Units & Modules
A Wargame Modding Tutorial

How to modify (almost) whatever unit you want in Wargame Airland Battle.

Version: Almost slightly proper
Todo : finish => review => spelling stuffs

Prerequisites

- pen & paper (except if you can do without…)
- basic understanding of Enohka's tools (tables, classes, instances, using the dictionaries, installing a mod, etc..)
- Golden rule : never work on an original file, always use copies. Better have something to make a fresh start than screwing up for all eternity.
- FOR CHRIST SAKE, DON'T TRY TO USE A MOD IN MULTIPLAYER. YOU MIGHT GET SPANKED HARD ! Seriously we don't know of the implication yet, so don't, just don't.

Some stuffs are written in the dreadful language that is French, don't worry it won't hurt you and I can even provide a translation, usually between parentheses.

Color coding (soon) :
Sure.
Need more investigation but pretty sure.

Just an uneducated guess.

1) Understanding the unit structure

First thing you have to do is to open dev\ndf\patchable\gfx\everything.ndfbin, everything (heh...) will happen there.

The units are instances of the TUniteAuSolDescriptor (i.e GroundUnitsDescriptor in French) class (id = 86) but despite its name, the class contains all units in the game, helicopter and planes included.

The More You Know : the TUniteAuSolDescriptor can be considered as a specialization of the TUnitDescriptor which contains every other kind of objects you can see in the game like missiles, companies or smoke.

An instance of this class is constitued of many fields which describe a unit's behavior and stats:

Properties

BinId	Id	Name	Type	Biniary Value	Value
DB 01 00 00	475	DescriptorId	Guid	00000000000000000000000	00000000-0000-0000-0000-0000e8030000
DC 01 00 00	476	Modules	List	D82400005C000000	Collection[23]
DD 01 00 00	477	_ShortDatabaseName	TableString	820B0000	Descriptor_Unit_152mm_SpGH_Dana
DE 01 00 00	478	ClassNameForDebug	TableString	830B0000	Unit_152mm_SpGH_Dana
DF 01 00 00	479	StickToGround	Boolean	01	True
E0 01 00 00	480	ArmorDescriptorFront	ObjectReference	D92400005D000000	93 : 9433 - TArmorDescriptor
E1 01 00 00	481	ArmorDescriptorSides	ObjectReference	DA2400005D000000	93 : 9434 - TArmorDescriptor
E2 01 00 00	482	ArmorDescriptorRear	ObjectReference	DA2400005D000000	93 : 9434 - TArmorDescriptor
E3 01 00 00	483	ArmorDescriptorTop	ObjectReference	DA2400005D000000	93 : 9434 - TArmorDescriptor
E4 01 00 00	484	ManageUnitOrientation	Boolean	01	True
E5 01 00 00	485	HitRollSizeModifier	Float32	CDCC4C3D	0,05
E6 01 00 00	486	DeathExplosionAmmo	ObjectReference	DB2400005E000000	94 : 9435 - TAmmunition
E7 01 00 00	487	IconeType	Int32	02000000	2
E8 01 00 00	488	PositionInMenu	Int32	04000000	4
E9 01 00 00	489	NameInMenuToken	LocalisationHash	8E14D59D2C0F8600	8E14D59D2C0F8600
EA 01 00 00	490	Category	Int32	03000000	3
EB 01 00 00	491	AcknowUnitType	Int32	20000000	32
EC 01 00 00	492	TypeForAcknow	Int32	77000000	119
ED 01 00 00	493	Nationalite	Int32	01000000	1
EE 01 00 00	494	MotherCountry	TableString	37010000	TCH
EF 01 00 00	495	ProductionYear	UInt32	B9070000	1977
F0 01 00 00	496	MaxPacks	UInt32	03000000	3
F1 01 00 00	497	UpgradeRequire	ObjectReference	0E22000056000000	86 : 8718 - TUniteAuSolDescriptor
F2 01 00 00	498	Factory	Int32	0C000000	12
F3 01 00 00	499	ProductionPrice	List	0F000000	Collection[5]
F4 01 00 00	500	MaxDeployableAmount	List	00000000	Collection[5]
F5 01 00 00	501	ShowInMenu	List	01	Collection[5]

Some fields of the Cz Dana SPG descriptor instance.

