STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 1: Matematika a statistika

Řídící software šestinohého robota

Milan Malina, Ondřej Brichta
Plzeňský kraj

Plzeň 2017

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 1: Matematika a statistika

Řídící software šestinohého robota

Software for six-leg robot controlling

Autoři: Milan Malina, Ondřej Brichta
Škola: Gymnázium, Mikulášské náměstí, Mikulášské náměstí 23, 326 00, Plzeň
Kraj: Plzeňský kraj
Konzultant: Ing. Roman Čečil

Plzeň 2017

Prohlášení

Prohlašujeme, že jsme svou práci SOČ vypracovali samostatně a použili jsme pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašujeme, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemáme závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Plzni dne 27. 2. 2017 ……

Poděkování

Tato práce by nikdy nevznikla nebýt dlouholeté spolupráce a vedení Ing. Romana Čečila, který nejen že zařizoval finanční podporu, ale také byl iniciátorem projektu, autorem konstrukce hexapoda a tvůrcem řídící elektroniky. Na stavbě funkčního modelu se také podílel Martin Klíma. Také jsme vděční Haně Brichtové a Martinu Stránskému za gramatickou a stylistickou revizi tohoto textu. Všem zmíněným tímto děkujeme.

Anotace

Naším cílem bylo vytvořit software pro řízení a vizualizaci šestinohého robota, naše práce je tomuto vývoji věnována. Prvý oddíl této práce popisuje herní engine Unity 3D, ve kterém je simulace vytvořena a všeobecně seznamuje se simulací. Ve druhém oddílu se zaměříme na matematické prostředky použité v této simulaci. Závěrem druhého oddílu seznámíme čtenáře s připravovanými vylepšeními simulace. Uživatelské rozhraní je koncipováno nejen pro ovládání simulace, ale také pro použití na reálném modelu.

Klíčová slova

robotika; hexapod; řídící software; souřadnicová soustava; matice přechodu; křivka; Bézierovy křivky

Annotation

Our goal was to develop six-leg robot controlling and visualization software, and this paper is dedicated to this development. The first part of this paper describes game engine Unity 3D, which has been used to build this simulation and to general description of the simulation itself. The second part is dedicated to mathematical apparatus used in this simulation. At the end of the second part, we will introduce planned features and enhancements of this simulation. User interface is meant to be not only used for controlling the simulation but also the real device.

Keywords

Robotics; hexapod; control software; coordinate system; transformation matrix; curve; Bézier curves

OBSAH

1 ÚVOD

Software slouží pro řízení, vizualizaci a simulaci šestinohého robota (hexapod). Pro tvorbu software jsme použili herní engine Unity 3D. Herní engine Unity 3D jsme vybrali hlavně proto, že poskytuje otevřené možnosti pro vývoj libovolného software. Jako programovací jazyk jsme zvolili C#. Tento jazyk je Unity 3D podporován více, než klasické C nebo C++.

Uživateli je umožněno vybrat z různých forem automatických pohybů (naklánění v osách x,

Obr. 1: Splash screen – úvodní obrazovka

y a z, chůze různými směry a otáčení po a proti směru hodinových ručiček). Dále je možno manuálně nastavovat natočení servomotorů. Je zde sekvenční editor umožňující uživateli sestavit si vlastní sled pohybů. (u tohoto editoru jsme se inspirovali vývojovým prostředím pro tvorbu programů u stavebnice Lego NXT).

2 UNITY 3D

2.1 Úvod

Jak již bylo řečeno, simulace běží v herním engine Unity 3D. Unity 3D hraje dominantní roli v herním průmyslu. Jen za 3. čtvrtletí roku 2016 bylo staženo přes 5 miliard her a aplikací vytvořených v Unity, jenž běží na více jak 2.4 miliardách mobilních zařízení. Tato fakta z Unity 3D dělají jednu z nejpopulárnějších herních platforem. [1]

2.2 Multiplatformní podpora

Počet platforem, jimiž je tento engine podporován je celkem 27. Zde je jejich výčet [2]:

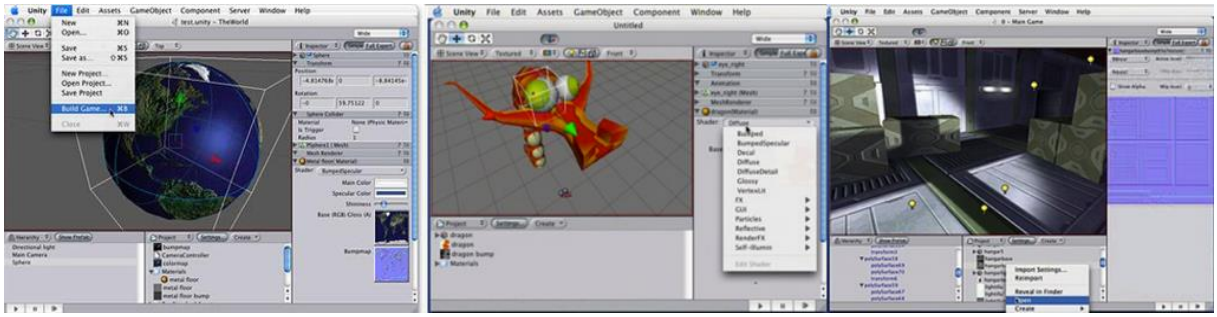
1. iOS
2. Android
3. Windows Phone
4. Tizen
5. Windows
6. Windows Store Apps
7. Mac OS
8. Linux/Steam OS
9. WebGL
10. PlayStation 4
11. PlayStation Vita
12. Xbox One
13. Wii U
14. Nintendo 3DS
15. Oculus Rift
16. Google Cardboard
17. Steam VR
18. Playstation VR
19. Gear VR
20. Microsoft Hololens
21. Daydream
22. Android TV
23. Samsung SMART TV
24. tvOS
25. Nintendo Switch
26. Fire OS
27. Facebook Gameroom

Windows, Mac OS, iOS a Android jsou ze všech dostupných platforem pravděpodobně nejdůležitější, a nejvhodnější pro námi vytvořenou simulaci. V budoucnosti plánujeme portovat celou aplikaci do zmíněných platforem. V současné době disponujeme plně funkční verzí pro MS Windows a začali jsme s přípravou portu pro iOS. Vzhledem k tomu, že nevlastníme vhodné mobilní zařízení s operačním systémem Android či macOS, je jedinou možností virtualizace.

2.3 Produkty

2.3.1 Historie

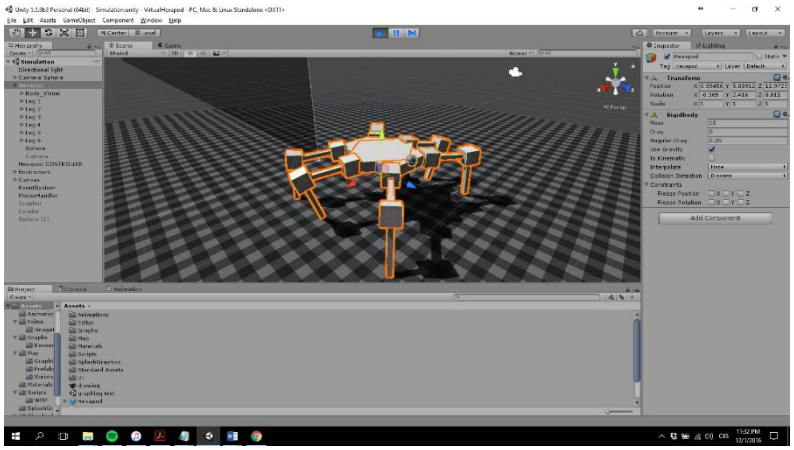
Unity 1.0 byl poprvé uveden roku 2005 na vývojářské konferenci Apple (WWDC 2005). Vydán byl 8. června 2005 a dostupný byl pouze pro tehdejší Apple Mac OSX. Cenově levnější balíček určený pro nezávislé vývojáře dostal název „Indie edition“. K dispozici byl také dražší balíček určený pro profesionální vývojáře a korporace s názvem „Pro edition“ [3].



Obr. 2: Unity verze 1.0 [4]

2.3.2 AKTUÁLNÍ VERZE

Aktuální verze Unity (v době psaní této práce) je 5.5.0 (vydána 30.11.2016) [5]. Pro vývoj software byla použita beta verze 5.5.0b3, aby mohl být použit tzv. „Splash screen editor“, tedy editor úvodní obrazovky, ještě před jeho oficiálním uvedením.



Obr. 3: Unity verze 5.5.0b3

V současně době jsou k dispozici 4 edice:

Personal edition – tato edice je určena pro nezávislé vývojáře a je zdarma. Ovšem i přes to jsou v ní k dispozici velmi účinné a efektivní nástroje pro vývoj.

Plus edition – v této edici je navíc oproti předchozí například rozšířené nastavení úvodní obrazovky, či možnost nastavit tzv. „Pro Editor UI Skin“ a další výhody. Tato edice vyjde na 35 USD měsíčně.

Pro edition – slouží výhradně pro profesionální vývojáře s neomezeným profitem. Tato edice vyjde na 125 USD měsíčně.

Enterprise edition – tuto edici využívají hlavně velké korporace. Pro aktivaci je nutno kontaktovat Unity team. [6]

3 TÉMATICKÉ ROZDĚLENÍ SOFTWARE

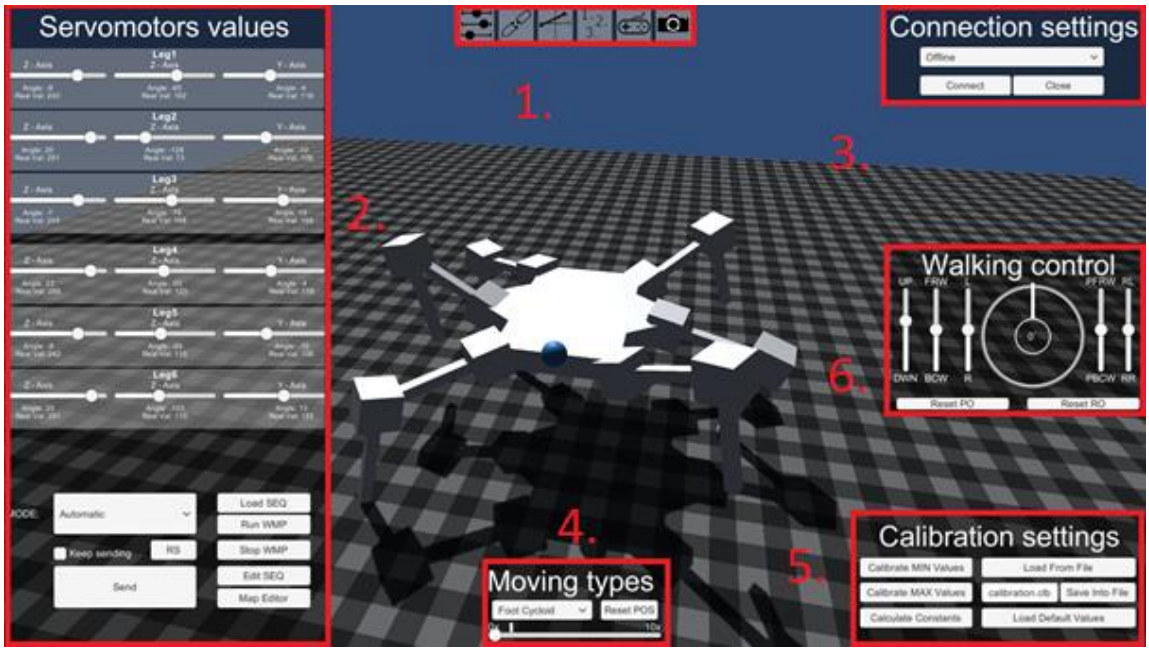
3.1 Úvod

Program je tematicky rozdělen na celkem 3 části:

- Popis uživatelského rozhraní a vizualizace
- Editor sekvence kroků
- Mapový editor

3.2 Simulace

Hlavní součástí software je samotná vizualizace modelu hexapoda (viz Obr. 4). Byla to první vytvořená část programu.



Obr. 4: Rozložení ovládacích panelů v simulaci

Na horním okraji zorného pole se nachází menu (1). Obsahuje tlačítka pro zobrazení těchto pěti ovládacích panelů:

Servomotors values (2) – hodnoty úhlů aktuálního natočení servomotorů – zde se posuvníky nastavují a zobrazují hodnosty odesílané do hexapoda.

Connection settings (3) – nastavení připojení – zde se vybírá RS232 port a následně se k němu dá kliknutím na tlačítko *Connect* připojit plošný spoj s mikrokontrolérem CC1111.

Moving types (4) – typy pohybu – zde si může uživatel zvolit pohybový mód a posuvníkem upravit rychlost toku času.

Calibration settings (5) – nastavení kalibrace – tento panel slouží k vytvoření a načtení kalibračního souboru.

Walking control (6) – ovládání chůze – zde je umožněno měnit pozici těla a směr chůze (do stran).

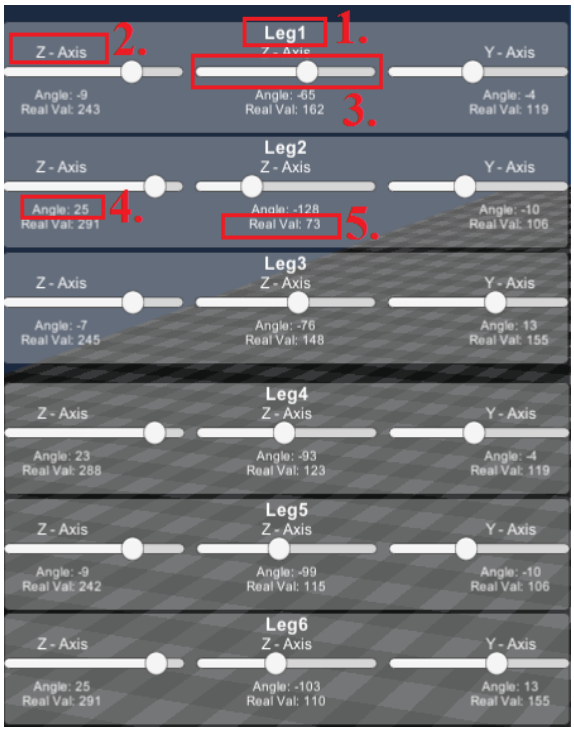
Kamera – Šesté tlačítko v menu aktivuje virtuální kameru umístěnou v přední části modelu hexapoda (modrá koule). Reálný model touto kamerou však nedisponuje.

3.2.1 Servomotor Values

Panel je rozdělen na dva funkční celky.

Část s posuvníky („slidery“) a část s tlačítky. Část s posuvníky (viz Obr. 5) je dále rozdělena na šest bloků (pro každou končetinu jeden). V hlavičce (1) každého bloku je pořadové číslo končetiny. Jednotlivé bloky obsahují vždy tři posuvníky (3) reprezentující jednotlivé servomotory

v dané končetině. Pokud byl zvolen manuální mód, umožňují posuvníky měnit natočení servomotorů.



