// Fixit Galatea generic boat script

// Motor Yacht v0.82
// This script uses the basic direction keys for boat operation and requires
// the final prim to be oriented at rotation <0, 0, 0>. The handling
// Version of this script
float version = 0.82;

```
//*****************************************************************************
//*****************************************************************************
//*****************************************************************************
//* The following items can be edited in this script. See the comment for each
//* item to see how it work when it is changed.
//*****************************************************************************
```

```
//* When the forward key is pressed and the mouse button is pressed at the
//* same time, this speed is used. This number should be between 0.0 and 30.0,
//* with higher numbers giving faster performance.
float fastspeed = 30;
```

```
//* When the forward key is pressed and the mouse button is NOT pressed, this
//* speed is used. This number should be between 0.0 and whatever the
//* fastspeed value was set at, with higher numbers giving faster performance.
float slowspeed = 30;
```

```
//* This is the reverse speed. It shouldn't be too fast, but can be between
//* 0.0 and 30.0.
float backspeed = 10.0;
```

```
//* This is the vehicle 'hover' height, or how far it sits out of the water.
```

```
//* Adjust this as necessary for your boat.
float floatheight = 0.75;



//* If the boat is to ramp up speed slowly, set the following value to '1'.
//* Otherwise, the board will instantly reach full speed when the forward
//* key is pressed.
integer fastforward = 0;  // NOT IMPLEMENTED YET



//* Turning speed, a small turning speed is best
float turnspeed = 1.5;



//* If the boat 'angles' into turns, set the following value to '1'.
//* Otherwise, the boat will do flat turns.
integer angledturn = 5;   // NOT IMPLEMENTED YET



//* Height offset of seat
vector seatheight = <0.5, -0.55, 0.35>;



//* Viewing position
vector viewpos = <-10, 0, 3>;



//*****************************************************************************



// Initialize the vehicle as a boat
vehicle_init()
{
    llSetVehicleType(VEHICLE_TYPE_BOAT);



    // least for forward-back, most friction for up-down
    llSetVehicleVectorParam(VEHICLE_LINEAR_FRICTION_TIMESCALE, <10, 1, 3>);
```

```
// uniform angular friction (setting it as a scalar rather than a vector)
llSetVehicleFloatParam(VEHICLE_ANGULAR_FRICTION_TIMESCALE, 0.1);


// linear motor wins after about five seconds, decays after about a minute
llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION, <0, 0, 0>);
llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_TIMESCALE, 0.1);
llSetVehicleFloatParam(VEHICLE_LINEAR_MOTOR_DECAY_TIMESCALE, 0.05);


// agular motor wins after four seconds, decays in same amount of time
llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION, <0, 0, 0>);
llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_TIMESCALE, 0.1);
llSetVehicleFloatParam(VEHICLE_ANGULAR_MOTOR_DECAY_TIMESCALE, 0.2);


// hover
llSetVehicleFloatParam(VEHICLE_HOVER_HEIGHT, floatheight);
llSetVehicleFloatParam(VEHICLE_HOVER_EFFICIENCY, 0.2);
llSetVehicleFloatParam(VEHICLE_HOVER_TIMESCALE, 0.4);
llSetVehicleFloatParam(VEHICLE_BUOYANCY, 1.0);


// Slight linear deflection with timescale of 1 seconds
llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_EFFICIENCY, 0.1);
llSetVehicleFloatParam(VEHICLE_LINEAR_DEFLECTION_TIMESCALE, 1);


// Slight angular deflection
llSetVehicleFloatParam(VEHICLE_ANGULAR_DEFLECTION_EFFICIENCY, 0.1);
llSetVehicleFloatParam(VEHICLE_ANGULAR_DEFLECTION_TIMESCALE, 6);


// somewhat bounscy vertical attractor
llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_EFFICIENCY, 0.5);
llSetVehicleFloatParam(VEHICLE_VERTICAL_ATTRACTION_TIMESCALE, 1);


// weak negative damped banking
```

```
    llSetVehicleFloatParam(VEHICLE_BANKING_EFFICIENCY, 1.0);
    llSetVehicleFloatParam(VEHICLE_BANKING_MIX, 1.0);
    llSetVehicleFloatParam(VEHICLE_BANKING_TIMESCALE, 0.1);



    // default rotation of local frame
    llSetVehicleRotationParam(VEHICLE_REFERENCE_FRAME,
        llEuler2Rot(<0, 0, 0>));



    // remove these flags
    llRemoveVehicleFlags(VEHICLE_FLAG_HOVER_TERRAIN_ONLY
