Introduction

AGen

AGen is my web based army-generator. Originally AGen was written to support NetEA (an Epic Armageddon variant), but it is flexible enough to support other game system. In the mean time Warmaster army lists are also supported.

AGen is written in Javascript and runs in most browsers including iOS and Android browsers - may I say as usual - there are problems with IE. I will not address them, I have no need for IE as I use a Mac and Firefox and Chrome do work on the PC.

Data is saved in the browsers local storage, but there are backup and restore options, for data safety and data exchange.

AGen can be found at http://brumbaer.de/Epic/AGen/index.html

myAGen

The one major drawback of AGen is that there is no way for users to create their own army lists. Especially as I'm not able to keep up with changes to the army lists.

myAGen allows everybody to create army lists and to distribute them.

myAGen uses the same routines as AGen, but some of the files are hosted locally on the users hard drive. He is able to modify these files and add army lists and game systems to the existing ones.

Hardware requirements

Any computer-device that can run a current browser that is able to run AGen.

200k storage space, on your computers hard disc or a memory stick.

Software requirements

A current browser that is able to run AGen.

A text editor to modify the army list file. Any old text editor that can edit "text only" will do. But a specialized Javascipt editor with command completion and auto indent ist much more comfortable.

The myAGen zip file, which can be downloaded from brumbaer.de/Epic/AGen/myAGen.zip. Install the files on any drive in any folder, but keep them together in the same folder.

A Javascript debugger is also handy. Some browsers like Safari on the Mac have the option to show an developer menu, which gives you access to a debugger and displays error messages.

The files in AGen.zip

At the moment there are 19 files in the zip.

12 of them are images. They are included because of relative path references. There is no need to change them, ignore them, but don't delete them.

5 are html files. index.html is the boss and the other ones will be called depending on the operations performed. Don't edit them, you can mess everything up, by doing so.

One is this document.

The last one is myAGenSample.js. This is a sample file. You can rename it to myAGen.js to see what the code in the sample file does. Or you can copy all or parts of it to your own myAGen.js as a starting point.

There is no myAGen.js, despite the fact that this is what this is all about. You will have to create your own one or copy/rename myAGen.Sample.js. This choice is deliberate, to prevent overriding of your work, by updating/restoring the contents from the zip.

All the files from the zip have to stay in the same folder and myAGen.js has to be in the same folder as well.

Is's a mess, but it's your mess

It's rather easy to mess up. But whatever you do, you will mess up only in your myAGen.js, your work will not affect AGen, or anything else on your computer.

Especially when you messed up one of the html files (I told you to keep away, didn't I) you always can download the zip and copy clean files back in.

Javascript

AGen is written in Javascript and as myAgen is build on it, it uses Javascript as well.

There is no layer between Javascript and the army lists, you will write the army lists in Javascript, but there is a Framework that makes life much easier.

If you never did any programming in Javascript or any programming language this is a good opportunity to start. But you can also write lists without the knowledge by just following the steps precisely.

myAGen.js

myAGen.js is the file in which you write your code or army lists.

You will have to create one or copy myAGenSample.js and rename the copy to myAGen.js.

Make sure your myAGen.js file is in the same folder as the files extracted from myAGen.zip.

myAGen.js is a Javascript source file that is executed just before the website is displayed. At this point the standard game systems and army lists are already loaded.

The file is in the global namespace, so you can overwrite variables used by other parts of the program.

To prevent this, please enclose your code with

```
(function () {
   Whatever you program goes here
})();
```

This will put your code in an anonymous function that will be executed immediately after it's definition.

If you now declare all your variables using the ***var*** keyword, the will be local to this function and the risk of conflicts with variables from other parts is minimal.

```
(function () {
   var me = "Brumbaer";
   Whatever else you program goes here
})();
```

Running myAGen

myAGen uses Javascript, so it does not need to be compiled.

To run it all you have to do is to double click the ***index.html*** file, even if you have no myAGen.js file yet.

For your own changes to be applied, you have to put them into the ***myAGen.js*** file.

If you make changes to this file they can only take effect, when you save the file after the changes and when you reload the website by double clicking index.html or using the reload button in your browser.

AGen's hierarchy and elements

AGen has a strict hierarchy for it's army lists. Parts of the hierarchy are called elements.

To get an idea what the elements relate to in real life

From the top :

Element	NetEA	Warmaster
Game system	NetEA	Warmaster
Army book	Codex Eldar	-
Army list	Biel Tan	Empire
Category	Warhosts	Cavalry
Formation	Guardian formation	Knights
Unit	Farseer stand, Guardian stands…	Knight stands
Datasheet/Profile	Inf, 15cm, 4+ …	Cav,, 3, 3, 4+ …

Some game systems have no representation of some elements. In this case just create one default element to put everything in or invent one.

I.e. Warmaster does not use categories, but you could divide your formations in Infantry, Cavalry and Other.

fInit and current element

When created or used many elements will define a function and within this function it's child elements will be created. This will lead to nested function, which a Javascript editor will indent automatically, which gives a visual clue to what belongs to what.

This function is referred to as ***fInit***.

You will create or use a game system. In it's fInit you will create or use an army book. Inside the army book's fInit, you will create an army list and so on.

The last fInit function started, but not ended, is called the current fInit and the element the fInit belongs to is the current element.

AGEN framework

The AGEN framework will provide functions to ease the creation and administration of those elements.

AGEN keeps track of what you are defining, reducing the number of variables needed to cases where you want to access some element out of order.

AGEN.xxxGameSystem

When you start AGen you will see the army lists grouped by game system.

To add an army list to an existent game system you have to declare it's use.

```
AGEN.useGameSystem (gs, fInit);
```

gs is an existing game system. There are two predefined game systems ***EA*** for NetEA and ***WM*** for Warmaster.

fInit is a function that contains the code used to add your stuff to the game system. Usually you define that function inline.

If you want to add something to the EA game system, the myAGen file would look like:

```
(function () {
   AGEN.useGameSystem (EA, function () {

   });
})();
```

If you do not want to add to an existing game system, but create a new one you use

```
AGEN.newGameSystem (name, descriptor, fInit);
```

or

```
AGEN.newGameSystem (name, descriptor, profile, fInit);
```

or

```
AGEN.newGameSystem (name, descriptor, profile, htmlFile, fInit);
```

name being the name displayed at the top of the column of army lists in the start screen.

descriptor a name describing the game system. This is also used as key to distinguish between game systems, so make sure it's different from all other game system descriptors.

profile is an array of strings, that holds all fields of an army entries profile. If you do not include an profile it will default to the Epic profile which is ***["Name", "Type", "Speed", "Armour", "CC", "FF", "Weapons", "Range", "Firepower"].***

htmlFile is the editors html file. A description is out of the scope of this document. If omitted the value defaults to ***"Epic.html"***.

fInit is a function that contains the code used to add your stuff to the game system. Usually you define that function inline.

