Inverted Pendulum 3d Engine

Vladan Vučković [1], Nikola Stojanović [2] and Vladimir Jakšić [3]

Abstract – **Inverted Pendulum 3d Engine is developed by the authors, as the result of some advanced research and real-time 3d simulation of an inverted pendulum. Nowadays, there are some practical applications and robots, controlled mostly using the microprocessor fuzzy controllers, generating the new interest for these machines. Our application has intention to simulate the movements of the inverted pendulum in real time and could interact with user during the generating process. It includes Inverted Pendulum Generator (written in Delphi) and 3d Pendulum Viewer. The application is standalone.**

Keywords – **Inverted pendulum, 3d simulation, programming, fuzzy logic, real-time computing.**

I. INTRODUCTION

The inverted pendulum (IP) problem is a classic control systems problem [1]. The problem of controlling and simulated an inverted pendulum has a very long history with approaches using linear or nonlinear dynamics, including both classical and fuzzy logic control techniques [2]. It has already been demonstrated that a fuzzy logic controller with the rules that can be learned by a genetic algorithms can control inverted pendulum system. Maintaining an equilibrium position of the pendulum pointing up is a challenge as this equilibrium position is unstable. As the inverted pendulum system is nonlinear it could be controlled by fuzzy logic. Typically a fuzzy controller is implemented in software running on microprocessor or microcontroller embedded in a device itself. In some applications, it is demonstrated how an integer-based fuzzy controller can be directly implemented as hardware in a Field Programmable Gate Array (FPGA).

Development of control techniques for inverted pendulum has been an interesting topic for engineers. This is largely due to its physical simplicity along with complete instability. Also these control techniques are applicable to control of unstable fast moving ground vehicle, robots and anti-seismic regulators for static buildings [3]. The control goal aims at keeping the IP at an upright position, despite the natural tendency of IP to fall on effect of a fast vertical oscillation applied to the pendulum base (Stephenson). Another control alternative is based on the application of a rotational torque to the pendulum base. Also, the control problem of inverted gyroscopic pendulum (GIP) is interesting, demonstrating how the control mechanism through a flywheel mounted at the top of the GIP governed by a fuzzy logic controller (FLC) achieves good stability and control performance of the system around vertical position.

Humans manage to balance the pendulums intuitively. Although, applying a gyroscopic motion based actuation at the top of the GIP pendulum is a novel idea. The mechanism is the same for the most of creatures walk and balance in everyday life, or when a person spreads their arms and rotates them rapidly to restore balance and keep from falling. There is always a process of learning various techniques based on previous goals set to balance the pendulum in a vertical position.

The main goal of our research is to construct useful and easy-to-control 3d tool for simulation and interactive control of an 3d inverted pendulum model. We used modular approach, so our solution could be used for simulation of the different kinds of similar devices and systems.

II. System Specification

The Inverted Pendulum Engine is stand-alone application. There are a two main executables that could be installed on same directory:

- INVERTPEN.EXE is main application. It is written in Delphi. This application is interface and main data generator.
- PENDULUM.EXE is 3d engine. Two other .dll files are connected with this .exe.

The system is based on these two separate programs connected via PENDULUM.TXT table that is presented in detail later (Fig. 1.). The user controls parameters in INVERTPEN.EXE and after pressing one button, the pendulum.txt is generated. The PENDULUM.EXE automatically loads data from disk and interpret them (convert them to 3d model movements).

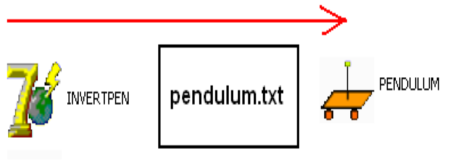


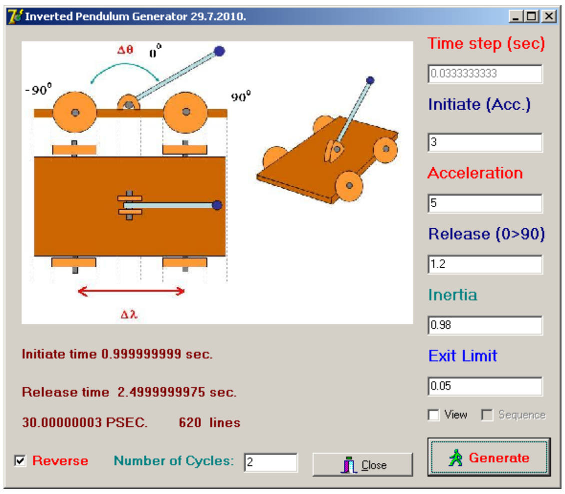
Fig. 1. Main application pipeline.

