"This project will explore to what extent physical responses could be learned from example, using deep learning to predict future state of a physical system. For simplicity this would most likely focus on 2D physics engines like Box2D, and investigate how much of the dynamics can be emulated by a learning approach rather than an explicit modeling approach. A more advanced version might involve learning dynamics from real-world videos to realistically approximate physical behaviour of real objects."

Our initial problem is one of creating a learning model that can predict upon a physical system. We must define what the physical system will be and how we would measure the extent to which that system has been learned.

Two dimensional physical systems are uncommon in our three dimensional world but they are very common in the world of games. Pong is a two dimensional physics system and due to its status makes a very interesting case to study for our own physics system. There are only two core aspects to Pong; the ball and the paddles. The ball moves at a constant speed on a two dimensional plane and the paddles move at a constant speed on a one dimensional plane. What makes this a physics system is that these two objects interact, and their interaction is defined by a system of mathematical equations. When the bounds of the ball touch the bounds of the paddles, the speed of the ball inverts and increases. This simple interaction, this one line of code, is enough to create the image and feel of a game of tennis within our heads. So we see that it does not take much to create a compelling and therefore useful physics engine. After much planning we determined our system would loosely model the game of pool, the same way Pong loosely models tennis. Circles enclosed by a rectangular box; the circles can move and collide with each other and the walls of the box, changing their speed and direction in the process. This was chosen as the starting point for the physics system for a number of reasons. The first was that it meets our requirements for a useful physics engine, it models a real life physical system in a compelling manner. Second was that it is simple to implement and easy to extend. An initial model may only have two circles, and constant speeds without a continuous deceleration. Once we know our neural network can understand this we can scale upwards, making the simulation more complex and realistic. This could include more objects, deceleration, varying sizes of circles, varying weights of circles, adding static obstacles, etc. In this manner we could identify what the limits of the neural network are and what aspects in particular cause it difficulty. Third was that it fit perfectly into plans to extend the project outside of the neural network itself. This included predicting based on visual information, as it would be simple to determine the speed and position of the circles visually, and modelling real life objects, since there is not much of a three dimensional component to pool in real life.

With our physics system defined, how would we measure the extent to which it has been learned. Although our system is modelled after pool we must make it clear that our goal is not to teach the neural network to play pool. The problem with that approach is that it is possible for a deep learning algorithm to learn to play pool without any understanding of the physics system behind it. It can learn by trial and error the best moves to make without any knowledge of the actual state of the objects. This is not learning the physics of the system and is not what we want. As such the neural network will not interact with the system itself, it will only observe and predict. Another trap we must avoid falling into is teaching the neural network only a segment of the system. This is where we must detach ourselves from the pool metaphor again. The initial configuration of a game of pool is the same each time so while the space of possible configurations of circles is massive, in practice you would encounter certain configurations more than others. As such this will cause more and more edge cases that cannot be found in your training data set that your neural network cannot resolve accurately. The measure of how much of the system has been learned is the neural network's ability to predict upon configurations it hasn't seen before and for that we must feed it randomised configurations to ensure it has as broad a knowledge as possible.

This will be the approach to the core of our problem.