Quake II: Physics

Q2 was coded by aliens

Written by **n00k1e**

Credit also goes to **draxi** aka SK3L3T0R

(For a huge help with testing and understanding q2 physics)

Intro

Trying to understand the physics of Quake 2 is very similar to what real physicists do in the real world. They do not know many things about the Universe, but just from observing what's going on around them they see patterns of behaviour and they can write down some rules to make sense from all of it. One day, hopefully, they will uncover all the details about the world but for now, the race to write down the theory of everything is still on.

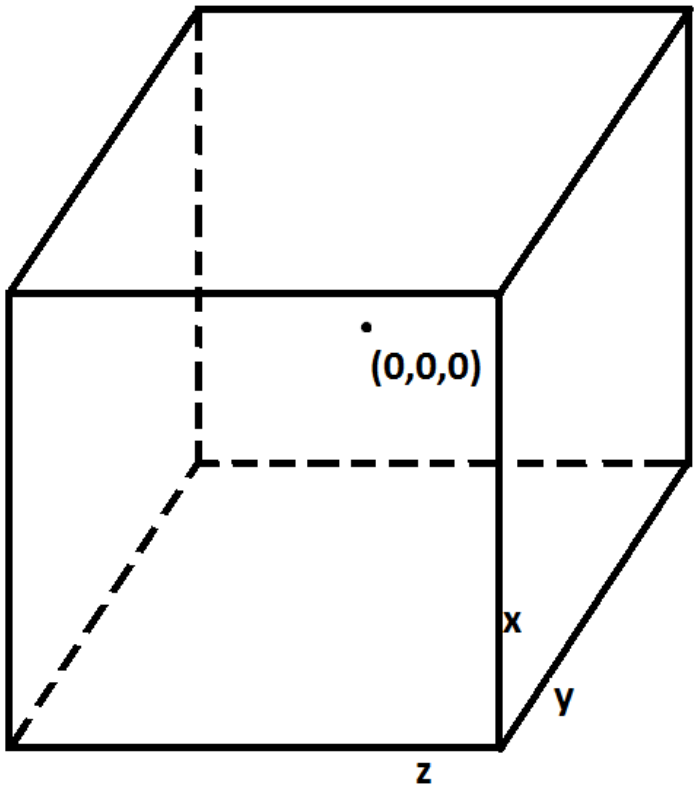
Same thing applies in our case. We know that someone programmed Q2, but we don't know how the hell does it really works, so just by playing the game I was trying to figure out as much as I possibly could. I don't have all of the answers, but I will try to cover few major things.

Here are the results that came out from many hours of testing.

1. Map

Understanding the effects that the map has on the movement is crucial so let's start here. Each map in Q2 is a 3-dimensional space that is placed in a huge box. That box is quite big; it has the dimensions of 8192x8192x8192 quake units.

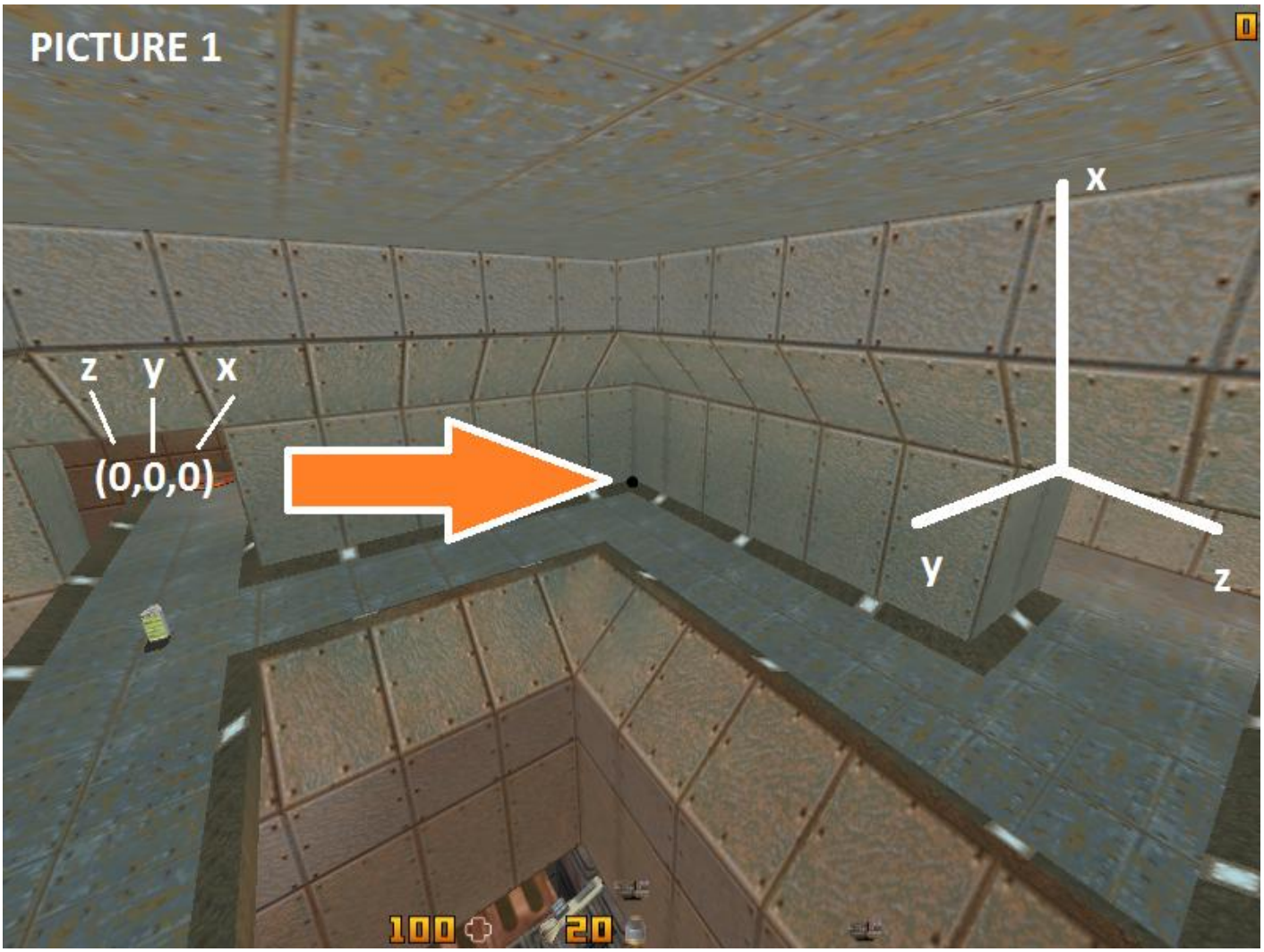
It has the point zero inside of it which, not surprisingly, is placed exactly in the middle.



