MOOS Conventions

Paul Newman

February 18, 2009

Abstract

This document states the conventions used by MOOS. This covers coordinate frames as well as processes and variable naming policies.

1 Coordinate Conventions

MOOS processes communicate using a defined coordinate system as illustrated in Figure 1. The salient points to note about this system are as follows.

- The global (earth) frame is a conventional East-North-Up frame.
- The vehicle body frame has been designed to align with the global frame when the vehicle has zero yaw.
- At zero heading the vehicle points north.
- Zero heading is equivalent to zero yaw.
- Yaw = -heading*π/180 (heading is in degrees).
- Depth is in the opposite sense to Z.

The fact that a coordinate system is defined that must be used between processes *does not mean* this system is inflicted on the internals of an application. Within the boundaries of a process, the developer is free to use whatever system he or she feels comfortable with. All that is needed is a little patch work to transform input and output data between coordinate frames. However, life can be made easier by using one coordinate frame throughout.

2 Geodesy

MOOS defines a geodetic called `MOOSGrid` . The geodesy tools within MOOS assume operation in a local area – say a square with sides less than 20 km long. With this assumption it is reasonable to superimpose a local cartesian

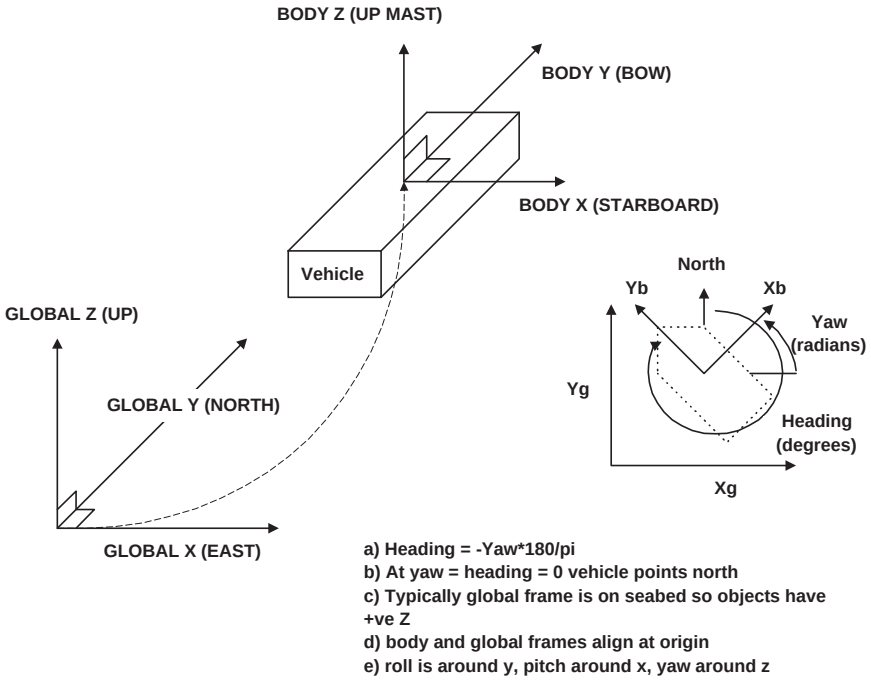

Figure 1: The MOOS coordinate system. Note that $\dot{yaw} = -\dot{heading}$ and that the body and earth frames align at the origin with the vehicle pointing north at zero yaw. Also the body 'y' axis is towards the front of the vehicle. (You may be used to having the body x axis point towards the nose of the vehicle). Note also that Z points up.

space over the work area and form a mapping from Lat/Long coordinates to local `MOOSGrid` . The class `CMOOSGeodesy` in `MOOSGenLib` performs all the mapping required. It is primed with the origin of the `MOOSGrid` (in decimal lat/long). It can then convert lat/long measurements to cartesian `MOOSGrid` coordinates using a geodetic radius that is a function of the origin location.

3 Units

MOOS was designed and built by a European and so uses easy SI units throughout. All distances are in meters and times in seconds. Angles are always in radians and wrapped to lie with $\pm\pi$. The only exception to this is "heading" variables, which are in degrees. However, heading variables are never used internally – they are for human display alone. Instead their "Yaw" counterparts are used, which are always in radians.