If you want to create a new game system for Epic UK lists, the myAGen file would look like:

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {

   });
})();
```

A game system will only be displayed in the start screen, if it has at least one army list. So if you double click on index.html, you will not see any Epic UK game system yet.

AGEN.xxxArmyBook

An army book is used to group army lists. What criteria you use for the grouping is up to you. It could be codex, saints or sinners, number of legs or whatever you fancy.

You can also use ***AGEN.addNotes*** (description somewhere further down) to add Information to the army book, which will be displayed at the start of the notes section of all army lists in this army book.

To use an existing army book, you use

```
AGEN.useArmyBook (ab, fInit)
```

ab is a variable you used to store a previously created army book or one that is already defined, like ***EA._orks, EA._tyranid, EA._imperial, EA._spaceMarines, EA._chaos, EA._eldar, EA._nekron, EA._darkeldar, EA._tau.***

fInit is a function that contains the code used to add your stuff to the army book. Usually you define that function inline.

Using an army book will also set the army to be used.

A file to add to EA's Space Marines would look like

```
(function () {
   AGEN.useArmyBook (EA._spaceMarines, function () {

   });
})();
```

To create a new army book you use

```
AGEN.newArmyBook (name, sortNumber,  lightColor, darkColor, offWhiteColor,
backgroundColor, fInit)
```

name is self explanatory.

sortNumber gives the relative position of this army book. It should be a value between 1 and 1000. Army books with lower sort numbers will be displayed ahead of army books with higher sort numbers.

lightColor, darkColor, offWhiteColor, backgroundColor are HTML colors packed in strings. They define the colors used when an army of this army book is displayed. The concept of sortNumber is a recurring one.

lightColor being the standard color to be used, when an army is edited.

darkColor being the color used for frames.

offWhiteColor is used for the background when an army is edited.

backgroundColor being the color to be used in the start screen. It is usually identical to lightColor.

fInit is a function that contains the code used to add your stuff to the army book. Usually you define that function inline.

The army book is added to the game system in which fInit function it is defined.

To add a Space Marine army book to our Epic UK game system we write.

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
      AGEN.newArmyBook ("Space Marines", 1,
         "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
         function () {
         });
      });
})();
```

AGEN.newArmyList

Finally we made it to the army list.

To create an army list, you use

```
AGEN.newArmyList (name, id, sortNumber, fInit)
```

name as usual.

id is a string that is used to identify that army list. The id must be unique within this game system. This id is used in backup and restore operations. When you made severe changes to an army list, you should change the id of the army list to prevent that incompatible army lists will be backed or restored.

sortNumber gives the relative position of the army list within it's army book.
fInit the usual function in this case to define the elements of the army lists.
The army list is added to the army book in which fInit function it is defined.
To add the Codex Astartes to our army book we have to write:

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
      AGEN.newArmyBook ("Space Marines", 1,
         "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
         function () {
            AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
            });
         });
      });
})();
```

Now as our game system has an army list, it will show up in the start screen. Of course you have to copy the above code into your myAGen.js first.

AGEN.newDatasheet

Each unit has a datasheet. The data sheet consist of the profile values, notes and some administrative values.
You create a profile by calling:

```
newDatasheet (id, sortNumber, profile)
```

or

```
newDatasheet (id, sortNumber, profile, notes)
```

id is a string that is used to identify that datasheet. The id must be unique within this game system.
sortNumber gives the relative position of the datasheet within the datasheet list of an army.
profile is an array of values according to the profile defined for the game system. If a datasheet would fill two profile lines, you just add fields for the second line. Note that you have to add values for all the fields in the second line, even if they are considered to be empty. Use the empty string "" for such fields.
notes is a text that is to be displayed below the profile. This is not part of the profile, so that it can be formatted independently from the profile.
The same datasheet is usually used more than once in any army lists, so it is a good idea to assign all datasheets to variables and use those instead of redefining the datasheets again and again.
Defining all datasheets is the first thing I do in the fInit function of an army list.
The Thunderbolt fighter and Marauder bomber have 3 lines in their profiles, but no notes.

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
      AGEN.newArmyBook ("Space Marines", 1,
         "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
         function () {
            AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
               var thunderboltfighterbomber = AGEN.newDatasheet (
                  "thunderboltfighter-bomber", 1290,
                  ["Thunderbolt Fighter-Bomber", "AC", "Fighter-Bomber", "6+", "n/a", "/a",
                  "Stormbolters", "15cm", "AP4+/AA5+, FxF",
                  "", "", "", "", "", "", "Multilaser", "30cm", "AP5+/AT6+/AA5+, FxF",
                  "", "", "", "", "", "", "Underwing Rockets", "30cm", "AT4+, FxF"
```

```
            ]);
            var marauderbomber = AGEN.newDatasheet ( "marauderbomber", 1300,
               ["Marauder Bomber", "AC", "Bomber", "4+", "n/a", "n/a", "2x Twin Heavy
               Bolter", "15cm", "AA5+",
               "", "", "", "", "", "", "Bomb Racks", "15cm", "3BP,FxF",
               "", "", "", "", "", "", "Twin Lascannon", "45cm", "AT4+/AA4+, FxF"
            ]);
         });
      });
   });
})();
```

Some datasheets are the same over different army lists, you could define them in their army book.
Check myAGenSample.js for many more samples of data sheets.

AGEN.newCategory

After the short break from hierarchy, we're back in line.
Categories will be displayed in the order of their definition.
A Category is defined by using

```
AGEN.newCategory (name, fInit)
```

name as usual.
fInit the usual function in this case to define the elements of the category.
The category is added to the army list in which fInit function it is defined.
Adding three Categories to our army lists.

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
      AGEN.newArmyBook ("Space Marines", 1,
         "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
         function () {
            AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
               var thunderboltfighterbomber = AGEN.newDatasheet (
                  "thunderboltfighter-bomber", 1290,
                  ["Thunderbolt Fighter-Bomber", "AC", "Fighter-Bomber", "6+", "n/a", "/a",
                  "Stormbolters", "15cm", "AP4+/AA5+, FxF",
                  "", "", "", "", "", "", "Multilaser", "30cm", "AP5+/AT6+/AA5+, FxF",
                  "", "", "", "", "", "", "Underwing Rockets", "30cm", "AT4+, FxF"
               ]);
               var marauderbomber = AGEN.newDatasheet ( "marauderbomber", 1300,
                  ["Marauder Bomber", "AC", "Bomber", "4+", "n/a", "n/a", "2x Twin Heavy
                  Bolter", "15cm", "AA5+",
                  "", "", "", "", "", "", "Bomb Racks", "15cm", "3BP,FxF",
                  "", "", "", "", "", "", "Twin Lascannon", "45cm", "AT4+/AA4+, FxF"
               ]);

               AGEN.newCategory("Detachments", function() {
               });
               AGEN.newCategory("Titan Legion", function() {
               });
               AGEN.newCategory("Imperial Navy Aircraft", function() {
               });
            });
```

```
        });
    });
})();
```

Only categories with at least one formation are shown in the army list edit screen. So the categories will still not be shown in the army list edit screen.

AGEN.newFormation

Formations will always be displayed in the order of their definition.

Formations are create with

```
AGEN.newFormation (name, startpoints, fInit)
```

or

```
AGEN.newFormation (name, startpoints, minCount, maxCount, fInit)
```

name as usual.

minCount the minimal number of formations of this type in this army list. The default is 0. If the value is greater than zero as many formations are created, when an army of that type is created, and the software ensures that he number of formations of that kind will not drop below.

maxCount the maximum number of formations of this type in this army list. The default is 99. The software ensures that he number of formations of that kind will not exceed this number.

minCount and ***maxCount*** do work with numbers only, so there is now way to change them dynamically (i.e. dependent on army size). If you have such a requirement you have to use ***AGEN.newErrorCheck*** (see somewhere below).

fInit the usual function in this case to define the elements of the formation.

The formation is added to the category in which fInit function it is defined.

Let's add formations for Thunderbolts and Marauders.

