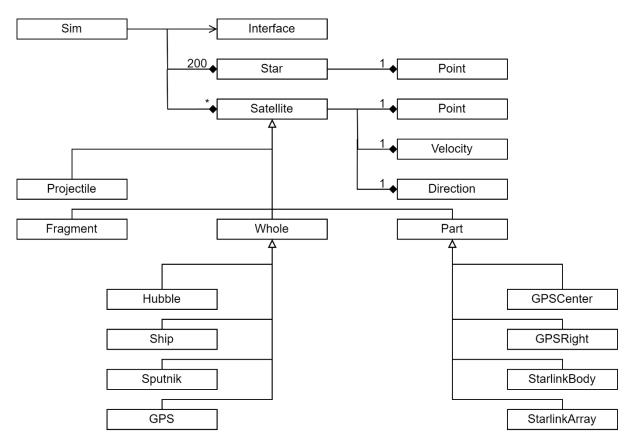
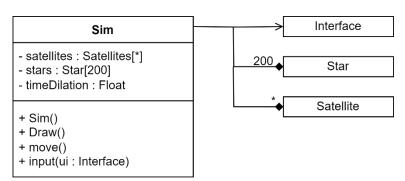
# Orbital Design

## Class Diagrams

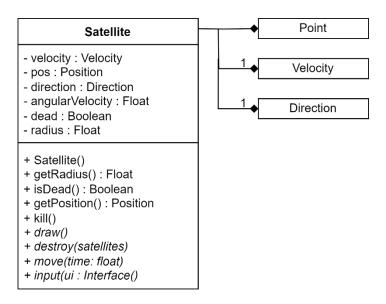
#### Overview



#### Sim



#### Satellite



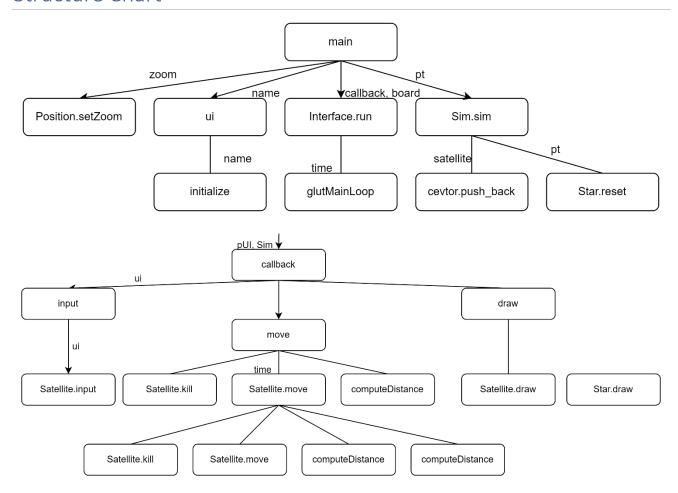
### Direction, Position, Velocity, and Acceleration

Position

#### - y - <u>m</u>e Direction + P - radians : Float + ge + ge + Direction() + setRadians(radians) + ge + setDegrees(degrees) + ge + se + setDxDy(dx, dy) + setUp() + setDown() + 56 + 56 + setLeft() + se + ac + setRight() + rotate(amount) + a + getRadians(): Float + getDegrees(): Float + getDx(): Float + getDy(): Float + a + a + 56

- x : Float - y : Float - <u>metersFromPixels :</u> Float	Velocity
+ Position() + assign(position) + getmetersX(): Float + getMetersY(): Float + getPixelsX(): Float + getPixelsY(): Float + setMetersX(x) + setMetersX(x) + setPixelsX(x) + setPixelsY(y) + addMetersX(dx) + addMetersX(dx) + addPixelsX(dx) + addPixelsY(dy) + addPixelsY(dy) + setZoom(metersFromPixels) + getZoom(): Float	- dx : Float - dy : Float  + Velocity() + assign(velocity) + getDx() : Float + getDy() : Float + getSpeed() : Float + getDirection() : Direction + setDx(dx) + setDy(dy) + setSpeed(speed) + setDirection(direction) + addDx(dx) + addDy(dy) + add(velocity)
0 0	

Acceleration
- ddx : Float - ddy : Float
+ Acceleration() + assign(velocity) + getVelocity(time) : Velocity + getDDY() : Float + getDDX() : Float



### 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 \leftarrow 0 ... 200
      Stars[i].reset(ptUpperRight)
   radiansInADay \leftarrow -2\pi
   radiansPerFrame ← (radiansInADay / framerate) x (timeDilation / secondsPerDay)
   satellites.push_back(Earth(radiansPerFrame))
   FOR i \leftarrow 0 ... 6
      Satellites.push_back(GPS(i))
   Satellites.push_back(Starlink)
   Satellites.push_back(Hubble)
   Satellites.push_back(Sputnik)
```

#### Sim::move()

#### 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))
```