť vykresľovania je potrebné, pri spustení programu reprezentujúceho riadiaci systém, zadať IP adresu zariadenia, na ktorom prebieha vykresľovanie v programe LexoCad. Pri použití rovnakého počítača na beh riadiaceho systému aj vykresľovania stačí zadať hodnotu `localhost`, pri použití iného počítača je potrebné zadať jeho IP adresu.



Obrázok 5.4: Komunikácia medzi komponentami simulácie

Kapitola 6

Testovanie

Testovanie správnosti simulácie (výpočet pohybu zariadení, komunikácia, správnosť vykresľovania v programe LexoCad) prebiehalo neustále počas rôznych etáp implementácie. Boli naň použité jednoduché metódy (kontrolné výpisy, jednoduché vlastné programy), ale aj profesionálne programy (Wireshark). Ako formu finálneho testovania, ktoré by jednoducho demonštrovalo funkčnosť programu, som zvolil vytvorenie 4 jednoduchých programov, ktoré simulujú komunikáciu od ovládacieho pultu. Každý z nich nadviaže komunikáciu s riadiacim systémom, následne odošle požiadavok na pohyb zariadení a ukončí sa.

Výsledky jednotlivých testov sú uvedené v tabuľkách, ktoré znázorňujú požadovanú koncovú polohu zaslanú riadiacim pultom, skutočnú koncovú polohu simulácie a polohu vykreslenú v programe LexoCad. Zariadenia označené písmenom D reprezentujú dolnú mechanizáciu, písmenom H hornú. Obrázky konečných stavov zariadení v programe Lexocad (obrázok 6.1) a zhodnotenie testov sa nachádzajú ďalej v tejto kapitole.

Test 1

Test bol zameraný len na pohyb dolnej mechanizácie. Zariadeniam bola zaslaná správa na pohyb rýchlosťou 100 mms^{-1}. Zariadenia sa pohybujú spoločne v jednej rovine a postupne zastavujú na definovaných polohách.

Zariadenie	D1	D2	D3	H1	H2	H3
Požadovaná poloha $[mm]$	1000	2000	3000	–	–	–
Dosiahnutá poloha $[mm]$	999	1999	3000	–	–	–
Vykreslená poloha $[mm]$	999	1999	3000	–	–	–

Tabuľka 6.1: Výsledky testu č. 1

Test 2

Test bol zameraný len na pohyb hornej mechanizácie, keďže pri vykresľovaní je použitý úplne iný princíp ako pri mechanizácii dolnej. Zariadeniam bola zaslaná správa na pohyb rýchlosťou $400\,mms^{-1}$, $350\,mms^{-1}$, $300\,mms^{-1}$ v zápornom smere (pohyb zhora nadol). Poloha zariadení leží počas pohybu v jednej naklonenej rovine.

Zariadenie	D1	D2	D3	H1	H2	H3
Požadovaná poloha $[mm]$	–	–	–	5000	5500	6000
Dosiahnutá poloha $[mm]$	–	–	–	4999	5500	6001
Vykreslená poloha $[mm]$	–	–	–	4999	5500	6001

Tabuľka 6.2: Výsledky testu č. 2

Test 3

Test kombinuje pohyb dolnej a hornej mechanizácie. Zariadenia dolnej mechanizácie vykonávajú rovnaký pohyb ako v prvom teste. Rýchlosť pohybu hornej mechanizácie odpovedá druhému testu s tým rozdielom, že prvé a tretie zariadenie majú rovnakú koncovú polohu. Tretie zariadenie hornej mechanizácie v tomto prípade dobieha ako posledné na požadovanú polohu, keď už všetky ostatné zariadenia zastavili.

Zariadenie	D1	D2	D3	H1	H2	H3
Požadovaná poloha $[mm]$	1000	1500	2000	5000	5500	5000
Dosiahnutá poloha $[mm]$	999	1499	1999	4999	5500	5000
Vykreslená poloha $[mm]$	999	1499	1999	4999	5500	5000

