Physics Behind the Simulation: A CS296 Group 31 Report.

Sanchit Garg
110050035
sanchitgarg@cse.iitb.ac.in

Ravi Kumar Roshan
110050033
sraviroshan@cse.iitb.ac.in

10 April, 2013

1 Introduction

A Rube Goldberg machine, contraption, invention, device, or apparatus is a deliberately over-engineered or overdone machine that performs a very simple task in a very complex fashion, usually including a chain reaction. The expression is named after American cartoonist and inventor Rube Goldberg (18831970). [3]

This is a small Project Report about the simulation and contraption using C++ Simulator engine Box2D under the CS296 Lab Course.

Here we first describe the design of the simulation followed by its timing and Profiling issues.

2 Design

First we start with implemented design. Design is very simple. Implemented and original design are almost same. We then describe some minor changes we have done in original design. [1]

2.1 Implemented Design

Here is the image of implemented design.

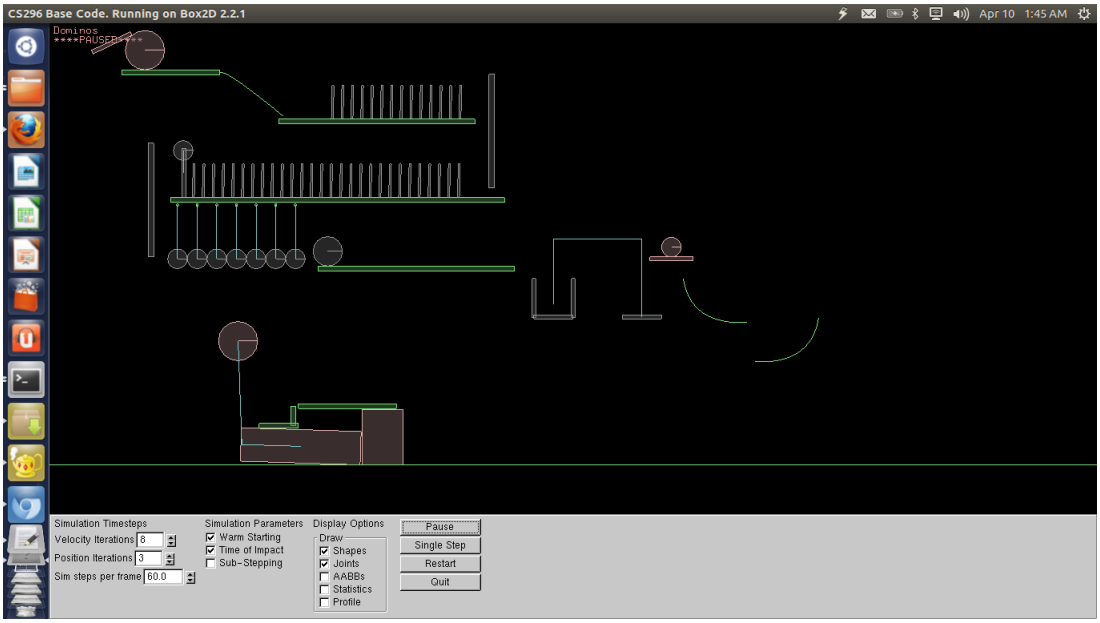

Fig.1

First a Motored plank. It starts revolving and strikes a ball resting on a platform. This ball rolls down from a slope and hits the train of Dominos standing on horizontal Platform. Dominos start falling. There is a rotating shaft near the end of this domino train. It gets hit by the dominos. This shaft while rotating hits another train of Dominos standing on another horizontal platform just below the first one. On the end of this domino train there is another rotating shaft which do the same purpose that as of the previous one. This rotates on hit by dominos. On the way of its rotaion Newton's Cradle is kept. Shaft hits the Newton's Cradle. Near the end of the Cradle a ball is kept. This ball get struck by the cradle and rolls horizontally and finally enters in pully system. There is a Bucket(open box) into which the ball falls. When this ball falls, the bucket becomes heavier and starting coming dowm and as a result the other side of pulley which is connected by a horizontal plank, starts moving up and make rotates the another horizontal platform. There is a ball resting on this platform which falls down due to movement of platform. This ball follows a zig-zag path on two curves and falls on ground and moves towards a piston. Finally the ball hits the piston. On the mouth of piston there is a block attached with a balloon. This balloon is basically a ball whose gravity is reversed. Ball coming towards the piston hits and balloon moves up along with block. [2]