[1] University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, (e-mail: vladan.vuckovic@elfak.ni.ac.rs)
[2] University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, (e-mail: nikola.stojanovic@elfak.ni.ac.rs)
[3] JKP "Naissus", Nis, (e-mail: jaksha@gmail.com)

pendulum.txt is ASCII file, so there is possibility for other application different from INVERTPEN.EXE to generate it in appropriate format and to control PENDULM.EXE. In our opinion, this solution is adaptive and elegant.

A. Running the Application

The first step is to run INVERTPEN.EXE. The main application window is presented in the following picture (Fig. 2):



Time step (sec) 0.0333333333
Initiate (Acc.) 3
Acceleration 5
Release (0>90) 1.2
Inertia 0.98
Exit Limit 0.05
☐ View ☐ Sequence

Initiate time 0.999999999 sec.
Release time 2.4999999975 sec.
30.00000003 PSEC. 620 lines

☑ Reverse Number of Cycles: 2

Fig. 2. Interactive pendulum movement generator.

This application AUTOMATICLY run PENDULUM.EXE in its shell. So, after a few seconds, the 3d application initializes and appears in second window. Just after initialisation, PENDULUM.EXE is started to generate image animation, 30 frames per second, waiting data form INVERTPEN.EXE via pendulum.txt. Two applications are separate, and running individually. The user must close both of them, where decide to close system. The main procedure is as follows:

1. User changes working parameters in edit or check boxes.
2. User clicks on GENERATE button.

Then application generates movement table (pendulum.txt). This change is immediately recognized by PENDULUM.EXE and 3d engine starts to interpret it generating 3d animation. User could alter parameters while 3d engine works. New click on GENERATE button will initiate new animation. The application has following controls:

- <u>Time step:</u> fixed on 30 frames per second.
- <u>Initiate (acc.):</u> constant for pendulum initiation. In this stage pendulum accelerates.
- <u>Acceleration:</u> acceleration constant. Greater value for faster acceleration.
- <u>Release</u> (from +-90 degrees to zero): Speed of release. In this stages pendulum decreases speed.
- <u>Inertia:</u> When decreases speed, this is the factor that multiplies old speed. For instance, lower factors means that pendulum breaks roughly.
- <u>Exit limit:</u> This controls break exit loop.
- <u>Number of cycles:</u> determines number of working cycles for pendulum animation
- <u>Reverse:</u> Pendulum bar moves on both sides.
- <u>View Check:</u> Loads ASCII file PENDULUM.TXT.

Default values are optimal for animation.

B. Main ASCII file generator

The main part of the generator application is the ASCII pendulum movement generator. This procedure collects interactive data from the user interface. Along with this inputs, the procedure generates textual file that controls the 3d engine. The (incomplete) listing is as follows:

```
AssignFile(F,  'PENDULUM.TXT');
  Rewrite(F);
  {.}
  timestep:=strtofloat(form1.Edit1.Text);
  accelerate:=strtofloat(form1.Edit5.Text);

initiatestep:=strtofloat(form1.Edit2.Text);

realisestep:=strtofloat(form1.Edit3.Text);
  inertia:=strtofloat(form1.Edit4.Text);
  downexit:=strtofloat(form1.Edit6.Text);
  REPEATING:=strtoint(form1.Edit7.Text);
  form1.Label6.Caption:='Initiate time
'+floattostr(timestep*90/initiatestep)+'
sec.';
  form1.Label7.Caption:='Release time
'+floattostr(timestep*90/realisestep)+'
sec.';
  {.}
  angle:=0;
  time:=0;
  put:=0;
  basicput:=0;
  counter:=0;
  {.}
  FOR FF:=1 TO REPEATING DO
  BEGIN
  {.}
  {    P O S I T I V E    }
  {.}
  acctime:=0; BASICPUT:=PUT;
  {.}
  repeat
      {Writeln(F,  time, '      ', angle, '
', put);}
      Writeln(F,  time, char(9), angle,
char(9), put); inc(counter);
      {.}
      time:=time+timestep;
      acctime:=acctime+timestep;
      {.}
      angle:=angle+initiatestep;  if
angle>90 then angle:=90;
      speed:=accelerate*acctime;
      {.}
      deltaput:=put;
      put:=BASICPUT +
accelerate*acctime*acctime/2;     {!!!}
      deltaput:=put-deltaput;
{deltaput}
```

```
      {.}
 until (angle>=90);
  {.}
  {.}  {realise time}
  {.}
  IF FORM1.CheckBox3.CHECKED THEN
writeln(F,  ' ');  {<<<<<}
  {.}
  repeat
      {Writeln(F,  time, '      ', angle, '
', put);}
      Writeln(F,  time, char(9), angle,
char(9), put);  inc(counter);
      {.}
      time:=time+timestep;
      angle:=angle-realisestep;  if angle<0
then angle:=0;
      put:=put+deltaput;
      deltaput:=deltaput*inertia;
      {.}
  until (angle<=0);
  {.}
  {.}  {slow down time}
  {.}
  IF FORM1.CheckBox3.CHECKED THEN
writeln(F,  ' ');  {<<<<<}
  {.}
  repeat
      {Writeln(F,  time, '      ', angle, '
', put);}
      Writeln(F,  time, char(9), angle,
char(9), put);  inc(counter);
      {.}
      time:=time+timestep;
      put:=put+deltaput;
      deltaput:=deltaput*inertia;
      {.}
  until (deltaput<downexit);
  {……}
  end;
    {.}
  CloseFile(F);
```

