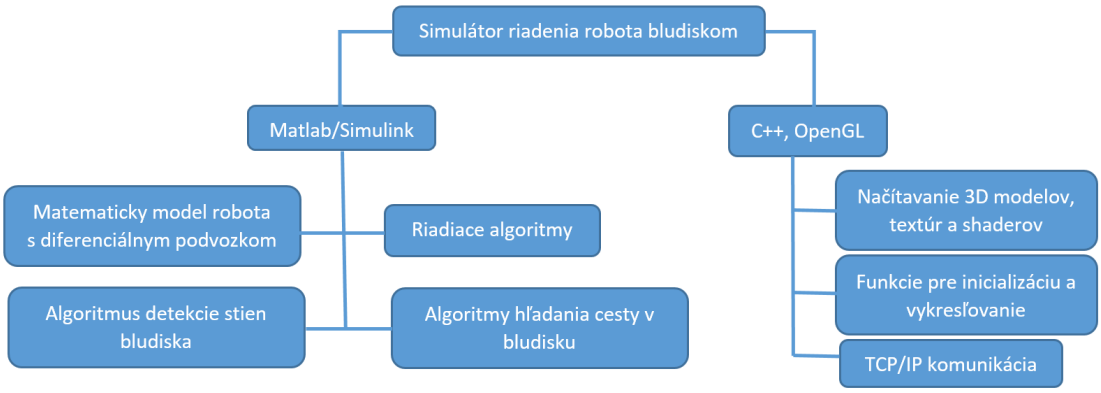
Simulátor riadenia robota v bludisku spolu s vizualizáciou bludiska

Príloha D

1 Úloha

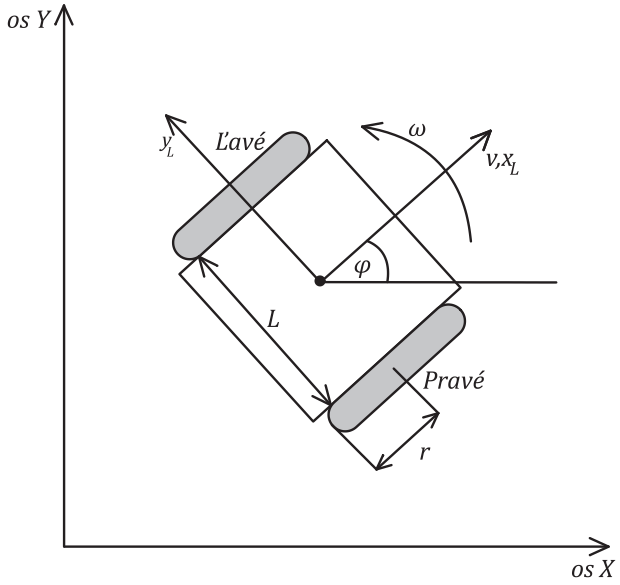
Úlohou je navrhnúť simulátor pre robota s diferenciálnym podvozkom spolu s vizualizáciou. K návrhu je potrebné nainštalovať dva programy a to Matlab/Simulink a Visual studio. V Simulinku je potrebné navrhnúť simulačný model robota, ktorý bude postavený na základe rovníc definovaných v ďalších kapitolách. Okrem riadenia v Simulinku je potrebné navrhnúť riadenie robota, ktoré bude riadiť robota bludiskom. V Matlabe navrhneme funkcie pre detekciu stien bludiska a funkcie, ktoré implementujú algoritmus hľadania cesty v bludisku. Výsledkom bude séria bodov, ktoré treba transformovať do súradníc reálneho sveta. Týmito bodmi budeme riadiť robota pomocou nami navrhnutého riadenia. Následne dáta uložíme do súboru a vizualizujeme robota v nami navrhnutej vizualizácií.

1. Zostavte simulačný model robota s diferenciálnym kolesovým podvozkom
2. Zostavte riadenie robota s diferenciálnym kolesovým podvozkom
3. Naprogramujte funkcie pre detekciu stien bludiska a funkciu pre nájdenie cesty v bludisku
4. Naprogramujte funkcie, ktoré sú potrebné k vizualizácií robota



