Aerodynamic Objects

Anyone can make things fly

Read Me

Last updated 20/12/2024 15:45:00

Welcome to Aerodynamic Objects – a tool to help you make things fly in Unity

What is it?

Aerodynamic Objects is a set of tools for real-time modelling of aerodynamic behaviour of physical objects.

Features include:

- Aerodynamics engine for calculating aerodynamic forces acting on objects
- Easy integration with Unity's physics engine
- Custom flow field models to create wind or wakes
- Visualisation tools to provide visual feedback for developers and players
- Expanding range of design tools to support development of complex assemblies such as aircraft, sail boats or flying animals

Aerodynamic Objects uses real physics with simplifying assumptions to make it both fast and easy to use. It will give an answer that is exactly right sometimes, but more importantly is guaranteed to give an answer that is plausible at all times. This means that whatever you ask the model to do, it will always give a usable answer.

Integration with the Unity physics engine is simple; the AeroObject component can be connected to a Rigidbody component to apply aerodynamic forces to that rigid body. This means you can extend existing models and game rigs in Unity to include aerodynamic behaviours as well as build new models from scratch.

Our design philosophy with Aerodynamic Objects is that users should be able to create interesting and complex models and behaviours driven by aerodynamics without having to have in-depth knowledge of the subject. Simple things should be easy to do immediately; complex things should be possible with experience. If you are used to working in traditional animation, there is a step up to physics-based animation (simulation). Physics-based animation is a hugely powerful creative tool, but you need to develop understanding to be able to direct it as you wish.

Online Resources

An Introduction to Aerodynamic Objects video tutorial series is available to help you learn the basics. It assumes the user has some background understanding of high-school physics and a working knowledge of the Unity interface. The tutorials start with the basics of simulating wind and simple objects, then move on to building up the components of a simple

but fully flyable propeller-driven aircraft. Scripting is not covered in the tutorials – where simple scripts are needed these are supplied readymade. Completed versions of the tutorials are also provided as Unity scenes as part of the distributed Unity package.

Getting started **video tutorials**:

https://www.youtube.com/playlist?list=PLohJe_Z4v94fXY7eGGRT3lbZg6NBrhAwg

Further **examples** can be found on:

www.AerodynamicObjects.com

Full code documentation can be found at:

https://conorzam.github.io/AerodynamicObjectsDocs/

Getting Started

This section provides a quick start guide to using Aerodynamic Objects to apply aerodynamic forces to objects, visualise the forces being applied, and to set up flow primitives to define fluid velocities throughout the scene.

We have a demos folder with scenes showing what can be done with the tools as well as prefab models that can be explored.

There is also a GameObject menu for creating various Aerodynamic Object tools and models. This can be accessed via the GameObject > Aerodynamic Objects menu at the top of the Unity editor, or by right clicking in the scene hierarchy and selecting Aerodynamic Objects.

Please ensure the Default Max Angular Speed is set appropriately.

Unity by default has a maximum of 7 rad/s for any rigid body's angular speed. This is far too low for objects like propellers which spin at thousands of rotations per minute. It is advised that you set the default max angular speed to infinity, though a high value such as 10,000 may suffice.

To set the default max angular speed, go to: Edit > Project Settings > Physics > Default Max Angular Speed

Aero Objects

The core functionality of Aerodynamic Objects comes from the AeroObject class. The AeroObject class calculates the chosen aerodynamic forces for an object based on the transform scale and the dimensions provided in the class.

The aerodynamic forces include:

- **Drag** – also known as air resistance, this force slows down an object moving through a fluid. It always acts in the opposite direction to the object's velocity relative to the fluid
- **Lift** – objects which have a low thickness relative to their length, such as wings, can produce lift. Lift acts perpendicularly to the object's relative velocity.
- **Buoyancy** – this force arises due to differences in density between an object and a fluid. If an object has a lower density than the fluid it is in, it will float.
- **Rotational Damping** – this is a torque which slows down an object with angular velocity.

- **Rotational Lift** – also known as the Magnus effect, this is a force which arises when a spinning object moves through a fluid. It is advised to exclude this force unless the application specifically requires it.

Aero Objects derive from the Flow Sensor class, which is used to detect the relative flow velocity for an object. The relative flow velocity can be calculated either by the velocity of a rigid body or by calculating the rate of change of the attached transform's position. Flow Sensors themselves derive from the Flow Affected class which is used to interact with Flow Primitives.

An AeroObject will always apply forces to a Rigidbody if it is provided a reference.

Arrows

Each aerodynamic force has an associated arrow for visualisation purposes. For example, the DragArrow component will render an arrow which points in the direction of the object's drag force and will have a length relative to the magnitude of the drag force. Other arrow components include wind velocity and the weight of the object.

To use the arrow components, attach them to the same GameObject as the AeroObject you want to visualise forces for.

Flow Primitives

Flow primitives are used to define the behaviour of the flow in Aerodynamic Objects. They do this using a velocity function which takes a position in the world and provides the velocity of the flow at the position.

We provide a range of example flow primitives including:

- uniform flow
- source/sink
- area source
- vortex filament
- ring vortex
- line source

These can be added by attaching the relevant script to an object in the scene or by right clicking in the scene hierarchy and navigating to Aerodynamic Objects > Flow Primitives

Users can create their own custom flow primitives by inheriting from the base FlowPrimitive class and overriding the velocity function.

Any object which inherits from the Flow Affected class will interact with flow primitives.

Flow primitives by default will affect everything in the scene. They can be contained using a Fluid Volume. By setting a fluid volume as a parent of a flow primitive, the fluid volume can be used to define the bounding volume for the flow primitive. Fluid volumes can contain any number of flow primitives and are useful to avoid unnecessary calls to the velocity function of flow primitives that are far away from other objects in the scene, or for separating flow primitives that don't need to affect certain objects.

Interaction between flow primitives and objects can be further managed using the Flow Interaction Manager. All flow primitives and flow affected objects also have functions to ignore their counterparts, e.g. *FlowAffected.IgnoreInteraction(flowPrimitive)*