

This document will walk you through setting up a basic CNC-missioned map for CNC Far Cry. It will also guide you through setting up a basic building and adding advance features to it including building lights, damage effects, and interface animations.

Make sure you have the latest CNCFC Mod installed and ready to go. This tutorial was designed around the 20507 English build.

History:

- 21807 Build:
  - Added Security Zones
  - Added Gun Emplacements and Security Guns
  - Added Custom Map Buildings documentation to Mission script information
  - Updated Tutorial level and Document to 21807 Build

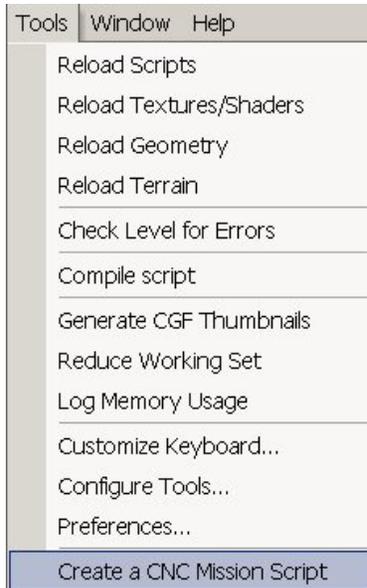
## Table of Contents

<b>1. Creating a CNC-missioned Map</b> .....	3
<b>2. Interfacing with a Building</b> .....	6
<b>3. Advance Building Features</b> .....	9
<b>Critical Points</b> .....	9
<b>Building Lights</b> .....	10
<b>Building Damage Effects</b> .....	12
<b>Security Alerts</b> .....	14
<b>Interface Animation</b> .....	15
<b>4. Other Essential Entities</b> .....	16
<b>Player Spawn Points</b> .....	16
<b>Vehicle Spawn Points</b> .....	17
<b>Gun Emplacements and Security Guns</b> .....	18
<b>Purchase Terminal Zones</b> .....	20
<b>5. Mission Script Internals</b> .....	21
<b>OnInit and ArgueVictory</b> .....	21
<b>OnPurchaseMenuInit</b> .....	21
<b>SetBuildingAttr</b> .....	22
<b>Custom Buildings</b> .....	22

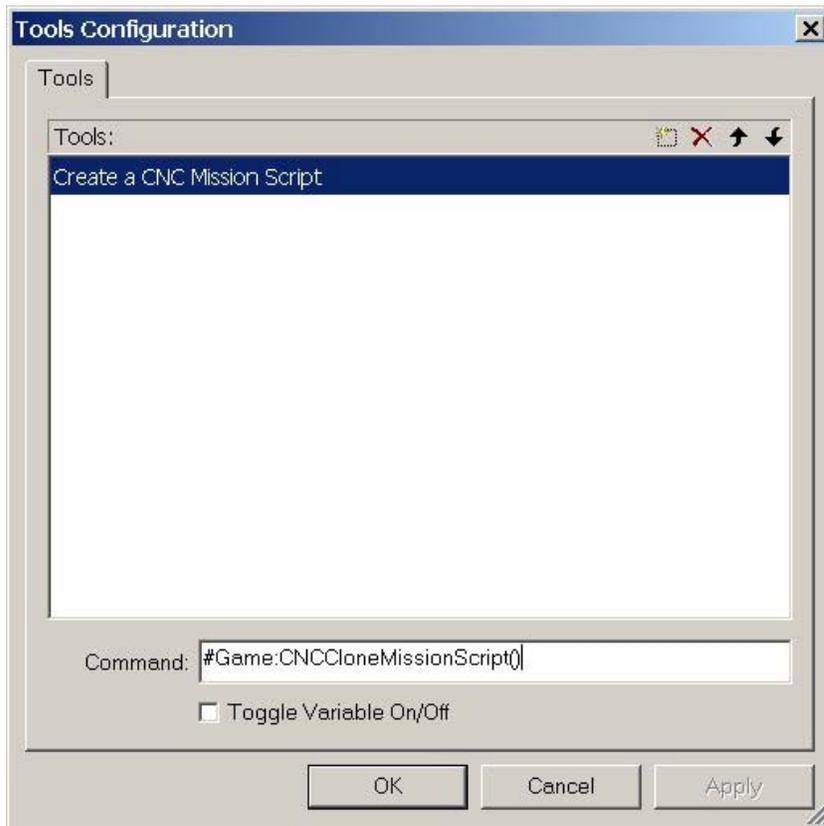
## 1. Creating a CNC-missioned Map

Open up the Sandbox editor. Ensure you are working with the CNCFC mod. Create a new map, naming it whatever you want, and set up your terrain.