2.2 Original Design

There are 3 changes apart from body's structure :

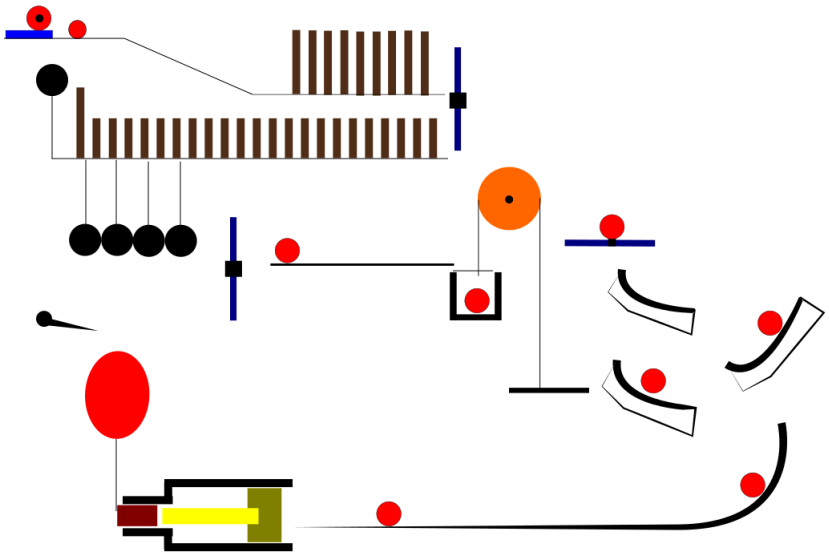

Fig.2

1. Motored Plank

   Instead of motored plank We thought of a revolving ball which is in contact with a plank. As ball rotates plank moves and hit a ball and continue.

2. Second Revolving Shaft

   Instead of second revovling shaft We planned that at the end of the second dominos train there is a inverted pendulum standing and on hit it moves down and starts the motion of Newton's Cradle.

3. Bursting of Balloon

   We thought that as the balloon rise it would come into the contact of a pointed thing and will burst.

2.3 Interesting Features

Interesting features are:

1. Newton's Cradle

   Newtons Cradle may look like interesting but in simulation the exact effect of it has not achieved i.e only end balls move.

2. Rising Balloon

   Rising Balloon (acieved by reversing its gravity) may look like interesting.

3 Timing Analysis

Timing Analysis is as follows:

3.1 Observations

Following are the observations while doing timing analysis.

1. Plot1: Loop time increases with no of iterations.

   It is obvious. As no. of iterations increases the time taken to simulate more steps increases.

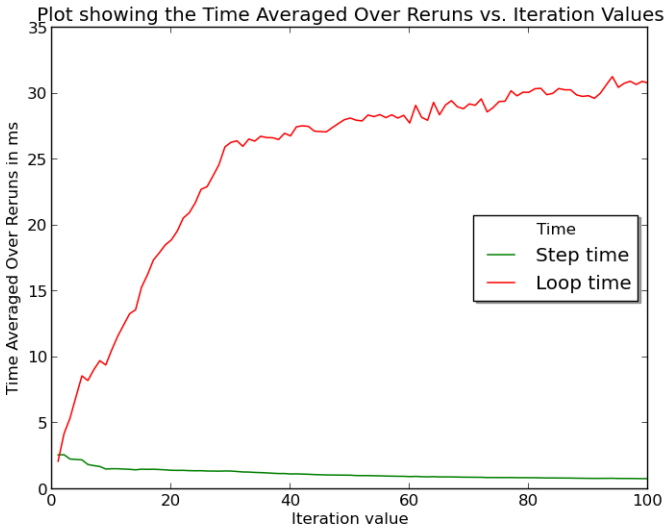

Fig.3 Plot01 : step and loop time averaged over all reruns for various iteration values

2. Plot1: Graph segmented into two linear parts for loop time for both load and no-load graph

   (a) 0-30 : higher slope, less deviation from linear; and

   (b) 30-100 : lesser slope, more fluctuations from being linear.

