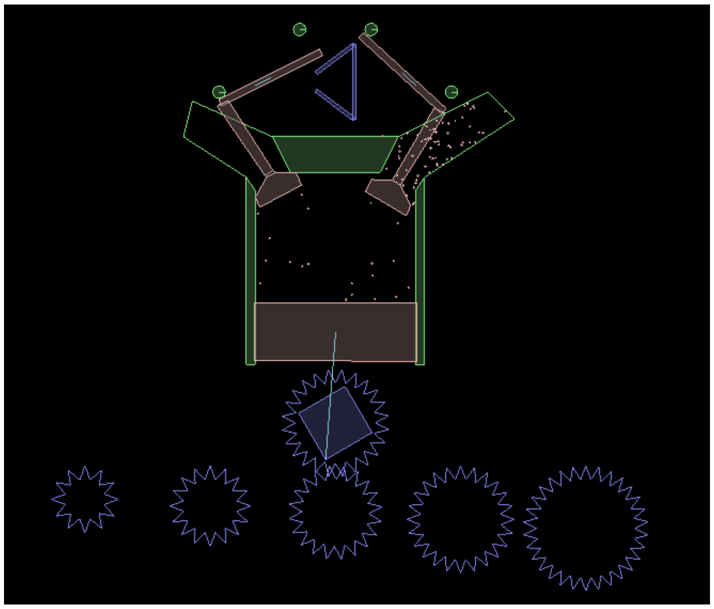
PROJECT REPORT

GROUP 19

inficoders

Rube Goldberg Machine
Simulation of 4 Stroke Combustion [?] [?] Engine
With Gears
by Naveen,Yathansh,Rajat



screenshot 4stroke combustion engine

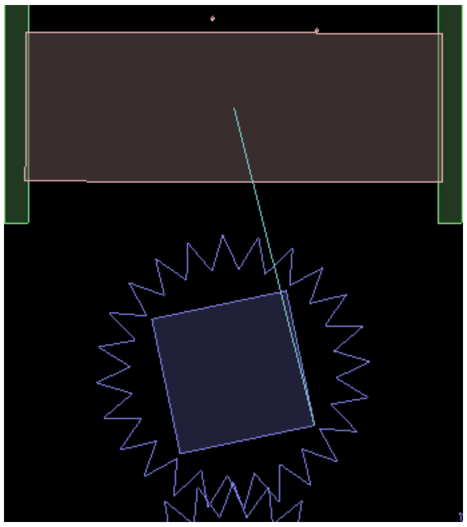
Introduction

In this project, we have developed a simulation for the 4-stroke combustion engine. [?] [?] Along with the main engine, we have attached 5 speed gears to it, out of which one can be attached to the engine at a time.

Structure:

The main body consists of the combustion engine. It has the following components:

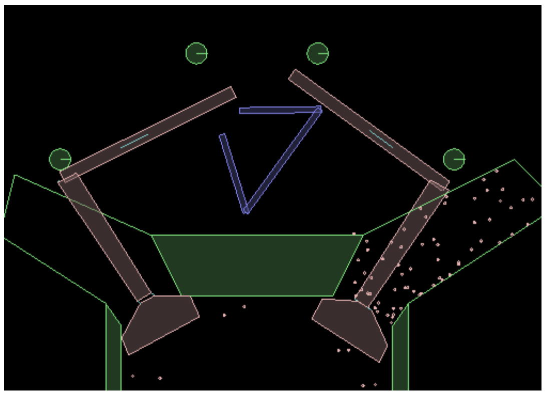
- **The Main Piston System**
  The main piston is the component which is used for the compression of the air in the compression chamber. This component is made up of a rectangular object, i.e., the piston, and the rotating square object which controls the motion of the piston. These two objects are attached with a distance joint. This ensures that the up and down movement of the piston results in the rotation of the square object, and vice-versa. The piston is fixed horizontally inside the compression chamber by two stands on each side, and its vertical motion is fixed by the rotation of the square object. The square object has a main gear attached to it, which rotates with the same angular speed as the rotating square.



