

Hammer tutorial: How to create a physics-based cloner

Half Life 2: Episode 2

Version 1.0

Author:	Jorge Montolio Conde
Document Date:	09/07/2015

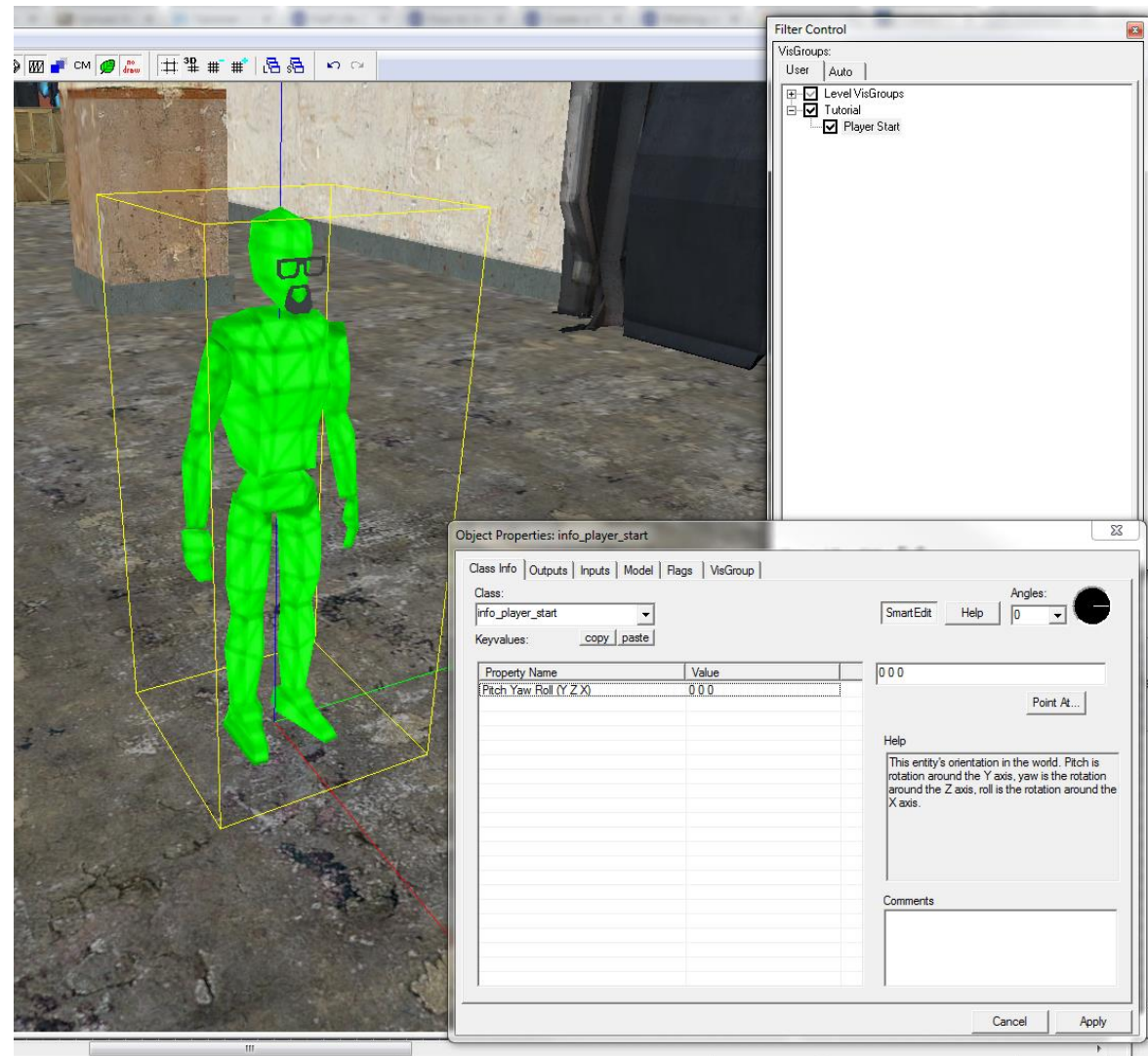
Summary

How does the Cloner work?

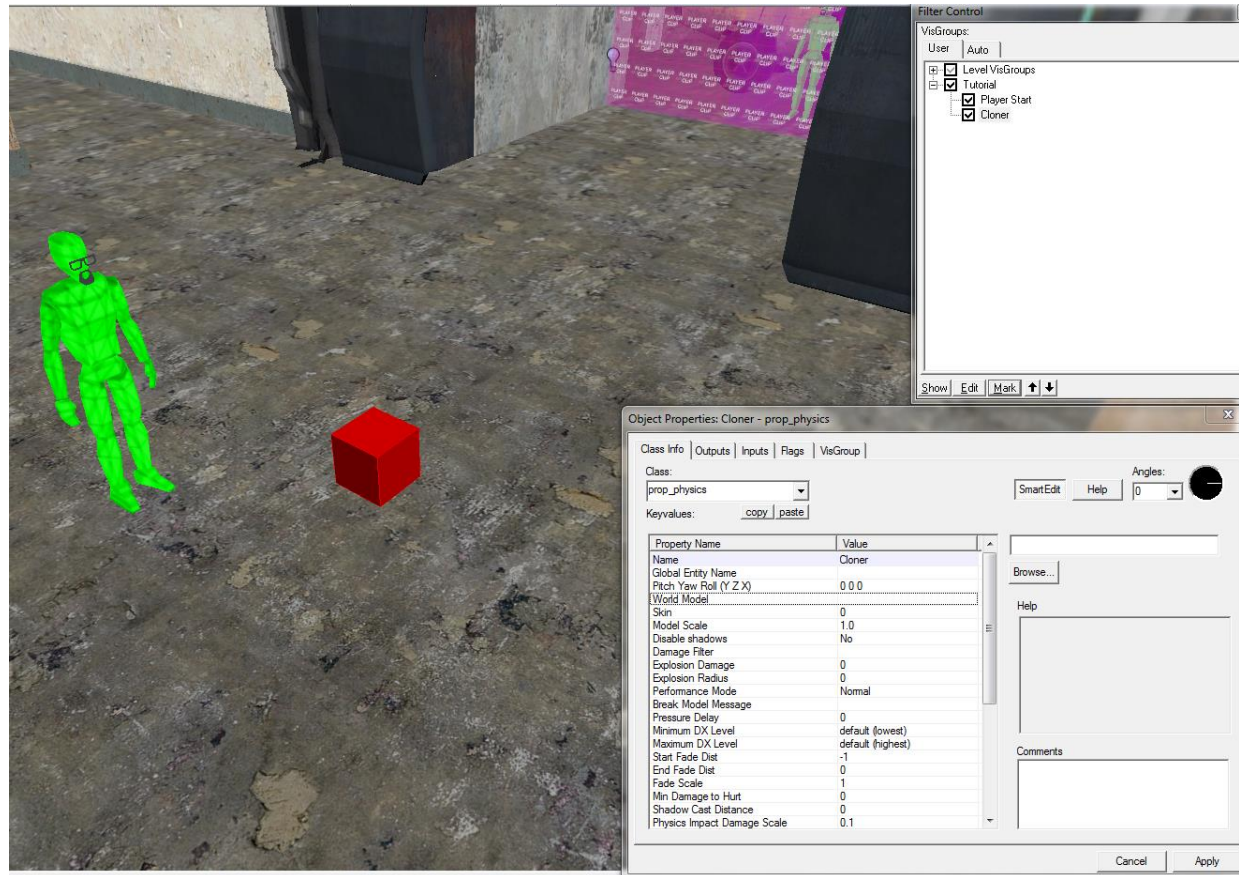
- The player picks up an object with his gravity gun
- The player throws the object.
- The player teleports to the object, once it lands. He is now controlling a "clone". If he looks at the position where he came from, he can see his original body standing there.
- After a few seconds, the player goes back to his original body. In the place of his "clone" we spawn a dummy.

Required entities

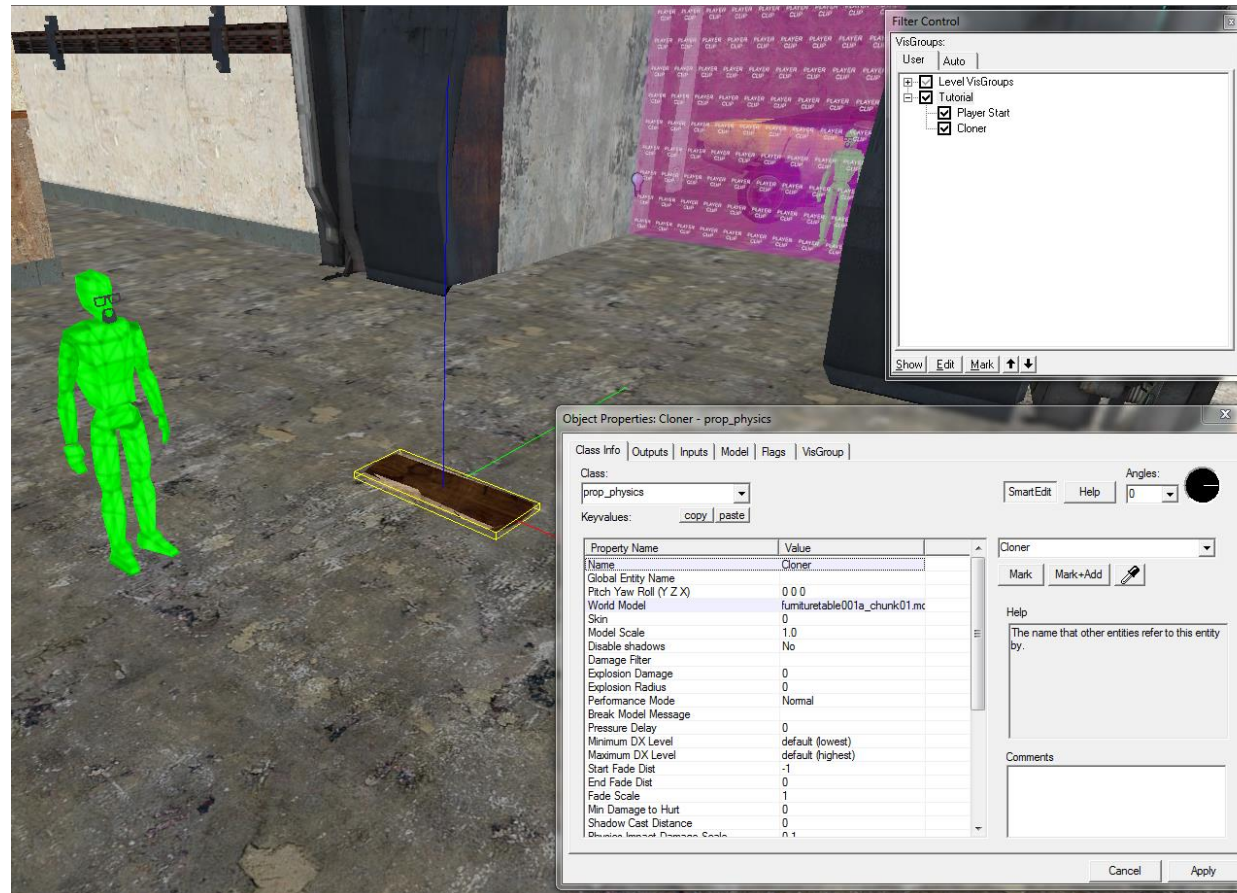
- logic_auto
- info_player_start
- prop_physics
- phys_keepupright
- logic_relay
- logic_timer
- logic_compare
- trigger_teleport
- env_entity_maker
- point_template
- npc_barney
- point_velocitysensor



2. Now we are going to create the object that we will use as a cloner. This object needs to be an entity called **prop_physics**.
- Inside the properties choose a **name** for the object. I used the name "**Cloner**".