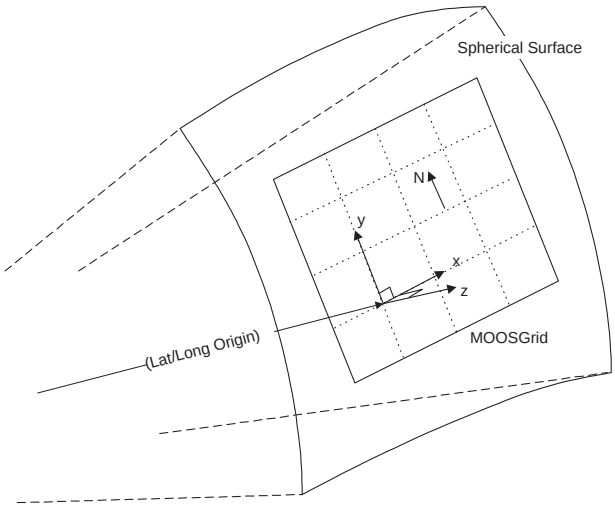

Figure 2: Defining a local cartesian `MOOSGrid` around a lat/long origin

.

4 Naming Conventions

4.1 Process Naming

MOOS developers are encouraged to adopt the naming convention for applications given in Table 4.1. The basic idea is that the prefix to the process gives some indication of its function. Processes beginning with "i" typically make use of `CMOOSInstrument` and perform some kind of I/O via serial port (often to a hardware sensor) or non-MOOS communications. Processes beginning with "p" are pure and perform no I/O other than via the MOOSDB. Finally "u" processes are utilities – performing tasks not critical to vehicle operation but which are perhaps useful at other times – for example, `uGeodesy` converts from "Lat/Long" to local-grid (`MOOSGrid`) coordinates.

Table 1: The naming convention for MOOS applications.

Name	**Description**	**Example**
i[*Name*]	Interface applications. Interacts (has I/O) with an external device, for example via a keyboard or serial port	`iGPS, iCompass, iRemote`
p[*Name*]	Pure applications. Only interacts with other MOOS applications	`pHelm, pLogger, pNav`
u[*Name*]	Utility applications. Not used at run time but useful at other times	`uPlayBack, uGeodesy`

4.2 Variable Naming

Data can take any string name. A developer could use any combination of characters to name the data published by his or her application. In practice

though, sticking to a convention makes some things easier. A good example of this is the `pNav` process that expects some kinds of sensor data to be named in a standard format. Sensor data can be classified into at least the following categories.

Quantity	**Description**	**Example**
`X,Y,Z`	Sensor measures vehicle position	GPS
`DEPTH`	Sensor gives depth	Pressure sensor
`YAW`	Sensor measures rotation around Z axis	Gyro
`BODY_VEL`	Sensor measures velocity in body frame	DVL, Odometry

Table 2: Sensor categories.

If a sensor, managed by a process called `iSensor`, measures one of these quantities then the name under which the data should be published has the format `SENSOR_`*CATEGORY*. This is best highlighted with a few examples.

- iGPS measures X and Y position. It publishes `GPS_X` and `GPS_Y`.
- iDepth measures depth. It publishes `DEPTH_DEPTH`.
- iLBL measures range and depth. It publishes `LBL_DEPTH` and `LBL_TOF` (time of flight).

This simple convention makes visual comprehension of the system (when using a tool like `MOOSScope`) simple. Additionally it avoids naming conflicts as the `MOOSDB` requires unique process names and therefore basing data names on process names will also result in unique data names.

4.3 Actuator Naming

At the time of writing MOOS applications are aware of three actuator types: `THRUST`, `ELEVATOR` and `RUDDER`. Clearly this illustrates the early history of MOOS as software for autonomous underwater vehicles (AUVs). On land the `ELEVATOR` actuator has no meaning. The `THRUST` and `RUDDER` axes however map well to throttle and steer angle.

Note that, perhaps counter-intuitively, but consistent with the MOOS coordinate frame, a positive elevator angle will cause a vehicle to make its pitch more negative. Similarly, positive rudder will make yaw decrease (but heading increase).