How to Fly:
Integrating Simple Aerodynamics into Game Physics Engines

Graham Rhodes

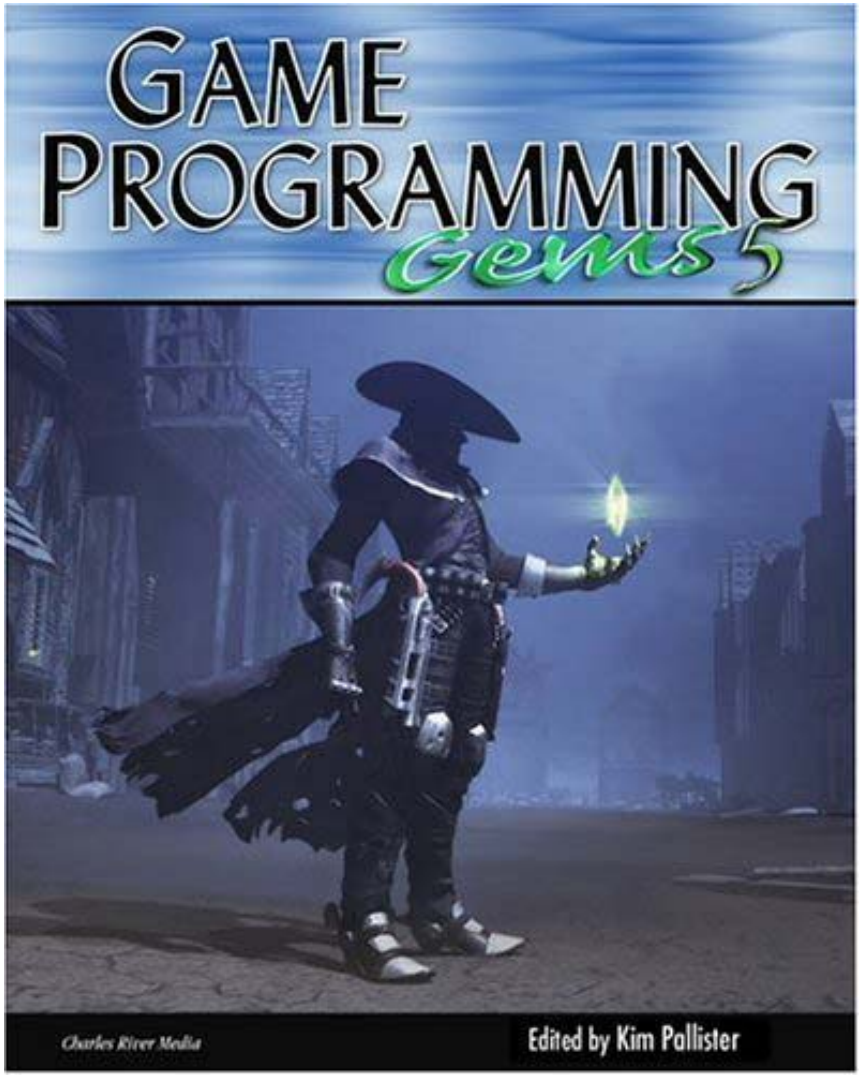
Applied Research Associates, Inc.