Map designers are usually placing their maps somewhere around point zero. Some maps are placed a little above, some below. This is VERY important for the player as it affects the movement. Let me explain why.

This place is like a sucking machine or like a black hole in the Universe. It has something like a force of gravity. It doesn't matter how far away you are from that place though, it will always pull on you with the same power. What matters is your position on the map and the direction you are jumping.

Let's see an example on a standard Q2 map, q2dm8. Point zero on this map is located behind the corner, just few units behind the Railgun *(see PIC1 on the next page)*.



2. Movement

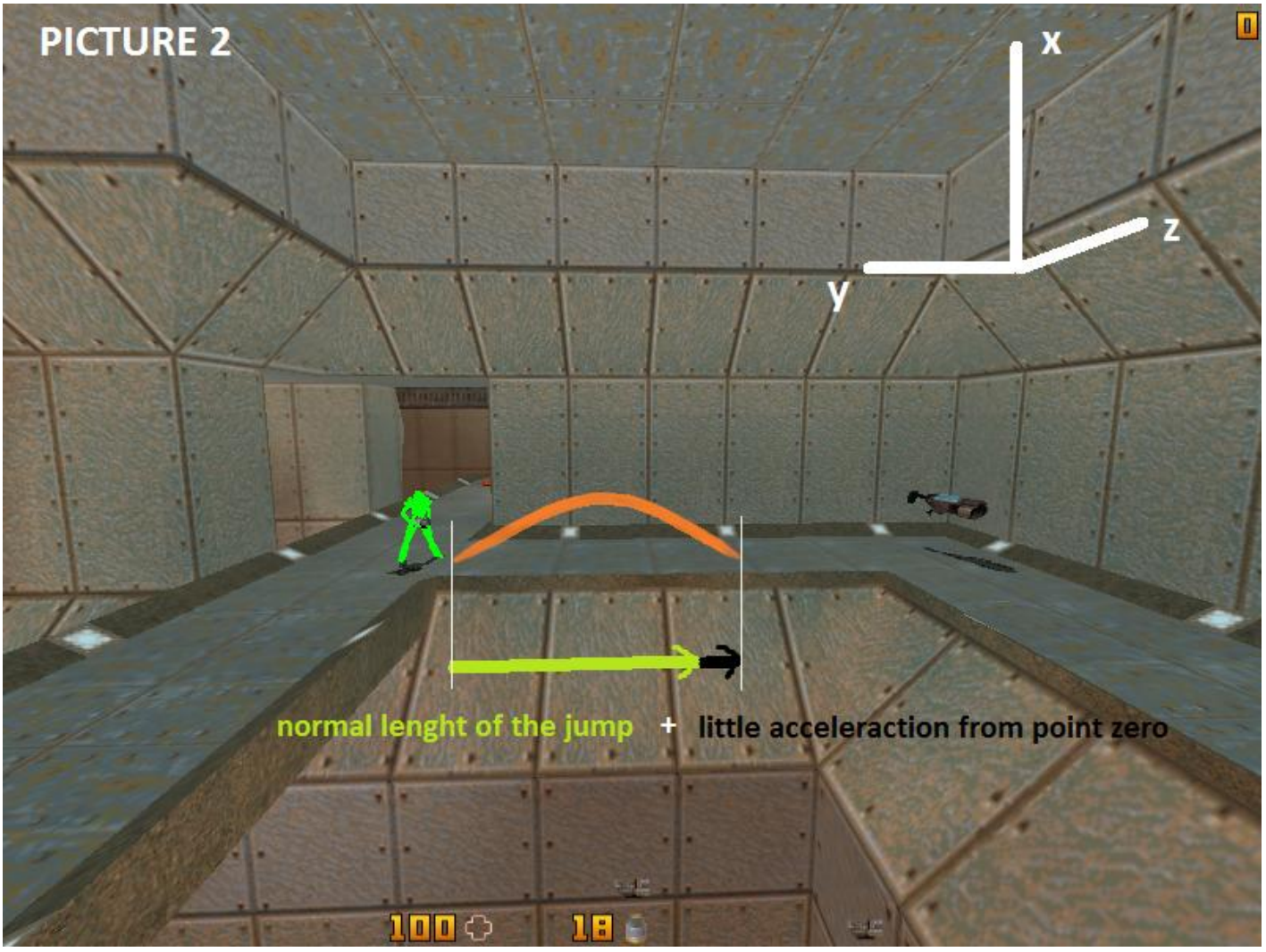
As I have mentioned before, each map has its own point zero. We know that each axis is equal to 8192 units. Let's split this into a half. 4096 units will be placed above the point zero and 4096 below. Every unit above the point zero will have a positive value (i.e. 235 or 778 or 3701) and every unit below will have a negative value (i.e. -80 or -950 or -2042). The further you go away from the point zero, the bigger the values will be. You can always check your position using 'viewpos' command in Q2 console.