The first thing we will need to do is create a basic CNC mission script. Luckily, there is a good template available for this, and using that template will be as simple as selecting an option from the menu. From the Tools menu, select the "Create a CNC Mission Script" option.

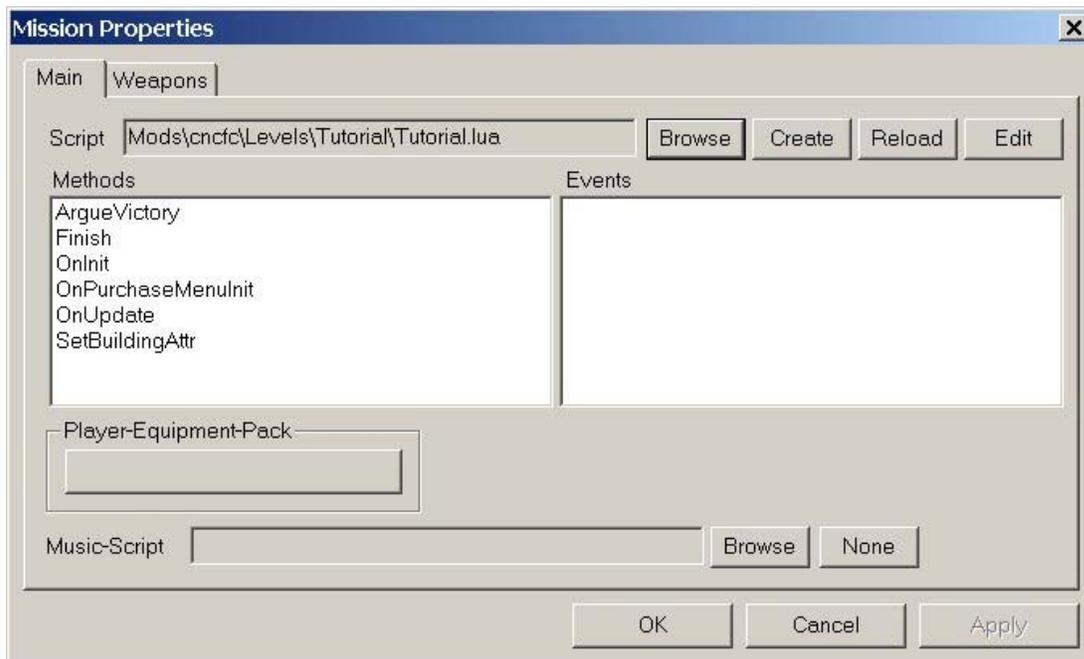


If this menu option is not present, we must add it. To do this, select the "Configure Tools..." option from the Tools menu. Select the "New (Insert)" option (next to the red 'X') and name the new option "Create a CNC Mission Script". You can use something else, but for sake of simplicity, keep it as this. Make sure you have this new tool selected. At the bottom, insert for the Command: "#Game:CNCCloneMissionScript()". The tool will now be available in the Tools menu.



After executing this tool by selecting it in the menu, a clone of a basic CNC mission script will be created and placed in your map folder. It will share the same name as your map name with the file extension "lua". Now, we must tell the Editor that we wish to use this file as our mission script. First, select the "New" option in the Mission menu. Type in "CNC" and click OK. This will create a new mission profile for your map with the name "CNC". To confirm that it was created (and to set it as the active mission), select the "Select..." option in the Mission menu. Highlight the CNC mission and click OK.

Now, select the "Properties..." option in the Mission menu. In the Main tab, you will see at the top several options to select a Mission Script. Click the "Browse" button at the top and traverse to your map folder. Open the ".lua" file that shares your map name. It will now become the active mission map.



Selecting the "Edit" button at the top will open up the mission script and allow you to alter its contents. There are many key elements that are occurring inside this mission script that come free with the method we have just used to create it. I will go over the internals of this mission script later. But for now, we must interface with our buildings!

## 2. Interfacing with a Building

Buildings are a key role in CNC gameplay. There are two things that identify a building: which team it belongs to and which class type it represents. The team is pretty much self-explanatory. It decides who receives its benefits and who can harm it. The class type determines what type of logic it is responsible for executing. By default, CNCFC focuses around 6 prominent building classes:

- SoldierFactory - Allows for Soldier and Weapon purchases. Without it, you can't buy these items.
- VehicleFactory - Allows for Vehicle purchases. Without it, you can't buy these items.
- Refinery - Periodically gives out resources to all players on the team, keeping them well-funded.
- PowerPlant - Ensures power to all structures to keep them fully functioning. Without it, production decreases, causing a rise in cost for raw materials and lesser amounts of productivity in certain structures (Refinery takes longer to dish out resources, VehicleFactory takes longer to cool down after a vehicle is constructed, etc.)
- BaseDefense - Operates remote gun turrets placed around the base for automated defense. Without it, these turrets go down, forcing manual operation.
- CommCenter - Handles team communication including radar operations. Also provides security measures for intruder detection in other team's buildings.