Obr. 5: Rozložení ovládacích prvků v první části panelu Servomotor values

Nad každým servomotorem je jeho lokace a funkce v končetině (2). Pod posuvníky se nacházejí ukazatele úhlu natočení servomotoru ve stupních (4) (za předpokladu, že je načten správný kalibrační soubor). Pod ukazateli úhlů natočení servomotorů jsou zobrazeny (5) hodnoty pro řídící jednotku hexapoda (vypočteny z lineární regrese pro kalibraci).

Nyní přejdeme k přesnému výkladu funkce tlačítek pod částí s posuvníky (viz Obr. 6). Zde nalezneme devět ovládacích prvků. Prvním je nabídka („dropbox“) umožňující výběr mezi těmito třemi možnostmi zadávání pohybů:

Automatic – tato volba umožňuje spustit předem naprogramované pohybové módy z nabídky pohybu (viz (4) z Obr. 4).

Manual – volbou této varianty se aktivují posuvníky a uživateli je umožněno manuálně nastavovat jednotlivé úhly natočení servomotorů.

Program – při aktivaci této možnosti se dá realizovat sekvence pohybů sestavená v sekvenčním editoru, nebo pohyb po trajektorii vytvořené v mapovém editoru. Podrobnosti budou rozebrány níže.

Pod nabídkou zadávání pohybů se nachází zaškrtávací pole (checkbox) (2). Pokud je pole zaškrtnuto, hodnoty se automaticky odesílají do řídící jednotky. Pro jednorázové odeslání aktuálních hodnot je zde tlačítko „Send“ (3). Pokud máme v úmyslu začít vytvářet kalibrační

Obr. 6: Rozložení ovládacích prvků ve druhé části panelu „Servomotor values“

soubor, načteme stisknutím tlačítka „LDSV“ [1] (4) výchozí hodnoty. Tyto výchozí hodnoty jsou zvoleny tak, aby se servomotory nenatočily do krajních poloh a nebyly poškozeny. Tlačítko „Map Editor“ (5) aktivuje mapový editor pro vytváření trajektorie ve formě lomené čáry a stisknutím tlačítka „Edit SEQ“ (6) se přejde do sekvenčního editoru. Tlačítkem „Run WMP“[2] (8) se spouští sekvence pohybů potom, co byla načtena stisknutím tlačítka „Load SEQ“ (9) (načtení probíhá z *.xml souboru). Pokud je zvolen „Program“, hexapod tuto sekvenci vykonává. Tlačítkem „Stop WMP“ se naopak sekvence pohybů ukončuje. Pokud je nadále zvolen „Program“, končetiny se nastaví do výchozí konfigurace.

3.2.2 CONNECTION SETTINGS

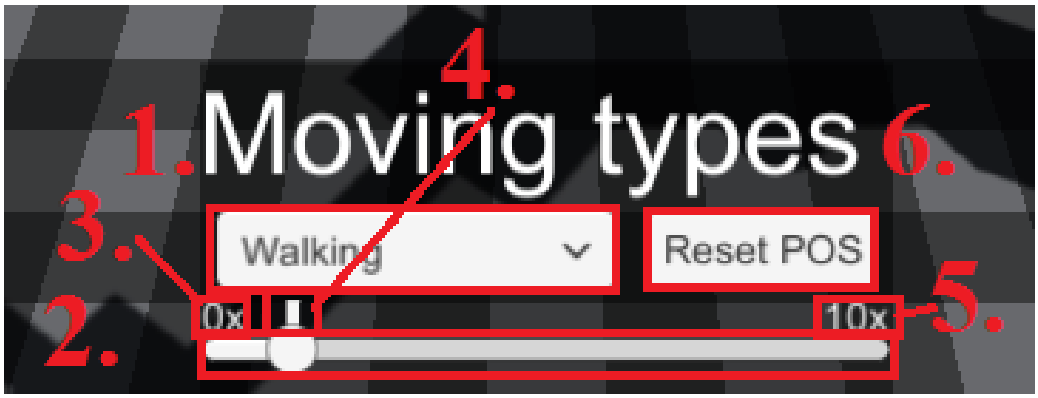
Úkolem panelu je volba komunikačního portu pro připojení mikrokontroléru CC1111 a navázání, případně ukončení spojení. Výběr portu je umožněn v nabídce (1) (viz „dropbox“ na Obr. 7). Navázání, respektive zrušení komunikace je umožněno stisknutím tlačítka „Connect“ (2), respektive „Close“ (3).

Obr. 7: Rozložení ovládacích prvků v panelu Connection settings

[1] Zkratka pochází z „Load default servomotor values“
[2] Zkratka WMP pochází z anglického „Washing Machine Programmer“, algoritmus pracuje na podobném principu.

3.2.3 MOVING TYPES

Díky tomuto panelu může uživatel zvolit druh pohybu a rychlost. Uživatel si může dále



Obr. 8: Rozložení ovládacích prvků v panelu Moving types

volit jednotlivé pohybové módy v nabídce (1) (viz Obr. 8). Výběr je možný z těchto variant pohybů[3]:

Rotating (X, Z) – hexapod se natáčí v osách x a z.

Leaning B/F – hexapod posouvá tělo dopředu a dozadu.

Rotating (Y) – hexapod se periodicky natáčí kolem své svislé osy (přičemž končetiny setrvávají na stejném místě v kontaktu s podložkou).

Waving – hexapod mává přední pravou končetinou (nahoru, dolů, dopředu a dozadu, nikoli do stran).

Up/Down – hexapod posouvá periodicky své tělo po svislé ose.

Walking – tělo hexapoda koná translační pohyb. Další charakteristiky si může uživatel zvolit použitím panelu „Walking settings“ nebo nastavením parametrů bloku s označením „Walking“ v sekvenčním editoru.

Turning – otáčení na místě po nebo proti směru hodinových ručiček (směr je možně měnit pomocí panelu „Walking settings“).

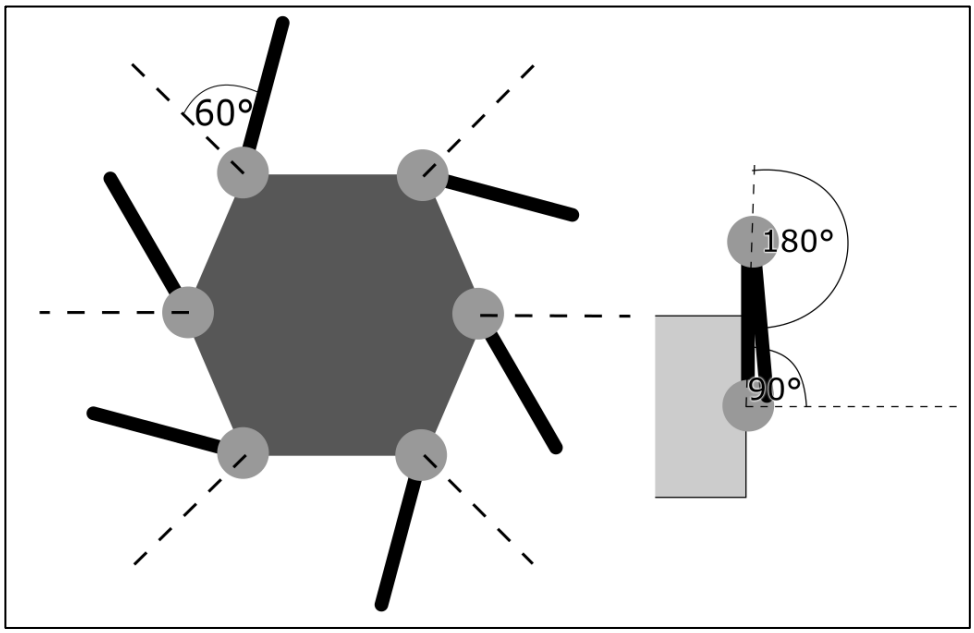
Zdá-li se uživateli rychlost pohybu příliš pomalá, respektive příliš rychlá, může jej upravit pomocí posuvníku (2). Pokud je posuvník umístěn vlevo, čas je pozastaven a žádné pohyby se nekonají. Maximální zrychlení je desetinásobné (posuvník je posunut doprava). Kliknutí na označení 0x (3) způsobí okamžité pozastavení pohybu, naopak kliknutí na označení 10x (5) pohyb maximálně urychlí. Normální rychlost pohybu je vyznačena ukazatelem (4). Tlačítko „Reset POS“ (6) navrátí hexapoda do pozice, kterou zaujímal při spuštění programu.

3.2.4 CALIBRATION SETTINGS

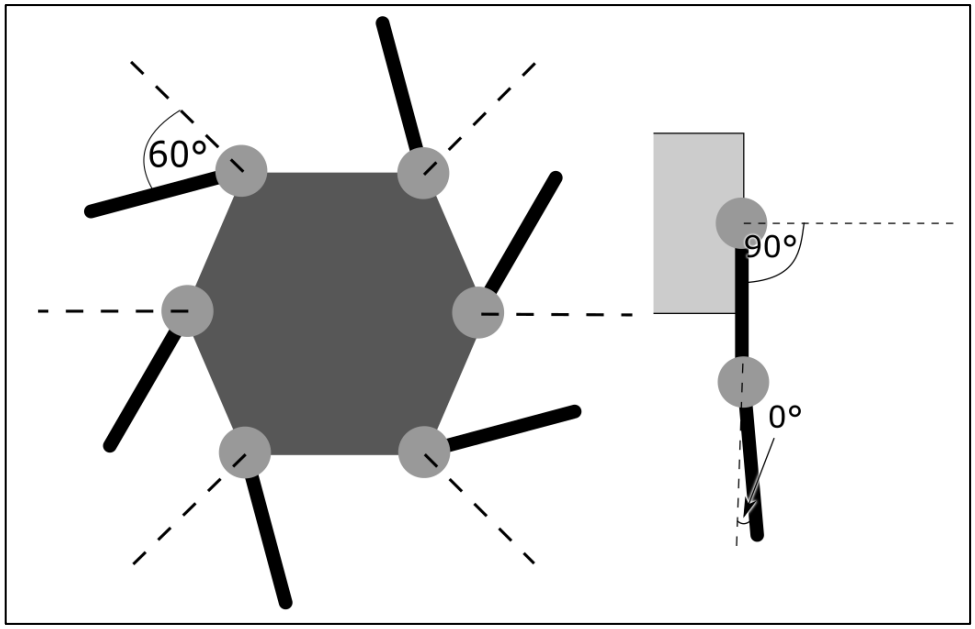
Každý servomotor je zatížen jistou individuální chybou, proto je nutné každý nezávisle zkalibrovat. Zjistili jsme, že je možné pro kalibraci použít lineární regresi $r = ka + q$, kde r je

[3] Tento výčet bude do budoucna rozšiřován o další možnosti.

hodnota odesílaná do řídící jednotky hexapoda, a je úhel odpovídající příslušné hodnotě, k a q jsou parametry lineární regrese. Pro dostatečně přesný výsledek nám stačí znát dvě hodnoty. Z fyzického uspořádání končetin hexapoda byly tedy zvoleny dvě kalibrační konfigurace (viz Obr. 9 a Obr. 10). Při první kalibrační konfiguraci jsou servomotory natáčející končetiny kolem osy y natočeny o 60° po směru hodinových ručiček od osy úhlu maximálního rozsahu. Prostřední části



Obr. 9: První část kalibrace



Obr. 10: Druhá část kalibrace

končetin svírají s vodorovnou rovinou 90° v kladném smyslu rotace a servomotory v kloubech končetin jsou natočeny tak, že, obě části končetin jsou přitisknuty k sobě (odchylky os prvních

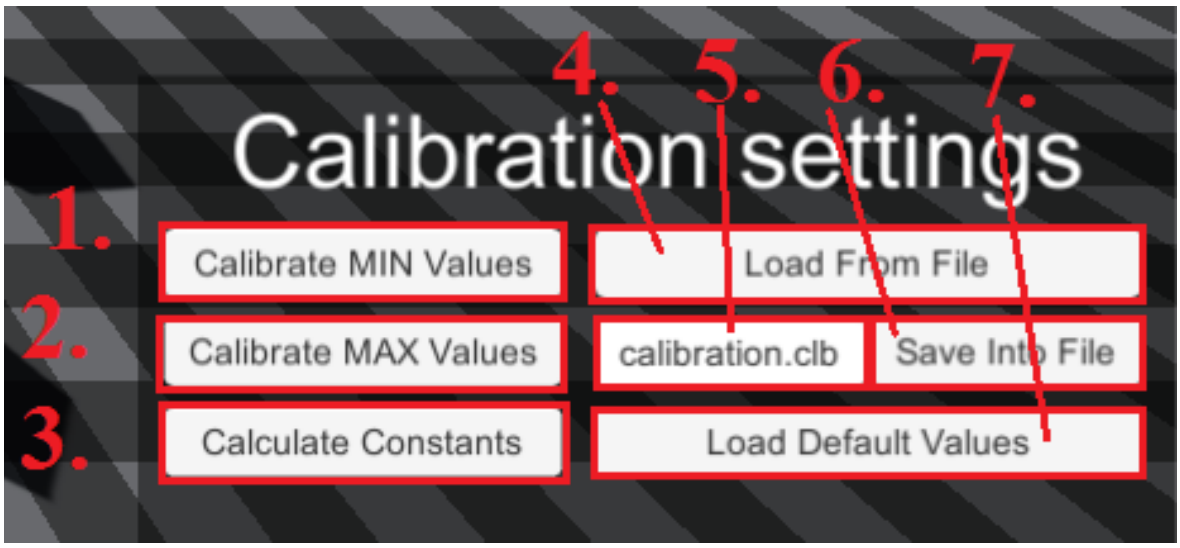
částí končetin a druhých částí končetin jsou 0°). Druhá konfigurace vznikne z té první změnou smyslu rotace všech úhlů (viz Obr. 10).

