SOLDIER OF FORTUNE II DOUBLE HELIX

NPC FILES

BY RAVEN SOFTWARE

1.0 Overview

All the characters available in SoF2Radiant are defined in external text files called Templates. These templates provide all the skins, critical statistics, and inventory configurations that make any particular character unique. Adding a new character type to SoF2 is as simple as adding the template to an existing file or creating a new file in the "base/npcs/" directory. These files have a .NPC extension. Both the game and Radiant scan through this directory when looking for available character types. Many of the features of the templates are solely for SP, some for MP, but both games share the same templates to avoid duplication.

2.0 Inheritance

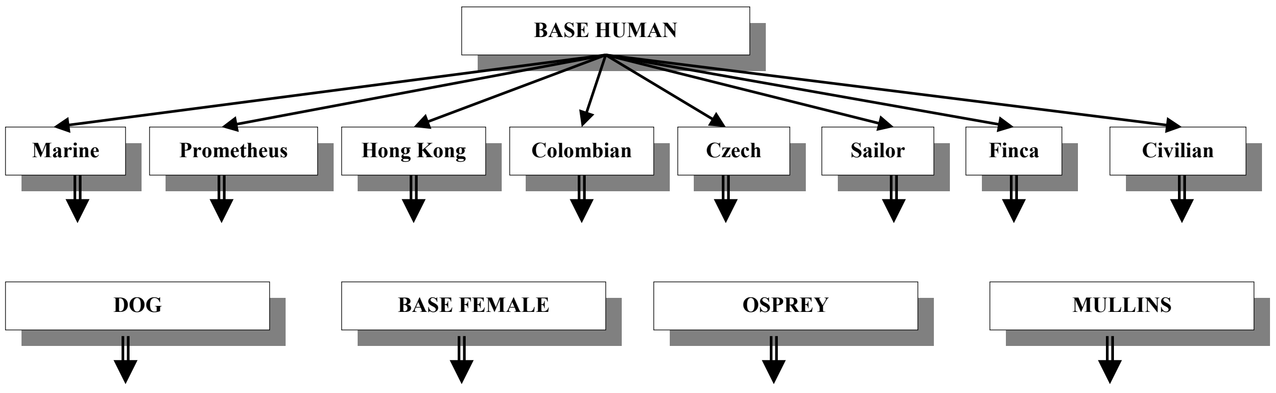
One important feature of the template system is inheritance. Every template has what is called a parent template, and will use the parent's values for various stats should none be specified for the child. This is a very common practice. For example, there are several basic character types, such as the "Colombian Rebel" for example, from which a number of specialized characters are derived. There could be a "Colombian Rebel Emplaced Gunner" who has a special skill for using emplaced guns, but otherwise has all the same stats as the other "Colombian Rebels". This was done for two reasons:

1) Ease of use in making new templates.
2) Ease to make changes in templates.

Say for example, we wanted all the Colombian Rebel characters in the game to be a bit easier, we could simply alter the accuracy or damage (see available stats) in the parent template and all the other Colombians would inherit the change.

Most of the parent templates currently reside in *base.NPC*   Nearly all the characters (male at least) are derived from Base Human.

Below is a partial inheritance chart of the SoF2 NPC Templates as of 4/30/02:



There are a few other exceptions to the inheritance tree, and some more unusual cases, like the Taylor character that is actually derived from the male Base Human so she can have the Chem Suit model. Inheritance for any given template is determined at the top of the file in which that template is defined. For example, at the top of Prometheus.npc is a Group Info which defines the skeleton and parent template for all the templates in that file:

```
GroupInfo
{
	Skeleton		"average_sleeves.skl"
	ParentTemplate	"NPC_Prometheus_Soldier"
}
```

3.0 Occupations And Ranks – SP only

The characters in SoF2 have unique abilities, tactics, and skills. A large portion of how a character behaves is based upon his occupation and rank. The mainstays of the game are soldiers with a rank of Private, and Thugs with a rank of Criminal; however there are a number of more exotic combinations. Below is a list of all the possible ranks and occupations available to the characters in SOF2:

Important note: If setting up a character solely for multiplayer, occupation and rank are unimportant. Add something standard, like Private Soldier.