These classes are actually defined from the mission script. By default, these 6 are being defined in the template mission script you are using. This is a good thing, because these six building classes are the only ones with building scripts created! If you want to make your own unique class, you will need to create your own building script. This is beyond the scope of this tutorial, so for now, limit yourself to these 6 building types.

Interfacing (and subsequently creating) these buildings is extremely easy. Buildings are interfaced through several objects, allowing you to build a building lego-style. First, create a Simple Entity. Select the model you'd like to use and place it anywhere on the map. Setup the properties as you wish. The key thing to do here is to describe which building you wish this entity to interface with. To do this, simply state the team and class type of the building you wish to interface with in the "CNC\_Building" section of its properties. For example: if you wish to interface with the Soldier Factory on the Red team, then type "red" for the Team property (case-sensitive!) and "SoldierFactory" for the Class property. That's it!



SimpleEntity Properties	
[-] Animation PoweredUp	
AnimStart	
AnimStop	
ab Animation	Default
? Loop	<input type="checkbox"/> False
? Playing	<input type="checkbox"/> False
n Speed	1
[-] CNC_Building	
ab Class	SoldierFactory
n DamageFactor	1
? PlayDestroyedAnim	<input type="checkbox"/> False
? PlayPoweredDownAnim	<input type="checkbox"/> False
? PlayPoweredUpAnim	<input type="checkbox"/> False
ab Team	red
[-] Physics	
? ActivateOnDamage	<input type="checkbox"/> False
n Density	-1
? FixedDamping	<input type="checkbox"/> False
Impulse	1,2,3
n Mass	700
? Resting	<input type="checkbox"/> False
? RigidBody	<input type="checkbox"/> False
? RigidBodyActive	<input checked="" type="checkbox"/> True
ab Type	Unknown
? UseSimpleSolver	<input type="checkbox"/> False
n damping	0
n max time step	0.01
n sleep speed	0.04
n water damping	0

When the map is loaded, because this particular type of building has been interfaced at least once somewhere, it will be created. A building controller (the abstract logical object that contains the building script and attributes) is created and this entity now serves as a method for a player to interact with the building (such as damaging it). It's a very cool method and is very open for designers to make their own unique buildings, both in looks and functionality!

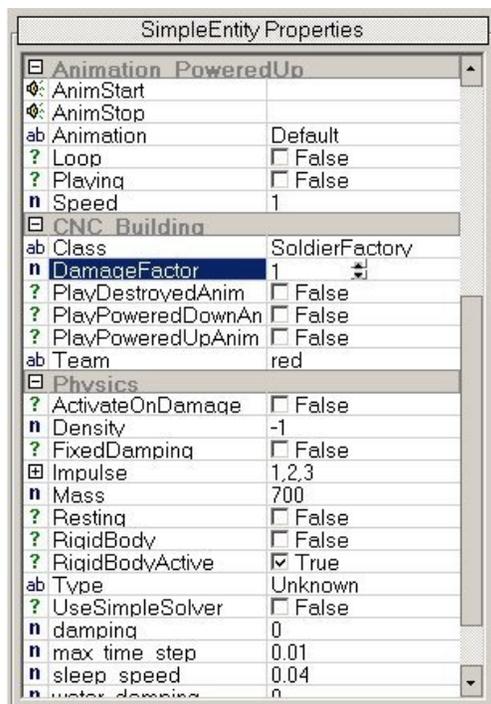
The other properties will be discussed later when we go over advance building features. Repeat this step as many times to place all your building interfaces on your map. The next thing we will do is add some lights, damage effects, and critical points to our building.

### 3. Advance Building Features

Although making a building "out of legos" can offer some very unique and creative designs, it simply isn't enough. Advance Building Features allow you to add elements to your buildings that add to both the overall look and feel of the map as well as the gameplay patterns that may evolve around it. Things like lights and damage effects can bring the building more to life while also serving as a method to inform the player of the building's current status. Critical Points can act as guides to lure the player into harder-to-reach areas with the promise of allowing more damage to be applied to the building.

