|  |
| --- |
| Orbital Design |

# Class Diagrams

## Overview

A black background with white rectangular boxes

AI-generated content may be incorrect.

## Sim

A black and white screen with white text

AI-generated content may be incorrect.

## Satellite

A black screen with white text

AI-generated content may be incorrect.

## Direction, Position, Velocity, and Acceleration

A black screen with white text

AI-generated content may be incorrect.

# Structure Chart

A group of white rectangular shapes with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Pseudocode

## Sim::Sim()

Sim.Sim(ptUpperRight)

Framerate 🡨 30

hoursPerDay 🡨 24

minutesPerHour 🡨 60

secondsPerMinute 🡨 60

secondsPerDay 🡨 hoursPerDay x minutesPerHour x secondsPerMinute

timeDilation 🡨 hoursPerDay x minutesPerHour

FOR i 🡨 0 … 200

Stars[i].reset(ptUpperRight)

radiansInADay 🡨 -2π

radiansPerFrame 🡨 (radiansInADay / framerate) x (timeDilation / secondsPerDay)

satellites.push\_back(Earth(radiansPerFrame))

FOR i 🡨 0 … 6

Satellites.push\_back(GPS(i))

Satellites.push\_back(Starlink)

Satellites.push\_back(Hubble)

Satellites.push\_back(Sputnik)

Satellites.push\_back(Dragon)

## Sim::move()

Sim.move()

Time 🡨 timedilation / framerate

FOR satellite IN satellites

Satellite.move(time)

FOR it1 🡨 satellites.begin() … satellites.end()

FOR it2 🡨 it1 + 1 … satellites.end()

IF NOT it1.dead and NOT it2.dead

Distance 🡨 computeDistance(it1.position, it2.position)

IF distance < it1.radius + it2.radius

It1.kill()

It2.kill()

FOR it 🡨 satellites.begin() … satellites.end()

IF it.dead()

It.destroy(satellites)

It 🡨 satellites.erase(it)

## Satellite::move()

Satellite.move(time)

aGravity 🡨 getGravity(pos)

updateVelocity(velocity, aGravity, time)

updatePosition(pos, velocity, gravity, time)

direction.rotate(angularVelocity)

## GPS::destroy()

GPS.destroy(satellites)

Satellites.push\_back(GPSCenter(this, Direction(90)))

Satellites.push\_back(GPSLeft(this, Direction(0)))

Satellites.push\_back(GPSRight(this, Direction(180)))

Satellites.push\_back(Fragment(this, Direction(330)))

Satellites.push\_back(Fragment(this, Direction(250)))

## GPSRight::destroy()

GPSRight.destroy(satellites)

Satellites.push\_back(Fragment(this, Direction(115)))

Satellites.push\_back(Fragment(this, Direction(325)))

## Ship::input()

Ship.input(ui, satellites)

Direction.rotate((ui.isRight() ? 0.1) + (ui.isLeft() ? -0.1))

IF ui.isDown()

Acceleration 🡨 30, direction

Velocity += acceleration

IF isSpace()

vBullet 🡨 9000, direction

satellites.push\_back(Projectile(position, vBullet))