Ranks

- **Civilian** – Mostly used with the Tourist type
- **Criminal** – Mostly used with the Thug type
- **Private** – By far the most common, pairs up into fire teams and squads, uses hand signals
- **Sergeant** – Runs a squad if scripted to do so, otherwise simply fights along with the team

Rank is a little less important than occupation. The real difference is that Privates organize themselves into fire team pairs, so it is important for privates to be spawned near each other, because they will try to work together, communicate, and give hand signals.

Occupations

- **Assassin**
- **Commando**
- **Demolitionist**
- **Emplaced Gunner**
- **Look Out**
- **Scout**
- **Script Guy**
- **Sniper**
- **Soldier**
- **Soldier Elite**
- **Soldier Cover**
- **Tourist**
- **Thug**

The Dog And Osprey are special case occupations.

Occupations are the largest determining factor in behavior. The basic character type is the Soldier, and the other types usually provide some variant on the soldier's behavior.

3.1 Occupation descriptions

Soldiers patrol a large radius, say 1000 some units. When idle, they will pull out a cigarette if they have one. They examine suspicious problems like footstep sounds, prints, bloodstains, and dead bodies. They will run toward sounds of combat and stop to shoot as soon as they can see an enemy target. If cover is near by, they will run to cover when they need to reload or when a grenade is in the air. They will use emplaced weapons if close enough (200 units), will jump and vault over obstacles, climb up and down ladders, and will lean around corners with lean points. They pick up weapons from the ground, when their ammo runs out or if their gun is shot from their hands. They run away if they have no weapon and can't find another to pick up. They throw grenades if they have any and they can't see their target but can throw a grenade near by. They pick up grenades thrown nearby and throw back if possible.

Assassins are actually quite different from soldiers. They are much more cautious about revealing themselves in the open and will back away if you are aiming at them. When they can, they will roll grenades in your direction. I don't think many have been used in maps. They work well in areas where nav points don't connect the grid, because they do not use A* for path finding.

Commandos work in pairs, advancing slowly from one cover point to the next. They use hand signals and report to each other when they have reached their next cover point. While one advances, the other will provide cover fire. Like assassins, they are slower and more cautious than normal soldiers. They only work well in areas heavy with cover and nav points.

Demolitionists are exactly the same as normal soldiers except they are much more likely to throw grenades, and – like assassins – they can roll grenades. Demolitionists and Assassins are the only two character types who can roll grenades. They are very good at it, so don't put too many of either type in an area. Most demolitionist guys I've seen are also heavily armored and equipped with powerful weapons, although that is not necessary.

Emplaced Gunners don't patrol. They are usually placed very close to an emplaced gun, with the idea that they will use the gun when the opportunity is presented. They also usually have only a pistol or grenade and no "primary" weapon at the ready. They will still do all the other normal combat behavior.

Look Outs walk a much smaller patrol radius than normal soldiers (100 units), and do not attempt to get to an enemy target that they are aware of but cannot see. They work well in towers, on walls, or in areas you want guys to stay put.

Scouts are the same as soldiers, except they will attempt to pair up with other scouts, and will go first in a squad of soldiers who are scripted to move through a level. They usually have radios or communications equipment but don't really use it. They are also usually lightly armored.

Script Guys are only used for cinematic sequences and will wait for commands from ICARUS before doing anything.

Snipers are usually hyper accurate and armed with a sniper rifle. They pause between shots and do not attempt to chase down a target, patrol, explore strange sounds, or throw grenades. They stand where they were spawned in and do not move from their spot, waiting for targets to come within their sight.

Soldier Elite characters are almost exactly the same as soldiers, except they are a little more aggressive in selecting cover and can run a little faster. Soldiers and Soldier Elite characters are the only type who can use lean points.

Soldier Cover, as with Soldier Elite, is almost exactly the same as soldier. The only difference is their initial reaction to seeing an enemy target is to first run for cover.

Tourist guys are usually civilians. By default they do not have any weapons, and are usually not on any team. If they are shot however, they will pick up any near by weapons and fight against the team that shot them. By default, they "Wander Area" around where they start and like to look at use entities (a whole other issue).

Thugs have many of the same behaviors as soldiers, except they do not stop and stand still to fire, they will continue to run as close to an enemy target as they can get. Usually these characters are armed with close range weapons like shotguns and have very little health. They often end up being fodder, but can be a little scary when they are running toward you.

Dogs have a completely different behavior set from humans. They essentially run toward a target and bite if they can. When idle, they will sleep and pee.

