

Welcome to 3DGS A8 world of physics.

Physics are used to make objects move as natural as possible. They react to each other can fall, bounce off from each other. Basically it gives us the power to give mass,friction,elastic to models. Make sure that the nVidia PhysX engine is installed on your PC - this is normally the case through installing Gamestudio/A8.

In order to use all the physics 3DGS A8 gives us we need to add its script to ours.

Remember the standard scripts that are included. We need to add on other : ackphysx.h

```
//////////////////////////
#include <acknex.h>
#include <default.c>
#include <mtlFX.c>
#include <ackphysx.h> //<<<< here it is. Now we will be able to use it in our game.
//////////////////////////
Then we need to open it in our main function :
```

```
function main()
{
video_aspect = 1.333; // enforce 4:3 mode
video_screen = 2;////« starts game in window mode
video_mode = 8;///< starts game in 1024x 768 mode
physX_open();/// << here we open the physX so it's active after the level loads.
level_load ("workshop06.wmb"); ///«loads our level
}
```

Let's give some physics to an entity. I create a ball model named : cau7seway.mdl.
Place this ball in the level and give it this simple code :

```
VECTOR ball_force;  
ENTITY* ball;///<<< there is another pointer ☺
```

```
action physic_ball()  
{  
  set(my,SHADOW);///<<< ah yes the old shadow maker  
  ball=me;///<<<pointer  
  pXent_settype (ball, PH_RIGID, PH_SPHERE); // set the physics entity type so it acts like a sphere.  
  pXent_setfriction (ball,80); // set the friction on the ground  
  pXent_setdamping (ball,10,10); // set the damping  
  pXent_setelasticity (ball,50); // set the elasticity  
  while (1)  
  {  
    ball_force.x = (key_cur-key_cul)*5*time_step; // rotate the ball about the global x axis  
    ball_force.y = (key_cuu-key_cud)*5*time_step; // rotate the ball about the global y axis  
    ball_force.z = 0; // no need to rotate about the vertical axis  
    pXent_addtorqueglobal (ball, ball_force); // add a torque (an angular force) to the ball  
    camera.x = ball.x - 250; // keep the camera 300 quants behind the ball  
    camera.y = ball.y; // using the same y with the ball  
    camera.z = 250; // and place it at z = 1000 quants  
    camera.tilt = -25; // make it look downwards  
    wait(1);  
  }  
}
```

Give this code to our ball model and see what you have done ☺ that's cool you can move the ball around. It bounces when you fall of a wall or run into it.

Here is a bit more explanation :

pXent_settype

PH_STATIC = register the entity as a static obstacle.

PH_RIGID = register the entity as a dynamic rigid body that can move (model entities only).

PH_CHAR = register the entity as a character controller (model entities only).

0 = unregister the entity from the physics system.

PH_BOX - Rectangular, box-shaped hull.

PH_SPHERE - Round hull.///<<< the one we used for the ball

PH_CAPSULE - Cylindrical collision hull with a semisphere at the top and bottom.

PH_POLY - Polygonal hull, for static (**PH_STATIC**) actors only.

PH_CONVEX - Simple convex polygonal hull for dynamic actors - 256 polygons maximum.

PH_TERRAIN - Heightmap; **PH_POLY** often works better for terrain.

PH_PLANE - Ground plane at position **z=0**.

Add **PH_MODIFIED** for re-registering a modified **PH_POLY**, **PH_TERRAIN**, or **PH_CONVEX** hull.

Oh boy oh boy so were into physics programming now ☺

You could use the physics to make your own bowling or pool game. Or another angry bird variant, after all these games use physics as we use now.

I create a model called balk.mdl Place them around in the level stack them on top of each other.

Let's give this models the following code. Then run it and see how they react to the ball.

```
ENTITY* balk;
```

```
action ph_balk() // simple physics-based box
{
    balk=me;
    pXent_settype(my, PH_RIGID, PH_BOX); // this entity behaves like a box
    pXent_setfriction (balk,5); // set the friction on the ground
    pXent_setdamping (balk,80,80); // set the damping
    pXent_setelasticity (balk,100); // set the elasticity
}
```

Yes you did it. Now you're able to turn those models over with the ball and make them crumble to the ground. Experiment with the numbers of friction,damping en elasticity. See then what your changes accomplished.

Here is the explanation of these settings :

setfriction

Set the entity's friction for collisions. Friction is what makes objects stick to each other, when moved sideways. **friction** can be set to values between 0 and 100, with 0 effectively disabling friction (similar to what you experience when walking on ice) and 100 setting maximum friction (rubber on rubber). Whenever possible you should set friction to 0, as this speeds up the calculations and prevents instability. Setting friction to a value larger than 100 percent is faster than setting friction to some arbitrary value between 0 and 100, but it can introduce some instability problems when objects move at high speeds (e.g. don't set car wheels to 110% friction).

Setdamping

A simple way to simulate airdrag and gear friction is to use velocity damping. Each frame a force & torque gets applied to the selected entity, compensating its movement. An amount of 0 disables damping and unless stopped by other objects or forces, a moving physical entity will continue moving indefinitely. When set to 100%, the object will only move while a force or torque is applied to it.

Setelasticity

Set the coefficient of restitution for an Actor - 0 makes the object bounce as little as possible, higher values up to 100 result in more bounce.

Well there you have it you know have all the basics for using physics. Good luck in testing and trying it all out. All you learned in the previous lesson can be used to. Make a sound when the ball collides or add points when it hits something let your creativity guide you.

Until next time

René Pol aka Realspawn
Realspawn@live.nl