|  |  |  |
| --- | --- | --- |
| ING100 Ingeniørfaglig innføringsemne  Beskrivelse: UiS%5Fnor%5Fcolor  PROSJEKT – HØSTEN 2014 | | |
| Prosjekt-  oppgaven | **Matlab og LEGO NXT i anvendt matematikk og fysikk** | |
|  | | |
| Gruppenummer | **1401** | |
| Gruppens  medlemmer | Navn | Studentnummer |
| **Helge Johannessen Bjorland** | **228290** |
|  |  |
|  |  |
|  | | |
| Sammendrag | Skriv et sammendrag av hvordan programmene virker. Et sammendrag skal være kort og konkret, og skal ikke inneholde figurer eller kode. Det skal være en oppsummering av problemstilling, hvordan dere har løst problemet og hvordan det ferdige programmet fungerer (som en slags konklusjon). Dere kan gjerne strukturere sammendraget slik at hvert program beskrives hver for seg, for eksempel:  **Program1 (Hele gruppen i fellesskap):**  - hva som var problemstillingen  - hvordan dere har løst dette (detaljer om programmet)  - virker det etter hensikten?  **Program2 (Utført av Per og Pål):**  - hva som var problemstillingen  - hvordan dere har løst dette (detaljer om programmet)  - virker det etter hensikten?  **Program 3 (Utført av Kari og Anne):**  .  .  Skriv slik at leseren får lyst til å lese mer om programmene i selve rapporten. | |

# 1. Prosjektgjennomføringen

Prosjektet startet med å avtale agenda for første gruppemøte og sette opp Github konto for å dele matlab kode med kildehåndtering. Første gruppemøte ble avholdt 16. September (2t). Ansvar ble fordelt og tisplan under ble vedtatt:

* 10 okt - ferdig med obligatorisk del
* 24 okt - Ferdig med skisse av kreativ del
* 7 nov - Ferdig med kreativ del slik at vi ferdigstiller rapport frem mot innleveringsfrist
* 21 nov - innleveringsfrist, prosjekt ferdig

Obligatorisk del ble gjennomført av alle gruppemedlemmene i hovedsak individuelt, men også noe i grupper alt etter hva som passet med arbeidssituasjon. Fokus var at alle gruppemedlemmer skulle få en god forståelse av den obligatoriske delen. Kommunikasjonsmåte mellom gruppemøter var via epost og telefon og roboten rullerte alt etter behov. Det tok ca 4 timer å bygge roboten (HJB) og hver person brukte omtrent 8 timer på obligatorisk del. dimer ﷽﷽﷽r:l følgende pl med. kle og middels enkle oppgaver og en vanskelig. Dette ble fordelt som følger:l følgende pl med.

10. oktober ble det avholdt nytt gruppemøte hvor vi ble enige om hvordan vi skulle kombinere den obligatoriske delen til en kode og fordelte kreative oppgaver å jobbe med. Vi ble enige om å gjøre flere enkle og middels enkle oppgaver og en vanskelig. Dette ble fordelt som følger:

Helge:

* Koordiner rapport
* Koordiner obligatorisk del i en fil
* Vis robotens hastighet under kjøring
* Initialiser joystick ved start
* Kreative oppgaver: 3, 7, 10, 22 og 26

Anders:

* Kreative oppgaver: 1 og 6

Espen:

* Kreative oppgaver: 23

Daniel:

* Kreative oppgaver: 5, 8 , 9 og 14

# 2. Program 1: Legorobot (Utført av hele gruppen)

Skriv og forklar hvordan funksjonene for integrering, derivering og filtrering fungerer. Få frem hvordan dette fungerer i forbindelse med Lego-roboten. Ta med ”screen shots” i forklaringen, utklipp fra koden og Matlabfigurer. Beskrivelsen skal fungere som en rask brukerveiledning slik at leseren skal kunne forstå hvordan dere har løst problemet. Husk å nevne eventuelle spesialiteter.

## 2.1 Numerisk integrasjon (Utført av hele gruppen)

Ta med kode og figurer som viser at dere har forstått hvordan numerisk integrasjon fungerer.