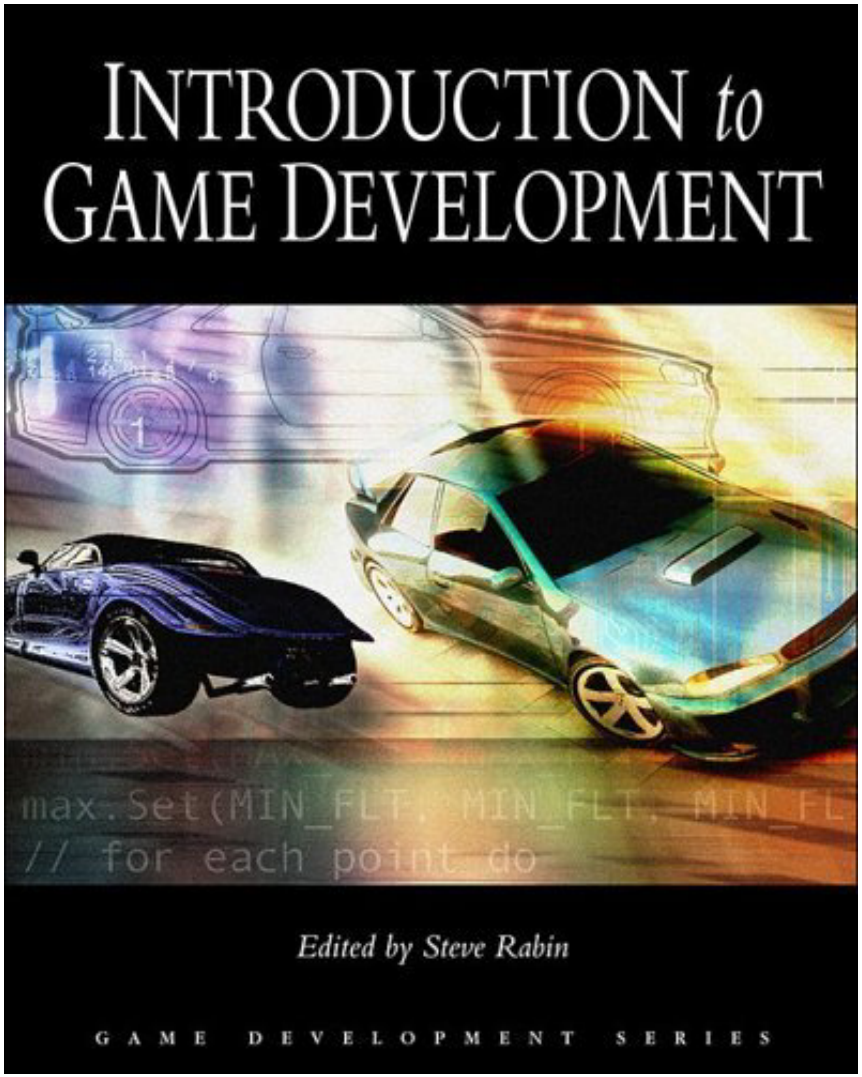
25 June 2005 - DgXpo - Raleigh, NC

Copyright (c) 2005 Graham Scott Rhodes

Shameless Plug

Game Programming Gems 5/Intro to Game Dev





Copyright (c) 2005 Graham Scott Rhodes

Outline

- **What/Why?**
- **Rigid Body Dynamics**
- **What are Physics Engines?**
- ***Simple Aerodynamic Formulas**
- **Computing for Physics Engines**
- **WIP: Integration with ODE + NovodeX**
- **Resources**

Copyright (c) 2005 Graham Scott Rhodes

What?

- **Real-time Physics**
- **Physics in Simulation Games**
  - Airplane physics - Flight simulators
  - Vehicle physics - Racing games
  - Rigid body dynamics
- **Games in General**
  - Rigid Body Dynamics
    - Tumbling, Sliding, Colliding Objects

Copyright (c) 2005 Graham Scott Rhodes

Why?

- **The Human Experience**
  - Real-world motions are physically-based
  - Physics can make simulated game worlds appear more natural
  - Makes sense to strive for physically-realistic motion for some types of games
- **Emergent Behavior**
  - Physics simulation can enable a richer gaming experience

- **ODE and NovodeX Demos**

Copyright (c) 2005 Graham Scott Rhodes

Rigid Body Dynamics on Two Pages

- **Basic Idea**
  - Use **F = ma** to move objects in game
  - That's Newton's 2nd Law of Motion
    - F = force; m = mass; a = acceleration
- **Equations of Motion**

$$\mathbf{a}(t) = \frac{d}{dt}\mathbf{V}(t) = \frac{\mathbf{F}(t)}{m}$$

Copyright (c) 2005 Graham Scott Rhodes

Rigid Body Dynamics on Two Pages

- **Numerical Simulation**

  1) Calculate force on object

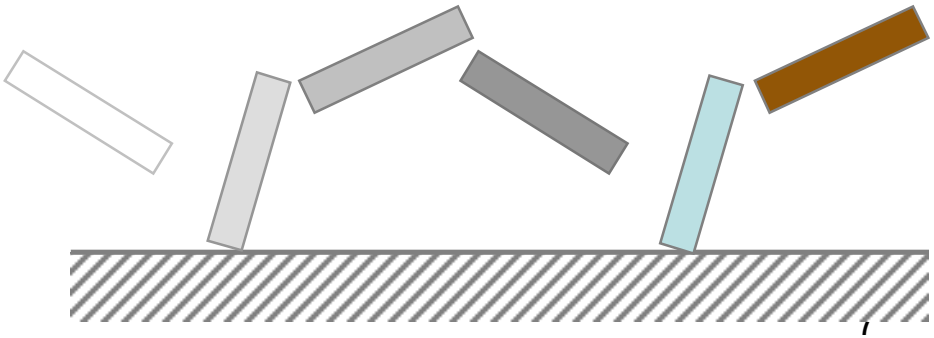
  2) "Integrate" the Equations of Motion

$$\mathbf{V}(t+\Delta t) = \mathbf{V}(t) + \Delta t \frac{\mathbf{F}(t)}{m}$$

$$\mathbf{p}(t+\Delta t) = \mathbf{p}(t) + \Delta t \mathbf{V}(t+\Delta t)$$

  3) Apply new position to object onscreen

- **Details, Details…**
  - Collisions
  - Rotation/spinning



Copyright (c) 2005 Graham Scott Rhodes

Physics Engines on a Page

- **What is a Physics Engine?**
  - Middleware API for simulating physics
  - Handles most of the dirty work
  - Must be wired to a graphics engine
- **Built-in Force Calculations**
  - Force due to gravity (weight of object)
  - Collision response forces
  - Friction
  - Spring/damper forces

Copyright (c) 2005 Graham Scott Rhodes

What's Missing?

- **Tumbling Crates are Nice, but…**
  - Worlds full of crates look dead except when player is blowing things up
  - Wouldn't you rather have a game world that looks alive?
- **The addition of wind effects can make game worlds appear more alive**
- **All that is required is to calculate additional forces during simulation**

Copyright (c) 2005 Graham Scott Rhodes

Lets explore some simple aerodynamic formulas that are suitable for approximating forces due to wind

Copyright (c) 2005 Graham Scott Rhodes

Don’t Be Afraid!