   As the no. of iteration increases more work has to be done by the system, so system provides more resources to this and gives priority to this process. Hence more work is done by the system to run the process and hence simulation runs relatively faster so average time decreases. We have run for iteration no. 5000,10000 and observed the avg. step time decreases.

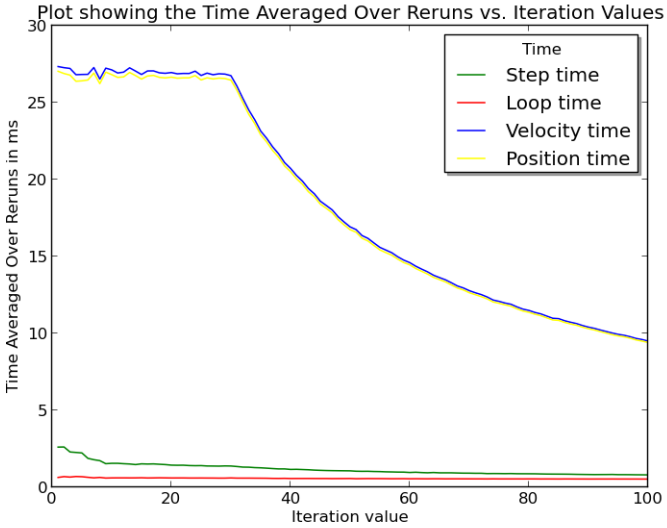

Fig.4 Plot02 : step,collision,position and velocity update time averaged over all reruns for various iteration values

3. Plot2: Times(Step, Velocity, Position, Colllision) for various iterations averaged over reruns decreases as iteration value increases.

   It may be possible that as simulation proceeds ahead the simulation steps are performed faster or the amount of work the simulation is doing get decreased in further steps. Or it may be possible also that initially the application requires some amount of time to stablize the process and hence takes more time in the beginning than afterwards. As a result time for single step averaged over 100 reruns decreases as no. of iteration value increases. Also if we run simulaiton for higher values let say 10000 or 50000 for various reruns, we can verify the trend. It is due to compiler optimison also. As no. of iteration increases, amount of work also increases. Same work is not computed many times by compiler. So averaged time decreases.

Fig.5 Plot03 : step and loop time averaged over all iterations for various rerun values

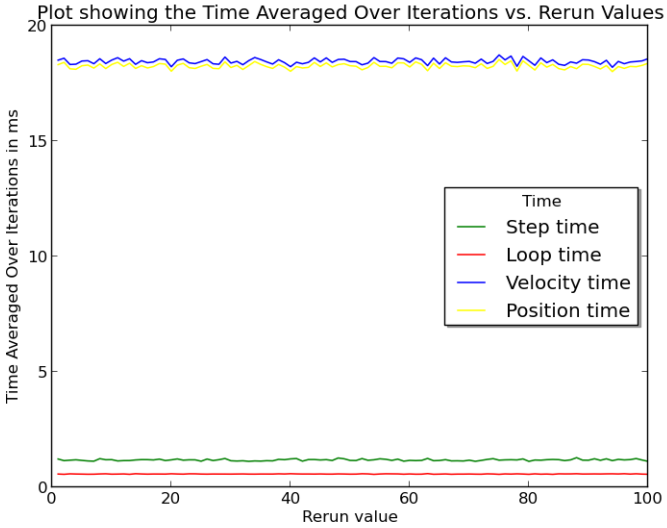

Fig.6 Plot04 : step,collision,position and velocity update time averaged over all iterations for various rerun values

4. Plot3 and Plot4 as load increases these plots fluctuate readily as compared when load is low and relative values is large in first one as compared to second

   As more and more programs run on machine, the process of memory and other resource allocation and usage fluctuates more. So as the system get more and more busy, efficiency and performence of the processes running on the machine is affected and get degraded.

