Why Maths and Physics are useful in Computer Science

Jim Mussared

Are Maths and Physics important?

"I know that CS theory requires more maths than you can chew, but when working on an actual software project, how often do you encounter maths stuff?"

"From what I know, understanding of maths is good, but there mostly isn't any application of maths."

"What in particular do you use that kind of mathematics for, and what kinds of theories are most important when you do need it?"

"Because nobody has been able to give me an answer and I don't get why I would need physics."

Source: NCSS Challenge "Industry Mentors" forum, 2012

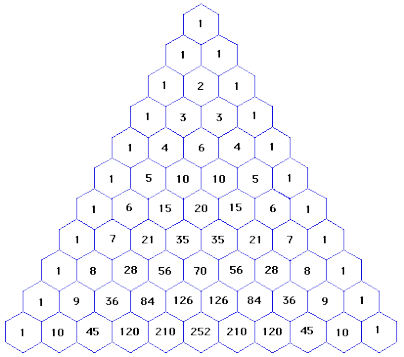
Discrete Mathematics

Discrete maths involves individual (discrete) things, as opposed to other areas of maths which can deal with infinitely divisible things.

In discrete maths, things happen or they don't, they're included in the set or not, etc.

This is highly relevant to computers because everything in a computer is a binary yes/no.

Discrete Mathematics - Combinatorics



n_Cr and n_Pr.

This pops up all over the place in programming, any time you've got some sort of queueing system or load balancing or partitioning.

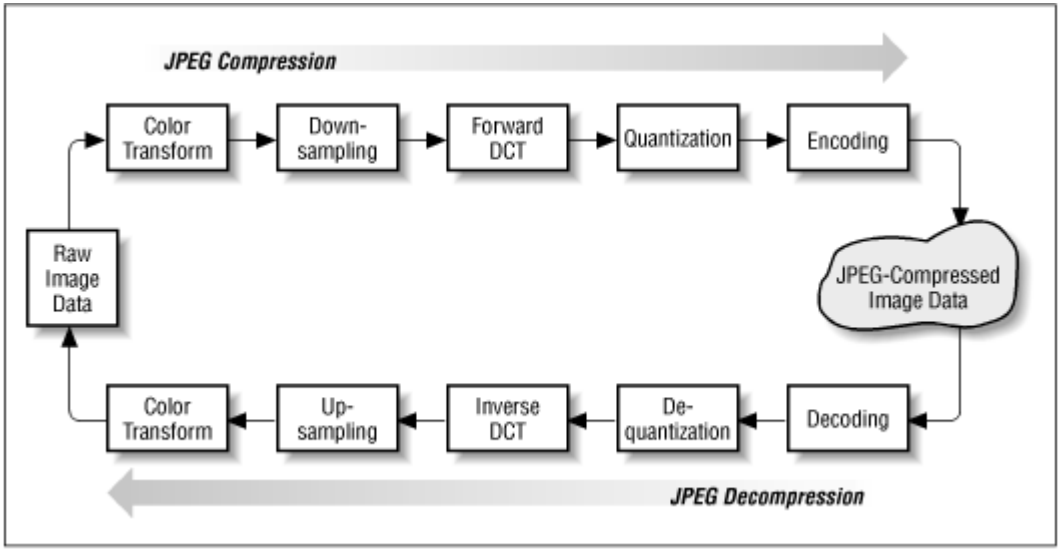
At Google:

I have X webservers, and Y databases, each webserver sends three queries to the database. What distribution of queries do I expect at the database? Or a simpler example, I need to find the right re-arrangement of these letters, how do I generate all the possible re-arrangements?

"Stirling numbers of the second kind" - the number of ways to partition a set of n objects into k non-empty subsets.

Discrete Mathematics - Information theory

This is useful in things like compression algorithms (like JPEG, ZIP), and also shows up in building efficient digital radios (like your mobile phone, WiFi, etc).



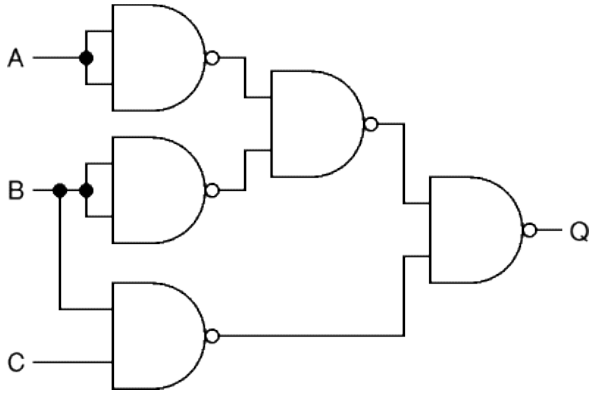
Discrete Mathematics - Algebra

A good example is boolean algebra, for example De Morgan's laws.

If you have:
not(a and b)

it is always logically equivalent to:
(not a or not b)

and vice versa.