The Osprey is a deadly boss character for a couple levels in the Single Player game and the hardest difficulty in the RMG. It does not use the nav point grid at all, but does a combination of flocking / seek / avoid behavior. It will hover and strafe around a target it is shooting at, will spawn troops to rappel out of it when it can, and once it is wounded it will use a combination of high speed strafe attacks and missiles to kill it's target.

4.0 Vital Statistics

In addition to specifying rank and occupation, character templates also define all the character's vital statistics. Below is a list of these stats and what they do:

Field	Type	Default	Description
Name	String	“”	The name of the character, this is what shows up in Radiant as the class name.
Model File	String	“”	This determines which model file the character uses.
Health	Range	-1 -1	The first number is the minimum health the character will spawn in with, and the second number is the maximum health he will have.
Team	Enum	“”	This is which team the character will associate with. Common Teams are “The Shop”, “Prometheus”, etc…
Burst	Bool	false	If set to true, this character will always use 3 round burst mode when shooting an automatic weapon. It is commonly found on characters with the auto shotgun (USAS-12) and the micro uzi.
Bravery	Float	1.0	Currently only used to determine if the osprey will fly away after having both of it's engines destroyed or if it will stay for the full fight.
Agility	Float	1.0	NOT USED AT ALL
Accuracy	Float	-1.0	This number sets what percentage of shots will hit. If the character's accuracy was 0.50, about half his shots would hit a target at 1000 units away. A perfect 1.0 accuracy will always hit a target at any distance.
DamageScale	Float	-1.0	This is the number that is multiplied against what the player's damage would be with the same weapon. For example, if the player would normally do 60 points of damage with the M4 and this npc had a damage scale of 0.5, he would do 30 points of damage.
Boss	Int	0	If set to 1 or 2, the character will exhibit boss abilities (ignore head shots, less pain animations, things like that)
DefaultReady	String	“*hand_r”	This is the location on the model where primary weapons will be held.

Values that default to an empty string “” or a number like –1 will default to the parent template for their value should none be provided.

Together Damage Scale and Accuracy pretty much determine how effective and dangerous an NPC is.

4.1 Bounds

Occasionally it is necessary to resize the bounding box of a character for different occasions. Each bounding box specifies two vectors Min and Max:

<u>From “Base Human Template”</u>

```
	Bounds
	{
		Pain
		{
			Min	"-30 -30 -45"
			Max	"30 30 45"
		}
```

Bounds are a very common inherited feature. They are usually defined in the lowest character template (Base Human for most cases), and never changed in any of the derived templates.

4.2 Voice – SP only

The voice portion of the template is where the character will go to find which sound files to play for any given response. The base template has a number of basic sounds for pain type events (Special Group). Other groups include: Focus, Orders, Reports, and Areas. The format for each line in the voice section usually has the type with an underscore and a number behind it. Adding a new sound to the template is as simple as writing another line:

For example if the character template had 11 existing sounds for the Target Acquired report and the last line looked like this:

```
TargetAcquired_11		"sound/enemy/russian/soldier/there_he_is"
```

Simply adding a new line below it with the new sound would guarantee that the character would occasionally say that sound file:

```
TargetAcquired_12		"sound/enemy/russian/soldier/XXXXX"
```

Some (many) of these sounds also have a “Token” line above them that refer to a text token which should be printed at the top of the screen when the character shouts his line. Often these tokens are not used in the game.

<u>Here are the reports available and when they are called:</u>

TargetAcquired	When a character sees an enemy
TargetEliminated	When an enemy has been killed
TargetLost	When the current focus of a character goes out of sight
ManDown	When a character sees a friend who has been killed
ObjectiveComplete	When a character has finished the current phase of his goal
PositionReached	When commandos signal each other to advance
UnderFire	When shot
FireInTheHole	When throwing a grenade

SoundAlarm	When a dead body is discovered or an ambush is detected
ExamineGroup	When more than one character goes to examine a sound or footprint
ExamineAlone	When only one character goes to examine
ExamineStop	When all suspected problems were looked at, and ready to patrol again
Blocked	When a character gets frustrated with trying to get around someone else

4.3 Skins

Skin selection in SOF2 is pretty straightforward. Each character can have any number of skins and will randomly select one at the time he is spawned. Should a template have no skins at all, it will look to the parent template and inherit any from there. For information on creating skins, see the document SoF2_Character_Skins. Each skin group looks like this one from base.NPC:

```
Skin
{
        File        "hk_ped_a1"
}
```

The File must refer to a g2skin file somewhere in the models/characters/skins directory. Sometimes it is necessary for certain skins to never play facial animations that cause the character's mouth to open (like skins with face masks painted on them). For such skins, right next to the file line add this line:

```
NoFaceAnims "true"
```

Each skin can have it's own specific inventory group as well. For example, some skin might need a special hood surface turned on or particular bolt on item added.