## 2.2 Numerisk derivasjon (Utført av hele gruppen)

Ta med kode og figurer som viser at dere har forstått hvordan numerisk derivasjonen fungerer.

## 2.3 Filtrering (Utført av hele gruppen)

Ta med kode og figurer som viser hvordan filtreringen fungerer. Her kan dere inkludere mange forskjellige filtervarianter og vise at dere har testet og forstått filtrering. Knytt gjerne denne filtreringen opp mot resultatene fra numerisk integrasjon og derivasjon ved å vise hva som skjer dersom man filtrerer *før* integrasjonen/derivasjonen.

## 2.4 Kjøring av Legorobot (Utført av hele gruppen)

Ta med kode og figurer som viser hvordan manuell kjøring av Legoroboten fungerer totalt sett.

# 3. Program 2: CannonGame (Espen Ro Eliassen)

- Lage et spill hvor et prosjektil skytes ut og flyr et visst antall meter før det treffer bakken.

- Skuddet skal følge fysikkregler, hvor en ser bort i fra luftmotstand. Lengden skal være gitt ved: 

- Skuddet skal illustreres/animeres, altså må et antall punkter i skuddbanen kalkuleres. Dette skal gjøres ved å dekomponere hastighetene, regne de som en funksjon av tiden. Finne hvor lang tid skuddet tar totalt og vise det med en oppløsning på 0.1 sekunder. Formel for strekning gitt ved tid:

 ; Gravitasjon= -9.81, startfart = 100m/s.

- Vinkel for skudd skal vises som en strek som varierer mellom 0° og 90° over tid. Skuddet avfyres ved å trykke “space” på tastatur. Da skal vinkelen på streken hentes og brukes i kalkulasjonene.

- Det skal være forskjellige vanskelighetsgrader (Easy, Normal, Hard), som bestemmer hvor fort vinkelen endrer seg.

- Ett spill skal bestå av 3, 5, 7 eller 10 skudd.

- Poengsum (score) skal vises hver gang ett spill er ferdig, både lengste skudd og total skuddlengde. Høyeste punkt på skudd skal også vises.

- Highscore skal lagres i en fil «cannongame\_highscore.mat», om filen ikke finnes skal den lages. Det skal være highscore lister for lengste skudd og høyeste total lengde for hvert av antall skudd. Hver highscore liste skal inneholde 10 elementer og være sortert med høyeste poengsum først.

- Det skal både være en- og flerspiller mulighet. Vinner av flerspiller skal kåres ved total lengde.

- Spillet skal ha et enkelt og selvforklarende grafisk brukergrensesnitt (GUI).

- Det skal være mulig å spille på nytt etter endt spill eller avslutte.

- Spillet skal skrives som en funksjon som kan kalles fra andre funksjoner eller script.

Kode finnes i Appendix A

# 4. Program 3: (kreativ del) (Utført av Kari, Anne og Pål)

# 6. Konklusjon

Skriv noen setninger som oppsummerer programmene og hvordan de virker. Denne konklusjonen kan også deles opp etter program slik som sammendraget.

# Referanser

[1] [www.mathworks.se](http://www.mathworks.se)

# A. Matlabfiler

Alle Matlabfiler i skriftstørrelse 6.

## A.1 Program 1 Obligatorisk del

## A.2 Program 2 CannonGame

function [] = cannonGame()

%%

%constants

screen = get(0,'screensize');

chartratio=(550/1050)\*screen(3)/screen(4); %ratio between plot axis times ratio between screen size

chartaxis=[0,1050,0,550]; %Contant to lock the displayed axis slightly larger than possible height and length with an initial speed of 100 m/s

g=-9.81; %Gravitational acceleration

initialSpeed = 100; %Shots intial speed m/s

%%

%init var

showangle=1; %Condition to run showangle function

space = 0; %Continue variable

stop = 0; %main loops variable

angle = 0; %angle of shot

shotno = 1; %shoot number

noshots = 0; %number of shots this round

dispshot=0; %display shot condition

showhighscore=false; %show highscore

noplayers=0; %number of players

playerno=1; %current players number

highscore=loadHighscore(); %structur variable used for highscores, and initially loading highscores from file

