Damage model in flt-files
********** by Andy *** 05.12.2003 ***

This is a short guide on how to edit the aircraft's Damage Modells of the flight simulation 'European Air War' by Microprose.
I'll use Chris Coon's original flt-file code to make some comments on how the damage modells for EAW aircraft can be altered to get them more realistic.

At first a general note: Please get Alessandro Borges' '3DZ! Studio'. You will need it to locate positions of hitbubbles and other important points of the aircraft's frame. Further more you will need a Hex-Editor to make changes to the flt-file, you cannot do all changes with Aircraft Edit because the meaning of many bytes was still unknown when it was released.
Ok, lets start:

Most coordinates especially these of the hitbubbles, gun tracers and other important points are set in feet. That means we can simply read the 3dz coodinates with '3dz! Studio' and convert them to feet. This is possible because of Anthony's very valueable work to determine the scale of EAW.
To convert the 3dz coordinates to feet you have to multiply them with a factor of:

-0,14708723

Unfortunately this is only true for x- and z-coordinates.
Please multiply your 3dz values for the y-axis with +0,14708723.
After having programmed many flt files with more accurate data I've found that this seems to be a bug of original EAW. There is no consistent definition for the y-values. This caused a lot of confusion to me and wrong designations of some bytes in my previous release of this tutorial, but after extensive testing these mistakes are now ironed out.

So your flt-file related values should have the sign like in the following sheme.

x-axis:
front +
tail -

z-axis:

underside (belly) +
top side -

BUT:

y-axis:
left -
right +

The flt-file values are set in float32 Hex-numbers.

A little example: The 3dz-y-coordinate of the Ju 88A's LEFT wingtip is -227. Multiply it with the factor mentionned above and you get -33,4 feet. BTW: the wingspan of the real Ju 88 A-4 was about 66 feet. So this value is quite close to reality. ;o)

//////////////////////
//.flt file format
//////////////////////

As you know the flt-files contain the info on the Flight Modell and on the Damage Modell. I'll simply make comments at the positions that are relevant for the DMs. You can easily indentify these lines because I added the byte offset at the beginning of the line. All other code given below is needed for the Flight Modell. Get 'Knegel's' and mine notes on the Flight Modell if you are also interested in modifying the FM.

```
typedef struct fm_aero_type_data
{

    // engine data
    short                engine_count;
    short                engine_type;
    float                engine_alt;
```

```
float           engine_slope_low;
float           engine_slope_high;
float           max_prop_np;
float           min_throttle;
fm_control_data  dT;                //Throttle control
{
    float   max;                    //max position
    float   min;                    //min position
    float   rate;                   //rate of change
}
fm_control_data  de;                //elevator
{
    float   max;                    //max position
    float   min;                    //min position
    float   rate;                   //rate of change
}
fm_control_data  df;                //flaps
{
    float   max;                    //max position
    float   min;                    //min position
    float   rate;                   //rate of change
}
fm_control_data  da;                //aileron
{
    float   max;                    //max position
    float   min;                    //min position
    float   rate;                   //rate of change
}
fm_control_data  dr;                //rudder
{
    float   max;                    //max position
    float   min;                    //min position
    float   rate;                   //rate of change
}
```

```
// performance spec
float		max_alt;
float		max_vel;
float		max_alpha;
float		critical_mach;
float		max_g;
float		min_g;
float		wing_y_mac;

// inertial data
float		mass_empty;
float		mass_fuel;

float		inv_Rx;
float		inv_Ry;
float		inv_Rz;

float		Izy;
float		Ixz;
float		Iyx;

// stability coefficient
fm_aero_coefficient coefficient;
{
	// drag stability derivatives
	float	X0;
	float	XL;
	float	Xdf;
	float	XdT;
	float	XG;

	// sideforce stability derivatives
	float	Yb;
	float	Yr;
	float	YdT1;
```

```
float	YdT2;
float	YdT3;
float	YdT4;
float	Ydr;
float	Yda;

// lift stability derivatives
float	Z0;
float	Za;
float	Zad;
float	Zq;
float	Zde;
float	Zdf;

// roll stability derivatives (x)
float	Lb;
float	Lp;
float	Lr;
float	Lda;
float	Ldr;

float	LdT1;
float	LdT2;
float	LdT3;
float	LdT4;

// pitch stability derivatives (y)
float	M0;
float	Ma;
float	Mad;
float	Mq;
float	Mde;
float	Mdf;

// yaw stability derivatives (z)
```

```
        float       Nb;
        float       Np;
        float       Nr;
        float       Nda;
        float       Ndr;

        float       NdT1;
        float       NdT2;
        float       NdT3;
        float       NdT4;

    }

    fm_lookup_table     aX0;                //tables for interpolating values between zero_x and (zero_x + delta_x *
table_size)
    {
        long        table_size;
        float       delta_x;
        float       inv_delta_x;
        float       zero_x;
        float       aTable[10];
    }
    fm_lookup_table     aZa;
    {
        long        table_size;
        float       delta_x;
        float       inv_delta_x;
        float       zero_x;
        float       aTable[10];
    }
    fm_lookup_table     aEe;
    {
        long        table_size;
        float       delta_x;
        float       inv_delta_x;
```

```
        float       zero_x;
        float       aTable[10];
    }
    fm_lookup_table     aAe;
    {
        long        table_size;
        float       delta_x;
        float       inv_delta_x;
        float       zero_x;
        float       aTable[10];
    }
    fm_lookup_table     aRe;
    {
        long        table_size;
        float       delta_x;
        float       inv_delta_x;
        float       zero_x;
        float       aTable[10];
    }

    // weapons data
    fm_weapon_data      aWeapons[6];
    {
        fm_vector3              position;
        {
            float x;
            float y;
            float z;
        }
        float               secondary_pos_x;
        float               secondary_pos_y;
        float               muzzle_velocity;
        float               rof;
        float               dispersion;
        float               eject_pos1;
```

```
        float                   eject_pos2;
        short                   muzzle_id;
        short                   duration;
    }
```