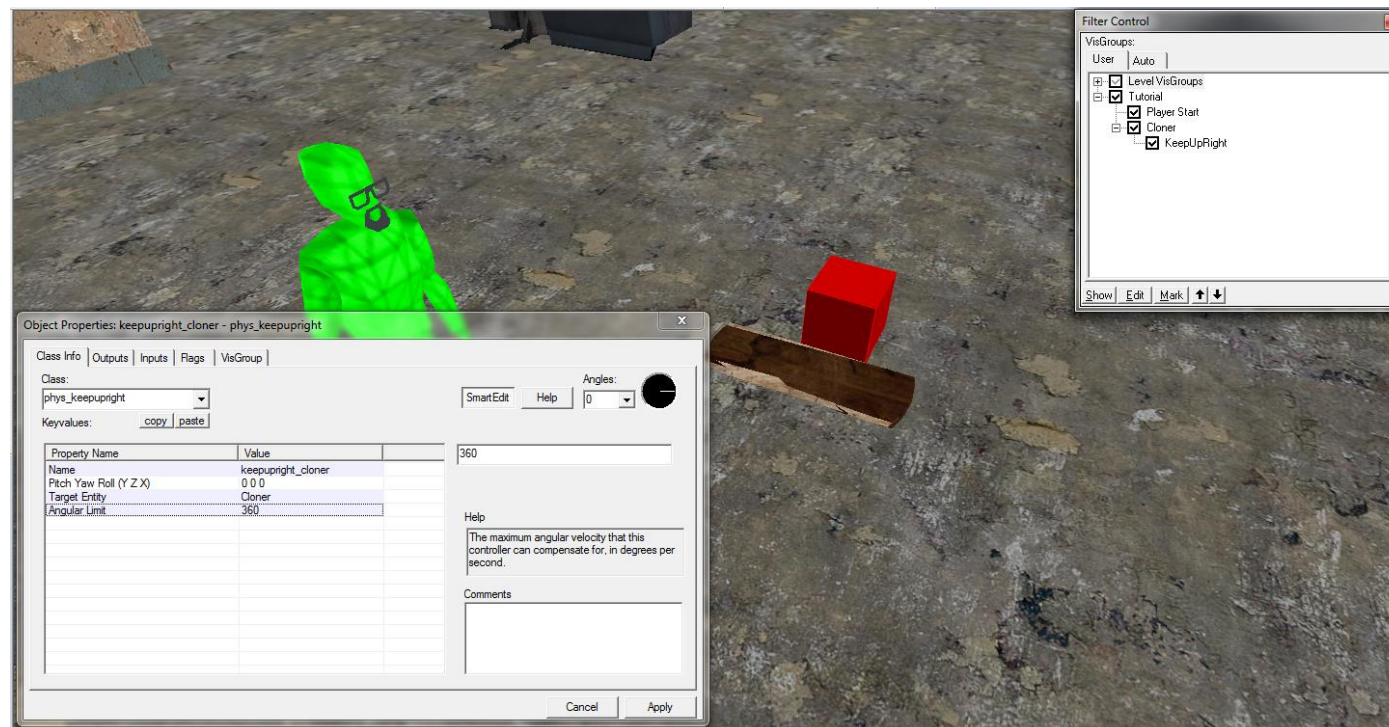


- The physics object does not have any model attached to it, so next we are going to choose a model. We do this by selecting a "World Model" inside the prop_physics properties. It is very important to choose an object with a **flat base**. Otherwise, the player will get teleported with weird angles. For this tutorial we are going to choose the model **furnituretable001a_chunk01.mdl**



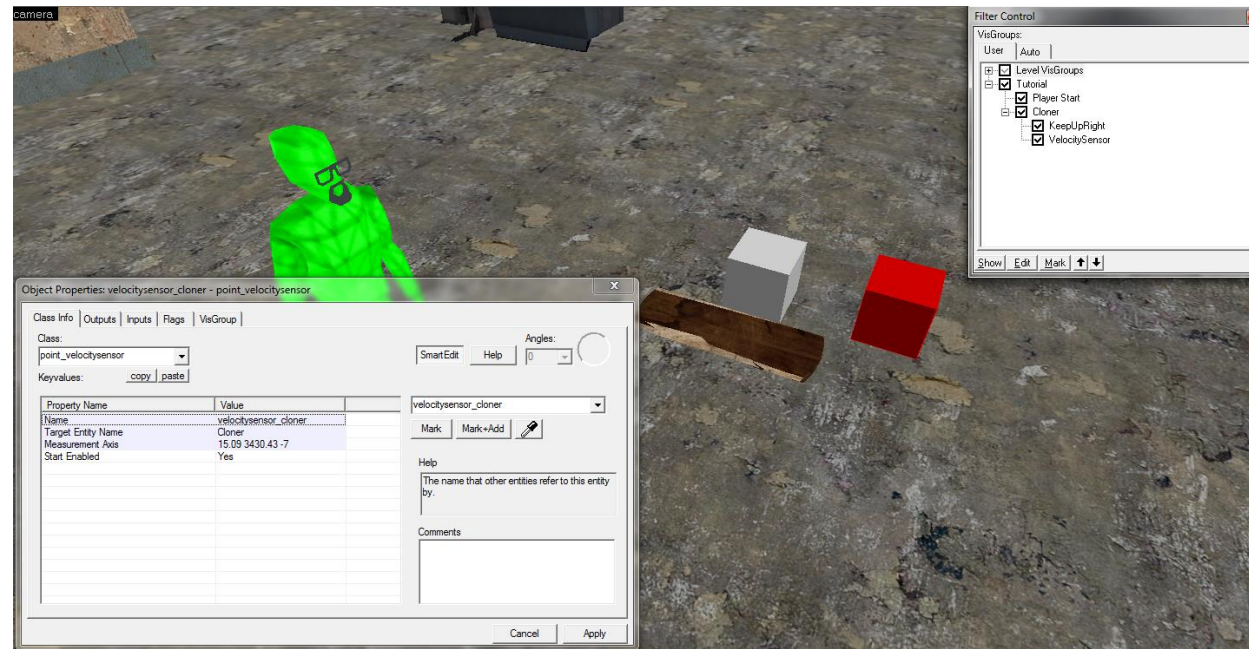
Cloner	
Property	Value
Name	Cloner
World model	<u>furnituretable001a_chunk01.mdl</u>

- Next, we are going to create a **phys_keepupright**. This object will make the Cloner face up permanently. That way, the Cloner cannot tilt when we throw it. If we do not use this object, we have the risk of teleporting to a tilted object, which will tilt the camera as well.
 - Create an entity **phys_keepupright** with the following properties:



keepupright_cloner		
Property	Value	What it does
Name	keepupright_cloner	
Target Entity	Cloner	Selects the entity that needs to be facing up. In our case, the Cloner.
Angular Limit	360-infinity	This is the amount of degrees that the phys_keepupright needs to compensate for, per second. This means that if the cloner rotates 360 degrees in one second, the phys_keepupright will apply an opposite rotation of -360 to compensate. You can set it to a higher number, although 360 seems to work fairly well.

4. Next, we are going to create a **point_velocitysensor** for the Cloner. This velocity sensor will detect when the cloner has stopped, so that we can teleport to it safely. Teleporting to a moving object will create all kind of weird situations that we want to avoid.
- Create a point_velocitysensor with the following properties:



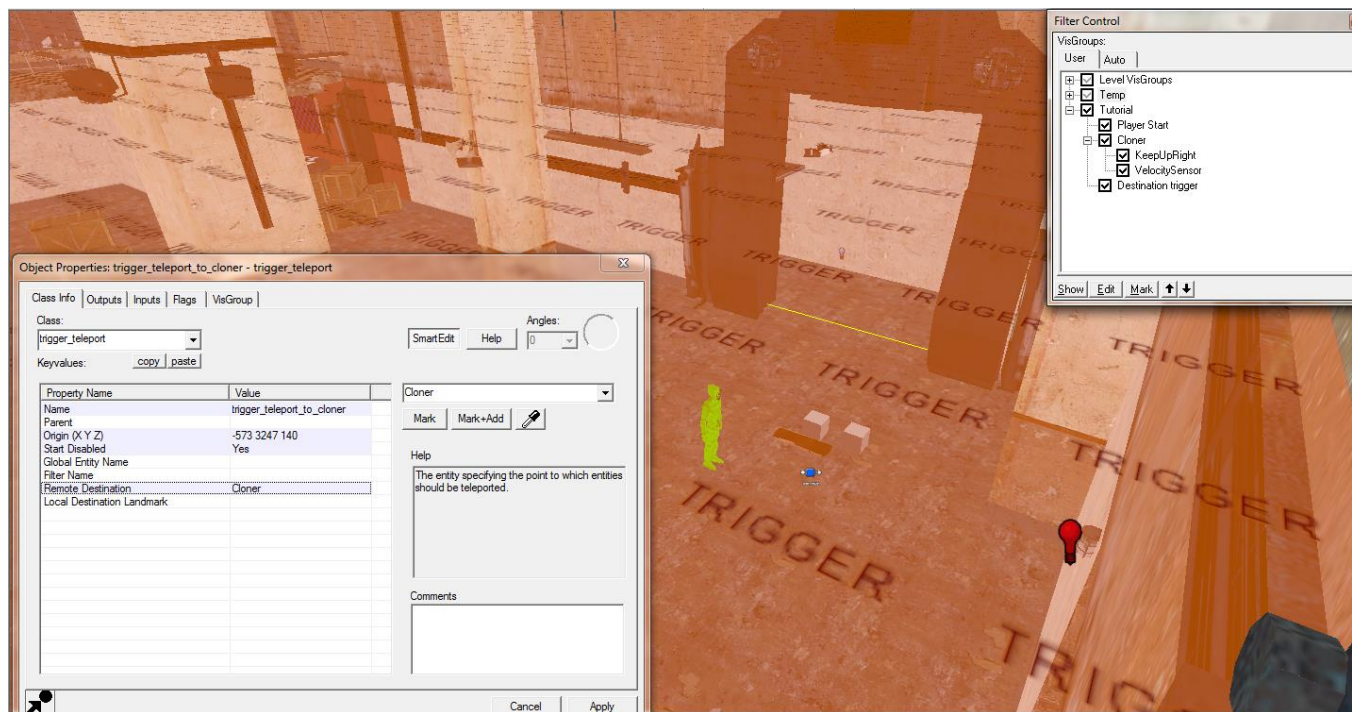
velocitysensor_cloner

Property	Value	What it does
Name	velocitysensor_cloner	
Target entity name	Cloner	The item which velocity we are going to measure

IMPORTANT: The Cloner should be in the map at the beginning of the level. **DO NOT** try to spawn it with an entity_maker or the like, because it will break the phys_keepupright

Setting up the teleporter's destination

1. To teleport ourselves to the Cloner, we are going to use a **trigger_teleport**. Trigger_teleports teleport anything that collides with them to a certain location of our choosing (or in our case, to a certain object). They work **automatically**, i.e., they will teleport things as soon as they are enabled.
 - Create a brush that covers the whole area where you want to use the Cloner. Be sure to include every area; if the Cloner gets out of this trigger, it will stop working.
 - Click on "To entity" and open its properties. In the Class drop-down menu, select **trigger_teleport**.