Tabuľka 6.3: Výsledky testu č. 3

Test 4

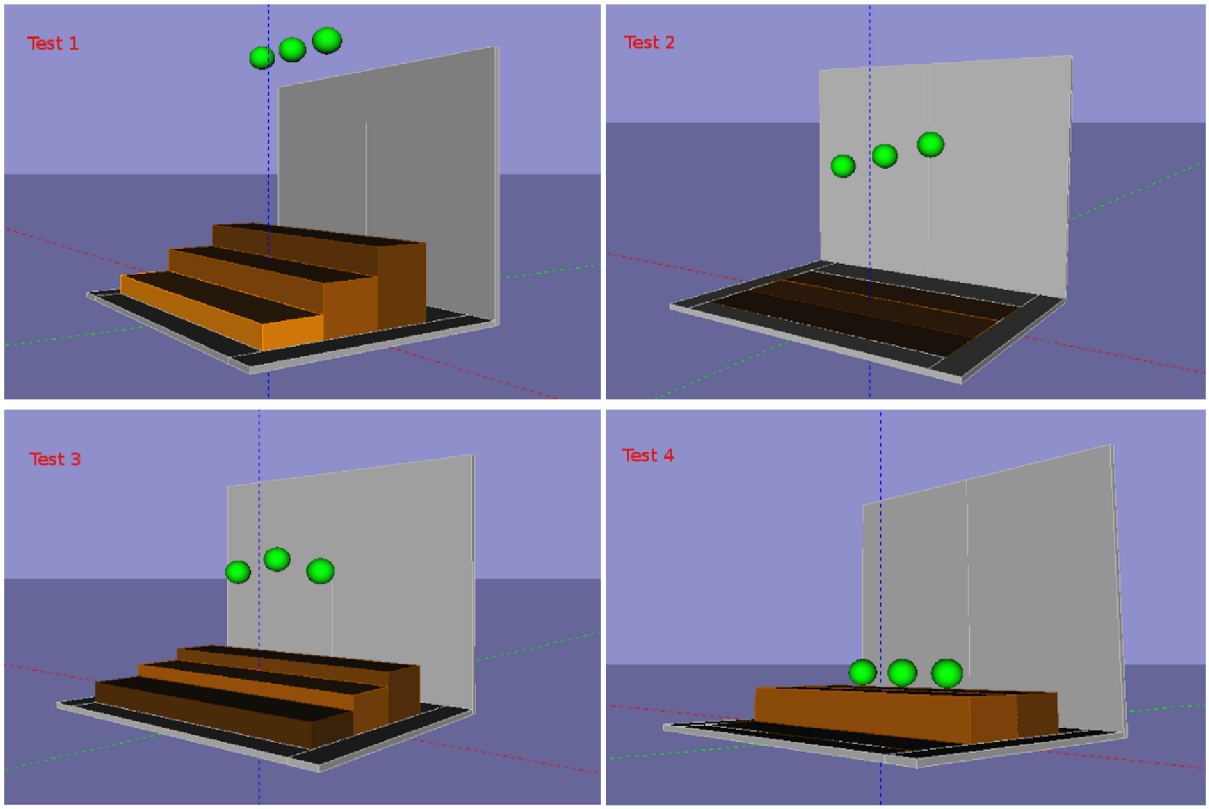
Test opäť kombinuje pohyb dolnej a hornej mechanizácie. Prvé zariadenie dolnej mechanizácie ostáva nehybné, zvyšné sa pohybujú v jednej rovine rovnakou rýchlosťou. Zariadenia hornej mechanizácie sú spustené $100\,mms^{-1}$ nad zariadenia dolnej mechanizácie. Rýchlosti sú nastavené tak, aby zariadenia zastavili približne v rovnaký okamžik.

Zariadenie	D1	D2	D3	H1	H2	H3
Požadovaná poloha $[mm]$	–	1500	1500	1600	1600	1600
Dosiahnutá poloha $[mm]$	–	1499	1499	1599	1599	1599
Vykreslená poloha $[mm]$	–	1499	1499	1599	1599	1599

Tabuľka 6.4: Výsledky testu č. 4

Zhodnotenie výsledkov

Pohyb zariadení pri testoch bol primerane plynulý, komunikácia prebiehala v poriadku a taktiež vykreslenie odpovedá vypočítaným hodnotám simulácie. V niektorých prípadoch sú však tieto hodnoty nepatrne rozdielne od požadovaných hodnôt. Spresnenie výpočtu by mohlo priniesť zmenšenie kroku výpočtu a tým aj zmenšenie hodnoty derivácie zrýchlenia v týchto okamžikoch (zmenšenie konštantnej hodnoty trhu).