%%

%initial game options, such as number of players, difficulty, highscore and

%exit

switch menu('Choose: ', 'Single player', 'Multiplayer', 'Highscore','Exit')

case 1

noplayers=1;

case 2

noplayers=menu('Number of players: ', '2', '3', '4', '5')+1;

case 3

noplayers=0;

showhighscore=true;

case 4

noplayers=0;

end

if noplayers>0

switch menu('Choose difficulty:', 'easy', 'normal', 'hard');

case 1

difficulty =0.01; %change from one angle to the next in radians

case 2

difficulty =0.02; %change from one angle to the next in radians

case 3

difficulty =0.03; %change from one angle to the next in radians

end

switch menu('Number of shots:', '3', '5', '7', '10')

case 1

noshots = 3;

case 2

noshots = 5;

case 3

noshots = 7;

case 4

noshots = 10;

end

end

%%

%initializing figure by defining position on screen, the keydownlistnere

%and removing the possibility to resize the figure.

figure('Position',[screen(3)/6, screen(4)/6,screen(3)/1.5, screen(4)/1.5]);

set(gcf, 'KeyPressFcn', @keyDownListener, 'Resize', 'off');

%%

%keyDownListener to know when "space" has been hit, also escape to stop

%main loops

function keyDownListener(src, event)

switch event.Key

case 'space'

space=1;

showangle=0;

case 'escape'

stop=1;

end

end

%%

%load highscore from file

function hs = loadHighscore()

try %try loading highscore file

hs=load('highscore\_cannongame.mat');

catch %if loading highscore file fails, make it

lShots = zeros(1,10); %longest shots

tot3shots = zeros(1,10); %highest total of 3 shot games

tot5shots = zeros(1,10); %highest total of 5 shot games

tot7shots = zeros(1,10); %highest total of 7 shot games

tot10shots = zeros(1,10); %highest total of 10 shot games

save('highscore\_cannongame','lShots','tot3shots','tot5shots','tot7shots','tot10shots'); %create the highscore file

hs=load('highscore\_cannongame.mat');

end

end

%%

%update longest shot highscore

function updateLongestShotHS(shotlength)

highscore.lShots(end+1) = shotlength; %add current shots to longest shot list

highscore.lShots = sort(highscore.lShots,'descend'); %resort the longest shot list

lShots = highscore.lShots(1:10); %remove the lowest value from longest shot list

save('highscore\_cannongame','-append','lShots'); %resave the longest shot list

end

%%

%update total length highscore see comments for update longest shot

%highscore, it follows exact same logic only for the rest of the lists.

function updateTotalLengthHS(totallength)

if noshots == 3

highscore.tot3shots(end+1) = totallength;

highscore.tot3shots = sort(highscore.tot3shots,'descend');

tot3shots=highscore.tot3shots(1:10);

save('highscore\_cannongame','-append','tot3shots');

elseif noshots == 5

highscore.tot5shots(end+1) = totallength;

highscore.tot5shots = sort(highscore.tot5shots,'descend');

tot5shots=highscore.tot5shots(1:10);

save('highscore\_cannongame','-append','tot5shots');

elseif noshots == 7

highscore.tot7shots(end+1) = totallength;

highscore.tot7shots = sort(highscore.tot7shots,'descend');

tot7shots=highscore.tot7shots(1:10);

save('highscore\_cannongame','-append','tot7shots');

elseif noshots == 10

highscore.tot10shots(end+1) = totallength;

highscore.tot10shots = sort(highscore.tot10shots,'descend');

tot10shots=highscore.tot10shots(1:10);

save('highscore\_cannongame','-append','tot10shots');

end

end

%%

%showAngle function, displaying a line that moves from 0° towards 90° then back

%towards 0°

function showAngle()

cannonLength = 100; %lengt of line to ilustrate the angle with

while ~space && angle<pi/2 && ~stop %show line from 0° towards 90°

x=cannonLength\*cos(angle); %furthest x value of line

y=cannonLength\*sin(angle)\*chartratio; %highest y value of line times chartratio display a more even size on screen

