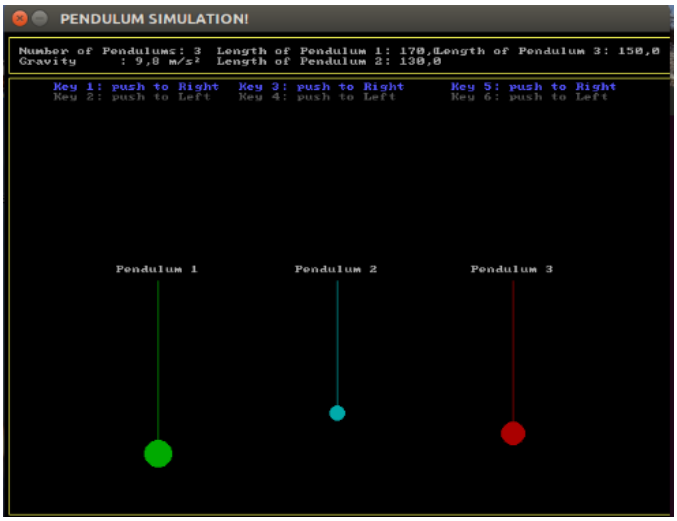
Real Time Systems Course Project
Title: Pendulum



Submitted to:

Prof. Giorgio Buttazzo

Prepared by:

Regassa, Yeneakal Girma

Matricula No: 539984

Email: yeneakalgirma28@gmail.com

Date 12/1/2016

ACKNOWLEDGEMENT

I would like to thanks my lecturer Professor Giorgio Buttazo for giving me this project in order to implement what i have learned in theory. And also I want to thank for the advising and supporting on project ideas.

Contents

1. Introduction

A **pendulum** is a weight suspended from a pivot so that it can swing freely. **Galileo Galilei** was the first to examine its unique characteristics. Galileo found that each pendulum has a constant period. The period is the time in which a pendulum completes a single oscillation, i.e., returns to the position it was in at the beginning of the period. When a pendulum is displaced sideways from its resting, equilibrium position, it is subject to a restoring force due to gravity that will accelerate it back toward the equilibrium position. When released, the restoring force combined with the pendulum's mass causes it to oscillate about the equilibrium position, swinging back and forth. The time for one complete cycle, a left swing and a right swing, is called the period. The period depends on the length of the pendulum, and also to a slight degree on the amplitude, the width of the pendulum's swing.

This document presents the design and implementation of the physics of Pendulum movement simulation. which is designed using C programming language and pthread library for multithreading and allegro library for graphics design. The simulation is intended to demonstrate the movement of pendulum and to show what will happen if the number of pendulum is more than one and collision occurs between them. The description of the project, what it looks like and how it works is thoroughly discussed. The user interface of the project is also presented.

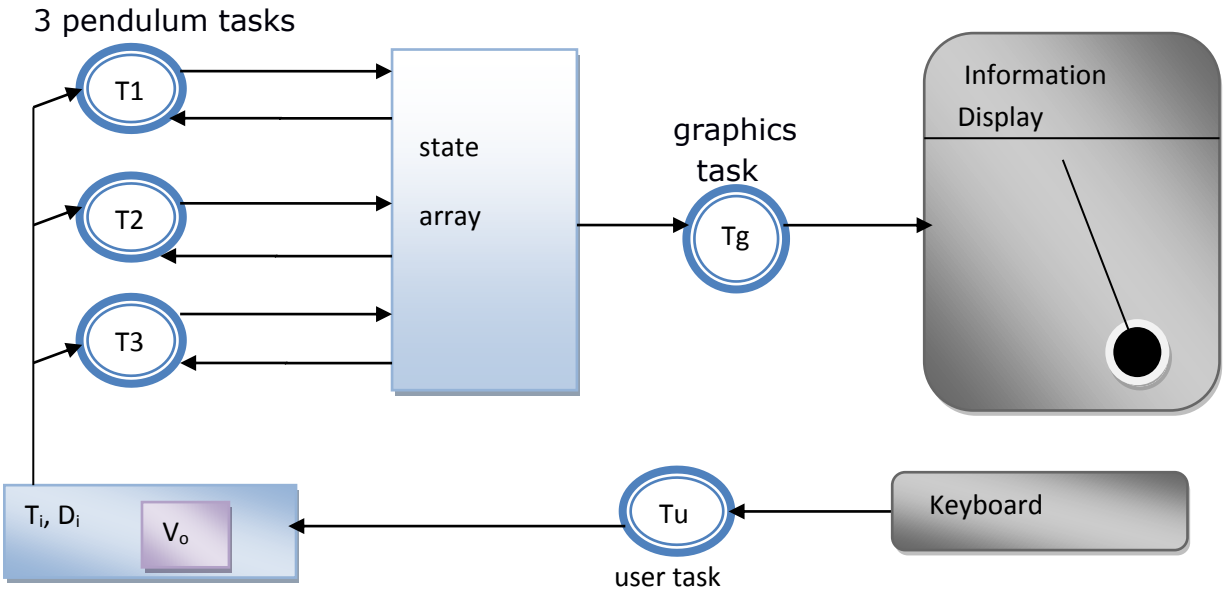
2. General Description of the project and requirements

The primary objective of this project is to know how to manage multiple threads with different period and deadline. Throughout this project there are different tasks which are managed by different threads for the simulation to work accurately. This section elaborates on the description of the simulation and user controls.