```
(function () {
  AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
    AGEN.newArmyBook ("Space Marines", 1,
      "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
      function () {
        AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
          var thunderboltfighterbomber = AGEN.newDatasheet (
            "thunderboltfighter-bomber", 1290,
            ["Thunderbolt Fighter-Bomber", "AC", "Fighter-Bomber", "6+", "n/a", "/a",
            "Stormbolters", "15cm", "AP4+/AA5+, FxF",
            "", "", "", "", "", "", "Multilaser", "30cm", "AP5+/AT6+/AA5+, FxF",
            "", "", "", "", "", "", "Underwing Rockets", "30cm", "AT4+, FxF"
          ]);
          var marauderbomber = AGEN.newDatasheet ( "marauderbomber", 1300,
            ["Marauder Bomber", "AC", "Bomber", "4+", "n/a", "n/a", "2x Twin Heavy
            Bolter", "15cm", "AA5+",
            "", "", "", "", "", "", "Bomb Racks", "15cm", "3BP,FxF",
            "", "", "", "", "", "", "Twin Lascannon", "45cm", "AT4+/AA4+, FxF"
          ]);

          AGEN.newCategory("Detachments", function() {
          });
          AGEN.newCategory("Titan Legion", function() {
          });
          AGEN.newCategory("Imperial Navy Aircraft", function() {
```

```
                AGEN.newFormation("Thunderbolt Squadron", 150, function() {
                });
                AGEN.newFormation("Marauder Squadron", 250, function() {
                });
            });
          });
        });
     });
})();
```

If you copy that to your myAGen.js and double-click index.html, you will be able to create an Codex Astartes army for Epik UK, which has only a Imperial Navy Aircraft category - the others are not shown, because they have no formations - which allows you to create Thunderbolt and Marauder formations.

AGEN.newUnit

Were getting close to the ability to do very simple army lists. All that's missing is to get the units and their profiles in.

Units will always be displayed in the order of the sortNumber of their datasheets.

We all know what a ***stand*** is, now we introduce the ***group***. A group is a group of stands. When you add stands to a formation, you add them as a group. Usually a group has only one stand, but sometimes you can buy something only in pairs or triples. In such a case a group would consist of 2 or 3 stands.

Units can be added with one of the following:

AGEN.newUnit (datasheet, startCountOfGroups)

AGEN.newUnit (datasheet, startCountOfGroups, fInit)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, fInit)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups, fInit)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups, maxGroups)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups, maxGroups, fInit)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups, maxGroups, standsPerGroup)

AGEN.newUnit (datasheet, costPerStand, startCountOfGroups, minGroups, maxGroups, standsPerGroup, fInit)

datasheet a datasheet as defined with AGEN.newDatasheet.

costPerStand the points cost per stand - not per group. if omitted it defaults to zero. See the paragraph about points below.

startCountOfGroups is the number of groups of this type of unit that will be added when the formation is created. This can be larger than minGroups, but must not be lower than minGroups or be larger than maxGroups. Otherwise you will not be able to edit he formation.

minGroups the minimal number of groups of this unit type in this formation.The software ensures that he number of formations of that kind will not drop below. The default value is 0.

maxGroups the maximum number of groups of this unit type in this formation.The software ensures that he number of formations of that kind will not exceed this number. The default value is the larger value of minGroups or 1.

minGroups and ***maxGroups*** do work with numbers only, so there is now way to change them dynamically (i.e. dependent on army size). If you have such a requirement you have to use ***AGEN.newErrorCheck*** or ***AGEN-choices*** (see somewhere below)

standsPerGroup the number of stands per group. if omitted it defaults to one.

fInit the usual function in this case to define additional stuff for this unit. If there is nothing to be added fInit can be omitted.

Thunderbolts and Marauders need only the simplest form of definition.

```
(function () {
   AGEN.newGameSystem ("Epik UK", "Brumbaer's AGen for Epic UK", function () {
      AGEN.newArmyBook ("Space Marines", 1,
         "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA",
         function () {
            AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
               var thunderboltfighterbomber = AGEN.newDatasheet (
                  "thunderboltfighter-bomber", 1290,
                  ["Thunderbolt Fighter-Bomber", "AC", "Fighter-Bomber", "6+", "n/a", "/a",
                  "Stormbolters", "15cm", "AP4+/AA5+, FxF",
                  "", "", "", "", "", "", "Multilaser", "30cm", "AP5+/AT6+/AA5+, FxF",
                  "", "", "", "", "", "", "Underwing Rockets", "30cm", "AT4+, FxF"
               ]);
               var marauderbomber = AGEN.newDatasheet ( "marauderbomber", 1300,
                  ["Marauder Bomber", "AC", "Bomber", "4+", "n/a", "n/a", "2x Twin Heavy
                  Bolter", "15cm", "AA5+",
                  "", "", "", "", "", "", "Bomb Racks", "15cm", "3BP,FxF",
                  "", "", "", "", "", "", "Twin Lascannon", "45cm", "AT4+/AA4+, FxF"
               ]);

               AGEN.newCategory("Detachments", function() {
               });
               AGEN.newCategory("Titan Legion", function() {
               });
               AGEN.newCategory("Imperial Navy Aircraft", function() {
                  AGEN.newFormation("Thunderbolt Squadron", 150, function() {
                     AGEN.newUnit(thunderboltfighterbomber, 2);
                  });
                  AGEN.newFormation("Marauder Squadron", 250, function() {
                     AGEN.newUnit(marauderbomber, 2);
                  });
               });
            });
         });
      });
})();
```

If you copy this code to your myAGen.js you will be able to create the formations as before, but now the profiles of the units will also be shown.

That's it

Close, but no cigar. What has been described so far allows you to create some basic army lists. For more complex lists you will need the advanced features of the AGEN framework.

AGEN advance framework

The starting point for the advanced framework is the following file. It does only contain things you have already read about.

Note that the unreadable start contains all the dottiest. If you copy this to an editor with a wide line length, it will be much more readable.

The Titan Legion category has been fleshed out and Thunderhawk and Landing Craft choices have been added, because it could be done without any new knowledge.

```
(function () {

// AGEN.useGamSystem (EA,);

 var myUkGameSystem = AGEN.newGameSystem("Epic UK", "Brumbaer's Epic UK list Generator", function
() {
  AGEN.newArmyBook ("Space Marines", 1, "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA", function () {
   AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {

    var sniper = AGEN.newDatasheet ("sniper", 1075, ["Sniper Scout", "INF", "15cm", "5+", "4+",
"5+", "Heavy Bolter", "30cm", "AP5+",  ], " Scout, Infiltrator, Sniper");
    var tdreadnought =  AGEN.newDatasheet ( "tdreadnought",  1140,  ["Tactical Dreadnought", "AV",
"15cm", "4+", "4+", "4+", "Power Fist", "(base contact)", "(Assault Weapons), MW, EA(+1)", "", "",
"", "", "", "", "Assault Cannon", "30cm", "AP5+/AT5+",], " Walker.");
    var hdreadnought =  AGEN.newDatasheet ( "hdreadnought",  1141,  ["Hellfire Dreadnought", "AV",
"15cm", "4+", "4+", "4+", "Missile Launcher", "45cm", "AP5+/AT6+", "", "", "", "", "", "", "Twin
Lascannon", "45cm", "AT4+",], " Walker.");

