

# TUBE CLEANER

A SIMPLE SHOOTING GAME

Carsten Wartmann



Tube Cleaner was designed by Freid Lachnowicz. The game is a simple shooter game that takes place in a tube. Three kinds of enemies are present. Try to collect as many points and bullets as you can on your way up the tube!

To play the game in its final stage load the scene `Games/TubeCleaner.blend` from the CDROM. It includes instructions.

This Tutorial is not supposed to explain how to make the full game. But you should be able to understand the extensions in the final result (you can also look for cheat

codes in the game...) with the help of this book and this tutorial. And of course you are encouraged to change and extend the game to your liking!

Tube Cleaner game controls

CONTROLS	DESCRIPTION
CURSOR KEYS	rotate the cannon left, right, up and down
SPACE	Shoot

## Loading the models

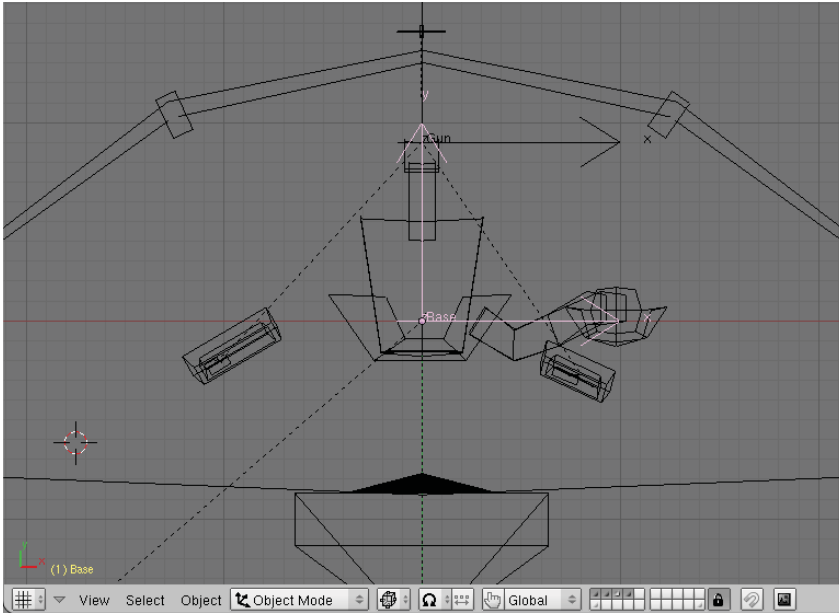
Tube Cleaner game



Start Blender and load `Tutorials/TubeCleaner/TubeCleaner_00.blend` from the disk. This scene contains all models to start. To make the scene interactive, this tutorial will lead you through the following tasks:

1. Adding game logic to the gun, allowing it to move up, turn and shoot
2. Adding game logic to the enemies
3. Creating the score system including display

The scene contains a camera view on the left, a wireframe view (view from top) on the right and a the Logic Buttons on the bottom. In the top view (Figure "Wireframe top view in the Tube Cleaner scene") you can see the "Base" object is already selected and active (purple color in wireframe). The "Base"-object will carry the cannon



Wireframe top view  
in the Tube Cleaner  
scene

and will contain some of the global logic of the game. The cannon itself is parented to this "Base". This hierarchy will make our job later easier because we won't have to worry about composite movements.

## Controls for the base and cannon

We start with the rotation around the vertical axis of the base. This will also rotate the cannon and the camera because they are parented to the base.



Logic Bricks to  
rotate the gun

Make sure that the "Base" object is selected (pink, **RMB** to select if not) and click on the "Add" Buttons in the Logic Buttons **F4** for each row of Sensors, Controllers and

Actuators. In every row a new Logic Brick will appear.

Now link (wire) the Logic Bricks by clicking and drawing a line from the little yellow balls (outputs) to the yellow donuts (input) of the Logic Bricks. These connections will pass the information between the Logic Bricks. Change the first Logic Brick to a Keyboard Sensor by click and hold its Menu Button with the left mouse button and select "Keyboard" from the pop up menu.