Let's go through all of them :

- DescriptorID : the instance's Id, nothing much to say and you probably won't have to touch this unless you want to remove/add units

- Modules : a list of different special instances of classes called "modules", modules

control most of a unit's behavior. They are very important for modding a units specific part. We 'll see later how to work with them.

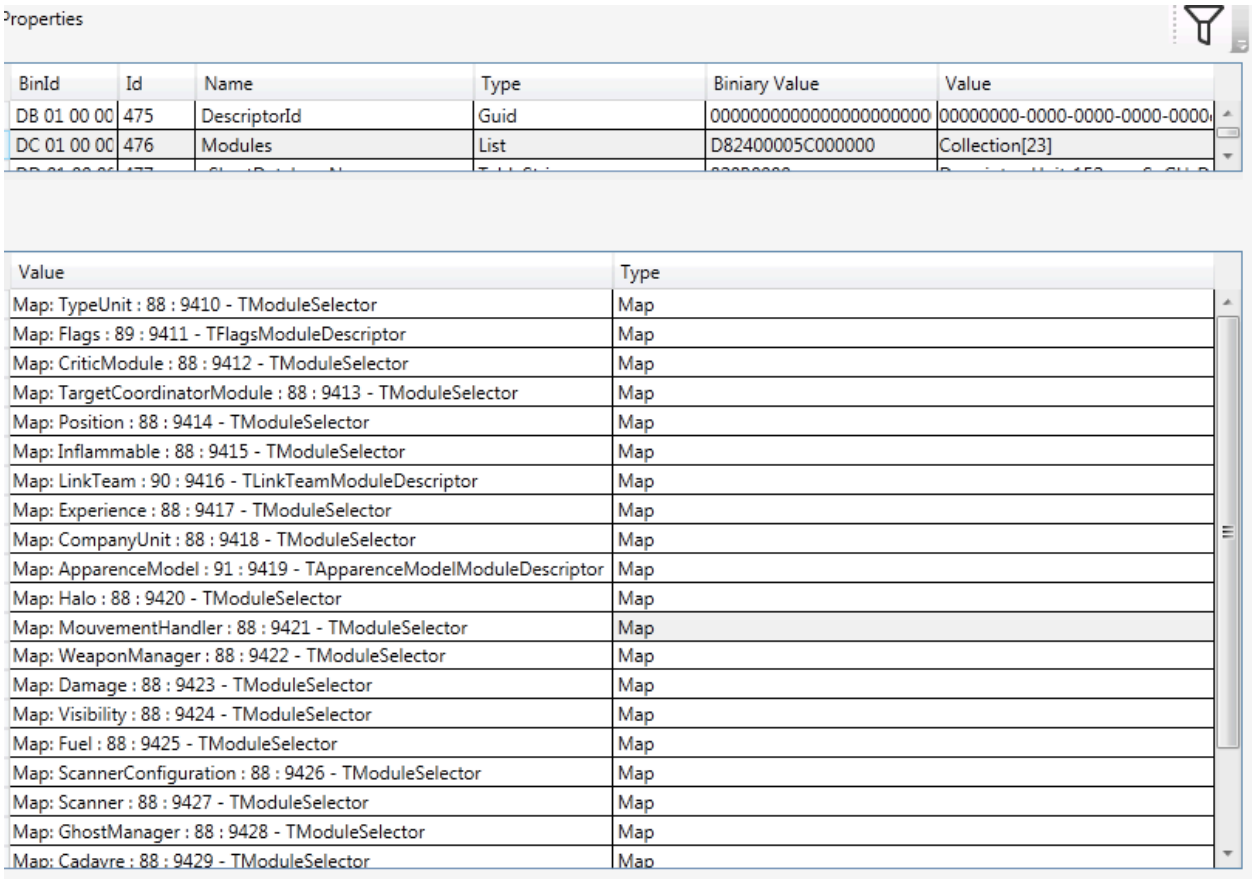
- _ShortDataBaseName : probably useless ? No idea, really.
- ClassNameForDebug : useless for modding.
- StickToGround : boolean determining… well if a unit stick to the ground I guess ? This field may have no use at all but seems to separate Helicopters from other kind of units.
- The ArmorDescriptor(s) : reference to a TArmorDecriptor instance which determine the armor value of the selected side. Armor values goes from 0 (null) to 23 (yes, 23) but are still displayed going from 0 to 20 anyway.
- ManageUnitOrientation : true = unit will always show they most armored side to the greatest threat, false = unit will show whatever side their pathfinding made them to.
- HitRollSizeModifier : float determining the accuracy malus/bonus due to the unit's size can be 0.05 (% ?) or -0.15.
- DeathExplosionAmmo : refer to a TAmmunition instance, is probably used to determine the explosion animation based on the Ammunition instance supplied. No idea yet about how it works exactly.
- IconeType (IconType) : integer pointing to a unit type list probably.
- PositionInMenu : probably the unit's position in the Armory, dunno if it's in a global list of just whithin a category (TANK, INF, ect) menu. Might be just a remanent of WEE.
- NameInMenuToken : a localisation hash (hence a reference to the hash table of the unit dictionary), it probably determine the unit's name diplayed in the uit's icon in the armory and production menu.
- Category : point to a category list (INF, REC, etc)
- AcknowUnitType : probably determine a unit's audio acknow when given an order.
- TypeForAcknow : honestly no idea…
- Nationalite (Nationality) : determine a unit's faction, null = NATO, 1 = Pact.
- MotherCountry :The actual country of the unit. Thanks to Fleff for this one.

