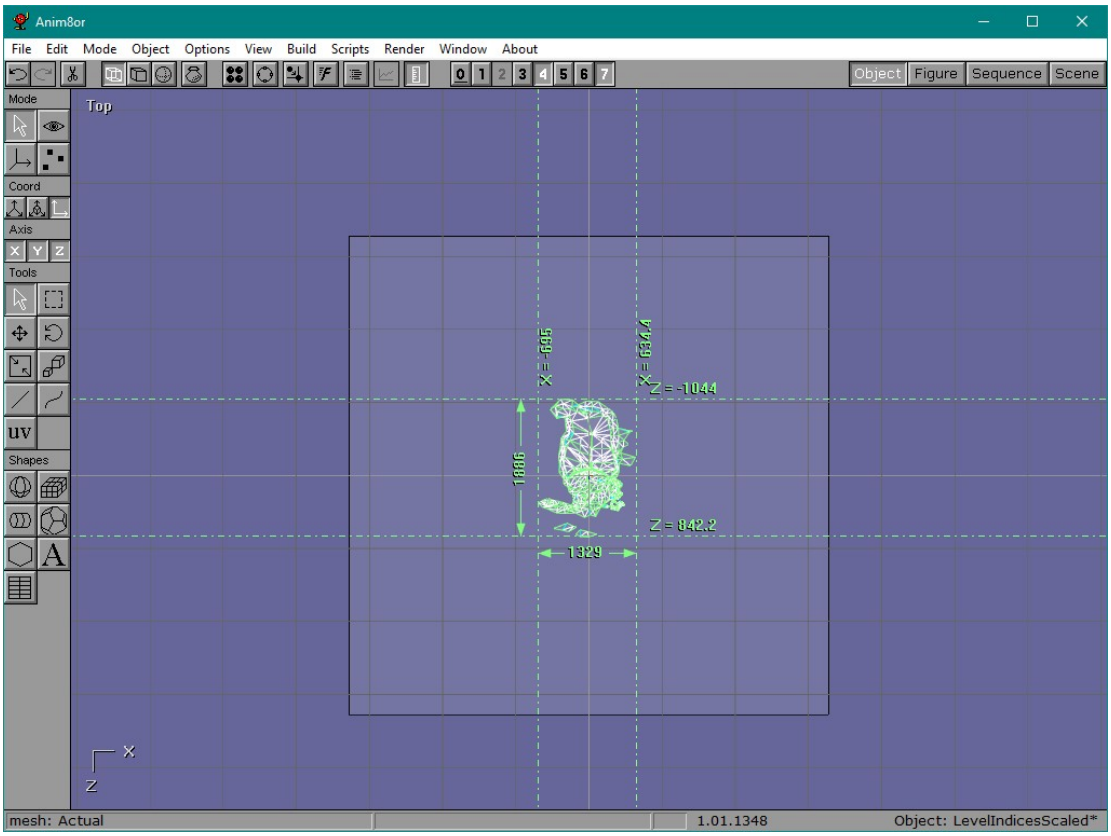
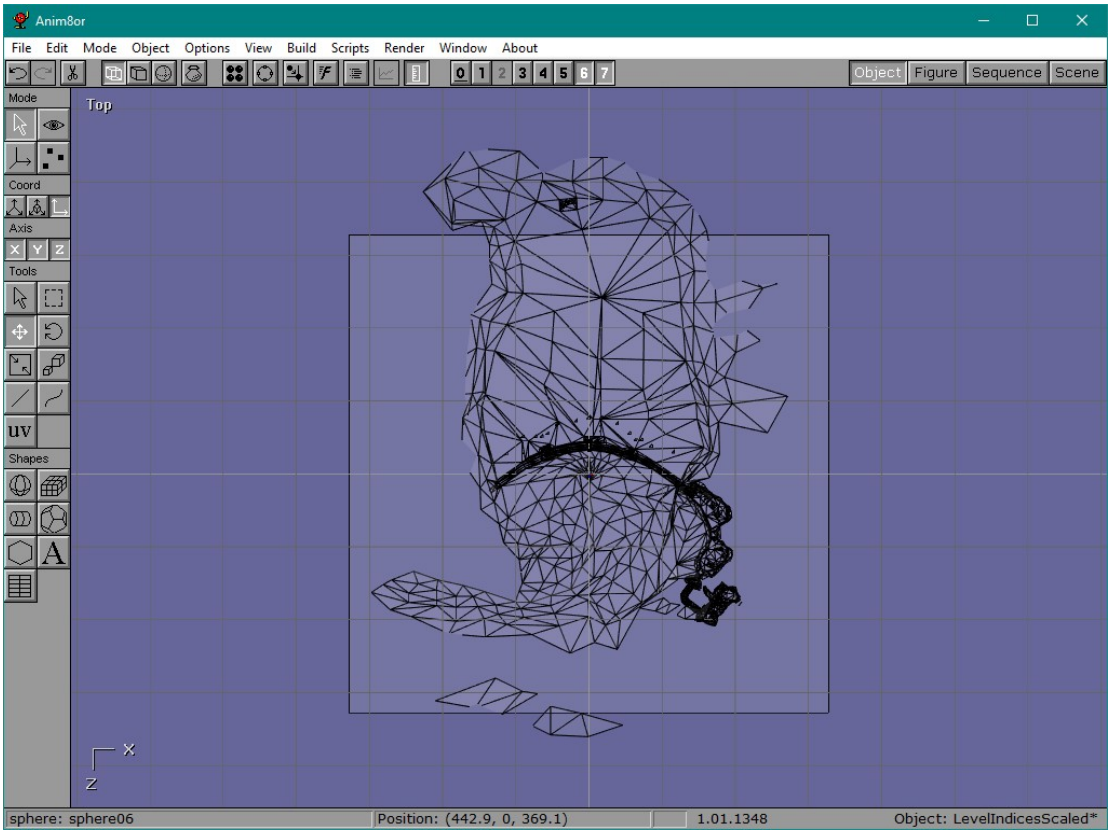
Viewport, RSP limitations and Modelling requirements