Po dosažení první kalibrační polohy je nutné příslušné hodnoty uložit, k čemuž slouží tlačítko „Calibrate MIN Values“ (1) na panelu „Calibration settings“ (viz Obr. 11). Hodnoty druhé kalibrační konfigurace se ukládají po stisknutí tlačítka „Calibrate MAX Values“ (2). Následuje výpočet parametrů k a q lineární regrese. Označme a_1 hodnotu úhlu nastavenou na libovolném servomotoru v první kalibrační konfiguraci a a_2 hodnotu úhlu nastavenou na tomtéž servomotoru ve druhé kalibrační konfiguraci. Nechť jsou r_1 a r_2 reálné hodnoty vypočtené pro tento servomotor. Hodnoty k a q získáme následovně:

$$k = \frac{r_2 - r_1}{a_2 - a_1}, \tag{1}$$

$$q = r_1 - a_1 k. \tag{2}$$

Tento výpočet se automaticky provede pro všech osmnáct servomotorů a je iniciován stisknutím



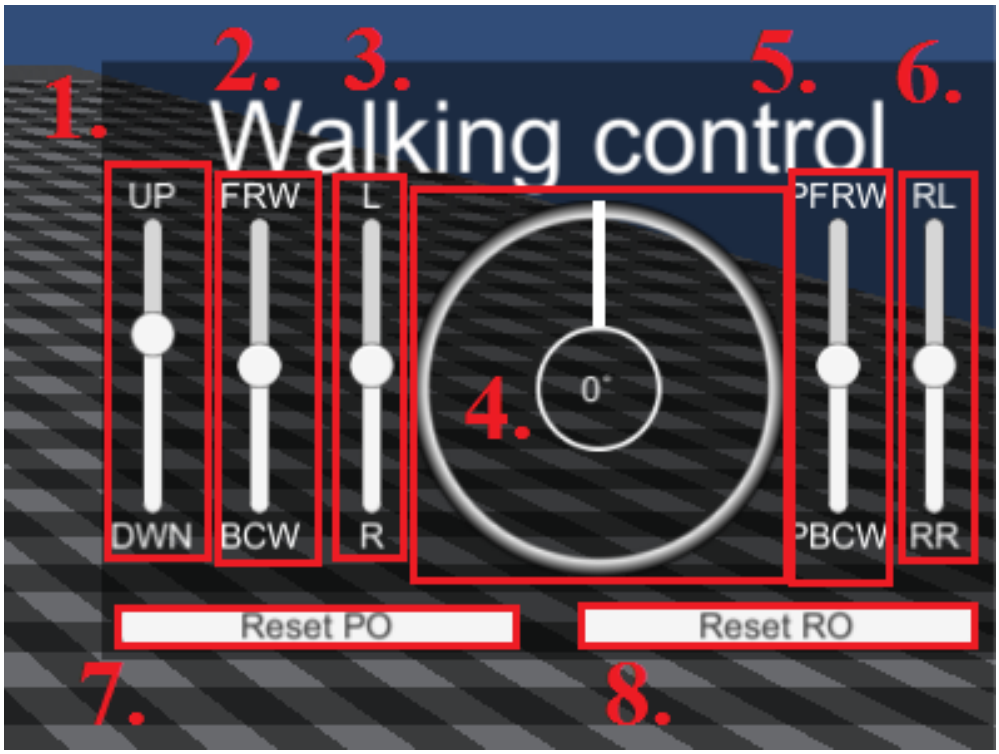
Obr. 11: Uživatelské rozhraní kalibračního panelu

tlačítka „Calculate Constants“ (3). K uložení těchto hodnot do specifikovaného souboru slouží tlačítko „Save Into File“ (6). K pojmenování tohoto souboru slouží textové pole (5). Pro použití kalibrací v simulaci je podstatné jejich načtení, což se provede stisknutím tlačítka „Load From File“ (4). V případě potřeby může uživatel načíst výchozí hodnoty stisknutím tlačítka „Load Default Values“ (7).

3.2.5 WALKING SETTINGS

Panel „Walking settings“ slouží k úpravě pohybu hexapoda (předpokladem pro použití tohoto panelu je volba pohybových módů chůze nebo otáčení). Konkrétně posuvník „UP-DWN“ (1) (viz Obr. 12) slouží k posunu těla podél svislé osy, posuvník FRW-BCW“ (2) k posunu těla hexapoda dopředu nebo dozadu a posuvník „L-R“ (3) k posunu těla doleva nebo doprava. Je-li zvolen mód chůze, nastavuje se otáčením ukazatele (4) směr pohybu hexapoda. Pokud je zvolen

mód rotace dá se tímto způsobem změnit směr otáčení. Otáčení po směru hodinových ručiček odpovídá hodnota 0° a proti směru hodinových ručiček hodnota 180°.[4] Dále jsou zde k nalezení



Obr. 12: Uživatelské rozhraní panelu Walking control

posuvníky upravující sklon těla. Posuvníkem „PFRW-PBCW“ (5) se mění naklopení těla dopředu či dozadu (tzv. pitch) a posuvníkem „RL-RR“ (6) se realizuje naklopení do stran (tzv. roll). Pokud se chceme navrátit k původnímu nastavení, využijeme tlačítka (7) a (8). Tlačítko „Reset PO“ vrací do původních pozic posuvníky (1), (2) a (3) a tlačítko „Reset RO“ vrací do původního nastavení posuvníky (5) a (6).

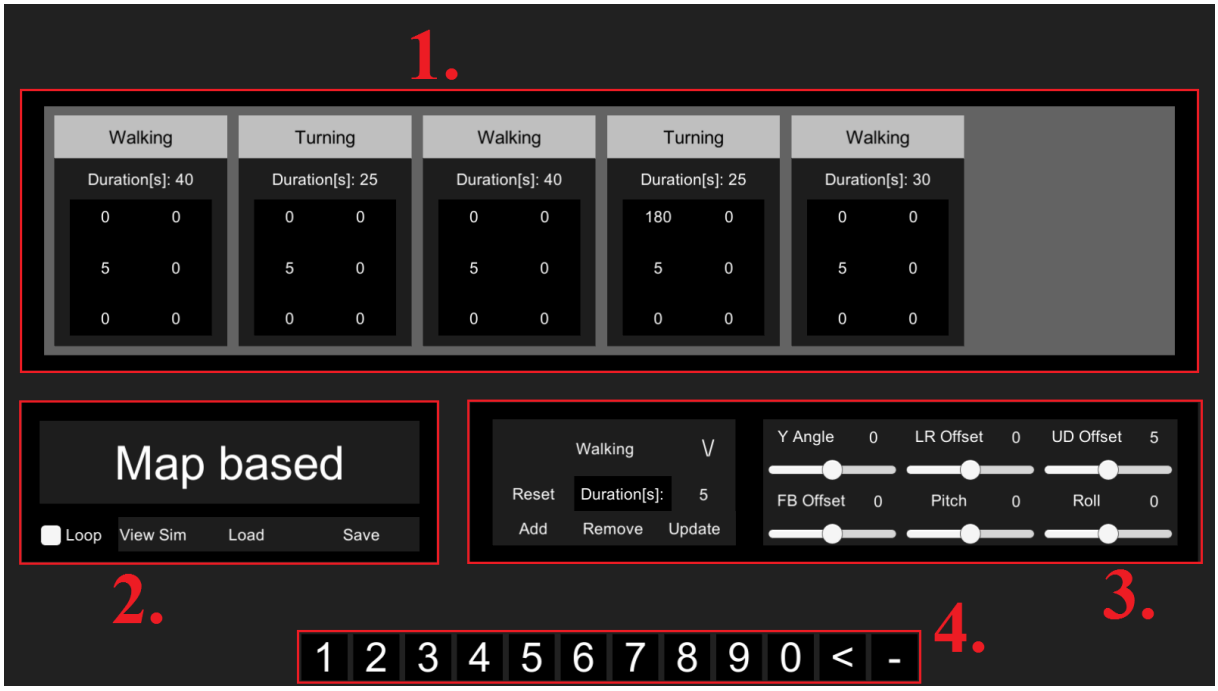
3.3 Editor sekvence kroků

Další částí uživatelského rozhraní je editor sekvence pohybů. Jeho vzhled je patrný z obrázku Obr. 13. V tomto editoru je uživateli umožněno vytvořit, upravit a uložit sekvenci pohybů, jinými slovy si hexapoda naprogramovat. Editor je koncipován pro spuštění na tabletu. Vytvořenou sekvenci uživatel uloží přes sdílení souborů do vzdáleného adresáře a PC tuto sekvenci načte. Toto spojení umožní uživateli, aby hexapoda ovládal dálkově jen pomocí tabletu.

[4]Ukazatelem se nastavuje hodnota argumentu pro rotační matici $R_y(\theta)$ (viz druhá část této práce). Pro hodnotu 0° přejde R_y v identitu. Pro hodnotu 180° změně směru oběhu po křivce, a proto se hexapod otáčí obráceně.

Druhá možnost dálkového ovládání by byla možná například pomocí tzv. „Photon view“, což je doplněk pro Unity. Ten by umožňoval spojení přes internet, ne pouze přes LAN. Pro implementování této možnosti by ovšem byla potřeba relativně velká úprava kódu a pokaždé by musel být PC i řídící tablet připojen k internetu, což přináší určité komplikace.

Editor se skládá celkem ze čtyř panelů. Prvním panelem je přehled pohybových módů (1)



Obr. 13: Rozložení ovládacích panelů v editoru sekvence kroků

v sekvenci, kde jsou tyto módy reprezentovány bloky obsahujícími číselné hodnoty jednotlivých parametrů těchto pohybů. Další panel (2) umožňuje základní operace na vytvořené sekvenci. Dá se jím měnit název sekvence, nastavit její opakování, uložit sekvenci, respektive ji načíst. Panel (3) slouží ke zvolení typu pohybu, nastavení parametrů pohybu, přidání nebo odebrání takto vytvořeného bloku do sekvence, popřípadě aktualizaci tohoto bloku. K nastavení parametrů na tabletu (či jiném zařízení) je zde implementována jednoduchá numerická klávesnice (4).

V následujících kapitolách podrobně popíšeme jednotlivé bloky tohoto vývojového prostředí.

3.3.1 Přehled kroků v sekvenci

Pro vytvoření sekvence pohybů jsou uživateli k dispozici pohybové módy identické s těmi v nabídce pohybových módů „Moving types“. Blok sekvence může uživatel označit kurzorem. Po označení jej může editovat nebo rovnou odstranit. Vzhled bloku můžete vidět na obrázku Obr. 14 pod číslem (1). V hlavičce (2) každého bloku je uveden název pohybového módu, který je blokem reprezentován. Dále je zde uvedena doba trvání pohybu v sekundách (3). Pod tímto údajem je k nalezení přehled parametrů. Rozložení jednotlivých údajů je patrné z tabulky Tab. 1.

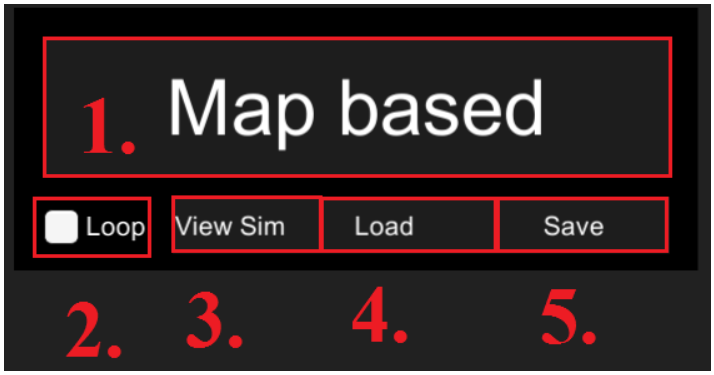
Obr. 14: Rozložení prvků v přehledu kroků sekvence

Y angle (argument matice R_y)	LR offset (posun do stran)
UD offset (posun nahoru nebo dolu)	FB offset (posun dopředu nebo dozadu)
Pitch	Roll

Tab. 1: Rozložení prvků dat

3.3.2 ZÁKLADNÍ OPERACE S EDITOVANOU SEKVENCÍ

Tento panel obsahuje základní nastavení sekvence. Změnu názvu sekvence umožňuje

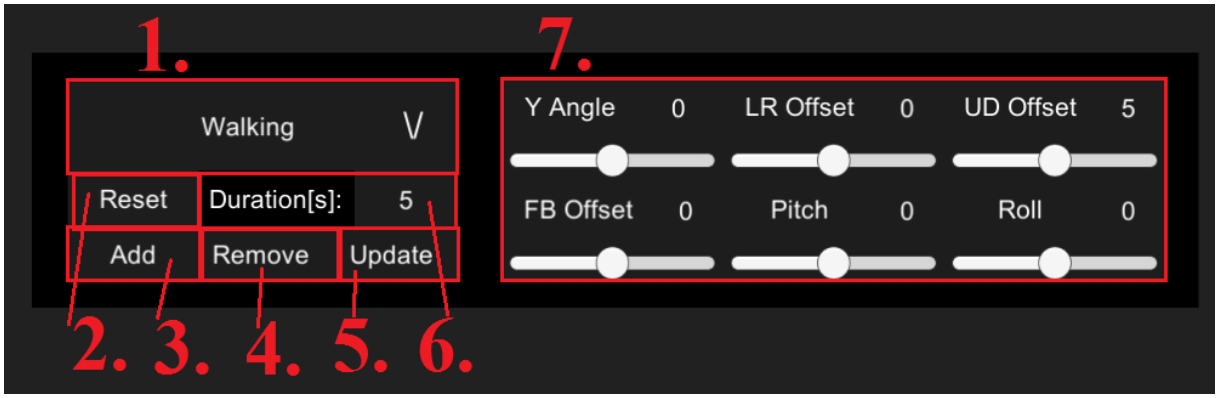


Obr. 15: Rozložení prvků v panelu se základními údaji aktuálně editované sekvence

textové pole (1). Zaškrtnutím checkboxu „Loop“ (2) se uvede sekvence do smyčky a bude se stále opakovat. Tlačítkem „Save“ (5) se sekvence uloží a stisknutím tlačítka „Load“ (4) se načte. Návrat k vizualizaci se provede kliknutím na tlačítko „View Sim“ (3).

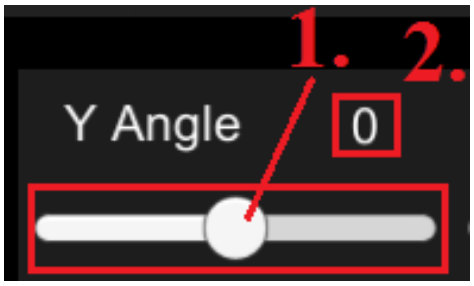
3.3.3 NASTAVENÍ ZVOLENÉHO BLOKU SEKVENCE

Tento panel slouží k vytvoření nového nebo upravení již vytvořeného bloku sekvence. Tvoření nového bloku obnáší výběr pohybového módu z nabídky (1) (Obr. 16), nastavení požadované doby trvání v textovém poli (6) (časovou jednotkou je zde sekunda) a poté přidání bloku stisknutím tlačítka „Add“ (3). Bloky se přidávají vždy na konec sekvence. Pro úpravu již existujícího bloku je nutné ho nejdříve vybrat (kliknutím levého tlačítka myši nebo použitím



Obr. 16: Rozložení prvků v panelu pro nastavení zvoleného bloku sekvence

dotykové obrazovky), poté nastavit požadované hodnoty a potvrdit tlačítkem „Update“ (5). Pro odebrání bloku slouží tlačítko „Remove“ (4). Volba parametrů pohybu je provedena posuvníky (7) nebo zadáním údajů do textových polí nad posuvníky (na obrázku Obr. 17 je uveden příklad (1) a (2) pro „Y Angle“). Význam jednotlivých parametrů je zřejmý z tabulky Tab. 1. Tlačítko „Reset“ (2) uvede blok do původního nastavení.