Obr. 1: Schematické zobrazenie štruktúry aplikácie

2 Zostavenie matematického modelu robota s diferenciálnym kolesovým podvozkom



Obr. 2: Schematické zobrazenie robota s diferenciálnym kolesovým podvozkom

Tabuľka 1: Parametre matematického modelu robota

Parameter	**Označenie**	**Hodnota**	**Jednotka**
Rozchod kolies	L	0.068	m
Polomer kolies	r	0.024	m
Moment zotrvačnosti tela	J_c	0.0004	kgm^2
Moment zotrvačnosti - os kolesa	J_w	0.0001	kgm^2
Moment zotrvačnosti - priemer kolesa	J_m	0.0001	kgm^2
Hmotnosť tela	m_c	0.48	kg
Hmotnosť motora a kolesa	m_w	0.01	kg
Vzdialenosť ťažiska od počiatku x_L, y_L	d	0.03	m
Odpor vinutia kotvy	R_a	3	Ω
Indukčnosť vinutia kotvy	L_a	0.5	H
Zosilnenie	K_b, K_t	0.05	Nm/A
Prevodový pomer	N	1	-

Tabuľka 2: Fyzikálne veličiny matematického modelu robota

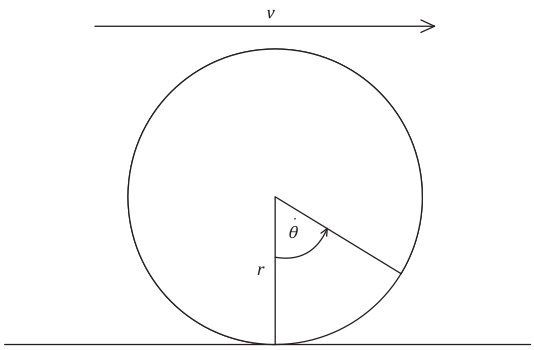
Fyz. veličina	**Označenie**	**Jednotka**
X-ová súradnica polohy robota	x	m
Y-ová súradnica polohy robota	y	m
Uhol natočenia robota	φ	rad
Celková lineárna rýchlosť robota	v	ms^{-1}
Celková uhlová rýchlosť robota	ω	$rads^{-1}$

Matematický model robota s diferenciálnym kolesovým podvozkom sa skladá z kinema-

tického modelu, dynamického modelu pre čiastkové momenty zotrvačnosti a dynamického modelu aktuátorov. Pri modelovaní kinematického modelu robota s diferenciálnym podvozkom budeme vychádzať z abstraktného predpokladu, že máme valec, ktorý sa pohybuje rovno. Smer pohybu do strán je nulový $v_y = 0$. Rýchlosť kotúča v smere osi x sa dá vyjadriť ako:

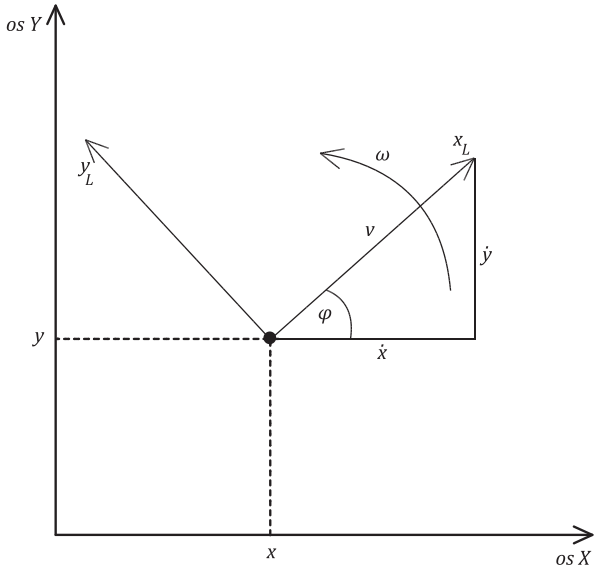
$$v = v_x = r\dot{\theta}, \tag{2.1}$$

kde r je polomer kotúča a $\dot{\theta}$ je uhlová rýchlosť kotúča.



Obr. 3: Lineárna rýchlosť valca

Ak zoberieme do úvahy, že lineárna rýchlosť v je zmena polohy v čase, tak po jednoduchom schematickom zobrazení zmeny polohy valca v rovine, ktorý má svoju x, y polohu, uhol natočenia φ a lineárnu rýchlosť v dokážeme vďaka týmto poznatkom kinematický model jednoducho odvodiť.