Original Misconception

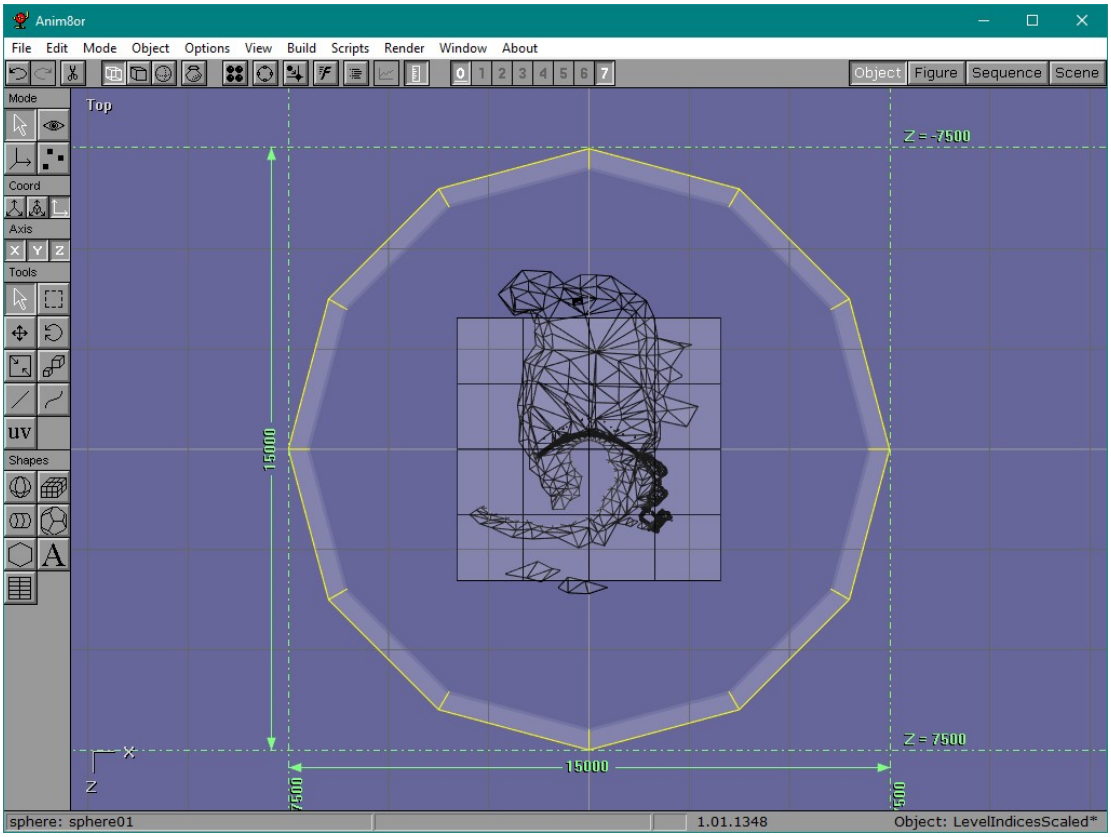
For a long time it was considered that the BG was scaled down for storage and then scaled back up at runtime for reasons unknown.
BG is stored as integer unit vertices so perhaps scale was something to do with that and so we built this picture



