**Understanding the Astrology Wheel**

The astrology wheel, also known as a natal chart or birth chart, is a circular diagram that represents the positions of celestial bodies at the exact moment of a person's birth. It serves as a cosmic map of the heavens from the perspective of Earth at that specific time and location.

**Key Components of the Astrology Wheel**

1. **The Zodiac Ring**: The outer circle divided into 12 equal segments representing the zodiac signs (Aries through Pisces).
2. **Houses**: The chart is further divided into 12 houses, each representing different areas of life (personality, possessions, communication, home, etc.).
3. **Planets**: Symbols representing the Sun, Moon, and planets are placed within the wheel according to their positions at birth.
4. **Aspects**: Lines connecting planets indicate relationships (aspects) between them, which can be harmonious or challenging.
5. **Angles**: The four main angles - Ascendant, Descendant, Midheaven, and Imum Coeli - represent crucial points in the chart.

**Integration with Replit Agent and Assistant**

Here's code that would integrate an interactive astrology wheel with Replit's agent and assistant features for your website:

javascript

Copy

// Astrology Wheel Component for Mystic Arcana

// This can be integrated with Replit's agent features

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const AstrologyWheel = ({ birthData, replitAgent }) => {

const [chartData, setChartData] = useState(null);

const [loading, setLoading] = useState(true);

const [selectedPlanet, setSelectedPlanet] = useState(null);

const [agentAnalysis, setAgentAnalysis] = useState(null);

// Calculate chart data based on birth information

useEffect(() => {

if (birthData) {

setLoading(true);

// Call to astrology calculation API

axios.post('/api/calculate-chart', birthData)

.then(response => {

setChartData(response.data);

setLoading(false);

})

.catch(error => {

console.error('Error calculating chart:', error);

setLoading(false);

});

}

}, [birthData]);

// When a planet is selected, request analysis from Replit agent

useEffect(() => {

if (selectedPlanet && replitAgent && chartData) {

// Get specific analysis from the Replit agent

replitAgent.analyze({

type: 'planetary\_position',

planet: selectedPlanet,

sign: chartData.planets[selectedPlanet].sign,

house: chartData.planets[selectedPlanet].house,

aspects: chartData.aspects.filter(a =>

a.planet1 === selectedPlanet || a.planet2 === selectedPlanet

)

}).then(analysis => {

setAgentAnalysis(analysis);

});

}

}, [selectedPlanet, replitAgent, chartData]);

// Render SVG-based chart

const renderChart = () => {

if (!chartData) return null;

const width = 600;

const height = 600;

const centerX = width / 2;

const centerY = height / 2;

const radius = Math.min(centerX, centerY) - 10;

return (

<svg width={width} height={height} viewBox={`0 0 ${width} ${height}`}>

{/\* Zodiac ring \*/}

{chartData.signs.map((sign, index) => {

const startAngle = index \* 30 \* Math.PI / 180;

const endAngle = (index + 1) \* 30 \* Math.PI / 180;

const x1 = centerX + radius \* Math.cos(startAngle);

const y1 = centerY + radius \* Math.sin(startAngle);

const x2 = centerX + radius \* Math.cos(endAngle);

const y2 = centerY + radius \* Math.sin(endAngle);

const largeArcFlag = 0;

const sweepFlag = 1;

const pathData = `

M ${centerX} ${centerY}

L ${x1} ${y1}

A ${radius} ${radius} 0 ${largeArcFlag} ${sweepFlag} ${x2} ${y2}

Z

`;

return (

<path

key={sign.name}

d={pathData}

fill={sign.element === 'fire' ? '#ffdddd' :

sign.element === 'earth' ? '#ddffdd' :

sign.element === 'air' ? '#ddddff' :

'#ddeeff'}

stroke="#333"

strokeWidth="1"

/>

);