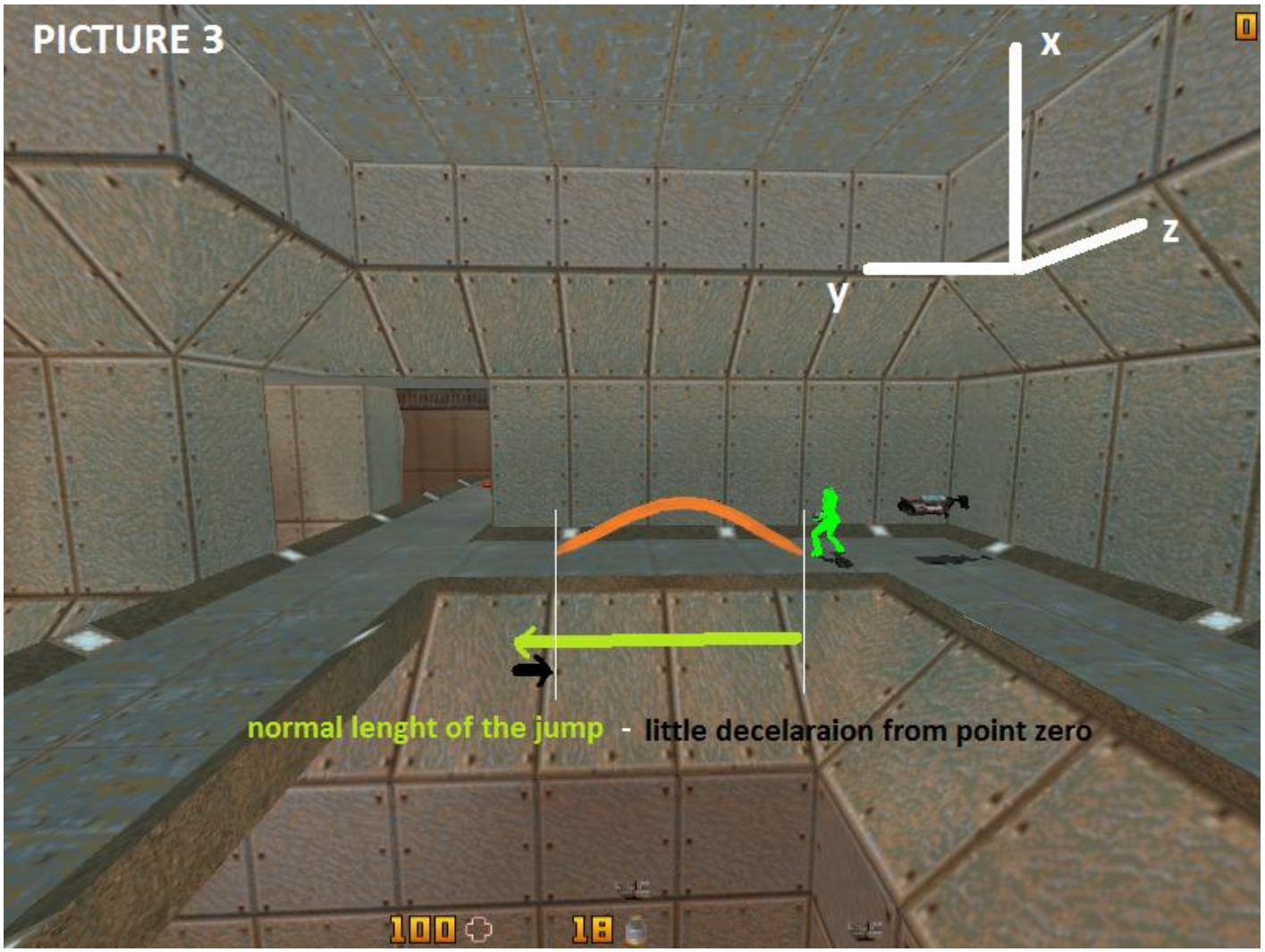
Explaining Axis Y

Player who is trying to jump towards the weapon *(see PIC2 on the next page)* will have a little longer jump than a player who is jumping from the weapon towards the exit *(see PIC3).* This is because the zero spot is located behind the weapon on this map and it is giving our player from *PIC2* some acceleration. It is giving some deceleration to player from *PIC3* respectively.

