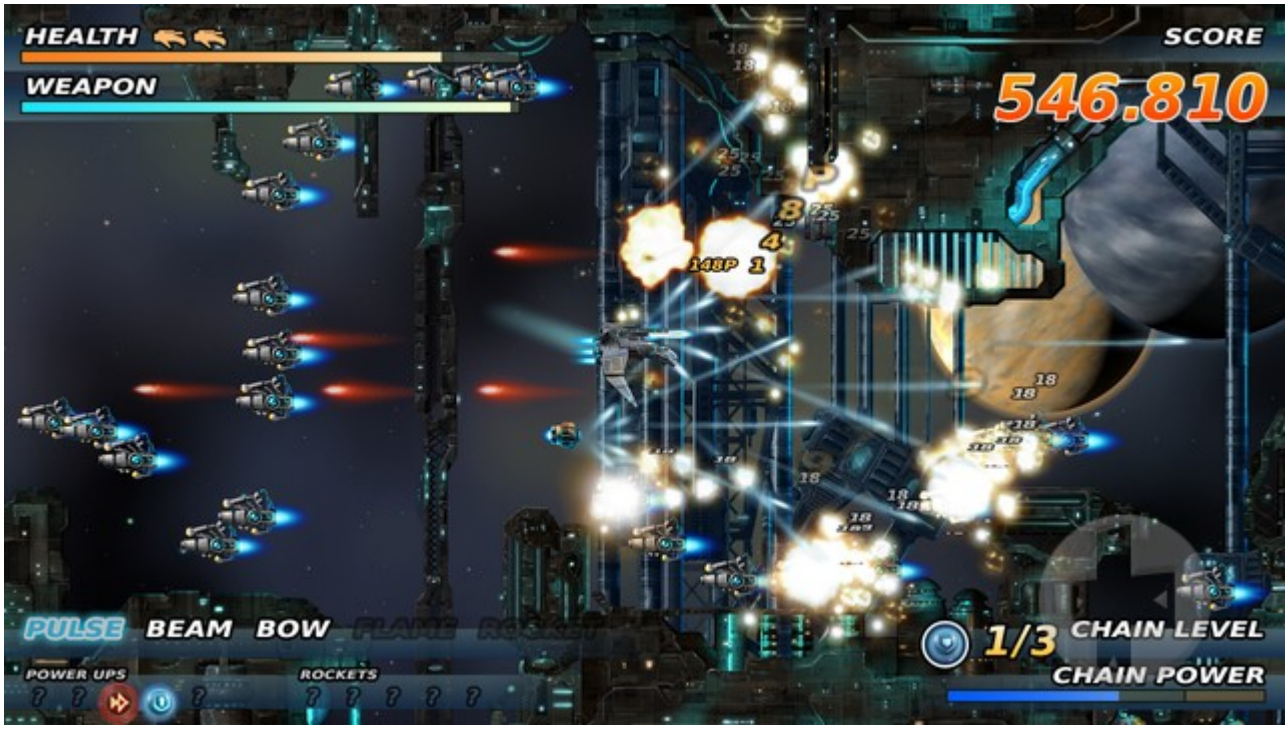
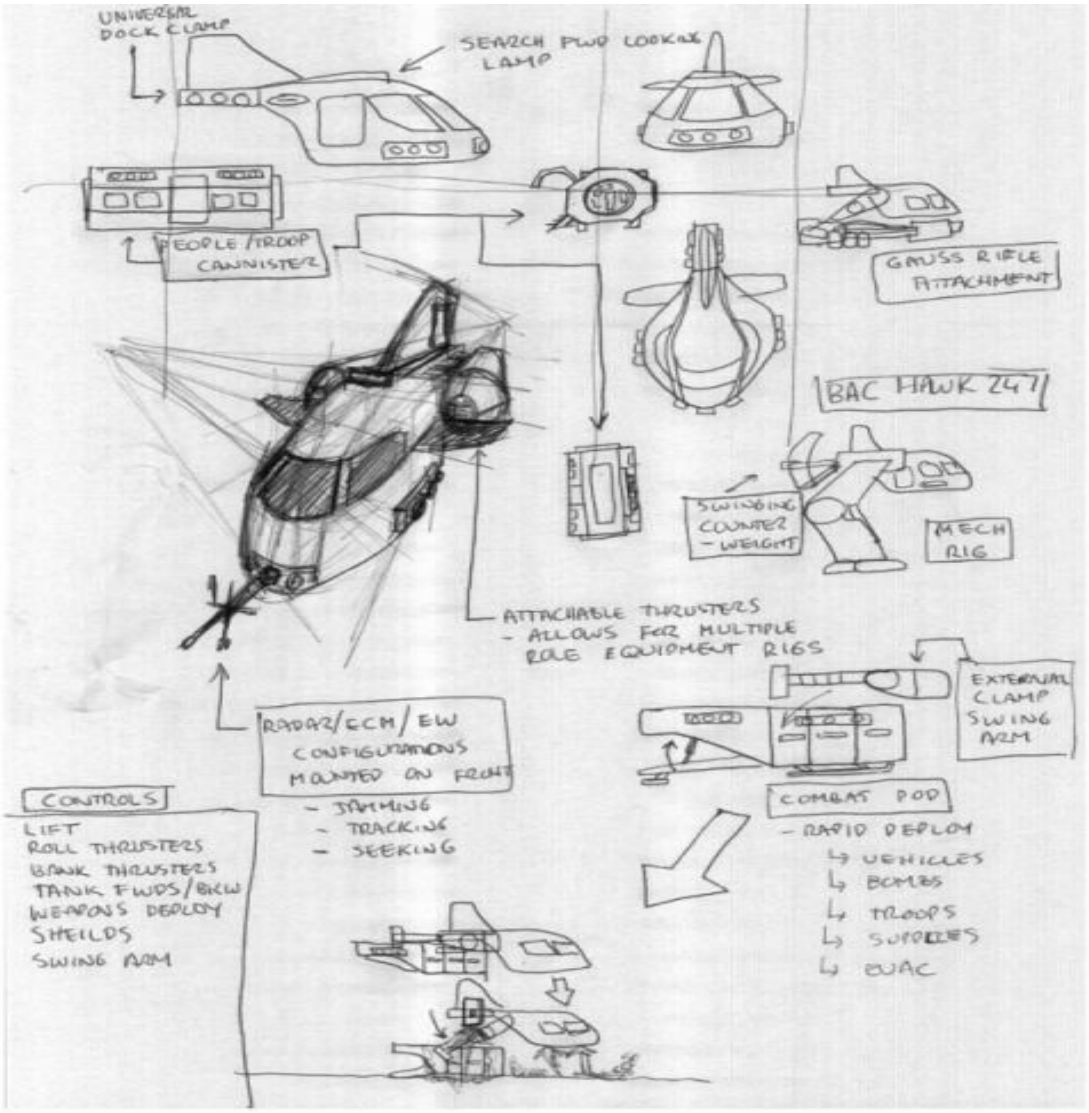
ABSTRACTION