    var captain =  AGEN.newDatasheet ( "captain",  1000,  ["Captain", "CH", "n/a", "n/a", "n/a",
"n/a", "Power Weapon", "(base contact)", "(Assault Weapons), MW, EA(+1)",  ], " Invulnerable Save,
Leader, Commander");
    var chaplain =  AGEN.newDatasheet ( "chaplain",  1010,  ["Chaplain", "CH", "n/a", "n/a", "n/a",
"n/a", "Power Weapon", "(base contact)", "(Assault Weapons), MW, EA(+1)",  ],  " Invulnerable Save,
Leader, Inspiring");
    var librarian =  AGEN.newDatasheet ( "librarian",  1020,  ["Librarian", "CH", "n/a", "n/a", "n/
a", "n/a", "Power Weapon", "(base contact)", "(Assault Weapons), MW, EA(+1)", "", "", "", "", "",
"", "Smite", "(15cm)", "(Small Arms), EA(+1), MW",  ], " Invulnerable Save, Leader");
    var supremecommander =  AGEN.newDatasheet ( "supremecommander",  1030,  ["Supreme Commander",
"CH", "n/a", "n/a", "n/a", "n/a", "Power Weapon", "(base contact)", "(Assault Weapons), MW, EA(+1)",
], " Invulnerable Save, Supreme Commander");
    var assault =  AGEN.newDatasheet ( "assault",  1040,  ["Assault", "INF", "30cm", "4+", "3+",
"5+", "Chainswords", "(base contact)", "(Assault Weapons)", "", "", "", "", "", "", "Bolt pistols",
"(15cm)", "(Small Arms)",  ], " Jump Packs");
    var bike =  AGEN.newDatasheet ( "bike",  1050,  ["Bike", "INF", "35cm", "4+", "3+", "4+",
"Chainswords", "(base contact)", "(Assault Weapons)", "", "", "", "", "", "", "Bolters", "(15cm)",
"(Small Arms)",  ], " Mounted");
    var devastator = AGEN.newDatasheet ( "devastator", 1060,["Devastator", "INF", "15cm", "4+",
"5+", "3+", "2x Missile Launcher", "45cm", "AP5+/AT6+",  ]);
    var scout =  AGEN.newDatasheet ( "scout",  1070,  ["Scout", "INF", "15cm", "5+", "4+", "5+",
"Heavy Bolter", "30cm", "AP5+",  ], " Scout, Infiltrator");
    var tactical = AGEN.newDatasheet ( "tactical", 1080,["Tactical", "INF", "15cm", "4+", "4+",
"4+", "Bolters", "(15cm)", "(Small Arms)", "", "", "", "", "", "", "Missile Launcher", "45cm",
"AP5+/AT6+",  ]);
    var terminator =  AGEN.newDatasheet ( "terminator",  1090,  ["Terminator", "INF", "15cm", "4+",
"3+", "3+", "Power Weapons", "(base contact)", "(Assault Weapons), MW, EA(+1)", "", "", "", "", "",
"", "Storm Bolters", "(15cm)", "(Small Arms)", "", "", "", "", "", "", "2x Assault Cannon", "30cm",
"AP5+/AT5+",  ], " Reinforced Armour, Teleport, Thick Rear Armour");
    var attackbike = AGEN.newDatasheet ( "attackbike", 1100,["Attack Bike", "LV", "35cm", "4+",
"5+", "4+", "Heavy Bolter", "30cm", "AP5+",  ]);
    var landspeeder =  AGEN.newDatasheet ( "landspeeder",  1110,  ["Land Speeder", "LV", "35cm",
"4+", "6+", "5+", "Multi-melta", "(15cm)", "(Small Arms), MW", "", "", "", "", "", "", "", "15cm",
"MW5+",  ], " Skimmer, Scout");
```

```
    var landspeedertornado =  AGEN.newDatasheet ( "landspeedertornado",  1120,  ["Land Speeder
Tornado", "LV", "35cm", "4+", "6+", "5+", "Assault Cannon", "30cm", "AP5+/AT5+", "", "", "", "", "",
"", "Heavy Bolter", "30cm", "AP5+",  ], " Skimmer, Scout");
    var landspeedertyphoon =  AGEN.newDatasheet ( "landspeedertyphoon",  1130,  ["Land Speeder
Typhoon", "LV", "35cm", "4+", "6+", "5+", "Twin Typhoon Missile Launcher", "45cm", "AP3+/AT5+", "",
"", "", "", "", "", "", "Heavy Bolter", "30cm", "AP5+",  ], " Skimmer, Scout");
    var dreadnought =  AGEN.newDatasheet ( "dreadnought",  1140,  ["Dreadnought", "AV", "15cm",
"4+", "4+", "4+", "Missile Launcher", "45cm", "AP5+/AT6+", "", "", "", "", "", "", "Twin Lascannon
OR", "45cm", "AT4+", "", "", "", "", "", "", "", "Power Fist", "(base contact)", "(Assault Weapons), MW,
EA(+1)", "", "", "", "", "", "", "Assault Cannon", "30cm", "AP5+/AT5+",  ], " Walker. A Dreadnought
is armed with a Missile Launcher and Twin Lascannon (Hellfire) OR a Power Fist and Assault Cannon
(Tactical), not both – select one option before the game.");
    var hunter = AGEN.newDatasheet ( "hunter", 1150,["Hunter", "AV", "30cm", "5+", "6+", "6+",
"Hunter-Killer", "60cm", "AT4+/AA4+",  ]);
    var landraider =  AGEN.newDatasheet ( "landraider",  1160,  ["Land Raider", "AV", "25cm", "4+",
"6+", "4+", "Twin Heavy Bolter", "30cm", "AP4+", "", "", "", "", "", "", "2x Twin Lascannon",
"45cm", "AT4+",  ], " Reinforced Armour, Thick Rear Armour, Transport: (may carry 1 Terminator unit
OR 2 of the following units: Tactical, Devastator)");
    var predatorannihilator = AGEN.newDatasheet ( "predatorannihilator", 1170,["Predator
Annihilator", "AV", "30cm", "4+", "6+", "5+", "2x Lascannon", "45cm", "AT5+", "", "", "", "", "",
"", "Twin Lascannon", "45cm", "AT4+",  ]);
    var predatordestructor = AGEN.newDatasheet ( "predatordestructor", 1180,["Predator Destructor",
"AV", "30cm", "4+", "6+", "4+", "2x Heavy Bolter", "30cm", "AP5+", "", "", "", "", "", "",
"Autocannon", "45cm", "AP5+/AT6+",  ]);
    var razorback =  AGEN.newDatasheet ( "razorback",  1190,  ["Razorback", "AV", "30cm", "5+",
"6+", "5+", "Twin Heavy Bolter OR", "30cm", "AP4+", "", "", "", "", "", "", "Twin Lascannon",
"45cm", "AT4+",  ], " Transport (may carry 1 of the following units: Tactical, Devastator & Scout) A
Razorback is armed with either a Twin Heavy Bolter OR a Twin Lascannon, not both – select one option
before the game.");
    var razorbackAT =  AGEN.newDatasheet ( "razorbackAT",  1190,  ["Razorback AT", "AV", "30cm",
"5+", "6+", "5+", "Twin Lascannon", "45cm", "AT4+",  ], " Transport (may carry 1 of the following
units: Tactical, Devastator & Scout) A Razorback is armed with either a Twin Heavy Bolter OR a Twin
Lascannon, not both – select one option before the game.");
    var razorbackAP =  AGEN.newDatasheet ( "razorbackAP",  1190,  ["Razorback AP", "AV", "30cm",
"5+", "6+", "5+", "Twin Heavy Bolter", "30cm", "AP4+", ], " Transport (may carry 1 of the following
units: Tactical, Devastator & Scout) A Razorback is armed with either a Twin Heavy Bolter OR a Twin
Lascannon, not both – select one option before the game.");
    var rhino =  AGEN.newDatasheet ( "rhino",  1200,  ["Rhino", "AV", "30cm", "5+", "6+", "6+",
"Storm Bolter", "(15cm)", "(Small Arms)",  ], " Transport: (May carry 2 of the following units:
Tactical, Devastator and Scout)");
    var vindicator =  AGEN.newDatasheet ( "vindicator",  1210,  ["Vindicator", "AV", "25cm", "4+",
"6+", "4+", "Demolisher", "30cm", "AP3+/AT4+, Ignore Cover",  ], " Walker");
    var whirlwind = AGEN.newDatasheet ( "whirlwind", 1220,["Whirlwind", "AV", "30cm", "5+", "6+",
"5+", "Whirlwind", "45cm", "1BP, Indirect Fire",  ]);
    var droppod =  AGEN.newDatasheet ( "droppod",  1230,  ["Drop Pod", "Special", "Immobile", "5+",
"n/a", "n/a", "Deathwind", "15cm", "AP5+/AT5+",  ], " Planetfall, Transport: (may carry 1 formation
that includes only Tactical, Devastator and Dreadnought units). Deathwind: After the drop pod lands,
its Deathwind attacks all enemy units within 15cm. Each enemy formation attacked receives a Blast
marker for coming under fire, and an extra Blast marker for each casualty. Then any troops carried
in the drop pod must disembark within 5cm of the drop pod or within 5cmof another unit from the same
formation that has already landed, so long as all units are placed within 15cm of the drop pod. Drop
pods may not be used to claim crossfire.");
    var reavertitan =  AGEN.newDatasheet ( "reavertitan",  1240,  ["Reaver Titan", "WE", "20cm",
"4+", "3+", "3+", "2x Turbolaser Destructor", "60cm", "4x AP5+/AT3+, Fwd", "", "", "", "", "", "",
"Rocket Launcher", "60cm", "3BP,FxF",  ], " DC6, 4 Void Shield, Fearless, Reinforced Armour, Walker
May step over units and impassable or dangerous terrain that is lower than the Titan's knees and up
to 2cm wide. Critical Hit Effect: Roll a D6 in the end phase of every turn. 1: Reactor explodes -
Destroyed, 2-3: an extra point DC, 4-6: Reactor repaired. If destroyed any units within 5cms will be
hit on a roll of 5+.");
    var warhoundtitan =  AGEN.newDatasheet ( "warhoundtitan",  1250,  ["Warhound Titan", "WE",
"30cm", "5+", "4+", "4+", "Vulcan Mega-Bolter", "45cm", "4x AP3+/AT5+, Fwd", "", "", "", "", "", "",
"Plasma Blastgun", "45cm", "2x MW2+, Slow-Firing, Fwd",  ], " DC3, 2 Void Shields, Fearless,
Reinforced Armour, Walker May step over units and impassable or dangerous terrain that is lower than
the Titan's knees and up to 2cm wide. Critical Hit Effect: Move it D6cm in a random direction. If
```