- ProductionYear: pretty explicit.
- MaxPack : number of card of said unit you can add in a deck.
- UppgradeRequire : not formal use in WAB beside putting some vehicle in the same line in the Armory.
- Factory : change a unit's category in the armory and production menu (in-game). Thanks to homerfcb.
- ProductionPrice: a unit's cost(s), actually only the top cost of the list actually matter, the rest are probably remanent of WEE or some old design, just dont care about the rest.
- MaxDeployableAmmount : a list of a unit's avaibilities, each of the value refer to a particular veterancy hence the first value is the avaibility at recruit vet'.
- ShowInMenu : list of boolean determining if a unit can be viewable in the Armory.
- ProductionTime: time need for the unit to appear at a reinforcement point (in seconds ?). Is the same for all units in WAB vanilla.
- CoutEtoile (StarCost) : relic of WEE and WAB early beta, useless.
- TextureForInterface: refer to a TUITextureRessource, is basically the unit's image in the production menu, the small unit "cards" you see in menu.
- TextureMotherCountryForInterface : The country flag used in the small unit "cards" you see in menu.
- UnitTypeToken : list of localisationhash that refer to a unit's possible deck types (motorized, etc…). Be sure to look for the hash at a interface_outgame.dic. Thanks to AJE.
- UnitMovingType : determine a unit's locomotion type (tracked, wheeled…). Might be useless since a proper module already take care of this.
- VitesseCombat (CombatSpeed) : determine a unit's locomotion speed ? Might be useless since a proper module already take care of this.
- IsPrototype : pretty explicit
- TextureTransportForInterface : probably refer to the image used to represent infantry in a transport.

- Key : absolutely no idea.
- HitRollECMModifier : Malus/bonus to hit roll due to ECM.

a) A unit's Modules.