2. Now we are going to set up some properties in the **trigger_teleport**.

- In the **Class info tab**:

trigger_teleport_to_cloner		
Property	Value	What it does
Name	trigger_teleport_to_cloner	
Start Disabled	Yes	
Remote destination	Cloner	When an object collides with this trigger, the object will be send to "Remote destination". This remote destination can be an entity, like a prop_physics. By setting the "Remote destination" to "Cloner", we are telling the trigger to teleport things to the Cloner's position.

- In the **flags tab**:

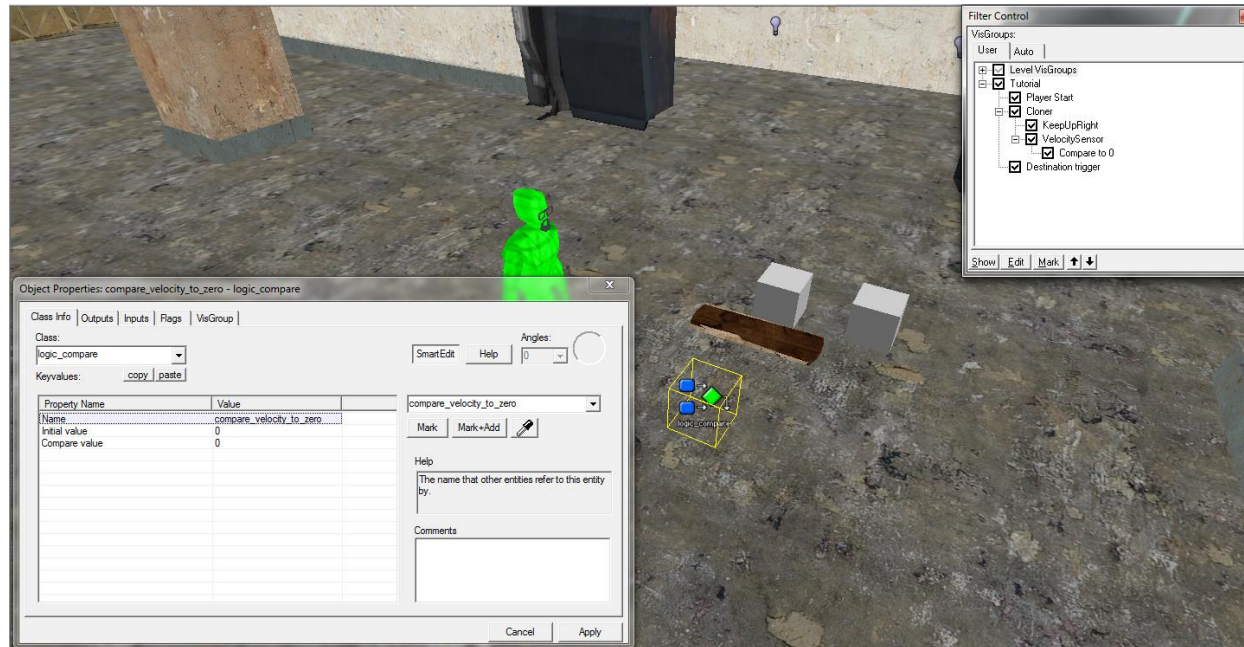
trigger_teleport_to_cloner (Flags tab)		
Flag	Value	What it does
Clients	Checked	Make sure that the trigger only teleports the player. This is important, because the trigger is covering the whole room.

All other flags	Unchecked	
-----------------------	-----------	--

Creating a one-way teleporter

Now that we have all of these elements, we can create the teleporter.

1. First, we need to detect when the Cloner stops. Once it stops, we know it is safe to teleport. We are going to create a **logic_compare** that will compare the Cloner's velocity to zero.
 - Create a **logic_compare** entity
 - Change its name to **compare_velocity_to_zero**. Leave both Value and Compare Value as 0



compare_velocity_to_zero

Property

Value

Name

compare_velocity_to_zero

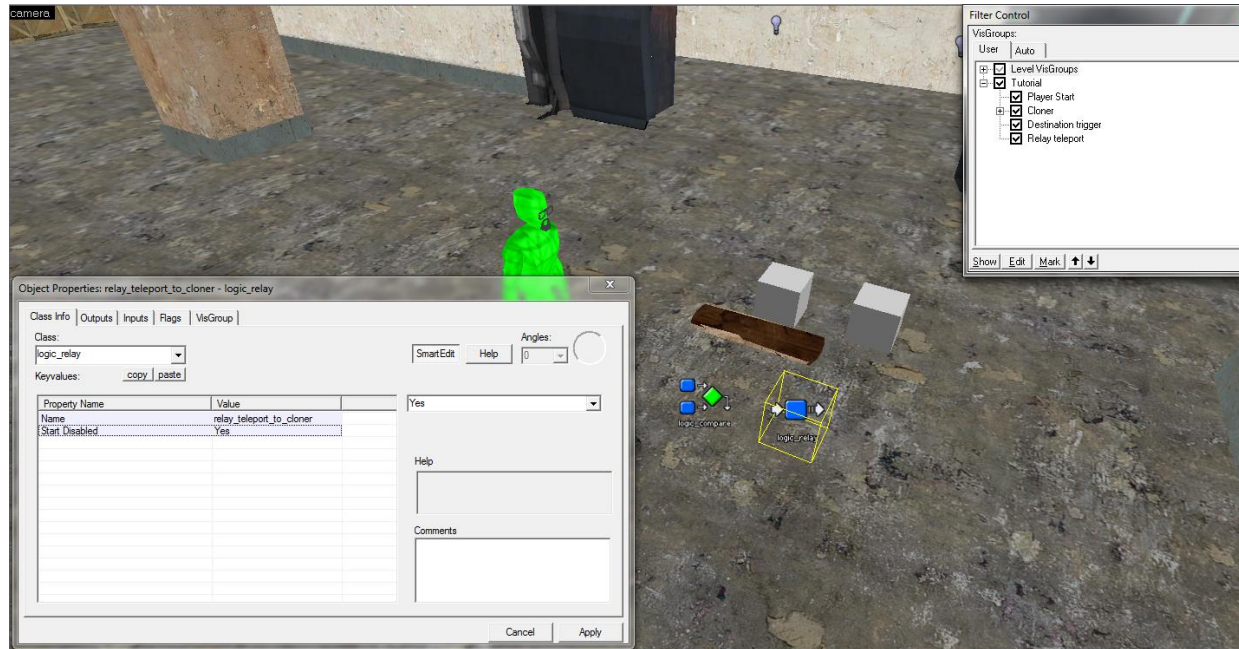
Compare value	0
------------------	---

2. Now we go into the **point_velocitysensor**, and set up the following outputs:

compare_velocity_to_zero (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
Velocity	compare_velocity_to_zero	SetValue		0.00	No	Whenever the velocity of the Cloner changes, the velocity output gets trigger. This command will send the Cloners velocity to the logic compare.
Velocity	compare_velocity_to_zero	Compare		0.01	No	Once the logic_compare has the Cloner's velocity, we tell it to compare it to "Compare Value" (in our case, Compare value is zero). This effectively checks if the planks velocity is zero at any moment.

3. Now we need to teleport the player if the Cloner's velocity is zero. HOWEVER, we do not want to teleport the player every time the Cloner's velocity is zero. If the Cloner is resting on the floor, we do not want the player to teleport to it. Hence, we only want to teleport the player after he picks up the Cloner and throws it. We are going to create a **logic_relay**, which is going to be in charge of activating the teleporting trigger. Whenever we do not want the player to teleport, we will disable this trigger.

- Create a logic_relay



- Change the following properties

relay_teleport_to_cloner		
Property	Value	What it does
Name	relay_teleport_to_cloner	
Start Disabled	Yes	

- Set the following outputs inside the relay:

relay_teleport_to_cloner (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTrigger	trigger_teleport_to_cloner	Enable		0.03	No	Whenever this relay is triggered, it will enable the teleport trigger. This will effectively teleport the player to the Cloner.

Object Properties: relay_teleport_to_cloner - logic_relay

Class Info | Outputs | Inputs | Flags | VisGroup

My Out...	Target Entity	Targ...	Delay	Only Once
OnTrigger	trigger_teleport_to_cloner	Enable	0.03	No

My output named:

Targets entities named:

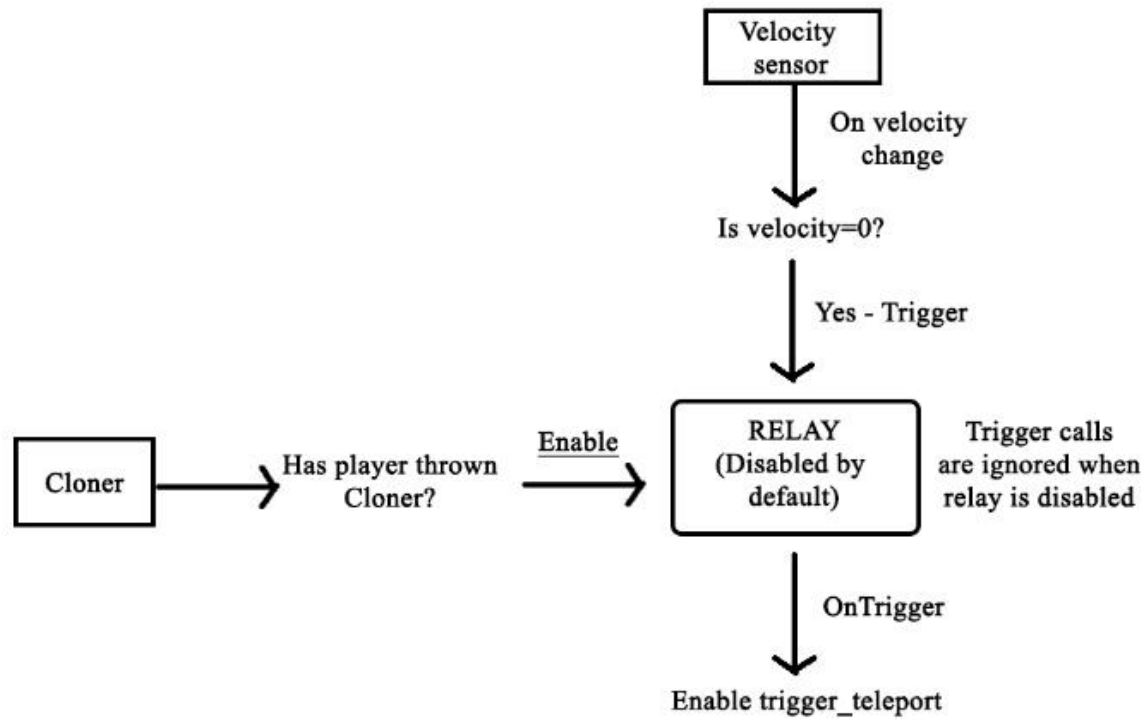
Via this input:

With a parameter override of:

After a delay in seconds of: ☐ Fire once only

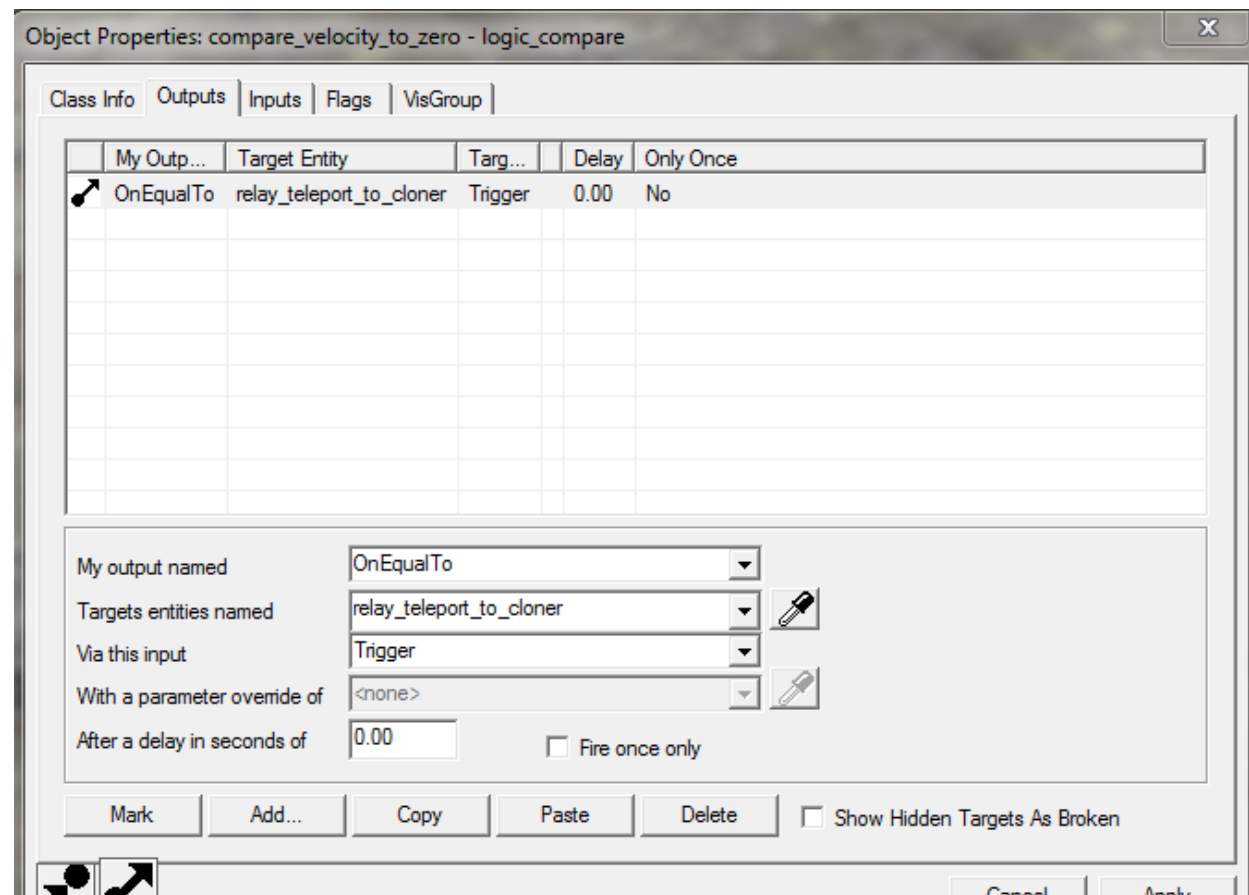
Mark Add... Copy Paste Delete ☐ Show Hidden Targets As Broken

4. Now that we have the relay, we can trigger it every time the Cloner is not moving. The logic for the relay is going to be as follows:



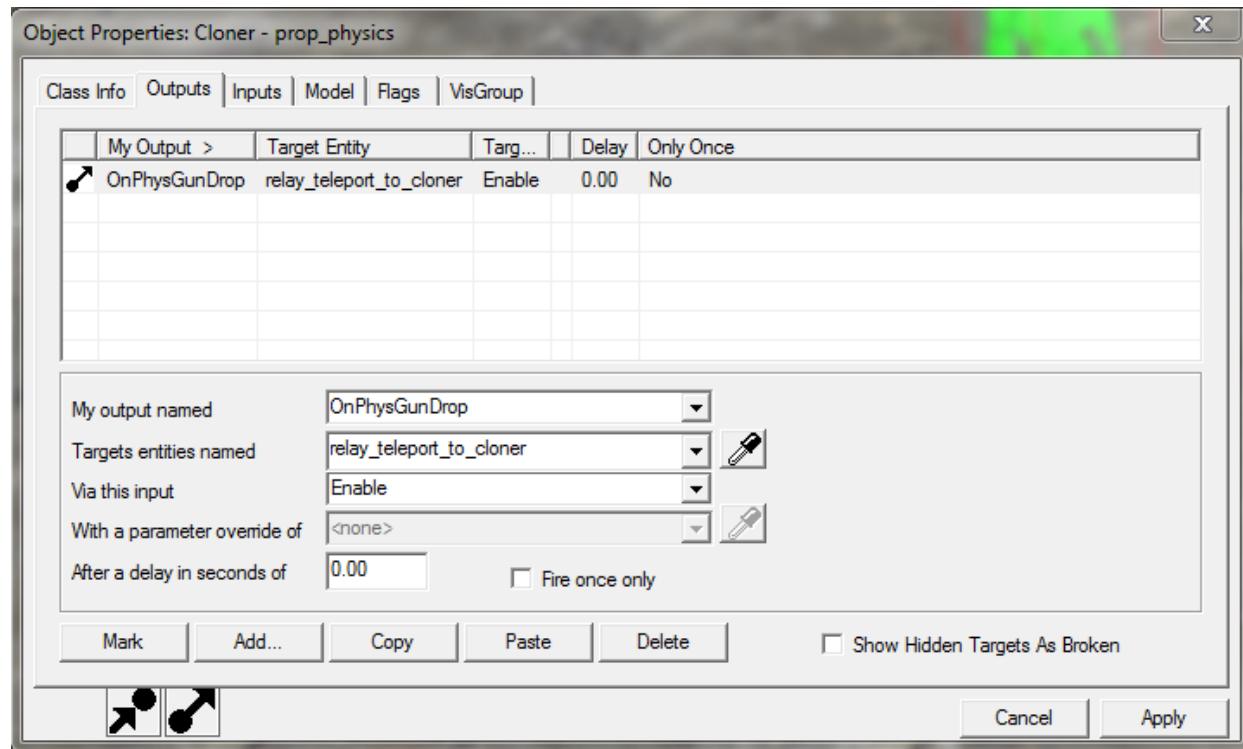
- To implement it, we are going to add some outputs. In the **logic_compare**, add the following outputs:

compare_velocity_to_zero (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnEqualTo	relay_teleport_to_cloner	Trigger		0.00	No	This means that whenever the Cloner's velocity is zero, the relay will be triggered.



- In the **Cloner**, add the following outputs:

Cloner (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnPhysGunDrop	relay_teleport_to_cloner	Enable		0.00	No	Only enables the relay after the player has picked it up and dropped it.



- Finally, we are going to add an output to the **logic_relay**. After the player has teleported, we want to return all of the objects to their original state, so that the player can teleport again if he wants to. For that, we add these two outputs to the logic_relay:

relay_teleport_to_cloner (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTrigger	trigger_teleport_to_cloner	Disable		0.04	No	
OnTrigger	relay_teleport_to_cloner	Disable		0.05	No	

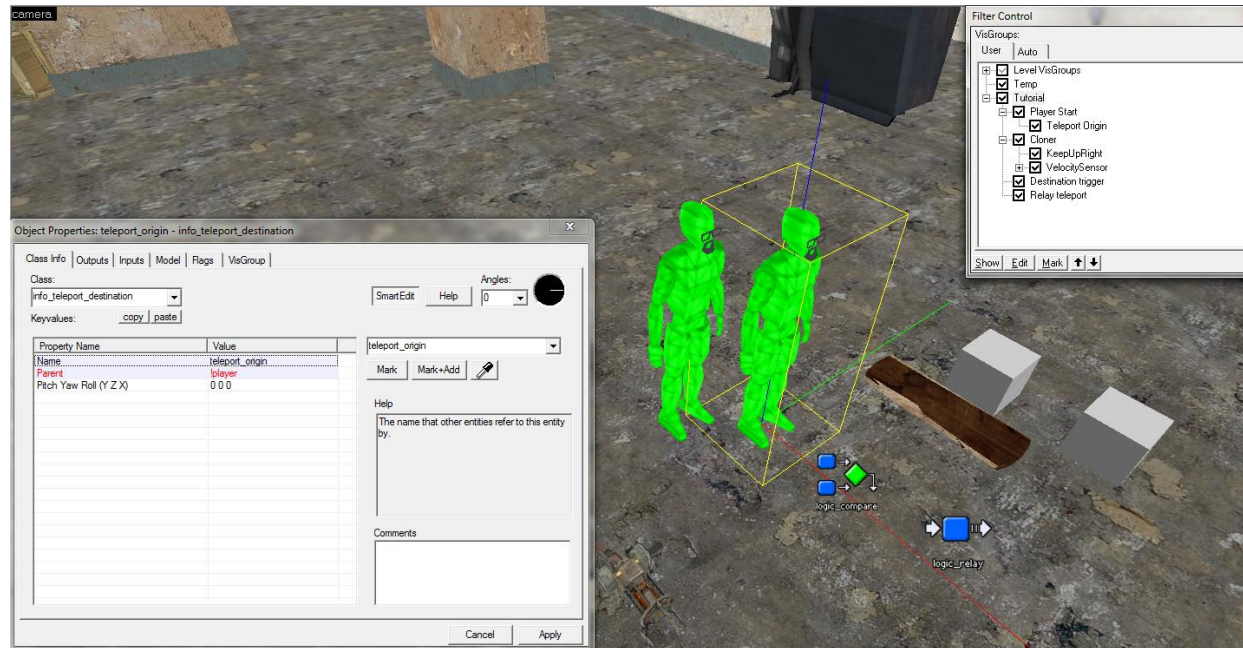
[illegible]

Step 2: Creating a timed teleporter

For this step, we are going to set a timer so that the player goes back to his original position after a certain delay.