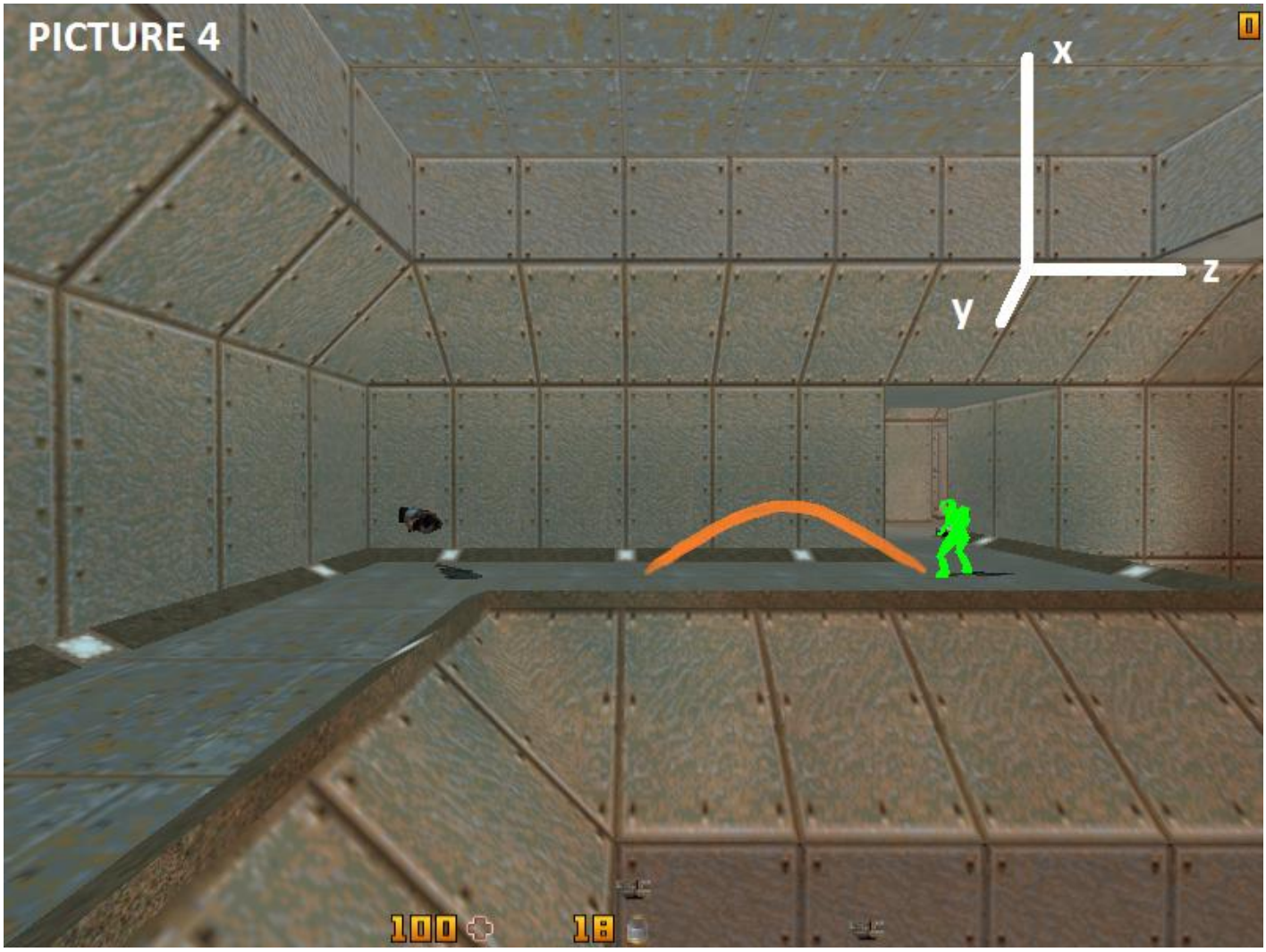
Explaining Axis Z

Same rules as above also apply here *(see PIC4 and PIC5).*

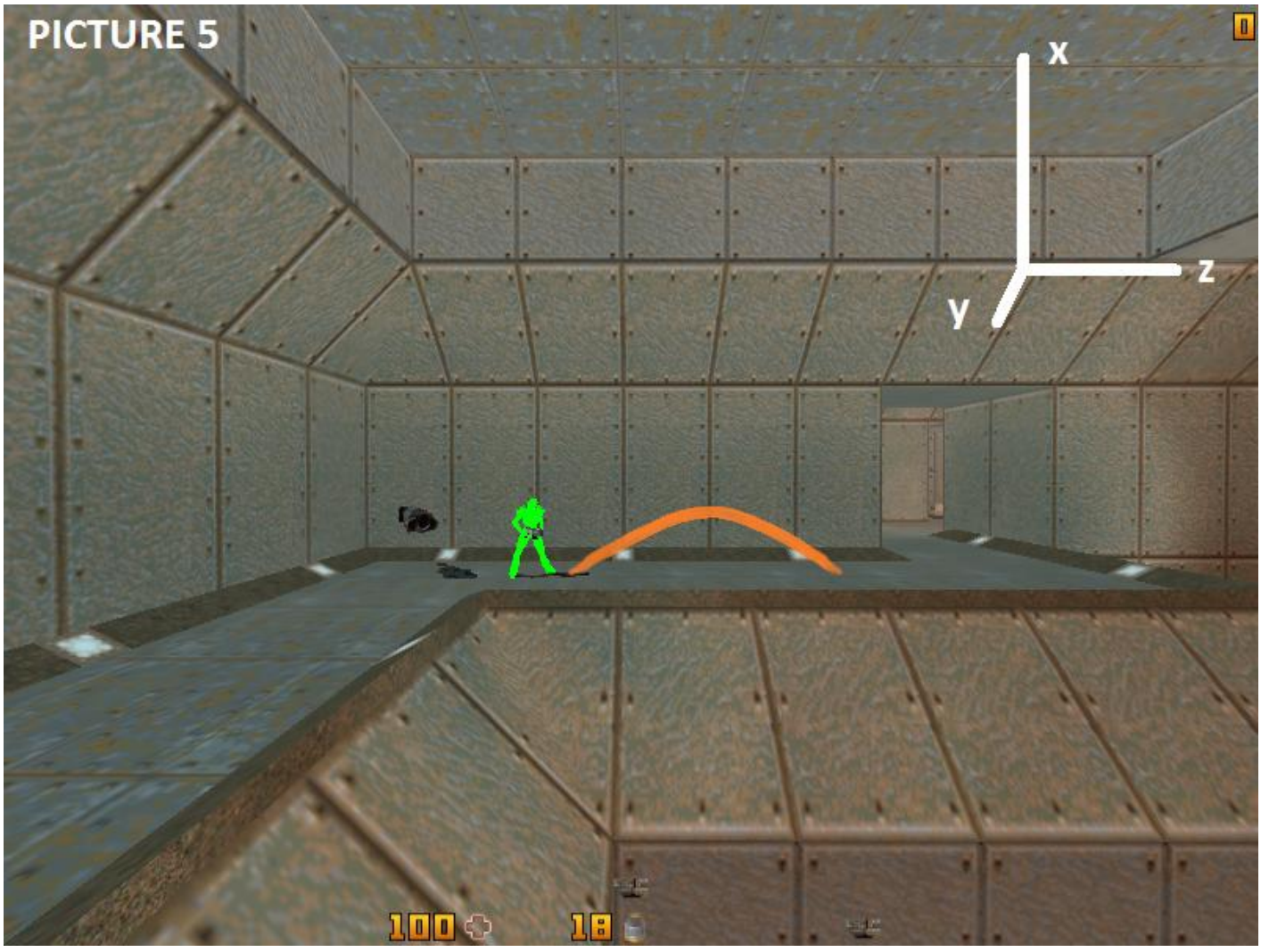

PICTURE 2
x
z
y
normal lenght of the jump + little acceleraction from point zero
100
18


PICTURE 3
x
z
y
normal lenght of the jump - little decelaraion from point zero
100
18