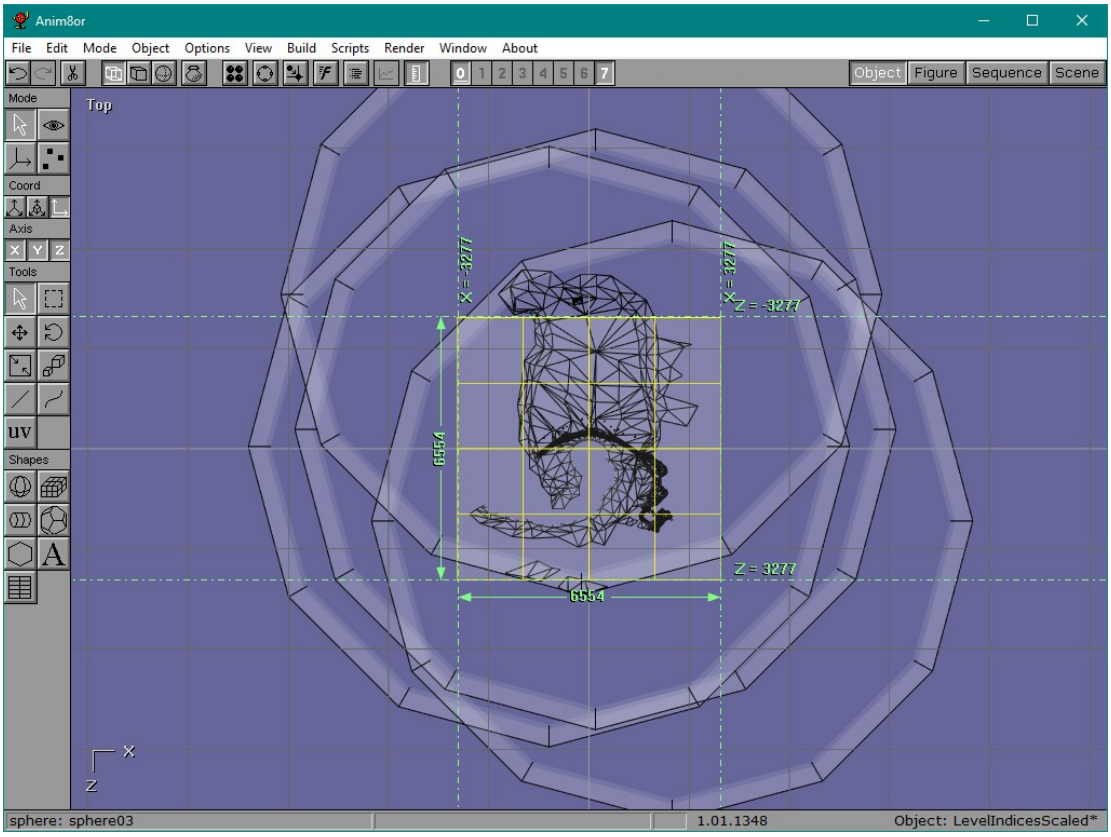
Raw Data for Dam



Misconceived Level.Scale purpose



Misconceived Visible radius



Misconceived Visible radius from different positions on map

As you can see, Dam poses many problems, the biggest being that both it and your visible radius exceed the 65,535 RSP Fixed point box that which nothing can escape.

Recap:

> *The RSP is a 32bit Fixed point machine that uses the format s15.16, that is – the biggest integer it can represent is 32,767 and the smallest is -32,768.*
> *What is always forgotten about is the 16bits of fraction able to represent 1/65535 of a unit.*
> *For example; if a unit is 1meter, the RSP can represent a fraction of 0.000015m or 15 microns*

This is where Level.Scale truly plays its magic and also shows that the N64 can actually have very large worlds if you used 1Km as your unit (I'm looking at you Rogue Squadron – could have had bigger maps or at least a non-bent trench-run.)

Revelation about Level.Scale

Eventually some thought was done and the realisation dawned that prevented GE from breaking the rules.

Scale has no “Units” so long as ALL objects use the same scale and ratios remain constant.

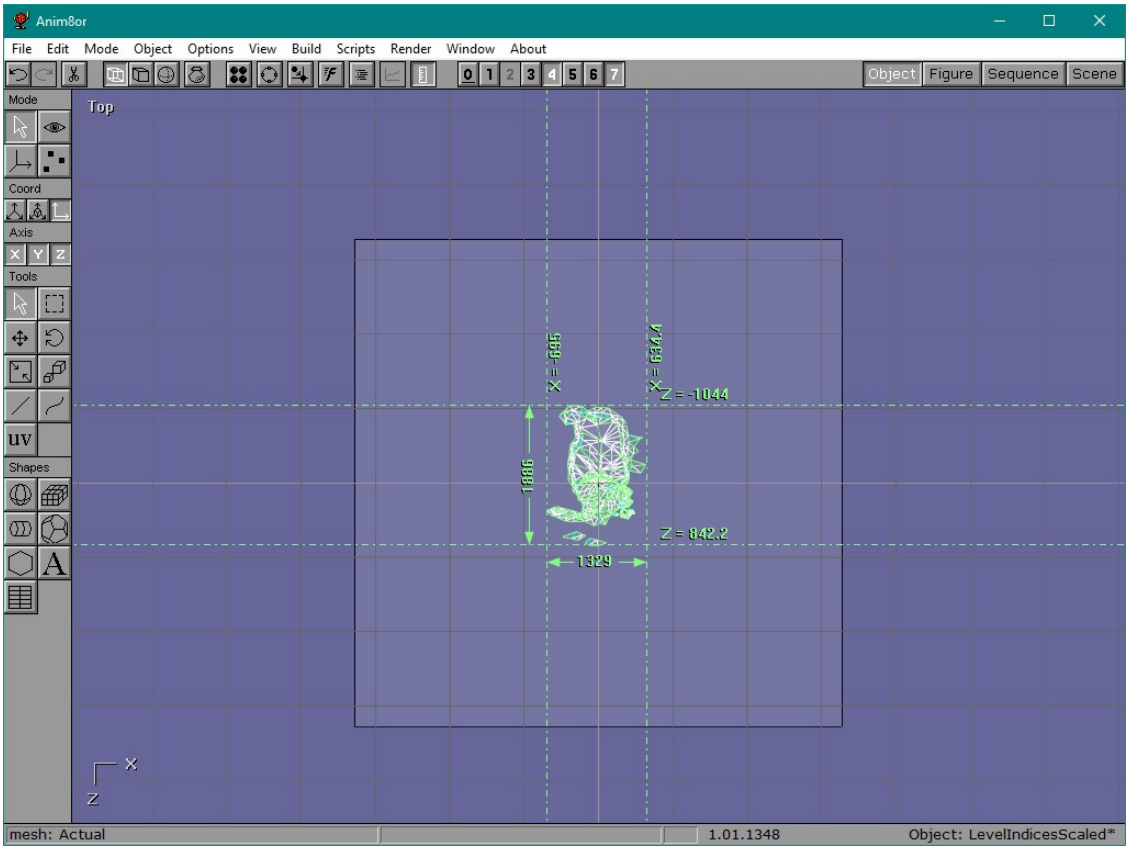
In other words, if you expect a door to be just taller than you, and a box to be half your height, you can be ANY size – if the door is still a little taller and the box is still half your height you will believe you are the *correct* size.