Obr. 4: Schematické zobrazenie valca v rovine

V pravouhlom trojuholníku je *cos* definovaný ako priľahla odvesna ku prepone a *sin* ako protiľahlá odvesna ku prepone. Uhlová rýchlosť ω je zmena uhla v čase, preto rovnice kinematického modelu majú tvar:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.2}$$

Následne si definujeme rovnice pre prepočet lineárnej a uhlovej rýchlosti robota v, ω na lineárne rýchlosti pravého a ľavého kolesa v_R, v_L. Rovnice majú tvar

$$\begin{bmatrix} v_R \\ v_L \end{bmatrix} = \begin{bmatrix} 1 & \frac{L}{2} \\ 1 & -\frac{L}{2} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.3}$$

Rovnice pre prepočet lineárnych rýchlostí pravého a ľavého kolesa robota v_R, v_L na uhlové rýchlosti jednotlivých kolies ω_R, ω_L majú tvar

$$\begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & 0 \\ 0 & \frac{1}{r} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix}, \tag{2.4}$$

Rovnice budú potrebné pri počte uhlových rýchlosti pravého a ľavého kolesa ω_R, ω_L na celkovú lineárnu a uhlovú rýchlosť robota v, ω a naopak.

Pomocou dynamiky sa snažíme priblížiť k reálnemu správaniu sa dvojkolesového robota v rovine. V tomto dynamickom modeli berieme do úvahy, že jednotlivé časti robota majú svoju hmotnosť a moment zotrvačnosti spolu s posunom ťažiska od stredového bodu robota na ktorý ho riadime. Rovnice dynamického modelu majú tvar

$$(m + \frac{2J_w}{r^2})\dot{v} - m_c d\omega^2 = \frac{1}{r}(\tau_R + \tau_L) \tag{2.5}$$

$$(J + \frac{L^2}{2r^2}J_w)\dot{\omega} + m_c d\omega v = \frac{L}{2r}(\tau_R - \tau_L) \tag{2.6}$$

Celková hmotnosť robota m a moment zotrvačnosti J sa počítajú na základe rovníc v tvare

$$m = m_c + 2m_w, \tag{2.7}$$

$$J = m_c d^2 + m_w \frac{L^2}{2} + J_c + 2J_m, \tag{2.8}$$

Náš matematický model okrem iného zohľadňuje aj dynamiku aktuátorov. Robot s diferenciálnym podvozkom obsahuje dva motory. Motor je riadený napájacím napätím, kde výsledkom je točivý moment kolies. Rovnice motora majú tvar

$$u_a = R_a i_a + L_a \frac{di_a}{dt} + u_e \tag{2.9}$$

$$u_e = K_b \omega_m \tag{2.10}$$

$$\tau_m = K_t i_a \tag{2.11}$$

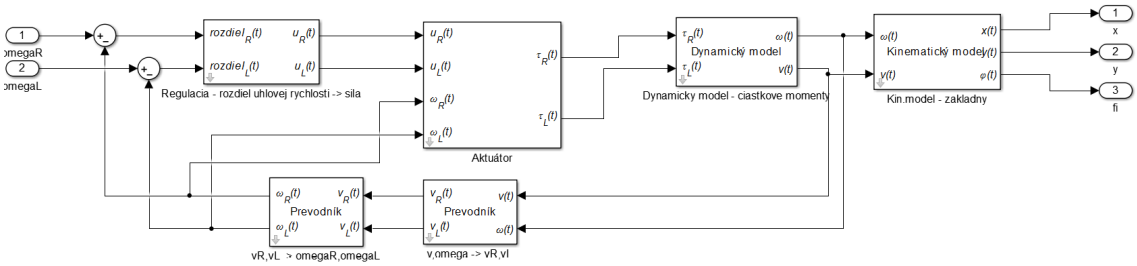
$$\tau = N \tau_m \tag{2.12}$$

Ďalej k vytvoreniu matematického modelu bude potrebné pridať spätno - väzobnú slučku pre potlačenie dynamiky, kde sa na základe aktuálnej uhlovej rýchlosti pravého a ľavého kolesa ω_R, ω_L a požadovanej uhlovej rýchlosti pravého a ľavého kolesa ω_{Rp}, ω_{Lp} vypočíta napätie U_R, U_L, ktoré je potrebné na vygenerovanie točivého momentu motora. Rovnice majú tvar