})}

{/\* Houses \*/}

{chartData.houses.map((house, index) => {

const startAngle = house.startDegree \* Math.PI / 180;

const innerRadius = radius \* 0.7;

const x1 = centerX + radius \* Math.cos(startAngle);

const y1 = centerY + radius \* Math.sin(startAngle);

const x2 = centerX + innerRadius \* Math.cos(startAngle);

const y2 = centerY + innerRadius \* Math.sin(startAngle);

return (

<line

key={`house-${index + 1}`}

x1={x1}

y1={y1}

x2={x2}

y2={y2}

stroke="#666"

strokeWidth="1"

strokeDasharray="5,5"

/>

);

})}

{/\* Planets \*/}

{Object.entries(chartData.planets).map(([planet, data]) => {

const angle = data.degree \* Math.PI / 180;

const distance = radius \* 0.6; // Place planets within houses

const x = centerX + distance \* Math.cos(angle);

const y = centerY + distance \* Math.sin(angle);

return (

<g

key={planet}

transform={`translate(${x}, ${y})`}

onClick={() => setSelectedPlanet(planet)}

style={{ cursor: 'pointer' }}

>

<circle

r={15}

fill={selectedPlanet === planet ? '#ffffcc' : 'white'}

stroke="#333"

strokeWidth="1"

/>

<text

textAnchor="middle"

dominantBaseline="middle"

fontSize="12"

>

{data.symbol}

</text>

</g>

);

})}

{/\* Aspects \*/}

{chartData.aspects.map((aspect, index) => {

const planet1 = chartData.planets[aspect.planet1];

const planet2 = chartData.planets[aspect.planet2];

const angle1 = planet1.degree \* Math.PI / 180;

const angle2 = planet2.degree \* Math.PI / 180;

const distance = radius \* 0.6;

const x1 = centerX + distance \* Math.cos(angle1);

const y1 = centerY + distance \* Math.sin(angle1);

const x2 = centerX + distance \* Math.cos(angle2);

const y2 = centerY + distance \* Math.sin(angle2);

return (

<line

key={`aspect-${index}`}

x1={x1}

y1={y1}

x2={x2}

y2={y2}

stroke={

aspect.type === 'conjunction' ? '#ff0000' :

aspect.type === 'trine' ? '#00ff00' :

aspect.type === 'square' ? '#0000ff' :

aspect.type === 'opposition' ? '#ff00ff' :

'#999999'

}

strokeWidth="1"

strokeDasharray={

aspect.type === 'conjunction' ? '0' :

aspect.type === 'trine' ? '0' :

aspect.type === 'square' ? '5,5' :

aspect.type === 'opposition' ? '10,10' :

'3,3'

}

/>

);

})}

</svg>

);

};

// Integration with Replit assistant for explanations

const getAssistantHelp = () => {

// This function would trigger the Replit assistant to provide

// educational content about chart reading

if (replitAgent) {

replitAgent.requestAssistance({

topic: 'astrology\_chart\_reading',

context: chartData,

userLevel: 'beginner' // or 'intermediate', 'advanced'

});

}

};

return (

<div className="astrology-wheel-container">

{loading ? (

<div className="loading-indicator">Calculating celestial positions...</div>

) : (

<>

<div className="chart-display">

{renderChart()}

</div>

<div className="chart-controls">

<button onClick={getAssistantHelp}>

Get Help Understanding Your Chart

</button>

</div>

{selectedPlanet && (

<div className="planet-analysis">

<h3>{selectedPlanet} in {chartData.planets[selectedPlanet].sign}</h3>

{agentAnalysis ? (

<div className="agent-analysis">

<p>{agentAnalysis.interpretation}</p>

<ul>

{agentAnalysis.keywords.map((keyword, i) => (

<li key={i}>{keyword}</li>

))}

</ul>

</div>

) : (

<p>Loading analysis...</p>

)}

</div>

)}

</>

)}

</div>

);

};

export default AstrologyWheel;

**Setting Up the Replit Agent Connection**

For the Replit agent to work with your astrology application, you'd need to configure the agent with astrological knowledge. Here's a sample setup:

javascript

Copy

// replit-agent-config.js

const initReplitAgent = () => {

return {

// Configure the Replit agent with astrological knowledge

init: async () => {

const agent = new ReplitAgent({

name: "AstroGuide",

knowledgeBase: "astrology",

capabilities: [

"chart\_interpretation",

"transit\_analysis",

"compatibility\_assessment",

"predictive\_insights"