4.4 Inventory

The character templates also provide a mechanism for giving the characters inventory in terms of items and weapons. There are two places, global inventory that will be given to all characters with that template, and skin specific inventory given to any guy who spawns with that skin. Global inventory is also inherited so global inventory in a parent template will be given to all guys in all derived groups no matter what skin they select.

4.4.1 Weapons – SP only

Weapons are usually the same as player weapons (with a couple unusual exceptions) and most are simply added. Some weapons are "secondary" in that they can reside in a holster spot until the character draws them (pistols and grenades can be holstered). For these weapons, a bolt point must also be provided for the holster model. For more on weapons, see section 5.0 Weapons.

```
Weapon
{
        Name            "US SOCOM"
        Bolt            "*hip_r"
```

```
		}
```

4.4.2 Items

Items can be much more complex. They range from surfaces that can be turned on and off (a hood or backpack for example) or actual models that may or may not be turned on. Models need to be given a bolt point where they should be attached to the model.

```
		Item
		{
			Name		"ammo_pack_green"
			Bolt		"*hip_fl"
		}
```

Both weapons and models can also have a “Chance” field. This is a number between 1 and 100 which determines what percent chance there is of that item / weapon being given go the character when he spawns.

Here is an example of adding inventory to the Marine Soldier template:

```
CharacterTemplate
{
	Name		"NPC_Marine_Soldier"
	Team		"The Shop"
	Rank		"Private"
	Occupation	"Soldier"
	Accuracy	"0.7"

	Inventory
	{
		Weapon
		{
			Name		"US SOCOM"
			Bolt		"*hip_r"
		}
		Item
		{
			Name		"ammo_pack_green"
			Bolt		"*hip_fl"
		}
	}
	Skin
	{
		File		"marine_camo2"
		Inventory
		{
			Weapon
			{
				Name		"M4"
			}
			Item
			{
				Name		"backpack"
				mp_onback	YES
			}
			Item
			{
				Name		"helmet_military"
				Bolt		"*head_t"
			}
```

```
                    Item
                    {
                              Name                "ammo_pack_green"
                              Bolt                "*uchest_l"
                    }
                    Item
                    {
                              Name                "canteen"
                              Bolt                "*hip_br"
                    }
          }
}
```

You may notice that there are two Inventory areas in the above example, the first one is by itself, and the second is inside the camo2 skin group. These are what are called global inventory and skin inventory. Global inventory is inherited to all characters and all skins, but skin inventory is only applied if the character selects the skin specified.

In the above example, all characters derived from the “Marine_Soldier” template will get a US SOCOM pistol and an ammo pack at their hip, no matter what skin they select or how far down the derived tree they are. Should they get the “camo2” skin, they would also get an M4, backpack, helmet, another ammo pack, and a canteen.

Global inventory is cumulatively inherited. That means that a character derived from “Marine_Soldier” who ALSO defines his own global inventory set will just add to the items in a cumulative fashion and not replace the parent’s group. There is no redundancy checking either, so it is important to be careful with global inventory because it is added to all the characters and all the skins within the characters.

For example if a new “Marine_Soldier_Cover” was derived from “Marine_Soldier” with a template that looked like this:

```
CharacterTemplate
{
          Name                "NPC_Marine_Soldier_Cover"
          Occupation          "Soldier_Cover"
          Inventory
          {
                    Weapon
                    {
                              Name                "canteen"
                              Bolt                "*hip_br"
                    }
          }
}
```

He would have a global inventory that included the pistol, ammo pack AND the canteen. Then if this character had no skins he might select the “camo2” skin from his parent and end up with 2 canteen models bolted to the same point.

All the inventory items and weapons available to characters in the spawn templates can be found in the “ext_data/sof2.item” file.

4.5 Multiplayer specific fields

Field	Type	Default	Description
FormalName	String	“”	The name of the character as it will appear in the multiplayer menu.
Deathmatch	String	“”	This determines whether the NPC is valid for multiplayer. If not included, assumes yes.

In addition to these fields, you can also specify whether a specific item will show up in multiplayer or not. See the items section below for details.

