3I/ATLAS Trajectory System - Quick Start Guide

Installation

```
# Install dependencies
pip install requests scipy numpy

# Make scripts executable
chmod +x generate_trajectory.py update_trajectory.py
```

Generate Trajectory Data

```
# Generate initial trajectory data
python3 generate_trajectory.py

# Output: 3iatlas_trajectory_data.json
```

Using the Data

Load in JavaScript

```
// Load trajectory data
fetch('3iatlas_trajectory_data.json')
  .then(response => response.json())
  .then(data => {
    // Access ATLAS trajectory
    const atlasTrajectory = data.atlas;

    // Each point has:
    // - jd: Julian date
    // - date: ISO date string
    // - position: [x, y, z] in AU (Three.js coordinates)
    // - velocity: [vx, vy, vz] in AU/day

    // Access milestones
    const milestones = data.milestones;
    // Each milestone has: name, date, description, position

    // Access planet trajectories
    const earthTrajectory = data.earth;
    const marsTrajectory = data.mars;
    const jupiterTrajectory = data.jupiter;
  });
```

Example: Render in Three.js

```
import * as THREE from 'three';

// Create trajectory line for ATLAS
function createTrajectoryLine(trajectoryData) {
  const points = trajectoryData.map(point => {
    return new THREE.Vector3(
      point.position[0],
      point.position[1],
      point.position[2]
    );
  });

  const geometry = new THREE.BufferGeometry().setFromPoints(points);
  const material = new THREE.LineBasicMaterial({ color: 0x00ff00 });
  const line = new THREE.Line(geometry, material);

  return line;
}

// Load and render
fetch('3iatlas_trajectory_data.json')
  .then(response => response.json())
  .then(data => {
    const atlasLine = createTrajectoryLine(data.atlas);
    scene.add(atlasLine);

    // Add milestone markers
    data.milestones.forEach(milestone => {
      const marker = new THREE.Mesh(
        new THREE.SphereGeometry(0.05, 16, 16),
        new THREE.MeshBasicMaterial({ color: 0xff0000 })
      );
      marker.position.set(
        milestone.position[0],
        milestone.position[1],
        milestone.position[2]
      );
      scene.add(marker);
    });
  });
```

Update Data (Automated)

```
# Setup cron job for twice-daily updates
python3 update_trajectory.py --setup-cron

# Or run manually
python3 update_trajectory.py
```

Key Data Points

Event	Date	Distance from Sun	Speed
Discovery	July 1, 2025	4.494 AU	61.22 km/s
Mars Flyby	October 3, 2025	1.658 AU	~65 km/s
Perihelion	October 29, 2025	1.360 AU	~68 km/s
Jupiter Approach	March 16, 2026	3.636 AU	~62 km/s

Data Specifications

- **Time Range**: July 1, 2025 - January 31, 2026
- **Time Step**: 6 hours
- **Total Points**: 857 per object
- **Coordinate System**: Three.js (right-handed, +Y up)
- **Units**: AU for position, AU/day for velocity
- **File Size**: ~1 MB

Common Operations

Get Position at Specific Date

```
function getPositionAtDate(trajectory, targetDate) {
  const target = new Date(targetDate);

  // Find closest point
  let closest = trajectory[0];
  let minDiff = Math.abs(new Date(closest.date) - target);

  for (const point of trajectory) {
    const diff = Math.abs(new Date(point.date) - target);
    if (diff < minDiff) {
      minDiff = diff;
      closest = point;
    }
  }

  return closest.position;
}

// Example usage
const positionAtPerihelion = getPositionAtDate(
  data.atlas,
  '2025-10-29'
);
```

Calculate Distance Between Objects

```
function getDistance(pos1, pos2) {
  const dx = pos1[0] - pos2[0];
  const dy = pos1[1] - pos2[1];
  const dz = pos1[2] - pos2[2];
  return Math.sqrt(dx*dx + dy*dy + dz*dz);
}

// Distance from ATLAS to Mars at flyby
const atlasPos = getPositionAtDate(data.atlas, '2025-10-03');
const marsPos = getPositionAtDate(data.mars, '2025-10-03');
const distance = getDistance(atlasPos, marsPos);
console.log(`Distance to Mars: ${distance.toFixed(3)} AU`);
```

Interpolate Between Points

```
function interpolatePosition(point1, point2, fraction) {
  return [
    point1.position[0] + (point2.position[0] - point1.position[0]) * fraction,
    point1.position[1] + (point2.position[1] - point1.position[1]) * fraction,
    point1.position[2] + (point2.position[2] - point1.position[2]) * fraction
  ];
}

// Smooth animation between data points
const pos = interpolatePosition(
  data.atlas[100],
  data.atlas[101],
  0.5  // halfway between
);
```

Troubleshooting

File Not Found

```
# Check if file exists
ls -lh 3iatlas_trajectory_data.json

# Regenerate if missing
python3 generate_trajectory.py
```

Invalid JSON

```
# Validate JSON
python3 -m json.tool 3iatlas_trajectory_data.json > /dev/null
```

Coordinate Issues

```
// Verify coordinate system
console.log('ATLAS at discovery:', data.atlas[0].position);
// Should be: [-4.413, 0.074, -0.846]

console.log('Earth at start:', data.earth[0].position);
// Should be: [~0.16, ~0.00, ~1.00]
```

Performance Tips

1. **Reduce Points for Preview**: Use every 4th point for fast preview
   `javascript`
   `const reducedTrajectory = data.atlas.filter((_, i) => i % 4 === 0);`

2. **Lazy Loading**: Load only visible date range
   `javascript`
   `const visibleData = data.atlas.filter(point => {`
   `const date = new Date(point.date);`
   `return date >= startDate && date <= endDate;`
   `});`

3. **Caching**: Cache processed geometry
   `javascript`
   `const trajectoryCache = new Map();`
   `function getCachedTrajectory(name) {`
   `if (!trajectoryCache.has(name)) {`
   `trajectoryCache.set(name, createTrajectoryLine(data[name]));`
   `}`
   `return trajectoryCache.get(name);`
   `}`

API Reference

See TRAJECTORY_GENERATION_README.md (TRAJECTORY_GENERATION_README.md) for complete API documentation.

Need Help?

1. Check the full documentation: `TRAJECTORY_GENERATION_README.md`
2. View logs: `cat trajectory_generation_log.txt`
3. Test API connection: `curl "https://ssd.jpl.nasa.gov/api/horizons.api?format=json"`

Quick Reference Card

Command	Purpose
`python3 generate_trajectory.py`	Generate initial data
`python3 update_trajectory.py`	Update with fresh data
`cat trajectory_generation_log.txt`	View logs
`du -h 3iatlas_trajectory_data.json`	Check file size
`python3 -m json.tool 3iatlas_trajectory_data.json | head -50`	Preview JSON