*Please (re)do the tutorial in the "Pumpkin Run" chapter if you have problems with the creating, changing and linking of Logic Bricks.*

Now, click the "Key" field with the **LMB** and press the **RIGHT** key when prompted by "Press any key" in the Keyboard Sensor. The "Key" field now displays "Rightarrow" and the Keyboard Sensor now reacts to this key only.

Now change the third number (Z-Axis) in the "dRot" row of the Motion Actuator to -0.03. Do this by using **SHIFT-LMB** on the number and entering the value with the keyboard. The three fields always denote the three axis (X,Y,Z) of an object. So we will rotate around the Z-axis.

Now move your mouse cursor over the camera view and press **P** to start the game. You should now be able to rotate the gun with the right cursor key.

*You should always name your Logic Bricks and other newly created elements in your scenes (click on the default name and edit with your keyboard). This will help you to find and understand the logic later. Take Figure "Logic Bricks to rotate the gun" as a reference.*

Use the same procedure as above to add Logic Bricks to rotate the gun to the left. Use **LEFT** as key in the Keyboard Sensor and enter "0.03" in the third "dLoc" field of the Motion Actuator.

As you can see the space in the Logic Buttons is getting sparse with only six Logic Bricks. Use the little orange arrow icon in the Logic Bricks to collapse the Logic Bricks to just showing a title. Now you also see another good reason for properly naming Logic Bricks.

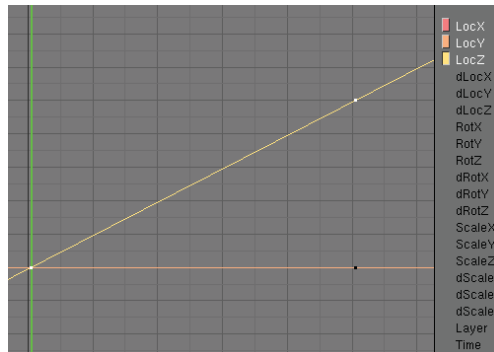
## Upwards Movement

In "Tube Cleaner" we want to have a continuous upwards movement within the tube. We could achieve this similarly to the rotation of the gun, but there is another way which will give us much more control over the movement and also allows you to move down to a specific level of the tube.

The method used here is to combine the possibilities of Blender's game engine and its powerful animation system. Move your mouse cursor over the camera view and press **ALT-A**. Blender will play back the animations already defined in the scene. Press **ESC** to stop the playback. But so far none of these animations is played back by the game engine. We have to tell the objects to play the animation. This way we can interactively control animations, for example play, stop or suspend.

Move your mouse cursor over the wireframe view and press **SHIFT-F6**. The window will change to an Ipo Window (see the Ipo Figure below), meant for displaying and editing Blender's animation curves. The Ipo Window is organized into axes, here the axes are the horizontal axis, showing the time in Blender's animation frames and the vertical axis showing Blender units. The yellow vertical line is the animation curve for the movement along the Z-Axis of the "Base" object, meaning upward movement for our object. So to move the object 10 units up you can move the Ipo cursor (green line) with a left mouse click to frame 10. The camera view will reflect this immediately.

To play this animation in the game engine, we use a special Logic Brick the "Ipo Actuator", set to "Property play" type. A Property is a variable owned by a game object, which can store values or strings. We will now create a new property which will hold the height (zLoc(ation)) of the "Base" object. To do so click on "ADD property" in the Logic Buttons for the "Base" object.



Ipo Window with the animation curve for upward movement

Click on "Name:" and change the default name "prop" to "zLoc", this Property will hold the height of the gun in the tube.

*Blender uses capitalization to distinguish between names of Objects and Properties. So "zloc" is not the same as "zLoc".*

Continue creating the Logic Bricks from Figure "Logic Bricks for the upward movement". The Always Sensor triggers the logic for every frame of the game engine animation, ensuring constant movement. The AND Controller just passes the pulses to two Actuators, both are connected to the Controller and will be activated at the same frame.