Obrázok 6.1: Testy – cieľové polohy

Kapitola 7

Záver

Cieľom práce bolo navrhnúť riadiaci systém pohyblivých častí divadelného javiska. Pod týmto pojmom sa myslia jednotlivé časti hracej plochy a predmety zavesené nad hracou plochou, schopné pohybu vo vertikálnej ose. Ako výstup tohto systému bol vytvorený jednoduchý grafický simulátor, ktorý reprezentuje tento pohyb.

Po oboznámení sa s potrebami riadenia javiskovej scény a možnosťami na jej realizáciu, bol navrhnutý riadiaci systém ich pohybu. Ten na základe požadovanej pohybovej rovnice počíta aktuálne hodnoty rýchlosti, zrýchlenia a polohy zariadení. Tieto hodnoty by v reálnom systéme boli vstupom pre regulačný algoritmus ovládajúci ich pohyb. V rámci práce boli tieto hodnoty použité na simuláciu pohybu grafického modelu vytvoreného v programe LexoCad. Riadiaci systém pozostáva z dvoch samostatných programov, riadiaceho a simulačného. Riadiaci program prijíma požiadavky na pohyb zariadení a stará sa o vykreslenie aktuálnych polôh v modeli. Simulačný program na základe príkazov na pohyb od riadiaceho systému počíta aktuálne stavy zariadení a spätne ho informuje o ich stave. Komunikácia medzi programami prebieha pomocou vytvoreného protokolu prenášaného TPC/IP paketmi.

Simulácia dosahuje prijateľné výsledky presnosti pohybu. Pre uplatnenie riadiaceho algoritmu v praxi bude potrebné zjemniť jeho výpočet nastavením menšieho výpočetného kroku a pokúsiť sa ho implementovať ako samostatný mikrokontrolér ovládajúci jedno zariadenie. Pre použitie simulácie na vytváranie nových divadelných inscenácii nie je potreba, aby sa skladal z dvoch častí. Zlúčením riadiaceho systému a simulácie do výsledného komerčného produktu sa odstráni medziprocesová komunikácia bez straty presnosti výpočtu. Avšak cieľom práce bolo aj otestovať komunikáciu vytvoreným protokolom,ktorého kompletnú implementáciu bude potrebné taktiež dopracovať.

Pre väčšiu realistickosť grafického modelu javiska bude potrebné vymodelovať jeho detailnejšiu reprezentáciu, prípadne pri jeho modelovaní spolupracovať s odborníkom skúseným v tejto oblasti a pokúsiť sa vytvoriť model skutočne existujúceho javiska. Taktiež je možná spolupráca s vývojovým tímom programu LexoCad,s ktorým sa podarilo nadviazať vzájomnú komunikáciu (vývoj programu stále prebieha).

Literatúra

[1] Balátě, J.: *Vybrané statě z automatického řízení*. Univerzita Tomáše Bati, Fakulta technologická, 2003, ISBN 80-7318-120-7.

[2] Bezděk, P.: *Jevištní technologické zařízení – Divadla*. Ministerstvo kultury ČSR, 1987.

[3] Halliday, D.; Resnick, R.; Walker, J.: *Fyzika: Vysokoškolská učebnice obecné fyziky*. VUTIUM, 2000, ISBN 80-214-1869-9.

[4] Jančík, J.: Moderní řídicí systémy jevištní techniky. *Automa*, ročník 2009, č. 3: s. 56–60.

[5] Javůrek, J.: *Regulace moderních elektrických pohonů*. Grada, 2003, ISBN 80-247-0507-9.

[6] Madarász, L.; Bučko, M.; Főző, L.: *Základy automatického riadenia*. elfa,s.r.o., 2007, ISBN 80-8086-042-4.

[7] Peringer, P.: *Modelování a simulace: Studijní opora*. 2008.

[8] Schmid, D.: *Řízení a regulace pro strojírenství a mechatroniku*. Europa-Sabotáles, 2005, ISBN 80-86706-10-9.