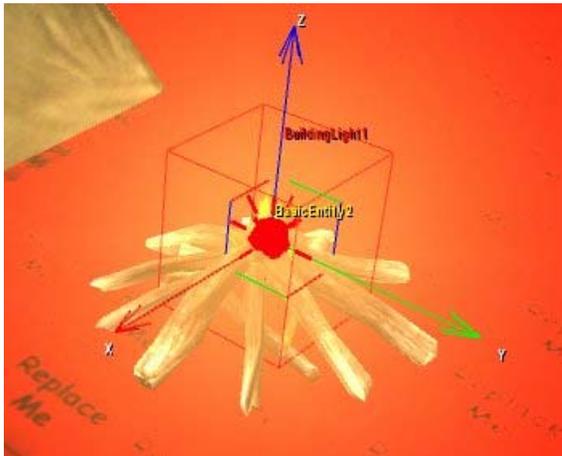
#### Critical Points

We will start with **Critical Points** as we have already seen them in action. In fact, each of your interfaces are Critical Points. A Critical Point is nothing more than an interface with a higher damage modifier applied to it. You may have seen this property earlier when setting up your interfaces. Included in the "CNC\_Building" table of the interfaces properties lies a numeric property called "DamageFactor". By default, this value is set to 1. Whenever this interface is attacked, the amount of damage received will be multiplied by this value; therefore, increasing it will cause higher amounts of damage to be applied to the building. Putting a generator deep inside your building and giving it a Damage Factor of '5' means if the player can make it to this generator, they can kill the building 5x faster by attacking it over the adjacent interfaces. Use it wisely!

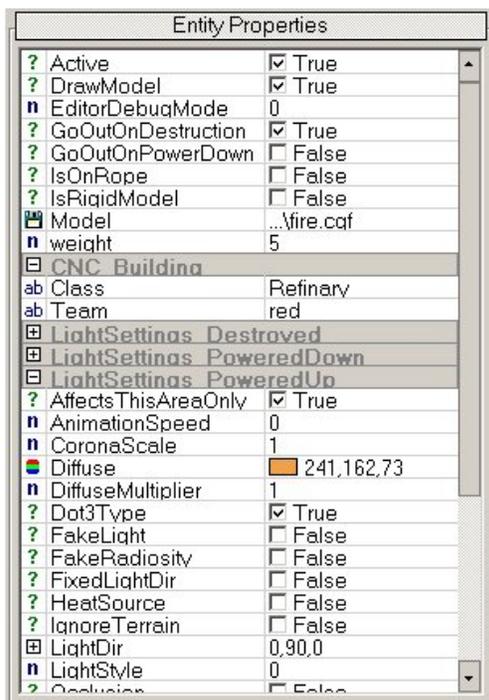


## Building Lights

Next, **Building Lights**. Building lights operate exactly the same as Dynamic lights with one major difference: each one houses three separate instances of properties that define how the light works. The one that is used is determined by the current status of the building for which it belongs - if the building is powered, if the building is not powered, or if the building is destroyed.



The building light entity is located in the Lights folder. Place one on your map and set up the light properties for each status as you see fit. They are named accordingly. To link a building light to a building, simply fill in the Team and Class properties in the CNC\_Building section of the properties.



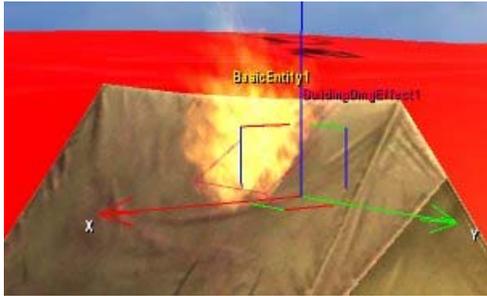
While in Editor mode, a special property can be manipulated to debug your building light. By setting the value of the "EditorDebugMode", you can view the building light in one of its states to tweak it. The valid values are:

- 0 - PoweredUp Mode
- 1 - PoweredDown Mode
- 2 - Destroyed Mode

This property is ignored in game mode.

## Building Damage Effects

**Building Damage Effects** can serve two purposes: they can be used as particle emitters to place elements such as fire or running water and/or they can be used to create explosions which can harm nearby players/entities. A damage effect is turned on/off depending on the health percentage of the building for which it belongs.



There are several key properties that allow you to customize their behavior. On top of the minimum and maximum values that define the health range the building must be in for the damage effect to be active, you can also define if the damage effect should permanently stay on once it has turned on once, if it should never become active again after the first time it was turned on, and/or if it should turn off once the building is destroyed.

The Particle properties are the same as they are when dealing with a Particle Spray or Emitter. The Explosion properties are self-explanatory. If you wish to have it create an explosion, be sure to set the "CreateExplosion" property to true. An explosion will be created each time the effect is turned on.