To put some number on this;
Bond is 1.8m tall.
He is represented as 1800 in model and scaled by 0.1 bringing him down to 180 - we assume this unit is a cm.
A doorway is 2.1m, but Dams doorways are 48 units

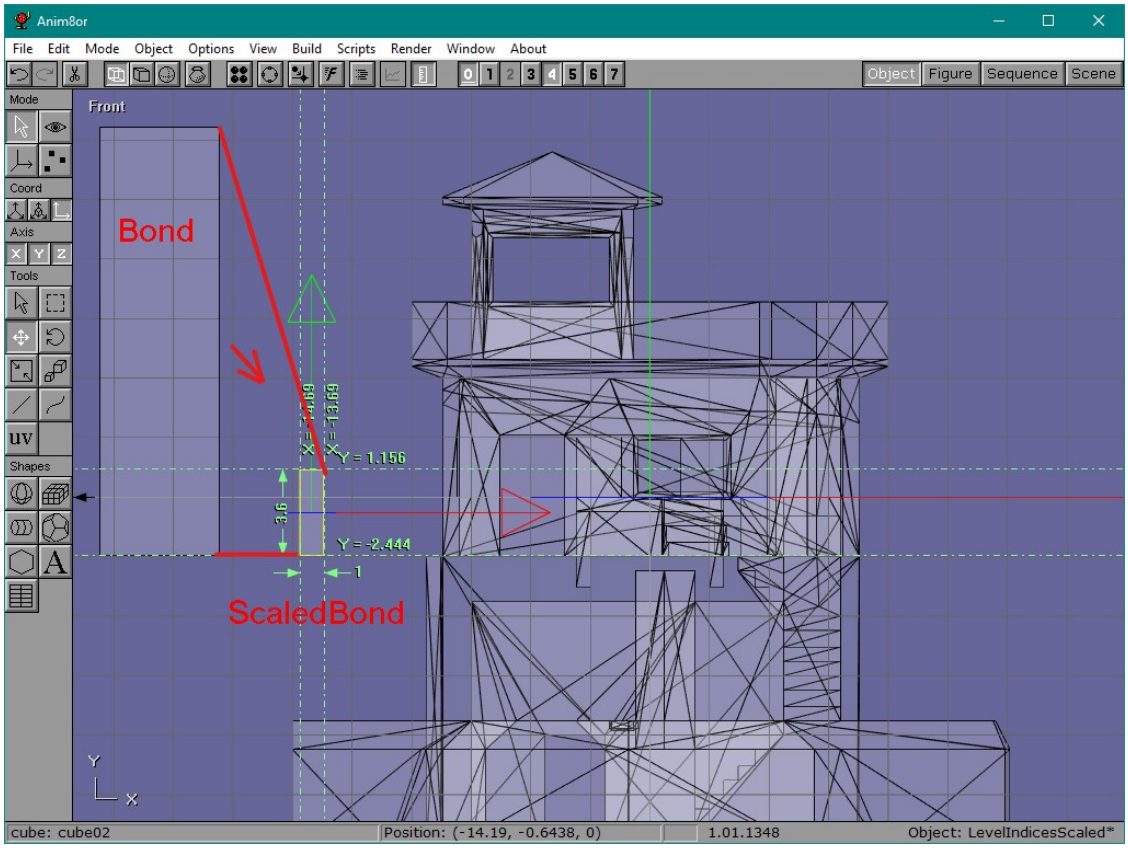
We used to divide 48 by Level.Scale (this case 0.2) to get 2.1m for a door. Bond and Dam were now using the same cm unit with the Bond to Door ratio of 1:1.16

However, if you instead scale bond by 0.2, the door remains 48 and Bond becomes 36, at first you think this is wrong, how can bond be 36cm, however check the ratio, its 1:1.16 meaning the “unit” is irrelevant so long as everything else uses the same scale and ratios match.

