Хлебникова Е. С.

Отчет о проделанной работе

Глобально моя задача состояла в написании автопилота для KSP, который получал бы некоторые параметры и, исходя из них, управлял бы полетом ракеты. Параметры должны были быть получены от программиста, это результат, выдаваемый программой, рассчитывающей нашу миссию. Это было необходимо для наиболее тщательной проверки точности наших вычислений и сопоставления нашей физической модели полета с физической моделью, реализованной в KSP.

Для написания автопилота было решено использовать мод KOS.

Мной было написано 2 варианта автопилота. Первый реализует пуск с Земли, вертикальный взлет до некоторой высоты, наклон, далее набор высоты под углом к нормали к поверхности, построение орбиты вокруг Земли, выход на орбиту Земли, маневр (выход на Гомановскую траекторию), построение орбиты вокруг Луны, выход на орбиту Луны. Второй не реализует начальные стадии, подразумевает, что на момент запуска кода ракета уже находится на орбите Земли. Оба кода пригодились для оценки точности наших вычислений.

KOS-скрипт автопилота:

// Програмные данные

set ALT1 to 15000.0000. // Высота подьема первой фазы

set ANGL2 to 90 - 55.0000.

set T2 to 28.4200. // длительность второй фазы

set transfer\_burn to 24.5926. //время работы двигателя для переходной орбиты

set start\_angle to -48.4601. // угол между местом старта и муной

set start\_transfer to 1155.9600. //Время когда кончается фаза LEO и начинается переходная орбита

set final\_t to 24120.5500. // полупериод переходной орбиты, время когда мы будем на пике гепотетической переходной орбиты

set final\_burn\_t to 4.8857. //Длительность последнего маневра

set eps to 0.5. // градус погрешности

function DoSafeStage{

wait until stage:ready.

stage.

}

function Do1stPhase {

lock steering to up.

lock throttle to 1.

DoSafeStage.

until altitude > (ALT1 - (ALT1 / 10)){

}

}

function Do2ndPhase {

lock throttle to 1.

set mysteer to heading(90, ANGL2).

lock steering to mysteer.

wait T2.

lock throttle to 0.

}

function set\_sasMode {

UNLOCK STEERING.

SET NAVMODE TO "Surface".

WAIT 0.5. // so sas has time to recognize navball change

SAS on.

SET NAVMODE TO "Orbit".

WAIT 1. // so sas has time to recognize navball change

SET SASMODE TO "Prograde".

}

function Do3rdPhase {

set\_sasMode.

wait until altitude > (apoapsis - 900).

sas off.

}

function Do4thPhase {

set apo to apoapsis.

set t0 to time:seconds.

print t0.

lock steering to heading(90, 0.4).

lock throttle to 1.

wait until (periapsis > apo - 1000).

lock throttle to 0.

unlock steering.

unlock throttle.

wait 10.

DoSafeStage.

wait 10.

DoSafeStage.

wait 2.

DoSafeStage.

}

function Do5thPhase {

set\_sasMode.

wait 5.

lock throttle to 1.

wait transfer\_burn.

lock throttle to 0.

unlock throttle.

sas off.

}

function Do6thPhase {

set\_sasMode.

wait 1.

set sasMode to "Retrograde".

wait 5.

lock throttle to 1.

wait final\_burn\_t.

lock throttle to 0.

sas off.

unlock steering.

unlock throttle.

}

function main {

set T to time:seconds.

Do1stPhase.

Do2ndPhase.

Do3rdPhase.

Do4thPhase.

WARPTO(T + start\_transfer - transfer\_burn).

until (time:seconds - T) > (start\_transfer - (transfer\_burn / 2)){

print((time:seconds - T) - (start\_transfer - (transfer\_burn / 2))).

}

print((time:seconds - T) - (start\_transfer- (transfer\_burn / 2))).

Do5thPhase.

wait 10.

WARPTO(T + final\_t - 10).

until (time:seconds - T) > (final\_t){

print((time:seconds - T) - (final\_t)).

}

Do6thPhase.

}

print ship:velocity:orbit:sqrmagnitude^0.5.

until (((mun:longitude - ship:longitude) > start\_angle - eps) and ((mun:longitude - ship:longitude) < start\_angle + eps)) {

print mun:longitude - ship:longitude - start\_angle.

wait 1.

}

main.