2-D MULTI-PEN PLOTTER SIMULATOR

Brief description

Develop a program for 2-d plotter simulation with zero or more pens. Each pen can be moved across plotter drawing plane by attached motors in 2-d Cartesian coordinate system.

Drawing plane is infinite in size, but infinite positions like (-Inf, 4.6) are assumed not possible in any case - just do not care about them!

Each pen can be in two modes:

- drawing (on) - leaves a trace on plotter plane during movements;
- no drawing (off) - leaves no traces on plane.

Program must:

- read control text commands from standard input;
- implement simulation procedure: calculate pens positions in real-time according to given motors command;
- output simulation world state in log/text files.

Special conditions

- **solve a given problem with C++ or Java language;**
- **send us your solution as zipped working directory (inc. ".git" dir if Git'll be used);**
- optional but preferable:
  - use Git to demonstrate development process;
  - prefer to use "clean code" conceptions;
  - share your solution with us on "github.com".

Detailed description

Program runtime phases

For simplicity program run-time separated to strict phases:

1. **CONF**: simulator configuration phase.
2. **SIMS**: simulation run phase (main workload).
3. **FIN**: simulation stops at "stop" command received and then program exit.

Transition graphs is: [CONF] -> [SIMS] -> [FIN]

Simulation domain

- objects of simulation are pens and motors;
- each pen have two movement axes: x and y;
- to each axis of a pen one and only one motor can be attached;
- one motor can be attached to several axes or even to different pens (see APPENDIX B example with DIAG pen);
- motor transforms it's shaft rotary movements into linear movements of pen across axes;
- axis position measured in abstract units (a.u.);
- assume there are NO collisions possible between motors (or pens), so you do not need to take it in account;
- individual parameters of each pen is:
  - motors attached to axes x and y;
  - toggled on (drawing) or off (not drawing) state;
- individual parameters of each motor is:

```
S_max_aups  - maximal speed in abstract units per second (aups);
A_aupss     - acceleration absolute value in abstract units per seconds^2 (aupss);
TP          - target position in abstract units (au);
P           - current position in abstract units (au) relative to zero point;
V           - current velocity in abstract units per second (aups);
```

- **current position P and velocity V of each motor must be periodically updated:**

```
P(new) = P(old) + V(old) * dT + a * dT^2 / 2
V(new) = V(old) + a * dT
P -> TP
```

Using:

- current requested target position "TP";
- previous motor position "P(old)";
- motor velocity "V(old)", where |V| <= S_max_aups;
- velocity "a". It can have only fixed values:

```
 0        - zero acceleration;
 A_aupss  - "acceleration";
-A_aupss  - "deceleration";
```

- see "set sim dT" command to determine time period value.

Control protocol

- program read simple ASCII case insensitive text commands from standard input;
- commands are separated with new line character "\n";
- command execution must start immediately after command received;
- commands are received and executed in asynchronous mode, so next command can be received and executed before previous one execution finished (see APPENDIX B example);
- **commands are:**

Command	CONF	SIMS	Description
create motor **<motor_name>**	+		create new motor
create pen **<pen_name>**	+		create new pen
attach **<motor_name>** with **<axis>** of **<pen_name>**	+		attach motor to pen axis
set sim dT=**<sim_dT_s>**	+	+	set period for simulation state update calculation
set log dT=**<log_dT_s>**	+	+	set period for logging
set motor **<motor_name>** S=**<S_max_aups>**	+	+	set motor max speed
set motor **<motor_name>** A=**<A_aupss**>	+	+	set motor acceleration magnitude
set pen **<pen_name>** on|off	+	+	toggle on/off pen drawing mode
set motor **<motor_name>** P=**<pos_au>**		+	set motor target position
start	+		start simulation phase
stop		+	stop simulation and program exit

NOTE: see APPENDIX A for parameters types description.
NOTE: "+" means "allowed during this phase".
NOTE: spaces are delimiters of command subitems and for simplicity do assume that there always will be only one space between subitems.