plot([0,x],[0,y]); %plot the line using point [0,0] and [x,y]

axis(chartaxis); %set the axis that is to be locked

title(['Player ', num2str(playerno), ': Shot no: ', num2str(shotno), '. Press "space" to fire at desired angle']); %give instructions in title

drawnow; %make sure it gets drawn

angle = angle + difficulty; %increase angle for next iteration by the value set by the selected difficulty

end

while ~space && angle>0 && ~stop %show line from 90° towards 0°

x=cannonLength\*cos(angle); %furthest x value of line

y=cannonLength\*sin(angle)\*chartratio; %highest y value of line times chartratio display a more even size on screen

plot([0,x],[0,y]); %plot the line using point [0,0] and [x,y]

axis(chartaxis); %set the axis that is to be locked

title(['Player ', num2str(playerno), ': Shot no: ', num2str(shotno), '. Press "space" to fire at desired angle']); %give instructions in title

drawnow; %make sure it gets drawn

angle = angle - difficulty; %decrease angle for next iteration by the value set by the selected difficulty

end

end

%%

%calculate and show shot

function [shotlength,shotheight] = shootCannon()

clf;

v0=initialSpeed; %initial speed

v0x=v0\*cos(angle); %initial speed in x direction

v0y=v0\*sin(angle); %initial speed in y direction

t=-(2\*v0y/g); %calculate time until shot hits ground again

tid=0:0.1:t; %make a time vector with steps 0.1 sec

tid(end+1)=t; %add the exact time of shot hiting the ground as last time in timevector

x=v0x.\*tid; %calculate each x value by time

y=v0y.\*tid+0.5\*g.\*tid.^2; %calculate each y value by time

for i=2:length(x) % simulate shot by drawing small lines

plot([x(i-1) x(i)],[y(i-1) y(i)]); %plot the small lines

axis(chartaxis); %set the axis that is to be locked

title(['Shot current length: ' num2str(x(i)) 'm. Shot current height: ' num2str(y(i)) 'm.']); %display details during shot

drawnow; %make sure it gets drawn

pause(0.01); %short pause to make sure it is a smooth illusration

end

title(['Shot length: ' num2str(x(end)) 'm. Shot height: ' num2str(max(y)) 'm. Press "space" to continue']); % give details about shot, such as shot length

dispshot=1; % set shot to be displayed until continued

angle=0; %reset angle

shotlength = x(end); %define shotlength output from function

shotheight = max(y); %define shotheight output form function

updateLongestShotHS(shotlength); %run function to update longest shot highscore

end

%%

%show highscores in figure

function showHighscores()

HiStL = {'Longest shots:'}; %highscore structure string longest shot

HiStT3 = {'Highest Total 3:'}; %highscore structure string sum of 3 shots

HiStT5 = {'Highest Total 5:'}; %highscore structure string sum of 5 shots

HiStT7 = {'Highest Total 7:'}; %highscore structure string sum of 7 shots

HiStT10 = {'Highest Total 10:'}; %highscore structure string sum of 10 shots

for i=1:10 % load the highscore lists into the correct structures

HiStL(end+1) = {[num2str(i) ': ' num2str(highscore.lShots(i)) 'm']};

HiStT3(end+1) = {[num2str(i) ': ' num2str(highscore.tot3shots(i)) 'm']};

HiStT5(end+1) = {[num2str(i) ': ' num2str(highscore.tot5shots(i)) 'm']};

HiStT7(end+1) = {[num2str(i) ': ' num2str(highscore.tot7shots(i)) 'm']};