>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>Damage Modell Description starts here<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>

byte offset

    // landing gear
       ============

035C   float   land_height;   //height of main gear (feet)
0360   float   land_pitch;   //angle of pitch when standing on ground (rad ---> 360°=2Pi)

I'm not really sure what these values are good for. Somehow the land_height seems to be the height of the landing gear when the plane is placed on the ground at mission beginning. As soon as you turn on the engine the byte values starting at offset 0370 are used. Land_pitch can be easily calculated with x- and z-coordinates that are used for main gear and tail wheel (see offset 0370). Wrong land_pitch values cause very strange/funny effects of jumping planes. It also influences the plane's ability to take off. Your aircraft may not take off if the angle of pitch is too high. Please reduce it in this case, even if it does not look right.
All in all setting up the gear heigth is a work of trial and error. If you plane is floating above the ground at mission beginning you need to reduce the bytes at offset 035C. If the plane pitch is not right edit bytes at offset 0360.

```
    fm_control_data     gear;
    {
        float       max;                    //max position
        float       min;                    //min position
        float       rate;                   //rate of change
    }
```

byte offset

```
    fm_gear_data        aGears[3];
    ============

    {
0370        float               n;

        fm_vector3              position;
        {
0374            float x;
0378            float y;
037C            float z;
        }
    }
```

Starting at byte 0374 you have three groups of 4 coodinates (n, x, y, z) each. These are the positions of the 2 main landing gear parts and the tail wheel. Just get the coordinates with '3dz! Studio' and convert them to feet. Pay attention: It can happen that your plane explodes on the ground because you changed (reduced) the landing gear height (z-value). This happens because any of the hitbubbles or the damage positions starting at byte offset 03A0 now hits the ground. Please correct them first.
I'm not sure about the n-value at the beginning of every line. I did some test but was not able to find any dependencies.

```
    // damage size
```

==========

```
03A0    fm_vector3              damage_pos[7];
        {
            float x;
            float y;
            float z;
        }
```

These are 7 damage positions used for crashes or belly landings: front_belly, mid_belly, tail_low, wingtip_left, wingtip_right, front_top, tail_top. The seven positions create something like an outline for the plane that is used to calculate crashes with groundobjects.
Use the lower wingtip coordinates for biplanes.
BTW: You will have very nice and realistic looking belly landings when these values are set properly. Also will it improove AI crash landings.