Obr. 17: Způsoby nastavování parametrů

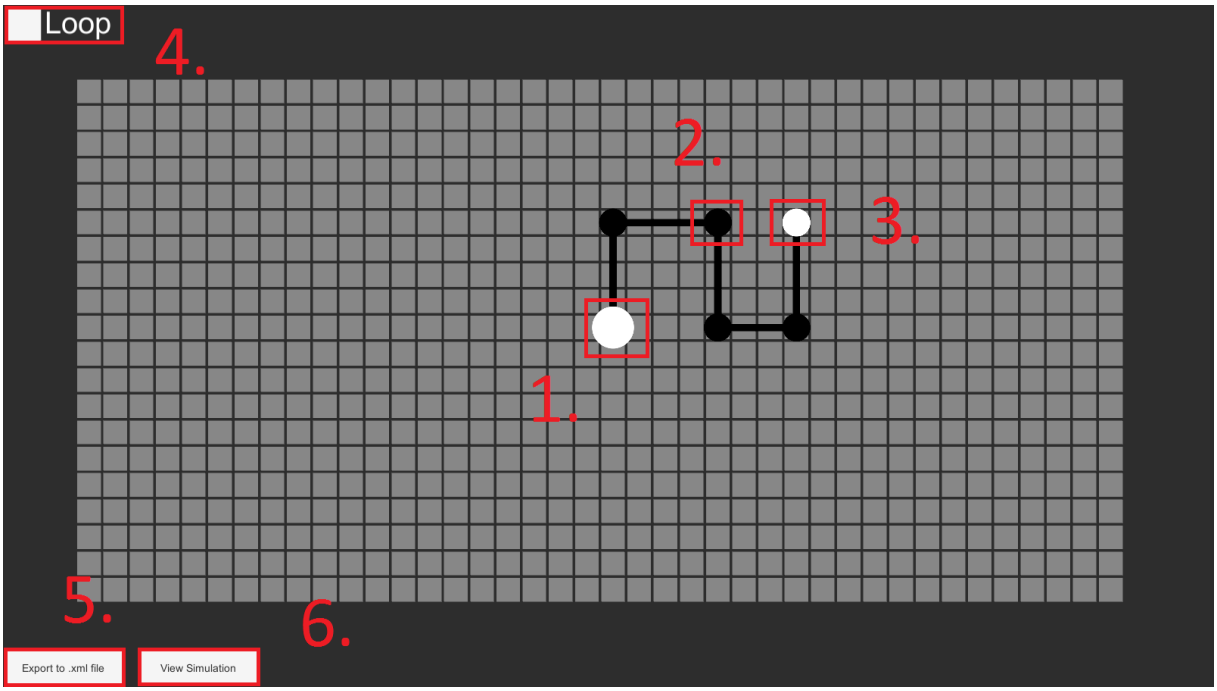
3.3.4 SOFTWAROVÁ KLÁVESNICE

Uživateli je pomocí této klávesnice (viz Obr. 18) umožněno měnit hodnoty textových (resp. číselných) polí bez hardwarové klávesnice. Například používá-li uživatel k ovládání tablet. Osahuje čísla od 0 do 9 (1) a dvě další klávesy. První klávesa (2) slouží k vymazání předchozího znaku a druhá (3) ke změně znamínka upravované číselné hodnoty.

Obr. 18: Softwarová klávesnice

3.4 Mapový editor

Mapový editor slouží k naplánování trasy pro hexapoda ve formě lomené čáry a následné převedení této lomené čáry do sekvence rotačních a translačních pohybů (otáčení a chůze dopředu), které mohou být dále upraveny v sekvenčním editoru a načteny do simulace. Vzhled mapového editoru je zřejmý z obrázku Obr. 19.



Obr. 19: Rozložení ovládacích a visuálních prvků v mapovém editoru

Kliknutím na libovolný čtvereček sítě je zde umístěn jeden bílý bod, nazvali jsem jej „Waypoint“ (3). Klikneme-li na další čtvereček, poslední „Waypoint“ změní barvu z bílé na černou (2) a je úsečkou propojen s novým bílým koncovým bodem lomené čáry. Počáteční „Waypoint“ (1) je větší než ostatní, má bílou barvu a je umístěn automaticky po aktivaci editoru. Opět je zde možnost opakování celkové trasy zaškrtnutím checkboxu „Loop“ (4). Než bude moci uživatel přejít stisknutím tlačítka „View Simulation“ (6) zpět k vizualizaci, kliknutím na tlačítko „Export to .xml file“ (5) vypočtené hodnoty exportuje do *.xml souboru.

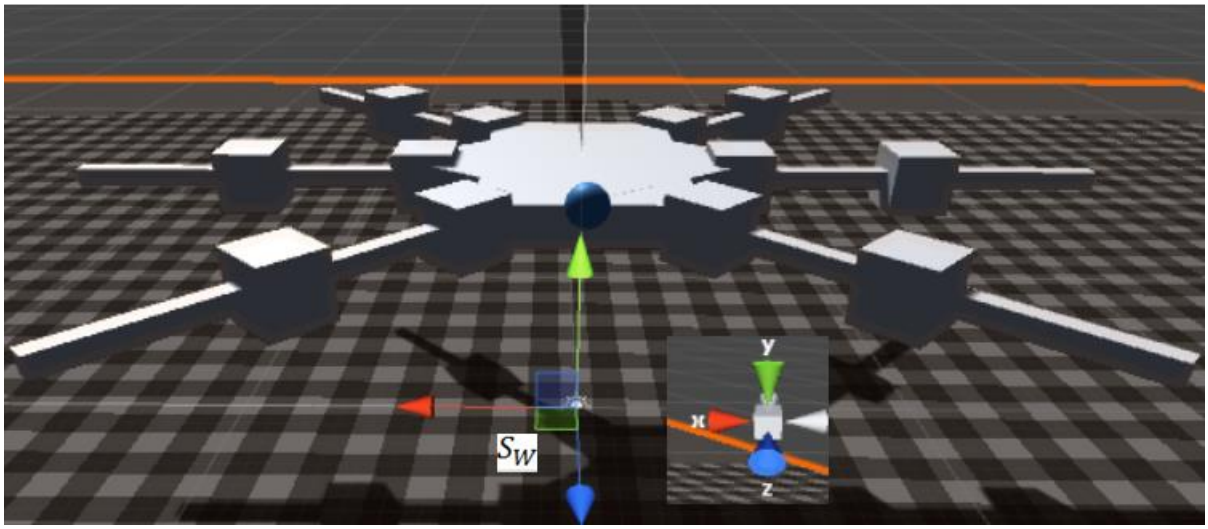
3.5 Připravované změny

Naším plánem je vytvořit nový mapový editor založený na modelování trajektorií v podobě spojité křivky. Původní editor založený na vytváření tras pomocí lomené čáry samozřejmě zachováme, ale vytvoříme jemu podobný. Náš nový editor bude na bázi Bézierových křivek, takže se design uživatelského rozhraní moc lišit nebude. Dalším uvažovaným vylepšením je panel zobrazující okamžité úhlové rychlosti rotace jednotlivých servomotorů. O matematické podstatě těchto dvou vylepšení se dočtete na konci následujícího oddílu naší práce.

4 MATEMATICKÝ MODEL

4.1 Souřadnicové soustavy

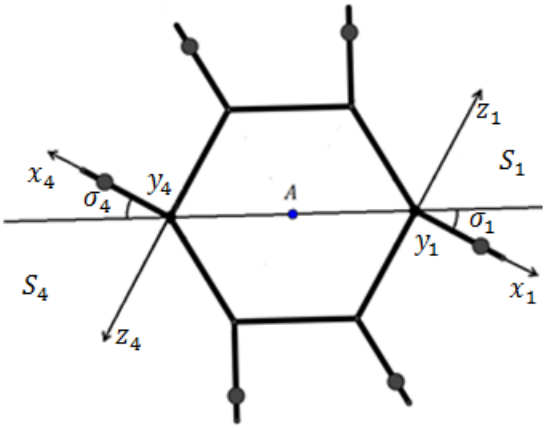
Pro popis polohy hexapoda je výhodné zavést referenční souřadnicovou soustavu. Nazvali jsem ji světová souřadnicová soustava a budeme ji značit S_W. V programu jsme její počátek umístili dvě délkové jednotky[5] pod tělo hexapoda (budeme uvažovat, že hexapod je v počáteční pozici ihned po aktivaci programu). Osa y_W této soustavy je kolmá k tělu hexapoda, prochází středem jeho těla (bod A na obrázku Obr. 21) a její kladná poloosa míří nad podložku. Osa x_W leží v kolmé projekci druhého a pátého paprsku[6] do roviny podložky. Její kladná poloosa je orientována ve směru druhého paprsku. Osa z_W tento systém doplňuje na levotočivou soustavu souřadnic. Celá situace je znázorněna na obrázku Obr. 20.



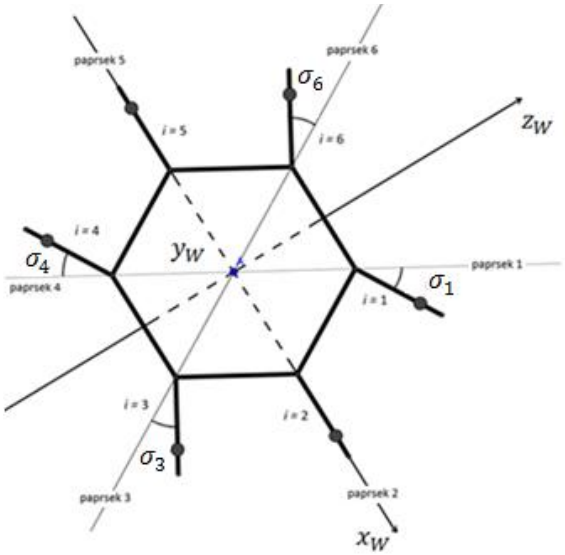
Obr. 20: Umístění světové souřadnicové soustavy při aktivaci programu, ve spodní části jsou jednotlivé osy pojmenovány

[5] V celé práci budeme užívat metrické jednotky z Unity. Budeme je značit „j“.
[6] Paprsky jsou znázorněné na obrázku Obr. 21 a tvoří je polopřímky začínající v bodě A a procházející jednotlivými počátky končetin. Každý paprsek nese číslo dané končetiny.

Pro popis polohy konců jednotlivých končetin je vhodné zavést další sadu souřadnicových soustav umístěných do počátků jednotlivých končetin. Každá končetina má jednu individuální soustavu. Označili jsme je S_1 až S_6 (číslování jednotlivých končetin je patrné z obrázku Obr. 22). Jednotlivé soustavy jsou koncipovány tak, že kladné poloosy x_i $(i = 1, \dots, 6)$ míří od středu těla hexapoda a obecně jsou mezi nimi a příslušnými paprsky nenulové[7] odchylky σ_i. Osy y_i leží v osách, kolem kterých se končetiny mohou natáčet v horizontální rovině[8]. Kladné poloosy y_i míří nad podložku. Osy z_i jsme zvolili tak, aby doplňovaly tyto soustavy opět do levotočivých souřadnicových soustav. Na obrázku obr. 21 jsou jako ilustrační příklad znázorněné soustavy S_1 a S_4.



Obr. 21: Názorné umístění souřadnicových soustav (pohled shora)



Obr. 22: Schématické zobrazení těla hexapoda v počátečním nastavení (pohled shora)

[7] Přesněji řečeno jsou tyto odchylky nenulové pro první, třetí, čtvrtou a šestou končetinu, viz tabulka parametrů Tab. 3 v příloze.

[8] Nosné přímky os y_i tedy odpovídají svislým osám hřídelí servomotorů připevněných k tělu hexapoda.

4.2 Matice přechodu

V této kapitole bychom rádi uvedli odvození a základní vlastnosti matic přechodu reprezentujících rotace v E^3 (trojrozměrném euklidovském prostoru). Ve druhé části této kapitoly zavedeme pojem homogenní souřadnice a budeme se zabývat maticovou reprezentací translace E^3. Definice pojmů jako vektorový prostor, báze vektorového prostoru, skalární součin a podobně zde nebudeme uvádět. Pro zájemce jsou k nalezení např. v [7] nebo [8].

Nechť $B_1 = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3\}$ a $B_2 = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3\}$ jsou dvě různé ortogonální báze prostoru E^3. Ortogonalita množiny vektorů $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$ znamená, že pro $i, j = 1, 2, 3$ je $(\boldsymbol{v}_i|\boldsymbol{v}_j) = 0$, pokud $i \neq j$ a $(\boldsymbol{v}_i|\boldsymbol{v}_j) \neq 0$ pokud $i = j$. Vektory báze B_1 jsou lineární kombinací vektorů báze B_2 (platí to i naopak, viz inverzní transformace). Tuto skutečnost můžeme zapsat takto[7]:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \boldsymbol{m}_3 \end{pmatrix} = \begin{pmatrix} \boldsymbol{n}_1 \\ \boldsymbol{n}_2 \\ \boldsymbol{n}_3 \end{pmatrix}. \tag{3}$$

Za pomoci stejné matice (označme si ji A) se transformují i složky vektorů. Této matici se říká matice přechodu od báze B_2 k bázi B_1.

Nyní budeme řešit obecné vlastnosti matic přechodu reprezentujících rotaci. Jednou z vlastností rotace je, že zachovává normy vektorů [7]. Normu vektoru $\boldsymbol{v}$ lze definovat vztahem $\|\boldsymbol{v}\| = \sqrt{(\boldsymbol{v}|\boldsymbol{v})}$ [7]. Nechť $\boldsymbol{y} = A\boldsymbol{x}$. Z rovnosti norem vektorů $\boldsymbol{x}$ a $\boldsymbol{y}$ tak plyne:

$$(\boldsymbol{x}|\boldsymbol{x}) = (\boldsymbol{y}|\boldsymbol{y}). \tag{4}$$

Rozepišme tuto rovnost:

$$\sum_{i=1}^{3} y_i^2 = \sum_{j=1}^{3} x_j^2. \tag{5}$$

Složky vektoru $\boldsymbol{y}$ můžeme vyjádřit rozepsáním součinu $\boldsymbol{y} = A\boldsymbol{x}$ takto:

$$y_i = \sum_{k=1}^{3} a_{ik} x_k. \tag{6}$$

Po dosazení (6) do (5) dostaneme:

$$\sum_{i=1}^{3} \left(\sum_{k=1}^{3} a_{ik} x_k \right)^2 = \sum_{i=1}^{3} \left(\sum_{l=1}^{3} a_{il} x_l \right) \left(\sum_{m=1}^{3} a_{im} x_m \right) = \sum_{j=1}^{3} x_j^2.$$

Úpravou získáme [9]:

$$\sum_{i,l,m=1}^{3} a_{il} a_{im} x_l \, x_m = \sum_{j=1}^{3} x_j^2,$$

tedy:

$$a_{il} a_{im} = \begin{cases} 1 \text{ pokud } l = m \\ 0 \text{ pokud } l \neq m \end{cases},$$

neboli:

$$AA^T = I. \tag{7}$$

Což znamená, že $A^{-1} = A^T$. To je tedy první obecná vlastnost námi hledaných matic.