All and all, all the fields previously decribed only show a parcel of what is possible to be edited. To modify specific stats, we need to take a look at the Modules list :



Properties

BinId	Id	Name	Type	Biniary Value	Value
DB 01 00 00	475	DescriptorId	Guid	00000000000000000000000000000000	00000000-0000-0000-0000-000000000000
DC 01 00 00	476	Modules	List	D82400005C000000	Collection[23]

Value	Type
Map: TypeUnit : 88 : 9410 - TModuleSelector	Map
Map: Flags : 89 : 9411 - TFlagsModuleDescriptor	Map
Map: CriticModule : 88 : 9412 - TModuleSelector	Map
Map: TargetCoordinatorModule : 88 : 9413 - TModuleSelector	Map
Map: Position : 88 : 9414 - TModuleSelector	Map
Map: Inflammable : 88 : 9415 - TModuleSelector	Map
Map: LinkTeam : 90 : 9416 - TLinkTeamModuleDescriptor	Map
Map: Experience : 88 : 9417 - TModuleSelector	Map
Map: CompanyUnit : 88 : 9418 - TModuleSelector	Map
Map: ApparenceModel : 91 : 9419 - TApparenceModelModuleDescriptor	Map
Map: Halo : 88 : 9420 - TModuleSelector	Map
Map: MouvementHandler : 88 : 9421 - TModuleSelector	Map
Map: WeaponManager : 88 : 9422 - TModuleSelector	Map
Map: Damage : 88 : 9423 - TModuleSelector	Map
Map: Visibility : 88 : 9424 - TModuleSelector	Map
Map: Fuel : 88 : 9425 - TModuleSelector	Map
Map: ScannerConfiguration : 88 : 9426 - TModuleSelector	Map
Map: Scanner : 88 : 9427 - TModuleSelector	Map
Map: GhostManager : 88 : 9428 - TModuleSelector	Map
Map: Cadavre : 88 : 9429 - TModuleSelector	Map

The module list of the Dana.

Modules concern pretty much every aspects of a unit: its movement, its weapons, its fuel consumption, its (received) damages , etc
Most of these module go first throught a (maybe) facade class (i.e ModuleSelector id = 88) only a few of them don't and directly access their module instance (AppearanceModel, LinkTeam, etc…).

The More You Know : Apparently, swapping a ModuleSelector with another (even if both point to the same kind of module like a WeaponManager) makes the game to crash when calling the

modified unit.

List of Modules:
Modules whose content are left blank need more investigations about their nature.

- TypeUnit
- Flags
- CriticModule : manager determining whatkind of critic can a unit receive.
- TargetCoordinatorModule : constant among all unit instances.
- Position
- Inflammable : constant among all unit instances.
- LinkTeam : constant among all unit instances.
- CompanyUnit
- Experience: constant among all unit instances except for supply units.
- AppearanceModel : manage how a unit is rendere in-game, include : sounds, special effects, models (meshes).
- Halo : probably manage the selection interface (the “halo” around a unit).
- MouvementHandler : control the way a unit moves, its speed and agility.
- WeaponManager : control the behavior and stats of a unit’s weapons.
- Dammage
- Visibility
- Fuel
- ScannerConfiguration
- Scanner

- GhostManager
- Cadavre (corpse)
- MissileCarriage
- AttacheAeroport
- IAStratModule :
- StatEngine
- Debug : nothing for us to touch.
- Transportable : (for infantries only)
- Capturable : (for supply units only)
- Supply : (for supply units only)
- Transporter : (for transport vehicles and helicopters only)

The ModuleSelector :

So you want to play around with a module, let's say : the weapon manager, then you go to the instance the module list is pointing you to, something like 88:2158.
And you find this :

BinId	Id	Name	Type	Biniary Value	Value
1C 02 00 00	540	ControllerName	TableString	77120000	WeaponManagerContro
1D 02 00 00	541	Selection	List	1040000064000000	Collection[1]
1E 02 00 00	542	Default	ObjectReference	2D4000006D000000	109 : 16429 - TWeaponI
26 02 00 00	550	InitStage	Unset		null

You find there a buch of fields :

- ControllerName : well, it's basically the kind of module you are looking for.
- Selection: is a list of references (but usualy only one and is pretty much alway the same) to ModuleFilter. Now I have no idea what it exactly filters but I suspect it is the culprit of crashes when swapping ModuleSelector. It's a mysterious entity but thanksfully you don't need it to mod (for now).

