# PROJECT 3

SPACE FLIGHT SIMULATION

# FIRST, SOME APOLOGIES

**Only runs on Windows (DirectX)**

**Bugs**

- 201-566-6610
- jas512@lehigh.edu

# KNOWN ISSUES

**Don’t deploy landing gear (g) on Observer plane.**

- Causes a snapping
- You won’t need it anyway

**bNormalizationResult Failure**

- Assertion thrown by the physics engine when things rotate really fast.  Not actually a problem, only happens because we’re running in debug mode.
- Don’t hit anything at a high speed and it won’t happen.

# SECOND, SOME (STRONG) RECOMMENDATIONS

**Use Visual Studio 2008**

- Project files already set up
- I know it works
- Download full version for free from Dreamspark

# YOU WILL GET

**A RAR Containing**

- VS2008 Project File
- Library header (.hpp) files
- Compiled libs (to link against)
- Dependencies
- Client executable and data folder
- Heavily commented ATC AI to use as an example
- Skeleton code for DockingAI

# YOU WILL MAKE

**An AI DLL**

**DockingAI**

- Thrust control & Maneuvering
- Radio communication

# INSTALLATION

**Extract the RAR file**

- Put it wherever you want
- Contains
  - ODE (physics engine)
  - CEGUI (user interface)
  - Project folder

**Installing DirectX**

- Download and install DirectX 9.0 SDK February 2010
- Install to “Program Files”, even on x64 systems
- http://www.microsoft.com/download/en/details.aspx?id=10084

# GETTING STARTED

**Source Code to Look Through**

- TrafficController.hpp and TrafficController.cpp
- neb_lib_scene\include\Capital.hpp
- neb_lib_scene\include\IWorldObject.hpp

**Math to brush up on/learn**

- Matrices and Vectors (particularly transformations)
- Quaternions
  - http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/index.htm

**File You'll be Modifying:**

- DockingAI.cpp and DockingAI.hpp

# COMMUNICATION

**Your ship has a radio, accessed via IWorldObject::GetRadio()**

**“Coms” allow you to listen and talk on multiple channels (make sure to turn them on)**

**You must tune your radio to the proper frequency**

- Hailing Frequency: 2231.5MHz (tune com 0 to this)

**ATC will contact you when you enter their airspace**

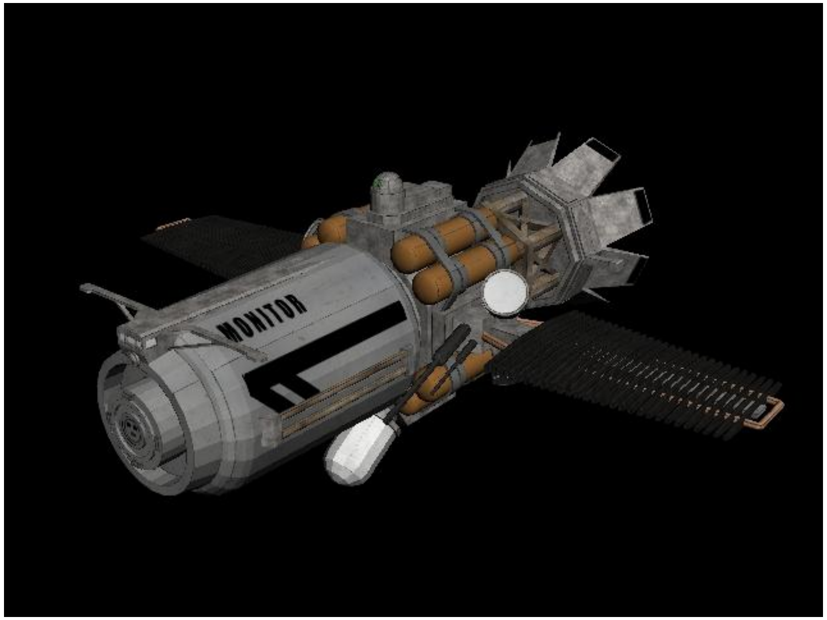
- You will need to check for new radio messages every frame
- Tune com 1 to the frequency they request contact on

**Signal that you want to dock using the proper message**

- Messages are in neb_lib_scene/include/RadioMessage.hpp
- Fill out the proper structure and transmit
  - Radio::Transmit(long com, RADIO_MESSAGE *Msg)