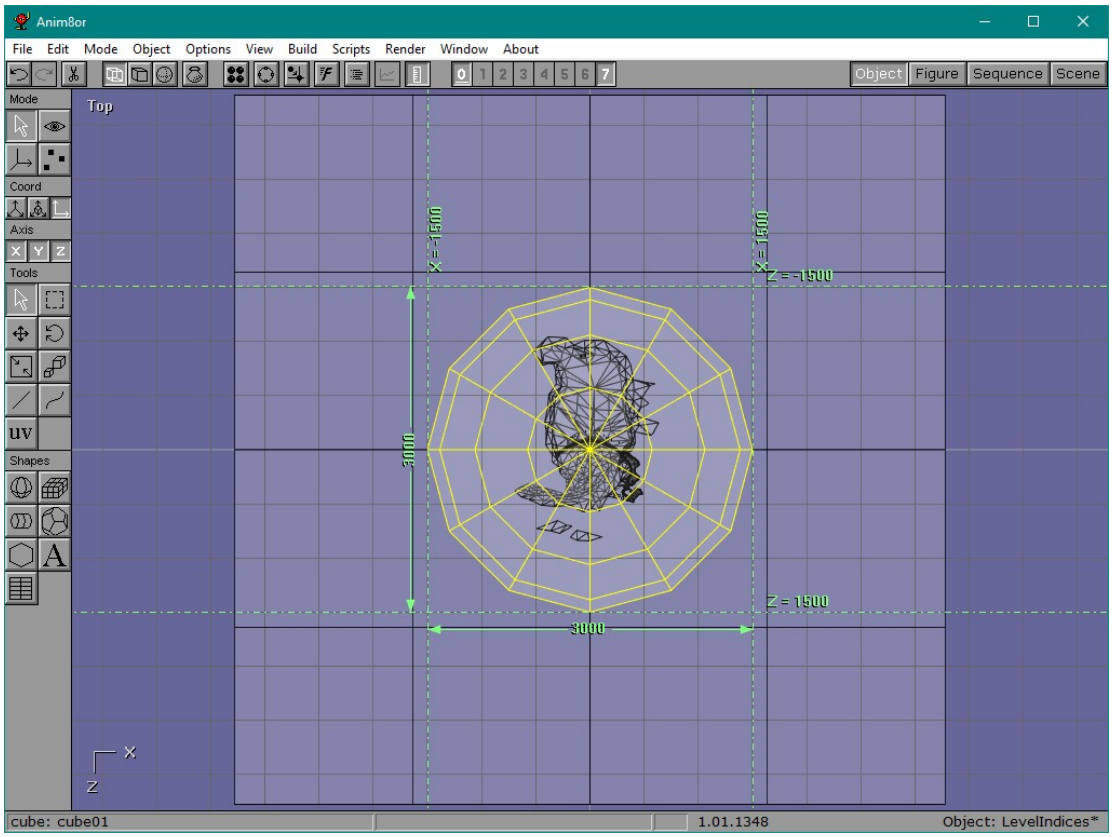
The bullet hole of 2cm now uses the fractional part of the RSP and becomes 0.399994 units which on-screen has the apparent size of 2cm Again, the ratio of Bond to Bullet is 90:1 whether Bond is 180 and bullet is 2 OR Bond is 36 and bullet is 0.399994.