- Default : is the module we are looking for, basically when you are working on a ModuleSelector, that's the only valuable field for you, you can discard the rest as "Eugen magic". For instance, for a WeaponController, this field point to our unit's TWeaponManagerModuleDescriptor, just go there.

- InitStage : USELESS EXCEPT FOR IA STUFFS, DON'T BOTHER (yet).

Now you finally manage to get the module descriptor you want, good job !
Next ? Well, it depends on what module you 've selected each work in a specific way, so far I've only done Mouvement (movement), Weapons and Appearance modules. The other might come soon in this tutorial but let's begin with what I bet you want.

b) A module example : the dreadful WeaponManager hydra.

Okay so, are you ready ?

Now that you found a WeaponManager you'd like to modify, let's get down to business :

BinId	Id	Name	Type	Biniary Value	Value
7E 02 00 00	638	ControllerName	TableString	77120000	WeaponManagerController
80 02 00 00	640	Salves	List	FFFFFFFF	Collection[8]
7F 02 00 00	639	TurretDescriptorList	List	595700008F000000	Collection[1]

A WeaponManagerModuleDescriptor, no idea whose unit is it from.

We've got 3 fields:
- ControllerName : pretty explicit huh ?
- Salves (salvos) : it's a list of integer that (may) determine the number of salvos avaible of a turret (I get about this in a line). Most of the time Salvos = ammo since except for units that shot many ammo in a single salvo (like autocanons).
- TurretsDescriptorList : contains the list of turretDescriptor a unit have.

The most important part is of course the Turret list, in my example ther is only one turret but it have bazillion of turret (or more exactly as much as the game can handle); let's see what it is !

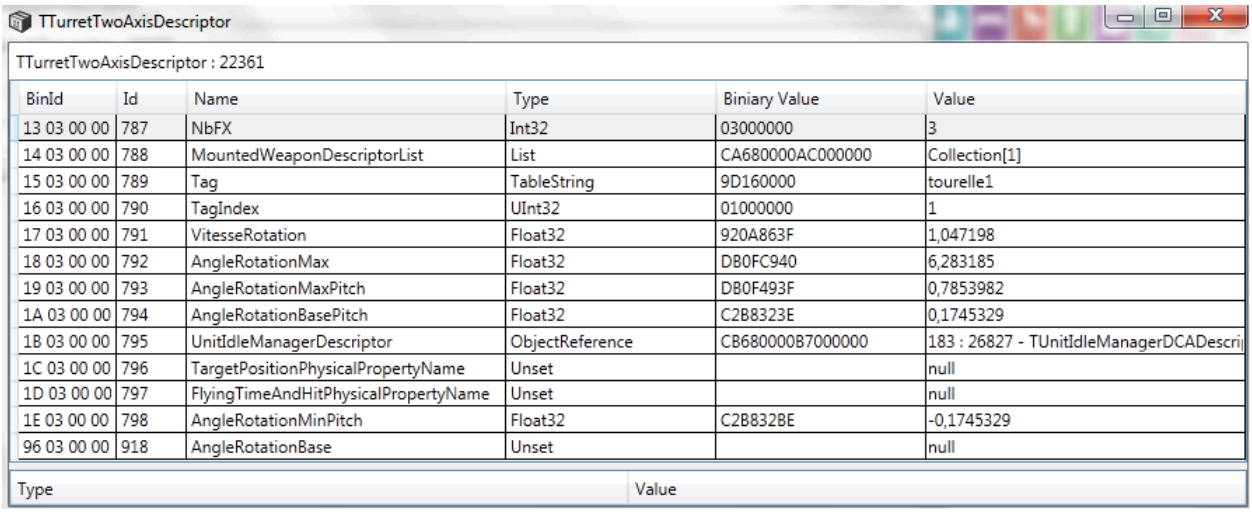
1) A world of Turrets.