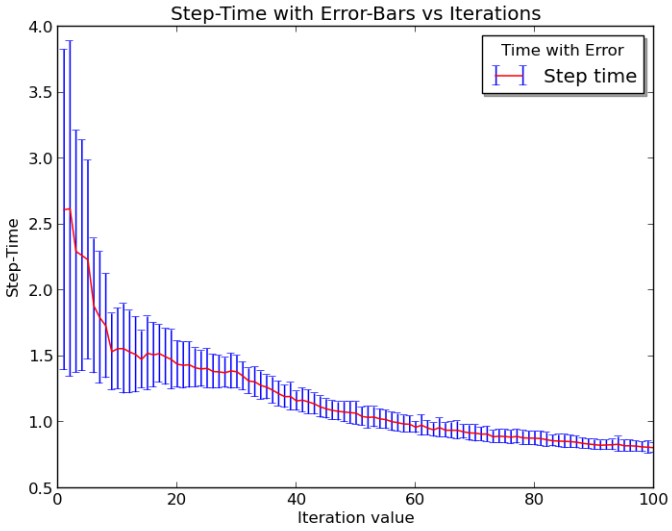

Fig.7 Plot05 : step time for various iteration values with error bars

5. Plot5: Comparing error bars

   (a) Length of error bars corresponding for the values is relatively low when load is low as compared when load is high.

   As described above, as the machine is highly loaded with so many processes, the fluctuations of time calculation and other processes are very high and there is very high chances of error in calculations and it happens as we compare the two plots.

(b) Window of error bar decreases as we increase no. of iterations.
As no. of datasets increases in a sample, error(Standard Deviation) decreases. This can also be inferred from Central Limit Theorem.

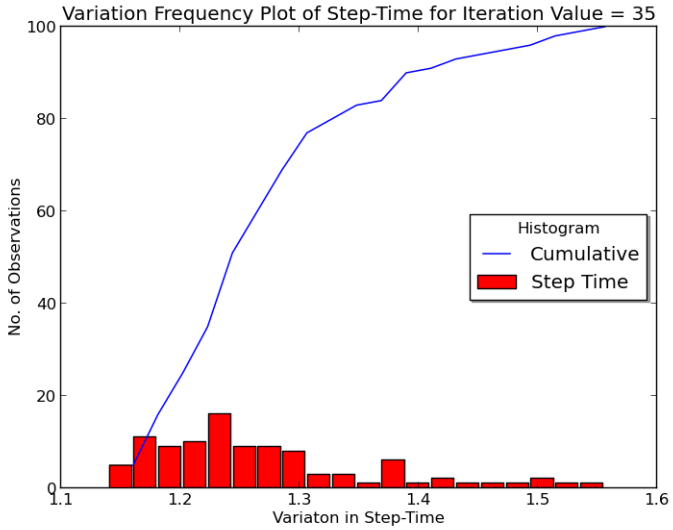

Fig.8 Plot06 : frequency values as a bar graph and the cumulative frequency values as a curve

6. Plot6: Frequency plot is baised towards left.
As in plot2 we saw that no. of low averaged value is more than high averaged value (because step time decreases readily in beginning and then gradually decreases) therefore we can say and observes that frequency plot is baised towards left.

4 Profiling

A profiler is used to identify what are the parts of your code are taking the maximum time. This helps in identifying parts of your code that can be optimized. Here below is the profiling done first with optimization i.e with release mode on and second with debugging mode on. For profiling we have used perf tool.

4.1 Observations

Analysis has been done on the iteration no. 200, 1000 and 5000.

1. For Iteration no. 200
In debug mode maximum time is taken by ”b2ContactSolver::SolveVelocityConstraints()” approx. 19.55% while in release mode maximum time is taken by ”b2World::Step(float, int, int)” approx. 11.13% but next is ”b2ContactSolver::SolveVelocityConstraints()” approx. 9.15%.