```
this move takes the Warhound into impassable terrain or another unit it can't move over then it
stops immediately and is destroyed. Any units moved or contected will take a hit on a D6 roll of
6+.");
     var warlordtitan =  AGEN.newDatasheet ( "warlordtitan",  1260,  ["Warlord Titan", "WE", "15cm",
"4+", "2+", "3+", "2x Turbolaser Destructor", "60cm", "4x AP5+/AT3+, Fwd", "", "", "", "", "", "",
"Gatling Blaster", "60cm", "4x AP4+/AT4+, Fwd", "", "", "", "", "", "", "Volcano Cannon", "90cm",
"MW2+, TK(D3), Fwd",  ], " DC8, 6 Void Shields, Reinforced Armour, Thick Rear Armour, Fearless,
Walker. May step over units and impassable or dangerous terrain that is lower than the Titan's knees
and up to 2cm wide. Critical Hit Effect: Roll a D6 in the end every turn. 1: Reactor explodes -
Destroyed, 2-3: an extra point DC, 4-6: Reactor repaired. If destroyed any units within 5cms will be
hit on a roll of 4+.");
     var landingcraft =  AGEN.newDatasheet ( "landingcraft",  1270,  ["Landing Craft", "AC/WE",
"Bomber", "4+", "5+", "3+", "Storm Bolters", "(15cm)", "(Small Arms)", "", "", "", "", "", "", "3x
Twin Heavy Bolter", "15cm", "AP4+/AA5+", "", "", "", "", "", "", "2x Twin Lascannon", "45cm",
"AT4+",  ], " DC4, Planetfall, Fearless, Reinforced Armour, Transport: (May carry 12 Space Marine
infantry units, Attack Bikes and Dreadnoughts. Terminators and Dreadnoughts take up 2 spaces each.
In addition, the Landing Craft can carry 4 vehicles based on the Land Raider or 6 based on the
Rhino, or one of the following combinations: 3 Land Raiders and 1 Rhino, 2 Land Raiders and 3
Rhinos, or 1 Land Raider and 4 Rhinos) Critical Hit Effect: The Landing Craft's magazine explodes,
destroying the drop ship and anybody on board. Any units within 5cm of the Landing Craft suffer one
hit.");
     var thunderhawkgunship =  AGEN.newDatasheet ( "thunderhawkgunship",  1280,  ["Thunderhawk
Gunship", "AC/WE", "Bomber", "4+", "6+", "4+", "Twin Heavy Bolter", "15cm", "AP4+/AA5+, RF", "", "",
"", "", "", "", "Twin Heavy Bolter", "15cm", "AP4+/AA5+, LF", "", "", "", "", "", "", "2x Twin Heavy
Bolter", "30cm", "AP4+/AA5+, FxF", "", "", "", "", "", "", "Battle Cannon", "75cm", "AP4+/AT4+,
FxF",  ], " DC2, Planetfall, Reinforced Armour, Transport: (May carry 8 of the following units:
Tactical, Assault, Devastator, Scout, Bike, Attack Bike, Terminator and Dreadnought. Terminators and
Dreadnoughts take up 2 spaces each.) Critical Hit Effect: The Thunderhawk's control surfaces are
damaged. The pilot loses control and the Thunderhawk crashes to the ground, killing all on board.");
     var thunderboltfighterbomber = AGEN.newDatasheet ( "thunderboltfighter-bomber", 1290,
["Thunderbolt Fighter-Bomber", "AC", "Fighter-Bomber", "6+", "n/a", "n/a", "Stormbolters", "15cm",
"AP4+/AA5+, FxF", "", "", "", "", "", "", "Multilaser", "30cm", "AP5+/AT6+/AA5+, FxF", "", "", "",
"", "", "", "Underwing Rockets", "30cm", "AT4+, FxF",  ]);
     var marauderbomber = AGEN.newDatasheet ( "marauderbomber", 1300,["Marauder Bomber", "AC",
"Bomber", "4+", "n/a", "n/a", "2x Twin Heavy Bolter", "15cm", "AA5+", "", "", "", "", "", "", "Bomb
Racks", "15cm", "3BP,FxF", "", "", "", "", "", "", "Twin Lascannon", "45cm", "AT4+/AA4+, FxF",  ]);
     var battlebarge =  AGEN.newDatasheet ( "battlebarge",  1310,  ["Battle Barge", "SC", "n/a", "n/
a", "n/a", "n/a", "Orbital Bombardment", "n/a", "14BP, MW",  ], " Transport: (May carry 60 of the
following units: Space Marine Tactical, Assault, Devastator, Scout, Bike, Terminator or Dreadnought
units; plus 60 of the following units: Rhinos, Land Raiders, Razorbacks, Hunters, Whirlwinds,
Predators or Vindicators; plus 9 Thunderhawks and enough Drop Pods or Landing Craft to carry any
other units on board) Slow and steady: may not be used on the first two turns of a battle unless the
scenario specifically says otherwise.");
     var strikecruiser =  AGEN.newDatasheet ( "strikecruiser",  1320,  ["Strike Cruiser", "SC", "n/
a", "n/a", "n/a", "n/a", "Orbital Bombardment", "n/a", "5BP,MW",  ], " Transport: (May carry 20 of
the following units: Space Marine Tactical, Assault, Devastator, Scout, Bike, Terminator or
Dreadnought units; plus 20 of the following units: Rhinos, Land Raiders, Razorbacks, Hunters,
Whirlwinds, Predators or Vindicators; plus 6 Thunderhawks and enough Drop Pods or Landing Craft to
carry any other units on board)");


    AGEN.newCategory("Detachments", function() {
     AGEN.newFormation("Thunderhawk", 200, function() {
      AGEN.newUnit(thunderhawkgunship, 0, 1);
     });

     AGEN.newFormation("Landing Craft", 350, function() {
      AGEN.newUnit(landingcraft, 0, 1);
     });
    });


    AGEN.newCategory("Titan Legion", function() {
     AGEN.newFormation("Warlord Titan", 850, function() {
```

```
       AGEN.newUnit(warlordtitan, 1);
      });

      AGEN.newFormation("Reaver Titan", 650, function() {
       AGEN.newUnit(reavertitan, 1);
      });

      AGEN.newFormation("Warhound Titan", 275, function() {
       AGEN.newUnit(warhoundtitan, 1);
      });

      AGEN.newFormation("Warhound Titan Pack", 500, function() {
       AGEN.newUnit(warhoundtitan, 2);
      });
     });

     AGEN.newCategory("Imperial Navy Aircraft", function () {
      AGEN.newFormation("Thunderbolt Squadron", 150, function() {
       AGEN.newUnit(thunderboltfighterbomber, 2);
      });

      AGEN.newFormation("Marauder Squadron", 250, function() {
       AGEN.newUnit(marauderbomber, 2);
      });
     });

    });
   });
  });
})();
```

You will realize that this gets rather long, so in future, I will only list changes and their immediate surroundings.

AGEN.xxxNotes

We already encountered notes when we defined our datasheets. These notes are displayed in the datasheet list, directly below the respective datasheet.

In addition there are notes that will be displayed below the datasheet list. These notes are valid for all of the army.

You can define those notes for army books and army lists.

The notes defined for an army book will be copied to all of it's army lists, before their finit routine is called.