- **Formulas are Simple Algebra**
- **Some Vectors/Matrices…..Arrrggg**
- **Formulas provided here...**
  - Are very approximate
  - Subsonic flow only
  - Technically, developed for static objects in a steady wind or no wind
  - Don’t treat object-object interactions and interference
  - Don’t work too well for rapid, spastic motion such as plunging

Copyright (c) 2005 Graham Scott Rhodes

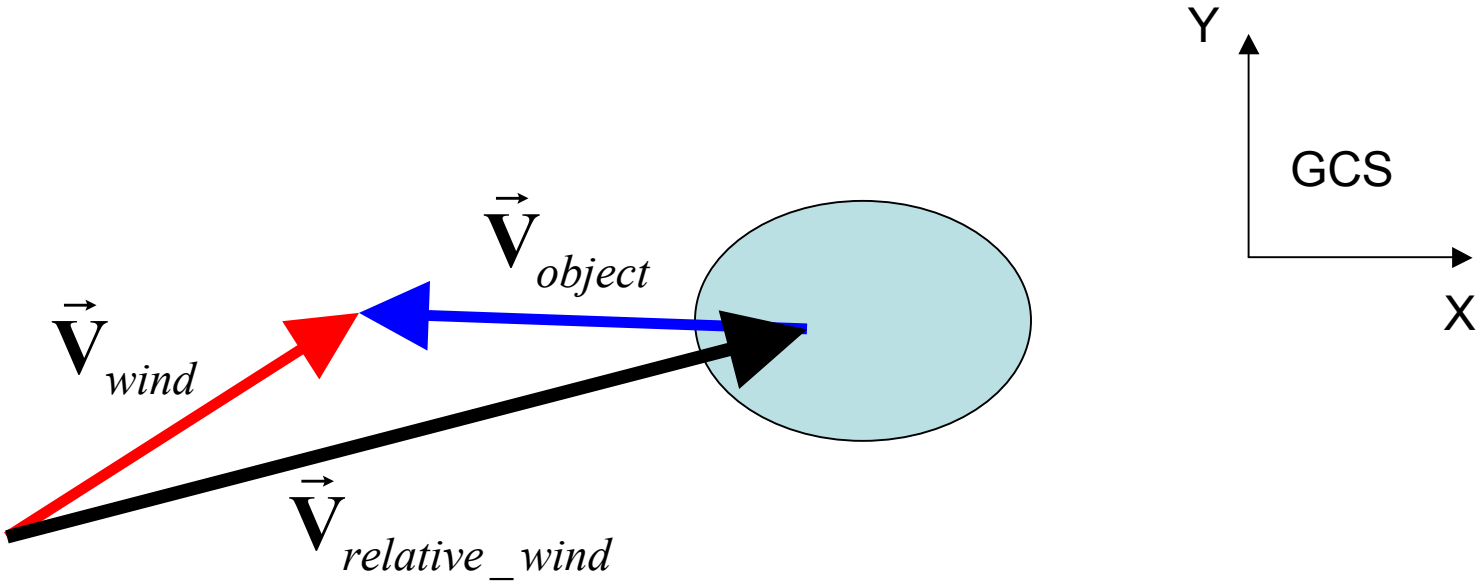
Common Parameters: "Reynold's Number"

$$R_e = \frac{\rho V l_{ref}}{\mu}$$

- ρ = Fluid density
  - 1.225 kg/m^3 for air at sea level
  - 1000 kg/m^3 for water at 20°C
- μ = Fluid dynamic viscosity
  - 1.789 x 10^{-5} Newton-seconds/m^2 for air
  - 1.0 x 10^{-3} Newton-seconds/m^2 for water
- V = relative wind speed, in meters/second
- l_{ref} = a reference length, in meters

Copyright (c) 2005 Graham Scott Rhodes

Common Parameters - Relative Wind



$$\vec{V}_{relative\_wind} = \vec{V}_{wind} - \vec{V}_{object}$$

Copyright (c) 2005 Graham Scott Rhodes

Aerodynamic Primitives

- **What the….**
- **Simple shapes that approximate game objects**
- **Used to compute aerodynamic forces**
- **Analogous to Collision Bounding Volumes**
- **Sphere**
- **Cylinder**
- **Flat Plate**

Copyright (c) 2005 Graham Scott Rhodes

Fluid Dynamic Drag on Bluff Bodies

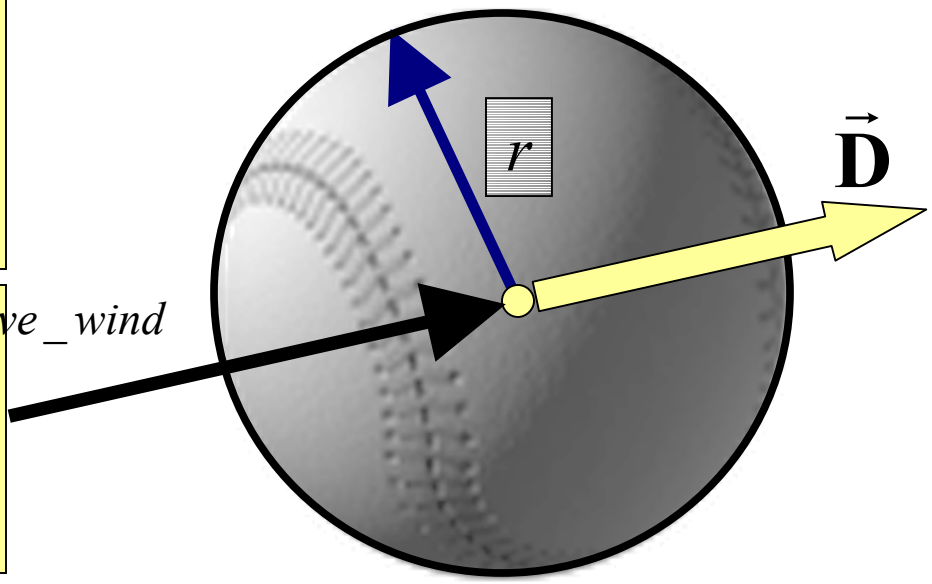
- **Primitive Shape: The Sphere**
- **The Formulas:**

