

## How to use an action?

An action is just a function attached to an entity: the player, an elevator, a monster, a weapon, etc. You can attach actions to any entity: model, sprite, wmb entity or terrain. They are a very convenient way to control the entity behavior without having to program a complicated 'game loop'. Here is a small example:

```
action hereiam()
{
set(my,SHADOW); /////« Give the model shadow.
}
```

If you run the level the model will show but doesn't do anything except create a shadow. Basically the first parts handles all the stuff the model needs to be shown (or not) and uses stuff that need to be done only once and stay this way until it has to do something else. Ok while the model is standing there we will let it rotate until we say it needs to do something else.

```
action hereiam()
{
set(my,SHADOW); /////« Give it shadow.
while(1) ///< so the model is shown and during this it will do the next thing here the while(1) is always used.
{
my.pan += time_step; /////« This will make the model rotate use – to rotate the other way
wait(1); ///< « The wait is needed it makes sure to continue this action till something else is needed.
}
}
```

Wow so i placed a model give it shadow and make it rotate. But say i want it to move on the x,y,or z axis ? Well we simply add one line of code, Do you see which one?

```
action hereiam()
{
set(my,SHADOW); /////« Give it shadow.
while(1) ///< so the model is shown and during this it will do the next thing {
{
my.pan += time_step; /////« This will make the model rotate at its position.
my.x += 0.1; ///< move on the x axis very slow during the rotation of the model
wait(1); ///< « The wait is needed it makes sure to continue this action till something else is needed.
}
}
```

If you want it to move the other way on the x axis just use a - instead of a plus. Try this out Make it move on the y or z axis to. Experiment to understand how this works. See if you also Can make the model rotate the other way around you should understand now how ☺ We will use this action in the next lessons so we learn how to expand from this point.

### How to use a camera view and what is a pointer?

Ok so in our action we make the model move but we lose it once it's off the screen. We want a camera to be aimed on the model so we can keep seeing the model move. First we need to give the model something to be recognized by. Above the action we add an extra line:

```
ENTITY* viper; «< we tell the engine we are using a model using the name viper
```

```
action hereiam()
{
viper = me; ///<« we say here that the viper is the model to use when we call this pointer in other functions or
            actions
set(my,SHADOW);///<« Give it shadow.
while(1) ///<« so the model is shown and during this it will do the next thing
{
my.pan -= time_step; ///<« This will make the model rotate at its position.
my.x += 0.01; ///<« move on the x axis very slow during the rotation of the model the higher the number the
faster it goes.
wait(1); ///< « The wait is needed it makes sure to continue this action till something else is needed.
}
```

What we did is using a so called pointer ☺ we made the action and model unique.

Now we will add a simple camera code so we will keep seeing the model no matter where it goes to

```
ENTITY* viper;
```

```
action hereiam()
{
set(my,SHADOW);///<« Give it shadow.
VECTOR temp; ///< a point where the camera will fix its view on viper = me; ///<< use a pointer to make this
model unique set
while(1) ///<< so the model is shown and during this it will do the next thing {
{
my.pan -= time_step; ///<« This will make the model rotate at its position.
my.x += 0.01; ///<« move on the x axis very slow during the rotation of the model the higher the number the
faster it goes.
camera.x = my.x-200;///<« " Distance camera x axis from the model
camera.y = my.y; ///<« We do nothing with the y axis unless you want to
camera.z = my.z + 150; ///< " Distance height z axis from the model
camera.tilt = -45; ///< "tilt the camera down a little so it's aimed to the floor and model
wait(1); ///< " The wait is needed it makes sure to continue this action till something else is needed.
}
}
```

Run this and you will see that the camera is fixating to the model and stays with it.

Change some numbers of the camera x,y,z, axis to experiment with different views.

Amazing job we learned to create an action, make an object move, give it shadow and use a pointer and a camera view. Not bad in such a short time.

### **What are variables and how to use them?**

A variable is a place in your computer's memory (just like a container) that can be used to store numbers. It could be used for scoring, ammo, health but also for timers or to set a function on or off.