```
AGEN.addNotes (text)
```

will add further notes at the end of existing notes.

```
AGEN.preNotes (text)
```

will add notes before existing notes.

text is the html string to be displayed. Being an html string allows some fancy formatting for the notes. The Warmaster lists use that feature to build spell tables.

The notes will be added to the element to which the containing fInit function belongs to.

This lengthy paragraph will add all the common Space Marine special rules to the Space Marine army book and therefore all Space Marine army lists.

You could split the single call to AGEN.addNotes in multiple calls, if you wanted, for better readability or modularity.

```
           :
     AGEN.newArmyBook ("Space Marines", 1, "#C5DFFA", "#223355", "#F0F8FF", "#C5DFFA", function ()
{

           AGEN.addNotes ("<p class='title'>1.0.1 AND THEY SHALL KNOW NO FEAR</p><p>Space Marines
are renowned for their tenacity and bravery. This is represented by the following rules:</p><p>It
```

```
takes 2 Blast markers to suppress a Space Marine unit or kill a unit in a broken formation (ignore
any leftover Blast markers). Space Marine formations are only broken if they have 2 Blast markers
per unit in the formation. </p>" +
                "<p>Space Marine formations count as having half their number of blast markers in
assault resolution (rounding down to a minimum of one blast marker). Halve the number of extra hits
suffered by a Space Marine formation that loses an assault, rounding down in favour of the Space
Marines. </p>" +
                "<p>When a broken Space Marine formation rallies, it receives a number of Blast
markers equal to the number of units, rather than half this number. Space Marines with the Leader
special ability remove 2 Blast markers instead of 1. </p>" +
                "<p class='title'>1.0.2 SPACE MARINES TRANSPORTS</p><p>The Space Marines are a
highly mobile army. Because of this, the points cost of a detachment usually includes enough Rhino
transport vehicles to transport it and any upgrades that may have been taken. Determine the number
of Rhinos needed after all upgrades have been purchased. The number of Rhinos will always be the
minimum number needed to carry the formation; you cannot take extras along to cover any losses. Note
that many formations don't receive Rhinos, usually because they can't fit into them. Detachments
that come with Rhinos will be noted as having 'plus transport' in the unit's section of the army
lists below. </p><p>Also note that you don't have to take Rhinos if you don't want to. If you'd
rather field formations on foot instead, so they can act as garrisons for example, or be transported
in a Thunderhawk Gunship, then you may do so. </p><p>In addition, you may choose to replace a
detachment's Rhinos with Drop Pods. If you do this then the detachment will enter play in a Drop Pod
using the rules for Planetfall (see section 4.4 of the Epic: Armageddon main rule book). Note that
if you choose to do this you will also require at least one Space Marine Strike cruiser or Battle
Barge to deploy the Drop Pods from. If the detachment contains units that can't be deployed in Drop
Pods this option can not be chosen.</p><p>Choosing transport options is part of the army selection
process. Portions of a formation may be left behind during deployment (to garrison, for example) and
the decision to exchange options, even free ones, must be determined when the army list is
determined.</p>" +
                "<p>*Please note: Chapter-specific transport vehicles carry their particular
Chapter's troop types and therefore are often renamed from the standard transport vehicle, e.g.
Rhino, becomes Dark Angels Rhino in the Dark Angels Chapter list and Drop Pod, becomes Space Wolves
Drop pod in that list and so on.</p>"
            );
            :
```

I usually define the notes for army book at the very start of the army books flnit function, so they are out of the way.

For our Codex Astartes list we now only have to add some notes as most is covered by the army book's notes.

As we want those notes to be shown before the army book notes, we use AGEN.preNotes.

```
            :
                AGEN.newArmyList("Codex Astartes", "EAUKCA", 100, function () {
                    AGEN.preNotes("<p>Codex Astartes Space Marine armies have a Strategy rating
of 5. All Space Marine and Titan Legion formations have an initiative rating of 1+. Imperial Navy
aircraft formations have an initiative rating of 2+.</p>");
            :
```

In this case the definition is added at the very beginning of the army lists's flnit. Usually I put it directly behind the datasheet definitions, but that would have made the listing so much longer.

AGEN choices

Sometimes entries in the army lists influence each other. A typical case is the Strike Cruiser formation. This can consist either of a single Strike Cruiser or a single Battle Barge.

Using our current knowledge, we would define our formation, which costs 200 points and may occur only once in any army.

```
            :
      AGEN.newFormation("Strike Cruiser", 200, 0, 1, function() {
        AGEN.newUnit(strikecruiser, 0, 1, 0, 1);
        AGEN.newUnit(battlebarge, 150, 0, 0, 1);
      });
```

```
          :
```

The formation is defined with minCount and maxCount, as we can only have between 0 and 1 of those formations in any army.

The formation costs 200 points.

When we start out we have 1 Strike Cruiser, which costs 0 points of those 200 points for the formation.

It's minCount and maxCount are 0 and 1, because we may have one, but must have none.

The Battle Charge is defined in the same way. It will cost an additional 150 points. The formation will start with 0 and may have 0 or 1.

So far everything seems to be ok, but when we insert this into our myAGen.js and run it, we will see that we start out correctly with 1 Strike Cruiser, but can have any combination of Strike Cruiser and Battle Barge.

AGEN.newChoice

A choice will ensure that the number of choices of all possible options will be within specified limits.

Typical uses are limits to the number of upgrades or if you have to choose one type over others.

```
AGEN.newChoice (lowerLimit, upperLimit, list)
```

lowerLimit specifies the minimum number of choices that must be chosen from all options of the elements of list.

upperLimit specifies the maximum number of choices that may be chosen from all options of the elements of list.

list a list of options. The elements of the list must be children of the element that owns the embedding fInit function. I.e. in the fInit of an formation, the children must be units or datasheets, in the fInit of a category, this may be formations, units or datsheets.

For the Strike Cruiser formation, this will result in

```
          :
    AGEN.newFormation("Strike Cruiser", 200, 0, 1, function() {
     AGEN.newChoice(1, 1, [
        AGEN.newUnit(strikecruiser, 0, 1, 0, 1),
        AGEN.newUnit(battlebarge, 150, 0, 0, 1)
       ]);
    });
          :
```

Note that there are no changes to any of our definitions. But the definitions for the units were put into a list and send as parameters to AGEN.newStandsChoice.

When you look at the Codey Astartes, you will se that the commander upgrade is quite common.

Expressed as a choice, we would get

```
AGEN.newChoice(0, 1, [
 AGEN.newUnit(supremecommander, 100, 0),
 AGEN.newUnit(captain, 50, 0),
 AGEN.newUnit(chaplain, 50, 0),
 AGEN.newUnit(librarian, 50, 0),
]);
```

Because we need this often, we make use of the fact that we use a programming language and put this into a function. As different parts of the army list will need this function, we define it directly behind the AGEN.preNotes within the army list's fInit.

```
          :
AGEN.preNotes("<p>Codex Astartes Space Marine armies have a Strategy rating of 5. All Space Marine
and Titan Legion formations have an initiative rating of 1+. Imperial Navy aircraft formations have
an initiative rating of 2+.</p>");
```

```
var commanderUpgrade = function () {
  AGEN.newChoice(0, 1, [
    AGEN.newUnit(supremecommander, 100, 0),
    AGEN.newUnit(captain, 50, 0),
    AGEN.newUnit(chaplain, 50, 0),
    AGEN.newUnit(librarian, 50, 0),
  ]);
}
```

```
:
```

Note the short form of AGEN.newUnit, which sets min and max values to 0 and 1.

Whenever we need a commander upgrade in the armlets, we just call the function an do not have to rewrite the complete definition.

So defining the Whirlwind formation is simple

```
:
        AGEN.newFormation("Whirlwind Detachment", 300, function() {
             commanderUpgrade();
             AGEN.newUnit(whirlwind, 0, 4);
             AGEN.newUnit(hunter, 75, 0);
         });
:
```

As we know there must be only one Supreme Commander in the army. How do we achieve that ?

As it should test all formations in the army, the choice will have to be added to the army's flnit.