The damage effect entity is located in the Particle folder. Place one on your map and set up the particle effect or explosion properties as you see. To link a building light to a building, simply fill in the Team and Class properties in the - you guessed it - CNC\_Building section of the properties. It is useful to use these as image notes for the building's health status as well as explosion emitters upon death to kill anyone in the building when it is destroyed, if you want to be evil.

Entity Properties	
? CreateExplosion	<input checked="" type="checkbox"/> True
n EditorDebugMode HealthPercent	25
? GoOutOnBuildingDeath	<input type="checkbox"/> False
n MaxHealth	50
n MinHealth	0
? RangeInclusive	<input checked="" type="checkbox"/> True
? StayOn	<input checked="" type="checkbox"/> True
? UseOnce	<input type="checkbox"/> False
<b>[-] CNC Building</b>	
ab Class	SoldierFactory
ab Team	red
<b>[-] ExplosionParams</b>	
n damage	1000
n impulsive pressure	200
n radius	3
n rmax	20
n rmin	2
<b>[-] ParticleParams</b>	
ab ParticleEffect	fire.fire a5
n Scale	1
n SpawnPeriod	0.1
n UpdateRadius	50

While in Editor mode, a special property can be manipulated to debug your damage effect. By setting the value of the "EditorDebugMode\_HealthPercent" property, you can activate/deactivate the effect as you see fit. Just set it to the % value that you wish to have the "building" be at. This property is ignored in game mode.

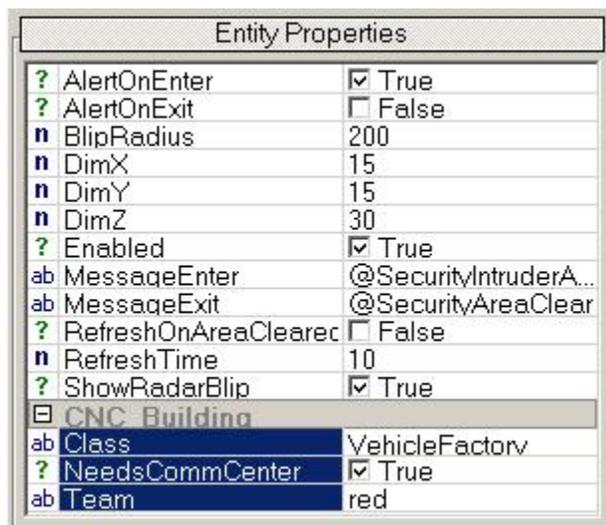
## Security Alerts

**Security Alert** Zones serve as a unique feature for altering gameplay balance on your map. A Security Alert is an encapsulated area of space that when penetrated by an enemy soldier, an alert message and radar blip are sent to all team players to alert them of the enemy's presence. A Security Alert is typically used in conjunction with the Communications Center - when the Comm Center goes down, they stop functioning - however this behavior can be turned off to implement a "stand alone" security zone. Regardless of this setting, if the building that the security zone is connected to is destroyed or loses power, the security zone will stop responding.

Security Alerts can send messages each time an enemy soldier enters the area and/or when all enemy soldiers have cleared the area. The message that is sent (and permission to send the message) is customizable in the entity's properties. You can also alter the blip creation flag and the blip size on the radar. A timer can be applied to the security alert, creating a rest period in-between alert messages.

Security Alerts can also be used to create fancy bells and whistles when used in conjunction with the Event system. You have the ability to manually trigger the alarms ("On RaiseAlarm", "On LowerAlarm") or manipulate other entities when the alarms are triggered ("On AlarmSignaled", "On AlarmCleared"). This gives you the ability to spaghetti link multiple security zones (one goes off, they all go off!) OR more excitingly turn dynamic lights and 3D sounds on/off. Think of the possibilities: when an enemy enters the back door of the Soldier Factory, a large light comes on over his head and loud siren goes off while he is surrounded by his enemy. It's all possible due to Far Cry's excellent event system!

These are located in the Triggers section. Just drop one in, set up the CNC\_Building table, and adjust the area dimensions and the other properties to your liking. To obey CNC rules, keep a Security Alert connected to the Communications Center.