$$\vec{\mathbf{D}} = \frac{\rho V^2 S_{ref} C_D}{2} \frac{\vec{\mathbf{V}}_{relative\_wind}}{V}$$

$$C_{D,sphere} = \frac{24}{R_e} + \frac{6}{1+\sqrt{R_e}} + 0.4$$

$$V = \left|\vec{\mathbf{V}}_{relative\_wind}\right|$$

$$S_{ref} = \pi r^2$$

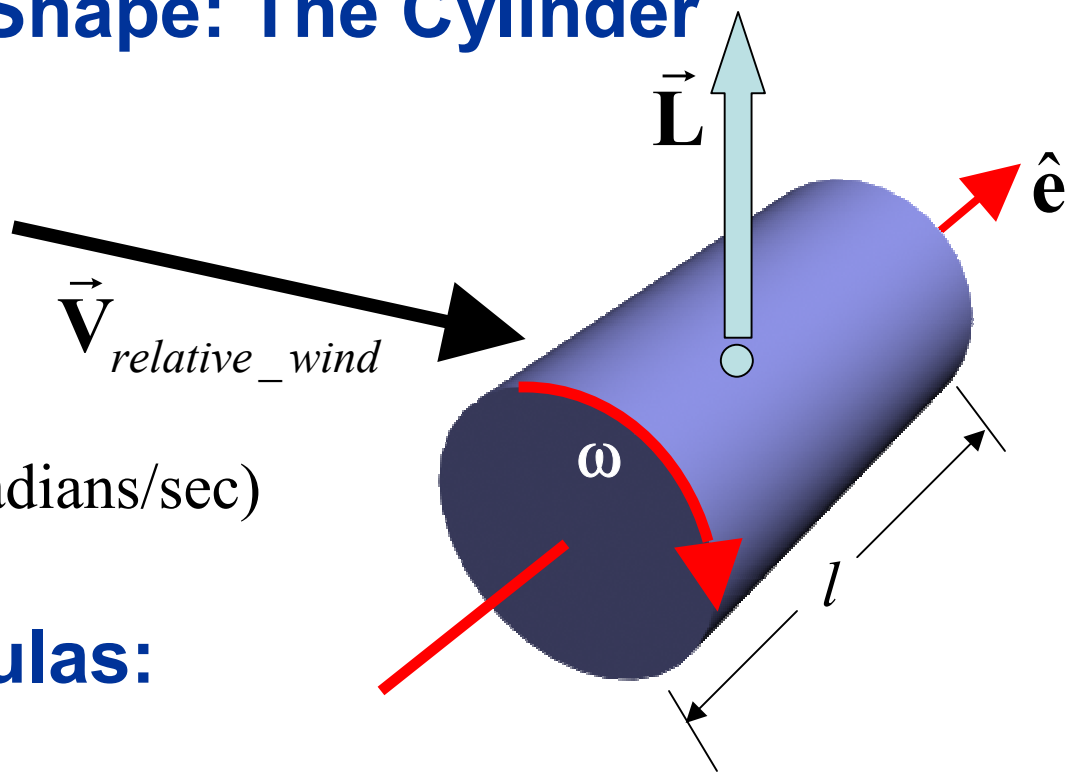


Works for Other Shapes Too!

Copyright (c) 2005 Graham Scott Rhodes

Lift on Spinning Bluff Bodies

- **Primitive Shape: The Cylinder**



ω = rate of spin (radians/sec)

- **The Formulas:**

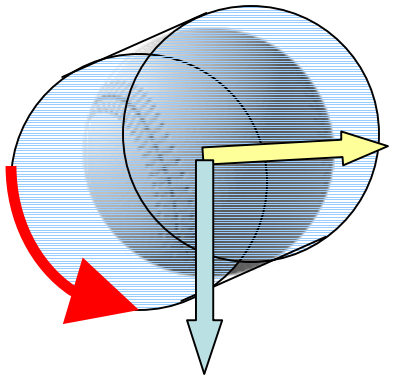
$$\vec{\mathbf{L}} = 0.785\, l \rho \vec{\mathbf{V}}_{relative\_wind} \times (2\pi\omega r^2 \hat{\mathbf{e}}_{spin})$$

