Hovercraft physics package

This package contains a few components that simulate a player-controlled hovercraft. A hovercraft hovers (duh!) above ground, suspended by force of huge fans blowing downwards. Or maybe by magnetic levitation, alien plasma technology or simply magic – whatever your game needs. There are also a few scripts that add effects to the hovercraft – particles, fans' rotation, engine sound; and a simple hovercraft model. In short, just drop a prefab into your scene – and you have a fully-configured, player-controlled hovercraft ready to rumble.

Hovercraft composition

Hovercraft physics is created using several components. First of all, hovercraft must have a `Collider` and a `Rigidbody` for the physics to work at all.

`HoverOrientation` script goes on the same object as the `Rigidbody` and controls rotation of the hovercraft, like turning and auto-leveling.

`MovementEngine` script controls forward (or lateral) movement. You can place one on the `Rigidbody` object for forward movement, and add another for strafing, should you need one. `MovementEngine` propels the hovercraft along *engine's* Z axis, so for strafing engine should be facing left or right.

`HoverEngine` script actually does the hovering. It pulls the `Rigidbody` upward, with force depending on height above ground. This force is largest when engine touches ground directly, and falls to zero at some predetermined height. `HoverEngine` applies force at its own position, not at `Rigidbody` center of mass (contrary to `MovementEngine`). This means that you should either place it at the center of mass, or use several `HoverEngines` to balance the hovercraft. The example prefab uses three, and this works nicely.

`HoverControl` script is really simple, but important. All it does is routes player input to other components, so that the player can control the hovercraft. The separate script allows to change control easily – for example, create an AI that controls a hovercraft using the same physical model.

This is basically all that's required for a working hovercraft. A few more scripts can be used to add effects, read about them below. Also, be sure to check the source code and comments there.

Detailed components description

CameraHandle

This script is not for the hovercraft, but for the camera – it handles following the craft and orbiting it with the mouse. See the example scene to understand how it's set up – basically, CameraHandle is put on an empty game object and causes it to follow the Tracked object. Camera itself is attached to this object (through an intermediary transform) – this way you can define camera position and orientation in Inspector, and it stays the same (relative to the hovercar) throughout the game.

In addition to following, CameraHandle allows for mouse orbiting target. When mouse is not moved for some predefined time, camera is (smoothly) returned to default rotation.

DustParticles

This script places dust particles under hover engines. Set up some particle effect, attach it to the hovercraft gameobject (so that it does not get lost), and put this component on it. It would automatically place the effect on the ground below the engine, and also scale particle emission rate based on engine power.

FlipHelper

This script flips the hovercraft in case it ends up upside down, so that you don't have to restart the game.

HoverControl

HoverControl routes player input to the engines. Put in on the hovercraft object and don't forget to link HoverOrientation and MovementEngine objects to it. This script does not support strafing engine, you can easily add it yourself if you need one.

HoverEngine

This script actually creates hover force. It works by raycasting downwards, until ground is hit. If it is, the engine script then pushes the hovercraft upwards, with force dependent on the height. The "downwards" direction is a bit complicated. When a hovercraft rotates, its engines rotate too, and the push-force should rotate too. However, if the force is not directed strictly upwards (against gravity), its sideways component causes hovercraft to drift. This means that in-game, the hovercraft never stands still, because of all the little rotations, imbalances, uneven ground etc. To combat this, HoverEngine adds a little "angle drift" - the actual force direction, which should be along local Y axis, is rotated towards global Y axis. The rotation is limited by MaxAngleDrift property. This allows to keep kinda realistic behavior and get rid of drift on mostly-even surfaces. Here's a picture to help imagine all this:

The amount of push-off force is determined by height above ground. Maximum force, set up through GroundForce property, is applied when the engine touches ground directly (this actually never happens, as the size of the engine collider should prevent it from being at 0 height). The MaxHeight property defines height at which force falls down to zero, and Exponent defines the falling curve: Exponent==1 means linear falloff, Exponent==2 – quadratic etc.

The Damping property defines additional damping force that engine applies. The way we've set up push-off force, it basically works like a spring. If there is no damping to the spring, our hovercraft would bounce up and down constantly; this property reduces the bouncing effect.

RaycastMask property allows to select what layers are considered “ground” for the hover engine. You might want to exclude some FX layers to keep hover from pushing off them.