**ATC will walk you through the docking procedure.  Follow their instructions**

# YOUR SHIP



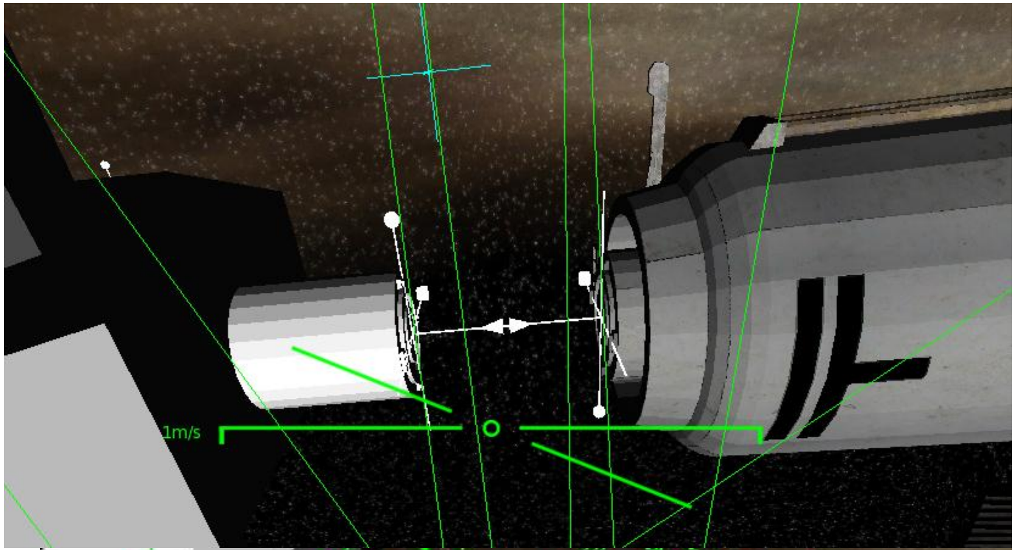
**Multiple Engine Types**

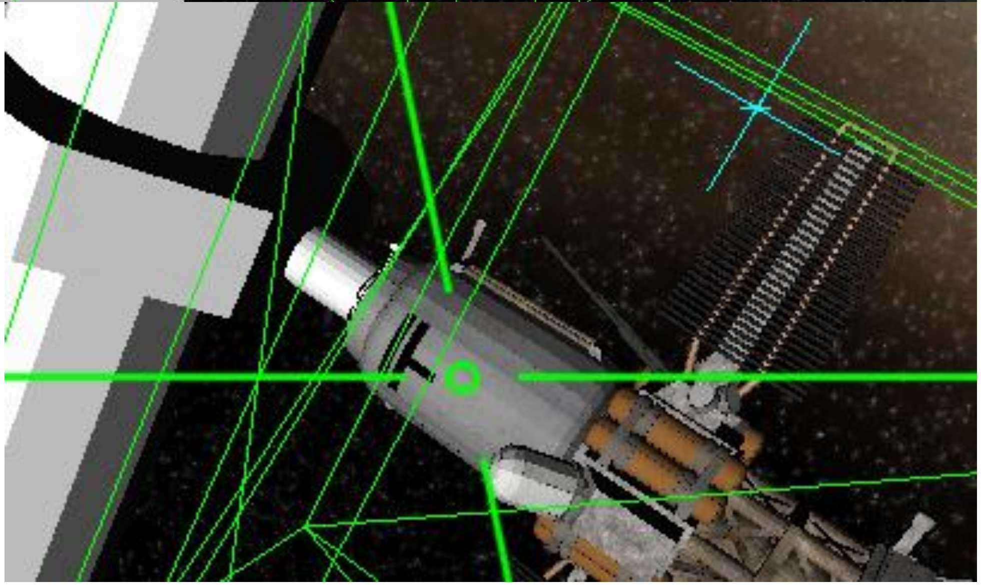
- Maneuvering Thrusters
  - Low thrust, low fuel consumption
- Main Engine
  - High thrust, high fuel consumption
  - Only works in one direction
- IMPORTANT: time spent accelerating must equal time spent decelerating (on same engine type)

**Docking point on the bow**

- In order for the Dock function to work:
  - Your docking point must be within 3m of any docking point on the ship/station you want to dock with
  - Your docking point's +look, +right, and +up vectors must be within 3 degrees of the station's –look, -right, and +up vectors

# DOCKING ALIGNMENT REFERENCES





# LIDAR

**Your ship's on-board sensor cluster**

- Operates in sweeps, performed at light-speed
- Accessed via Capital::GetLidar()

**Sweeps**

- Lasers move at light speed.  Distance will not be a problem at the range you're operating.
- Sweep will not finish until all signals have returned from its 360* scan
- Sweep time is 5 seconds + ((sweep_radius * 2) / C)
- Poll for completion with Lidar::GetSweepDone()
- Get results with Lidar::GetSweepResults()