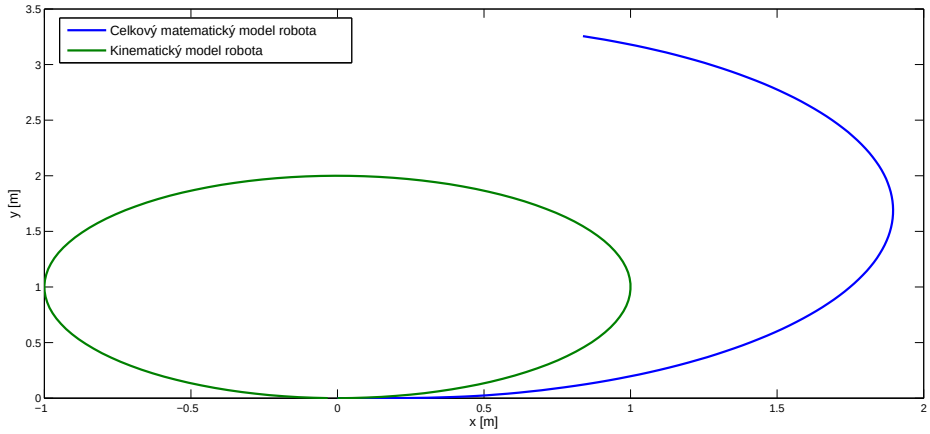
$$U_{R/L} = P(\omega_{Rref/Lref} - \omega_{R/L}) \; pre \; |P(\omega_{Rref/Lref} - \omega_{R/L})| < U_{max}, \tag{2.13}$$

$$U_{R/L} = U_{max} sign(\omega_{Rref/Lref} - \omega_{R/L}) \; pre \; |P(\omega_{Rref/Lref} - \omega_{R/L})| \geq U_{max}, \tag{2.14}$$

Výsledkom je matematický model robota s diferenciálnym podvozkom, ktorý je potrebné následne implementovať v simulačnom prostredí Simulink. Simulačná schéma je zobrazená na obr. 5.



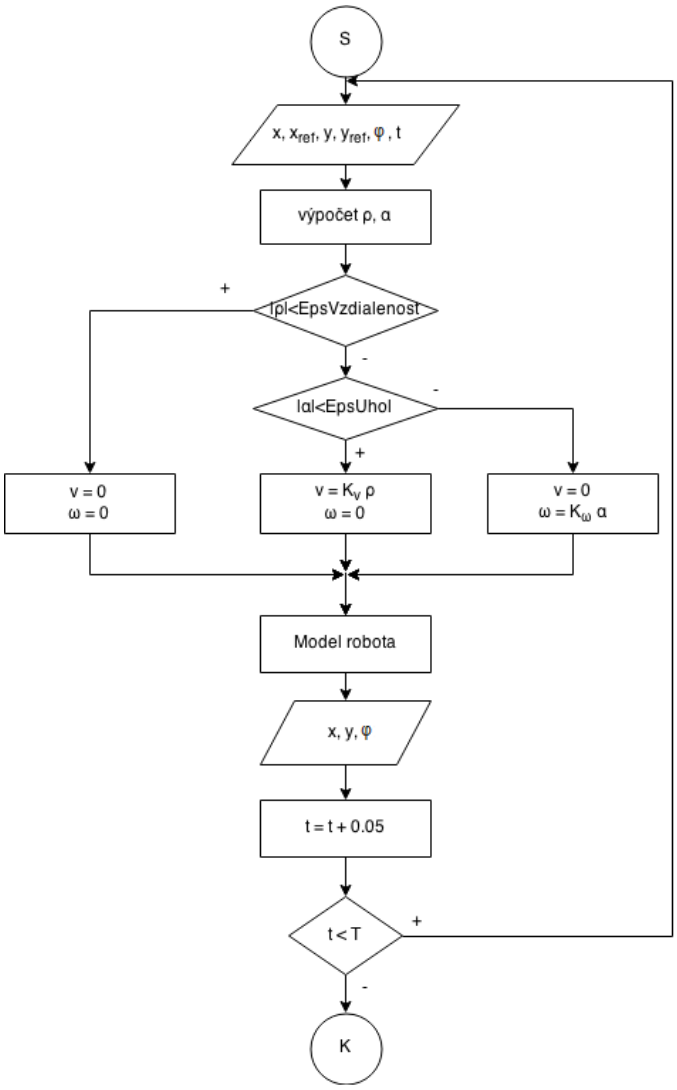
Obr. 5: Simulačný model robota s diferenciálnym kolesovým podvozkom



Obr. 6: Porovnanie kinematického a celkového matematického modelu robota s diferenciálnym podvozkom

3 Algoritmus riadenia robota bludiskom

Algoritmom riadenia robota bludiskom je prepínacie riadenie robota do bodu s presnosťou *EpsUhol*. Po natočení robota k bodu s určitou presnosťou *EpsUhol* začne na robota pôsobiť celková lineárna rýchlosť v a robot sa začne k bodu približovať. Riadenie sa nazýva prepínacie z dôvodu, že na výstupe z algoritmu pôsobí len lineárna v alebo uhlová rýchlosť ω, nikdy nie oboje súčasne. Vývojový diagram algoritmu riadenia robota do bodu zobrazený na nasledujúcom obrázku.