Cross Product

Copyright (c) 2005 Graham Scott Rhodes

Bluff Body Forces

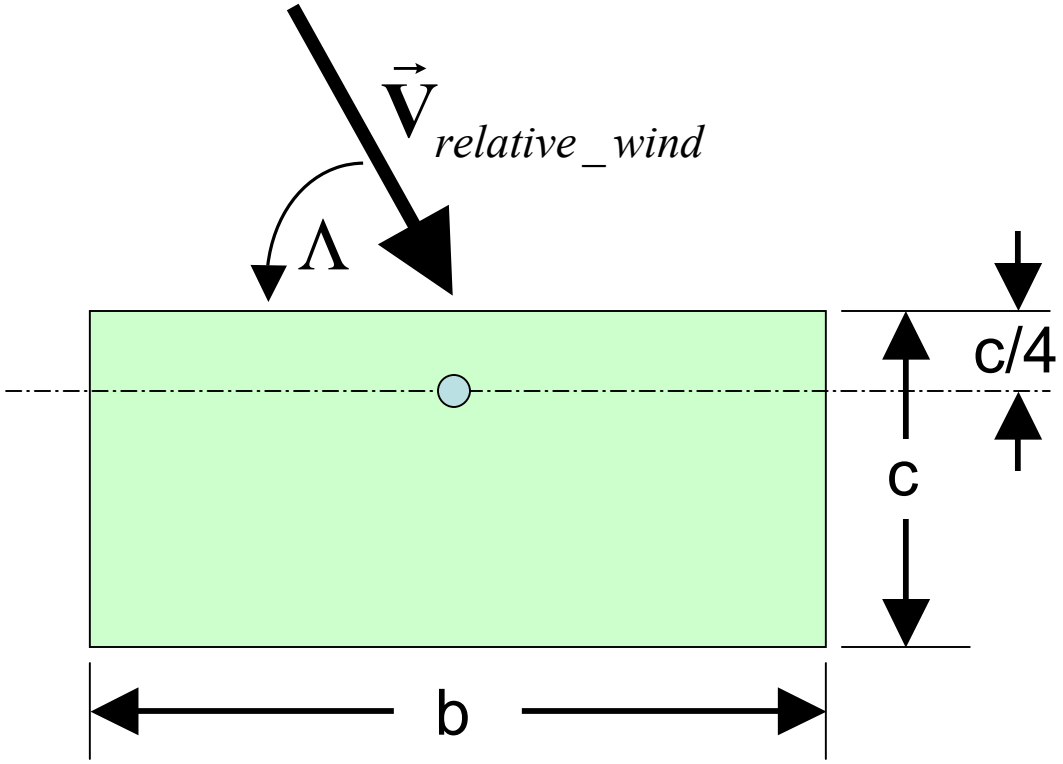
- **What the….**
- **Example Uses in Games…**
  - Wind-blown particles (demo)
  - Spinning Baseball
    - Choose a bounding cylinder
    - Align with spin axis
    - Compute bluff body drag ***and*** lift
    - For baseball, bluff body lift approximates the "Magnus Force"