Real Size of Dam in RSP



Actual Scaling

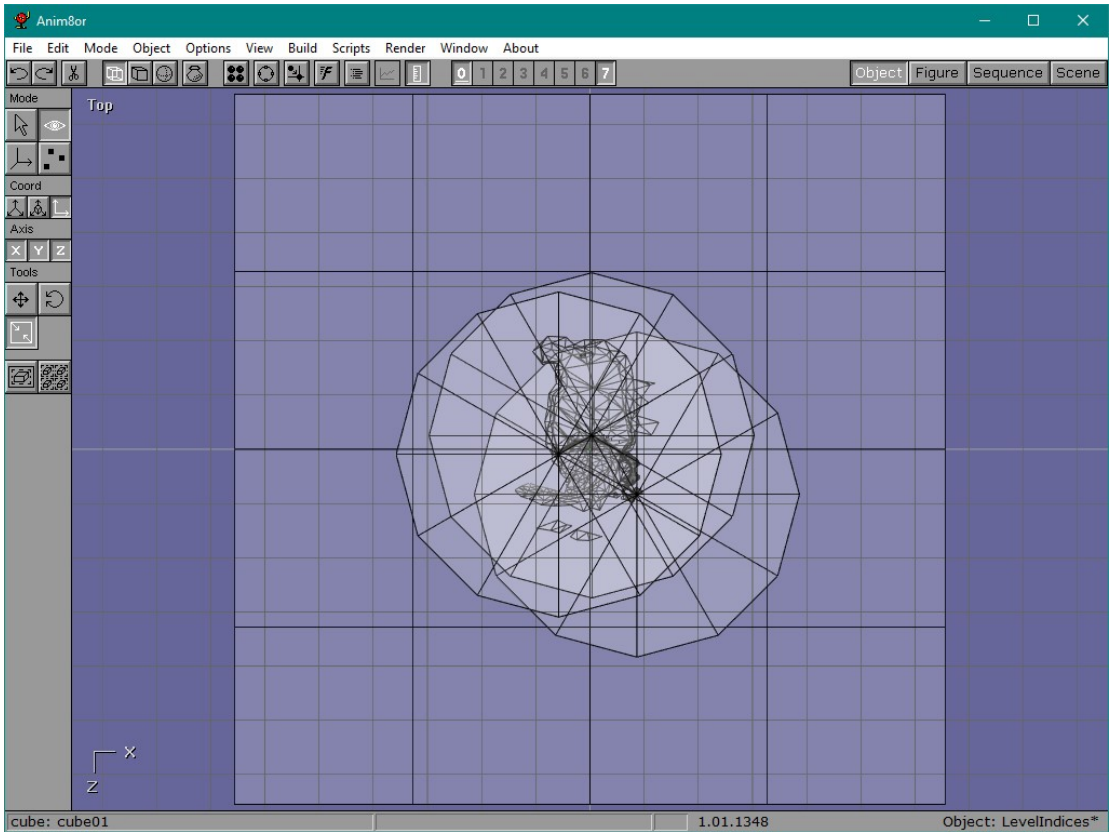


Scaled Visible radius

With that realisation, the visible radius now always fits the RSP box too – however…. Here too lies a misconception…