HiStT10(end+1) = {[num2str(i) ': ' num2str(highscore.tot10shots(i)) 'm']};

end

%write highscore lists to texboxes

annotation('textbox', [0.1,0.9,0.75,0.05],'backgroundcolor', 'w', 'String', 'Highscores, Press "space" to continue');

annotation('textbox', [0.1,0.1,0.15,0.8],'backgroundcolor', 'w', 'String', HiStL);

annotation('textbox', [0.25,0.1,0.15,0.8],'backgroundcolor', 'w', 'String', HiStT3);

annotation('textbox', [0.4,0.1,0.15,0.8],'backgroundcolor', 'w', 'String', HiStT5);

annotation('textbox', [0.55,0.1,0.15,0.8],'backgroundcolor', 'w', 'String', HiStT7);

annotation('textbox', [0.7,0.1,0.15,0.8],'backgroundcolor', 'w', 'String', HiStT10);

while ~space % draw and wait until player is ready to continue

drawnow;

end

end

%%

%main single player script

if noplayers ==1;

shotlength=zeros(1,noshots); %define vector for putting the length of the shots into

shotheight=zeros(1,noshots); %define vector for putting the height of the shots into

while ~stop %main single player game loop

drawnow;

if showangle %show the angle to the player

showAngle();

elseif space && ~dispshot %shot with the angle when player presses space

[shotlength(shotno), shotheight(shotno)] = shootCannon();

space=0;

shotno=shotno+1;

elseif shotno<=noshots && space %if more shots remaining reset conditions

showangle = 1;

dispshot = 0;

space=0;

elseif shotno>noshots && space %all shots completed, finish game

stop=1;

end

end

updateTotalLengthHS(sum(shotlength)); %update total length highscores

clf;

%display game details

tempstr = {['Longest shot: ', num2str(max(shotlength)), 'm'],

['Highest shot: ', num2str(max(shotheight)), 'm'],

['Total shot length: ' num2str(sum(shotlength)), 'm'],

'Press "Space" to continue'};

annotation('textbox', [0.1,0.1,0.8,0.8],'backgroundcolor', 'w', 'String', tempstr);

space=0;

while ~space %wait until player is ready to continue

drawnow

end

close gcf;

end

%%

%main multiplayer

if noplayers>1;

shotlength=zeros(noplayers,noshots); %define vector for putting the length of the shots into

shotheight=zeros(noplayers,noshots); %define vector for putting the height of the shots into

playerno=1;

while ~stop %main multiplayer game loop

drawnow;

if showangle() %shot with the angle when player presses space

showAngle();

elseif space && ~dispshot %shot with the angle when player presses space

[shotlength(playerno,shotno), shotheight(playerno,shotno)] = shootCannon();

space=0;

if playerno==noplayers %iterate through players and shot numbers

shotno=shotno+1;

playerno=1;

else

playerno=playerno+1;

end

elseif shotno<=noshots && space %if more shots remaining reset conditions

showangle = 1;

dispshot = 0;

space=0;

elseif shotno>noshots && space %all shots completed, finish game

stop=1;

end

end

clf;

winner = 0;

highTotThisGame = 0;

while playerno <= noplayers

tempTot=sum(shotlength(playerno,:)); %get total length for each player

updateTotalLengthHS(tempTot); %update highscore with total length for each player

if highTotThisGame < tempTot %decide who is winning

winner = playerno;

highTotThisGame = tempTot;

end

%create details to send to textbox

tempstr={['Player ',num2str(playerno)],['Longest shot: ', num2str(max(shotlength(playerno,:))), 'm'],

['Highest shot: ', num2str(max(shotheight(playerno,:))), 'm'],['Total shot length: ' num2str(tempTot), 'm']};

%create textbox for each player

annotation('textbox', [0.1+(0.15\*(playerno-1)),0.1,0.15,0.8],'backgroundcolor', 'w', 'String', tempstr);

playerno=playerno+1;

end

%create textbox with winner and continue instructions

annotation('textbox', [0.1,0.9,0.75,0.05],'backgroundcolor', 'w', 'String', ['Winner is Player ',num2str(winner), ': Press "space" to continue']);

space=0;

while ~space %wait until player is ready to continue

drawnow

end

close gcf;

end

%%

% end

if noplayers>0 %if game just ended, give intial options again

cannonGame();

elseif noplayers == 0 && showhighscore %show highscore, then show initial options again

showHighscores();

close gcf;

cannonGame();

end

close gcf;

end