Determinanty transponované matice a matice původní jsou stejné a determinant součinu matic je roven součinu jednotlivých determinantů těchto matic [7]. Platí tedy $\det(AA^T) = (\det(A))^2 = 1$. Dostáváme tak obecnou hodnotu determinantu matice rotace:

$$|\det(A)| = 1. \tag{8}$$

To, že rotace zachovává délku implikuje zachování skalárního součinu $(A\boldsymbol{u}|A\boldsymbol{v}) = (\boldsymbol{u}|\boldsymbol{v})$ [7]. Důkaz je snadný, stačí využít vlastností skalárního součinu:

$$\|\boldsymbol{u}+\boldsymbol{v}\|^2 = (\boldsymbol{u}+\boldsymbol{v}|\boldsymbol{u}+\boldsymbol{v}) = (\boldsymbol{u}|\boldsymbol{u}) + 2(\boldsymbol{u}|\boldsymbol{v}) + (\boldsymbol{v}|\boldsymbol{v}) = \|\boldsymbol{u}\|^2 + \|\boldsymbol{v}\|^2 + 2(\boldsymbol{u}|\boldsymbol{v}).$$

Z toho:

$$(\boldsymbol{u}|\boldsymbol{v}) = \frac{1}{2}(\|\boldsymbol{u}+\boldsymbol{v}\|^2 - \|\boldsymbol{u}\|^2 - \|\boldsymbol{v}\|^2), \tag{9}$$

a také

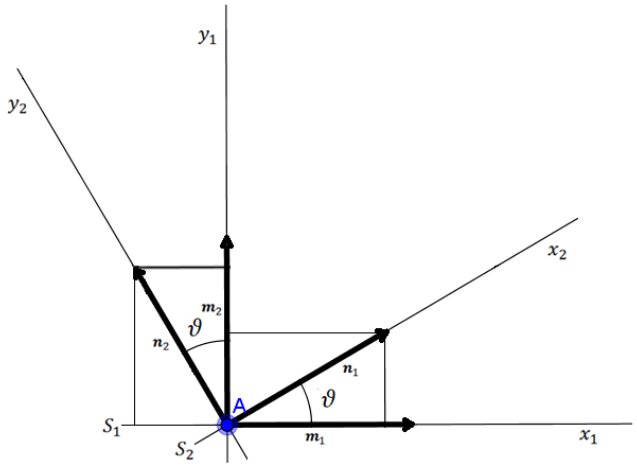
$$(A\boldsymbol{u}|A\boldsymbol{v}) = \frac{1}{2}(\|A(\boldsymbol{u}+\boldsymbol{v})\|^2 - \|A\boldsymbol{u}\|^2 - \|A\boldsymbol{v}\|^2).$$

Řekli jsem si, že rotace zachovává vzdálenosti, proto při použití (9) dostaneme:

$$(A\boldsymbol{u}|A\boldsymbol{v}) = \frac{1}{2}(\|\boldsymbol{u}+\boldsymbol{v}\|^2 - \|\boldsymbol{u}\|^2 - \|\boldsymbol{v}\|^2) = (\boldsymbol{u}|\boldsymbol{v}).$$

Je zřejmé, že rotace dvojice vektorů o stejný úhel kolem stejné osy zachovává jejich odchylky, svědčí o tom i tento výsledek. Reálná matice lineárního zobrazení splňující podmínku $A^{-1} = A^T$ se nazývá ortogonální [7].

Přejdeme tedy k samotnému odvození matice rotace, provedeme jej pouze pro matici rotace kolem osy z, ostatní se odvozují analogicky.



Obr. 23: Jednotlivé báze reprezentují příslušné souřadnicové soustavy, osa z míří kolmo nad nákresnu v bodě A

Souřadnicová soustava S_1 je reprezentována bází $\{\boldsymbol{m}_1; \boldsymbol{m}_2; \boldsymbol{m}_3\}$. Přechod od S_1 k S_2 lze tedy realizovat přechodem od báze $\{\boldsymbol{m}_1; \boldsymbol{m}_2; \boldsymbol{m}_3\}$ k bázi $\{\boldsymbol{n}_1; \boldsymbol{n}_2; \boldsymbol{n}_3\}$ reprezentující S_2. Naším cílem

je vyjádřit vektory $\boldsymbol{n}_1$, $\boldsymbol{n}_2$ a $\boldsymbol{n}_3$ jako lineární kombinaci vektorů $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ a $\boldsymbol{m}_3$. Z obrázku Obr. 23 je patrné, že:

$$\boldsymbol{n}_1 = \boldsymbol{m}_1 \cos(\vartheta) + \boldsymbol{m}_2 \sin(\theta),$$
$$\boldsymbol{n}_2 = -\boldsymbol{m}_1 \sin(\vartheta) + \boldsymbol{m}_2 \cos(\theta),$$
$$\boldsymbol{n}_3 = \boldsymbol{m}_3.$$

Maticově zapsáno:

$$\begin{pmatrix} \boldsymbol{n}_1 \\ \boldsymbol{n}_2 \\ \boldsymbol{n}_3 \end{pmatrix} = \begin{pmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 \\ -\sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \boldsymbol{m}_3 \end{pmatrix}. \tag{10}$$

Matici na pravé straně této rovnice budeme značit $R_z(\vartheta)$. Rotační matice pro rotaci kolem os x a y by byly [9]:

$$R_x(\vartheta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\vartheta) & \sin(\vartheta) \\ 0 & -\sin(\vartheta) & \cos(\vartheta) \end{pmatrix}, \tag{11}$$

a

$$R_y(\vartheta) = \begin{pmatrix} \cos(\vartheta) & 0 & -\sin(\vartheta) \\ 0 & 1 & 0 \\ \sin(\vartheta) & 0 & \cos(\vartheta) \end{pmatrix}. \tag{12}$$

Dále si povšimněme, že matice jsou opravdu ortogonální[9]. Determinanty těchto matic jsou proto rovny jedné:

$$\det\big(R_x(\vartheta)\big) = \det\Big(R_y(\vartheta)\Big) = \det\big(R_z(\vartheta)\big) = (\sin(\vartheta))^2 + (\cos(\vartheta))^2 = 1.$$

Nyní je na řadě již avizovaný maticový zápis translace. Kvůli tomu je ale nutné zavést homogenní souřadnice [10]. Na místo vektorů typu 3×1 je výhodné používat vektory typu 4×1. Na poslední pozici každého vektoru se zapíše číslo 1:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

Podobně převedeme všechny matice A z typu 3×3 na typ 4×4 podle tohoto blokového schématu (značení matic však zachováme)[10]:

$$A \rightarrow \begin{pmatrix} & & & 0 \\ & A & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Pokud budeme provádět translaci bodu $P = [x, y, z]$ reprezentovanou vektorem $\boldsymbol{t} = (a, b, c)^T$, sestavíme matici takto [10]:

$$T = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{13}$$

[9] Vyhovují podmínce $A^{-1} = A^T$. Inverzní matici k obecné matici rotace o úhel ϑ obdržíme záměnou $\vartheta \rightarrow -\vartheta$.

Bod P je určen v homogenních souřadnicích polohovým vektorem $\boldsymbol{p} = (x, y, z, 1)^T$. Platí:

$$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + a \\ y + b \\ z + c \\ 1 \end{pmatrix},$$

Vektor $(x + a, y + b, z + c, 1)^T$ je polohový vektor obrazu bodu P. Zde se nejedná o matici přechodu mezi bázemi (skutečná matice přechodu reprezentující translaci je jednotková [7]).

V závěru této kapitoly sestavíme matici P_i přechodu mezi světovou souřadnicovou soustavou a soustavami S_i umístěnými do počátků končetin. Jak ukážeme později, trajektorie jsou popsány parametrickými rovnicemi vůči světové souřadnicové soustavě. První transformace odpovídá translaci počátku světové souřadnicové soustavy do středu horní strany těla hexapoda. Takto posunutou světovou soustavu souřadnic označme S_W^T. Příslušná matice translace vypadá takto:

$$T_W = \begin{pmatrix} 1 & 0 & 0 & -x_W \\ 0 & 1 & 0 & -y_W \\ 0 & 0 & 1 & -z_W \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{14}$$

kde x_W, y_W a z_W představují souřadnice středu horní strany těla hexapoda ve světové souřadnicové soustavě. V dalším kroku transformace otočíme takto posunutou světovou souřadnicovou soustavu S_W^T kolem její osy y_W^T o úhel η_y. Novou soustavu označme S_W^{TY}. Tím přejde osa z_W^T na osu z_W^{TY}, kolem které následně soustavu S_W^{TY} otočíme o úhel η_z. Vzniklou soustavu označme S_W^{TYZ}. Třetí otočení provedeme kolem osy x_W^{TYZ} o úhel η_x. Soustavu S_W^{TYZ} převedeme na soustavu S_W^{TYZX}. Úhly η_x, η_y a η_z volíme tak, aby byla osa y_W^{TYZX} kolmá k horní straně těla hexapoda, osa z_W^{TYZX} tvořila osu úhlu mezi prvním a šestým paprskem a osa x_W^{TYZX} tvořila nosnou přímku druhého a pátého paprsku (viz výše). V současné době připravujeme gyroskopy a kompas, kterými tyto úhly budeme měřit. Předposlední transformací je translace do počátků jednotlivých končetin daná maticí:

$$T_i = \begin{pmatrix} 1 & 0 & 0 & -x_i \\ 0 & 1 & 0 & -y_i \\ 0 & 0 & 1 & -z_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{15}$$

kde x_i, y_i a z_i jsou souřadnice počátku i-té končetiny v S_W^{TYZX}. Poslední transformací je individuální natočení těchto posunutých soustav o úhly σ_i v rotacích reprezentovaných maticemi $R_y(\sigma_i)$. Transformaci P_i můžeme schematicky znázornit takto:

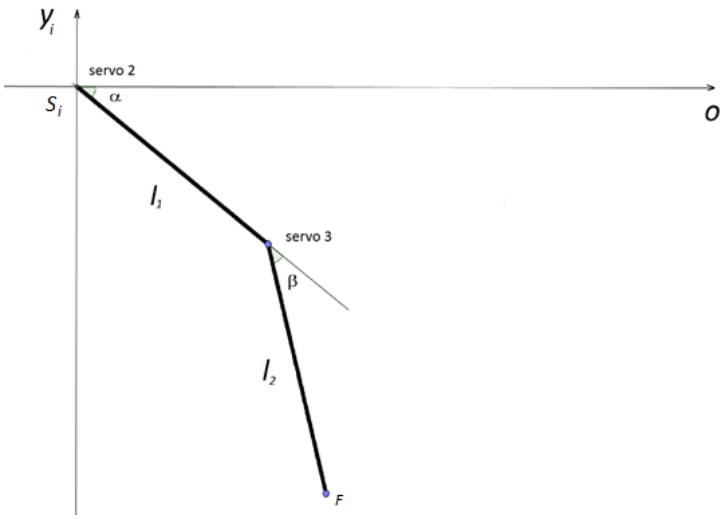
$$S_W \xrightarrow{T_W} S_W^T \xrightarrow{R_y(\eta_y)} S_W^{TY} \xrightarrow{R_z(\eta_z)} S_W^{TYZ} \xrightarrow{R_x(\eta_x)} S_W^{TYZX} \xrightarrow{T_i R_y(\sigma_i)} S_i,$$

z čehož plyne příslušná maticová reprezentace:

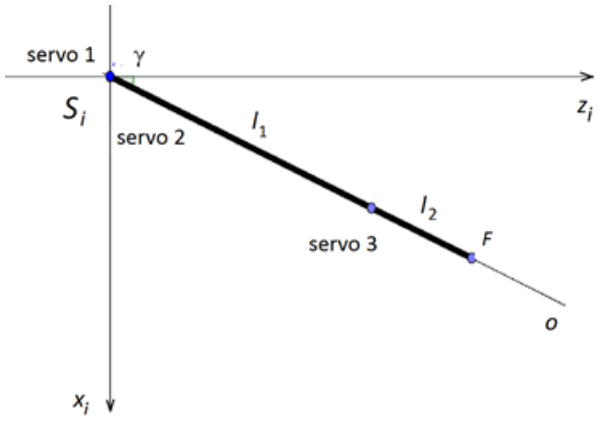
$$P_i = R_y(\sigma_i) T_i R_x(\eta_x) R_z(\eta_z) R_y(\eta_y) T_W. \tag{16}$$

4.3 Úhly natočení servomotorů

Každá končetina se skládá ze dvou částí o délkách l_1 a l_2 (konkrétní hodnoty jsou uvedeny v tabulce Tab. 3, viz příloha).



Obr. 24: Pohled na i-tou končetinu z boku, osa o odpovídá kolmé projekci končetiny do roviny horní strany těla



Obr. 25: Pohled na i-tou končetinu ze shora

Vyznačené úhly odpovídají jednotlivým stupňům volnosti. Každý servomotor ovládá jeden tento stupeň. V simulaci jsou hodnoty úhlů omezeny, přesněji $\alpha_i \in \langle -\frac{\pi}{2}; \frac{\pi}{2} \rangle$, $\beta_i \in \langle -\pi; 0 \rangle$ a $\gamma_i \in \langle -\frac{\pi}{3}; \frac{\pi}{3} \rangle$. Nechť $F = [x_i, y_i, z_i]$, z kosinové věty jsme odvodili vztahy:

$$\alpha_i = \sin^{-1}\left(\frac{|y_i|}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\right) - \cos^{-1}\left(\frac{x_i^2 + y_i^2 + z_i^2 - l_1^2 + l_2^2}{2l_1\sqrt{x_i^2 + y_i^2 + z_i^2}}\right), \tag{17}$$

$$\beta_i = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x_i^2 - y_i^2 - z_i^2}{2l_1 l_2}\right), \tag{18}$$

$$\gamma_i = \sin^{-1}\left(\frac{z_i}{\sqrt{x_i^2 + z_i^2}}\right). \tag{19}$$

Je nutné si uvědomit, že tyto body musí být dosažitelné, musí vyhovovat podmínce:

$$\sqrt{x_i^2 + y_i^2 + z_i^2} \leq l_1 + l_2. \tag{20}$$

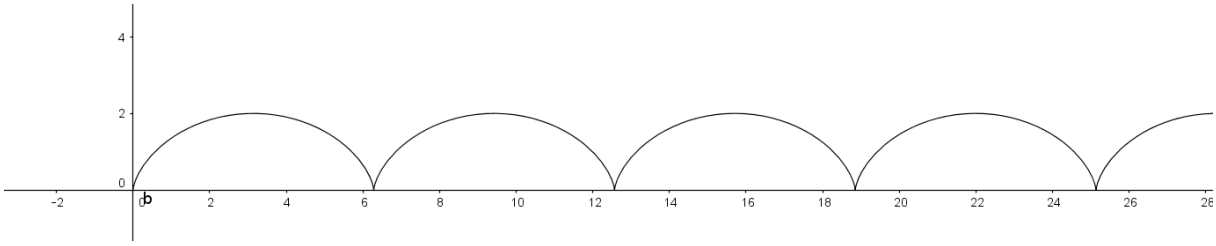
4.4 Kinematika

V nabídce pohybových módů jsou nejdůležitější pohyby translační a rotační (kolem středové osy). Jsou zde k nalezení pod názvy *Walking* a *Turning* (viz první oddíl této práce).