The screenshot shows the 'Entity Properties' window for a Security Alert entity. The window is titled 'Entity Properties' and contains a list of properties and their values. The properties are:

Property	Value
? AlertOnEnter	<input checked="" type="checkbox"/> True
? AlertOnExit	<input type="checkbox"/> False
n BlipRadius	200
n DimX	15
n DimY	15
n DimZ	30
? Enabled	<input checked="" type="checkbox"/> True
ab MessageEnter	@Security/IntruderA...
ab MessageExit	@SecurityAreaClear
? RefreshOnAreaCleared	<input type="checkbox"/> False
n RefreshTime	10
? ShowRadarBlip	<input checked="" type="checkbox"/> True
<b>CNC_Building</b>	
ab Class	VehicleFactory
? NeedsCommCenter	<input checked="" type="checkbox"/> True
ab Team	red

### **Interface Animation**

The final advance feature, **Interface Animation**, are located in the same area as Critical Points - that is, in the interface entity's properties. Similar to building lights, there are three separate instances of animation settings that can be defined. Three boolean properties allow for these animations to be enabled/disabled.

## 4. Other Essential Entities

There are a few other entities that should be noted as they serve important roles in a CNC-missioned map. These entities include spawn points and purchase terminal access areas.

### Player Spawn Points

**Player Spawn Points** server as locations where a player can spawn based on their team status. A player spawn point is very simple and is located in the Characters folder. Place one on your map and define which team can spawn here. You can also easily enable/disable a spawner. The orientation of the spawner is used to define the facing as well as the position of the player who spawns at it.

One other property that may catch your eye is the "SpawnClassType" property. This can be used to define the character class type a player inherits if they spawn at this location. By default, this value is "basicinfantry" but you can change it to any of the following:

- "basicinfantry" - Basic infantry soldier with modern weaponry.
- "engineer" - Engineer class with an engineer tool and sticky bombs.
- "closeinfantry" - Shotgun trooper
- "sniperinfantry" - Long-distance sniper infantry
- "heavyinfantry" - Lots of ass kickin'.

These class types are pre-defined for you. You can make your own; however, this is out of the scope of this tutorial.



Entity Properties	
? Enabled	<input checked="" type="checkbox"/> True
ab SpawnClassType	basicinfantry
ab Team	red

## Vehicle Spawn Points

**Vehicle Spawn Points** work just the same as player spawn points, only for vehicles. Namely vehicles that are purchased and created by the Vehicle Factory. They, too, have a team property used to define which team's vehicles can be created there as well as an enable/disable property. These are located in the Vehicles folder.

Vehicle Spawn Points can be restricted to certain types of vehicles. By enabling this option, you can specify up to three types of vehicles that can be spawned at this spawn location. This is useful if you have a tight restricted area that a larger vehicle may get stuck in if created there. If the option is disabled, any vehicle can and will be created here.



Entity Properties	
? Enabled	<input checked="" type="checkbox"/> True
ab Team	red
[-] VehicleSpawnTypes	
? RestrictSpawnByType	<input checked="" type="checkbox"/> True
ab VehicleSpawnType1	Buggy
ab VehicleSpawnType2	
ab VehicleSpawnType3	

## **Gun Emplacements and Security Guns**

**Gun Emplacements** - and their weaker counterparts **Security Guns** - work in conjunction with the Base Defense system and allow for automated base defense logic to help protect the friendly base from enemy infiltrations. Automatic defense is sort of a double-edged sword: since we cannot always plan for an exact number of players per team, Gun Emplacements can help protect a base, letting the players in the game focus more on attacking the other enemy. However, Automatic defenses can also prove to be quite a deterrent in regards to team strategy and can severely disillusion a player. Plus, where's the fun in having something else always kill for you? It's in my opinion that Gun Emplacements should be kept at a minimum and left with "back doors" where the enemy can sneak past them without being detected. Security Alerts should be used to warn the player of enemy intrusions in these areas.

But if you insist on using one (and you should use at LEAST one!), the Gun Emplacements and Security Guns will use a goal-orientated basic AI to hunt down and destroy any enemy soldiers that stray too close to them. When the Base Defense System building is destroyed or loses power, these guns will stop functioning. This is where the difference from the two come in to play:

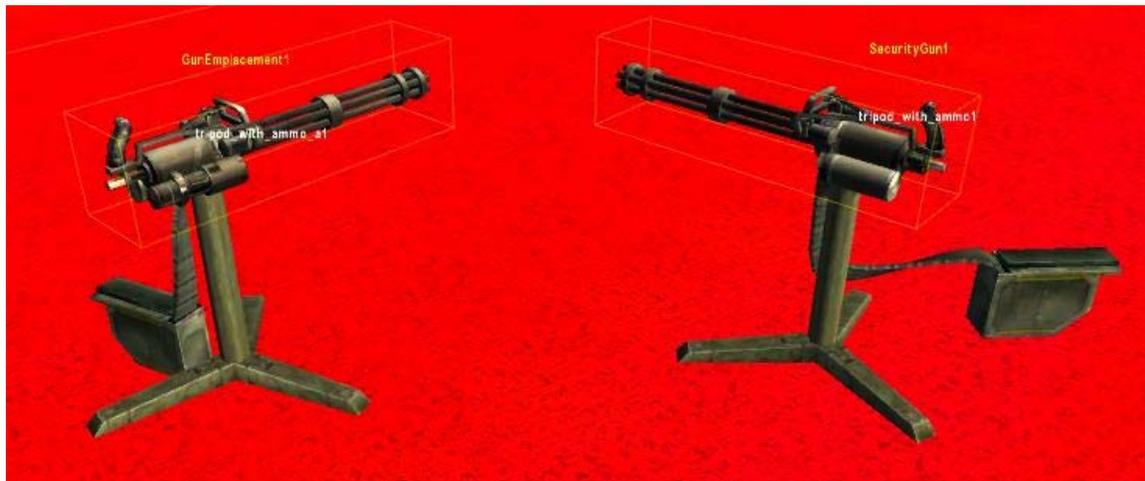
- Gun Emplacement - Linked to the Base Defense System. A player can manually control a Gun Emplacement at any time. The AI operates it when a player is not. If the BDS goes down, a player can still jump in and use it to protect the base.
- Security Guns - Linked to the Base Defense System and a building of your choice (specified in the entity's properties; can be ignored). When either building goes down, they stop operating. A user cannot operate one manually.

Both can have the model used for the weapon and the actual weapon itself specified from their properties. Typically "MG" (Machine Gun) and "RL" (Rocket Launcher) are used for the weapons, but any weapon can be used. Remember, when the AI is controlling the weapon, it will appear like a ghost is firing the weapon. This means things like fire rate and damage come from the weapon's properties and not the Gun Emplacement's. What AI settings can be adjusted are:

- Range - Area of affect that the GE will look in for any enemies. The closest enemy is the one fired upon.
- IdleMovementSpeed - How fast the weapon rotates when the AI does not have a target. (Idle animation - used to alert the player that the AI is active on this)
- UpdateTargetTime - How often (in seconds) the AI will look for a new target. The smaller the number, the more "alert" the AI appears, but having too many Gun Emplacements looking for a new target too quickly may cause the game to perform less optimally. 0.2 (1/5th of a second) is a reasonable number.
- StickToTarget - If turned on, the AI will "stick" to a target until it is either destroyed or leaves firing range. If another enemy gets closer than the one it is firing at, it will ignore it.

Other properties worth mentioning are the "ang..." properties on the Gun Emplacement. These specifically the angle limits a player is fixed to when using the weapon. If you want the player to be able to look all around when controlling the Gun Emplacement, set the H limit to '0'.

Both Gun Emplacements and Security Guns can be found in the AI folder.



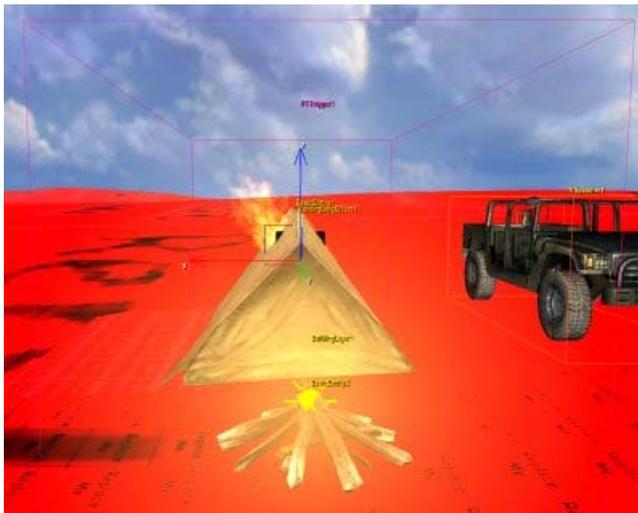
Entity Properties	
? DestroyOnBaseDefens	<input checked="" type="checkbox"/> True
GunModel	...miniqun.cqa
ab Team	red
n angHLimit	60
n angVLimitMax	10
n angVLimitMin	-20
n mountHandle	0.6
n mountHeight	1.1
n mountRadius	0.8
ab weapon	MG
AI	
n IdleMovementSpeed	30
n Range	100
? StickToTarget	<input type="checkbox"/> False
n UpdateTargetTime	0.2

Entity Properties	
GunModel	...miniqun.cqa
ab weapon	MG
AI	
n IdleMovementSpeed	30
n Range	100
? StickToTarget	<input type="checkbox"/> False
n UpdateTargetTime	0.2
CNC Building	
ab Class	Refinary
? GoOutOnBuildingDeath	<input checked="" type="checkbox"/> True
ab Team	red

## Purchase Terminal Zones

Finally, **Purchase Terminal Access Areas** are basic triggers which encapsulate an area of space that allow any player on the specified team who enters it to access the Purchase Menu. The Purchase Menu, or PM, is a menu that lets a player trade in credits for a new soldier class, build a new vehicle, obtain a new weapon or refill their health/armor/ammo. The contents are defined in the mission script and this will be explained later.