main piston

- **The Left and Right valve System**

  The left and right valves are separate systems, and are dependent on gravity and a rotating triangle for their movement. the valves consist of three polygon-shaped objects attached with joints. The lowest part is a six-sided figure, and serves as the main part of the valve. The opening and closing of the valve depends upon the forward and backward motion of this object. The middle object is a rectangular stick attached singly to the lower object, and provides support for the motion of the lower object. Also, we have made use of the sensor data member, to make the middle object invisible to the air particles, and thus prevents any collisions there. The top object is another rectangular stick, which is attached doubly to the middle object. This double attachment makes the joint stiff and immovable and not rotatable. The density of this upper object is high, and it is pivoted at a point to the left of the centre of mass of this object. This fixed rotation at a point off-centre helps gravity determine the motion of the object, and hence the whole valve. There are two spherical fixed objects, which restrict the motion of the upper body. The rotating triangle has an angular velocity with magnitude exactly half of that of the rotating square and direction opposite to it. The two upper parts of both the valves are predominantly forced by gravity to lean inside, keeping both the valves closed. Whenever the two vertices of the largest side of the triangle come in contact with these upper bodies, they force them to move in the opposite direction, forcing the valve to open for as long as the vertices and the sticks are in contact. This process is an important part of the 4-stroke cycle. The initial position of the rotating triangle is set up so that it is in sync with the motion of the piston, and hence is necessary for the completion of the 4-stroke cycle.

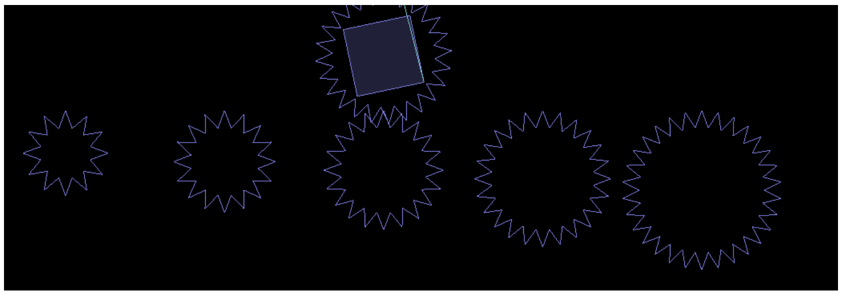

valve system

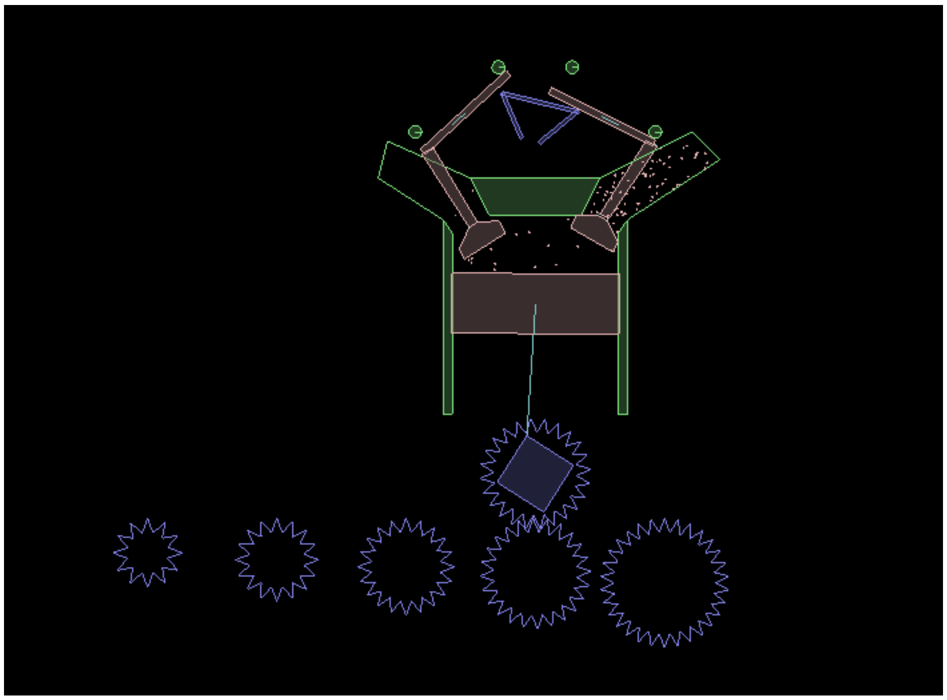
- **The Supporting Structures**
  The compression chamber is confined on both the sides by the left and the right stands, and from the top by a trapezoidal structure. These structures ensure that the compression chamber remains closed. Also the stands and the trapezoidal structure are placed such that they do fix the position of the lower part of the valve in the closed-valve state.

