Creating structures with ammo.js and three.js project blog

**Introduction**

This project I started as a part of my module computer science challenges it is a course which you learn and produce a project using technology you have never used before, I ended up choosing a topic robotics and manufacturing to learn about the technologies involved in it, Because of this I was assigned the structures project :

Project objective:

**Model cheaply purchasable components e.g., 2x4 planks, nuts and bolts and create them in three.js/bullet with appropriate physics properties, including weight, friction. Objects should include costs and links for purchase.**

**Create models of physical parts with appropriate bending, twisting and breaking effects (model that they can occur) so that structures of suitable strength can be developed and limitations under forces can be incorporated into the design of machines.**

**Creating models of physical attachments e.g. nuts and bolts, nails, glue etc. that have parameters and can replicate plausible behaviours when subject to physical forces within a physics engine (including vibration)**

Three.js and ammo.js

To do this carry out this task I was assigned to use three.js and ammo.js both javascript libraries three.js used to create 3d graphics in a web browser and ammo.js to create a physics world these two libraries could be used in conjunction to display objects that I want to simulate and show how they are effected by physics. This task was overwhelming at first for me as I was not only new to physics and graphics engines but also I had never used JavaScript, So as a pre -requisite I had to learn this language to learn this language I followed a youtube series created by a a popular youtube called mosh <https://www.youtube.com/watch?v=W6NZfCO5SIk>, This tutorial not only got me set up on the basics of javascript which I picked up pretty quickly because of my experience in other programming languages similar to javascript however I found out during the duration of my project that javascript is a little harder than showed in the turoail due to some of the basic functions that other languages had built in which I had to implement myself like 2d arrays this tutorial also introduced me to the IDE I would be using ofr the duration of the project visual studio code. I had never really used this ide before so I also had to do some tutorials learning it as well it was not that hard and I have ended up with visual studio code being one of my favourite ides.

Ammo.js concepts

During my learning experience I covered some basic concepts of ammo.js which were the fundamentals of which my project would be based for each of these concepts I created sample projects which I have added to a public github ( ) which you can take and use to help you get used to using and understanding ammo.js.

Rigid bodies

Soft bodies

Constraints

Springs

**Project layout**

Setting up the library’s and ide

Creating the three.js world

Creating the ammo.js world

Creating the static world plane

Animating the rigid bodies in the world

Creating the plank

Creating the plank

When I was deciding how I should make the plnka object I ran into the issue of what I should do to represent this object, I ended up deciding using the resources in ammo.js to simulate the properties of a plank that I should use several rigid bodies connected with p2p constraints and containing springs which allows the planks connections to simulate the changes in shape of the plank and be able to return to original positions.

To Start off I the plank I created two variable for containing the planks length and width in this project represented in inches, Then using this information I created a 2d array the same amount of arrays as the length variable previously instantiated, and in each of these arrays the same amount of elements as the width assigned in the width variable because 2d arrays are not a built in function in JavaScript I used this method of creating the 2d arrays:

I created the 2d array so when I created the cube bodies in ammo.js I could add each of them to the 2d array with a position representing their actual positions in the ammo.js world so I could later carry out other function to these blocks like adding the p2p constraints and eventually add springs to them.

I\*mage of 2d array\*

Adding collision constraints

During the project I had issues with the blocks falling through the static plane I realised after some code searching that my issue was my collision mask, masks in ammo.js decide whether are not two object should collide together through the use of a bitwise and operation if two objects are in close proximity in the physics world the object that is going to collide does an and operation and if the and operation does not cause the binary to be equal to 0 the two bodies collide otherwise they do not collide an example of this would be

Ball collision mask = 1

Block collision mask = 2

And operation = 0001

= 0010

0000 = 0 therefore the two objects collide

Otherwise 0001

0001

0001 = 1 therefore the two objects don’t collide. This is because the two objects are in the same collision groups.

Creating the plank hinges

To add the hinges I had to iterate through each of the columns and connect each of the objects using p2p constraints using the objects of the 2d array. I then went through each incrementally went through each row and connected each of the column items this then led to the plank being completely connected.

**Goals**

**Throughout the duration of this project the objective has slowly deviated from what I was originally assigned, Although I have so far tried to complete to project assigned, I have found that it is more important that I learn and document how to use ammo.js as it has so little documentation causing the library to be pretty inaccessible to someone who has not used a physics engine before. When I have finished this module, I hope to have made good progress in simulating some structures however I want to have made the library more accessible and easier to learn making it easier for someone to pick up from where I left my project off.**

**Plan of how it can be achieved.**