These are located in the Triggers folder. Just drop it, size it to your desired dimensions, give it a team, and enable or disable it. When a player on the same team enters the area, they will be prompted with instructions on how to access the PM.



Entity Properties	
n DimX	5
n DimY	10
n DimZ	5
? Enabled	<input checked="" type="checkbox"/> True
ab Team	red

## 5. Mission Script Internals

The Mission Script is responsible for setting up the attributes that are used in the CNC game mode. These properties include the contents of the Purchase Menu, the valid building team and class types, and Pools. The Mission Script also has the ability to alter the initial properties of a building to tweak their behavior in accordance to the map design.

### **OnInit and ArgueVictory**

When the level is first loaded, the **OnInit()** routine is called. Here, you must define the building teams and classes that are valid. You must also set up any Pools. Pools are used to execute special behavior routines when certain buildings have been destroyed. They are typically used for winning conditions. A building can be added and removed from a pool. When the pool has been depleted of buildings, the team that owns the pool claims victory. However, if you specify the owner of the pool to be the argue victory team, the **ArgueVictory()** routine is called. Here you can define special behavior rules. This is an advance concept so it is best to leave it alone until you get more familiar with it.

By default, the "red" and "blue" teams are added as valid building owners. Furthermore, the building classes "SoldierFactory", "VehicleFactory", "Refinery", "PowerPlant", "BaseDefense", and "CommCenter" are added as valid building class types. Also, two victory pools that belong to the red and the blue team are added, with each building be placed in the appropriate pool. So, when all red buildings are killed, the Blue team will win and vice versa.

### **OnPurchaseMenuInit**

The **OnPurchaseMenuInit()** routine is called when the contents Purchase Menu is ready to be defined. Each team has its own purchase slots which can each be filled with their own purchase selection. Each purchase selection has a specified type, a label that is displayed in the PM, a purchase type, a cost amount, and a flag that determines if it is available to be purchased at this time or not. Valid types include:

- "Soldier" - Character class
- "Vehicle" - Vehicle that is created at a vehicle spawn point
- "Weapon" - One weapon and two ammo types/amounts.
- "Refill" - Health, armor, or unlimited ammo pack amounts rewarded to player.

Each purchase selection also holds on to what is physically bought. What is placed here is based on the type. By default, a wide range of different combinations are placed in the Purchase Menu ranging from all four types. Studies these and tweak them as you like.

## **SetBuildingAttr**

Finally, the **SetBuildingAttr()** routine is called each time that a building controller is freshly created. This is where you can overwrite a building's attributes and/or add the building to any number of pools that you wish. The building is passed in as an argument. You can get the team and class type from it via the `GetTeam()` and `GetClassName()` routines. By default, each building is added to the appropriate pool for end game conditions. However, you can detect when a certain building is created and alter specific properties about it. These properties include:

- All Building Types
  - `cnt.health` - (2500) Current health of the building
  - `cnt.maxhealth` - (2500) Maximum health of the building
  - `Properties.Damageclass` - ("Building") Damage class used to modify damage to building based on source and type.
- Vehicle Factory
  - `Properties.iCooldownRate_Powered` - (10s) Length of time (in seconds) that the Vehicle Factory must wait before another vehicle can be purchased after one has been created when fully powered.
  - `Properties.iCooldownRate_Unpowered` - (20s) Length of time (in seconds) that the Vehicle Factory must wait before another vehicle can be purchased after one has been created when not powered.
- Refinery
  - `Properties.iResourceAmount` - (400) Amount of credits each player on the team receives when the resource tick counts down to 0.
  - `Properties.iResourceTick_Powered` - (120s) Length of time (in seconds) between each resource tick when the building is fully powered.
  - `Properties.iResourceTick_Unpowered` - (240s) Length of time (in seconds) between each resource tick when the building has no power.

## **Custom Buildings**

Maps can be packed with custom building scripts which can be used within the map. Located in the Buildings folder, you can place building scripts here for your custom-made unique buildings in your map. Ensure that you describe the building's class in the missions script in order to use the custom building!

The tutorial level is packaged with a unique building implemented as an example.

**Consult the included Tutorial level for a completed CNC-missioned map which has been hinted at throughout this tutorial.**