[9] Valter, J.: PID regulátor. [online], 2009-11-16 [cit. 2010-04-24]. URL http://valter.byl.cz/pid.html

Dodatok A

Komunikačný protokol

Položky označené červenou farbou zatiaľ nemajú implementovanú žiadnu funkciu, s ich využitím sa počíta ďalšom pokračovaní práce.

smer: Riadiaci systém -- Simulátor
smer: Simulátor - Riadiaci systém

GET_INFO	Požiadavka na inicializáciu zariadení	

Z_INIT	Inicializačné informácie o zariadeniach	
n	integer: n	počet zariadení
id	Integer: 1 – n	ID zariadenia
status	1 – stojí, 2 – v pohybe	stav daného zariadenia (stojí, v pohybe)
type	1 – H, 2 – D, 3 – T	typ zariadenia (dolná / horná mech., točňa)
pos_max	Integer: [mm]	maximálna možná poloha zariadenia
speed_max	Integer: [mm]	maximálna možná rýchlosť zariadenia

Z_INIT;n;id1,status1,type1,pos_max1,speed_max1;id2,...;...idn...;

OK_GET_INFO	Uspešná inicializácia

SET_MOTION	Príkaz na pohyb zariadenia	
id	Integer: 1 – n	ID zariadenia
pos1	Integer: [mm]	prvá požadovaná cieľová poloha
pos2		druhá požadovaná cieľová poloha
speed1	Integer: [mm]	rýchlosť pohybu do polohy 1
speed2		rýchlosť pohybu do polohy 2
mode		režim chovania zariadenia

SET_MOTION;id,pos1,pos2,speed1,speed2,mode

NEW_FEEDBACK	Aktualizácia stavu zariadenia	
id	Integer: 1 – n	ID zariadenia
status	1 – stojí, 2 – v pohybe	stav daného zariadenia (stojí, v pohybe)
pos_actual	Integer: [mm]	aktuálna poloha
speed_actual	Integer: [mm]	aktuálna rýclosť
error		chyba

NEW_FEEDBACK;id,status,pos_actual,speed_actual,error;

smer: Riadiaci systém -- Ovládací pult

smer: Ovládací pult -- Riadiaci systém

INIT_REQUEST Vyžiadanie informácií o zariadeniach

INIT_OK Odpoveď na inicializačnú požiadavku

desk_num	Integer: 1 – n	ID pultu priradené od RS
id	Integer: 1 – n	ID zariadenia
status	1 – stojí, 2 – v pohybe	stav daného zariadenia (stojí, v pohybe)
type	1 – H, 2 – D, 3 – T	typ zariadenia (dolná / horná mech., točňa)
pos_max	Integer: [mm]	maximálna možná poloha zariadenia
pos_min	Integer: [mm]	minimálna možná poloha zariadenia
speed_max	Integer: [mm]	maximálna rýchlosť pohybu
group_id		priradenie zariadenia k skupine
desk_id		priradenie zariadenia ovládaciemu pultu

INIT_OK,desk_num;id1,status,type,pos_max,pos_min,speed_max,group_id,desk_id;id2,...;...idn...;

FEEDBACK Aktuálny stav zariadení

n	integer: n	počet zariadení
id	Integer: 1 – n	ID zariadenia
status	1 – stojí, 2 – v pohybe	stav daného zariadenia (stojí, v pohybe)
speed	Integer: [mm]	aktuálna rýchlosť
position	Integer: [mm]	aktuálna pozícia
group_id		priradenie zariadenia k skupine
desk_id		priradenie zariadenia ovládaciemu pultu
time		časová informácia
error		chyba

FEEDBACK,n;id1,status,speed,position,group_id,desk_id,time,error;id2,...;...idn...;

SET_GROUP Nastaví skupiny viacerých zariadení, ktoré sa chovajú podľa stanovenej funkcie

group_id	ID skupiny
id	ID zariadenia
function	bezpečnostná fynkcia skupiny

SET_GROUP;group_id,id,function;

DELETE_GROUP Zruší skupinu zariadení

group_id	ID skupiny

SET_GROUP;group_id;

MOTION Príkaz na pohyb zariadení

id	Integer: 1 – n	ID zariadenia
pos0	Integer: [mm]	aktuálna poloha zariadenia
pos1	Integer: [mm]	prvá požadovaná cieľová poloha
speed1	Integer: [mm]	rýchlosť pohybu do polohy 1
pos2		druhá požadovaná cieľová poloha
speed2		rýchlosť pohybu do polohy 2
time		časová informácia
mode		režim chovania zariadenia

MOTION;id,pos0,pos1,speed1,pos2,speed2,time,mode;

CHOOSE_ON_DESK Vybranie zariadenia na ovládaciom pulte

id	Integer: 1 – n	ID zariadenia
desk_id	Integer: 1 – n	ID ovládacieho pultu
operation	1 – výber, 2 – zrušenie	zariadenie vybraté/zrušené

CHOOSE_ON_DESK;id,desk_id,operation;