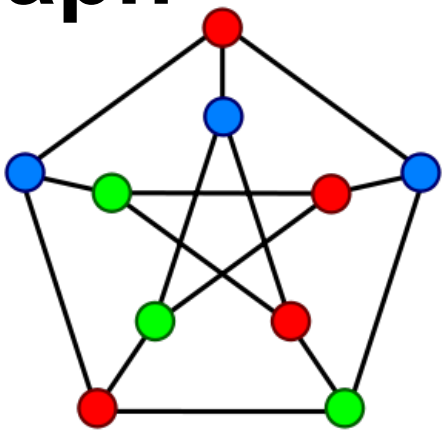
This can be useful for making your if statements easier to read, or sometimes faster.

Discrete Mathematics - Number Theory

Keeping your credit card details safe on the internet revolves around the magic of number theory.

The RSA algorithm is probably the most famous example and allows two parties to exchange a "secret key" via an untrusted channel.

Discrete Mathematics - Graph Theory



This is a branch of mathematics to do with processing things that are "graphs". A graph is a collection of vertices and edges (not to be confused with the sort of graph you draw on graph paper).

Perhaps the vertices represent cities and the edges represent roads. You want to find the shortest path between two cities. Perhaps they were power lines instead, and you want to know the minimum total distance of power lines you can use to connect all the cities together.

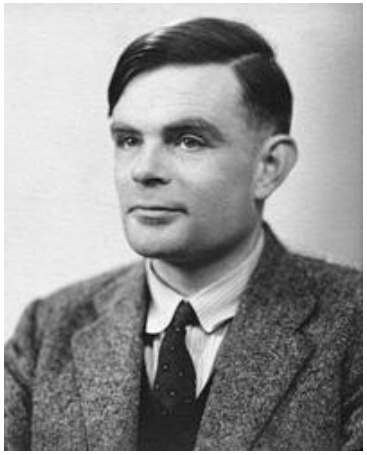
Famous example: the Travelling Salesman problem.

Discrete Mathematics - Theoretical Computer Science

This is about what can and can't be computed.

Famous examples include The Turing Machine, the Halting Problem and Gödel's incompleteness theorems.

They are amazing and counterintuitive results with some pretty profound implications for what computing and mathematics really are.



Discrete Mathematics - Probability

A lot of software is written to make predictions.

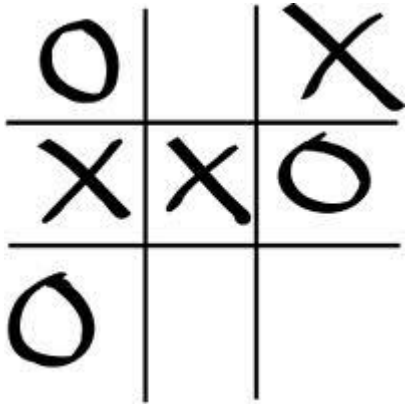
The stock market, whether users will click on ads on a webpage, congestion in a road traffic network, weather and climate.

Probability is everywhere.

Discrete Mathematics - Game theory

Artifical intelligence & machine learning.

Not just for games!



Discrete Mathematics - Operations Research

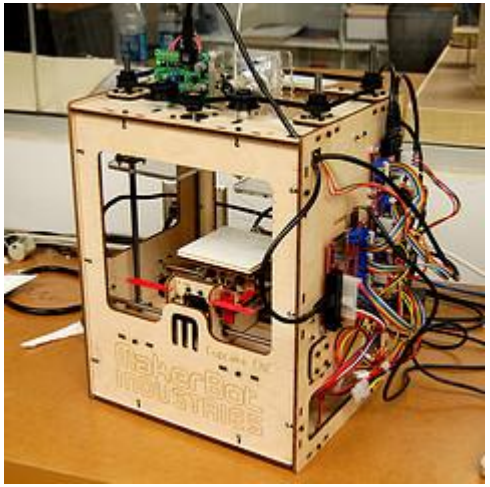
This is about using computers to come up with more efficient ways of doing things.

This can save millions of dollars for a business by reducing overheads or wasted costs.

Physics

Physics turns up a lot in computing.

Computers are just applied electronic engineering, which is a great skill to have if you're ever trying to understand how computers work, or interface anything to the real world.



Physics - Forces & Motion

Mechanics, kinematics and ballistics (i.e. traditional physics, forces, energy, etc) is absolutely critical to any sort of game programming, robotics, physical modelling, etc.

You're writing a computer game that involves a character (let's call him Mario) jumping between platforms. When he jumps he falls back to the group with gravity.

That's forces and acceleration

(which is just calculus).

$F = ma$

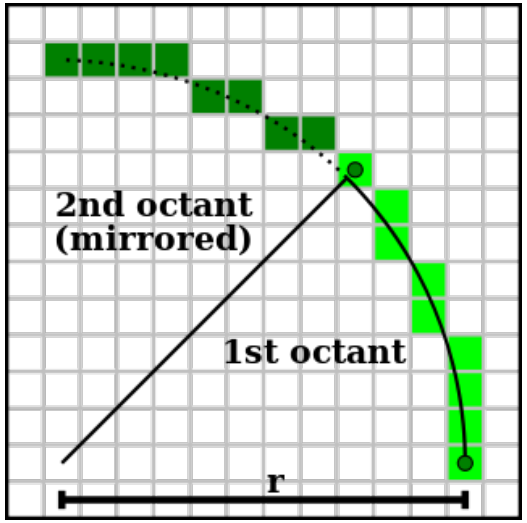
$s = ut + 0.5\ a\ t^2$

Physics - Circular motion

Lets say you wanted to draw a button on the screen with rounded corners. And your computer only supports integers (i.e. no floating point numbers).