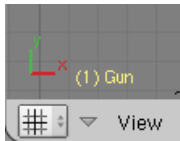


Logic Bricks for the upward movement

The Ipo Actuator will play the Ipo animation according to the value in the Property "zLoc". To get a constant motion we increase the "zLoc" Property every frame with the Property Actuator . In this case it is of the type "Add" which adds "Value:" (here 0.01) to the "zLoc" Property every frame. Try to change the "Value:" to get a feeling for the speed of the animation. If you'd like to move the cannon downwards try entering -0.01 into the "Value:" field. After experimenting a bit please use 0.01 for the value field as shown in the Figure. To play the game at this stage move your mouse to the camera view and press **P**.

*Blender can show you the used Properties and their values while the game runs. To do so, choose "Show debug properties" from the "Game" menu and activate the "D" Buttons (Debug) for every Property you'd like to have printed on screen*

## Shooting



Switch the Ipo Window back to a 3D View by pressing **SHIFT-F5** over the Ipo Window. Now select the "Gun" object with the right mouse. You can click on every wire from the "Gun" object, a proper selection will be reflected in the lower left corner of the 3D View and in the Logic Buttons where "Gun" will appear in the columns for the Logic Bricks.

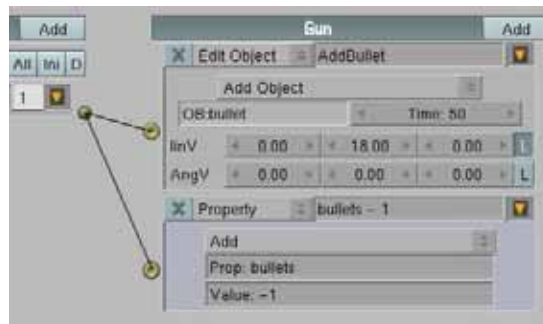
Now add a Sensor, Controller and Actuator to the "Gun" object and wire them as you learned earlier in the tutorials. Change the Sensor to a Keyboard Sensor (please name the Sensor "Space") and choose (click in the "Key" field) Space as trigger for the gun. Change the Actuator to an "Edit Object" Actuator. The default type is "Add Object" which adds an object dynamically when the Actuator is triggered.



Logic Bricks to fire the gun

Enter "bullet" into the "OB:" field by clicking it with **LMB** and using the keyboard to enter the name. This will add the object "bullet", a pre-made object, which is on a hidden layer in the scene. Enter 18.00 as the second number in the "linV" fields. This will give the bullet an initial speed. Also activate (press) the little "L" button behind the "linV" row. This way the bullets will always leave the gun in the direction aimed. Enter 50 in the "Time:" field. This will limit the life time of the bullets to 50 frames, avoiding ricochets to bounce forever. As usual, try to run the game now and shoot a bit.

So far we have unlimited ammunition. To change this we again use a Property storing the number of bullets left. Add a new Property by clicking on "ADD property". Name this Property "bullets" and change its type to "Int" with the Menu Button now labeled "Float" (the standard type for new Properties). An "Int(eger)" Property



only holds whole numbers, this is ideal for our bullets as we don't want half bullets. **SHIFT-LMB** on the field to the right of the "Name:" field to enter 10 here. This is the initial number of bullets available at the start of the game.

To actually decrease the number of bullets on every shot we use the "Property Add" Actuator that we also used to make the base of the cannon move up. So add another Actuator by clicking on "Add" in the Actuator's column of the gun. Wire it to the AND Controller we created in the last step. Change the Actuator type to "Property" choose "Add" as the action. Enter "bullets" in the "Prop:" field and -1 in the "Value:" field. This will subtract 1 (or add -1) from the Property "bullets" on every shot triggered by Space.

So far the gun doesn't take any notice of the number of bullets. To change this we will use an Expression Controller which allows to add single line expressions to process game logic. Change the AND Controller with the Menu Button to an Expression Controller. Then click on the "Exp:" field and enter "Space AND bullets>0" (without the quotation marks) and press **ENTER**. Here "Space" is the name (exactly as you typed it in the Logic Brick) of the Keyboard Sensor and "bullets" is the bullets Property. The Controller now only activates the following Actuators if the Sensor "Space" is active (meaning that Space is pressed) AND bullets is bigger than zero. Try again to run the



game, you now can only shoot 10 times. Read more about Expressions in Section [Expressions].