]

});

// Load custom astrological knowledge

await agent.loadKnowledge([

"zodiac\_meanings.json",

"planet\_interpretations.json",

"aspect\_definitions.json",

"house\_systems.json"

]);

return agent;

},

// Methods for the agent to analyze chart components

analyze: async (params) => {

// This would connect to Replit's AI capabilities

const response = await fetch('/api/replit-agent/analyze', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(params)

});

return response.json();

},

// Method to get explanations from the Replit assistant

requestAssistance: async (params) => {

// This would trigger the Replit assistant to provide help

const assistantWindow = window.open('', 'replit-assistant');

// Send data to the assistant

assistantWindow.postMessage({

type: 'ASTROLOGY\_ASSISTANCE\_REQUEST',

data: params

}, '\*');

}

};

};

export default initReplitAgent;

**Integration with Your Website's Backend**

On your server side, you'd need to handle chart calculations:

javascript

Copy

// server.js - Express example

const express = require('express');

const swisseph = require('swisseph');

const app = express();

app.use(express.json());

// Initialize ephemeris data

swisseph.swe\_set\_ephe\_path('./ephemeris');

app.post('/api/calculate-chart', (req, res) => {

const { date, time, latitude, longitude } = req.body;

try {

// Convert date and time to Julian day

const year = parseInt(date.split('-')[0]);

const month = parseInt(date.split('-')[1]);

const day = parseInt(date.split('-')[2]);

const hour = parseInt(time.split(':')[0]);

const minute = parseInt(time.split(':')[1]);

const julday = swisseph.swe\_julday(

year, month, day, hour + minute/60, swisseph.SE\_GREG\_CAL

);

// Calculate positions of planets

const planets = {};

const bodies = [

{ name: 'Sun', id: swisseph.SE\_SUN },

{ name: 'Moon', id: swisseph.SE\_MOON },

{ name: 'Mercury', id: swisseph.SE\_MERCURY },

{ name: 'Venus', id: swisseph.SE\_VENUS },

{ name: 'Mars', id: swisseph.SE\_MARS },

{ name: 'Jupiter', id: swisseph.SE\_JUPITER },

{ name: 'Saturn', id: swisseph.SE\_SATURN },

{ name: 'Uranus', id: swisseph.SE\_URANUS },

{ name: 'Neptune', id: swisseph.SE\_NEPTUNE },

{ name: 'Pluto', id: swisseph.SE\_PLUTO }

];

bodies.forEach(body => {

const result = swisseph.swe\_calc\_ut(julday, body.id, swisseph.SEFLG\_SPEED);

const position = result.longitude;

const sign = Math.floor(position / 30);

const degree = position % 30;

planets[body.name] = {

degree: position,

sign: getZodiacSign(sign),

position: degree.toFixed(2),

symbol: getPlanetSymbol(body.name)

};

});

// Calculate houses (Placidus system)

const houses = swisseph.swe\_houses(

julday, latitude, longitude, 'P'

);

// Format houses data

const formattedHouses = houses.cusps.map((cusp, index) => ({

number: index + 1,

startDegree: cusp

}));

// Calculate aspects between planets

const aspects = calculateAspects(planets);

// Prepare zodiac signs information

const signs = [

{ name: 'Aries', element: 'fire', symbol: '♈' },

{ name: 'Taurus', element: 'earth', symbol: '♉' },

{ name: 'Gemini', element: 'air', symbol: '♊' },

{ name: 'Cancer', element: 'water', symbol: '♋' },

{ name: 'Leo', element: 'fire', symbol: '♌' },

{ name: 'Virgo', element: 'earth', symbol: '♍' },

{ name: 'Libra', element: 'air', symbol: '♎' },

{ name: 'Scorpio', element: 'water', symbol: '♏' },

{ name: 'Sagittarius', element: 'fire', symbol: '♐' },

{ name: 'Capricorn', element: 'earth', symbol: '♑' },

{ name: 'Aquarius', element: 'air', symbol: '♒' },

{ name: 'Pisces', element: 'water', symbol: '♓' }

];

res.json({

planets,

houses: formattedHouses,

aspects,

signs

});