Chris Coons told me:
The damage_pos array is only used when landing without landing gear (or landing rough)... it seems to only be checked to determine drag along the ground and damage as the aircraft slides. Probably it was intended to be a more realistic damage model in the air (such as increased drag when one wing is shot), but we ran out of time so never got to put it in all the way.

```
        fm_vector3              wing_pos[2];
        {
03F4        float x;
            float y;
            float z;
        }
```

Don't know it's meaning, I think it controls the position where the torn off wings appear.

```
    // engine positions (smoke)
      ========================


    fm_vector3              engine_pos[4];
    {
040C        float x;
        float y;
        float z;
    }
```

Controls the position of the engine smoke.

```
    // hit locations
      =============


    fm_hit_data             aHitLocation[8];
    {
        fm_vector3              position;
        {
043C            float x;
            float y;
            float z;
        }
        float                   radius2;
```

These are the positions and sizes of the hitbubbles. You can use Aircraft Edit to change them. The locations are: front (engine for single engined planes), pilot, mid (fuselage), tail, inner_wing_right (right engine for twin engined planes), outer_wing_right (outer right engine for 4 engined planes), inner_wing_left (left engine), outer_wing_left (outer left

engine).
Pay attention with multi engined planes. While the wings contain only the mecanical parts for single engined planes they are also used for the engines on 2- or 4- engined planes. That's why you have to add the hitpoints of the engine to the hitpoints of the wings for multi engined aircraft.
The positions of the hitbubbles can easily estimated with '3dz! Studio' but should be extensively tested in game. Test every Hitbubble for itself setting the radius of all other hitbubbles to zero. Now change the plane you want to test to bomber and fly a mission, now 'attack' your comrades watching the hits at close distance and do corrections to the size and positions of the hitbubbles if necessary. You can also start a mission on the ground and slowly drive around your comrades shooting at them to test the position of the hitbubbles from different directions...
Radius2 means the radius squared of the hitbubble. Best you take the values from Knegels RP models.
The hitpoints themselves are stored in planes.dat and can easily be changed with Aircraft Edit.

```
    }
    long                BigRadius;
    float               radius2;
```

Chris Coons told me:
The value BigRadius is the radius of the bounding cube around the aircraft that the bullets are first checked against. If the bullet is within the cube, then it is checked against each hit location. Each hit location is a sphere with the same radius, which is the value in Radius2 (radius squared).
BigRadius is also used for determining plane-plane collision.
So ideally BigRadius would cover the entire aircraft, and the hit locations would be evenly spread out so the sum of the radius2 covered the entire body. But you could alter those to make a bullet that really "hit" the aircraft go through two different hit locations and it would become a "miss," for example.

```
    // engine damage points
    float               EngineRating;



    // weapons hardpoints
      ==================
```

```
fm_vector3              aHardPoints[5];
{
        float x;
        float y;
        float z;
}
```

These are the positions where the bombs appear when they are dropped. The positions should be the same as the 3dz-hardpoints positions.

Ok, that's it. Please excuse my English, these notes have been taken very quickly. :o)
If you have questions just email me or ask at the EAW forum at SimHQ.

http://oldsite.simhq.com/cgi-bin/ultimatebb.cgi?ubb=forum&f=41&DaysPrune=5&submit=Go

Good luck,
Andy (andy@wiesel-network.de)

Some more notes I copied from SimHQ:

Undercarriage Height Settings

Now, in order to achieve deck landings on a 3dz that does not actually support anything, I have been making the “virtual” undercarriage in the .FLT files for each aircraft VERY tall. You see, the values for undercarriage height in the .FLT files are expressed in feet. It is as simple as that (something had to be simple). You set the “land height” float value at offset # 35C to your desired height and the mission will commence with all squad members at that height. The HUD displays 0 ft which is why I think it is a “land” height. As soon as the player starts his engine, the correct altitude is displayed and he drops from this height. The AI squad members remain at that “land” height until the player takes them over. This 35C figure can be set to any height, 1000, 10 000, 30 000 ft.

The undercarriage maximum heights for each strut at offset # 37C, 38C and 39C are limited to 100 ft in height. Once you start your engine, your aircraft comes to rest at the z coordinate maximums for each gear strut. (This probably explains why some aircraft have exploded on engine start in the past). So if you set 35C at a compatible height (less than 100ft) your aircraft comes to rest on the z heights with no visible change to the player. This takes a lot of trial and error to achieve (because I can't work out the mathematical or structural relationship between the 35c value and the z coordinates and it changes as you increase height).