- **The Inlet/Outlet Chambers and Air Particles**
  There are two chambers on the left and right of the compression chamber, which act as outlet and inlet respectively. The inlet chamber allows air particles to move to the compression chamber through the right valve. Similarly, the outlet chamber receives the compressed air from the compression chamber through the left valve. This completes the 4-stroke cycle. The air particles that are received in the outlet chamber are transported back to the inlet chamber. This ensures the continuation of the simulation. To accomplish this, we have implemented the contactListener class, and kept the last wall of the outlet as a contactListener object. Whenever an air particle comes into contact with this object, its coordinates are transformed, and it is transported back to the inlet chamber.

- **The Gears**
  There are 5 gears, and they are located below the combustion engine. At a particular time, exactly one of the gears is in contact with the main gear, that is attached to the rotating square object. All the

gears have different sizes, but same size of spokes, thus resulting in different number of spokes on each gear. They are placed in increasing order of size and therefore number of spokes from left to right. This means that the rightmost gear is the first gear and the leftmost gear is the fifth gear. As it happens in real life, the first gear provides the least speed and maximum torque, whereas the fifth gear provides maximum speed and least amount of torque. These gears are implemented by creating fixtures containing multiple lines, that are criss-crossed, thus giving the shape of the gears. We had to assign a fixed angle to each spoke in each gear, so that the size of the spoke remained same in all the gears. Correspondingly, the number of such spokes increased on increasing radius and angle for each spoke decreased.


gear


when air is compressed

Features

1) The speed of both the controlling gears can be increased or decreased by using the '+' and '-' keys on the keyboard. While high speed resembles an actual combustion engine, lower speed allows us to analyse and study the simulation carefully.

2) Also, the gears can be shifted one at a time, by using the 'left' and the 'right' arrow keys. This is implemented by making the appropriate changes in the file 'callbacks.cpp'. We assigned functions to transform the co-ordinates of the gears based on the keyboard input.

3) The process of reusing air molecules by their instan-

taneous transportation from outlet to inlet chamber is clever and remarkable.

4) The creation of the gear-shaped object for gears, is a remarkable feat, since it was created by us.

Limitations

1) At high speeds, the air molecules start ”leaking” from the engine. This happens because (we found this as a plausible reason on the internet) Box2D has a certain interval after which it detects collisions. Then, if a collision occurs in between this interval, then Box2D is not able to detect it and it goes unnoticed. This results in the so-called ”leaking” of air molecules. We could find no way of preventing this.

2) The vertices of the rotating triangle should collide with exactly one of the two upper parts of the valves, and should remain invisible to other. We were not able to accomplish this, resulting in an extra opening and closing of the valves in one cycle. We tried to use the group index feature, to implement this, but met with some or the other error at all points.

3) The main rotating gear is a kinetic object. In its motion, it should collide with other gears, and cause them to move. But in our implementation, maybe because that shape was our own creation (again, this is only a plausible explanation we found on the internet), we could not

make it interact with other objects, i.e., it became invisible to collisions. We could find no way to make it visible to collisions.

Some Difficulties and How We Overcome Them

1) Any place where we required the use of concave/non-convex objects, we had to create smaller objects and join them. e.g. valves, rotating triangle.

2) First we wanted to continuously create new air molecules and delete the molecules that reached the outlet. We tried many ways of doing this, but failed everytime. Then we found out about the contactListener class, and thus implemented a method to transport the air molecules from outlet back to the inlet.

3) We overcame the difficulty of the non-colliding nature of the main gear, by manually providing a function which kept the appropriate angular velocity of the gears, so that the rotation matched with that of the main gear.