# DOCKING STEPS

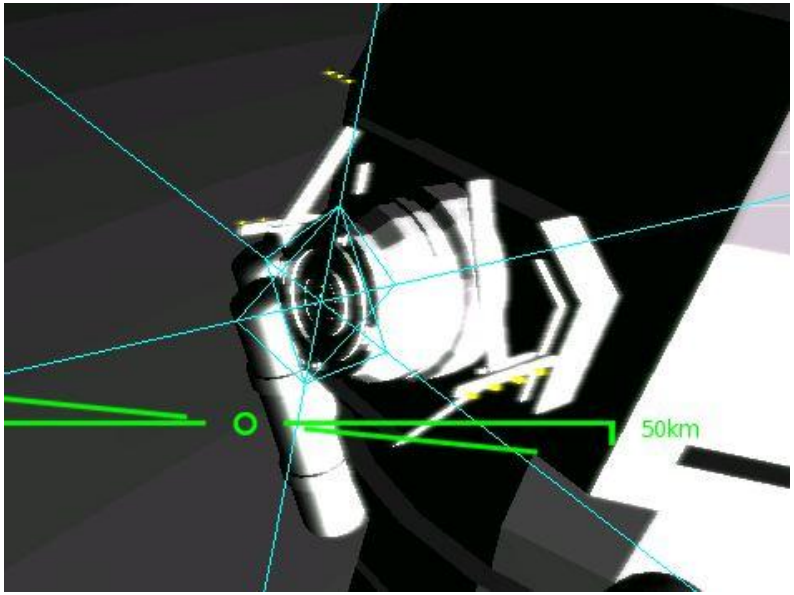
Wait for ATC to hail you with AppCon frequency

Tune to that frequency

Request docking permission from ATC

Fly to waypoints ATC gives you

- The last waypoint (GetNextWaypoint() will return NULL) will be the docking point’s location



# RELATIVE POSITIONS AND VELOCITIES

**You can get an object’s velocity with IWorldObject::GetLinearVelocity()**

- This is generally useless to you.
- station_linear_velocity - your_linear_velocity more useful

**Positions from IWorldObject::GetPos() are huge numbers. Relative positions are more useful and intuitive.**

- Do the same thing as above

# WAYPOINTS

**Waypoints are given in a linked list of Waypoint classes**

- Calling Waypoint::GetPos() will give you the position of the waypoint in the solar system (even through they are stored as relative positions).
- Waypoints have a rotation.  You'll need to line your docking point's axes up with the waypoint that represents the station's docking point.
- Getting axes:
  - D3DXVECTOR3 look;
  - D3DXVec3TransformNormal(&look, &UNIT_Z, D3DXMatrixRotationQuaternion(&D3DXMATRIX(), &Waypoint->GetRot()));
  - Look = Z axis, Right = X axis, Up = Y axis

# ENGINES

**Burning the Main Engine**

- Capital::FireCruiseEngine(float burnTime)

**Fine Maneuvering**

- Capital::FireManeuverThrusters(unsigned long direction, float delta);
- Capital::FireAttitudeThrusters(unsigned long direction, float delta);