PICTURE 4

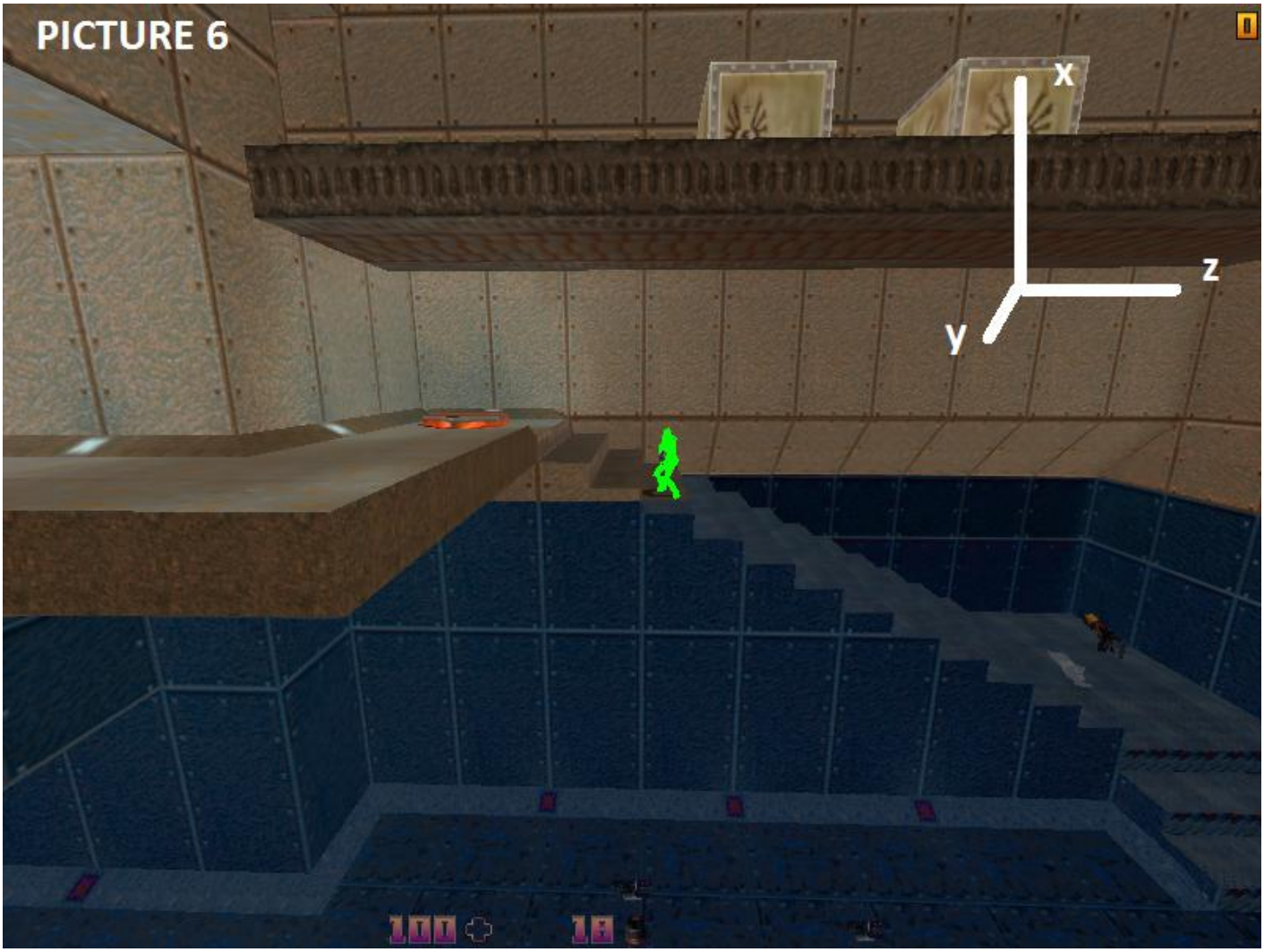


PICTURE 5



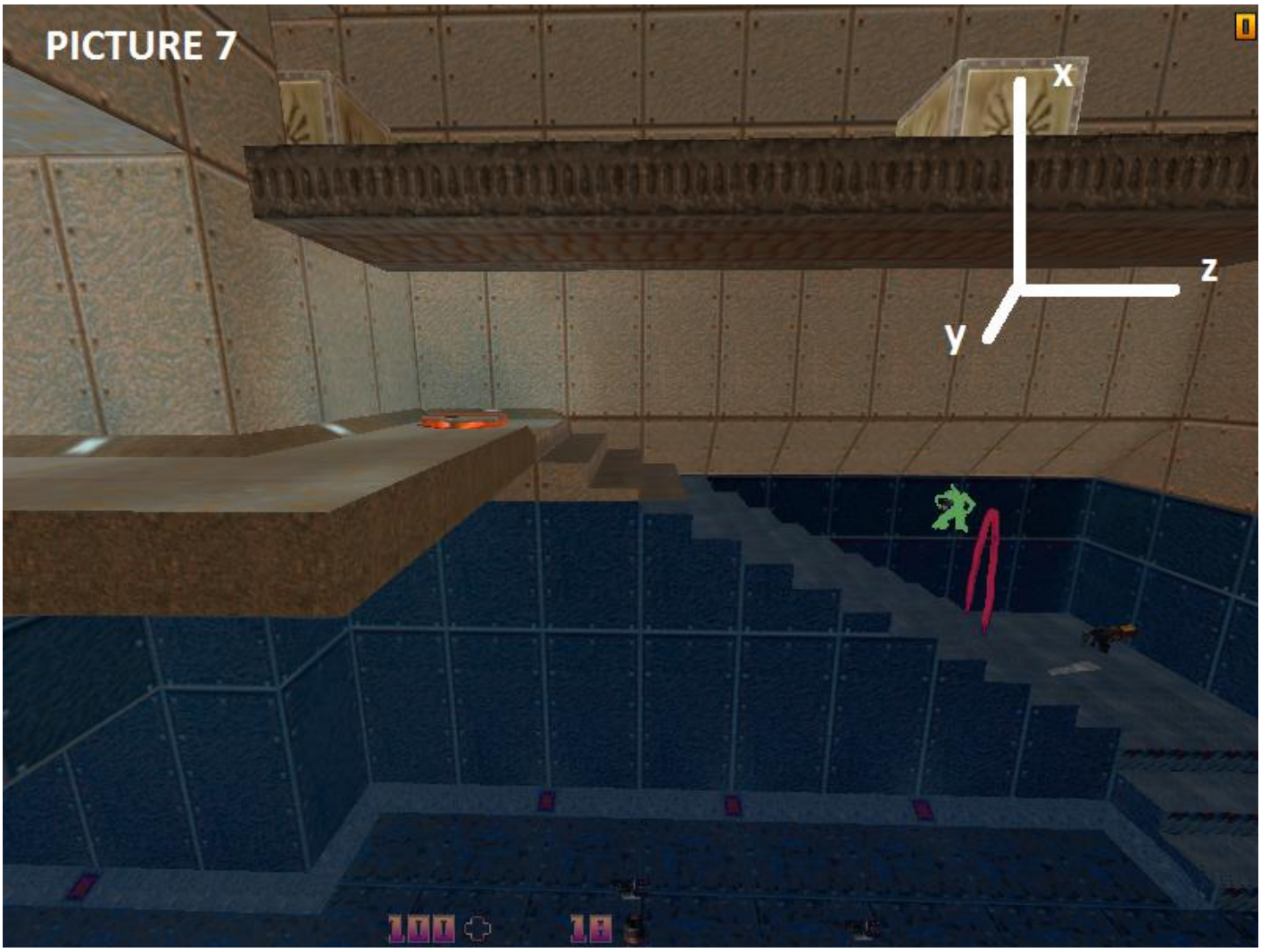
Explaining Axis X

Now let's see how it works vertically as this is the most important axis. What you can see on the *PIC6* is a player standing at (85, 1516, -1) so it is only one unit away from being at the point zero for the axis X. All the space that is below the player (blue colour) has a negative value for the axis X and all the space above that (normal colour) has a positive value for the axis X.