2. For Iteration no. 1000
In debug mode maximum time is taken by ”b2ContactSolver::SolveVelocityConstraints()” approx. 9.55% while in release mode maximum time is taken by ”b2EdgeSeparation(b2PolygonShape const*, b2Transform const&, int, b2PolygonShape const*)” approx. 20.13% but next is ”b2ContactSolver::SolveVelocityConstraints()” approx. 15%.

3. For Iteration no. 5000
In both mode maximum time is taken by ”b2ContactSolver::SolveVelocityConstraints()” approx. 6.33% in debug mode and 18.20% in release mode.

4.2 Call Graphs

The call graph shows how much time was spent in each function and its children. Here is the call grpahs for both release and debug mode for iteration no. 200. We can see clearly the complexity of the running the simulation in both the cases. There are two call graphs.

1. Debug Profile

   Debug Call graph is more complex.

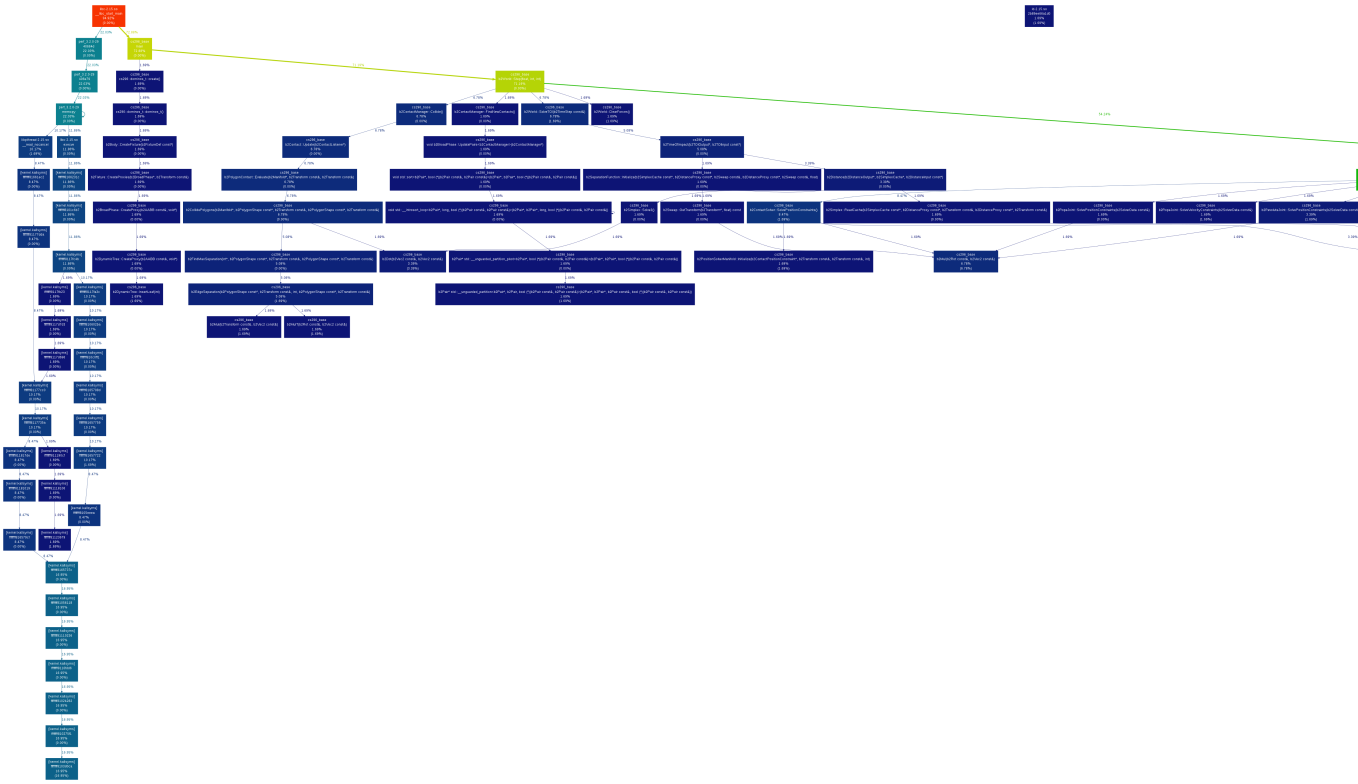

Fig.9 : Call Graph for Debug Mode:Itr value 200

2. Release Profile

   Release Call graph is more simple.