Rigidbody property is the hovercraft's rigidbody. Don't forget to set this up!

HoverOrientation

This script does two things. First, it turns the hovercraft left and right as directed by Turn property (which is set up by HoverControl or some other control script). Second, it applies auto-leveling, so that the hovercraft always tends to stand upright.

Turning is controlled by properties MaxTorque and BrakingTorque. MaxTorque is the torque to apply when actively turning (i.e. when player presses the left or right button). BrakingTorque is applied to counter rotation when the button is *not* pressed (i.e. Turn==0). The larger the torque, the faster hovercraft turns (or stops turning, in case of BrakingTorque).

The RollOnTurns property defines additional roll angle when turning. It causes the hovercraft to rotate sideways, simulating “leaning-in” on turns – like a motorcycle. It looks nice, although does not serve any other purpose. RollOnTurnsTorque is the torque that causes this roll – the larger it is, the faster roll happens.

RollCompensationTorque and PitchCompensationTorque control auto-leveling system. Whenever the hovercraft leans in any direction, this system attempts to rotate it back, applying these torques. Using high values would make the hovercraft really steady and hard to overturn. The HoverEngines array is used to limit auto-leveling when hovercraft is too high. If you put links to any hover engines here, auto-leveling will stop working whenever any of these engines stops (i.e. when it is higher than MaxHeight).

HoverSound

This script changes volume and pitch of engine sound depending on engine power. Power of a HoverEngine is 1 when touching ground (force is at maximum), and 0 when force is 0. Power of a MovementEngine is simply its Thrust (i.e. how fast the player wants to go). HoverSound takes maximum hover power, maximum movement power, weighs them according to MovementEngineWeight property, and scales volume and pitch according to the result.

MovementEngine

This script actually propels the hovercraft forward (or sideways, if you so choose). The Thrust property is used to accelerate/decelerate; it is meant to be set by HoverControl or some other control script. MovementEngine applies force to the hovercraft center of mass, so it can never cause rotation. The direction of the force is determined by engine's transform, so make sure it faces right where you want to go (force is applied in direction of positive Z axis, to be specific)

Acceleration is controlled by properties MaxForwardAcceleration, MaxReverseAcceleration and MaxBrakingDeceleration. These define maximum values, that are multiplied by Thrust value, and further by AccelerationBySpeed curve. This curve defines acceleration as a function of speed, where 1 on curve is MaxSpeed. The curve allows for fast acceleration when starting, and then slowly reaching maximum speed, for example.

MaxSpeed property defines maximum speed, obviously. It is possible for a hover to go faster than MaxSpeed, but MovementEngine will not be working in this case (i.e. hovercraft can go faster if accelerated by something else, like gravity).

The AutoBrakingDeceleration property is the deceleration that is applied “be default” when no buttons are pressed and Thrust==0. This makes hovercraft stop when not accelerating – probably not realistically, but conveniently.

The MaxPitchAngle and VerticalReduction properties together prevent hovercraft from climbing on near-vertical walls. By default MovementEngine will happily accelerate the hovercraft in any direction, including upwards. But if hovercraft pitch angle is more than MaxPitchAngle, engine acceleration is reduced. The reduction scales linearly from MaxPitchAngle to 90º – so that at exactly 90º, acceleration is divided by VerticalReduction.

HoverEngines array can be used to prevent MovementEngine from working when hovercraft is too high (just as with HoverOrientation). If any engine in this array has no power (i.e. is higher than MaxHeight), MovementEngine will stop working. This prevent hovercraft from, basically, flying.

Rigidbody property is the hovercraft's rigidbody. Don't forget to set it, unless you put the MovementEngine component directly on the rigidbody itself.

SideDependentDrag

This script applies drag to the rigidbody that is dependent on its orientation. You can use it to make hovercraft have more drag in sideways direction than in forward – this makes it easier to handle. The DragCoeffs property defines drag coefficients

per-axis. You don't *need* to set additional drag on rigidbody when using this script, but you can.

TurbineRotation

This script animates hovercraft's turbines according to engine power. The script is meant to be attached to the turbine mesh itself. Link it to the hover or movement engine (or both – maximum power of the two would be used), and set up MinRPM and MaxRPM rotation speeds. The rotation is around local Y axis.