**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

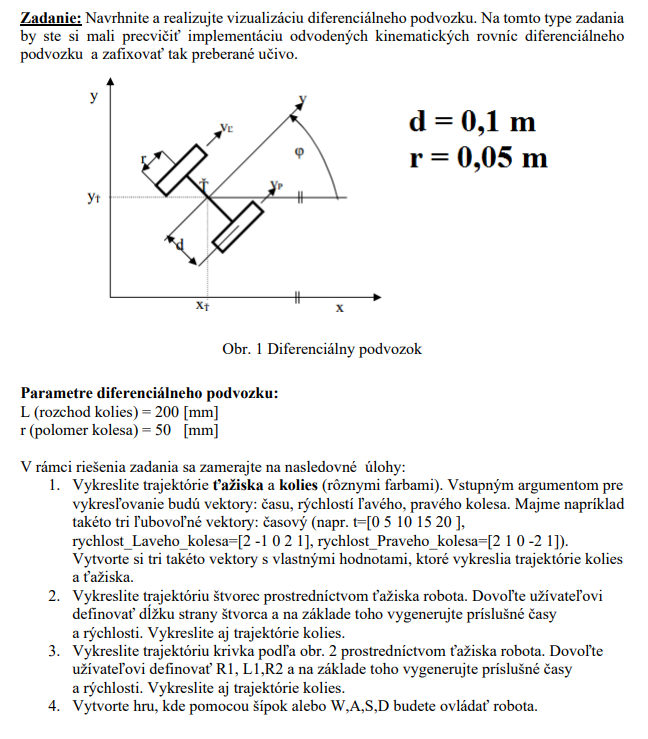
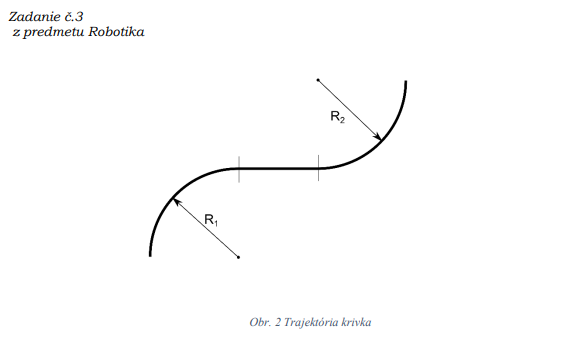
**FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

**MOBILNÝ KOLESOVÝ ROBOT**

**Robotika**

**3 ZADANIE**

**2022 Daniel Fundárek**

**** ****

**Riešenie:**

Na vykreslenie trajektórií ťažiska a kolies mobilného robota s diferenciálnym podvozkom sme implementovali kinematické rovnice diferenciálneho podvozku(1.1 – 1.3) v rámci programovacieho jazyka C# .NET.

(1.1)

(1.2)

(1.3)

Z uhlovej a lineárnej rýchlosti vieme následne odvodiť uhol natočenia robota(1.4) a jeho polohu(1.5, 1.6).

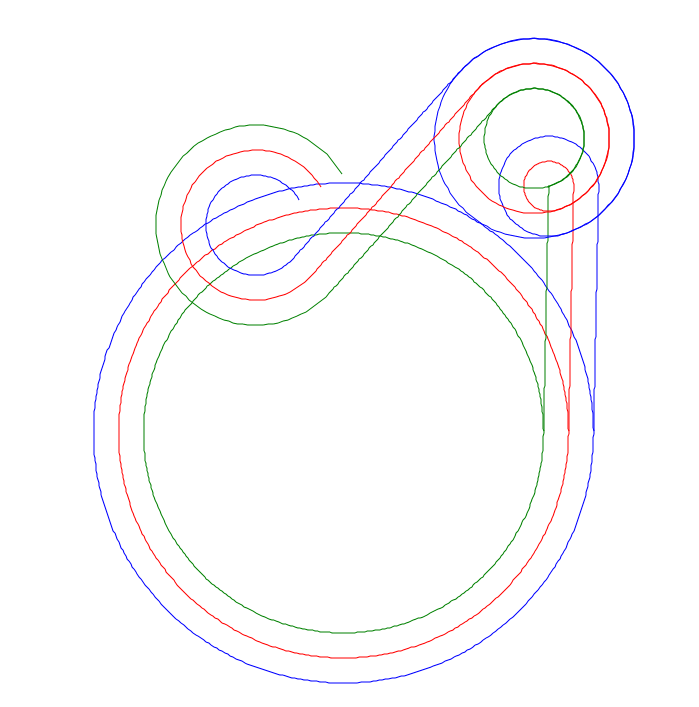
(1.4)

(1.5)