} catch (error) {

console.error(error);

res.status(500).json({ error: 'Failed to calculate chart' });

}

});

// Helper functions

function getZodiacSign(index) {

const signs = ['Aries', 'Taurus', 'Gemini', 'Cancer', 'Leo', 'Virgo',

'Libra', 'Scorpio', 'Sagittarius', 'Capricorn', 'Aquarius', 'Pisces'];

return signs[index % 12];

}

function getPlanetSymbol(planet) {

const symbols = {

'Sun': '☉',

'Moon': '☽',

'Mercury': '☿',

'Venus': '♀',

'Mars': '♂',

'Jupiter': '♃',

'Saturn': '♄',

'Uranus': '♅',

'Neptune': '♆',

'Pluto': '♇'

};

return symbols[planet] || planet.charAt(0);

}

function calculateAspects(planets) {

const aspects = [];

const planetNames = Object.keys(planets);

// Aspect configurations (type, angle, orb)

const aspectTypes = [

{ type: 'conjunction', angle: 0, orb: 8 },

{ type: 'opposition', angle: 180, orb: 8 },

{ type: 'trine', angle: 120, orb: 8 },

{ type: 'square', angle: 90, orb: 8 },

{ type: 'sextile', angle: 60, orb: 6 }

];

// Check for aspects between all planet pairs

for (let i = 0; i < planetNames.length; i++) {

for (let j = i + 1; j < planetNames.length; j++) {

const planet1 = planetNames[i];

const planet2 = planetNames[j];

const pos1 = planets[planet1].degree;

const pos2 = planets[planet2].degree;

// Calculate angle between planets

let angle = Math.abs(pos1 - pos2);

if (angle > 180) angle = 360 - angle;

// Check if angle matches any aspect type

for (const aspectType of aspectTypes) {

if (Math.abs(angle - aspectType.angle) <= aspectType.orb) {

aspects.push({

planet1,

planet2,

type: aspectType.type,

angle: angle.toFixed(2),

orb: Math.abs(angle - aspectType.angle).toFixed(2)

});

break;

}

}

}

}

return aspects;

}

app.listen(3000, () => {

console.log('Astrology server running on port 3000');

});

**Replit Assistant Integration**

For the Replit assistant to provide explanations about the chart, you would set up a dedicated component:

javascript

Copy

// AstrologyAssistant.js

import React, { useEffect, useState } from 'react';

const AstrologyAssistant = ({ replitAgent, chartData, userLevel }) => {

const [isActive, setIsActive] = useState(false);

const [assistantMessage, setAssistantMessage] = useState('');

const [userQuery, setUserQuery] = useState('');

// Listen for messages from the assistant

useEffect(() => {

const handleMessage = (event) => {

if (event.data.type === 'ASTROLOGY\_ASSISTANT\_RESPONSE') {

setAssistantMessage(event.data.message);

}

};

window.addEventListener('message', handleMessage);

return () => window.removeEventListener('message', handleMessage);

}, []);

const activateAssistant = () => {

setIsActive(true);

// Initial greeting

setAssistantMessage("Hello! I'm your astrology assistant. What would you like to know about your chart?");

};

const askAssistant = () => {

if (!userQuery.trim()) return;

// Send query to Replit agent

replitAgent.requestAssistance({

topic: 'chart\_interpretation',

query: userQuery,

context: chartData,

userLevel

});

setUserQuery('');

};

if (!isActive) {

return (

<button

className="activate-assistant-btn"

onClick={activateAssistant}

>

Get Astrological Guidance

</button>

);

}

return (

<div className="astrology-assistant">

<div className="assistant-header">

<h3>Celestial Guide</h3>

<button onClick={() => setIsActive(false)}>Minimize</button>

</div>

<div className="assistant-messages">

{assistantMessage && (

<div className="assistant-message">

{assistantMessage}

</div>

)}

</div>

<div className="assistant-input">

<input

type="text"

value={userQuery}

onChange={(e) => setUserQuery(e.target.value)}

placeholder="Ask about your chart..."

