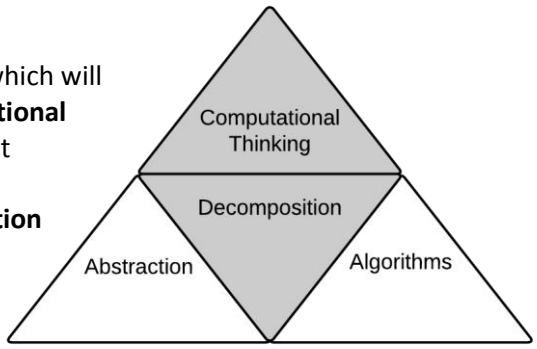
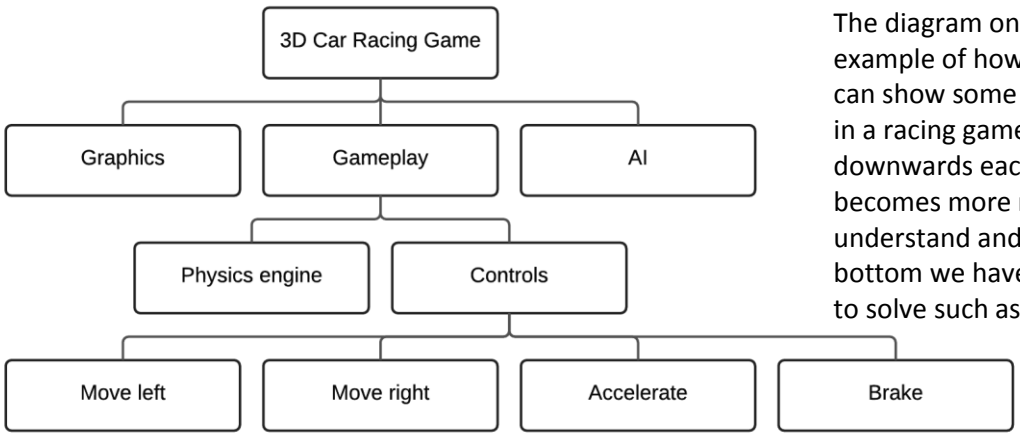
Decomposition - Reading



In order to use computers we need to be able to think in a way which will allow us to solve problems with them. This is known as **computational thinking** and is a useful skill for solving any problem that has a set procedure even if it doesn't require a computer. There are three components to computational thinking: **abstraction**, **decomposition** and **algorithmic thinking**.

Imagine a modern 3D computer racing game. It is incredibly complicated to understand and program all the different parts to it such as graphics, Artificial Intelligence (AI) and game play. **Decomposition** is breaking down a complex problem into smaller sub-problems. Each of these serves an identifiable task and can be further subdivided. When plants or animals die they are decomposed into smaller forms of matter so that they can be reused. This is exactly the same with decomposition of problems and it makes writing programs a more **manageable** task and allows us to **reuse** solutions to smaller parts of the problem. For instance, you could decompose the task of "making a cup of tea" into the smaller tasks of "fill kettle", "boil water", "take out milk", "add teabag", "pour water", "pour milk" "remove teabag".

A **top-down design** could be used to make such a program. This starts with the idea of a computer game and then decomposes it into smaller and smaller problems.



The diagram on the left shows an example of how the top down design can show some of the parts needed in a racing game. As we move downwards each of the tasks becomes more manageable to understand and program. At the bottom we have individual problems to solve such as making the car move right or brake.

It is easy to see how we could now program an entire game having **decomposed** the overall problem into much smaller parts. In programming, the smaller parts such as brake or move right would be developed using **subroutines** such as **procedures** or **functions**. For instance, there might be a procedure brake(20) which means apply the brake with 20% pressure. There might be a procedure called moveRight(30) which means turn the steering wheel by 30 degrees to the right. A programmer only needs to understand the **inputs** and **outputs** for each function or procedure without needing to worry about the details of how they work.

Decomposition is carried out in real life too. The problem of making a real car is decomposed into smaller problems such as making the engine or making the wheels. A car wheel is something which the car manufacturer could buy from another company. All wheels have similar properties such as being round and having rubber on the outside. The specifics of how the wheel is made don't need to be known. This makes it easier for the car manufacturer to build the car as they don't need to build every single component. The wheel is an example of an **abstract** object where we **don't need to worry about the detail** of how it is made or how it works. This is an example of **abstraction**.