The last step to make the gun work is to make the bullets display functional. Select the display (the right one with the flash on it) with the right mouse button. It is best to hit the little dot on the "@" sign (which is a visual hint for placing the object). The name "BulletsDisplay" should appear in the Logic Buttons and in the 3D View. Alternatively you can zoom into the wireframe view to make the selection easier.

You can see in the Logic Buttons that there is already a Property called "Text" for the display object. The object has a special text-texture assigned which will display any information in the Property called "Text" that is on it. You can test this by changing the value "10" in the Property, the change will be displayed immediately (a bit covered by the "@" sign) in the camera view and also during runtime of the game.

Because Properties are local to the object they are owned by, we have to find a way to pass the values of Properties between objects. Inside a single scene this can be done with the Property Copy Actuator. Another way would be to use messages, in this case you can also pass information between multiple scenes.

Add a line of Logic Bricks to the "BulletsDisplay" like you did before and wire them. Change the Actuator to a Property Actuator, type "Copy".




Logic Bricks to display the amount of bullets

Enter "Text" into the first "Prop:" field, this is the name of the Property we copy into. Enter "Gun" into the "OB:" field, again watch for correct capitalization, Blender will blank the input field if you enter a non-existing object. From this object we will get the value. Enter "bullets" into the field "Prop:" beneath "OB:" this is the Property name from which we get the value. It is not needed to actually copy the information in every frame, so enter a 10 in the "f:" field of the Always Sensor, this will copy the Property every 10th frame (about 1/6 of a second) and will so preserve performance.

Start the game and shoot until you have no more bullets.

## More control for the gun

Tilting the gun will add more freedom in movement and dynamic to the game. We will use a similar technique as for the movement up, by combining animation curves and Logic Bricks.

Select the gun and collapse the Logic Bricks by clicking their arrow icons , this will give us more space for the coming logic.

As for the upward movement of the gun, it already contains a motion curve that we can use. We also need a Property which contains the actual rotation (tilt, rotation around x-axis). So add a new Property by clicking on "ADD property" and name it "rotgun".

Again use the "Add" Buttons to add a Sensor, Controller and one... nope, you are right this time two Actuators. You already know this from the upward movement. We need one Actuator to change the Property and one to play the lpo. Wire the new Logic Bricks, as shown in Figure "Logic Bricks to rotate the gun upwards". The collapsed Logic Bricks are the ones you made for shooting.