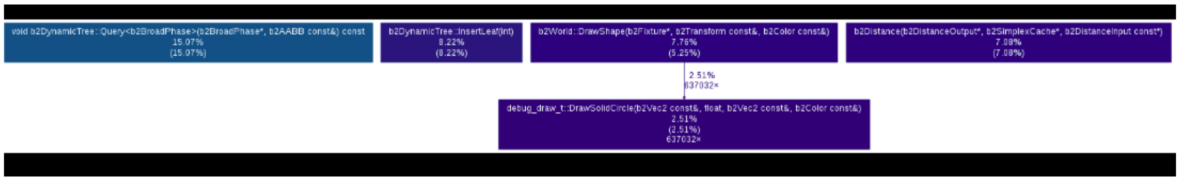
Callgraph:Debug Mode

debug mode

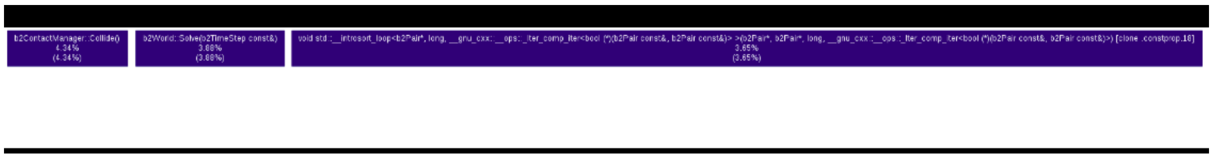
Callgraph:Release Mode

release mode

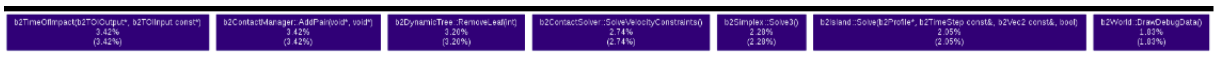
zoomed callgraph of release mode are as follows:


void b2DynamicTree::Query<b2BroadPhase>(b2BroadPhase*, b2AABB const&) const
15.07%
(15.07%)
b2DynamicTree::InsertLeaf(int)
8.22%
(8.22%)
b2World::DrawShape(b2Fixture*, b2Transform const&, b2Color const&)
7.76%
(5.25%)
b2Distance(b2DistanceOutput*, b2SimplexCache*, b2DistanceInput const*)
7.08%
(7.08%)
2.51%
637032×
debug_draw_t::DrawSolidCircle(b2Vec2 const&, float, b2Vec2 const&, b2Color const&)
2.51%
(2.51%)
637032×

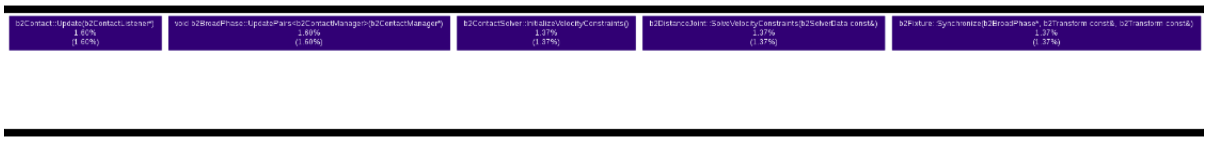
release mode


b2ContactManager::Collide()
4.34%
(4.34%)
b2World::Solve(b2TimeStep const&)
3.88%
(3.88%)
void std::__introsort_loop<b2Pair*, long, __gnu_cxx::__ops::_Iter_comp_iter<bool (*)(b2Pair const&, b2Pair const&)> >(b2Pair*, b2Pair*, long, __gnu_cxx::__ops::_Iter_comp_iter<bool (*)(b2Pair const&, b2Pair const&)>) [clone .constprop.18]
3.65%
(3.65%)

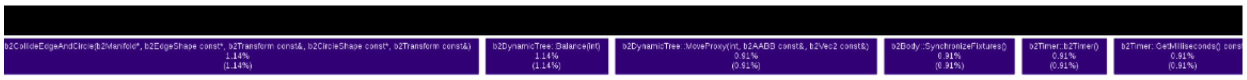
release mode


b2TimeOfImpact(b2TOIOutput*, b2TOIInput const*)
3.42%
(3.42%)
b2ContactManager::AddPair(void*, void*)
3.42%
(3.42%)
b2DynamicTree::RemoveLeaf(int)
3.20%
(3.20%)
b2ContactSolver::SolveVelocityConstraints()
2.74%
(2.74%)
b2Simplex::Solve3()
2.28%
(2.28%)
b2Island::Solve(b2Profile*, b2TimeStep const&, b2Vec2 const&, bool)
2.05%
(2.05%)
b2World::DrawDebugData()
1.83%
(1.83%)