Creating the objects

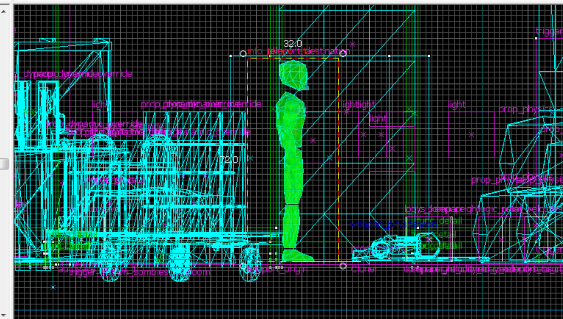
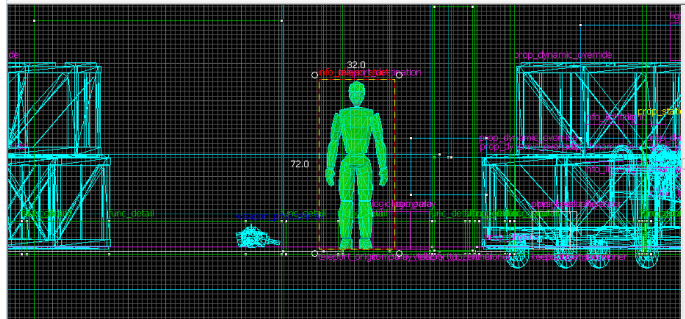
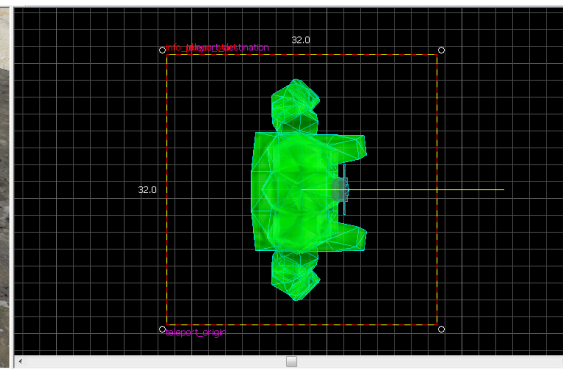
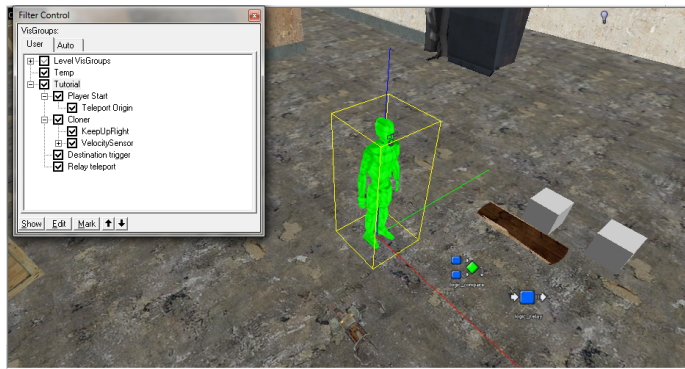
1. First, we are going to create an **info_teleport_destination**. This represents the original position of the player, before he teleported to the Cloner (NOT the position before he threw it).



- Change the following properties in the **info_teleport_destination**:

teleport_origin		
Property	Value	What it does
Name	teleport_origin	
Parent	!player	The info_teleport_destination will go wherever the player goes

- **IMPORTANT:** Make sure that the **info_teleport_destination** overlaps with the **info_player_start** perfectly. Their centers should be on the same point in all the axes. This will reduce the chances of the player being stuck on something after he comes back from the teleporting.\



- Add the following outputs:

logic_auto (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnMapSpawn	teleport_origin	SetParent	!player	0.00	No	

Object Properties: logic_auto

Class Info | Outputs | Inputs | Flags | VisGroup

	My Output >	Target Entity	Target I...	Para...	Delay	Only Once
	OnMapSpawn	teleport_origin	SetParent	!player	0.00	No

My output named

Targets entities named

Via this input

With a parameter override of

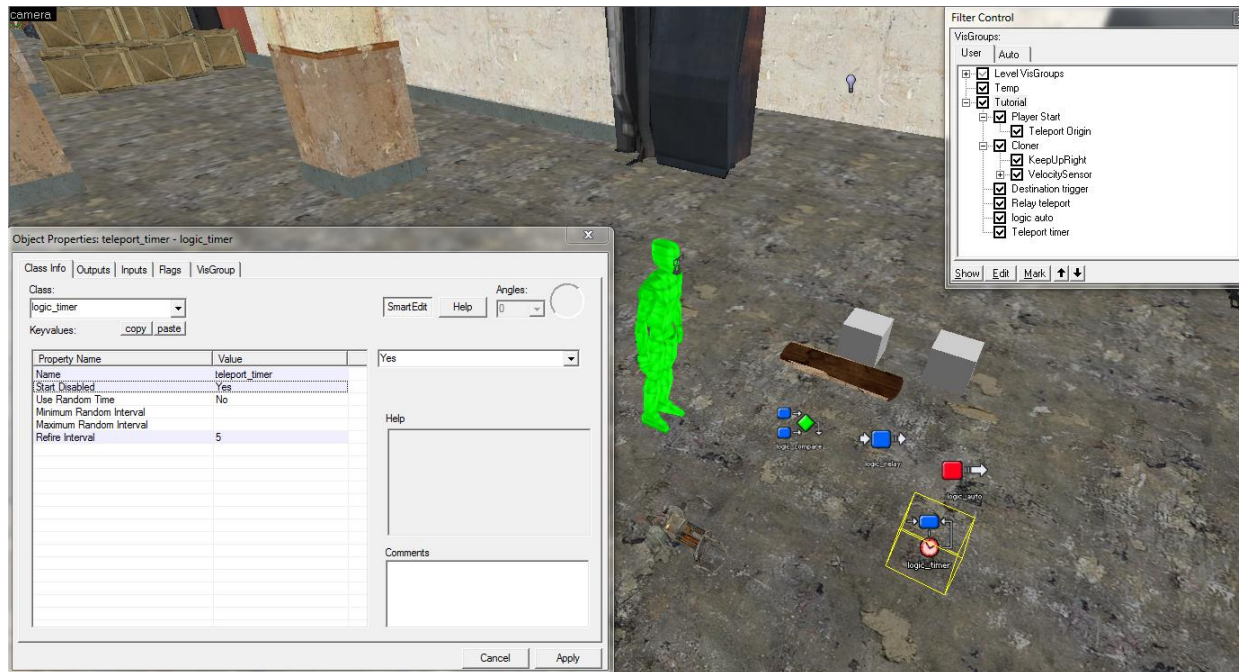
After a delay in seconds of ☐ Fire once only

Mark Add... Copy Paste Delete ☐ Show Hidden Targets As Broken

Cancel Apply

3. The last object we need is a **logic_timer**. After the timer runs out, the player will go back to the teleport origin.

- Create a **logic_timer**:



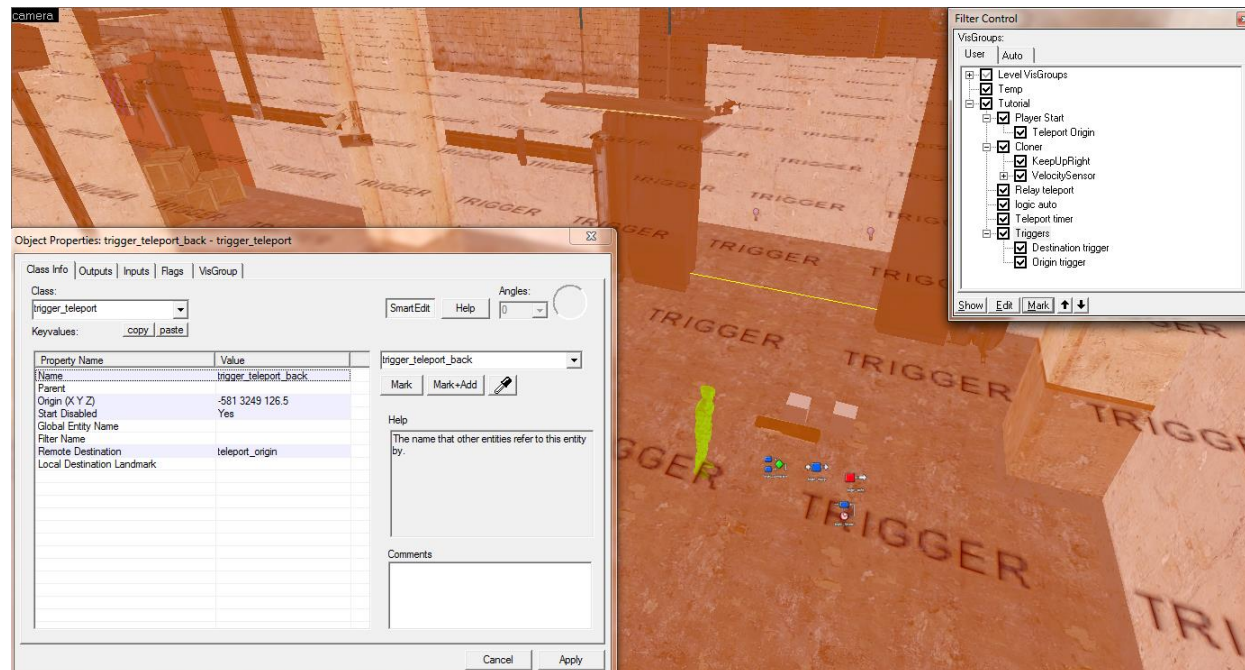
- Change the following properties:

teleport_timer		
Property	Value	What it does
Name	teleport_timer	

Start Disabled	Yes	
Refire Interval	5	Time before the player goes back to his original position, after he has teleported

4. The last thing we need to create is a **trigger_teleport**. Like the previous trigger, we need it to cover the whole room (or the area where we want to use the teleport)

- Create a **trigger_teleport**



- Change the following properties:

trigger_teleport_back		
Property	Value	What it does
Name	trigger_teleport_back	
Start Disabled	Yes	Like the other trigger, this trigger is going to be disabled most of the time
Remote destination	teleport_origin	

- Make sure that the trigger only collides with the player in the **flags tab**

trigger_teleport_back (Flags tab)		
Flag	Value	What it does
Clients	Checked	
All other flags	Unchecked	

Setting the logic for teleporting back

1. This is what we will do for teleporting back:








2. To start we will need to add some outputs to the relay_teleport_to_cloner

- The first thing we need to do is to disconnect the **teleport_origin** from the player, right before we teleport to the Cloner. That way, the teleport_origin stays at our original position, and we can go back to it later. We also want to activate the **teleport_timer** at this point.
- For that, we add the following outputs to the **relay_teleport_to_cloner**:

relay_teleport_cloner (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTrigger	teleport_origin	ClearParent		0.00	No	Drops the teleport_origin right before we teleport to the Cloner, so that we can go back to it later
OnTrigger	teleport_timer	Enable		0.00	No	

Object Properties: relay_teleport_to_cloner - logic_relay

Class Info | **Outputs** | Inputs | Flags | VisGroup

	My Out...	Target Entity	Target In...	Delay	Only Once
	OnTrigger	teleport_origin	ClearParent	0.00	No
	OnTrigger	teleport_timer	Enable	0.00	No
	OnTrigger	trigger_teleport_to_cloner	Enable	0.03	No
	OnTrigger	trigger_teleport_to_cloner	Disable	0.04	No
	OnTrigger	relay_teleport_to_cloner	Disable	0.05	No

My output named Targets entities named Via this input With a parameter override of After a delay in seconds of ☐ Fire once only

Mark

Add...