4.4.1 WALKING

Než se pustíme do samotného problému translačního pohybu, popíšeme elegantní způsob generování křivek [11]. Tímto způsobem je odvalování pohyblivé (generující) křivky po stacionární (vodící) křivce. Rozdělme vodící křivku body $X_1, X_2 \ldots, X_n$ a generující křivku body $Y_1, Y_2 \ldots, Y_n$ tak, že délka oblouku na vodící křivce s hraničními body X_i a X_{i+1}, je stejná jako délka oblouku na generující křivce ohraničeného body Y_i a Y_{i+1}, pro všechna $i \in \{1, 2, \ldots, n\}$. Nechť na začátku pohybu bod X_1 splývá s bodem Y_1 (křivky mají v tomto bodě společný dotyk). Odvalováním generující křivky po vodící křivce rozumíme pohyb, kdy postupně splývá bod Y_i s X_i, $i = 1, 2, \ldots, n$.

Jako optimální trajektorie jednotlivých končetin pro přímočarou chůzi se nám jeví cykloida.



Obr. 26: Cykloida

Cykloida je taková křivka, kterou opisuje bod pevně spjatý s kružnicí o poloměru r odvalující se po přímce [11]. V praxi to znamená translaci kružnice $(-r\sin(t), -r\cos(t), 0)^T$ podle vektoru $(rt, r, 0)^T$:

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & rt \\ 0 & 1 & 0 & r \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -r\sin(t) \\ -r\cos(t) \\ 0 \\ 1 \end{pmatrix}.$$

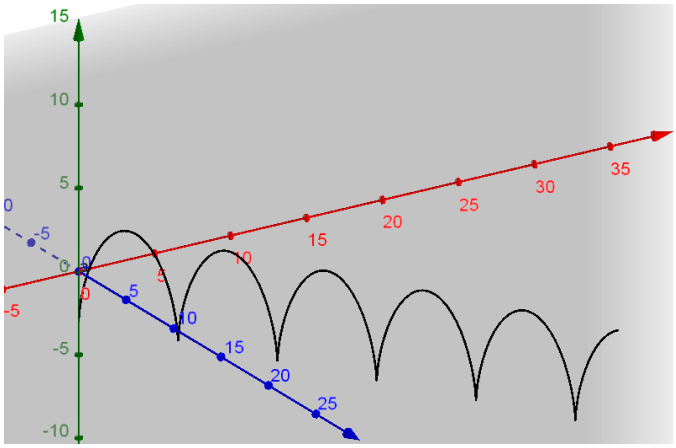
Po vynásobení dostaneme:

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \\ 1 \end{pmatrix} = \begin{pmatrix} r(t - \sin(t)) \\ r(1 - \cos(t)) \\ 0 \\ 1 \end{pmatrix}. \tag{21}$$

V současné době je použita modifikovaná cykloida (trajektorie pro jednotlivé končetiny jsou vůči sobě fázově posunuty):

$$\begin{pmatrix} x_W(t) \\ y_W(t) \\ z_W(t) \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ -b\cos(t) \\ t - a\sin(t) \\ 1 \end{pmatrix}. \tag{22}$$

V našem modelu je tato křivka ještě dále upravena, na y-ovou složku je aplikována funkce Mathf.Clamp$(y_W(t), 0, b)$ [12], která vrací hodnoty z intervalu $\langle 0, b \rangle$. Úhel θ upravuje směr chodu hexapoda. Jedna z konkrétních trajektorií je patrná z obrázku Obr. 27 (konkrétní hodnoty parametrů a a b jsou opět k nalezení v Tab. 3).



Obr. 27: Modifikovaná cykloida bez použití funkce Mathf.Clamp$(y_W(t), 0, b)$

Pro modelování dráhy v mapovém editoru je potřeba znát střední hodnotu rychlosti tohoto translačního pohybu. Okamžitá velikost této rychlosti je $v \approx |a\cos(t)|$. Celý pohyb konce končetiny se dá relativně dobře přiblížit jako pohyb po elipse:

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -b\cos(t) \\ -a\sin(t) \\ 1 \end{pmatrix}.$$

Vektor rychlosti konce končetiny by vypadal takto:

$$\boldsymbol{v} = \left(\frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t}, \frac{\mathrm{d}z}{\mathrm{d}t}\right)^T = \begin{pmatrix} 0 \\ b\sin(t) \\ -a\cos(t) \end{pmatrix}.$$

Rychlost hexapoda vzhledem k podložce (S_W) je rovna velikosti projekce tohoto vektoru do roviny podložky, což je výše uvedený vztah pro okamžitou rychlost. Střední hodnota potom je:

$$\langle v\rangle = \frac{1}{2\pi}\int_0^{2\pi} |a\cos(t)|\mathrm{d}t.$$

Po integraci ($a \geq 0$):

$$\langle v\rangle = \frac{2a}{\pi}. \tag{23}$$

Tento odhad (číselná hodnota je uvedena v Tab. 3) se shoduje s měřením, a proto je relevantní pro mapový editor.

4.4.2 TURNING

Dalším důležitým pohybem je rotace kolem středové osy hexapoda. Jednotlivé končetiny se pohybují po elipsách (parametry viz Tab. 3):

$$\begin{pmatrix} x_W(t) \\ y_W(t) \\ z_W(t) \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -b\cos(t) \\ -a\sin(t) \\ 1 \end{pmatrix}. \tag{24}$$

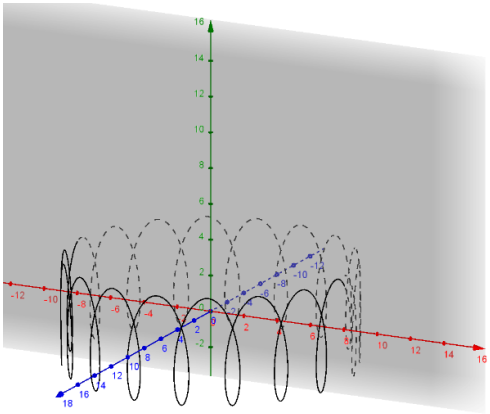
V programu je opět užita funkce Mathf.Clamp$(y_W(t), 0, b)$ [12]. Elipsy jsou natočeny v rotaci $R_y(\eta_y)$ (první elipsa je otočena o $\pi/3$ rad, druhá o 0 rad, třetí o $-\pi/3$ rad atd., vždy se odečte $\pi/3$ rad) a posunuty ve směru paprsků o poloměr rotace R (viz tab. 3). Trajektorie končetiny odpovídá této křivce:

$$\begin{pmatrix} x_W(t) \\ y_W(t) \\ z_W(t) \end{pmatrix} = \begin{pmatrix} \cos(\langle\Omega\rangle t) & 0 & -\sin(\langle\Omega\rangle t) \\ 0 & 1 & 0 \\ \sin(\langle\Omega\rangle t) & 0 & \cos(\langle\Omega\rangle t) \end{pmatrix} \begin{pmatrix} R \\ -b\cos(t) \\ -a\sin(t) \end{pmatrix},$$

Po vynásobení:

$$\begin{pmatrix} x_W(t) \\ y_W(t) \\ z_W(t) \end{pmatrix} = \begin{pmatrix} R\cos(\langle\Omega\rangle t) + a\sin(\langle\Omega\rangle t)\sin(t) \\ -b\cos(t) \\ R\sin(\langle\Omega\rangle t) - a\sin(t)\cos(\langle\Omega\rangle t) \end{pmatrix}, \tag{25}$$

kde $\langle\Omega\rangle$ je střední hodnota úhlové rychlosti rotace hexapoda. Hodnota $\langle\Omega\rangle$ bude použita v mapovém editoru.



Obr. 28: Trajektorie končetin při rotaci na místě bez použití funkce Mathf.Clamp$(y_W(t), 0, b)$

Platí (což se dá opět odvodit z velikosti projekce vektoru rychlosti do podložky):

$$\Omega R \approx a|\cos(t)|.$$

Přibližnou hodnotu střední hodnoty úhlové rychlosti určíme takto:

$$\langle\Omega\rangle \approx \frac{a}{2\pi R}\int_0^{2\pi} |\cos(t)|\,\mathrm{d}t.$$

Po integraci dostaneme:

$$\langle\Omega\rangle \approx \frac{2a}{\pi R}. \tag{26}$$

Výsledek je pro mapový editor vyhovující. Hodnota $\langle\Omega\rangle$ je zanesena do tabulky Tab. 3.

4.4.3 DALŠÍ POHYBOVÉ MÓDY

V nabídce pohybových módů jsou kromě translačního a rotačního pohybu ještě módy připomínající nativní pohyby. Tyto typy pohybů jsou v seznamu módů k nalezení pod názvy (viz prvý oddíl) *Waving*, *Rotating (X, Z)*, *Rotating (Y)*, *Leaning B/F* a *Up/down*. V následující tabulce jsou uvedeny konkrétní tvary těchto rovnic:

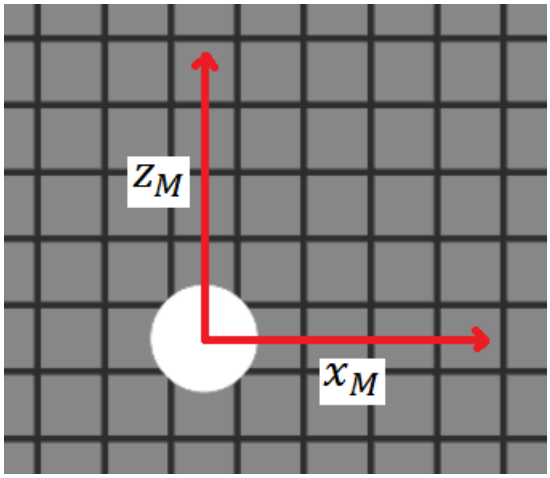
Název pohybu	Polohový vektor končetin vůči světové souř. soustavě	Natočení hexapoda vůči světové souř. soustavě	Natočení servomotorů
Waving	$\begin{pmatrix}0\\2\\0\\1\end{pmatrix}$	$\eta_x = 0, \eta_y = 0, \eta_z = 0$	$\alpha_1 = 40 + 20\lvert\sin(t)\rvert,$ $\beta_1 = -20 + 20\lvert\sin(t)\rvert,$ $\gamma_1 = 30 + 20\sin\left(t + \frac{\pi}{2}\right)$
Rotating (X, Z)	$\begin{pmatrix}0\\2\\0\\1\end{pmatrix}$	$\eta_x = \frac{1}{4}\sin\left(\frac{\pi}{5}t\right), \eta_y = 0, \eta_z = \frac{1}{4}\sin\left(\frac{\pi}{10}t\right)$	Určeny ze vzorců pro úhly natočení servomotorů.
Rotating (Y)	$\begin{pmatrix}0\\2\\0\\1\end{pmatrix}$	$\eta_x = 0, \eta_y = \frac{1}{4}\sin\left(\frac{\pi}{5}t\right), \eta_z = 0$	Určeny ze vzorců pro úhly natočení servomotorů.
Leaning B/F	$\begin{pmatrix}0\\0\\\sin(t)\\1\end{pmatrix}$	$\eta_x = 0, \eta_y = 0, \eta_z = 0$	Určeny ze vzorců pro úhly natočení servomotorů.
Up/down	$\begin{pmatrix}0\\1+2\sin(t)\\0\\1\end{pmatrix}$	$\eta_x = 0, \eta_y = 0, \eta_z = 0$	Určeny ze vzorců pro úhly natočení servomotorů.

Tab. 2: Charakteristiky jednotlivých pohybových módů

4.5 Mapový editor

Mapový editor slouží k vytváření a zadávání trajektorie ve formě lomené čáry. Jeho hlavní částí je čtvercová síť typu $(2k+1)\times(2l+1)$. Tato síť reprezentuje světovou souřadnicovou soustavu (přesněji rovinu $z_W x_W$, osa y_W míří nad nákresnu). Zavedli jsme zde novou

souřadnicovou soustavu S_M, $x_M \in \{-k, -k+1, \dots, k-1, k\}$, $z_M \in \{-l, -l+1, \dots, l-1, l\}$ (viz obr. 29).



Obr. 29: Reprezentace světové souřadnicové soustavy v mapovém editoru

Reálné souřadnice $(x_W;\ z_W)$ ve světové souřadnicové soustavě dostaneme součinem příslušných souřadnic $(x_M;\ z_M)$ se škálovacím faktorem ω.