                       | VEHICLE_FLAG_LIMIT_ROLL_ONLY
                       | VEHICLE_FLAG_HOVER_GLOBAL_HEIGHT);
    // set these flags
    llSetVehicleFlags(VEHICLE_FLAG_NO_DEFLECTION_UP
                    | VEHICLE_FLAG_HOVER_WATER_ONLY
                    | VEHICLE_FLAG_HOVER_UP_ONLY
                    | VEHICLE_FLAG_LIMIT_MOTOR_UP);
}



// Setup everything
all_setup()
{
    // Display version number
//    llWhisper(0, "Motor Yacht script " + (string) version);
    llSetStatus(STATUS_PHYSICS, FALSE);



    // Set sit direction (forward) and sight location slightly up and behind
    llSitTarget(seatheight, llEuler2Rot(<0, 0, 0>));
    llSetCameraAtOffset(<6, 0, 6>);
    llSetCameraEyeOffset(viewpos);
    llSetSitText("Ride!");



    // Initialize vehicle states
    vehicle_init();
```

```
    // Set up listener callback function
    llListen(0, "", llGetOwner(), "");
}



// State (default) event handlers
default
{
    state_entry()
    {
        all_setup();
    }



    on_rez(integer start_param)
    {
        llSetStatus(STATUS_PHYSICS, FALSE);
    }



     run_time_permissions(integer permissions)
    {
        // Get user permissions
        if ((permissions & PERMISSION_TAKE_CONTROLS) ==
            PERMISSION_TAKE_CONTROLS)
        {
            llTakeControls(CONTROL_ML_LBUTTON | CONTROL_FWD |
                CONTROL_BACK | CONTROL_LEFT | CONTROL_RIGHT |
                CONTROL_ROT_LEFT | CONTROL_ROT_RIGHT, TRUE, FALSE);
        }
    }



    changed(integer change)
    {
        if (change & CHANGED_LINK)
        {
            key agent = llAvatarOnSitTarget();
            if (agent)
            {
                if (agent != llGetOwner())
                {
```

```
                llSay(0, "This boat isn't yours, but you can buy a copy!");
                llUnSit(agent);
            }
            else
            {
                llRequestPermissions(agent, PERMISSION_TAKE_CONTROLS);
                llSetStatus(STATUS_PHYSICS, TRUE);
            }
        }
        else
        {
            llReleaseControls();
            llSetStatus(STATUS_PHYSICS, FALSE);
        }
    }
}



control(key name, integer levels, integer edges)
{
    float side = 0.0;
    float forw = 0.0;
    float move = 0.0;
    float turn;



    if (levels & CONTROL_ML_LBUTTON)
    {
        move = fastspeed;
        turn = 1.5 * turnspeed;
//          forw = 2 * PI / 3;
    }
    else if (levels & CONTROL_FWD)
    {
        move = slowspeed;
        turn = 1.0 * turnspeed;
//          forw = PI / 2;
    }
    else if (levels & CONTROL_BACK)
    {
        move = -backspeed;
//          forw = -PI / 3;
//          turn = -1.0 * turnspeed;
    }
```

```
    if (levels & (CONTROL_LEFT | CONTROL_ROT_LEFT))
    {
        if (move == fastspeed)
        {
            side = turnspeed;
//              forw = PI;
        }
        else if (move != 0)
        {
            side = turnspeed;
//              forw = PI / 3;
        }
        else
        {
            side = .67 * turnspeed;
//              forw = PI / 4;
            move = 0.1;
        }
    }
    else if (levels & (CONTROL_RIGHT | CONTROL_ROT_RIGHT))
    {
        if (move == fastspeed)
        {
            side = -turnspeed;
//              forw = PI;
        }
        else if (move != 0)
        {
            side = -turnspeed;
//              forw = PI / 3;
        }
        else
        {
            side = -.67 * turnspeed;
//              forw = PI / 4;
            move = 0.1;
        }
    }



    if (move == 0)
    {
        llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION, <0, 0, 0>);
    }
    else
```

```
        {
            llSetVehicleVectorParam(VEHICLE_LINEAR_MOTOR_DIRECTION,
                <move, 0, 0>);
            llSetVehicleVectorParam(VEHICLE_ANGULAR_MOTOR_DIRECTION,
                <-side / 5, 0, side>);
        }
    }



    listen(integer channel, string name, key id, string message)
    {
        list params = llParseString2List(message, [" "], [":"]);
        string cmd = llList2String(params, 0);
        integer enable = llList2Integer(params, 1);



        // Commands
        ;
    }



    touch_start(integer total_number)
    {
        key id = llDetectedOwner(total_number - 1);
       // llGiveInventory(id, "Info card");
    }
}
```