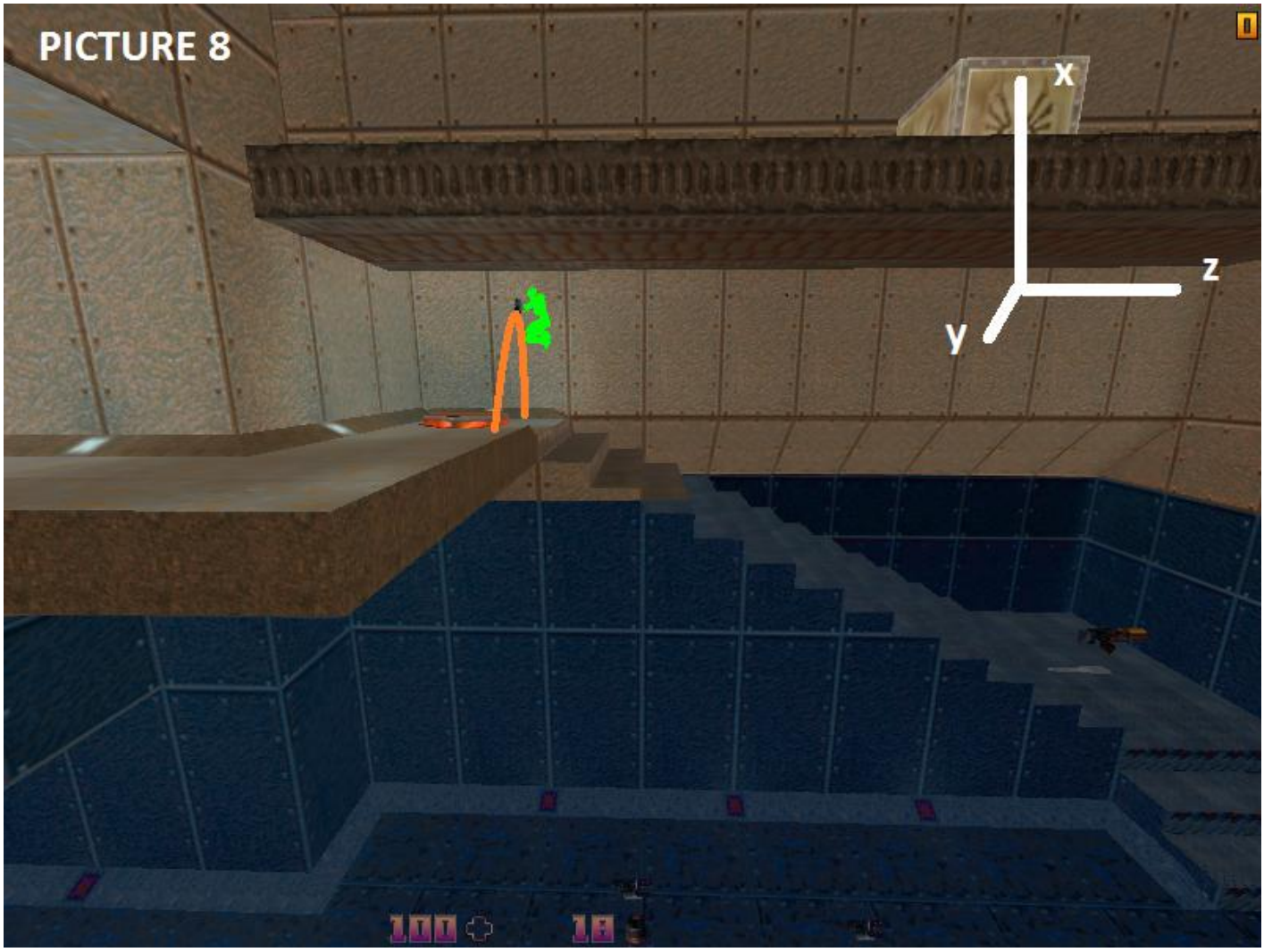


On the next page there are two pictures. *PIC7* is showing a normal vertical jump (just a simple +moveup) performed in the negative value area and *PIC8* is showing a player performing a jump in a positive value area. Jumping in the negative value area will boost you up a little bit (that point zero near Railgun is sucking you up) and jumping up in the positive value area will pull you down a little bit so your jump won't be as high (that point zero near Railgun is pulling you down).

PICTURE 7



PICTURE 8



3. Big Box Paradox

Here is why we can jump on the top of the big boxes near RL and GL on this map using 100+ fps but we can't do it on q2dm1 for example.

On q2dm8, big boxes are located in the negative value area as you can see on *PIC9*. We know already that jumping up in this area will give a player a small boost upwards. Combined with a cl_maxfps value equal or higher than 100 on cl_async 0 or equal or higher than 91 on cl_async 1 (tested with r_maxfps 1000), will give us that little boost. This is all that we need to get few more units and jump on top of the big boxes.

Why can't we do the same thing on q2dm1? It is very simple. The whole map was placed above the point zero so it doesn't matter where you are; you will never get that little boost up. Point zero on q2dm1 will be always pulling you down.

4. Big Box Paradox vs FPS Values

I have mentioned about the fps values on the previous page. It turns out that the values that I have provided are not the only values needed to jump on the top of the big boxes. Here are some examples (using cl_async 0):

PIC10 – Jumping straight on to the big boxes:
Minimum cl_maxfps needed: 100

PIC11 – Making a pre-jump before jumping on the boxes:
Minimum cl_maxfps needed: 83

PIC12 – Jumping from a small height before jumping on the boxes:
Minimum cl_maxfps needed: 53 to get on top of the first box
59 if you want to get to the top

PIC13 – Jumping from a big height before jumping on the boxes:
Minimum cl_maxfps needed: 42