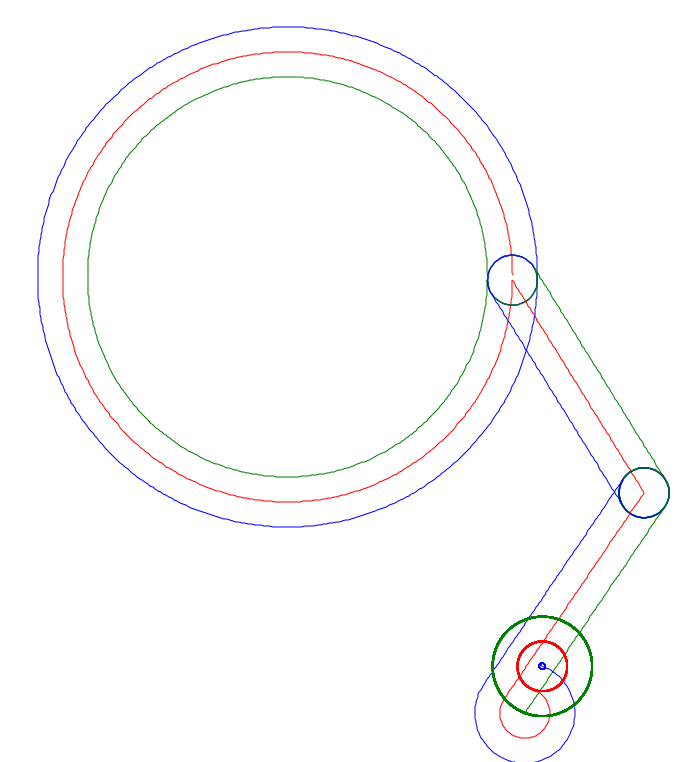
(1.6)

Trajektórie ťažiska a kolies robota som vykresľoval na Canvas v C#. Trajektórie reprezentujú čiary ktoré sú definovane aktuálnou a predošlou pozíciou ťažiska/ kolesa.

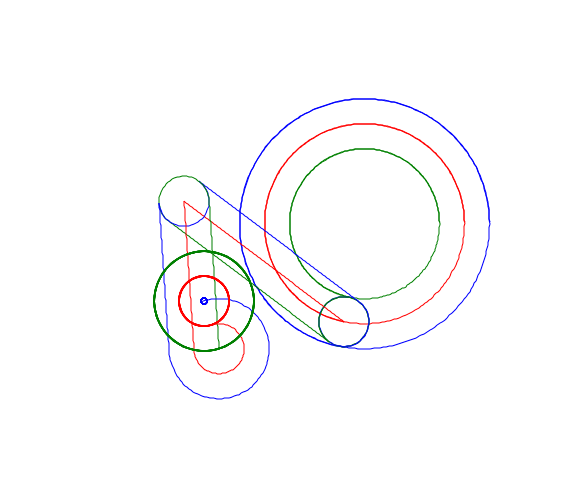
1. V prvej úlohe sme vykreslili trajektórie pomocou vektorov rýchlosti pravého a ľavého kolesa a vektorov časov v ktorých sa rýchlosti kolies menili.



*Obr. 1: VL = [ 1.25, 1, 1, 2, 1, 1, 0], VR = [1, 1, 0, 1, 1, 2, 0], T = [0, 5, 6, 7, 9, 10, 11]*



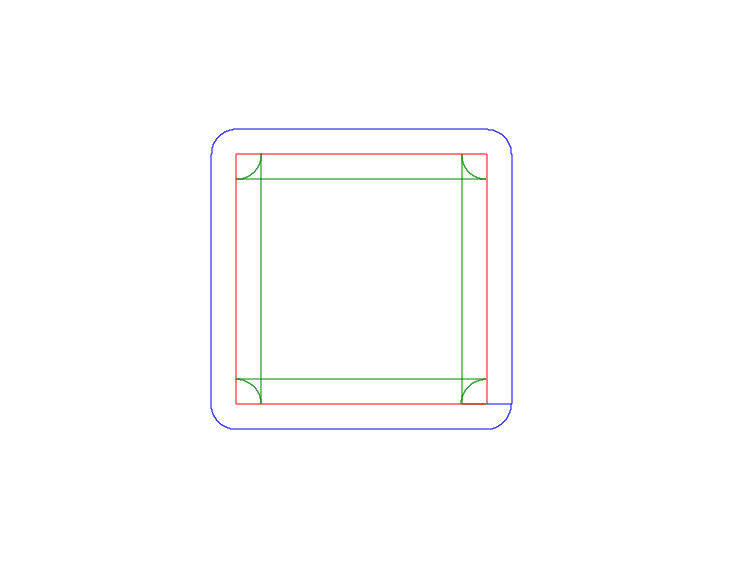
*Obr. 2: VL = [ 1.25, 1, 1, -1, 1, 1, 0], VR = [1, -1, 1, 1, 1, 0.5, 0], T = [0, 5, 6, 7, 9, 10, 11]*



*Obr. 3: VL = [ 1.25, 1, 1, -1, 1, 1, 0], VR = [0.75, -1, 1, 1, 1, 0.5, 0],*

*T = [0, 4.3, 5.2, 6, 6.4, 7, 8]*

1. V druhej úlohe sme vykresľovali štvorcovú trajektóriu ťažiska s nastaviteľnou stranou štvorca a taktiež sme vykresľovali trajektórie kolies.