So I was cool but I didn't said what is a turret yet. I guess you kinda know already. In real life, a turret is the stuff that move a tank canon..
In Wargame, it is worse than that. EVERY weapons is mounted on a turret.
And that include, rockets, infantry rifles, ATGM and even bombs. In the game, a turret is a container for weapons and determines how a unit behave before shooting : for instance, a tank will turn its turret toward an ennemy using a rotation speed and a plane will put himself at a certain altitude before bombing. These behaviors are described in these TurretsDescriptor.

How does it do this ?

Well, there are many kind of turret descriptors some are pretty generic : TTurretTwoAxisDescriptor and TTurretUnitDescriptor and some are more specialized descriptor : TTurretInfanterieDescriptor (for Infantry units) and TTurretBombardierDescriptor (for bombing weaponery).
The TTurretTwoAxisDescriptor is the most used kind of turret, it designs a turret that can rotate over 2 axis like… any tank turret.

For know let's return to our example:


TTurretTwoAxisDescriptor

TTurretTwoAxisDescriptor : 22361

BinId	Id	Name	Type	Biniary Value	Value
13 03 00 00	787	NbFX	Int32	03000000	3
14 03 00 00	788	MountedWeaponDescriptorList	List	CA680000AC000000	Collection[1]
15 03 00 00	789	Tag	TableString	9D160000	tourelle1
16 03 00 00	790	TagIndex	UInt32	01000000	1
17 03 00 00	791	VitesseRotation	Float32	920A863F	1,047198
18 03 00 00	792	AngleRotationMax	Float32	DB0FC940	6,283185
19 03 00 00	793	AngleRotationMaxPitch	Float32	DB0F493F	0,7853982
1A 03 00 00	794	AngleRotationBasePitch	Float32	C2B8323E	0,1745329
1B 03 00 00	795	UnitIdleManagerDescriptor	ObjectReference	CB680000B7000000	183 : 26827 - TUnitIdleManagerDCADescri
1C 03 00 00	796	TargetPositionPhysicalPropertyName	Unset		null
1D 03 00 00	797	FlyingTimeAndHitPhysicalPropertyName	Unset		null
1E 03 00 00	798	AngleRotationMinPitch	Float32	C2B832BE	-0,1745329
96 03 00 00	918	AngleRotationBase	Unset		null

Type | Value

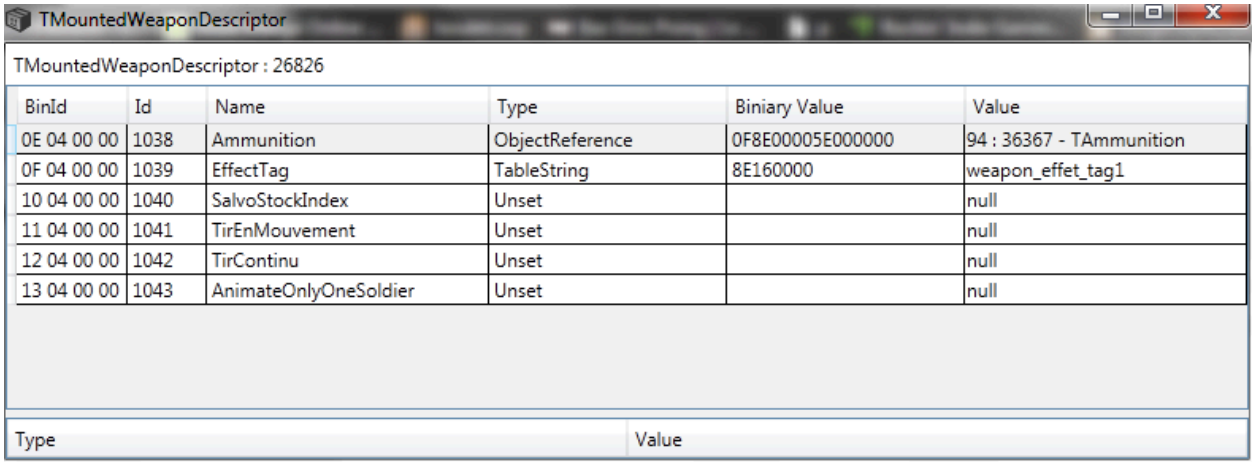
This our turret descriptor, it has quite a few fieds that determine the turret's "physic":

- VitesseRotation (RotationSpeed)
- Various angles
- A tag and tag Index for the turret
- A idleManagerDescriptor to determine the idle animation (by the name of it, we can deduce our weapon is a AA weapon since DCA = AA in French).
- And most importantly: a list that contains all the weapons monted on that turret.