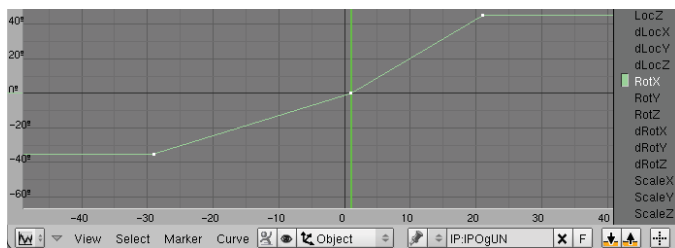


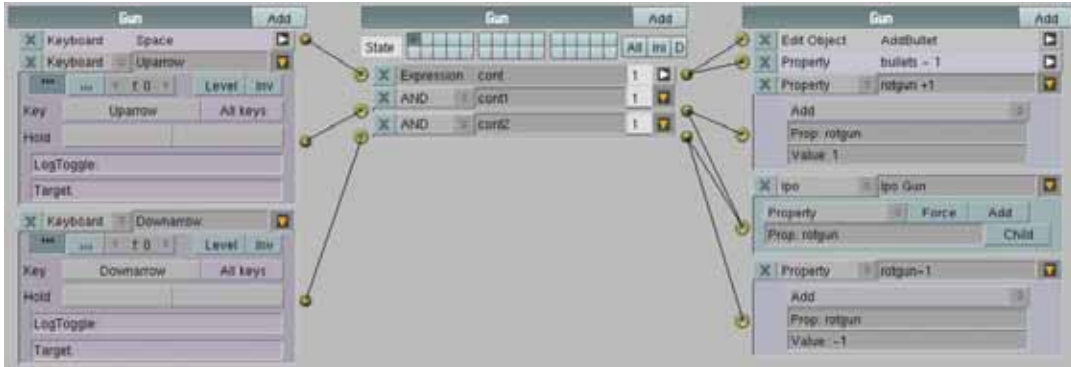
Logic Bricks to rotate the gun upwards

Change the Bricks according to the Figure above and enter all the necessary information. Using the Property Add Actuator we increase "rotgun" by one every time **UP** is pressed and then play the lpo according to the "rotgun" Property, this will rotate the gun up.

Now test the game. Every time you press **UP** the gun will rotate a bit up. Stop the game with **ESC**. To get a continuous motion we have to activate the pulse mode icon for the Keyboard Sensor, this will give a keyboard repeat here, so the gun will rotate as long as you press the key without the need to release it.

Now test the rotation again, and you will see that the gun only rotates a specific amount and then stops. This is because of the animation curve (lpo), you can visualize the curve when you switch a window to an lpo Window using **SHIFT - F6** (use **SHIFT - F5** to return to the 3D View). You can see that the curves are horizontally from frame 21 (horizontal axis), meaning no further rotation is possible. You also see that we need to make the "rotgun" negative to rotate down.





Completed Logic

Bricks to tilt the gun. Again add a Sensor, Controller and one (yes, this time, it's really only one) Actuator. Wire and name them as shown in Figure "Completed Logic Bricks to tilt the gun". Note that we use the Ipo Actuator for the tilting down also. This is perfectly ok, this way we can save a Logic Brick. It would have also been working ok to use a second Ipo Actuator here.

*If you prefer "pilot-controls" just swap the UpArrow and DownArrow in the Keyboard Sensors.*

Expressions to correctly stop the rotation



There is one drawback: if you press **UP** for too long the gun will stop rotating but "rotgun" is still incremented. This will cause that the cannon is not rotating back immediately when you press **DOWN** again. To prevent this we can use an Expression Controller again, see the Figure to the left for the correct Expressions.

These Expressions will prevent "rotgun" from counting when "rotgun" is already greater than 21 or less than -21.

It is time to save your project now! Blender scenes are usually very compact so saving only takes seconds. First you need to pack (include in the Blenderfile) the textures, use the "File-menu → External Data → Pack into .blend file" to do so. Then save the file to your hard disk. This makes sure that all textures (and in case also sounds and fonts) will be on your harddisk, so that opening the file again will not need the books disk inserted.

This way you can also send the file to a friend by e-mail and he will get all textures with the game.

For further working on the scene it is recommended to unpack the data again with "File-menu → Data → Unpack into Files..."; then choosing "Use files in current directory (create when necessary)".

## An enemy to shoot at

It is now time to add something to shoot at. Select the "Target" object with the right mouse, it may be needed to rotate the view a bit using **MMB** and mouse movements to get a good view.



The tasks a game logic on the enemy has to do are:

1. React on hits (collisions) with the bullets
2. Make a silly face when hit, and die
3. Add some points to the players score

You can see that I try to keep the game logic on the target itself. Although it is possible to put all that to the player or any other central element, it would make a very complex and difficult to maintain logic for this object. Another advantage of using local game logic is that it makes it much easier to re-use the objects and the logic, even in other scenes.

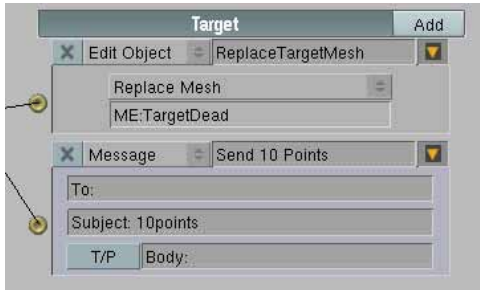
We start again by adding a Sensor, Controller and an Actuator and wiring them. You should be familiar with that procedure by now. To react to a collision, change the Sensor to a Collision Sensor. Enter "bullet" into the "Property:" field, this is the name of a Property carried by the bullet, this way the Collision Sensor will only react to collisions with bullets.