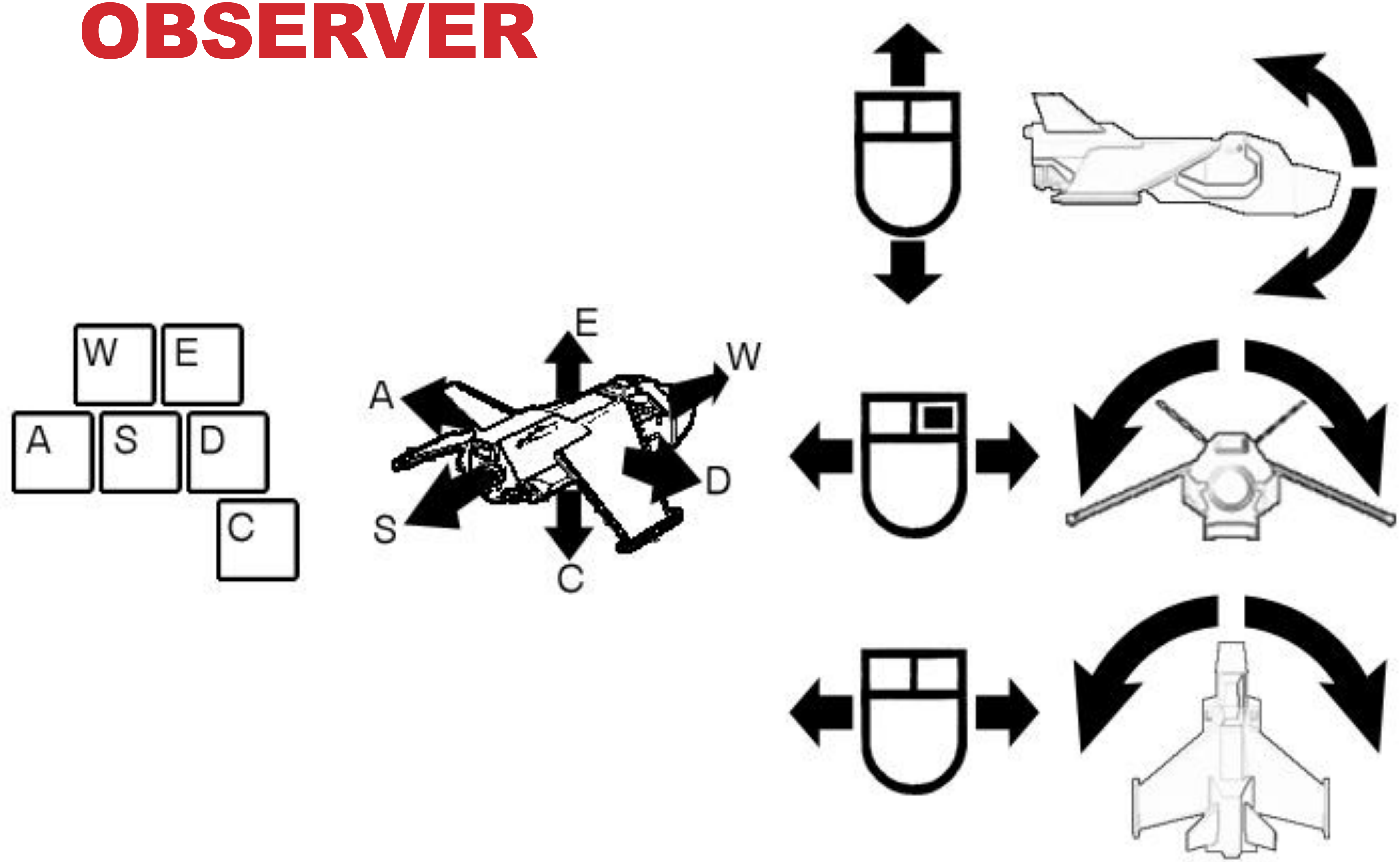
**Throttle Control**

- Capital::SetManeuveringThrottle(float throttle)
- Capital::SetCruisingThrottle(float throttle)

**Direction bitfield**

- Maneuvering: D_FORWARD, D_BACKWARD, D_LEFT, D_RIGHT, D_UP, D_DOWN
- Attitude: D_FORWARD, D_BACKWARD, D_LEFT, D_RIGHT, D_ROLLLEFT, D_ROLLRIGHT

# FLYING THE OBSERVER



# KEYS

W,A,S,D,E,C – Thrusters

LCtrl – Toggle mouse capture

T – Cycle targets

F5 – Radio Controls

F10 – Starmap

When in starmap: right-click + drag moves. Right-click + left-click + drag moves in and out

+/- – increase or decrease throttle

To deselect target: Uncapture mouse (using ctrl) and click anywhere in space.

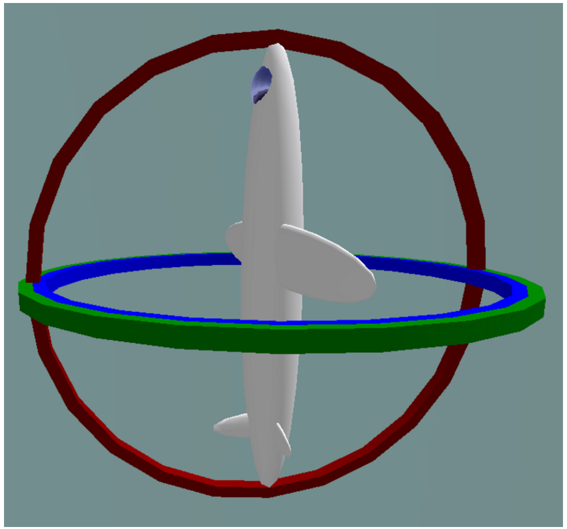
# GIMBAL LOCKING

IWorldObject::GetRot() returns an Euler rotation.

Don’t use it!

Use IWorldObject::GetQuat() instead.  Returns a quaternion.

You need to do this because using Euler rotations will result in Gimbal Locking



# THINGS TO NEVER DO

**Calling the following functions will break literally everything:**

- IWorldObject::SetLinearVelocity()
- IWorldObject::SetAngularVelocity()
- IWorldObject::SetPos()
- IWorldObject::SetRot()
- IWorldObject::Update()
- IWorldObject::Draw()
- Pretty much everything else