A turret can have a lot of weapons, it is usually done for “co-linear” weapons like ATGM tanks, autocanon and low velocity canon on the BMP3, etc
Here the turret only contains a single weapons, it refers to a TMountedWeaponsDescriptor instance, let’s go there.

2) The Weapon Descriptor

Her is what you can find in a TMountedWeaponsDescriptor:


TMountedWeaponDescriptor

TMountedWeaponDescriptor : 26826

BinId	Id	Name	Type	Biniary Value	Value
0E 04 00 00	1038	Ammunition	ObjectReference	0F8E00005E000000	94 : 36367 - TAmmunition
0F 04 00 00	1039	EffectTag	TableString	8E160000	weapon_effet_tag1
10 04 00 00	1040	SalvoStockIndex	Unset		null
11 04 00 00	1041	TirEnMouvement	Unset		null
12 04 00 00	1042	TirContinu	Unset		null
13 04 00 00	1043	AnimateOnlyOneSoldier	Unset		null

Type | Value

It contains fields that determine the weapon’s firing procedures : TirEnMouvement (FireWhileMoving), TirContinu (SustainedFire), some animation for the infantry for LMGs and rockets lunchers.
But the most important part is the Ammunition fields, it refers to a TAmmunition instance and (despite its name) this is where you ‘ll be able to modify whatever you want about the weapon.

3) Ammunition rules the nation.

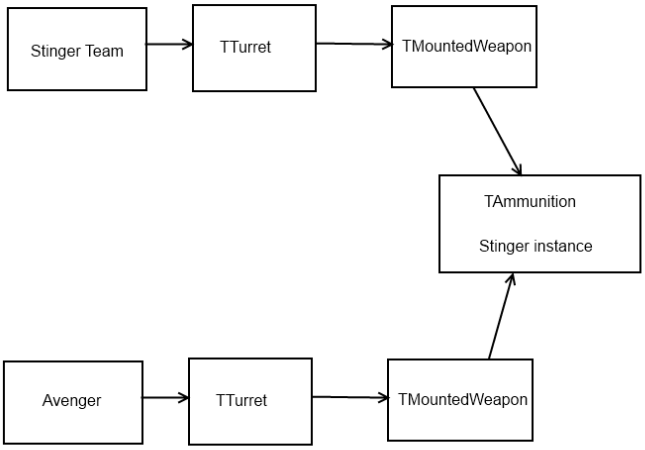
Okay, there wasn’t much to do gameplay-wise (but that was still interesting, right ? right ?) but now we ‘ll get into real business. The TAmmunition class is the container of most of the stuff that have teared apart the official forum and drained much whiney’s tears.

So yeah, it basically looks like this:

Properties

BinId	Id	Name	Type	Biniary Value	Value
2B 02 00 00	555	DescriptorId	Guid	00000000000000000060000	00000000-0000-0000-0600-0000
2C 02 00 00	556	Name	LocalisationHash	9CF6380000000000	9CF6380000000000
2D 02 00 00	557	TypeName	LocalisationHash	9CF6380000000000	9CF6380000000000
2E 02 00 00	558	TypeArme	LocalisationHash	9CF6380000000000	9CF6380000000000
2F 02 00 00	559	Arme	UInt32	04000000	4
30 02 00 00	560	ProjectileType	UInt32	03000000	3
31 02 00 00	561	Puissance	Float32	0000FA43	500
32 02 00 00	562	TempsEntreDeuxTirs	Float32	0000003F	0,5
33 02 00 00	563	PorteeMaximale	Float32	00204B48	208000
34 02 00 00	564	AngleDispersion	Float32	CDCCCC3D	0,1
35 02 00 00	565	RadiusSplashSuppressDamages	Float32	00204B47	52000
36 02 00 00	566	SuppressDamages	Float32	00003443	180
37 02 00 00	567	RayonPinned	Float32	0080A246	20800
38 02 00 00	568	TirIndirect	Boolean	01	True
39 02 00 00	569	TirReflexe	Boolean	01	True
3A 02 00 00	570	FX_tir_sans_physic	Boolean	01	True
3B 02 00 00	571	FX_vitesse_de_depart	Float32	00401C46	10000
3C 02 00 00	572	FX_frottement	Float32	6F12833A	0,001
3D 02 00 00	573	FX_tir_tendu	Boolean	01	True
3E 02 00 00	574	TempsEntreDeuxSalves	Float32	0000A040	5
3F 02 00 00	575	NbrProjectilesSimultanes	UInt32	01000000	1
40 02 00 00	576	NbTirParSalves	UInt32	01000000	1
41 02 00 00	577	AffichageMunitionParSalve	UInt32	01000000	1
42 02 00 00	578	Level	Int32	05000000	5
43 02 00 00	579	HitRollRule	ObjectReference	2540000075000000	117 : 16421 - TWargameHitRollRu
44 02 00 00	580	FireDescriptor	ObjectReference	2640000076000000	118 : 16422 - TUniteDescriptor
45 02 00 00	581	FireTriggeringProbability	Float32	9A99193E	0,15

Lot of stuff to see, but we won't go throught all parameters, Darkmils's guide will give you more informations about them. But I will show you the more "correct way to change stuffs.
The more important thing to understand is that changing a TAmmunition instance may change the stat of more than a single unit. For instance, if you change stuffs on the Stinger Tammunition instance, you effectvely change stuffs on the Stinger team but also on the OH-58C or the Avenger.
This happen because those units refer to the same TAmmunition instance, every time you play around with something that is "refered", keep in mind that more than one object may be refering it when modifying the properties. This is the source of many errors.



Now let's try to modify something, let's say you want to modify the accuracy.
The accuracy is determined by the HitRoleRule parameter, it's a reference to a TWargameHitRollRule instance. Let's go there.

TWargameHitRollRule

TWargameHitRollRule : 46824

BinId	Id	Name	Type	Biniary Value	Value
99 02 00 00	665	MinimalHitProbability	Float32	CDCC4C3D	0,05
9A 02 00 00	666	MinimalCritProbability	Float32	0AD7233C	0,01
0B 05 00 00	1291	HitProbability	Float32	CDCC4C3E	0,2
0C 05 00 00	1292	HitProbabilityWhileMoving	Float32	CDCC4C3D	0,05

You can see many (like, 4…) parameters that define the hit probability and many other things such as stabilizer efficiency and critical probability…

Please note that the accuracy (just like many other parameter) isn't put in the same way as you can read it in-game. Here it's a pure proba value (therefore, somewhere between 0 and 1) while in-game it is a pure unsigned integer (between 0 and 14). The translation between those values is done automatically, don't bother about. If you are interested about it, look around the TWargameUniteDescriptor's unique intance.

The easiest way to modify it is simply to change the parameter… but remember what I previously said : you would then not only modify the accuracy of the weapon you want but also

all those that refer to that particular TWargameHitRollRule instance. It's clearly not what you want so what you have to do is:

1. Look around the TWargameHitRollRule instances for an instance that match what you want. (Or if it doesn't exist, add it when the mod tools will able you to do so).
2. Go back to the TAmmunition instance and change the TWargameHitRollRule.

This way you 'll only change this particular TAmmunition.

2) Modeling and Depiction

Interruptions interruptions….
Oh God too much people are watching me, I feel pressured. :s Don’t mind the typos. :<