Logic Bricks to make the target look silly

Change the Actuator to an Edit Object Actuator and choose "Replace Mesh" as the type. Enter "TargetDead" into the "ME:" field. This mesh show the dead target and will be shown from now on when you hit the target with a bullet. The dead target is on a hidden Layer, you can look at it when you switch Layer 11 on by pressing **SHIFT-ALT-1**.

*Don't forget to switch of Layer 11 again with **SHIFT-ALT-1**, or shooting will not function properly. This is because all added objects need to be in a hidden layer to make the Add Object Actuator work.*



To actually score a hit we will use Blender's messaging system. This allows us to send messages from objects to objects. In this case we will signal the score-display to add some points to our score. So add a second Actuator to the target, wire it with the existing Controller and change it to a Message Actuator.



We can leave the "To:" and "Body:" fields blank, just fill in the "Subject:" field with "10points". This is equal to shouting into the room and the score-keeper will note the score.

We now need to set up the score display to react to the score messages. Select the "Score-display" object and add Logic Bricks as shown in the following Figure:

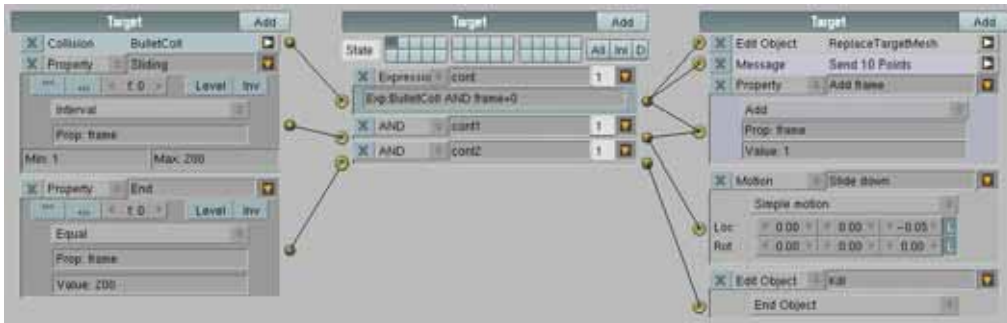


The "Score-display" is again an object with a special texture on it showing the content of the "Text" property as explained for the bullets display. Be sure to change the 999 to zero or the score will start with 999 points. The Message Sensor will only hear messages with the "10points" subject and then trigger the Property Actuator to add 10 to the "Text" Property which is then displayed. This way it is also very easy to add different scores to different actions, just add a new line of Logic Bricks listening for different subjects and then add the appropriate amount of points.

You should now try out the game so far and shoot at the enemy. This should add 10 points to your score. If anything fails to work, check especially the correct wiring, and that the names and capitalization of Properties and message subjects are as they should be.

In the final game the targets start to slide down the tube, look at Figure "Advanced animation for dead targets" for a possible solution for that. The simple target also has the drawback that it will still add a score when you hit a dead target.

We have already used most of these Logic Bricks. Basically it just uses an Expression Controller to avoid counting scores by shooting on dead targets. This is done by checking the frame Property for being zero which gets increased by one when a hit occurs. As soon as frame is one a Property Sensor type "Interval" fires as long as



frame is between 1 and 200, this triggers adding frame by one and sliding the target a bit down in every frame. When frame reaches 200 the object gets killed by the Edit Object Actuator. Advanced animation for dead targets

Now you can place more targets in the tube. For that select the target, then use **ALT-D** to copy it. Now use grabbing (**G**) and rotating (**R**), use **CTRL** for a snap) to place the new enemy.

Together with the reference part of this book and the final game on the disk you can now try to extend the file or just enjoy playing the game. Don't desparate and keep on experimenting. By breaking the task into small steps, even complex logic is possible without getting lost.