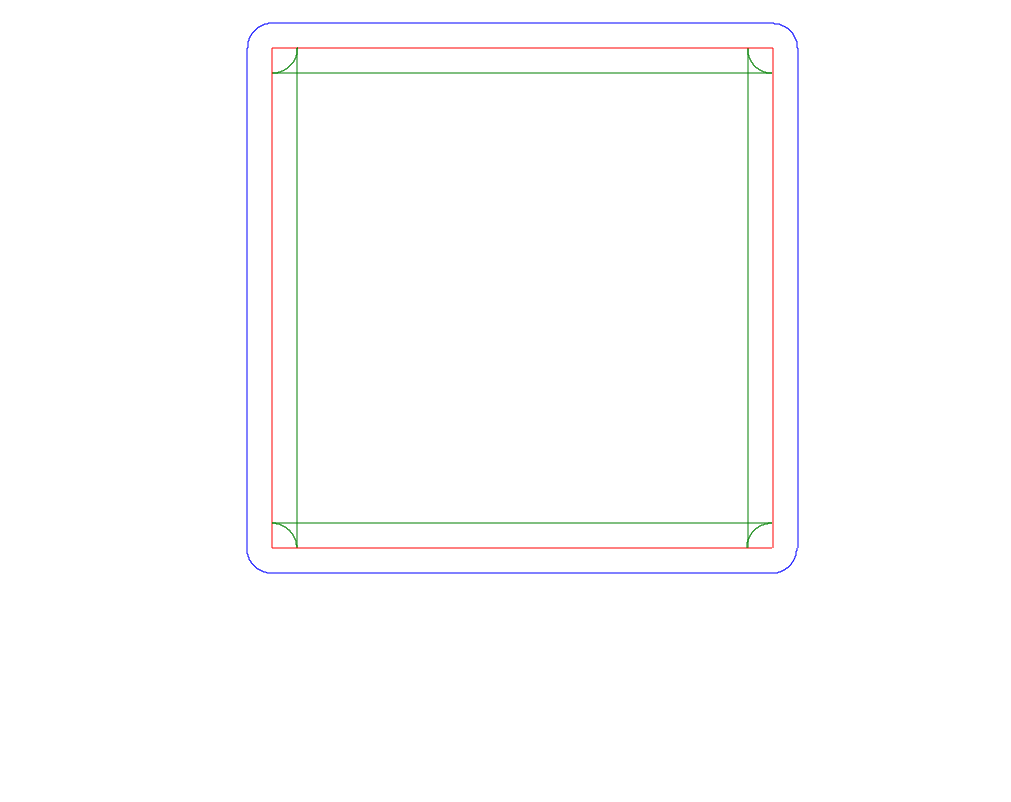


1.2 m

0.8 m

1 m

*Obr. 4: Štvorcová trajektória ťažiska s dĺžkou strany 1 m*



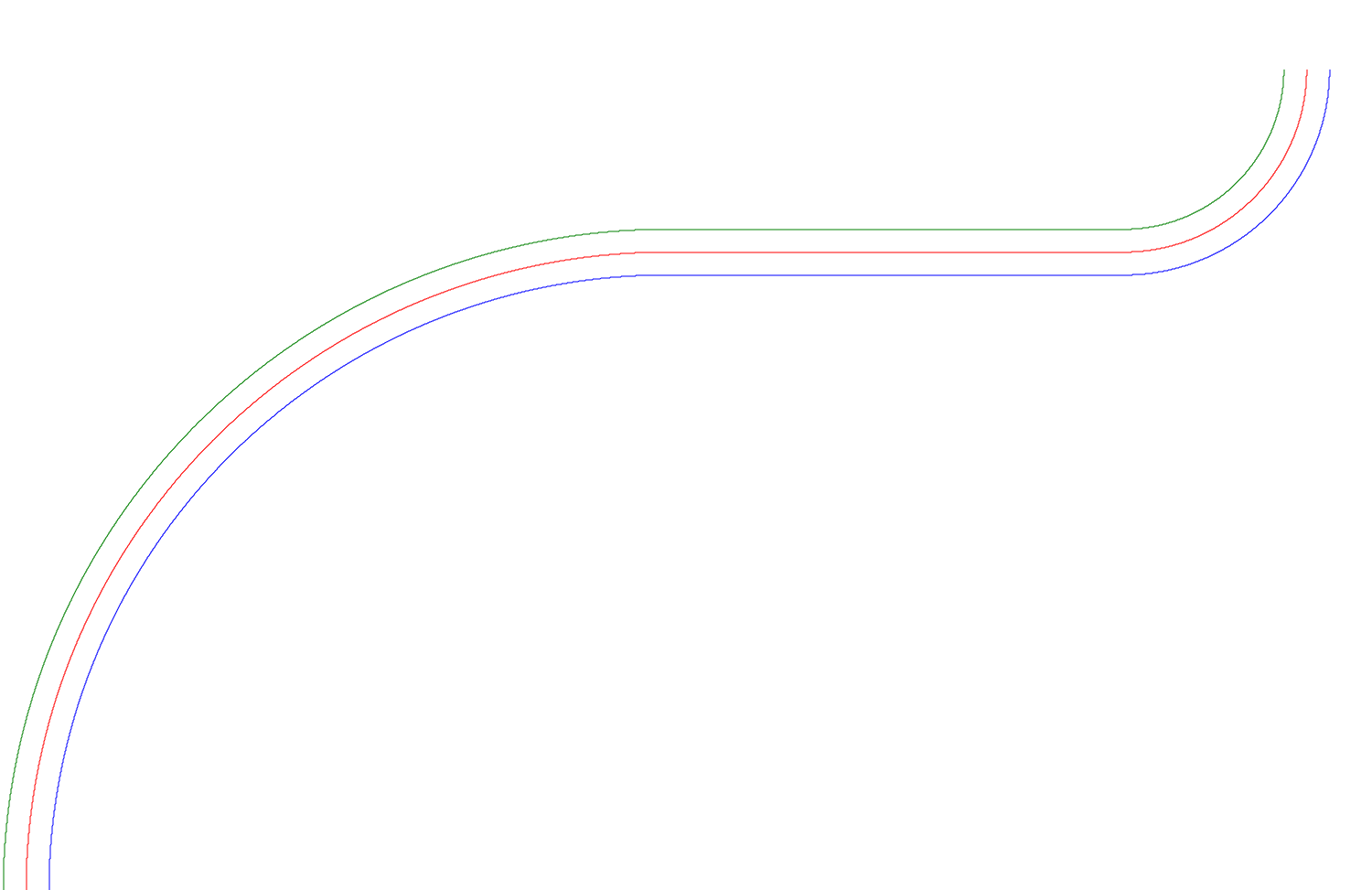
1.8 m

2 m

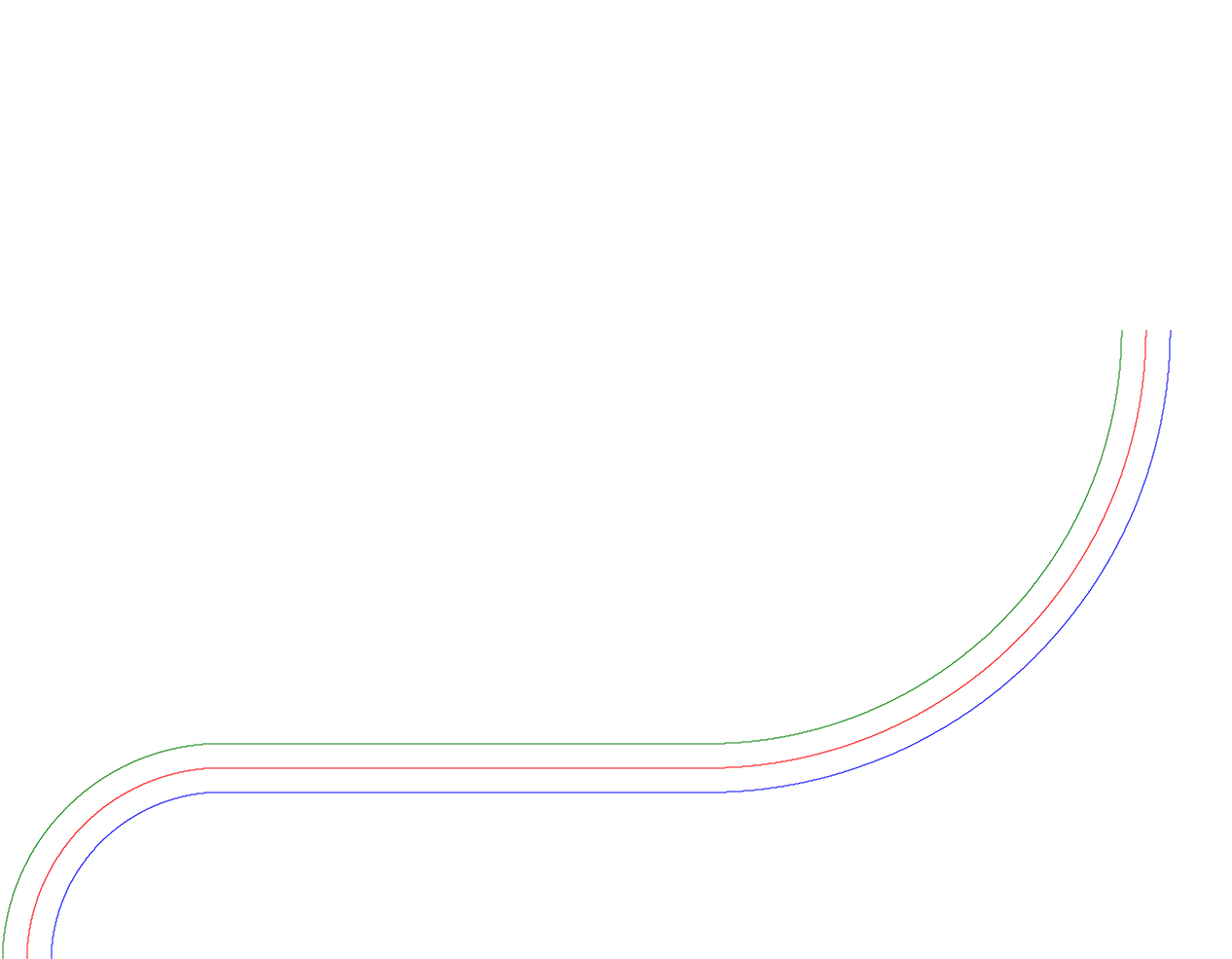
2.2 m

*Obr. 5: Štvorcová trajektória ťažiska s dĺžkou strany 2 m*

V 3 úlohe sme vykresľovali S krivku na základe zadaných polomerov(radius) otáčania R1, R2 a dĺžky L.

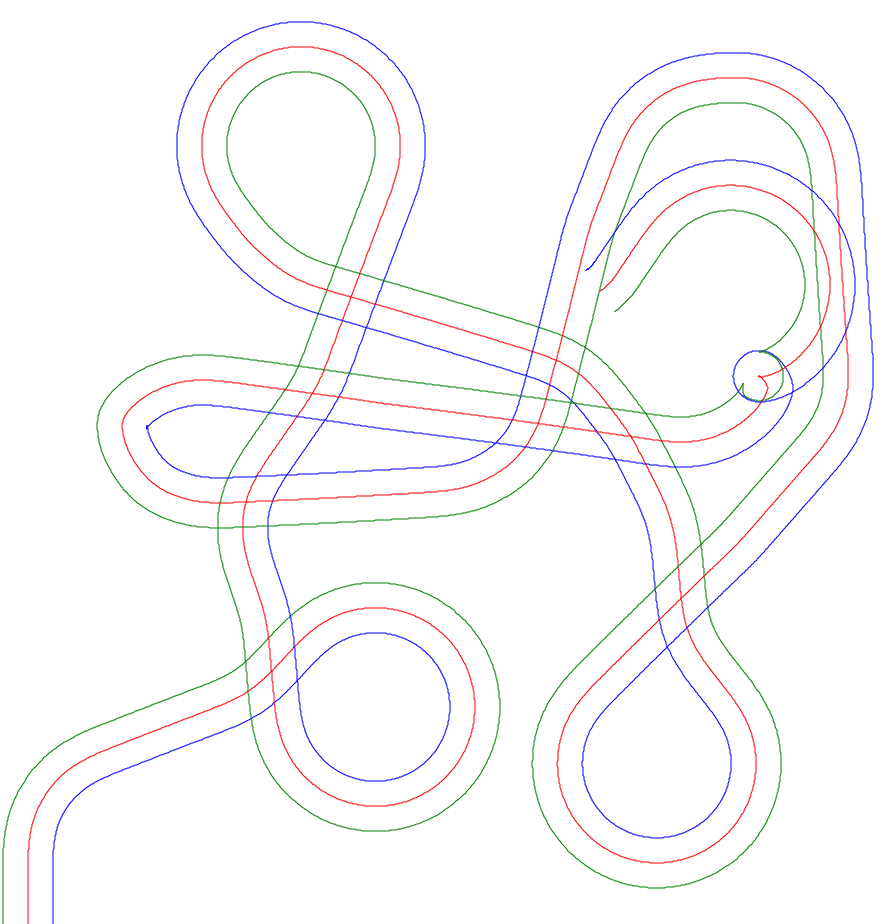


*Obr. 6: Trajektória ťažiska opisujúca S krivku s parametrami R1 = 3 m, L = 2 m, R2 = 1 m*



*Obr. 7: Trajektória ťažiska opisujúca S krivku s parametrami R1 = 1 m, L = 2 m, R2 = 2 m*

V poslednej úlohe sme vytvorili hru v ktorej ovládame robota s diferenciálnym podvozkom. Pomocou šípok UP a DOWN sa ovláda lineárna rýchlosť a LEFT a RIGHT sa ovláda rotačná rýchlosť robota. V hre je zakomponovaná aj zotrvačnosť pohybu. To jest keď pustime šípky a robot sa pohybuje začne postupne lineárne spomaľovať dokým nezastane.



*Obr. 8: Trajektória ťažiska a kolies robota vykreslená pri ovládaní robota pomocou šípok na klávesnici*

**Spustenie:**

Na spustenie aplikácie je nutné mať nainštalovaný program Visual Studio alebo jeho ekvivalent. Súčasne je nutné mať nainštalovaný framework .NET verziu 4.7.2. Projekt WpfApp1 otvoríme vo Visual Studiu. Následne zapneme aplikáciu pomocou klávesy F5 alebo v menu Debug – start with debugging.