Copy

Paste

Delete

☐ Show Hidden Targets As Broken

Cancel

Apply





3. After the **teleport_timer** runs out, we need to send the player back to the origin.

- We add the following outputs to the timer:


teleport_timer (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTimer	trigger_teleport_back	Enable		0.03	No	First, we want to activate the trigger that will send us back to our original position.
OnTimer	trigger_teleport_back	Disable		0.04	No	After that, we will disable the trigger, since the teleport is done
OnTimer	teleport_origin	SetParent	!player	0.05	No	Once the player has teleported back, we parent the teleport_origin to the player again.
OnTimer	teleport_timer	Disable		0.06	No	Finally, we disable the timer. Otherwise the timer will start over and the player will keep teleporting to the Cloner's position.

Object Properties: teleport_timer - logic_timer


Class Info | Outputs | Inputs | Flags | VisGroup

	My Ou...	Target Entity	Target I...	Para...	Delay	Only Once
	OnTimer	trigger_teleport_back	Enable		0.03	No
	OnTimer	trigger_teleport_back	Disable		0.04	No
	OnTimer	teleport_origin	SetParent	!player	0.05	No
	OnTimer	teleport_timer	Disable		0.06	No

My output named

Targets entities named 

Via this input

With a parameter override of 

After a delay in seconds of ☐ Fire once only

Mark

Add...

Copy

Paste

Delete

☐ Show Hidden Targets As Broken

Cancel

Apply

(Optional) Step 3: Finishing the Cloner = spawning dummies

In this section, we are going to do two things:

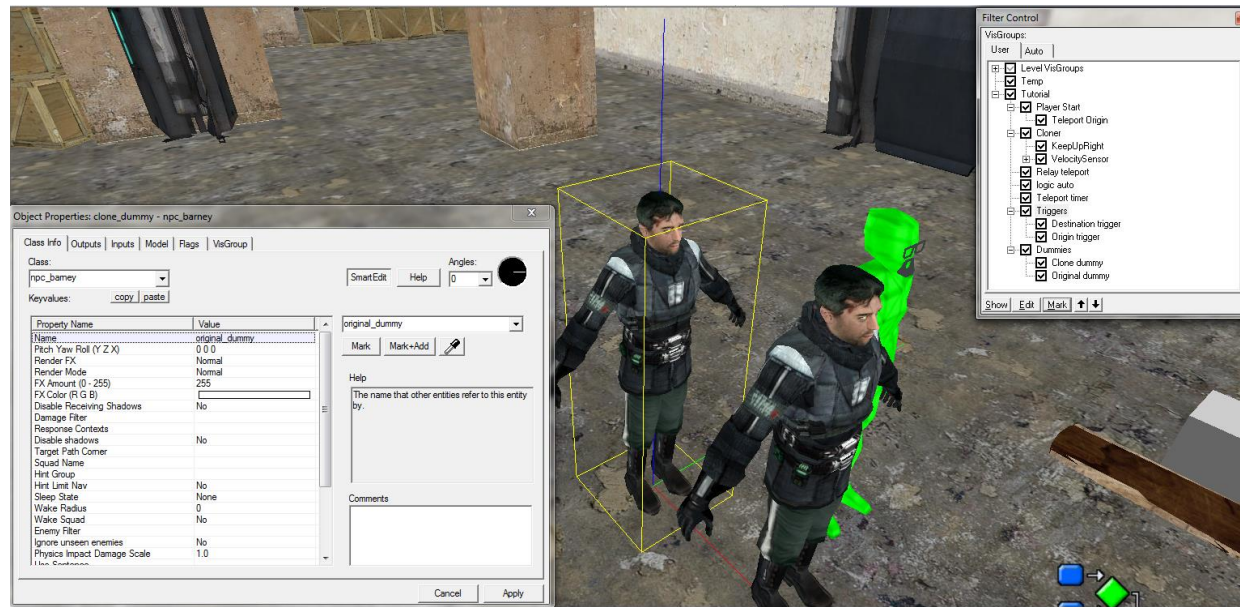
- Spawn a dummy after you teleport to the cloner. This dummy represents your "original body".
- Spawn a dummy after you teleport back. This dummy represents your "clone".

Keep in mind that for this, we will need an npc model. Since Gordon Freeman does not have a model, your level's character should be someone else (for my level I chose **Barney**)

Creating the spawners

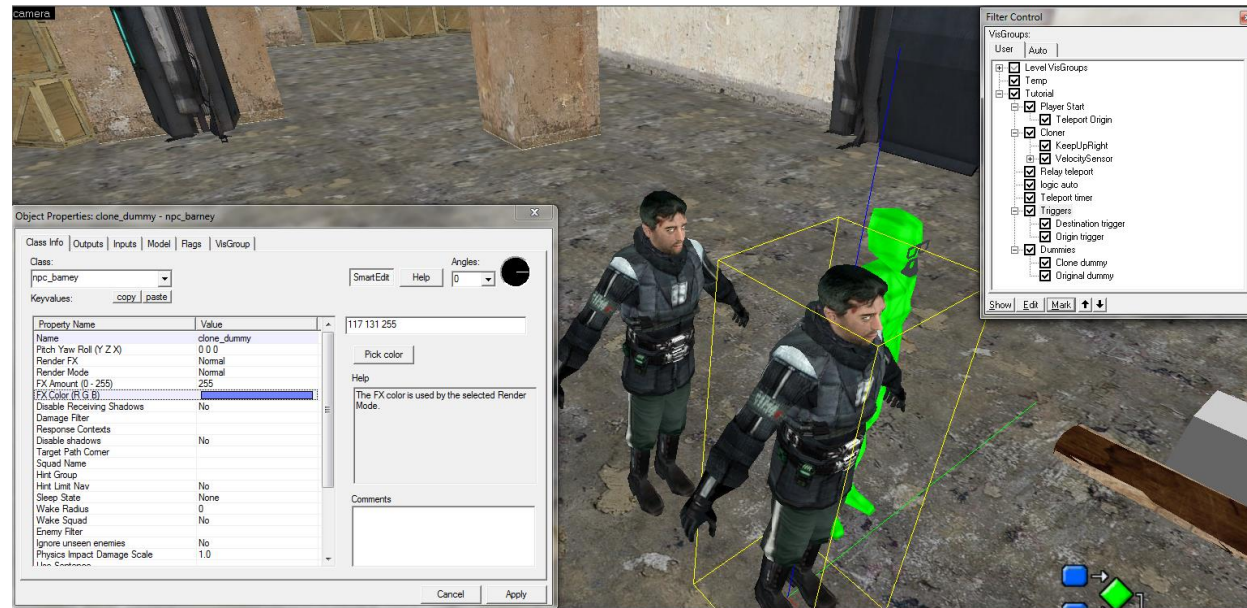
1. First thing we need are the dummies (**npc_barneys**)

- First we create one **npc_barney** for the original body



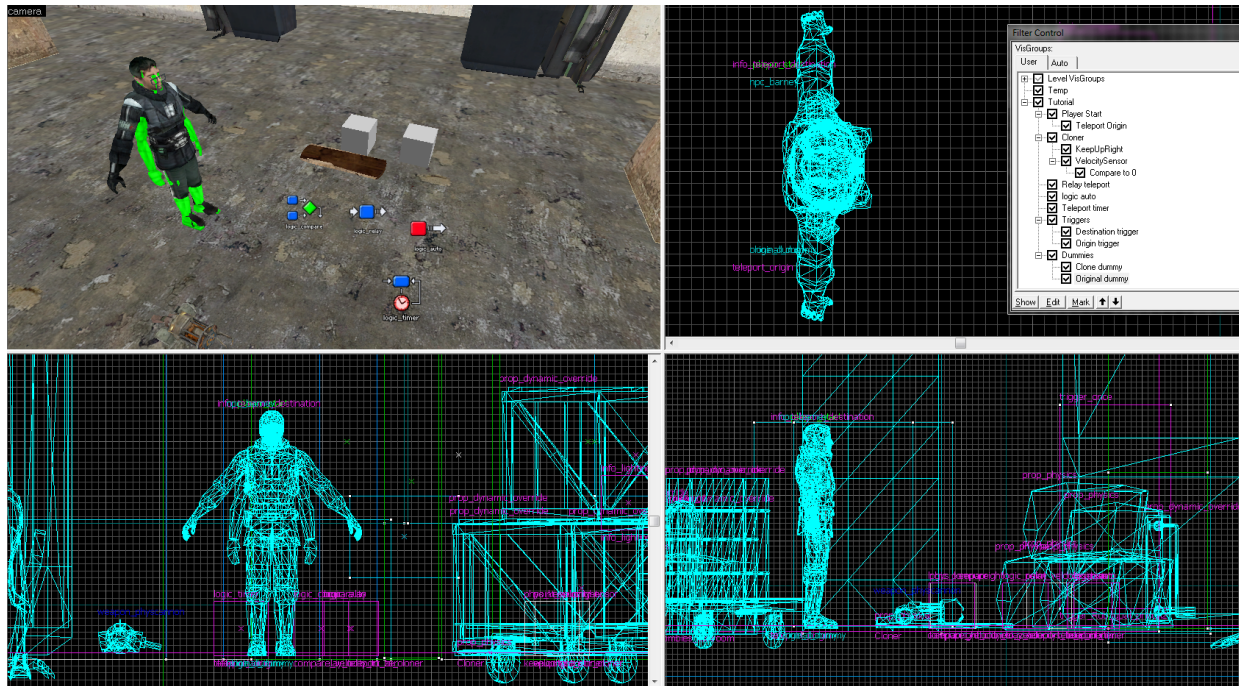
original_dummy	
Property	Value
Name	original_dummy