Copyright (c) 2005 Graham Scott Rhodes

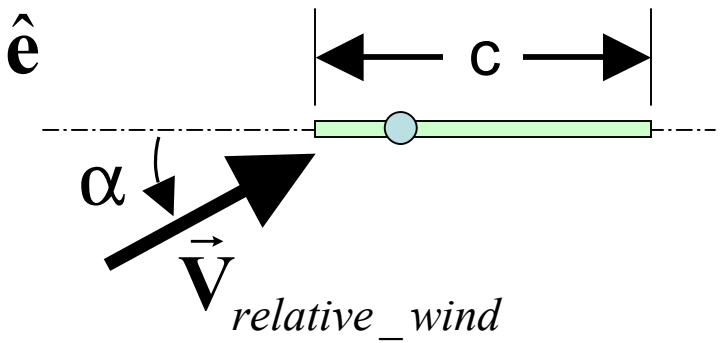
Lift and Drag on Lifting Surfaces

- **Primitive Shape: The Rectangular Flat Plate**



Λ= sweep angle

α = angle of attack

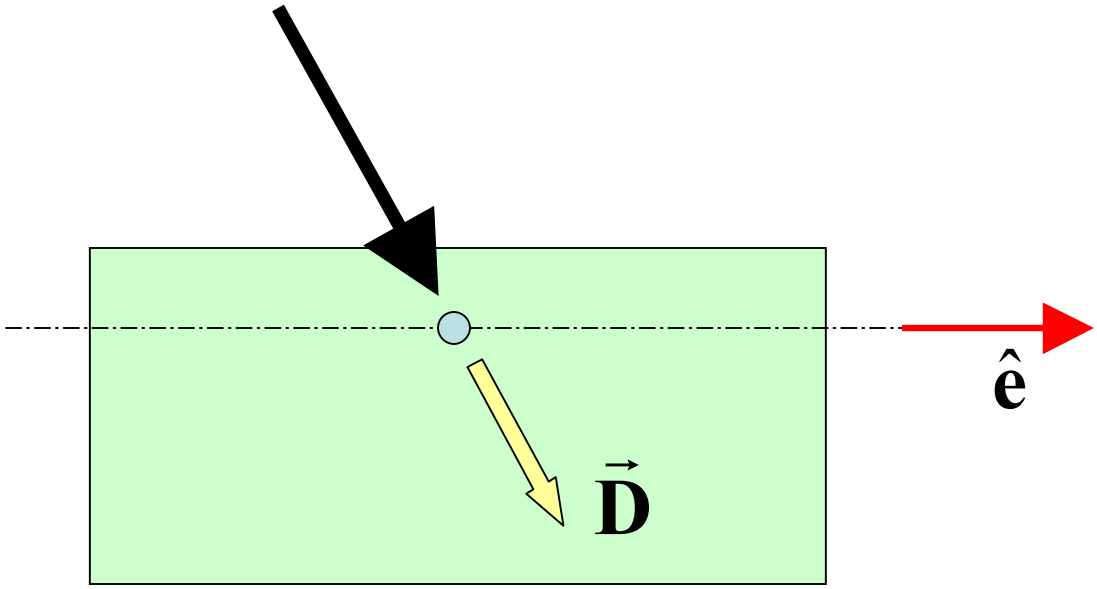


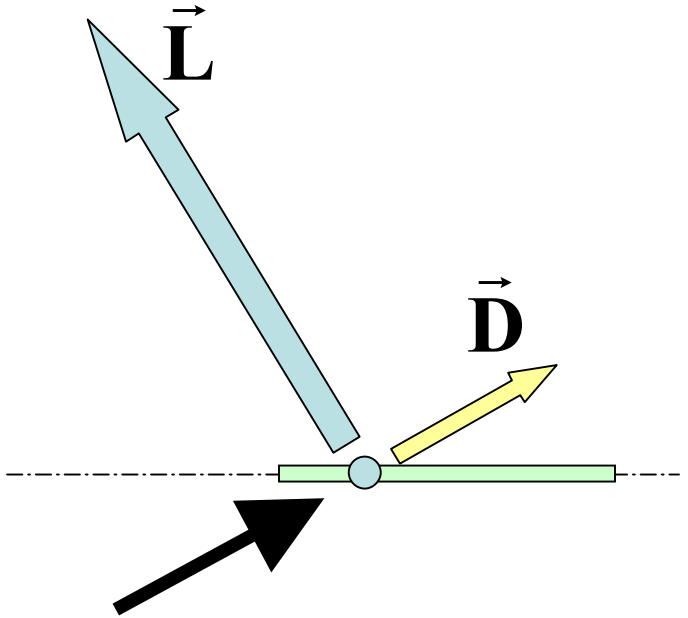
This is a Top View!

This is a Side View!

Copyright (c) 2005 Graham Scott Rhodes

Lift and Drag on Lifting Surfaces





$$\vec{\mathbf{L}} = \frac{\rho V^2 S_{ref} C_L}{2} \frac{\vec{\mathbf{V}}_{relative\_wind}}{V} \times \hat{e}$$

Cross Product

Copyright (c) 2005 Graham Scott Rhodes

Lift and Drag on Lifting Surfaces

- **More Equations (α in radians)**

$$C_L \approx \frac{\pi A \alpha}{1+\sqrt{1+\left(\frac{A}{2\cos\Lambda}\right)^2}}$$

$$C_D = C_{Do} + \frac{1}{\pi A e} C_L^2$$

- C_{Do} **= parasite drag coefficient**
  - Similar to bluff body drag
  - Use 0.045
- A **= aspect ratio =** b/c
- e = *Oswald Span Efficiency Factor*
  - Use 0.8

Copyright (c) 2005 Graham Scott Rhodes

Computing Forces for Physics Engines

- **Global Coordinate System**
  - Many physics engines prefer forces be computed in world coordinates
- **What is the Appropriate Object Velocity?**
  - You must use the kinematic velocity of the point on the rigid body where the force is to be applied:

$$\vec{\mathbf{V}}_{center-of-force} = \vec{\mathbf{V}}_{c.g.} + \vec{\boldsymbol{\omega}} \times \vec{\mathbf{r}}_{center-of-force}$$

$$\vec{\mathbf{V}}_{relative\_wind} = \vec{\mathbf{V}}_{wind} - \vec{\mathbf{V}}_{center-of-force}$$

  - $\mathbf{r}_{\text{center-of-force}}$ is the vector from the c.g. to the center-of-force, measured in world coordinates
  - $\boldsymbol{\omega}$ is the angular velocity in world coordinates