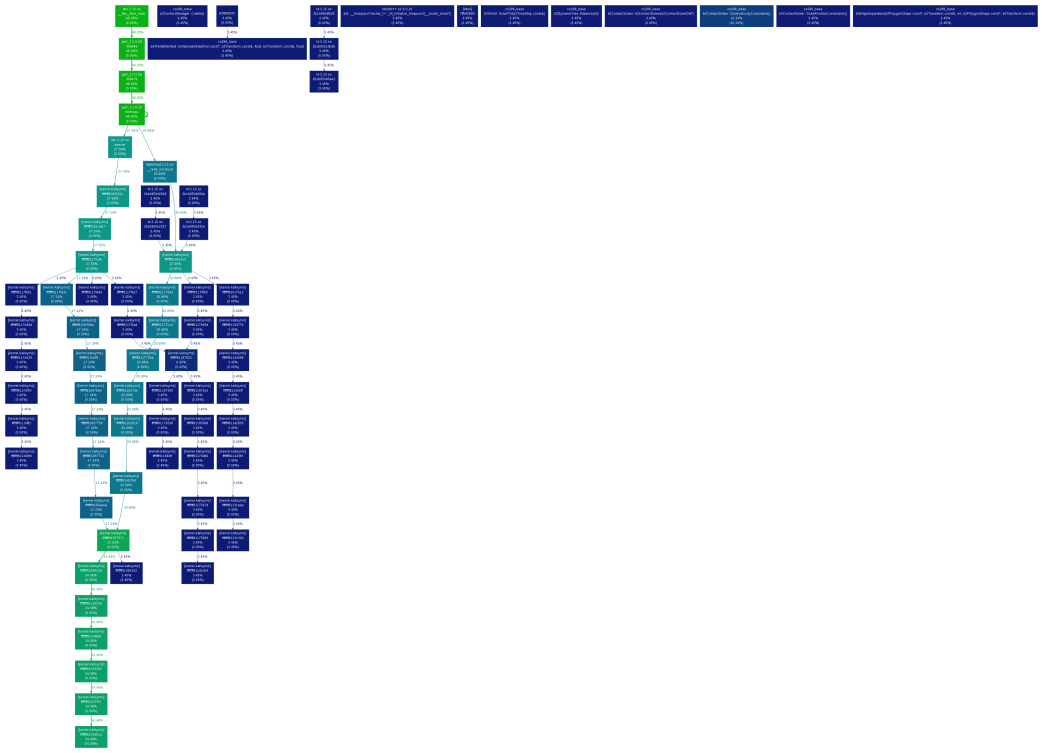

Fig.10 : Call Graph for Release Mode:Itr value 200

4.3 Conclusion

1. We can see that number of functions calls for constructors are reduced drastically in the release part as compared to debug part and hence indicates that optimization ags work on constructors.

2. Here is a consistent observation. Most of the time is taken by method ”b2ContactSolver::SolveVelocityConstraints()”. Hence there is a possiblity of implementation optimization of this function in Box2D Library.

5 Conclusion

Thus in the report we have analyzed and timed the code under different conditions and summerized the results.

References

[1] Kleppner and Kolenkow. *An Introduction to Mechanics*. McGraw-Hill, 1973.

[2] Kleppner and Kolenkow. *An Introduction to Mechanics*. McGraw-Hill, 1973.

[3] Wikipedia. *Rube Goldberg Machine*. Wikipedia, 1973.