- Then we create another **npc_barney**, this time for the Clone

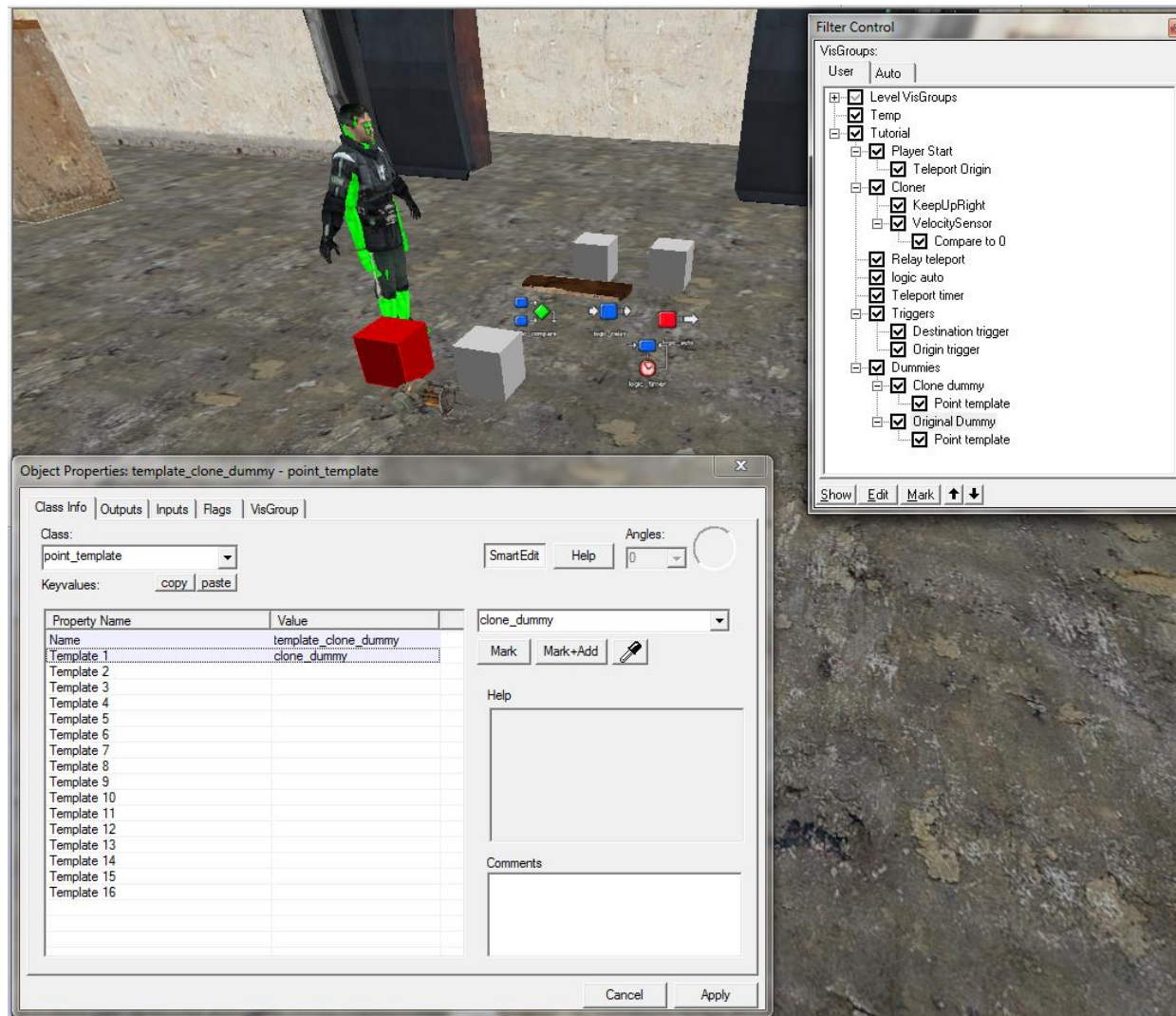


clone_dummy		
Property	Value	What it does
Name	clone_dummy	
FX Color	a color of your choosing	Changing the FX Color will let us differentiate between the original body and the clone body. This property is optional.

- Like we did with the teleport_origin, we need to make sure the dummies are in the exact same position as the **info_player_start**



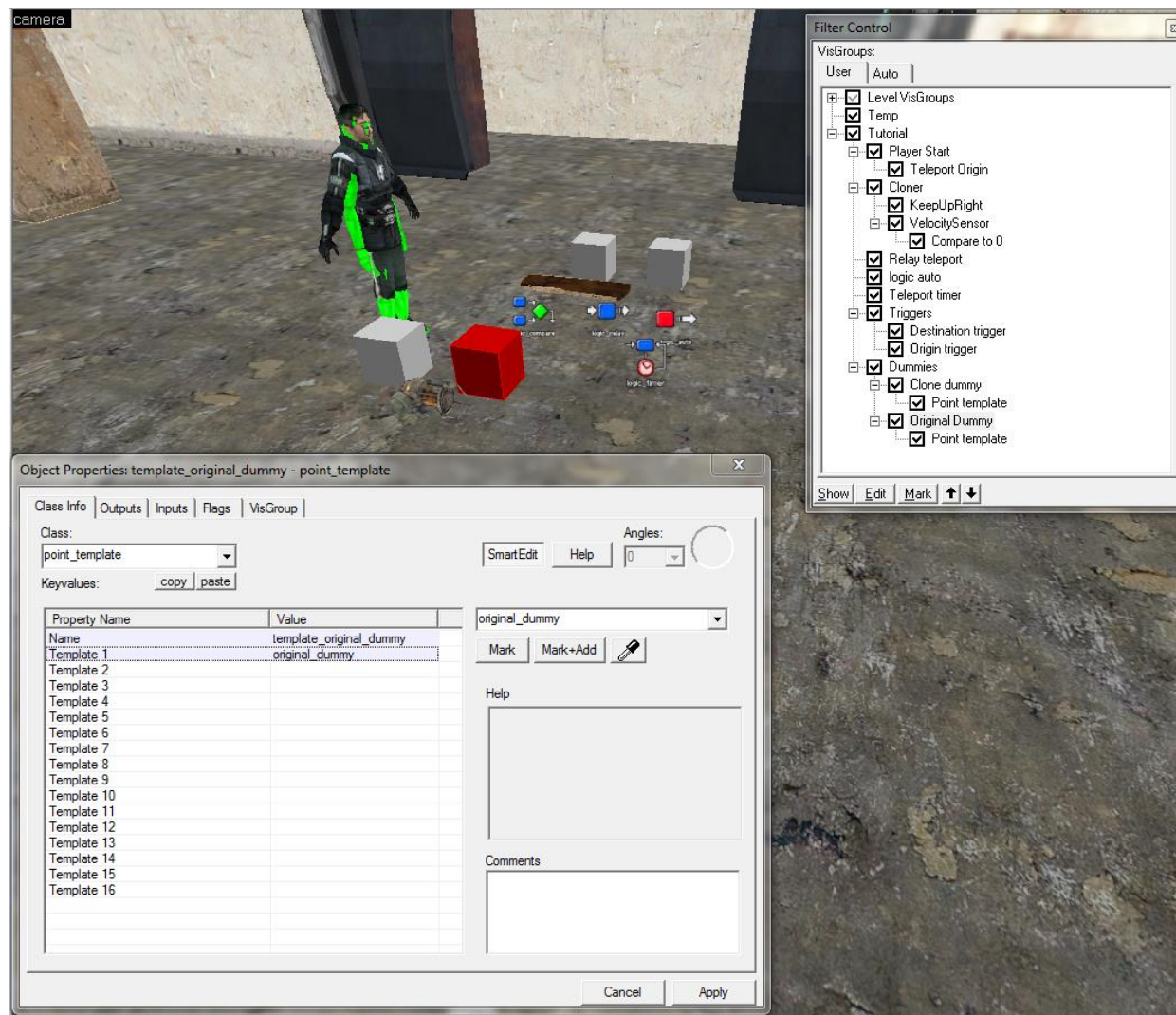
2. Next, we are going to create two **point_templates**. Point_templates are necessary when we want to spawn an object,
- We create one point_template for clone_dummy



template_clone_dummy

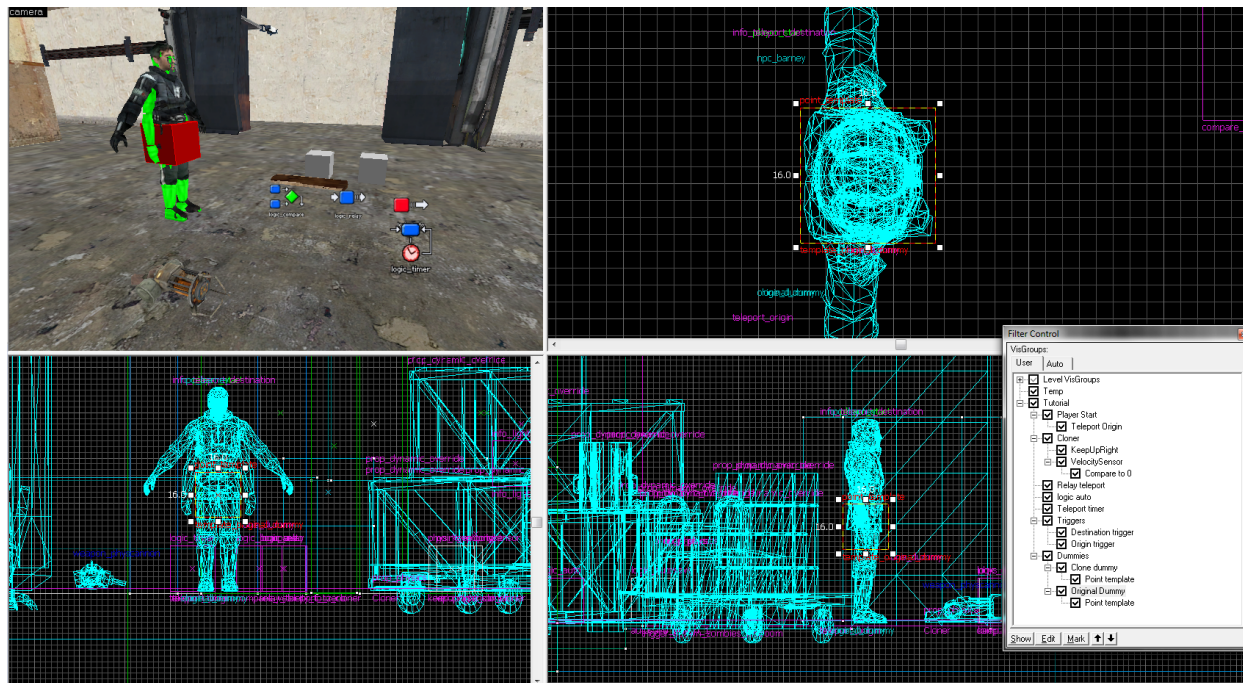
Property	Value	What it does
Name	template_clone_dummy	
Template 1	clone_dummy	We can add more objects to the other template slots, and these objects will spawn along with the dummy.