5. 0 Weapons – SP only

All the weapons that can be carried by characters in SoF2 are defined in the “SOF2.item” file under the directory “ext_data”. Creating a new weapon or item in the game just requires an entry in the .item file and then inclusion in a template.

Weapons also need to be included in the SoF2.wpn file, which determines the other weapon properties. For information on the .wpn file, see the document SoF2_Weapons_Overview. Both Non Player Characters and the Player can use most weapons, but a number of special fields need to be added for NPCs to use them.

Field	Type	Default	Description
name	String	“”	This is the name of the weapon, should be the same as one of the weapons defined in SOF2.wpn
fxname	String	“”	What muzzle flash effect should this play (normally leave blank and let the SOF2.wpn file determine)
boltpoint	String	“”	Where the character holds the weapon (right hand, left hand…)
canholster	Bool	False	If true, then this weapon can be holstered somewhere on the character’s body and drawn out at a later time.
holstmodel	String	“”	If it is holster-able, this is the model for the holster.
holstsurf	String	“”	This is the surface to turn off in the holster model when the weapon is drawn out.
canjam	Bool	False	If true, then there is a chance that the gun will jam up after being shot and the character will have to drop the weapon.
canshootoff	Bool	False	If true, then this weapon can be shot out of the character’s hands.
clips	Int	-1	How much ammo he starts with (if –1 use the SOF2.wpn value). Really only used for grenades.

level	Int	1	How good this gun is. 1=pistol 2=rifle, 3=emplaced gun
atktype	String	“”	Melee, slug, projectile, or missile
muzzle	String	“”	Special thing for Osprey Rockets
muzzle2	String	“”	Special thing for Osprey Rockets
onsurf	String	“”	Any thing with a key of onsurf will attempt to turn on these surfaces when this weapon is given to the entity
offsurf	String	“”	Similar to on surf, but turns them off

6.0 Items

Character items in SoF2 range from simple boltons that enhance appearance to surfaces for Asian characters to cigarettes that can be drawn and used, to smoke grenades that can be thrown. Much like weapons, there are a bevy of fields, which can determine how an inventory item works:

Field	Type	Default	Description
name	String	“”	This is the name of the item.
model	String	“”	The item’s model
onsurf	String	“”	A surface this item will turn on
offsurf	String	“”	A surface this item will turn off
bolt	String	“”	The position this model is attached to if “drawn” out
health	Int	0	How much damage this item can take before it is destroyed
skin	Bool	false	NEVER TESTED – This should set a g2skin on the model, but has never been tried. For a long time if the same model needed a new skin, they exported a new model.
canshootoff	Bool	false	Will fall off the character when shot.
candamage	Bool	false	Will take damage before being shot off.
protects	Bool	false	If a trace his this item, it will stop there and not go through to the rest of the character.
candraw	Bool	false	Can be drawn out and used (cigarettes and smoke grenades are good examples)
canholster	Bool	false	Can be put away again (DOES NOT WORK)
canthrow	Bool	false	Can toss the item away
canuse	Bool	false	Can “use” the item (play use animation)
canexamine	Bool	false	Can “examine” the item (play examine animation)
useeffect	String	“”	If the use animation for this item has a call back, it will play this effect at that time.
usebone	String	“”	The bone where the use effect is played on.

usefrequency	Int	50	How often should he use it
usecount	Int	5	How many times should he use it before throwing / holstering it away.
lockstance	String	“”	The stance (Stand, Crouch, Sit) that this will force the guy into when drawing it out.
usetarget	String	“”	Targetname of an entity on the level that will get used when the player touches this item (used for Boss Sequence on HK7)
detonatetimer	Int	0	How long before the item will detonate after it is dropped / thrown. If zero, it never detonates. (Used for smoke grenade)
detonateeffect	String	“”	The effect it will play when the item detonates.
detonateloseffect	String	“”	Will only play this effect if the item can directly see the player when it detonates.
inaireffect	String	“”	The effect it will play while it is in the air after thrown / dropped
alwaysprecache	Bool	false	If true, all the assets for this item will be preached, even if no character is spawned with it in his inventory

To summarize, items can be drawn, holstered, used, examined and thrown. They can have various parameters for usage and detonation. They can have health and can protect their owner.

6.1 Multiplayer specific item fields

Field	Type	Default	Description
Deathmatch	String	“”	This determines whether an item can show up in multiplayer. If not added, it assumes yes. All hand-held items are given a ‘no’ field.