So, this text file drives the Virtuls 3d viewer.

III. 3D Application for Visualizing of the Inverted Pendulum

The 3d application gets control data from the interactive generator and transforms them to the 3d movements. This function is assigned to the ***PENDULUM.EXE*** application. Application is written in C++.

A. Visualization

The application ***PENDULUM.EXE*** consists of one main window that can be full screen or in windowed mode. **Windowed mode** is used for testing in real time. In windowed mode application looks like Fig. 3. There are two buttons in the left upper corner:

FullScreen / Window mode switch is for changing the overall appearance of the application.
Screen resolution is used for looping through the set of four window sizes and resolutions (1024x768, 800x600, 640x480, 512x384) and is visible only in windowed mode. **Full screen** is for presentation purposes and it comes in 1024x768 resolution. The close button terminates the application.

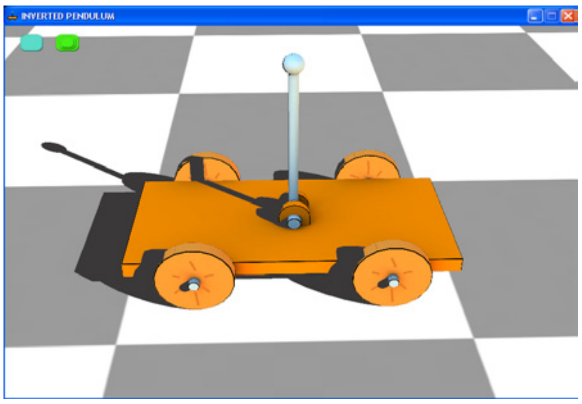


Fig. 3. Pendulum 3d model.

B. Running

Application is interpreting the data that are generated with other application. The data are written to the PENDULUM.TXT file. When data is generated with other application it sets the value of 1 in the START.TXT file. 3D application is constantly checking the START.TXT file for this change to occur. After it receives the start signal it sets the value of 0 to the START.TXT file and loads the generated data (Fig. 4.).

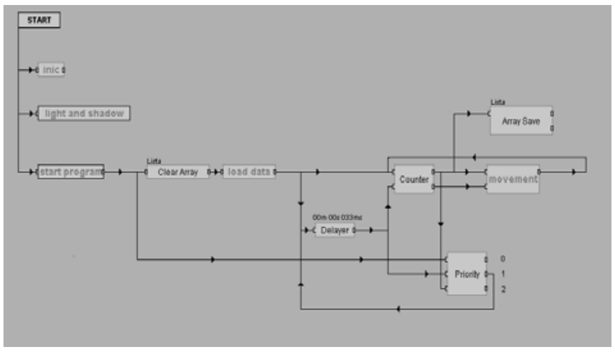


Fig. 4. Pendulum 3d viewer flow chart.

When application is finished loading data, it starts to interpret it and the rendering begins.