We know we want to make a space shooter, and we know we will need a way to create spaceships. We don't care while we are designing the game what memory to use, what the graphics card is, how to handle shields, or shooting lasers and collision detection.

By using abstraction we can design the game and solve problems at a higher level. Finally we can code the particulars once the design is complete – then we worry about the details.

ENCAPSULATION



The spaceStation base class has an interface to its behaviours defined by its public methods. We can design the spaceStation to produce ships based on what we want them to do in game.

It means when we spawn a ship we suddenly have access to all it's outward facing controls such as SetSpeed(), GetShields(), AttackPlayer() but we don't need to know how these work.

Internally the functionality of these behaviours are protected and are mostly future proof. For example the AttackPlayer() method can change to do something new and the calling code doesn't need to change – like an invisible upgrade.

INHERITANCE

A class is like a alien spaceStation.

These 3 spaceships are created in the spaceStation but each can have unique properties, such as different speeds, different shields, different bullets.
In object oriented programming terms they can be thought of as a derived classes, or sub-class of the
base class/original spaceStation.

INHERITANCE

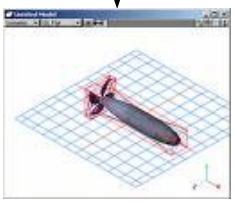
The base class (or spaceStation) has predefined data types also called attributes

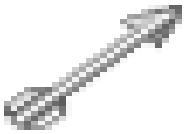
Each of these ships (or sub-classes/child classes) inherit the attributes from the base class (spaceStation), kind of like passing on the values when creating ships.

For example the spaceStation will have speed and shields attributes. These are float data types.
Each derived spaceship /sub-class can then change the value of speed and shields to suit the new ship/object.

INHERITANCE

The base class / spaceStation also has embedded functionality called behaviours.





The derived ships / sub-classes also inherit these behaviours. An example of a behaviour could be Fire() – each ship knows how to fire weapons.

However each ship can be defined to have it's own weapons systems. Some have lasers, some have photon torpedos and some have missiles.

This is called inheritance and allows for code re-use with little or no modification. What this means is you could spawn 100 ships all firing lasers and then spawn 10 ships that fire missiles by changing only a few lines of code.

POLYMORPHISM

The spaceStation base class has a behaviour or method called DodgePlayer()

Each of the ships inherits the DodgePlayer() behaviour but in ship number one it dodges by flying up. Ship 2 flies away in the opposite direction really fast and ship 3 rams the player in an attempt to damage their shields.

But we don't care about this as a programmer. All we want to do is tell the alien to DodgePlayer() so we can use this ability of classes called Polymorphism to call Ship.DodgePlayer() and each ship will deal with the movement how it sees fit to do so.