Integration Guide

This guide explains how to integrate the 3I/ATLAS Flight Tracker into your existing project.

Table of Contents

Integration into 3iatlas.mysticarcana.com

Step 1: Copy Components

Copy the entire `frontend/src/components` directory into your existing project:

```
cp -r 3iatlas-flight-tracker/frontend/src/components /path/to/3iatlas/src/
```

Step 2: Copy Data Files

Copy trajectory data to your public directory:

```
cp -r 3iatlas-flight-tracker/frontend/public/data /path/to/3iatlas/public/
```

Step 3: Install Dependencies

Add required packages to your existing `package.json`:

```
npm install @react-three/fiber @react-three/drei three react-markdown framer-motion
npm install -D @types/three
```

Step 4: Import and Use

In your desired page/component:

```
import { Atlas3DTrackerEnhanced } from '@/components/Atlas3DTrackerEnhanced';

export default function FlightTrackerPage() {
  return (
    <div className="w-full h-screen">
      <Atlas3DTrackerEnhanced
        autoPlay={true}
        initialSpeed={2}
        initialFollowMode={true}
      />
    </div>
  );
}
```

Step 5: Update Routes

Add a new route in your routing configuration:

Next.js (App Router):

```
// app/flight-tracker/page.tsx
import { Atlas3DTrackerEnhanced } from '@/components/Atlas3DTrackerEnhanced';

export default function FlightTrackerPage() {
  return <Atlas3DTrackerEnhanced />;
}
```

Next.js (Pages Router):

```
// pages/flight-tracker.tsx
import { Atlas3DTrackerEnhanced } from '@/components/Atlas3DTrackerEnhanced';

export default function FlightTrackerPage() {
  return <Atlas3DTrackerEnhanced />;
}
```

Step 6: Update GitHub Repository

```
cd /path/to/3iatlas
git add .
git commit -m "Add 3I/ATLAS Flight Tracker"
git push origin main
```

Next.js Integration

App Router (Next.js 13+)

1. **Create a dedicated page:**

```tsx
// app/tracker/page.tsx
'use client';

import dynamic from 'next/dynamic';

const Atlas3DTracker = dynamic(
  () => import('@/components/Atlas3DTrackerEnhanced').then(
    (mod) => mod.Atlas3DTrackerEnhanced
  ),
  { ssr: false }
);

export default function TrackerPage() {
  return (
    <main className="w-full h-screen">
      <Atlas3DTracker />
    </main>
  );
}
```

1. **Add to navigation:**

```tsx
// components/Navigation.tsx
export function Navigation() {
  return (
    <nav>
      <Link href="/tracker">Flight Tracker</Link>
    </nav>
  );
}
```

Pages Router (Next.js 12 and earlier)

```tsx
// pages/tracker.tsx
import dynamic from 'next/dynamic';

const Atlas3DTracker = dynamic(
  () => import('@/components/Atlas3DTrackerEnhanced').then(
    (mod) => mod.Atlas3DTrackerEnhanced
  ),
  { ssr: false }
);

export default function TrackerPage() {
  return <Atlas3DTracker />;
}
```

Metadata Configuration

```
// app/tracker/page.tsx
export const metadata = {
  title: '3I/ATLAS Flight Tracker - Ride with an Interstellar Comet',
  description: 'Experience the journey of 3I/ATLAS, the third interstellar visitor to
our solar system, in immersive 3D.',
  openGraph: {
    title: '3I/ATLAS Flight Tracker',
    description: 'Ride with an interstellar comet through our solar system',
    images: ['/og-image.png'],
  },
};
```

React Integration

Create React App

1. **Copy components:**

```
cp -r frontend/src/components src/
cp -r frontend/public/data public/
```

1. **Install dependencies:**

```
npm install @react-three/fiber @react-three/drei three react-markdown
```

1. **Use in App.tsx:**

```
import { Atlas3DTrackerEnhanced } from './components/Atlas3DTrackerEnhanced';

function App() {
  return (
    <div className="App">
      <Atlas3DTrackerEnhanced />
    </div>
  );
}

export default App;
```

Vite

Already configured! Just follow the Quick Start guide.

Component-Level Integration

As a Modal/Overlay

```
import { useState } from 'react';
import { Atlas3DTrackerEnhanced } from '@/components/Atlas3DTrackerEnhanced';

export function FlightTrackerModal() {
  const [isOpen, setIsOpen] = useState(false);

  return (
    <>
      <button onClick={() => setIsOpen(true)}>
        Open Flight Tracker
      </button>

      {isOpen && (
        <div className="fixed inset-0 z-50 bg-black">
          <button
            onClick={() => setIsOpen(false)}
            className="absolute top-4 right-4 z-50 text-white"
          >
            Close ×
          </button>
          <Atlas3DTrackerEnhanced />
        </div>
      )}
    </>
  );
}
```