Copyright (c) 2005 Graham Scott Rhodes

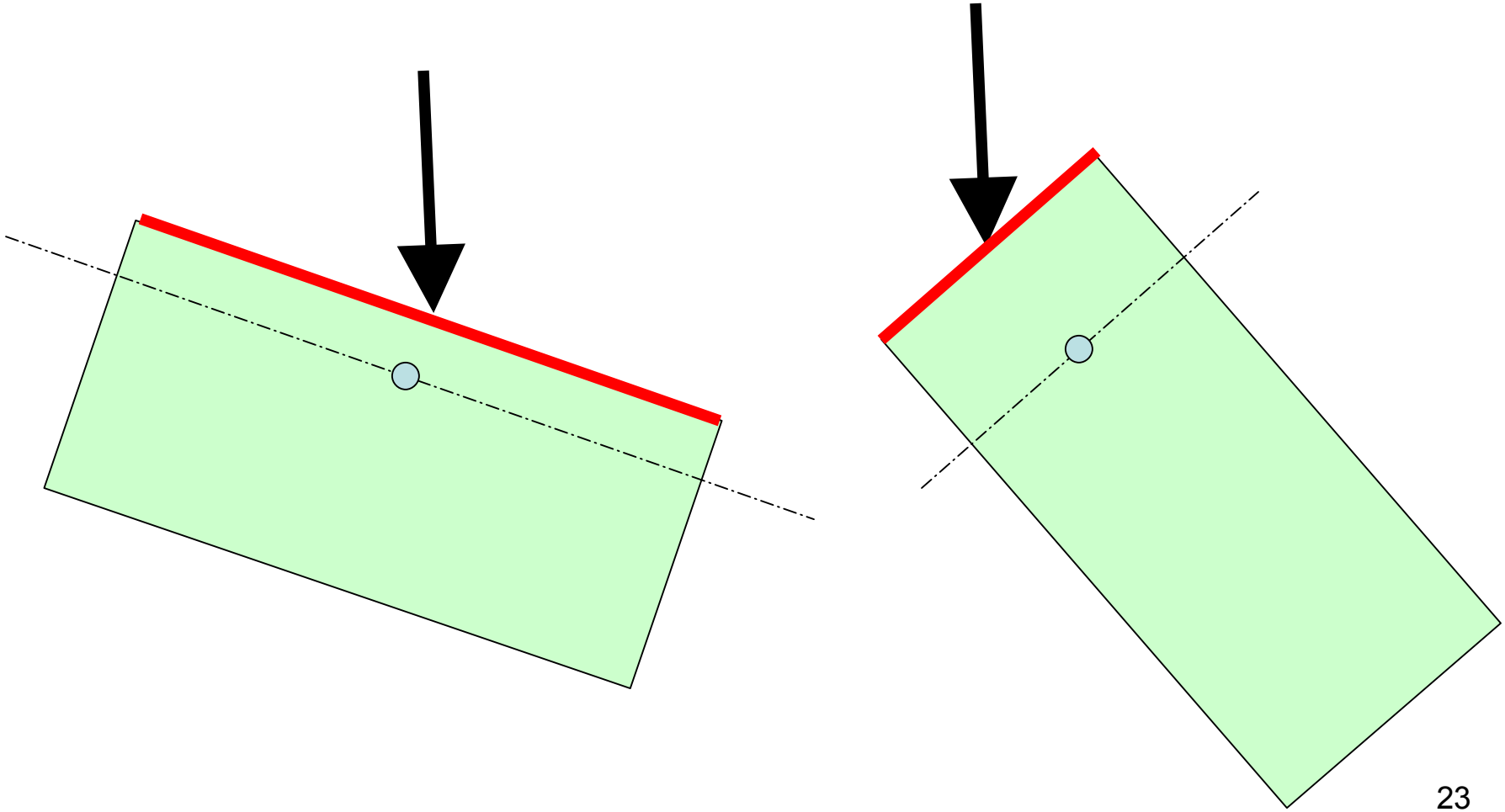
Computing Forces for Physics Engines

- **The "circulation axis" direction, $\hat{\mathbf{e}}$, should be calculated in world coordinates**
  - For spinning bluff bodies, this may simply be a column or row (depending on matrix orientation) of the 3x3 object local-to-world rotation matrix
  - See next page for flat plates
- **Once $\hat{\mathbf{e}}$ and $\vec{\mathbf{V}}_{relative\_wind}$ are known in world coordinates, the prior equations naturally produce lift and drag in world coordinates**
  - Ready to plug into the physics engine!
  - Forces must be applied at the center-of-force

Copyright (c) 2005 Graham Scott Rhodes

Computing Forces for Physics Engines

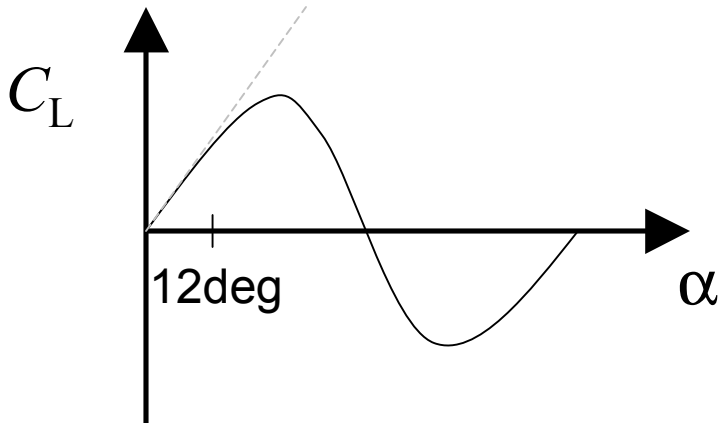
- **An issue with flat plates….which edge is the leading edge?**



Copyright (c) 2005 Graham Scott Rhodes

Computing Forces for Physics Engines

- **Realistic Limits on Lifting-Body Lift**
  - The C_L equation on page 20 is a good estimate up to about α = 10 degrees (0.17~ radians)
  - Beyond that, realistically C_L varies nonlinearly with α.
  - For games that will have objects at arbitrary angles-of-attack, use a curve that matches the equation for low-α, but gradually turns around and passes through zero at $\alpha = \pi/2$.