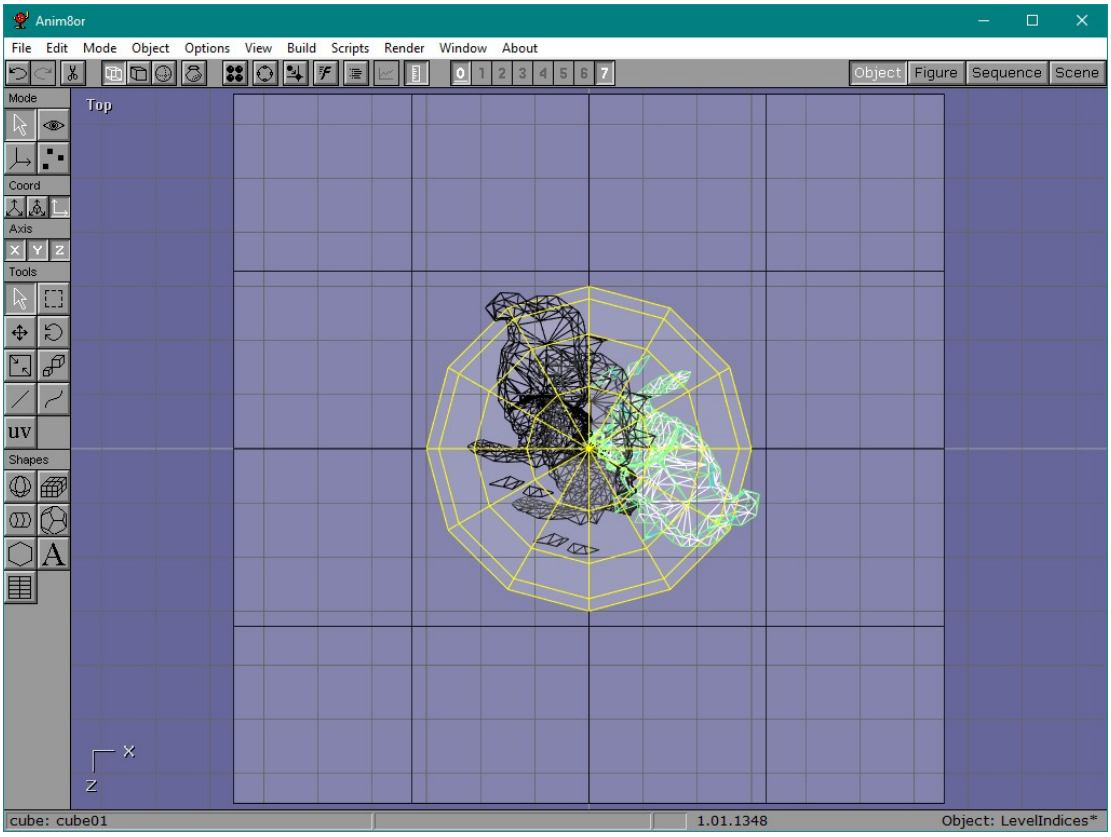
The World revolves around YOU

Most people think of Moving within a world like below



Misconceived Visible radius from different areas of map

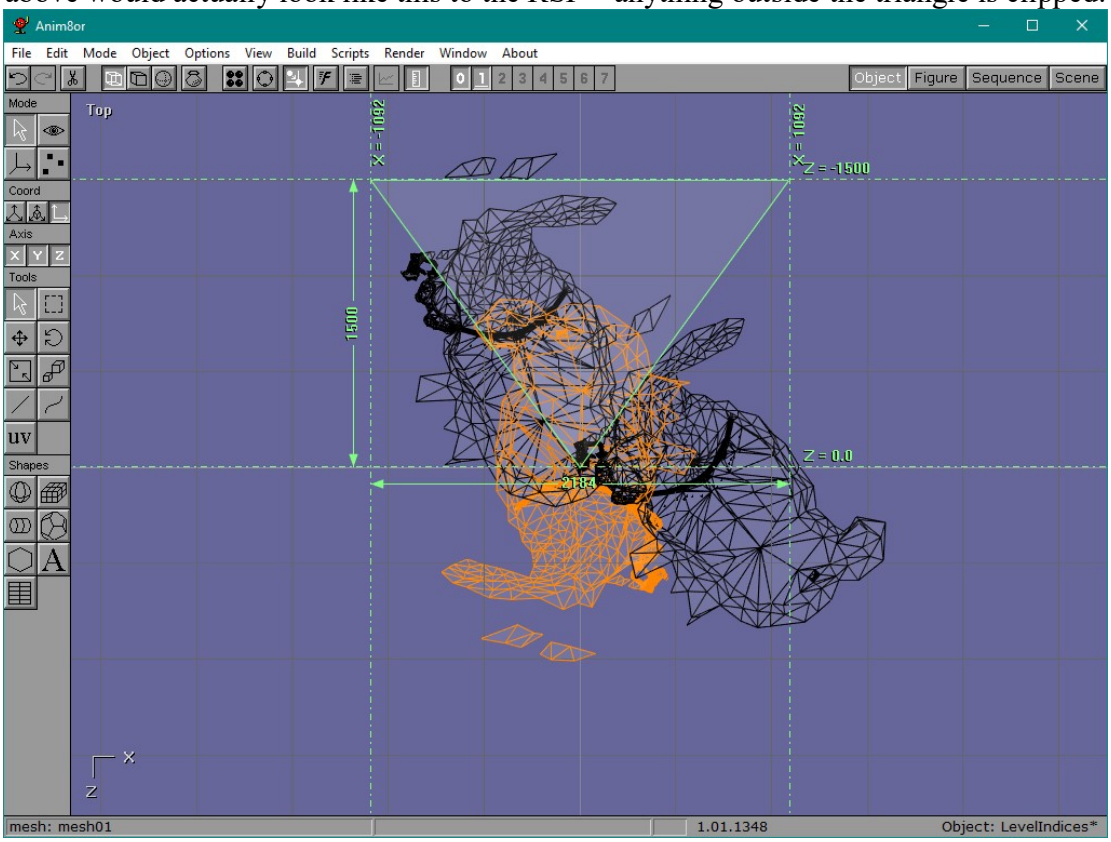
However, YOU are always at (0,0,0) and the world revolves around you.
This seems strange but its how computers generate images. The computer first position objects within the world, then it positions you, then it moves everything so that you are back at the (0,0,0) and you are facing (0,0,0) then it renders the screen. Every time you move or turn, the computer does the opposite and renders the screen with you back at (0,0,0)



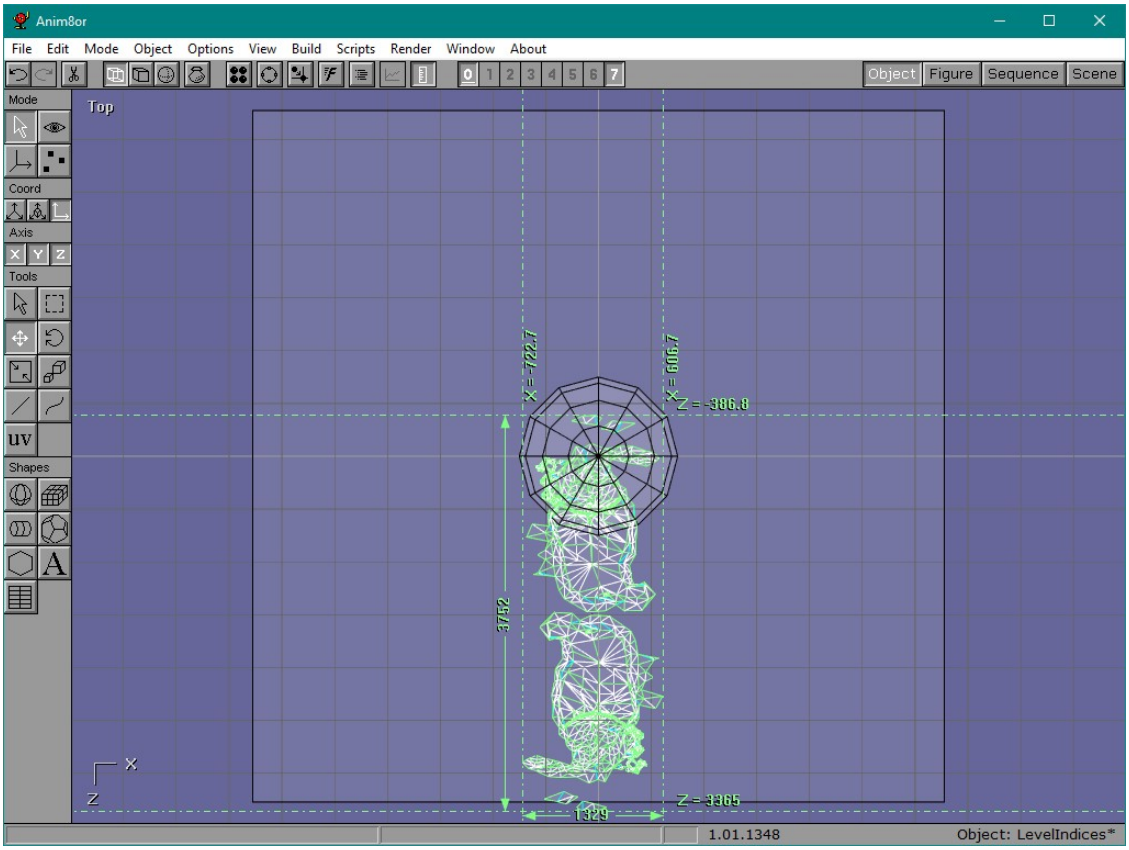
Actual Visible radius at different places on map

