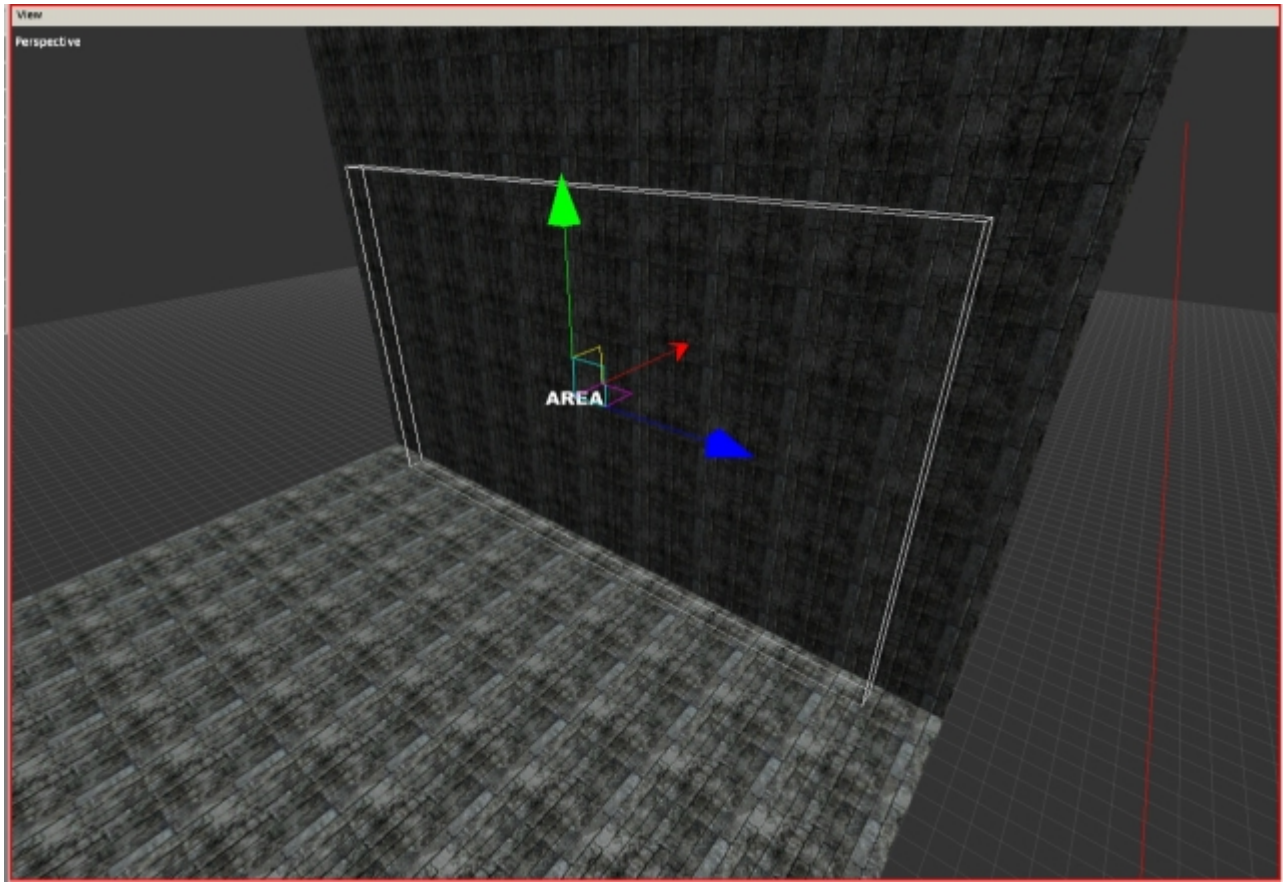
# Running up walls

This code will simulate the player running up walls.
Since code ends up being too large and thus confusing, i'm dividing it in parts.

1. How it works




This feature uses areas to determine where the wall is and to detect when the player jumps to it.
Using the function *AddPlayerBodyForce* i can simulate the force of each leg running up.

To get the interval between steps, i'm using *Timers*.

We declare the area collision: (*AreaWall1* is the area covering the wall and *WallRunCollide* is the function.)

```
void OnStart {
    AddEntityCollideCallback( "Player", "AreaWall1", "WallRunCollide",
false, 1  );
}
```

And this is the function that triggers when the players collides with the area:

```
void WallRunCollide ( string &in p, string &in c, int s ) {
    AddPlayerBodyForce( , 20000, , false );
}
```

2. Adjusting code +*Timer* +*Loop*

*BodyForce* function uses large numbers, so i'm using a variable instead:

```
void WallRunCollide ( string &in p, string &in c, int s ) {
    float M = 10000;
    AddPlayerBodyForce( , 2*M, , false );
}
```

This will push the player up just once, so i'm adding a *Timer+Loop* to repeat the impulse (5 times):

```
void WallRunCollide ( string &in p, string &in c, int s ) {
    float i = 0.25; int n = 5; float r = ;


    for ( int v = 1; v <= n; v++ ) {
        AddTimer ( "", r, "RunOnMe" );
        r = r +i;
    }
}
```

As you can see, the content changed, i moved it to another new function.
I added 3 variables, interval, number and a counter.
These are used in the *for loop.*
This loop will set a *Timer* to call the function "*RunOnMe*" *n* times, on an interval of *i* seconds.

Below is the new function i created that the *Timers* will trigger:

```
void RunOnMe ( string &in t ) {
    float M = 10000;
    AddPlayerBodyForce( , 2*M, , false );
}
```

This is supposed to be a step, i'm adding a sound:

```
void RunOnMe ( string &in t ) {
    float M = 10000;
    AddPlayerBodyForce( , 2*M, , false );
    PlayGuiSound( "step_run_female_rock.snt", 0.4 );
}
```

At this point, the code is working fine. You can test it out.

3. Preventions

Now we have to consider possible bugs.
We don't want to trigger this if the player didn't jump.
It would look weird and just fail cause there's not enough starting force.
We can avoid that using *GetPlayerYSpeed()* function, anything below 1 is not jumping:

```
void WallRunCollide ( string &in p, string &in c, int s ) {


    if ( GetPlayerYSpeed() <1 ) { return; }
    float i = 0.25; int n = 5; float r = ;


    for ( int v = 1; v <= n; v++ ) {
        AddTimer ( "", r, "RunOnMe" );
        r = r +i;
    }
}
```

That new condition i just added will stop the code by using the *return* function.
So player may walk into the wall area and not get thrown up by the wall running code.

4. Direction

We are adding now a small tweak to our code to let me player direct his wall running.
Using *AddPlayerBodyForce* we can add a slight push ahead: ( just to help the player go where he's looking at)

```
void RunOnMe ( string &in t ) {
    float M = 10000;
    AddPlayerBodyForce( , 2*M, , false );
    AddPlayerBodyForce( , , 0.7*M, true );
    PlayGuiSound( "step_run_female_rock.snt", 0.4 );
}
```

0.7* M should be enough, remember this will run on every step. There's 5.

5. Functionality

This is the end of this guide. With this code you can experiment new functionalities and expand its possibilities.
If you're interested, you can take a look at my code in any script file from my mod Riukka for more developed functions on wall running.

Amn.-