**Záver:**

V rámci zadania sme si utvrdili znalosti o diferenciálnom robote, taktiež sme sa naučili základy programovania v jazyku c# a osvojili sme si jeho grafický framework wpf. Zostrojili sme program v ktorom sa dajú prepínať módy v ktorých robot vykresľuje trajektórie ťažiska a kolies podľa zadania. Súčasne sme naprogramovali hru v ktorej ovládame pohyb robota pomocou šípok. Zadanie som vypracoval sám.

**Literatúra:**

Cvičenia a prednášky z robotiky

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.threading.dispatchertimer?view=windowsdesktop-6.0>

<https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2022>

**Prílohy:**

*Position.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfApp1

{

// Position class consists of the position x,y and their INT equivalent

public class Position

{

private double x;

private double y;

public int IntX { get; set; }

public int IntY { get; set; }

public Position(double x, double y)

{

setX(x);

setY(y);

}

public Position()

{

this.x = 0;

this.y = 0;

}

public void setX(double x)

{

this.x = x;

this.IntX = (int) x;

}

public void setY(double y)

{

this.y = y;

this.IntY = (int) y;

}

public double getX()

{

return this.x;

}

public double getY()

{

return this.y;

}

}

}

*Controller.cs*

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Diagnostics.Eventing.Reader;

using System.Linq;

using System.Printing;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

using System.Windows.Media;

namespace WpfApp1

{

internal class Controller

{

private const int OFFSETX = 30;//200; // pozicia na canvase na zaciatku

private const int OFFSETY = 10;//200; // pozicia na canvase na zaciatku

private const int SCALE = 250; // mierka

private double L = 0.2; // m rozchod kolies

private const double D = 0.1; // m vzdialenost medzi taziskom a kolesom

private double r = 0.05; // m polomer kolesa

///////////////////////////////////////////////////////////////////////////////////////////////////////////// zadane premenne ///////

private double[] velLeftWheelArray = new double[] { 1.25, 1, 1, -1, 1, 1, 0}; // vektor rychlosti laveho kolesa ///

private double[] velRightWheelArray = new double[] { 1, -1, 1, 1, 1, 0,5, 0}; // vektor rychlosti praveho kolesa ///

private double[] timeStamps = new double[] { 0, 5, 6, 7, 9, 10, 11 }; // vektor casov ///

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

private double velRightWheel = 0; // rychlost praveho kolesa

private double velLeftWheel = 0; // rychlost laveho kolesa

private int timeIndex = 0;

public double timeDev = 0.001; // rychlost simulacie/frekvencia casovaca v sekundach

private int leftRightArrow; // Keyboard sipky LEFT = -1 || RIGHT = 1

private int upDownArrow; // Keyboard sipky DOWN = -1 || UP = 1

private double rotVel; // rad/s

private double linVel; // m/s

private double angle; // rad

private double currTime; //s

public Position prevPosition = new Position(OFFSETX, OFFSETY); // predchadzajuca pozicia Taziska

public Position position = new Position(OFFSETX, OFFSETY); // pozicia Taziska

public Position prevLeftWheel =new Position(); // predchadzajuca pozicia laveho kolesa

public Position prevRightWheel = new Position(); // predchadzajuca pozicia praveho kolesa

public Position leftWheel= new Position(OFFSETX - SCALE\*D, OFFSETY); // pozicia laveho kolesa

public Position rightWheel= new Position(OFFSETX + SCALE \* D, OFFSETY); // pozicia praveho kolesa

public Controller()

{

}

// runGame fcn - incorporates main logic that control the game

// runGame fcn - implementuje logiku ktora ovlada hru

public void runGame()

{

KeyListener();

Debug.WriteLine(leftRightArrow);

changeVel();

angle = calculateAngle(rotVel, angle, timeDev);

calculatePosition(position, prevPosition, linVel, angle, timeDev);

calculateWheelPosition(position, rightWheel, prevRightWheel, D, angle + Math.PI / 2);

calculateWheelPosition(position, leftWheel, prevLeftWheel, D, angle - Math.PI / 2);

}

// runLogic - logic for tasks 1,2,3

// logika na vykonanie uloh 1,2,3 zo zadania

public void runLogic()

{

changeVelOfWheel();

linVel = calculateLinVel(velLeftWheel, velRightWheel);

rotVel = calculateRotVel(velLeftWheel, velRightWheel, L);

angle = calculateAngle(rotVel, angle, timeDev);

calculatePosition(position, prevPosition, linVel, angle, timeDev);

calculateWheelPosition(position, rightWheel, prevRightWheel, D,angle +Math.PI/2);

calculateWheelPosition(position,leftWheel, prevLeftWheel, D, angle -Math.PI / 2);

currTime += timeDev;

}

// calculate linear velocity of the robot

// vypocet linearnej rychlosti robota

private double calculateLinVel(double leftWheelVel, double RightWheelVel)