Output

- each pen's position must be written to separate log file (text file) with name:

  <pen_name>.log

- each line in log file must be in format:

  <simulation_time_s>;<pen_axis_position_x>;<pen_axis_position_y>

- if pen is toggled on then program do writes pen's position to text file with given period (see "set log dT" command);
- if pen is toggled off then program must once write a string in format:

  <simulation_time_s>;-;-

  and do not write anything for this pen till it is toggled on again.

Initial states

- at start program runs to CONF phase;
- initial state of simulator: nor any pens and nor any motors are exist;
- initial state of new created pen:
  - pen is toggled on;
  - no motors attached to pen's axes;
- initial state of new created motor:

```
S_max_aups = 1.0 aupss;
A_aupss    = 1.0 aups;
TP         = 0.0 aup;
P          = 0.0 aup;
V          = 0.0 aups.
```

Notes

- motor shaft rotation assumed to be equivalent to pen's axis linear movements. So angular acceleration, velocity and position of motor shaft replaced here with axis linear acceleration, velocity and position;
- if motor attached to several axes then these axes acceleration, velocity and positions will be the same.

APPENDIX A: Types

```
<pen_name>            ::= <name>          - unique name of a specific pen
<motor_name>          ::= <name>          - unique name of a specific motor
<S_max_aups>          ::= <decimal>       - maximum speed value for specific motor
<A_aupss>             ::= <decimal>       - acceleration magnitude for specific motor
<simulation_time_s>   ::= <decimal>       - simulation world update time delta in seconds
<pen_axis_position_x> ::= <position>      - position of pen's x axis in a.u.
<pen_axis_position_y> ::= <position>      - position of pen's y axis in a.u.
<position>            ::= <decimal>       - motor (or pen) position in a.u.
<axis>                ::= x | y           - name of the axis

<name>                ::= string of character from set ['A'..'Z', '0'..'9', '_']
<decimal>             ::= decimal representation of real number with '.' for decimal point.
                          No Scientific form for number.
                          Value can be negative (with '-' sign).
```

APPENDIX B: Commands sequence and outputs example

PHASE	INPUT: COMMANDS	TIME s	OUTPUT: LEFT_HAND.log	OUTPUT: RIGHT_HAND.log	OUTPUT: DIAG.log
CONF	create pen LEFT_HAND	-			
	create pen RIGHT_HAND	-			
	create pen DIAG	-			
	create motor LX	-			
	create motor LY	-			
	create motor RX	-			
	create motor RY	-			
	create motor DIAG	-			
	attach LX with x of LEFT_HAND	-			
	attach LY with y of LEFT_HAND	-			
	attach RX with x of RIGHT_HAND	-			
	attach RY with y of RIGHT_HAND	-			
	attach DIAG with x of DIAG	-			
	attach DIAG with y of DIAG	-			
	set sim dT=0.1	-			
	set log dT=1.0	-			
	set pen LEFT_HAND on	-			
	set pen RIGHT_HAND on	-			
	set pen DIAG on	-			
SIMS	start	0	0;0;0	0;0;0	0;0;0
	…				
	set motor LX P=300	10	10;0;0	10;0;0	10;0;0
	…	11	11;4.4;0	11;0;0	11;0;0
	…				
	…	29	29;230;0	29;0;0	29;0;0
	set motor LY P=-2.3	30	30;234;0	30;0;0	30;0;0
	…	31	31;238;-0.1	31;0;0	31;0;0
	…				
	set pen RIGHT_HAND off	44	44;267;-2.3	44;-;-	44;0;0
	…	58	58;300;-1.9		58;0;0
	set motor DIAG P=35	59	59;300;-2.0		59;0;0
	…	60	60;300;-2.1		60;2;2
	…				
	…	89	89;300;-2.3		89;33;33
	…	90	90;300;-2.3		90;35;35
	set pen DIAG off	91	91;300;-2.3		91;-;-
	…				
FIN	stop	99	99;300;-2.3		

NOTE: position changes in this example are totally inaccurate – they're given only for the purpose of demonstrating the basic principles!