Copyright (c) 2005 Graham Scott Rhodes

Work In Progress
Integration with ODE + NovodeX

- **Goal**
  - Framework for swappable physics engines
  - For experimentation with aerodynamic forces
- **Toolset**
  - Wild Magic rendering engine v3.3
  - NovodeX physics v2.2
  - ODE physics v0.5
  - 3ds max 6/7

Copyright (c) 2005 Graham Scott Rhodes

Work In Progress
Integration with ODE + NovodeX

- **Current Art Path**
  - 3ds max for visual and physics model
  - Custom MAXScript and file format
  - Open Dynamics Framework provides an alternative, but requires more work to integrate with rendering engine
    - www.physicstools.org (currently down)

Copyright (c) 2005 Graham Scott Rhodes

Work In Progress
Integration with ODE + NovodeX

- **Sample Script Output**

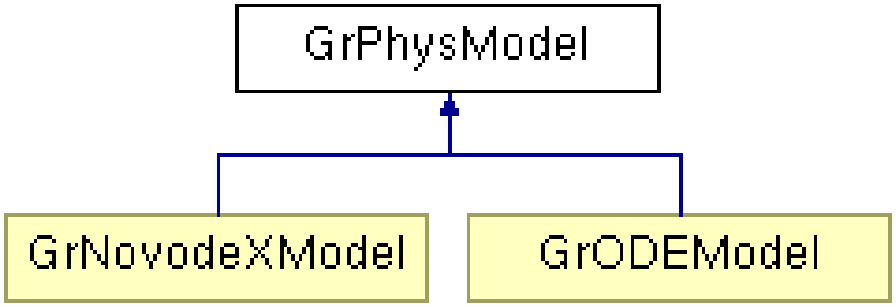
```
Box
Length: 3.15781
Width: 4.82672
Height: 0.25
Name: Box01
Density: 10.0
Position: (-12.1113, 10.3556, 6.5059)
Rotation: (1.0, 0.0, 0.0, -0.000872665)
CGOffset: (0, 0, 0.125)
AeroPrimitive: FlatPlate

BallAndSocketJoint
RigidBody1: Link1
RigidBody2: Link2
WorldLocation: (0.0, -150.0, 42.0)
```

- **3ds max user props**
  - GSRRigidBody
  - GSRDensity
  - GSRAeroPrimitive
  - GSRJoint
    - Ball and socket joint

Copyright (c) 2005 Graham Scott Rhodes

Integration Framework

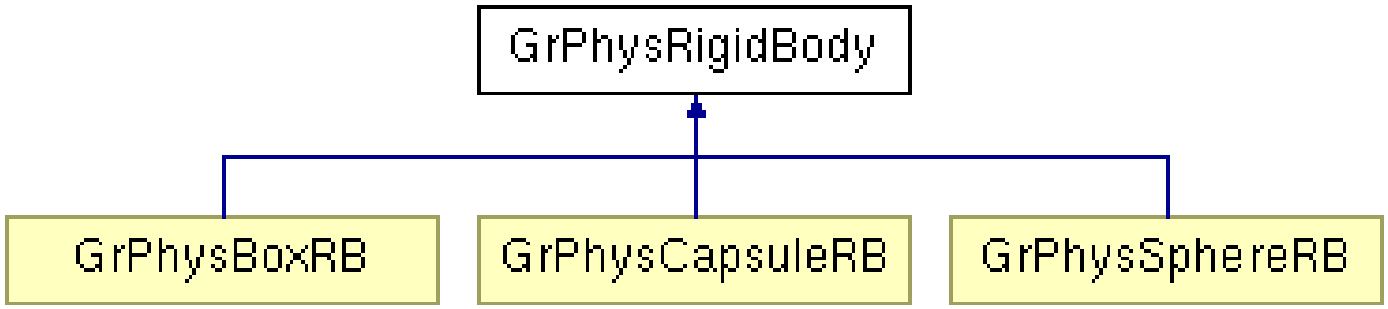
- **In Code**
  - GrPhysModel
    - Loads export file
  - Derived classes:
    - Instantiate API-specific physics model
    - Create associations between visual and physical models
    - Perform physics simulation step
    - Map results back to visual model
  - This is similar to ODF



Copyright (c) 2005 Graham Scott Rhodes

Integration Framework

- **In Code**
  - GrPhysRigidBody
    - Represents a single rigid body from export file
    - Derived classes are shape specific
  - GrPhysJoint
    - Represents a single rigid body joint from export file
    - Currently only ball and socket joint supported



Copyright (c) 2005 Graham Scott Rhodes

Resources

- **How to Contact Me**
  - WillieTheKing@gsrhodes.com
  - Math & Physics forum @ www.gamedev.net
- **Slides – posted tomorrow, 26 June**
  - See my website www.gsrhodes.com
- **Demos – hopefully late July**
  - Post vacation
  - Pre-SIGGRAPH

Copyright (c) 2005 Graham Scott Rhodes

Resources

- **References**
  - "Back-of-the-Envelope Aerodynamics for Game Physics," *Game Programming Gems 5, February 2005*
  - "Real-time Game Physics," *Introduction to Game Development*, June 2005

- **URL's**
  - *NovodeX:* www.ageia.com
  - ODE: www.ode.org
  - Wild Magic: www.geometrictools.com
  - Bluff Body Drag Calculator

    www.fluidmech.net/jscalc/cdcal26.htm

Copyright (c) 2005 Graham Scott Rhodes