{

return (leftWheelVel + RightWheelVel) / 2;

}

// calculate rotational velocity of the robot

// vypocet rotacnej rychlosti robota

private double calculateRotVel(double leftWheelVel, double rightWheelVel, double wheelBase)

{

return (rightWheelVel - leftWheelVel) / wheelBase;

}

// calculatet angle of rotation in radians

// vypocet uhla otocenia v radianoch

private double calculateAngle(double rotVelocity, double previousAngle, double timeDeviation)

{

double angle = previousAngle + rotVelocity \* timeDeviation;

if (angle > 2 \* Math.PI)

{

angle -= 2 \* Math.PI;

}

else if (angle < 0)

{

angle += 2 \* Math.PI;

}

return angle;

}

// calculate position - X and Y coordinates of the robot

// vypocitaj poziciu - x,y suradnice robota

private void calculatePosition(Position \_position, Position \_prevPosition, double \_linVel, double angleR, double timeDeviation)

{

\_prevPosition.setX(\_position.getX());

\_prevPosition.setY(\_position.getY());

double linDeviation = \_linVel \* timeDeviation \* SCALE;

\_position.setX(\_position.getX() + linDeviation \* Math.Sin(angleR)); // \*3

\_position.setY(\_position.getY() + linDeviation \* Math.Cos(angleR)); // \*3

}

// calculate position of the wheel

// vypocitaj poziciu - x,y suradnice kolesa

private void calculateWheelPosition(Position reference,Position \_position, Position \_prevPosition, double linDeviation, double angleR)

{

\_prevPosition.setX(\_position.getX());

\_prevPosition.setY(\_position.getY());

\_position.setX(reference.getX() + SCALE \* linDeviation \* Math.Sin(angleR)); // \*3

\_position.setY(reference.getY() + SCALE \* linDeviation \* Math.Cos(angleR)); // \*3

}

// iteration over velocity vectors of the wheels and setting duration of that velocity

// iterovanie cez vektor rychlosti kolies a nastavenie casu trvanie rychlosti kolies

private void changeVelOfWheel()

{

if (timeStamps[timeIndex] <= currTime)

{

velLeftWheel = velLeftWheelArray[timeIndex];

velRightWheel = velRightWheelArray[timeIndex];

if (timeIndex != timeStamps.Length - 1)

{

timeIndex++;

}

}

}

//draw rectangle of defined size

// kreslenie stovrca definovanej velkosti

public void drawRect(int size)

{

velLeftWheelArray = new double[] { 2, 2, 2, 2, 2, 2, 2, 2, 0 };

velRightWheelArray = new double[] { 2, -2, 2, -2, 2, -2, 2, -2, 0 };

timeStamps = new double[velLeftWheelArray.Length];

for (int i = 0; i < velLeftWheelArray.Length - 1; i++)

{

if (i == 0)

{

timeStamps[0] = 0;

}

if (calculateLinVel(velLeftWheelArray[i], velRightWheelArray[i]) != 0)

{

timeStamps[i+1] = timeStamps[i] + size / Math.Abs(calculateLinVel(velLeftWheelArray[i], velRightWheelArray[i]));

}

else if (calculateRotVel(velLeftWheelArray[i], velRightWheelArray[i],L)!= 0)

{

timeStamps[i+1] = timeStamps[i] +(Math.PI / 2) / Math.Abs(calculateRotVel(velLeftWheelArray[i], velRightWheelArray[i],L));

}

else

{

timeStamps[i+1]= 0;

}

}

}

// draw S line with parameters R1 L R2

// nakresli S krivku so zadanymi parametrami R1 L R2

public void drawSLine(double R1, double L1, double R2)

{

velLeftWheelArray = new double[4];

velRightWheelArray = new double[4];

timeStamps = new double[4];

double vt = 2;

double vr = R1 \* vt \* 2 / L - vt;

double vl = vr - vt \* 2;

double speedLimit = (vr + vl / 2) / vt;

velLeftWheelArray[0] = vl/speedLimit;

velRightWheelArray[0] = vr/speedLimit;

timeStamps[0] = 0;

timeStamps[1] = (Math.PI / 2) /calculateRotVel(vl,vr,L)\*speedLimit;

velLeftWheelArray[1] = 2;

velRightWheelArray[1] = 2;

timeStamps[2] = timeStamps[1]+ L1 / calculateLinVel(2, 2);

vl = R2 \* vt \* 2 / L - vt;

vr = vl - vt \* 2;

velLeftWheelArray[2] = vl/speedLimit;

velRightWheelArray[2] = vr/speedLimit;

velLeftWheelArray[3] = 0;

velRightWheelArray[3] = 0;

timeStamps[3] = timeStamps[2] + (-Math.PI / 2) / calculateRotVel(vl, vr, L)\*speedLimit;

}

// keyboard Listener

// reaguje na stlacenie tlacidiel UP DOWN LEFFT RIGHT na klavesnici

private void KeyListener()