- And one point_template for the original_dummy

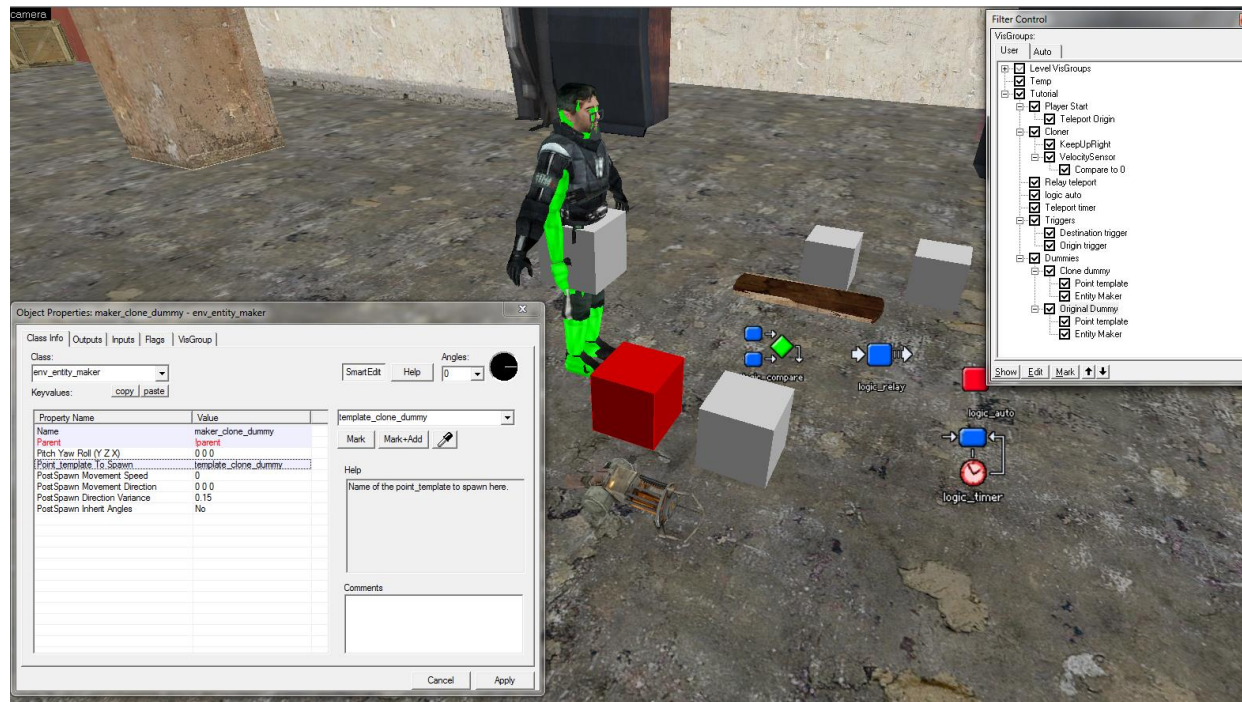


template_clone_dummy	
Property	Value
Name	template_original_dummy
Template 1	original_dummy

- Finish by putting these two boxes in the same position as the npc_barneys and the info_player_start



3. The last step is to create the **env_entity_makers**. These entities are the ones in charge of spawning the dummies. They will spawn whatever is in a point_template of our choosing.
 - Create one **env_entity_maker** for the clone dummy

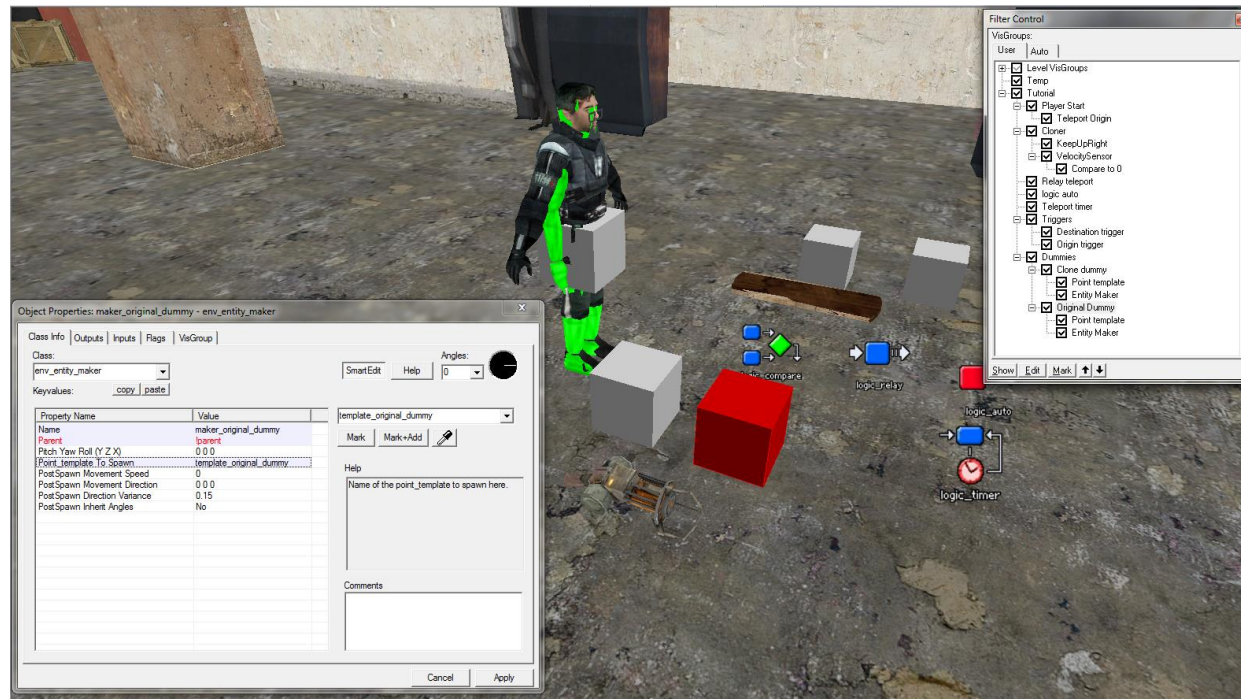


maker_clone_dummy

Property	Value	What it does
Name	maker_clone_dummy	
Parent	!player	We want to spawn the clone dummies in the same position as the player, before he goes back to his original position. Because of that, we want the entity maker to follow the player.

Point template To Spawn	template_clone_dummy	
-------------------------	----------------------	--

- Create one **env_entity_maker** for the original dummy



maker_original_dummy

Property	Value	What it does
Name	maker_original_dummy	

Parent	!player	We want to spawn the original dummies in the same position as the player, as he is about to teleport. Because of that, we want the entity maker to follow the player.
Point template To Spawn	template_original_dummy	

- To make sure that the entities get parented correctly, we add the following outputs to the logic_auto:

logic_auto (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnMapSpawn	maker_clone_dummy	SetParent	!player	0.00	No	

Object Properties: logic_auto

Class Info | **Outputs** | Inputs | Flags | VisGroup

	My Output >	Target Entity	Target I...	Para...	Delay	Only Once
	OnMapSpawn	teleport_origin	SetParent	!player	0.00	No
	OnMapSpawn	maker_clone_dummy	SetParent	!player	0.00	No
	OnMapSpawn	maker_original_dummy	SetParent	!player	0.00	No

My output named: OnMapSpawn

Targets entities named: teleport_origin

Via this input: SetParent

With a parameter override of: !player

After a delay in seconds of: 0.00 ☐ Fire once only

Mark Add... Copy Paste Delete ☐ Show Hidden Targets As Broken

Cancel Apply

OnMapSpawn	maker_original_dummy	SetParent	!player	0.00	No	
------------	----------------------	-----------	---------	------	----	--

- Like we did with the point_templates, move the env_entity_makers to the same position as the info_player_start







Spawning/despawning the original body

1. The original body will spawn right before we teleport to the Cloner. For that, we will use the **relay_teleport_to_cloner**:


relay_teleport_to_cloner (Outputs)						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTrigger	maker_original_dummy	ForceSpawn		0.00	No	We need to spawn it before the player teleports, since the env_entity_maker is going to move with the player.

Object Properties: relay_teleport_to_cloner - logic_relay


Class Info | Outputs | Inputs | Flags | VisGroup

	My Out...	Target Entity	Target Input	Delay	Only Once
	OnTrigger	teleport_origin	ClearParent	0.00	No
	OnTrigger	teleport_timer	Enable	0.00	No
	OnTrigger	maker_original_dummy	ForceSpawn	0.00	No
	OnTrigger	trigger_teleport_to_cloner	Enable	0.03	No
	OnTrigger	trigger_teleport_to_cloner	Disable	0.04	No
	OnTrigger	relay_teleport_to_cloner	Disable	0.05	No

My output named

Targets entities named 

Via this input

With a parameter override of 

After a delay in seconds of ☐ Fire once only

Mark

Add...

Copy

Paste

Delete

☐ Show Hidden Targets As Broken

Cancel

Apply

2. After we teleport, and the teleport timer runs out, we will go back to our original position. That is when we will kill the original_dummy. In order to do that, we add the following output to the **teleport_timer**.

teleport_timer						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTimer	original_dummy*	Kill		0.00	No	original_dummy* means that this output will kill any entity in our level whose name starts with "original_dummy". We write it this way as a precaution, since sometimes the point_templates change the names of the entities they spawn (based on the value of one of their flags)

Object Properties: teleport_timer - logic_timer

Class Info

Outputs

Inputs

Flags

VisGroup

	My Ou...	Target Entity	Target I...	Para...	Delay	Only Once
	OnTimer	original_dummy*	Kill		0.00	No
	OnTimer	trigger_teleport_back	Enable		0.03	No
	OnTimer	trigger_teleport_back	Disable		0.04	No
	OnTimer	teleport_origin	SetParent	!player	0.05	No
	OnTimer	teleport_timer	Disable		0.06	No

My output named

OnTimer

Targets entities named

trigger_teleport_back

Via this input

Enable

With a parameter override of

<none>

After a delay in seconds of

0.03

☐ Fire once only

Mark

Add...

Copy

Paste

Delete

☐ Show Hidden Targets As Broken

Cancel

Apply

Spawning the clone

1. Before we go back to our original body, we are going to spawn a clone dummy. We can kill this clone dummy based on any conditions we want (we could use it as a distraction for enemies, for example, and destroy it after a timer runs out). I will not cover the killing of the clone dummy, but it is as easy as it was killing the original dummy (3.3)

2. We will put the logic to spawn the original dummy in the **teleport_timer**. As we did with the original_dummy, we will use the env_entity_maker for this.

teleport_timer						
My Output	Target Entity	Target Input	Parameter	Delay	Only Once	What it does
OnTimer	maker_clone_dummy	ForceSpawn	0.00	No		