Vstupem je posloupnost bodů:

$$P_0, P_1, P_2, \dots, P_{n-1}, P_n.$$

Každé dva sousední body $P_i[x_{Mi};\ z_{Mi}]$ a $P_{i+1}[x_{Mi+1};\ z_{Mi+1}]$ definují směrový vektor $\boldsymbol{s}_i = P_{i+1} - P_i$, $i = 0, 1, 2 \dots, n$. Vzdálenost, kterou má hexapod mezi body P_i a P_{i+1} urazit je rovna $\|\boldsymbol{s}_i\|$. Střední hodnotu rychlosti translačního pohybu jsme již dříve určili. Pro dobu t_i^T trvání translace můžeme psát:

$$t_i^T = \omega \frac{\|\boldsymbol{s}_i\|}{\langle v \rangle}. \tag{27}$$

Pokud se hexapod dostane z bodu P_{i-1} do bodu P_i, automaticky se nasměruje do bodu P_{i+1}. Nechť je:

$$\begin{aligned} \delta_i = \tan^{-1}2[\omega(x_{Mi} - x_{Mi-1}), \omega(z_{Mi} - z_{Mi-1})] \\ - \tan^{-1}2[\omega(x_{Mi+1} - x_{Mi}), \omega(z_{Mi+1} - z_{Mi})], \end{aligned} \tag{28}$$

kde funkce $\tan^{-1}2(x, y)$: $\mathbb{R}\times\mathbb{R} \to (-\pi;\ \pi\rangle$ je definována takto [13]:

$$\tan^{-1}2(x,y) = \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) \text{ pokud } x > 0 \\ \tan^{-1}\left(\frac{y}{x}\right) + \pi \text{ pokud } x < 0 \wedge y \geq 0 \\ \tan^{-1}\left(\frac{y}{x}\right) - \pi \text{ pokud } x < 0 \wedge y < 0 \\ \frac{\pi}{2} \text{ pokud } x = 0 \wedge y > 0 \\ -\frac{\pi}{2} \text{ pokud } x = 0 \wedge y < 0 \\ \text{nedef. pokud } x = 0 \wedge y = 0 \end{cases} \tag{29}$$

Pokud je $|\delta_i|$ nenulové číslo menší než π rad, položíme $\langle\Omega\rangle t_i^R = \delta_i$. Z toho potom za použití střední hodnoty úhlové rychlosti dostaneme trvání rotace t_i^R (uvažujeme i zápornou hodnotu času, rotace potom bude probíhat v opačném smyslu):

$$t_i^R = \frac{\delta_i}{\langle\Omega\rangle}. \tag{30}$$

Pokud je $|\delta_i| > \pi$ rad, bude $\langle\Omega\rangle t_i = 2\pi - |\delta_i|$. Pro trvání rotace potom dostaneme:

$$t_i^R = \frac{2\pi - |\delta_i|}{\langle\Omega\rangle}. \tag{31}$$

Posloupnost bodů $\{P_j\}_{j=0}^{n+1}$ jsme tak převedli na posloupnost uspořádaných dvojic reálných čísel $\{[t_j^R;\ t_j^T]\}_{j=0}^{n}$. Zajímavým údajem je celková doba trvání pohybu hexapoda t_c:

$$t_c = \sum_{j=0}^{n} (t_j^R + t_j^T). \tag{32}$$

4.6 Připravovaná vylepšení

Dalším cílem projektu je navrhnout nový mapový editor na bázi Bézierových křivek. V této kapitole nastíníme jeden z možných principů, na kterém by mohl být editor založen. Je možné, že výsledný produkt se bude v některých detailech od tohoto lišit.

Bézierovy křivky byly přímo navrženy pro počítačové modelování [14]. Základem editoru bude opět čtvercová síť reprezentující rovinu $z_W x_W$ ve světové souřadnicové soustavě (budeme se držet stejného označení a opět zavedeme S_M, index M však u souřadnic uvádět nebudeme). Křivka je určena řídícím polygonem zadaným posloupností bodů:

$$P_{00}, P_{10}, P_{20}, \dots, P_{(n-1)0}, P_{n0}.$$

Na každou úsečku tohoto polygonu umístíme vždy jeden bod. Tak vznikne nová sada bodů:

$$P_{01}(\tau), P_{11}(\tau), P_{21}(\tau), \dots, P_{(n-1)1}(\tau),$$

kde $\tau \in \langle 0; 1\rangle$ je parametr. Můžeme psát:

$$P_{k1}(\tau) = (1-\tau)P_{k0} + \tau P_{(k+1)0}, k = 0, 1, 2, \dots, n-1,$$

Tyto body opět definují úsečky, na kterých stejným způsobem vygenerujeme další posloupnost bodů:

$$P_{02}(\tau), P_{12}(\tau), P_{22}(\tau), \dots, P_{(n-2)2}(\tau)$$

tak, aby platilo:

$$P_{k2}(\tau) = (1-\tau)P_{k1}(\tau) + \tau P_{(k+1)1}(\tau), k = 0, 1, 2, \dots, n-2, \tau \in \langle 0; 1\rangle.$$

Postupně vytvoříme systém:

$$P_{00}, P_{10}, P_{20}, \dots, P_{(n-1)0}, P_{n0}$$
$$P_{01}(\tau), P_{11}(\tau), P_{21}(\tau), \dots, P_{(n-2)1}(\tau), P_{(n-1)1}(\tau)$$
$$P_{02}(\tau), P_{12}(\tau), P_{22}(\tau), \dots, P_{(n-3)2}(\tau), P_{(n-2)2}(\tau)$$

$$\vdots$$
$$P_{0(n-2)}(\tau), P_{1(n-2)}(\tau), P_{2(n-2)}(\tau)$$
$$P_{0(n-1)}(\tau), P_{1(n-1)}(\tau)$$
$$P_{0n}(\tau).$$

Toto schéma nám dává návod, jak sestavit rekurzivní algoritmus pro generování bodů v jednotlivých řádcích (algoritmus nese jméno svého objevitele de Casteljau algoritmus) [14].

Pokud τ probíhá interval $\langle 0; 1\rangle$, opisuje bod $P_{0n}(\tau)$ (dále tento bod budeme značit $B(\tau)$) Bézierovu křivku. Po postupném dosazení a úpravách bychom dostali [14]:

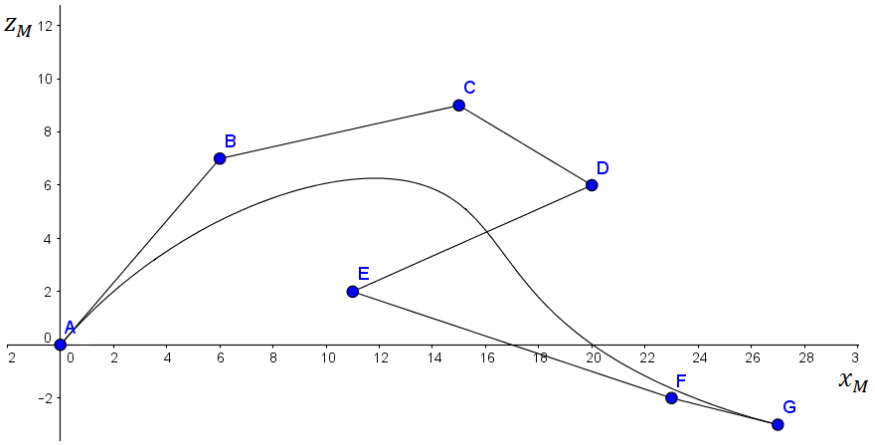
$$B(t) = \sum_{j=0}^{n} B_{nj}(\tau) P_j, \tag{33}$$

kde $B_{nj}(\tau)$ jsou Bernsteinovy bázové polynomy:

$$B_{nj}(\tau) = \binom{n}{j} \tau^j (1-\tau)^{n-j}, j = 0, 1, 2, \dots, n, \tag{34}$$

přičemž se definice doplňuje o případ $B_{nj}(\tau) \coloneqq 0$ pro $j < 0$ nebo $j > n$. Symbol $\binom{n}{j}$ zde značí kombinační číslo:

$$\binom{n}{j} = \frac{n!}{(n-j)!\, j!}.$$



Obr. 30: Bézierova křivka pro $n = 6$

Snadno se dá ukázat, že:

$$\frac{\mathrm{d}B_{nj}}{\mathrm{d}\tau} = n\left(B_{(n-1)(j-1)} - B_{(n-1)j}\right). \tag{35}$$

Více o Bernsteinových bázových polynomech v příloze. Nyní odvodíme některé klíčové vlastnosti těchto křivek. Derivováním $B(\tau)$ podle τ a užitím vztahu (35) dostaneme:

$$\frac{\mathrm{d}B}{\mathrm{d}\tau} = \sum_{j=0}^{n} \frac{\mathrm{d}B_{nj}}{\mathrm{d}\tau} P_j = n \sum_{j=0}^{n} \left(B_{(n-1)(j-1)} - B_{(n-1)j}\right) P_j. \tag{36}$$

Tečný vektor v bodě P_0 má tvar [14]:

$$\left(\frac{\mathrm{d}B}{\mathrm{d}\tau}\right)_{\tau=0} = n(P_1 - P_0). \tag{37}$$

Tečný vektor pro bod P_n je:

$$\left(\frac{\mathrm{d}B}{\mathrm{d}\tau}\right)_{\tau=1} = n(P_n - P_{n-1}). \tag{38}$$

Pro naše účely bude vhodnější použít jistou modifikaci těchto křivek, totiž racionální Bézierovy křivky [14]. Každému bodu P_j, $j = 0, 1, 2, \ldots, n$ přiřadíme váhu $w_j \in \mathbb{R}_0^+$, ovlivňující přimykání křivky k příslušnému bodu. Předpis pro racionální Bézierovu křivku vypadá následovně [14]:

$$B_r(t) = (B_x(\tau), B_z(\tau)) = \frac{\sum_{j=0}^{n} B_{nj}(\tau) w_j P_j}{\sum_{j=0}^{n} B_{nj}(\tau) w_j}. \tag{39}$$

Pro tečné vektory bude potom v bodech P_0 a P_n platit:

$$\left(\frac{\mathrm{d}B_r}{\mathrm{d}\tau}\right)_{\tau=0} = \frac{w_1}{w_0} n(P_1 - P_0), \tag{40}$$

$$\left(\frac{\mathrm{d}B_r}{\mathrm{d}\tau}\right)_{\tau=1} = \frac{w_{n-1}}{w_n} n(P_n - P_{n-1}). \tag{41}$$

Změna parametrů w_0, w_1, w_{n-1} a w_n tedy mění velikost počátečních a koncových tečných vektorů, nikoliv jejich směr.

Nyní přejdeme k samotnému popisu principu editoru. Pro další výklad však musíme zavést pojem ekvidistanty křivky. Dvojicí ekvidistant ke křivce $B_r(\tau)$ budeme rozumět takové křivky $E_1(\tau)$ a $E_2(\tau)$, které v bodech odpovídajících jedné hodnotě parametru mají společnou hlavní normály s $B_r(\tau)$ v příslušném bodě [15]. Parametrické předpisy pro křivky $E_1(\tau)$ a $E_2(\tau)$ jsou:

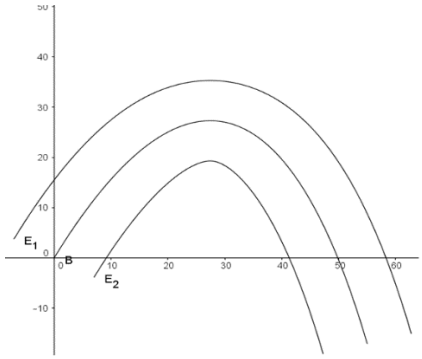
$$E_1(\tau) = B_r(\tau) + \frac{R}{\|\boldsymbol{n}(\tau)\|}\boldsymbol{n}(\tau), \tag{42}$$

$$E_2(\tau) = B_r(\tau) - \frac{R}{\|\boldsymbol{n}(\tau)\|}\boldsymbol{n}(\tau), \tag{43}$$

kde:

$$\boldsymbol{n}(t) = \left(-\frac{\mathrm{d}B_{rz}(\tau)}{\mathrm{d}\tau}, \frac{\mathrm{d}B_{rx}(\tau)}{\mathrm{d}\tau}\right). \tag{44}$$

je vektor hlavní normály. Z rovnic (42) a (43) je patrné, že jsme zvolili vzdálenost ekvidistant rovnou poloměru rotace R.



Obr. 31: Bézierova křiva a její dvě ekvidistanty

Ekvidistanty budou sloužit jako „vodící“ křivky pro končetiny. Dále tyto ekvidistanty rozdělme body $K_l = E_1(l/d)$ a $L_l = E_2(l/d)$, $l = 0, 1, \dots, d$ na jednotlivé oblouky. Číslo d bude určeno tak, aby byla splněna podmínka:

$$\max_{l\in\{0,1,\dots,d\}} \rho(K_l, K_{l+1}) \leq 2a_{\max} \wedge \max_{l\in\{0,1,\dots,d\}} \rho(L_l, L_{l+1}) \leq 2a_{\max}, \tag{45}$$

kde $\rho(.,.)$ značí euklidovskou metriku a hodnota $a_{\max}$ zaručuje dosažitelnost bodů na elipsách (viz níže) a závisí na rozměrech končetin. Každá dvojice po sobě jdoucích bodů definuje úsečku. Označme střed úsečky K_lK_{l+1} jako S_l^K a střed úsečky L_lL_{l+1} jako S_l^L pro $l = 0, 1, \dots, d$. Body S_l^K a S_l^L budou současně sloužit jako středy elips, po kterých se budou končetiny pohybovat. Posledním krokem je umístění elips tak, že každá z úseček K_lK_{l+1} a L_lL_{l+1} bude tvořit nosnou úsečku hlavních poloos jedné z elips (které budou posazeny kolmo k podložce). Délka hlavní poloosy takto umístěné elipsy nad úsečku K_lK_{l+1}, respektive L_lL_{l+1} bude rovna $\rho(K_l, K_{l+1})/2$, respektive $\rho(L_l, L_{l+1})/2$ pro $l = 0, 1, \dots, d$. Parametrické rovnice výsledných elips pro E_1 budou:

$$\begin{pmatrix} x_l(t) \\ y_l(t) \\ z_l(t) \\ 1 \end{pmatrix}_{E_1} = \begin{pmatrix} 1 & 0 & 0 & (K_{lx} + K_{(l+1)x})/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & (K_{lz} + K_{(l+1)z})/2 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_y(\varphi_l) \begin{pmatrix} -\rho(K_l, K_{l+1}) \sin(t)/2 \\ -b\cos(t) \\ 0 \\ 1 \end{pmatrix}, \tag{46}$$

kde $\varphi_l = \tan^{-1}2\left(K_{(l+1)x} - K_{lx}, K_{(l+1)z} - K_{lz}\right)$ a $t \in \langle 0; 2\pi)$. Pro ekvidistantu E_2 platí rovnice obdobné (body K_l jsou pouze nahrazeny body L_l). Vzniknou tak dvě množiny elips, každá množina pro jednu ekvidistantu. Z těchto množin budou vybírány trajektorie jednotlivým končetinám v přesně daném sledu vždy po uplynutí jedné periody (obecně vůči sobě fázově posunuté). Jedním z problémů, kterým budeme při vytváření editoru čelit je stanovení vhodných proporcí.