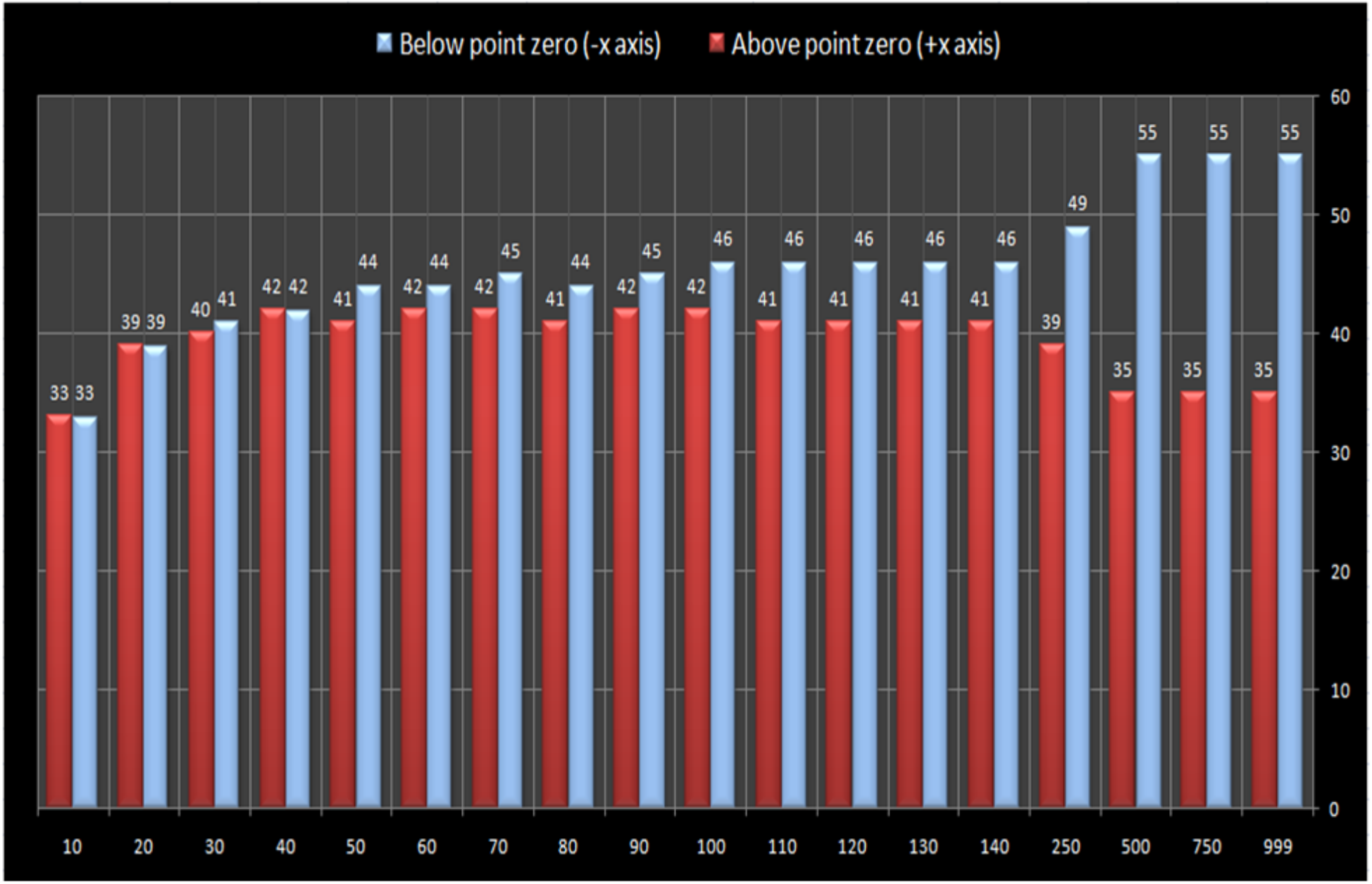
PICTURE 11

PICTURE 12

As you can see, the higher the player is starting from on the map, the lower the fps value is required to make a successful jump on top of the big boxes. This is some kind of bug in Q2 physics that I cannot explain at this point in time but at least we know that something like this exist.

5. Player Position vs FPS Values

We have learned about the negative and positive value areas in Q2 maps. Now I will try to explain how a certain cl_maxfps value can help us make the most of the knowledge that we have so far. I have tested 18 different fps values in both, positive and negative value areas. I was trying to see how fps affects height of the jump in those areas. We can clearly see from the graph (next page), that using lower fps (10 to 40) in both areas of the map isn't very different. Jumps will be definitely higher with cl_maxfps 50 and above, if you are positioned in the negative value area on the map, but they will be lower in the positive value area. For example using cl_maxfps 500 for a TDM game would give you VERY high but also VERY low jumps respectively to your position on the map. The best thing is to stick with fps values between 60 and 120 as everyone is already doing.


Below point zero (-x axis)
Above point zero (+x axis)
60
50
40
30
20
10
0
Jump height
(measured in quake units)
33 33
39 39
40 41
42 42
41 44
42 44
42 45
41 44
42 45
42 46
41 46
41 46
41 46
41 46
39 49
35 55
35 55
35 55
10
20
30
40
50
60
70
80
90
100
110
120
130
140
250
500
750
999
cl_maxfps values (from 10 to 999)

Thanks for reading

...and please, don't ask any questions ☺

Technical Specifications:

PC: Testing was performed on a Windows 7 x64 machine with ATI Radeon video card.
Monitor: 60hz LCD

Q2 Client: R1Q2

Main settings:
gl_swapinterval 0
cl_async 0 (except one big box test)
vid_ref r1gl
gl_mode 4

15 pages in total

Document version 1.1

22nd July 2011