The Highlighted BG shows the effect of you turning to look at the first guard tower at the start of the level.

One further technical note, while this document describes “Visible Radii” the above image shows that the computer can actually simplify its work by using a flat plane.
So the image above would actually look like this to the RSP – anything outside the triangle is clipped.



Modelling Requirements - When problems occur



Level too large (at least 1 vertex exceeds +/-32,768)

Problems occur when you exceed the 65535 box as shown above, and one of 2 things can happen: The consol will crash or vertices will be clipped – causing textures to “wobble” on surfaces or triangles to shoot across the whole level.

To prevent this all vertices must remain within the 65535 box at all times. First I will quote the SDK:

> *... If numbers in the final coordinate system (or intermediate coordinate systems) are too big, then the geometry of objects can be distorted, textures can shift erratically, and clipping can fail to work correctly. In order to avoid these problems keep the following notes in mind:*
>
> 1. *No coordinate component (x, y, z, or w) should ever be greater than 32767.0 or less than -32767.0*
> 2. *The difference between any 2 vertices of a triangle should not have any components greater than 32767.0*
> 3. *The sum of the difference of w's of any 2 vertices plus the sum of the differences of any of the x, y, or z components should be less than 32767.0. In other words for any 2 vertices in a triangle, v1=(x1,y1,z1,w1), and v2=(x2,y2,z2,w2), these should all be true:*
>
> *abs (x1 - x2) + abs (w1 - w2) < 32767.0*
> *abs (y1 - y2) + abs (w1 - w2) < 32767.0*
> *abs (z1 - z2) + abs (w1 - w2) < 32767.0*
>
> *One way to check this is to take the largest vertices that you have and run them through the largest matrices you are likely to have, then check to make sure that these conditions are met.*
> *A recommended way of avoiding trouble is to never allow any component to get larger than 16383.0 or smaller than -16383.0. If you want to keep all components of coordinate value within the range of value above, ensure that M*S+T<16383.0. M, S and T are as follows:*

__M__ the largest component (x, y, or z) of the largest model in your database.

__S__ The largest scale (i.e., number in the upper 3 rows of the matrix) in the matrix made up of the concatenation of the largest modeling matrix, the largest LookAt matrix, and the largest Perspective matrix you will use.

__T__ the largest translation (i.e., number in the 4th row of the matrix) in the matrix made up of the concatenation of the largest modeling matrix, the largest LookAt matrix, and the largest Perspective matrix you will use.

If you experience textures wobbling or shifting over a surface, clipping not working correctly, or geometry behaving erratically, this is a good place to check.

What all this means is clear now, as shown in the previous section, everything moves around you, so the maximum co-ordinate of any vertex in the BG is simply +/-16,383 OR the furthest vertex from the furthest place you can stand must be <= +/-32768 (meaning you can exceed 16,383 if you cannot walk to such a place that such a vertex would exceed +/-32.787

eg, If dam was mirrored it still fits because you cannot walk out to the edge as shown - meaning all vertices are within 32,767 of you at all times.

Now Rares scaling makes sense, they made a BG to within 16,000 and scale bond to fit standard ratios.

PD changes this slightly. Because no BG exceeds 32,768 they removed Scaling and instead use fixed Units where each integer unit = 1cm
This is why Dam will never fit in PD, it is simply FAR too big without the ability to scale away from cm units.

One other point to mention, z-buffer looses accuracy with long view radii.
The z-buffer uses 18bit precision but if you have a draw distance utilising all 16bits integer, the z-buffer will fight among surfaces that are less than 0.25 units apart.
To take dams scale for example, this would mean fighting between 2 surfaces that appear > 1.5cm (Equivalent to the thickness of the Alarm meaning the Alarm would fight with the wall)

The shorter the view radius the more fractional numbers can be assigned the z-buffer allowing it to correctly order surfaces at shorter distances apart.

For further information on **Visibility, Fog, Object Bounding Volumes and Fade-in** the doc is available here
http://www.shootersforever.com/forums_message_boards/viewtopic.php?t=6991