```
:
AGEN.newChoice(0, 1, [
]);
:
```

But what is there to choose from ?

We could know add all units which use a supreme commander stand. But for that we would have to remember all of them, which is ugly and tedious.

Instead for the unit we look for the datasheet that we defined and stored in the variable supreme commander as described somewhere above..

```
  :
AGEN.newChoice(0, 1, [
 supremecommander
]);
  :
```

AGEN.newStandsChoice

An other kind of choice if stands, can be chosen from different units, but the total number of stands is limited.

In our commanderUpgrade function or our Strike Cruiser definition we could replace AGEN.newChoice with AGEN.newStandsChoice and there would be no difference.

But the choice between Predator types could not be handled with AGEN.newChoice.

The Predator formation can have 4 Annihilators, 4 Destructors or 2 of each type.

Because we always have to take two of the same type the group size is 2. So we can have 2 groups of a single type or one group of each Type.

If we use AGEN.newChoice with limits of 1 and 2, we can create illegal combinations, like having only 2 Predators instead of 4.

A different way to express the restriction, we want to enforce is:

You must have always have 4 stands of predators and the type can only be changed in pairs.

The pair stuff is already taken care off, by using a group size of two.

The stand limit is taken care of by

```
AGEN.newStandsChoice (lowerLimit, upperLimit, list)
```

lowerLimit specifies the minimum number of stands that must be chosen from all options of the elements of list.

upperLimit specifies the maximum number of stands that may be chosen from all options of the elements of list.

list a list of options. The elements of the list must be children of the element that owns the embedding fInit function. I.e. in the fInit of an formation, the children must be units or datasheets, in the fInit of a category, this may be formations, units or datsheets.

Here is the complete definition of the Predator formation.

```
                AGEN.newFormation("Predator Detachment", 225, function() {

                    commanderUpgrade();
                    AGEN.newStandsChoice(4, 4, [
                        AGEN.newUnit(predatorannihilator, 12.5, 0, 0, 2, 2),
                        AGEN.newUnit(predatordestructor, 0, 2, 0, 2, 2)
                    ]);
                    AGEN.newUnit(vindicator, 50, 0, 0, 2);
                    AGEN.newUnit(hunter, 75, 0, 0, 1);
                });
```

Because the minimum number of stands is 4, 4 stands - in form of 2 groups - must also be chosen as start configuration.

How you select the start composition is up to you, but remember to set the formations points cost accordingly.

Now we know enough to define all formations except those that have the transport option, namely Tacticals, Devastators and Scouts.

AGEN.newExclusiveChoice

We start with the Scout formation, which is the easiest.

```
          AGEN.newFormation("Scout Detachment", 150, function() {
            commanderUpgrade();
            AGEN.newChoice (1, 1, [
              AGEN.newUnit(scout, 0, 1, 0, 1, 4),
              AGEN.newUnit(sniper, 12.5, 0, 0, 1, 4),
            ]);

             AGEN.newStandsChoice(0, 4, [
               AGEN.newUnit(razorbackAP, 25, 0, 0, 4),
               AGEN.newUnit(razorbackAT, 25, 0, 0, 4),
             ]);
             AGEN.newUnit(rhino, 0, 0, 1, 1, 3);
             AGEN.newUnit(droppod, 0, 0);
          });
```

Not that the Sniper upgrade is realized as a troop choice and that the Razorbacks are divided in two type, the one with AP and the one with AT weapon.

You are not allowed to combine droppods neither with Rhinos, nor with Razorbacks. AGEN.newExclusiveChoice will give us that behavior.

```
AGEN.newExclusiveChoice (list1, list2)
```

list1 and ***list2*** are two lists of options.. If any one in list1 is chosen, you will not be able to choose any option in list2 and vice versa.

```
          AGEN.newFormation("Scout Detachment", 150, function() {
            commanderUpgrade();
            AGEN.newChoice (1, 1, [
              AGEN.newUnit(scout, 0, 1, 0, 1, 4),
              AGEN.newUnit(sniper, 12.5, 0, 0, 1, 4),
            ]);

          AGEN.newExclusiveChoice (
```

```
            [
              AGEN.newStandsChoice(0, 4, [
               AGEN.newUnit(razorbackAP, 25, 0, 0, 4),
               AGEN.newUnit(razorbackAT, 25, 0, 0, 4),
              ]),
              AGEN.newUnit(rhino, 0, 0, 0, 1, 3)
            ],
            [
              AGEN.newUnit(droppod, 0, 0)
            ]);
        });
```

The only thing still not right is the number of Rhinos, which would have to change with number of Razorbacks.

AGEN.computeStands

We change the definition of the Rhino to

```
              AGEN.newUnit(rhino, 0, 0, 0, 1, function () {
              })
```

This will add an flnit function and give us just one Rhino or none. See it as an Rhino on/off switch.

The real number of stands in that unit of Rhinos shall be computed, this can be done with

```
AGEN.computeStands(func)
```

func is the function used to compute the number of stands. It has to return the number of stands the unit is supposed to have. If the result is negative it will be set to zero.

```
      AGEN.computeStands(function() {
         return Math.ceil( (4 - AGEN.standsInUnitsWithDatasheet(razorbackAP)
                           - AGEN.standsInUnitsWithDatasheet(razorbackAT)) / 2);
       });
```

AGEN.standsInUnitsWithDatasheet does just what you think. It is described somewhere further down with the other Query Functions.

So the number of uncared passenger stands is 4 (the number of passengers we could have used AGEN.standsInUnitsWithDatasheet to compute it, but as there are always 4 passengers why should we) - the number of Razorbacks AP and AT. As two fin in a Rhino this number is divided by two and rounded up.

So the complete definition of the Scout detachment looks like

```
AGEN.newFormation("Scout Detachment", 150, function() {
                       commanderUpgrade();
   AGEN.newChoice (1, 1, [
    AGEN.newUnit(scout, 0, 1, 0, 1, 4),
    AGEN.newUnit(sniper, 12.5, 0, 0, 1, 4),
   ]);
                      AGEN.newExclusiveChoice(
                          [
                              AGEN.newStandsChoice(0, 4, [
                                  AGEN.newUnit(razorbackAP, 25, 0, 0, 4),
                                  AGEN.newUnit(razorbackAT, 25, 0, 0, 4),
                              ]),
                              AGEN.newUnit(rhino, 0, 0, function() {
                                  AGEN.computeStands(function() {
                                      return Math.ceil( (4
                                        - AGEN.standsInUnitsWithDatasheet(razorbackAT)
                                        - AGEN.standsInUnitsWithDatasheet(razorbackAP)
                                       ) / 2);
                                  });
                              })
                          ],
                          [
                              AGEN.newUnit(droppod, 0, 0)
                          ]
                      );
```

```
                    });
```

The transport option for the Tacticals would be identical, but would start out with 6 instead of 4 passengers.

AGEN.newChoiceWithGroups

Choices use arrays to hold their options . If one of the options is a choice (called subchoice), the options of the subchoice are treated as if they would be part of the outer choice.

Sometimes you want the subchoice to be treated as one single choice. Typical use: An army list allows two upgrades, but one of the upgrades consists of different options. The Leman Russ upgrade can consist of Demolishers or standard Russ. But for the number of upgrades they should be counted as 1 choice made.

AGEN Query Functions

Query Functions return some information about some element(s) of the army.

We'v already seen

AGEN.standsInUnitsWithDatasheet

Like most Query Functions it comes in 4 different flavors