How do you calculate the pixels that you need to draw?

The midpoint circle algorithm.

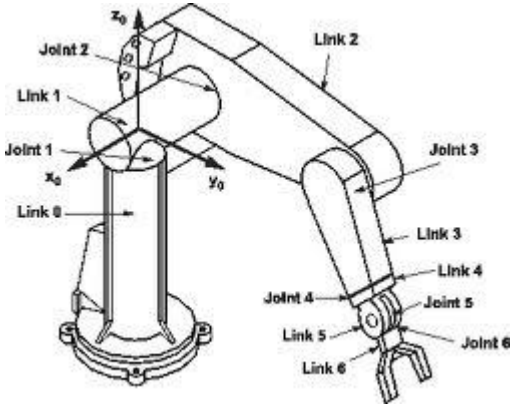


Physics - Inverse kinematics

You're writing the software to control a robot arm manufacturing car parts.

You need to move the arm to a certain position, then grab something, then move it at a constant speed to another point. Figure out the commands to send to the motors.

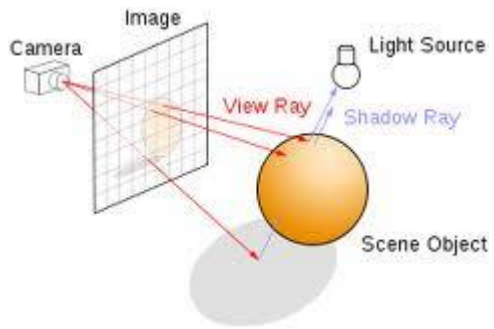
This is some seriously heavy duty physics & mathematics known as inverse kinematics

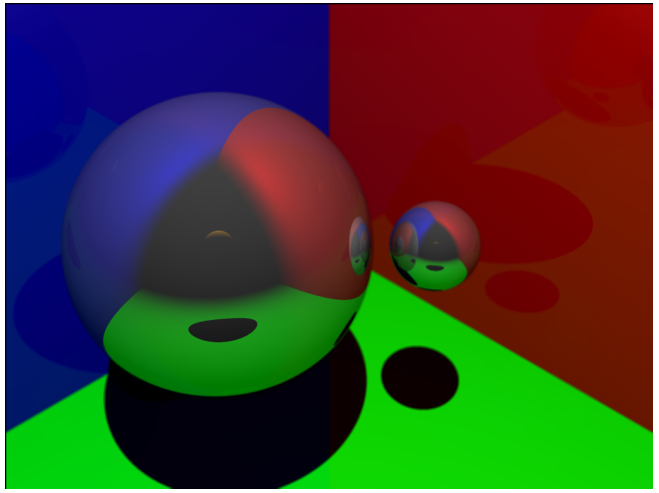


Physics - Special effects and digital images

You're writing the software to render special effects for a movie. You're given the information about a scene (objects, textures, positions, lighting, etc) and have to generate the resulting image.

There's all sorts of physics there, especially optics, and a lot of maths (mainly geometry).



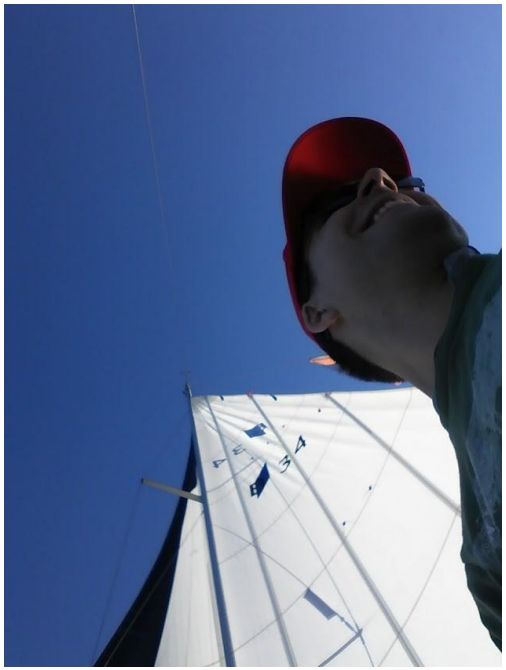


Physics is just useful

Every day, I'm grateful for the physics I learnt in high school and university.

Just knowing how things work, understanding forces and energy, etc is really wonderful.

And this isn't just because I'm a huge nerd, it's actually useful a lot of the time too.



It is important to know this stuff!

No one person is expected to be an expert in all of these things, but most programmers will come across most of these topics at some point and be able to recognise the different types of problems.

At Google, we have teams of people that specialise in each of these areas, so whenever we run into something we know who to ask!