- Each pendulum is animated by a dedicated periodic task that updates its state to new position every period.
- The motion of each pendulum is independent on the other pendulum instead it depends on its current position.
- The program simulates air friction that affects the motion of pendulum by decreasing the amplitude through time.
- But If collision occurs between adjacent pendulums the pendulum position and movement direction will be affected according to their mass.
- The user can interact with the program by pushing a desired pendulum. when the program starts the pendulum is positioned at equilibrium position so the user pushes the pendulum by giving initial velocity. In addition the user can change the pendulum movement to be fast or slow by increasing and decreasing the angular velocity respectively.
- The system provides information like string length of each pendulum and gravity.

In my project for graphics display I choose the approach that each pendulum task draws the pendulum it controls every period. And for demonstration purpose I used three pendulums.

Application Structure



3. The User Interface

The user can push and affect the movement of each pendulum to make them faster or slower. I used two number for each pendulum one to push the pendulum to left and the other to the right.

- For Pendulum One press number 1 to push to right and press number 2 to push to left respectively.
- For Pendulum Two press number 3 to push to right and press number 4 to push to left respectively.
- For Pendulum Three press number 5 to push to right and press number 6 to push to left respectively.
- ESC key to exit from the program.

4. Shared Data Structures

There are two data structures that are defined in the project. I used Pthread Mutex functionality to guarantee mutual exclusion of access to this shared data structure.

a) **task_par** : which defines the real time parameter

```
struct task_par {
    int arg;                /* task argument */
    long wcet;              /* in microseconds */
    int period;             /* in milliseconds */
    int deadline;           /* relative (ms) */
    int priority;           /* in [0,99] */
    int dmiss;              /* no. of misses */
    struct timespec at;     /* next activ. time */
    struct timespec dl;     /* abs deadline */
};
```

b) **pen_structure :** defines the pendulum's parameters

```
typedef struct  pen_structure
{
     int   c;            /* color [1,15]*/
     float length;       /* length of the string(m)*/
     float mass;         /* bob mass*/
     float r;            /* radius of the bob (m)*/
```

```
float origin_x;   /* x coordinate for the pivot origin*/
float origin_y;   /* y coordinate for the pivot origin*/
float end_x;      /* x coordinate for string end and bob center*/
float end_y;      /* y coordinate for string end and bob center*/
float theta;      /* pendulum angle (rad)*/
float omega;      /* pendulum angular velocity velocity (m/s)*/
float alpha;      /* pendulum angular acceleration*/
}Mypendulum;
```

5. Tasks Description

Each task is scheduled by Linux real time policy, **SCHED_FIFO** (Fixed-Priority Scheduling + FIFO) scheduler which threads with the same priority are handled by a FIFO policy. A thread is executed until termination, cancellation, blocking, or preemption. The tasks(threads) in the pendulum program are as follows:

I. pendulum_update

This task controls the movement of the pendulum and basically do four functions. First of all it initializes all necessary parameters of the pendulum structure . It reads mass and length of the pendulum from the text file provided.

Secondly it calculates the next position of the pendulum in x and y coordinate using the following equation that governs motion of pendulum

$\frac{d2(\theta)}{d2t}$ = -g/L*(sin(ϴ))

where ϴ is the angle formed from the equilibrium position
g is the gravitational acceleration and
L is the string length.

When the program starts angular acceleration(alpha), angular velocity(omega) and the initial angle(theta) is initialized as zero. Then when the user presses a key that enables to push the pendulum means giving initial angular velocity.

Angular acceleration is calculated by
alpha = (-1 * g/length) * sin(theta)
Then the new angular velocity is calculated as
omega = omega_old + alpha

Then to simulate the air friction we multiply the angular velocity by some dumping constant that will slow down the motion of pendulum and eventually stops it.

omega = omega*Dump

finally we can get the new angle formed due to the new angular velocity using

theta = theta_old +omega

But as we see the pendulum movement is expressed in angle form but to draw into our computer screen we have to convert into Cartesian coordinate using basic trigonometric relations.

x = x_old + length * sin(theta)
y = y_old + length * cos(theta)