/>

<button onClick={askAssistant}>Send</button>

</div>

<div className="quick-questions">

<button onClick={() => setUserQuery("What does my Sun sign mean?")}>

Sun Sign

</button>

<button onClick={() => setUserQuery("Explain my Moon placement")}>

Moon Placement

</button>

<button onClick={() => setUserQuery("What are the strongest aspects in my chart?")}>

Key Aspects

</button>

</div>

</div>

);

};

export default AstrologyAssistant;

**Putting It All Together**

To integrate these components on your Mystic Arcana website:

javascript

Copy

// App.js or your main component

import React, { useState, useEffect } from 'react';

import AstrologyWheel from './components/AstrologyWheel';

import AstrologyAssistant from './components/AstrologyAssistant';

import initReplitAgent from './utils/replit-agent-config';

const AstrologyPage = () => {

const [birthData, setBirthData] = useState(null);

const [replitAgent, setReplitAgent] = useState(null);

const [userLevel, setUserLevel] = useState('beginner');

// Initialize Replit agent on component mount

useEffect(() => {

const setupAgent = async () => {

const agent = await initReplitAgent().init();

setReplitAgent(agent);

};

setupAgent();

}, []);

const handleSubmitBirthData = (formData) => {

setBirthData(formData);

};

return (

<div className="astrology-page">

<h1>Your Cosmic Blueprint</h1>

{!birthData ? (

<BirthDataForm onSubmit={handleSubmitBirthData} />

) : (

<div className="chart-container">

<AstrologyWheel

birthData={birthData}

replitAgent={replitAgent}

/>

<div className="chart-settings">

<label>

Experience Level:

<select

value={userLevel}

onChange={(e) => setUserLevel(e.target.value)}

>

<option value="beginner">Beginner</option>

<option value="intermediate">Intermediate</option>

<option value="advanced">Advanced</option>

</select>

</label>

</div>

<AstrologyAssistant

replitAgent={replitAgent}

chartData={birthData}

userLevel={userLevel}

/>

</div>

)}

</div>

);

};

// Form component to collect birth data

const BirthDataForm = ({ onSubmit }) => {

const [formData, setFormData] = useState({

name: '',

date: '',

time: '',

latitude: '',

longitude: '',

location: ''

});

const handleChange = (e) => {

setFormData({

...formData,

[e.target.name]: e.target.value

});

};

const handleSubmit = (e) => {

e.preventDefault();

onSubmit(formData);

};

return (

<form onSubmit={handleSubmit} className="birth-data-form">

<h2>Enter Your Birth Details</h2>

<div className="form-group">

<label htmlFor="name">Name</label>

<input

type="text"

id="name"

name="name"

value={formData.name}

onChange={handleChange}

required

/>

</div>

<div className="form-group">

<label htmlFor="date">Birth Date</label>

<input

type="date"

id="date"

name="date"

value={formData.date}

onChange={handleChange}

required

/>

</div>

<div className="form-group">

<label htmlFor="time">Birth Time (if known)</label>

<input

type="time"

id="time"

name="time"

value={formData.time}

onChange={handleChange}

/>

</div>

<div className="form-group">

<label htmlFor="location">Birth Location</label>

<input

type="text"

id="location"

name="location"

value={formData.location}

onChange={handleChange}

placeholder="City, Country"

required

/>

</div>

<button type="submit" className="submit-btn">

Generate Chart

</button>

</form>

);

};

export default AstrologyPage;

This comprehensive implementation provides a foundation for integrating an interactive astrology wheel with Replit's agent and assistant capabilities on your Mystic Arcana website. The system allows users to generate and explore their birth charts while receiving personalized interpretations and guidance from the AI-powered assistant.