{

if ((Keyboard.GetKeyStates(Key.Left) & KeyStates.Down) > 0)

{

leftRightArrow = -1;

}

if ((Keyboard.GetKeyStates(Key.Right) & KeyStates.Down)>0)

{

leftRightArrow = 1;

}

if (((Keyboard.GetKeyStates(Key.Right) & KeyStates.Down) == 0) && ((Keyboard.GetKeyStates(Key.Left) & KeyStates.Down) == 0) )

{

leftRightArrow = 0;

}

if ((Keyboard.GetKeyStates(Key.Down) & KeyStates.Down) > 0)

{

upDownArrow = -1;

}

if ((Keyboard.GetKeyStates(Key.Up) & KeyStates.Down) > 0)

{

upDownArrow = 1;

}

if (((Keyboard.GetKeyStates(Key.Up) & KeyStates.Down) == 0) && ((Keyboard.GetKeyStates(Key.Down) & KeyStates.Down) == 0))

{

upDownArrow = 0;

}

}

// change rotational and linear velocity acording pressed buttons on keyboard

// zmen rotacnu a linearnu rychlost podla stlaceneho tlacitka na klavesnici

public void changeVel()

{

// rot velocity

//arrow 1 = right

//arrow -1 = left

// arrow 0 = no change

if (upDownArrow == 1 && linVel < 4)

{

linVel += 0.05;

}

if (upDownArrow == -1 && linVel > -4)

{

linVel -= 0.05;

}

if (upDownArrow == 0 && linVel > 0)

{

linVel -= 0.05;

}

else if (upDownArrow == 0 && linVel < 0)

{

linVel += 0.05;

}

//////////////

/// lin velocity

if (leftRightArrow == 1 && rotVel < 10)

{

rotVel += 0.2;

}

if (leftRightArrow == -1 && rotVel > -10)

{

rotVel -= 0.2;

}

if (leftRightArrow == 0 && rotVel > 0)

{

rotVel -= 0.2;

}

else if (leftRightArrow == 0 && rotVel < 0)

{

rotVel += 0.2;

}

}

}

}

*MainWindow.xaml.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace WpfApp1

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private int mode = 3; // nastavenie modu 0 - zakladny rezim pohyb pomocou dopredu naprogramovanych vektorov rychlosti kolies a casovych vektorov

// 1 - kreslenie stvorca pri kresleni stvorca aby sme vykreslili pravouhly stvorec musime zmensit frekvenciu casovaca

// alebo zmensit mierku pretoze dochadz k zaokruhlovaniu

// 2 - kreslenie S line

// 3 - hra

private Canvas myCanvas;

private Controller controller;

///

public MainWindow()

{

InitializeComponent();

this.controller = new Controller();

switch (mode)

{

case 0:

break;

case 1:

controller.drawRect(2); // rozmer strany stvorca

break;

case 2:

controller.drawSLine(1,2,2); // nastavenie polomerov S krivky

break;

default:

break;

}

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(controller.timeDev);

timer.Tick += timerFcn;

timer.Start();

}

// timer function - called by the timer every x millisecs

// timer funkcia - je volana casovacom ktora sa periodicky opakuje kazdych x millisekund

public void timerFcn(object sender, EventArgs e)

{

if (mode == 3)

{

controller.runGame();

}

else

{

controller.runLogic();

}

drawLine(MyCanvas,controller.prevPosition, controller.position,0);

drawLine(MyCanvas, controller.prevLeftWheel, controller.leftWheel, 1);

drawLine(MyCanvas, controller.prevRightWheel, controller.rightWheel, 2);

}

// draw function - used for drawing lines representing trajectory of the center of mass and the wheels

// drawLine - sluzi na vykreslovanie ciar reprezentujuce trajektorie taziska a kolies robota

public void drawLine(Canvas MyCanvas, Position lastPosition, Position currPosition,int color)

{

Line line = new Line();

line.X1 = lastPosition.IntX;

line.Y1 = MyCanvas.ActualHeight - lastPosition.IntY;

line.X2 = currPosition.IntX ;

line.Y2 = MyCanvas.ActualHeight - currPosition.IntY ;

if (color == 0)

{

line.Fill = Brushes.Red;

line.Stroke = Brushes.Red;

}

else if (color == 1)

{

line.Fill = Brushes.Green;

line.Stroke = Brushes.Green;

}

else

{

line.Fill = Brushes.Blue;

line.Stroke = Brushes.Blue;

}

MyCanvas.Children.Add(line);

// MyCanvas.Children.RemoveAt(MyCanvas.Children.Count); // na zmazanie robota

}

}

}

*MainWindow.xaml*

<Window x:Class="WpfApp1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Diferencialny podvozok" WindowState="Normal" ResizeMode="CanMinimize" WindowStartupLocation="CenterScreen" Height="1080

" Width="1920">

<Grid>

<Canvas Name="MyCanvas" Margin="0 0 0 0"></Canvas>

</Grid>

</Window>