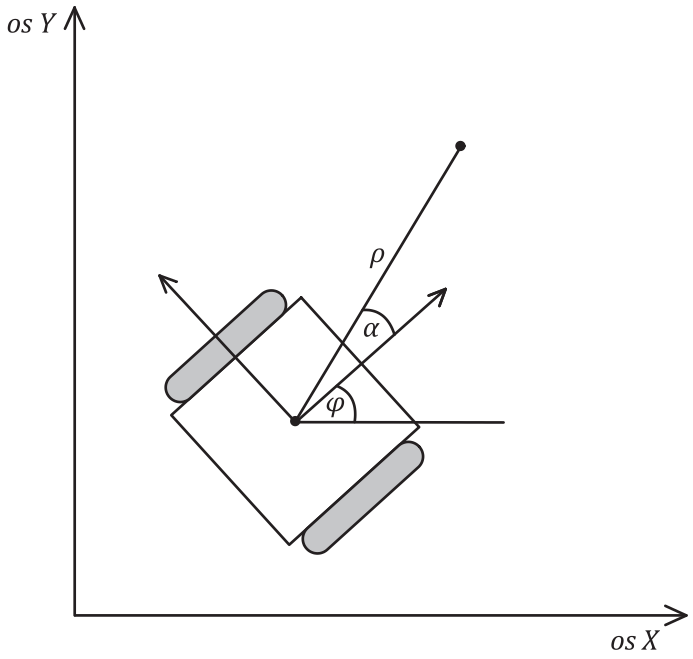


Obr. 7: Vývojový diagram algoritmu riadenia

Vzdialenosť robota ρ od bodu bude počítaná pomocou euklidovskej vzdialenosti

$$\rho = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2}, \tag{3.1}$$

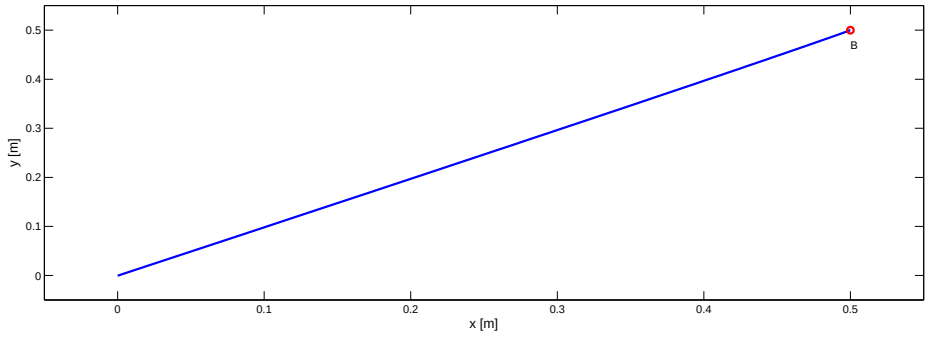
kde x_{ref}, y_{ref} sú súradnice polohy bodu a x, y sú súradnice polohy robota.



Obr. 8: Schematické zobrazenie uhla α

Uhol medzi priamkou vzdialenosti a vodorovnou osou x označíme α a vypočítame na základe rovnice

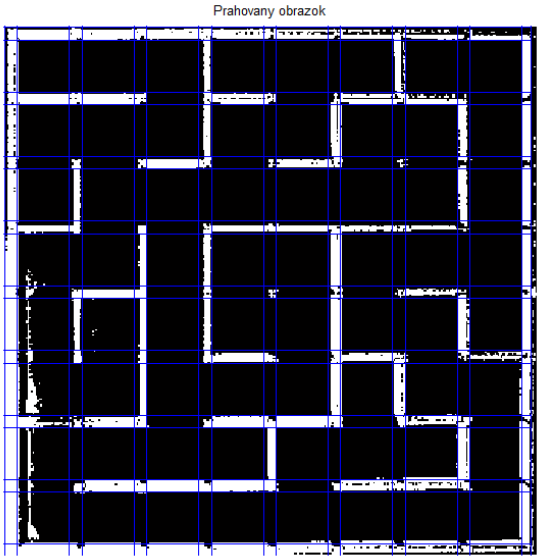
$$\alpha = \tan 2^{-1} \frac{(y_{ref} - y)}{(x_{ref} - x)} \tag{3.2}$$



Obr. 9: Prejdená dráha robota do bodu [0.5 0.5]