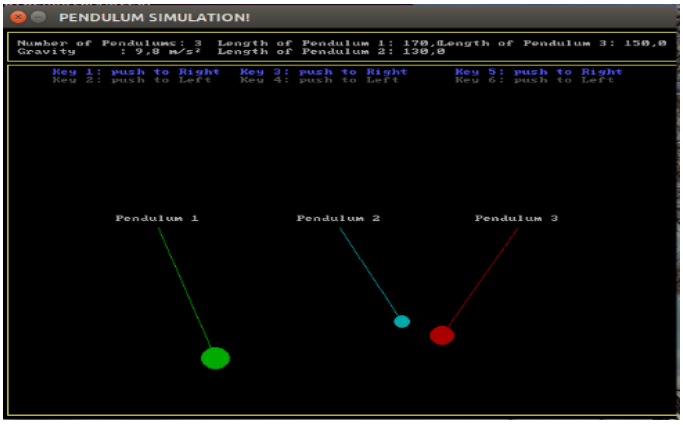
Thirdly this task calls a function which checks weather collision has been occurred between two adjacent pendulums or not and affects their movement accordingly.

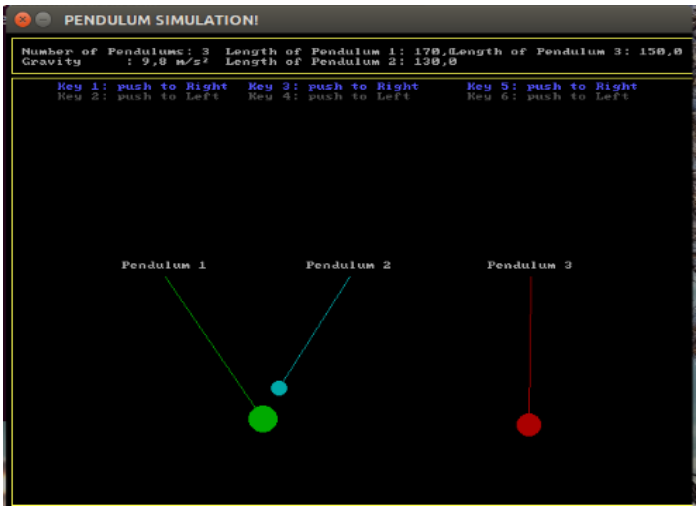
Finally this task draws the pendulum in the new x and y position. in addition when it wakes up in the next period before doing any computation it erases the previously drawn pendulum and continuous calculating the new position .

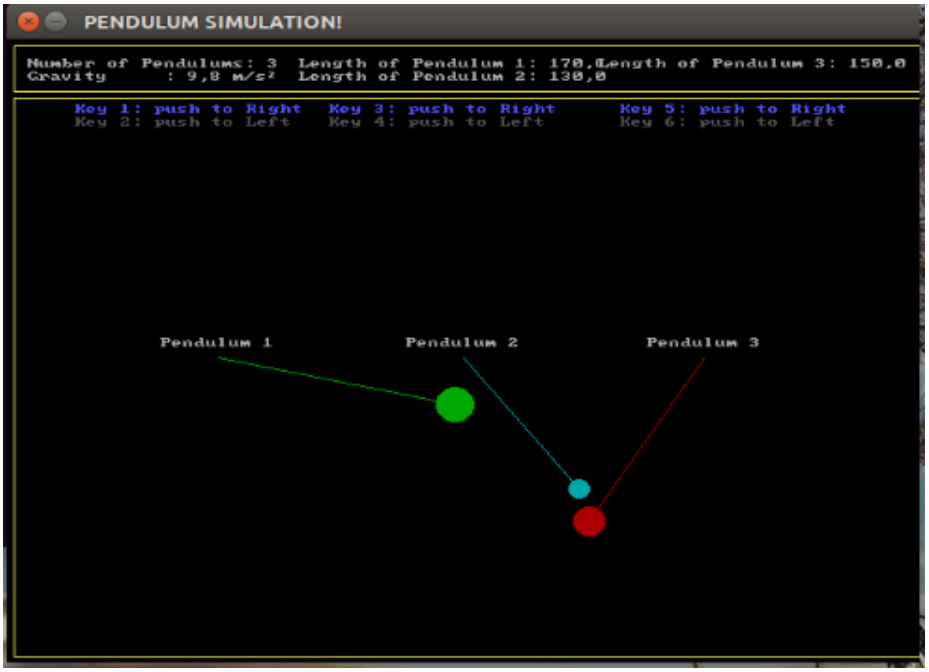
II. **input_task** : task which detects and interprets key press and changes the parameter value accordingly.

III. **display_task**: is a task which draws frames and displays basic information.

6. Set of Experimental result screen snapshots







7. Conclusion

The project pendulum is developed to implement set of periodic tasks with different period and deadline which help us to analyze the behaviour of multiple periodic tasks in real time environment.

8. References

https://computing.llnl.gov/tutorials/pthreads/

https://en.wikipedia.org/wiki/Pendulum

http://wiki.allegro.cc/index.php?title=Allegro_4_Tutorial

http://www.physicsclassroom.com/class/waves/Lesson-0/Pendulum-Motion