```
AGEN.standsInUnitsWithDatasheet (id)
AGEN.standsInUnitsWithDatasheet (idArray)
AGEN.standsInUnitsWithDatasheet (datasheet)
AGEN.standsInUnitsWithDatasheet (datasheetArray)
```

id the id of any datasheet in the army as defined in AGEN.newDatasheet

idArray an array of ids of datasheets in the army as defined in AGEN.newDatasheet

datasheet any datasheet in the army

datasheet Array an array of datasheets

If you enter id(s) the function looks up the corresponding datasheets and proceeds as it would have called one of the datasheet variants of the function.

The function sums the number of stands who are based on the specified datasheets. The function searches all children of the current element (i.e. the element in which fInit function the Query Function is called).

AGEN.standsInThisFor

```
AGEN.standsInThisFor (name)
AGEN.standsInThisFor (nameArray)
AGEN.standsInThisFor (element)
AGEN.standsInThisFor (element)
```

name the name of any element (category, formation, unit, etc) in the army

name***Array*** an array of names as described above

element any element in the army

element Array an array of elements

If you enter name(s) the function looks up the corresponding elements and proceeds as it would have called one of the element variants of the function.

The function sums the number of stands of all specified elements. The function searches all children of the current element.

AGEN.pointsInThisFor

```
AGEN.pointsInThisFor (name)
AGEN.pointsInThisFor (nameArray)
AGEN.pointsInThisFor (element)
AGEN.pointsInThisFor (element)
```

name the name of any element (category, formation, unit, etc) in the army

name***Array*** an array of names as described above

element any element in the army

element Array an array of elements

If you enter name(s) the function looks up the corresponding elements and proceeds as it would have called one of the element variants of the function.

The function sums the points of all the specified elements. The function searches all children of the current element.

AGEN.standsInArmyFor

AGEN.pointsInArmyFor

These are identical to their siblings, but will always start searching for elements as if the function had been started in the army fInit function.

We have nearly all detachments defined, the last one missing is the Devastator detachment.

With our current knowledge we would define it as (the dreadnought update is like the commanderUpgrade function. It defines the choice of Dreadnoughts)

```
AGEN.newFormation("Devastator Detachment", 250, function() {
  commanderUpgrade();
  AGEN.newUnit(devastator, 4);
  dreadnoughtUpgrade();

  AGEN.newExclusiveChoice(
   [
      AGEN.newUnit(vindicator, 50, 0, 0, 2),
      AGEN.newUnit(hunter, 75, 0),
      AGEN.newUnit(landraider, 75, 0, 0, 4),
      AGEN.newStandsChoice(0, 4, [
        AGEN.newUnit(razorbackAP, 25, 0, 0, 4),
        AGEN.newUnit(razorbackAT, 25, 0, 0, 4),
      ]),
      AGEN.newUnit(rhino, 0, 0, function() {
        AGEN.computeStands(function() {
          return Math.ceil( (AGEN.standsInUnitsWithDatasheet(devastator)
             - AGEN.standsInUnitsWithDatasheet(razorback)
             - 2 * AGEN.standsInUnitsWithDatasheet(landraider)) / 2);
        });
      })
    ],
    [
      AGEN.newUnit(droppod, 0, 0)
    ]
  );
});
```

That leaves the problem that I could select more Razorbacks than allowed. It would be possible to correct the maximum number on Razorbacks on the fly, but when it changes the number of Razorbacks might have to change as well. And when the maximum changes again, should the number of Razorbacks return to the intended number or stay at the current number or …

We leave it alone. We do not correct the number of Razorbacks, but we will issue a warning when too many Razorbacks are chosen.

AGEN.newErrorCheck

```
AGEN.newErrorCheck (func)
```

func is the function doing the checking. If it return an empty string everything is ok. If it return a non empty string this string will be displayed in the error section of the editor.

We want to check the number of Razorbacks in the in the Devastator Detachment, so we add the check to the fInit function of that detachment.

```
AGEN.newFormation("Devastator Detachment", 250, function() {
  commanderUpgrade();
  AGEN.newUnit(devastator, 4);
  dreadnoughtUpgrade();

  AGEN.newExclusiveChoice(
   [
      AGEN.newUnit(vindicator, 50, 0, 0, 2),
      AGEN.newUnit(hunter, 75, 0),
      AGEN.newUnit(landraider, 75, 0, 0, 4),
      AGEN.newStandsChoice(0, 4, [
         AGEN.newUnit(razorbackAP, 25, 0, 0, 4),
         AGEN.newUnit(razorbackAT, 25, 0, 0, 4),
      ]),
      AGEN.newUnit(rhino, 0, 0, function() {
         AGEN.computeStands(function() {
            return Math.ceil( (AGEN.standsInUnitsWithDatasheet(devastator)
               - AGEN.standsInUnitsWithDatasheet(razorback)
               - 2 * AGEN.standsInUnitsWithDatasheet(landraider)) / 2);
            });
         })
    ],
    [
      AGEN.newUnit(droppod, 0, 0)
    ]
  );

  AGEN.newErrorCheck (function () {
    var n = AGEN.standsInUnitsWithDatasheet(razorbackAP)
          + AGEN.standsInUnitsWithDatasheet(razorbackAT);

   return (n > 0 && n + 2 * AGEN.standsInUnitsWithDatasheet(landraider) > 4) ?
      "Formation has too many Razorbacks" : "";
  });
});
```

So we are nearly finished, the last thing missing is the one third points check for the Categories.

It looks like

```
   AGEN.newErrorCheck(function () {
     var gruppe1 = AGEN.pointsInArmyFor("Detachments");
     var gruppe2 = AGEN.pointsInArmyFor([
                       "Titan Legion",
                       "Imperial Navy Aircraft"
                   ]);
     return (2 * gruppe2 > gruppe1) ?
         "Titan Legion and Navy exceed their points limit of " + (gruppe1 / 2) + " points" : "";
});
```

Where to put it ?

As the test is related to the army list, it belongs into the army lists fInit.

Points

There are formation points and units points.

The total cost of the formation is calculated from those.

First the cost of all the stands the formation starts with is deducted from the formation cost. Then the cost of all stands currently in the formation are added.

If the number of stands of a unit can not change, you can set it's cost to any value like 0, because it is first subtracted and than added again.

Let's look at

```
AGEN.newFormation("Predator Detachment", 225, function() {

    commanderUpgrade();
    AGEN.newStandsChoice(4, 4, [
        AGEN.newUnit(predatorannihilator, 12.5, 0, 0, 2, 2),
        AGEN.newUnit(predatordestructor, 0, 2, 0, 2, 2)
    ]);
    AGEN.newUnit(vindicator, 50, 0, 0, 2);
    AGEN.newUnit(hunter, 75, 0, 0, 1);
});
```

Note that the cost of an Annihilator is 12.5 points, because that is it's point difference to an destructor.

Also noteworthy is that the group size of the Predators is 2, so you buy always two of the same kind together.

```
AGEN.newFormation("Predator Detachment", 275, function() {

    commanderUpgrade();
    AGEN.newStandsChoice(4, 4, [
        AGEN.newUnit(predatorannihilator, 12.5, 2, 0, 2, 2),
        AGEN.newUnit(predatordestructor, 0, 0, 0, 2, 2)
    ]);
    AGEN.newUnit(vindicator, 50, 0, 0, 2);
    AGEN.newUnit(hunter, 75, 0, 0, 1);
});
```

This will give the same result, but will now start with four Annihilators (two groups of two).

In the Epic UK Eldar list you buy an Engine of Vaul Troupe for 250 points, but upgrades cost only 200 points. This would be written as

```
AGEN.newFormation("Engines of Vaul", 250, function() {
   AGEN.newStandsChoice(1, 3, [
        AGEN.newUnit(scorpion, 200, 1, 0, 3),
        AGEN.newUnit(cobra, 200, 0, 0, 3),
        AGEN.newUnit(stormserpent, 200, 0, 0, 3)
    ]);
});
```