```
// Script that controls a car or other land-based vehicules in Second Life.        // Retrieved from Free SL Scripts on http://www.gendersquare.org/sl
// Encog's Magic Wagon
// Very simple vehicle script

float forward_power = 15; //Power used to go forward (1 to 30)
float reverse_power = -15; //Power ued to go reverse (-1 to -30)
float turning_ratio = 2.0; //How sharply the vehicle turns. Less is more sharply. (.1 to 10)
string sit_message = "Ride"; //Sit message
string not_owner_message = "You are not the owner of this vehicle ..."; //Not owner message

default
{
    state_entry()
    {
        llSetSitText(sit_message);
        // forward-back,left-right,updown
        llSitTarget(<0.2,0,0.45>, ZERO_ROTATION );

        llSetCameraEyeOffset(<-8, 0.0, 5.0>);
        llSetCameraAtOffset(<1.0, 0.0, 2.0>);

        llPreloadSound("car_start");
        llPreloadSound("car_run");

        //car
        llSetVehicleType(VEHICLE_TYPE_CAR);
        llSetVehicleFloatParam(VEHICLE_ANGULAR_DEFLECTION_EFFICIENCY, 0.2);
        llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_EFFICIENCY, 0.80);
        llSetVehicleFloatParam(VEHICLE_ANGULAR_DEFLECTION_TIMESCALE, 0.10);
        llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_TIMESCALE, 0.10);
        llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_TIMESCALE, 1.0);
        llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_DECAY_TIMESCALE, 0.2);
        llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_TIMESCALE, 0.1);
        llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_DECAY_TIMESCALE, 0.5);
        llSetVehicleVectorParam(VEHICLE_LINEAR_FRICTION_TIMESCALE, <1000.0, 2.0,
1000.0>);
        llSetVehicleVectorParam(VEHICLE_ANGULAR_FRICTION_TIMESCALE, <10.0, 10.0,
1000.0>);
        llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_EFFICIENCY, 0.50);
        llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_TIMESCALE, 0.50);
    }

    changed(integer change)
    {
```

```
    if (change & CHANGED_LINK)
    {

        key agent = llAvatarOnSitTarget();
        if (agent)
        {
            if (agent != llGetOwner())
            {
                llSay(0, not_owner_message);
                llUnSit(agent);
                llPushObject(agent, <0,0,50>, ZERO_VECTOR, FALSE);
            }
            else
            {
                llTriggerSound("car_start",1);


                llMessageLinked(LINK_ALL_CHILDREN , 0, "WHEEL_DRIVING", NULL_KEY);
                llSleep(.4);
                llSetStatus(STATUS_PHYSICS, TRUE);
                llSleep(.1);
                llRequestPermissions(agent, PERMISSION_TRIGGER_ANIMATION |
PERMISSION_TAKE_CONTROLS);

                llLoopSound("car_run",1);
            }
        }
        else
        {
            llStopSound();

            llSetStatus(STATUS_PHYSICS, FALSE);
            llSleep(.4);
            llReleaseControls();
            llTargetOmega(<0,0,0>,PI,0);

            llResetScript();
        }
    }

}

run_time_permissions(integer perm)
{
    if (perm)
```

```
        {
            llTakeControls(CONTROL_FWD | CONTROL_BACK | CONTROL_DOWN |
CONTROL_UP | CONTROL_RIGHT |
                        CONTROL_LEFT | CONTROL_ROT_RIGHT | CONTROL_ROT_LEFT,
TRUE, FALSE);
        }
    }

    control(key id, integer level, integer edge)
    {
        integer reverse=1;
        vector angular_motor;

        //get current speed
        vector vel = llGetVel();
        float speed = llVecMag(vel);

        //car controls
        if(level & CONTROL_FWD)
        {
            llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION,
<forward_power,0,0>);
            reverse=1;
        }
        if(level & CONTROL_BACK)
        {
            llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION,
<reverse_power,0,0>);
            reverse = -1;
        }

        if(level & (CONTROL_RIGHT|CONTROL_ROT_RIGHT))
        {
            angular_motor.z -= speed / turning_ratio * reverse;
        }

        if(level & (CONTROL_LEFT|CONTROL_ROT_LEFT))
        {
            angular_motor.z += speed / turning_ratio * reverse;
        }

        llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION, angular_motor);

    } //end control
```

```
} //end default
```