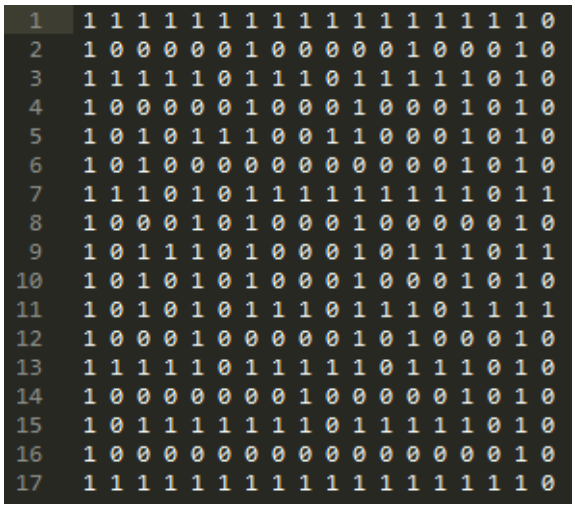
4 Algoritmus detekcie stien bludiska

V našom simulátore sme štruktúru bludiska generovali na základe fotky odfotenej z modelu reálneho bludiska postaveného v labe. Snažíme sa vyhotoviť čo najlepší obrázok s konštantným osvetlením. Následne obrázok prahujeme, ak jas pixelu je väčší ako hodnota prahu, tak pixel nastavíme na "1ïnač na "0". Výsledkom je binárny obrázok, ktorý prerozdelíme na segmenty na základe sumy pixelov. Ak napr. 70 percent pixelov segmentu je "1", tak sa jedná o stenu a naopak o voľné miesto.



Obr. 10: Rozdelenie obrázku na segmenty

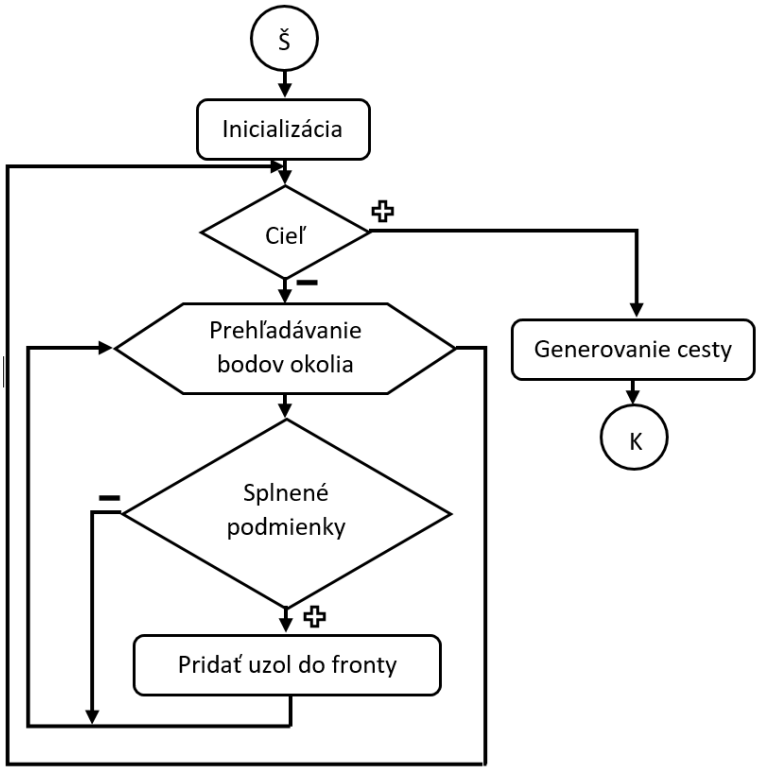
Výsledkom je matica *nxn*, ktorá reprezentuje bludisko a bude využívaná pri algoritmoch hľadania cesty bludiskom.



Obr. 11: Reprezentácia bludiska

5 Algoritmus hľadania cesty v bludisku

Po rozpoznaní štruktúry bludiska je potrebné nájsť cestu z bodu A do bodu B, ktorú prejde robot. Na vyriešenie tohto problému si pomôžeme teóriou grafov a algoritmami pre prehľadávanie grafu. Je potrebné si naštudovať tematiku prehľadávania grafov a vybrať si jeden z algoritmov. V našom tutorialy sme sa rozhodli pre algoritmus Breadth first search, pretože je jednoduchý a ľahký na implementáciu.