Dalším vylepšením je ukazatel okamžité úhlové rychlosti natočení servomotorů. Tato rychlost je konstrukčně limitována. Pro jednotlivé úhly α_i, β_i a γ_i označme úhlové rychlosti jako Ω_{α_i}, Ω_{β_i} a Ω_{γ_i}. Platí:

$$\Omega_{\alpha_i} = \frac{\mathrm{d}\alpha_i}{\mathrm{d}t}, \tag{47}$$

$$\Omega_{\beta_i} = \frac{\mathrm{d}\beta_i}{\mathrm{d}t}, \tag{48}$$

$$\Omega_{\gamma_i} = \frac{\mathrm{d}\gamma_i}{\mathrm{d}t}. \tag{49}$$

Úhly α_i, β_i a γ_i jsou však funkcemi proměnných x_i, y_i a z_i. Po rozepsání příslušných totálních derivací [15][16] dostaneme:

$$\Omega_{\alpha_i} = \frac{\partial \alpha_i}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}t} + \frac{\partial \alpha_i}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}t} + \frac{\partial \alpha_i}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}t}, \tag{50}$$

$$\Omega_{\beta_i} = \frac{\partial \beta_i}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}t} + \frac{\partial \beta_i}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}t} + \frac{\partial \beta_i}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}t}, \tag{51}$$

$$\Omega_{\gamma_i} = \frac{\partial \gamma_i}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}t} + \frac{\partial \gamma_i}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}t} + \frac{\partial \gamma_i}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}t}, \tag{52}$$

Což může být maticově zapsáno do tvaru:

$$\begin{pmatrix} \Omega_{\alpha_i} \\ \Omega_{\beta_i} \\ \Omega_{\gamma_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial \alpha_i}{\partial x_i} & \frac{\partial \alpha_i}{\partial y_i} & \frac{\partial \alpha_i}{\partial z_i} \\ \frac{\partial \beta_i}{\partial x_i} & \frac{\partial \beta_i}{\partial y_i} & \frac{\partial \beta_i}{\partial z_i} \\ \frac{\partial \gamma_i}{\partial x_i} & \frac{\partial \gamma_i}{\partial y_i} & \frac{\partial \gamma_i}{\partial z_i} \end{pmatrix} \left(\frac{\mathrm{d}x_i}{\mathrm{d}t}, \quad \frac{\mathrm{d}y_i}{\mathrm{d}t}, \quad \frac{\mathrm{d}z_i}{\mathrm{d}t}\right)^T. \tag{53}$$

Matici na pravé straně se říká Jacobiho matice funkcí α_i, β_i a γ_i s proměnnými x_i, y_i a z_i [16][7]. Pro obecné funkce $f_1, f_2, \dots, f_l$ o l reálných proměnných $x_1, x_2, \dots, x_l$ je tato matice definována tímto způsobem[10]:

$$\frac{D(f_1, f_2, \dots, f_l)}{D(x_1, x_2, \dots, x_l)} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_l} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_l} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_l}{\partial x_1} & \frac{\partial f_l}{\partial x_2} & \cdots & \frac{\partial f_l}{\partial x_l} \end{pmatrix}. \tag{54}$$

Potom můžeme rovnici (53) zapsat ve tvaru:

$$\begin{pmatrix} \Omega_{\alpha_i} \\ \Omega_{\beta_i} \\ \Omega_{\gamma_i} \end{pmatrix} = \frac{D(\alpha_i, \beta_i, \gamma_i)}{D(x_i, y_i, z_i)} \left(\frac{\mathrm{d}x_i}{\mathrm{d}t}, \quad \frac{\mathrm{d}y_i}{\mathrm{d}t}, \quad \frac{\mathrm{d}z_i}{\mathrm{d}t}\right)^T. \tag{55}$$

Souřadnice x_i, y_i a z_i jsou však obecně funkce proměnných x_W, y_W a z_W závislých na čase, proto můžeme vztah (55) upravit:

$$\begin{pmatrix} \Omega_{\alpha_i} \\ \Omega_{\beta_i} \\ \Omega_{\gamma_i} \end{pmatrix} = \frac{D(\alpha_i, \beta_i, \gamma_i)}{D(x_i, y_i, z_i)} \frac{D(x_i, y_i, z_i)}{D(x_W, y_W, z_W)} \left(\frac{\mathrm{d}x_W}{\mathrm{d}t}, \quad \frac{\mathrm{d}y_W}{\mathrm{d}t}, \quad \frac{\mathrm{d}z_W}{\mathrm{d}t}\right)^T. \tag{56}$$

[10] V publikaci [7] je symbolem $\frac{D(f_1, f_2, \dots, f_l)}{D(x_1, x_2, \dots, x_l)}$ označen Jacobiho determinant, my jej však použili pro matici.

5 ZÁVĚR

Cílem naší práce bylo vytvořit software na řízení hexapoda. Vytčeného cíle se nám podařilo dosáhnout. Uživatelské rozhraní nabízí možnost manuálního ovládání jednotlivých servomotorů, různé pohybové módy (z nichž nejdůležitějšími jsou módy pohyb přímočarý a otáčení na místě), editor sekvence pohybů umožňující jednotlivé pohyby plynule navázat a mapový editor usnadňující navrhování trajektorie ve formě lomené čáry. V současné době je připravována nová verze mapového editoru umožňující zadávat trajektorie za pomoci vodícího polygonu a ukazatel okamžité úhlové rychlosti jednotlivých servomotorů. Jak již bylo mnohokrát podotknuto, paralelně s vývojem této simulace vznikal i reálný model (který náš vývoj inicioval). Dalším připravovaným vylepšením reálného modelu je nahrazení současného plošného spoje s mikrokontrolérem CC1111 novým plošným spojem s procesorem architektury ARM a přidání senzorů (kompas, ultrazvuk, gyroskop, akcelerometr, příp. magnetometr).

6 REFERENCE

1. *Unity (game engine)*. Na: *Wikipedia: the free encyclopedia* [Online] [cit. 2016-10-07, 17:00 SEČ]. Aktualizováno: 21. 2. 2017, 18:16. Dostupné z: https://en.wikipedia.org/wiki/Unity_(game_engine).

2. *What is Unity?* [online] [cit. 2016-10-07, 18:42 SEČ] Dostupné z: https://unity3d.com/unity.

3. *Unity Technologies Celebrates 5 Years of Unity.* [online] [cit. 2016-10-07, 20:12 SEČ] Dostupné z: ttps://unity3d.com/http%3A//unity3d.com/company/public-relations/news/unity-5year-press.

4. *Unity 1.0 released 10 years ago! Time flies when you're having fun :) #unity3d*. Na: *Twitter* [online] [cit. 2016-10-07, 20:12 SEČ] Dostupné z: https://twitter.com/unity3d/status/607884169528676352?lang=cs.

5. *Roadmap.* [online] [cit. 2016-10-07, 20:52 SEČ] Dostupné z: https://unity3d.com/unity/roadmap.

6. *Get Unity.* [online] [cit. 2016-10-07, 21:12 SEČ] Dostupné z: https://store.unity.com/

7. BEČVÁŘ, Jindřich. *Lineární algebra*. Vyd. 4. Praha: MATFYZPRESS, 2010 ISBN 978-80-7378-135-4.

8. PYTLÍČEK, Jiří. *Lineární algebra a geometrie*. Vyd. 1. Praha: Nakladatelství ČVUT, 2008, ISBN 978-80-01-04063-8.

9. Weisstein, Eric W. *Rotation Matrix.* [Online] [cit. 2016-11-03]. Dostupné z: http://mathworld.wolfram.com/RotationMatrix.html.

10. BLOOMENTHAL, Jules, ROKNE, Jon. *Homogeneous Coordinates*. [Online] [cit. 2016-11-05]. Dostupné z: http://www.unchainedgeometry.com/jbloom/pdf/homog-coords.pdf.

11. Neznámý. *Úvod do kinematické geometrie v rovině.* [Online] [cit. 2016-11-07]. Dostupné z: mathonline.fme.vutbr.cz/download.aspx?id_file=1133.

12. *Mathf.Clamp.* Na: *Unity Scripting API documentation* [Online] [cit. 2016-11-07]. Dostupné z: https://docs.unity3d.com/ScriptReference/Mathf.Clamp.html.

13. atan2. Na: *Wikipedia: the free encyclopedia* [Online] [cit. 2017-01-11, 13:40 SEČ]. Dostupné z: https://en.wikipedia.org/wiki/Atan2.

14. FAROIKI, Rida T. *The Bernstein polynomial basis: a centennial retrospective.* [Online] [cit. 2017-01-20]. Dostupné z: https://pdfs.semanticscholar.org/0ad3/053eb32ecc5a13d9a6aa868b5c86c5663372.pdf.

15. REKTORYS, Karel, kolektiv. *Přehled užité matematiky.* Vyd. 2. Praha: SNTL, 1968.

16. KOPÁČEK, Jiří. *Matematická analýza nejen pro fyziky*. Vyd. 4. Praha: MATFYZPRESS, 2015 ISBN 978-80-7378-282-5.

6.1 Software

Program byl vytvořen v editoru Unity 3D, MonoDevelop. K tvorbě obrázků jsme použili *Microsoft Paint, GeoGebra* a *Inkscape*. Grafy byly vykreslovány v programu *GeoGebra*. Výpočty byly prováděny v *Microsoft Mathematics* a v online verzi *Wolfram alpha*, dostupné zde: https://www.wolframalpha.com/.

7 SEZNAM OBRÁZKŮ A TABULEK

7.1 Obrázky

7.2 Tabulky

8 PŘÍLOHY

8.1 Tabulka použitých parametrů

Název veličiny	Označení veličiny	Hodnota	Jednotka
Individuální natočení souřadnicové soustavy S_1	σ_1	$-\frac{3\pi}{20}$	rad
Individuální natočení souřadnicové soustavy S_2	σ_2	0	rad
Individuální natočení souřadnicové soustavy S_3	σ_3	$\frac{3\pi}{20}$	rad
Individuální natočení souřadnicové soustavy S_4	σ_4	$-\frac{3\pi}{20}$	rad
Individuální natočení souřadnicové soustavy S_5	σ_5	0	rad
Individuální natočení souřadnicové soustavy S_6	σ_6	$\frac{3\pi}{20}$	rad
Délka první části končetiny	l_1	2.5	j
Délka druhé části končetiny	l_2	2.5	j
Hlavní poloosa	a	1	j
Vedlejší poloosa	b	3	j
Střední hodnota translačního pohybu	$\langle v \rangle$	≈ 0.63	$\mathrm{j} \cdot \mathrm{s}^{-1}$
Průměrná hodnota poloměru rotace	R	$7\sqrt{2}$	j
Střední hodnota úhlové rychlosti rotačního pohybu	$\langle \Omega \rangle$	≈ 0.064	$\mathrm{rad} \cdot \mathrm{s}^{-1}$
Škálovací faktor	ω	6.18	j
Maximální vzdálenost sousedních dělících bodů pro obě ekvidistanty	$a_{\max}$	1	j

Tab. 3: Tabulka užívaných parametrů simulace

8.2 Bernsteinovy bázové polynomy

V kapitole o plánovaných vylepšeních jsme se zmínili o Bernsteinových bázových polynomech. V této příloze chceme čtenáře seznámit s jejich některými základními vlastnostmi. Nejprve však zopakujme jejich definici:

$$B_{nj}(\tau) = \binom{n}{j} \tau^j (1-\tau)^{n-j}, j = 0, 1, 2, \dots, n, \qquad \text{(p.1)}$$

kde $\binom{n}{j}$ je kombinační číslo:

$$\binom{n}{j} = \frac{n!}{(n-j)!\,j!}.$$

Definice doplňuje ještě pro případy, kdy $j < 0$ nebo $j > n$, polynomy se pokládají identicky rovny nule. Pro každé n tak máme $n + 1$ polynomů stupně n. Při informatických aplikacích je zajímavý tento rekurentní vztah:

$$B_{(n+1)j}(\tau) = \tau B_{n(j-1)}(\tau) + (1-\tau)B_{nj}(\tau), \qquad \text{(p.2)}$$

kde se začíná s $B_{00}(\tau) = 1$. Důkaz, že definiční vztah tuto formuli splňuje je velmi jednoduchý, stačí si jen uvědomit skutečnost:

$$\binom{n-1}{j} + \binom{n-1}{j-1} = \binom{n}{j}.$$

Jak už název polynomů napovídá, tvoří systém $\{B_{nj}(\tau), j = 0, 1, 2 \dots n\}$ bázi lineárního prostoru všech polynomů stupně n nad $\mathbb{R}$ (dalo by se rozšířit i na $\mathbb{C}$). Každý reálný polynom $p(\tau)$ stupně n můžeme tedy jednoznačně zapsat jako lineární kombinaci těchto polynomů:

$$p(\tau) = \sum_{j=0}^{n} p_j B_{nj}(\tau), \qquad \text{(p.3)}$$

kde $p_j \in \mathbb{R}$ pro $j = 0, 1, 2, \dots, n$. Za pomoci binomické věty se dá dále ukázat, že:

$$\sum_{j=0}^{n} B_{nj}(\tau) = 1 \qquad \text{(p.4)}$$

Jedná se proto o konvexní kombinaci [8]. To pro nás v praxi znamená, že Bézierova křivka leží v konvexním obalu svého vodícího polygonu. Posledním vztahem souvisejícím s Bernsteinovými polynomy je jejich derivace:

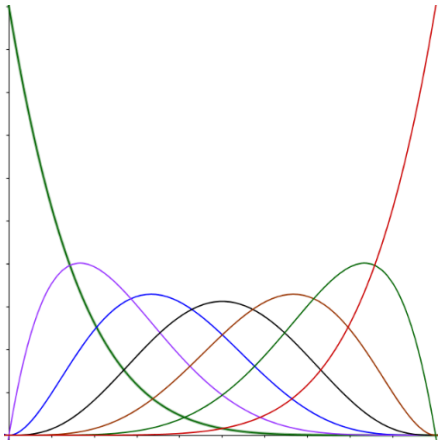
$$\frac{\mathrm{d}B_{nj}}{\mathrm{d}\tau} = n\big(B_{(n-1)(j-1)} - B_{(n-1)j}\big). \qquad \text{(p.5)}$$

Tento vztah je pro nás velmi důležitý, a proto si jej dokážeme:

$$\begin{aligned}
\frac{\mathrm{d}B_{nj}}{\mathrm{d}\tau} &= \binom{n}{j}\frac{\mathrm{d}}{\mathrm{d}\tau}\left[\tau^j(1-\tau)^{n-j}\right] = \binom{n}{j}\left[j\tau^{j-1}(1-\tau)^{n-j} - \tau^j(n-j)(1-\tau)^{n-j-1}\right] \\
&= n\frac{(n-1)!}{(n-j)!\,j!}j\tau^{j-1}(1-\tau)^{n-j} - n\frac{(n-1!)}{(n-j)!\,j!}\tau^j(n-j)(1-\tau)^{n-j-1} \\
&= n\left[\frac{(n-1)!}{(n-j)!\,(j-1)!}\tau^{j-1}(1-\tau)^{n-j} - \frac{(n-1)!}{(n-j-1)!\,j!}\tau^j(1-\tau)^{n-j-1}\right] \\
&= n\left(\binom{n-1}{j-1}\tau^{j-1}(1-\tau)^{n-j} - \binom{n-1}{j}\tau^j(1-\tau)^{n-j-1}\right) \\
&= n\big(B_{(n-1)(j-1)} - B_{(n-1)j}\big).
\end{aligned}$$

Což jsme chtěli dokázat.

V případě zájmu o tuto problematiku vřele doporučujeme monografii [14].



Obr. 32: Bernsteinovy bázové polynomy pro $n = 6$