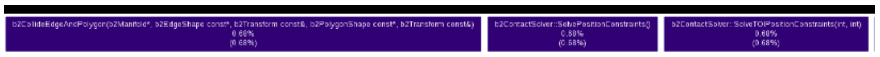
release mode


b2Contact::Update(b2ContactListener*)
1.60%
(1.60%)
void b2BroadPhase::UpdatePairs<b2ContactManager>(b2ContactManager*)
1.60%
(1.60%)
b2ContactSolver::InitializeVelocityConstraints()
1.37%
(1.37%)
b2DistanceJoint::SolveVelocityConstraints(b2SolverData const&)
1.37%
(1.37%)
b2Fixture::Synchronize(b2BroadPhase*, b2Transform const&, b2Transform const&)
1.37%
(1.37%)

release mode

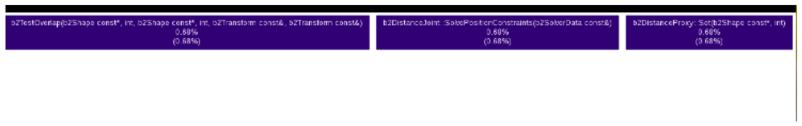

b2CollideEdgeAndCircle(b2Manifold*, b2EdgeShape const*, b2Transform const&, b2CircleShape const*, b2Transform const&) 1.14% (1.14%)
b2DynamicTree::Balance(int) 1.14% (1.14%)
b2DynamicTree::MoveProxy(int, b2AABB const&, b2Vec2 const&) 0.91% (0.91%)
b2Body::SynchronizeFixtures() 0.91% (0.91%)
b2Timer::b2Timer() 0.91% (0.91%)
b2Timer::GetMilliseconds() const 0.91% (0.91%)

release mode


b2CollideEdgeAndPolygon(b2Manifold*, b2EdgeShape const*, b2Transform const&, b2PolygonShape const*, b2Transform const&) 0.68% (0.68%)
b2ContactSolver::SolvePositionConstraints() 0.68% (0.68%)
b2ContactSolver::SolveTOIPositionConstraints(int, int) 0.68% (0.68%)

release mode

•


b2TestOverlap(b2Shape const*, int, b2Shape const*, int, b2Transform const&, b2Transform const&) 0.68% (0.68%)
b2DistanceJoint::SolvePositionConstraints(b2SolverData const&) 0.68% (0.68%)
b2DistanceProxy::Set(b2Shape const*, int) 0.68% (0.68%)

release mode

SRS

A Linux Operating Software such as Ubuntu.
A Box2D environment to run the applications.

Work Distribution

Naveen Kumar(140050013):
-Basic structure of the code
-Creating callgraphs.
-Handling makefile

Yathansh Kathuria(140050021):
-Responsible for the idea of simulating combustion engine
-Beamer presentation
-Web Page

Rajat Chaturvedi
-Debugging of code
-Project report
-Project video

Note: All three group members were responsible for major ideas on how to go on with the project. Each member contributed to the success of the project. There were many times where a member's idea or insight helped us in going forward, and we won't be able to list them all.

Link To Webpage

www.cse.iitb.ac.in/~knaveen/box2d.html

References

[1] Box2D 4-Stroke Combustion Engine https://www.youtube.com/watch?v=8kZRpouZ3OQ

[2] Box2D mannual http://www.box2d.org/manual.html

[3] Box2D Tutorial http://www.iforce2d.net/b2dtut/

[4] Box2D Blog http://codingowl.com/readblog.php?blogid=119

[5] Making Repository http://github.com/

[6] doxygen tutorial https://www.stack.nl/~dimitri/doxygen/manual/

[7] Non Convex Shape Box2D http://www.emanueleferonato.com/2011/09/12/create-non-convex-complex-shapes-with-box2d/

[8] Bibliography https://www.sharelatex.com/learn/Bibliography_management_with_bibtex

[9] git tutorial https://www.atlassian.com/git/tutorials/undoing-changes/git-checkout

[10] youtube video https://www.youtube.com/watch?v=QPaUJfA1KsY

[11] piazza.com

[12] tutorial for webpage `w3school.com`

[13] video making software- kazam `https://apps.ubuntu.com/cat/applications/kazam/`