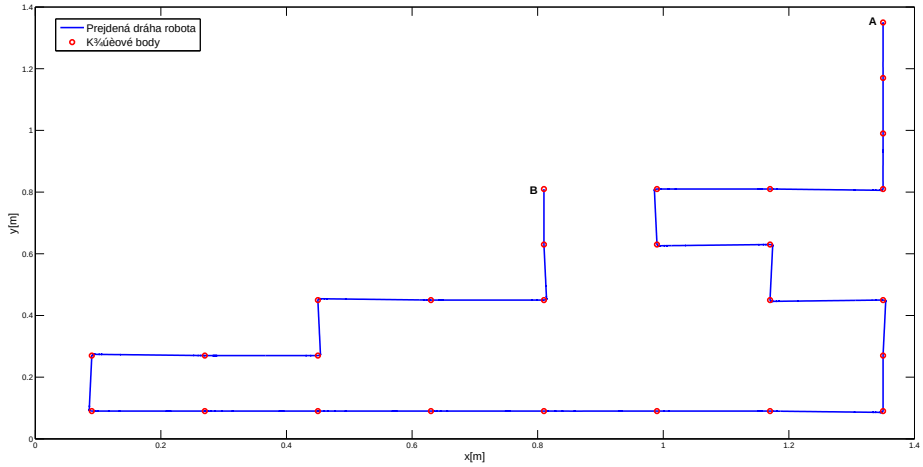
Obr. 12: Vývojový diagram algoritmu BFS

Algoritmus BFS prechádza všetky uzly grafu(v našom prípade sú to súradnice bodov bludiska), pričom si poznamenáva predchodcov jednotlivých uzlov, čím je vytvorený strom najkratších ciest k požadovanému uzlu. Ak prehľadávaný bod grafu je bodom z bludiska, ktorý je označený ako cieľový algoritmus skončí s prehľadávaním a vygeneruje cestu od koreňového uzla (bod bludiska, kde začína robot) až do cieľového bodu. Algoritmus z aktuálneho bodu prehľadáva okolité body grafu a ak sú splnené tieto podmienky, tak je uzol pridaný do fronty. Informácie, ktoré si uzol nesie sú x, y súradnica bodu bludisku spolu s poradovým číslom uzla z ktorého vychádza z vyššej úrovne. Podmienky pridania uzla do fronty sú:

1. Ak x a y súradnica bodu prehľadávaného v grafe sa nachádza vo vnútri bludiska (ohraničenie aby sa neprehľadávali súradnice mimo bludiska).
2. Ak x a y súradnica bodu prehľadávaného v grafe nie je rovnaká ako je aktuálny bod.
3. Ak x a y súradnica bodu prehľadávaného v grafe nie je označená ako stena bludiska.

Po skončení prehľadávania sa prechádza fronta od posledného uzla fronty. Algoritmus sa pozrie na poradové číslo bodu z ktorého vychádzal a odpamätá si x a y súradnicu

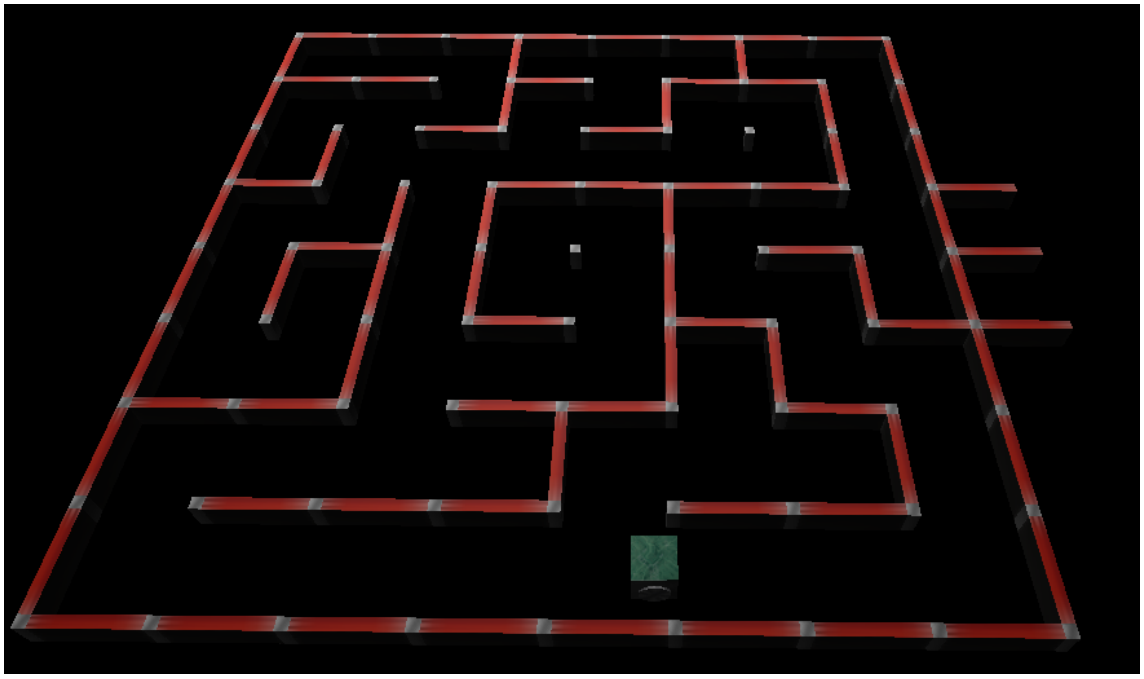
bodu bludiska a skočí na uzol vo fronte s daným poradovým číslom. Výsledkom algoritmu je vektor x a y súradníc cesty robota bludiskom. Následne je potrebné tieto hodnoty transformovať do súradníc do reálneho sveta.