Embedded in Page

```
export function LandingPage() {
  return (
    <div>
      <header>
        <h1>3I/ATLAS Mission</h1>
      </header>

      <section className="w-full h-[600px]">
        <Atlas3DTrackerEnhanced
          autoPlay={false}
          initialSpeed={1}
          initialFollowMode={false}
        />
      </section>

      <section>
        <p>More content...</p>
      </section>
    </div>
  );
}
```

Data Management

Using Existing Data

If you already have trajectory data:

```
// Custom hook to provide your own data
import { useState, useEffect } from 'react';

export function useCustomTrajectoryData() {
  const [data, setData] = useState(null);

  useEffect(() => {
    // Load from your own source
    fetch('/api/trajectory')
      .then(res => res.json())
      .then(setData);
  }, []);

  return data;
}
```

API Integration

Connect to your own backend:

```
// lib/api.ts
export async function fetchTrajectoryData(startDate: string, endDate: string) {
  const response = await fetch('/api/trajectory', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ startDate, endDate }),
  });

  return response.json();
}
```

Caching Strategy

For production, implement caching:

```
// lib/cache.ts
const CACHE_KEY = 'atlas_trajectory_cache';
const CACHE_DURATION = 7 * 24 * 60 * 60 * 1000; // 7 days

export async function getCachedTrajectory() {
  const cached = localStorage.getItem(CACHE_KEY);

  if (cached) {
    const { data, timestamp } = JSON.parse(cached);
    if (Date.now() - timestamp < CACHE_DURATION) {
      return data;
    }
  }

  const freshData = await fetchTrajectoryData();
  localStorage.setItem(CACHE_KEY, JSON.stringify({
    data: freshData,
    timestamp: Date.now(),
  }));

  return freshData;
}
```

Custom Styling

Tailwind Integration

Merge the tailwind config:

```
// tailwind.config.js
module.exports = {
  // ... your existing config
  theme: {
    extend: {
      colors: {
        'atlas-green': '#00ff88',
        'atlas-blue': '#00aaff',
      },
    },
  },
};
```

CSS Modules

Convert to CSS modules if needed:

```
// Atlas3DTracker.module.css
.container {
  width: 100%;
  height: 100vh;
  background: black;
}

// Component
import styles from './Atlas3DTracker.module.css';

export function Atlas3DTracker() {
  return <div className={styles.container}>...</div>;
}
```

Environment Variables

Create `.env.local` for configuration:

```
# API Configuration
NEXT_PUBLIC_API_URL=https://api.3iatlas.com
NEXT_PUBLIC_HORIZONS_CACHE_DURATION=604800000

# Feature Flags
NEXT_PUBLIC_ENABLE_CINEMATIC=true
NEXT_PUBLIC_ENABLE_AUDIO=false
```

Use in components:

```
const API_URL = process.env.NEXT_PUBLIC_API_URL;
const CACHE_DURATION = parseInt(
  process.env.NEXT_PUBLIC_HORIZONS_CACHE_DURATION || '604800000'
);
```

Testing

Component Testing

```tsx
// __tests__/Atlas3DTracker.test.tsx
import { render } from '@testing-library/react';
import { Atlas3DTrackerEnhanced } from '@/components/Atlas3DTrackerEnhanced';

describe('Atlas3DTrackerEnhanced', () => {
  it('renders without crashing', () => {
    const { container } = render(<Atlas3DTrackerEnhanced />);
    expect(container).toBeInTheDocument();
  });

  it('loads trajectory data', async () => {
    const { findByText } = render(<Atlas3DTrackerEnhanced />);
    const telemetry = await findByText(/3I\/ATLAS TELEMETRY/i);
    expect(telemetry).toBeInTheDocument();
  });
});
```

Performance Optimization

Code Splitting

```tsx
// Lazy load the tracker
const Atlas3DTracker = lazy(() =>
  import('@/components/Atlas3DTrackerEnhanced').then((mod) => ({
    default: mod.Atlas3DTrackerEnhanced,
  }))
);

export function TrackerPage() {
  return (
    <Suspense fallback={<LoadingSpinner />}>
      <Atlas3DTracker />
    </Suspense>
  );
}
```

Bundle Size Optimization

Add to `next.config.js`:

```js
module.exports = {
  webpack: (config) => {
    config.externals = {
      ...config.externals,
      three: 'THREE',
    };
    return config;
  },
};
```

Deployment Checklist

- [ ] Copy all component files
- [ ] Copy data files to public directory
- [ ] Install dependencies
- [ ] Test in development
- [ ] Build for production
- [ ] Test production build
- [ ] Configure CDN for data files (optional)
- [ ] Set up monitoring
- [ ] Update documentation
- [ ] Push to GitHub
- [ ] Deploy to hosting

Support

For issues or questions:
1. Check the main README.md
2. Review component documentation
3. Open an issue on GitHub
4. Contact the development team

Happy integrating! 🚀