I will use a way different than the original explanation on the tutorial site. This way you can use the code in any of your own stuff as it looks cool to ☺

Let's say we will have an ammo number on screen. This is how we do it:

```
var ammo_count = 0; /// the name of our variable where we store the ammo number and we set it on 0 at start.
```

```
FONT* fnt_pan = "Times#40b"; /// The true type font we want to use size and bold
PANEL* pan_ammo_count = {digits=10,10,"AMMO: %00.0f",fnt_pan,1,ammo_count; ///the panel
                        where the number is shown on 10-10 is position x and y axis.
layer = 20; ///the layer of the panel
flags = SHOW;green=255; blue=0; red=255;/// the color of the font that is used. This is yellow.
}
```

Place this code right before the main function and test it. If all went right you will see In the left upper corner in yellow letters: ammo : 0

You can change the font and the color of the font easy here are some handy color choices you can test:

```
green=255; blue=255; red=255;    = White
green=255; blue=0; red=255;      = Yellow
green=255; blue=0; red=0;        = Green
green=0; blue=255; red=0;        = Blue
```

test this in your script and see if you can change the position of the variable, the font and the colors. Add a new variable to the script and name it what you want, Experiment as you will use it a lot later.

### **Pick something up and add or subtract from a variable**

Ok Realspawn this is fun and all but in games we see always that characters are picking up stuff like extra lives, ammo or health boxes and so on. How can we do this? Well we will simply make an action for that shall we? here we go :

Place this action under the player action.

```
action pickup_ammo()
{
set(my,SHADOW | PASSABLE);/// set the model passable and give it shadow
wait(1); ///wait one frame
c_setminmax(my);/// set proper collision BBOX
while(!viper) ///wait till the player is created the hereiam action
{
wait(1);/// wait one frame
}
while(vec_dist(my.x,viper.x) >50)/// the distance between the viper and the pickup object
{
wait(1); wait one frame
}
ammo_count +=10;/// yes we ad 10 points to our variable. Use - if you want to subtract.
ent_remove(my);/// we remove the pickup model
}
```

Now place some models on the path of our viper and give it this pickup ammo action. Run the level and see what you made.

That's right! It picks up the ammo models and each time 10 points are added to the variable we created. That's so cool ☺ A lot is done in a short time shall we continue since we are on a roll?

### Playing sound files for sound fx

Well i got this brilliant idea. When the ammo is picked up let's hear a sound ☺ every sound we use (wav mp3 ogg) must be defined first in order to be able to be played I will use a sound file named ammo.wav

Place this line above the main function:

```
SOUND* ammo_snd = "ammo.wav";/// this defines a sound we will use
```

Ok we defined the sound to use so now we need to add a line to play it, and where better than in the pickup action we created.

Add this line : `snd_play (ammo_snd, 100, 0);///«< play the sound file we defined`

So you should have this action now:

```
action pickup_ammo()
{
set(my,SHADOW | PASSABLE);// set the model passable and give it shadow
wait(1); //wait one frame
c_setminmax(my);// set proper collision BBOX
while(!viper) //wait till the player is created the hereiam action
{
wait(1);// wait one frame
}
while(vec_dist(my.x,viper.x) >50)// the distance between the viper and the pickup object
{
wait(1);/// wait one frame
}
ammo_count +=10;// yes we added 10 points to our variable. Use - if you want to subtract.
snd_play (ammo_snd, 100, 0);///"«< play the sound file we defined
ent_remove(my);// we remove the pickup model
}
```

Run the level and there you have it ☺ each time an ammo model is picked up you hear the sound play this is amazing ☺ i love it! I know this is not a full game but admit it is a cool feeling to create and make stuff come to life. So you have learned a lot now in this part. Try this stuff out. Make new pick up actions with different variables and make yourself familiar with this stuff. It is the start of perhaps a cool game project of your own. You could make the pick-up models rotate as you know how to do that right?

Next chapter we will expand, we will do this step by step and i hope you enjoy it as much as i do. See you next time.

Rene Pol aka Realspawn.  
[Realspawn@live.nl](mailto:Realspawn@live.nl)