C. Controls

Application also use keyboard controls:

ESC - button terminates the program.
F5 - button changes the resolution in windowed mode.
P - button pauses and plays the application.
Page UP - zooms in the camera.
Page Down - zooms out the camera.
- Orbits the camera up
⇕ - Orbits the camera down
-
⇦ - Orbits the camera left
-
⇨ - Orbits the camera right

D. Error Messages

If there is an error in the application, the following error messages can occur:

- ERROR: READING START.TXT FILE - there is no start.txt file in the folder where pendulum.exe is located.
- ERROR: WRONG DATA IN START.TXT - there is wrong character in the start.exe file.
- ERROR: CANNOT WRITE TO START.TXT FILE - there is no start.txt file in the folder where pendulum.exe is located, or the file is writing protected.
- ERROR: READING PENDULUM.TXT FILE - there is no pendulum.txt file in the folder where pendulum.exe is located.

IV. Data Format

All applications use simple ASCII data structure to commutate information

A. Input Data

Data in PENDULUM.TXT file must be organized in the following order:

```
0.00000000000000E+0000  0.00000000000000E+0000
0.00000000000000E+0000
 3.33333333000000E-0002  3.00000000000000E+0000
2.77777777222222E-0003
 6.66666666000000E-0002  6.00000000000000E+0000
1.11111110888888E-0002
 9.99999999000000E-0002  9.00000000000000E+0000
2.49999999500000E-00020
```

The first column contains generated time line. The second column contains the angle from the up position of the pole. Angles greater then 0^o moves the pole to the right and angles smaller than 0^o moves the pole to the left. Pole is rotating around Y coordinate (Fig. 5.).

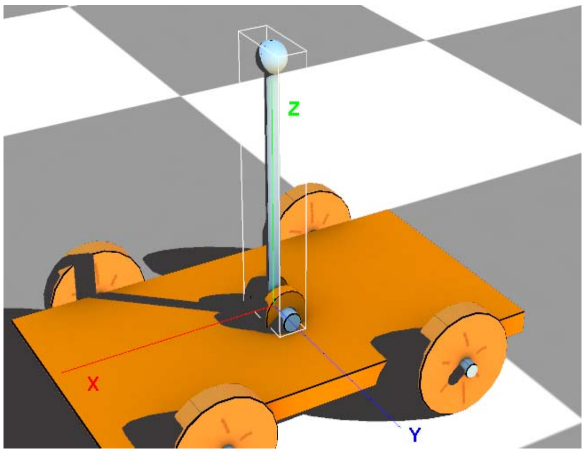


Fig. 5. Pendulum XYZ coordinates.

The third column is the absolute path that is covered. Between the columns there must be the TAB character "Char (9) " for spacing.

B. Output Data

The application generates " I_PENDULUM.TXT" file and stores it in the same folder where the application is placed. It contains interpreted values of the PENDULUM.TXT files. It is used for checking the application.

```
00m 00s 033ms: 0: 0
00m 00s 033ms: 3: 0.00277778
00m 00s 044ms: 6: 0.0111111
00m 00s 046ms: 9: 0.025
00m 00s 033ms: 12: 0.0444444
00m 00s 033ms: 15: 0.0694444
00m 00s 035ms: 18: 0.1
00m 00s 033ms: 21: 0.136111
…
```

V. Conclusion

In this paper, the application Inverted Pendulum 3d Engine is presented in detail. The application simulates the movement of inverted pendulum in real time. It is very useful for educational and research purposes. The implemented two-stage application structure could be generally used for simulation of a different kinds of simple machines, like motors, generator etc. The pipeline and application structure enables that 3d viewer could be run separately, from different application, according to user's demands.

Acknowledgement

Authors appreciate support and suggestion from Prof. Claudio Moraga (Fakultät für Informatik, Technische Universität Dortmund, Germany and prof. Radomir Stankovic (Faculty of Electronic Engineering Niš, Serbia).

References

[1] J. Lam, "Control of an Inverted Pendulum", University of California, Santa Barbara, 10 June 2004, http://www.ece.ucsb.edu/~roy/student_projects/Johnny_Lam_report_238.pdf.

[2] T. Takagi, and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control, " IEEE trans. on systems, man, and cybernetics, vol. 15, no 1, pp. 116-132, 1985. Engineering Letters, 18:1, EL_18_1_02

[3] R. Ooi, "Balancing a Two-Wheeled Autonomous Robot", University of Western Australia, 3 Nov. 2003, http://www.cs.cmu.edu/~mmcnaugh/kdc/as7/2003-Balance-Ooi.pdf. 1992.