Obr. 13: Výsledok prepínacieho riadenia robota do bodu pri riadení cez bludisko

6 Vizualizácia mobilného robota

K 3D vizualizácií mobilného robota sme sa rozhodli pre API OpenGL. Všetky funckie, ktoré budu spomínané sú bližšie popísane v systémovej príručke mojej diplomovej práce. Okrem prečítania diplomovej práce odporúčam si prečítať knihu OpenGL Pruvodce programátora. V prvom rade je potrebné vytvoriť 3D modely časti robota a bludiska v modelovacom nástroji ako napr. Blender, 3Ds max atď. Následne je potrebné vytvoriť funkcie, kde sa inicializujú buffery pre vertexy, uv súradnice a normály. Ďalej je potrebné vytvoriť funkciu na vykreslenie týchto vertexov na monitore spolu s patričnou textúrou. Tieto buffery však treba naplniť. V našej diplomovej práci sme 3D modely vytvorili v modelovacom nástroji Blender a modely sme vyexportovali do formátu obj. Preto je potrebné vytvoriť funkciu na načítanie modelu napr. vo formáte obj, ktoré uložia jednotlivé dáta do bufferov. Každý objekt ma však svoju textúru, ktorú je potrebné načítať. V našom programe sme implementovali funkciu pre načítanie obrázku vo formáte BMP. V 3D svete máme pohyblivú kameru. Polohu tejto kamery treba prepočítavať na základe vstupov. Naším vstupom bolo stlačenie šípok klávesnice pre pohyb kamery shit pre priblíženie a ctrl pre oddialenie, preto sme si vytvorili funkciu na prepočet projekčnej matice a matice pohľadu. Okrem iného potrebujeme načítať aj shadere, čo sú programy, ktoré pracujú s jednotlivými vertexami. Najčastejšie operácie sú transformácie vrchola, poprípade aplikácia textúry, k danému vrcholu. Preto sme vytvorili funkciu pre načítanie vertex a fragment shaderu. Pre naprogramovanie shaderu sme použili shader jazyk GLSL. Všetko už máme pripravené a v poslednom rade treba vytvoriť funkcie pre načítanie dát zo simulácie ako sú poloha robota x, y, uhol natočenia robota φ a uhly kolies φ_R a φ_L, spolu so štruktúrou bludiska.



Obr. 14: Vizualizácia mobilného robota prechádzajúceho bludiskom

Na záver je potrebné jednotlivé funkcie použiť v nasledujúcej postupnosti. V prvom kroku si vytvoríme objekt, ktorý reprezentuje aplikačné okno, na ktorom sa nám bude vykresľovať robot. Následne načítame vertex a fragment shader pomocou vytvorenej funkcie. Ďalším krokom je vytvorenie matíc modelu, pohľadu, projekcie, ktoré využijeme na posun a rotáciu 3D modelov vo svete. Následne načítame 3D modely spolu s textúrami jednotlivých časti robota spolu s časťami bludiska a dáta o polohe a natočení robota v priestore. Následne v nekonečnom cykle vykresľujeme hlavnú scénu a to bludisko postavené na základe načítanej matice zo simulácie, spolu s celým robotom, ktorý je postupne posúvaný a rotovaný podľa údajov zo simulácie. V závere cyklu je časovač, ktorý počíta ako dlho trval cyklus spolu s delayom aby sme zaistili požadované FPS.