Go Alchemy API Documentation

This document provides an overview of the main APIs and components used in the Go Alchemy application.

Table of Contents

Core Contexts

GameContext

Manages the Go game logic and board state.

```
interface GameContextType {
  board: Board;
  boardState: BoardState;
  currentPlayer: StoneColor;
  boardSize: number;
  placeStone: (vertex: Vertex) => Promise<boolean>;
  isValidMove: (vertex: Vertex) => boolean;
  autoPlayOpponent: boolean;
  setAutoPlayOpponent: (value: boolean) => void;
  autoPlayDelay: number;
  setAutoPlayDelay: (value: number) => void;
}
```

Usage:

```
const { board, placeStone, currentPlayer } = useGame();
```

GameTreeContext

Handles SGF game tree navigation and problem management.

```
interface GameTreeContextType {
  isLoading: boolean;
  gameTree: GameTreeType | null;
  currentNode: GameTreeNode | null;
  startingNode: GameTreeNode | null;
  currentComment: string | null;
  addMove: (vertex: Vertex, currentPlayer: Sign) => void;
  navigate: {
    forward: () => void;
    backward: () => void;
    first: () => void;
    last: () => void;
  };
  canNavigate: {
    forward: boolean;
    backward: boolean;
  };
  boardSize: number;
  range: BoardRange;
}
```

Usage:

```
const { currentNode, navigate, canNavigate } = useGameTree();
```

TransformContext

Manages board transformations (rotation, mirroring, color inversion).

```
interface TransformContextType {
  transformation: BoardTransformation;
  setTransformation: (transform: BoardTransformation) => void;
  randomizeTransformation: () => void;
}
```

ProblemContext

Manages problem sets and navigation between problems.

```
interface ProblemContextType {
  problemIds: string[];
  category: string;
  currentProblemIndex: number;
  setCurrentProblemIndex: (index: number) => void;
}
```

Components

GoBoard

The main game board component that renders the Go board using SVG.

Props:

```
interface GoBoardProps {
  availableWidth: number;
  availableHeight: number;
}
```

Features:

- Touch interaction for stone placement
- Board grid and star points rendering

- Stone display with realistic images
- Board marks (circles, triangles, squares, X marks)
- Optional coordinate labels
- Hint highlighting
- Hover effects

ControlPanel

Navigation controls for problems and moves.

Features:

- Problem navigation (previous/next)
- Move navigation (forward/backward/first/last)
- Hint toggle
- Responsive button states

CommentDisplay

Shows problem descriptions and move feedback.

Features:

- Displays SGF comments
- Success/failure messages
- Problem instructions

State Management

Redux Store Structure

```
interface RootState {
  date: DateState;
  settings: SettingsState;
  user: UserState;
}
```

Settings Slice

```
interface SettingsState {
  darkMode: boolean;
  sfxEnabled: boolean;
  hapticsEnabled: boolean;
  showHint: boolean;
  showCoordinates: boolean;
  randomizeBoard: boolean;
}
```

Actions:

- `setDarkMode(boolean)`
- `toggleSfx()`
- `toggleHaptics()`
- `toggleShowHint()`
- `resetShowHint()`
- `toggleShowCoordinates()`
- `toggleRandomizeBoard()`
- `resetSettings()`

Utilities

sgfUtils

Coordinate conversion utilities.

```
// Convert SGF coordinates to vertex
sgfToVertex(sgfCoord: string): Vertex
// Example: sgfToVertex("pd") => [15, 3]

// Convert vertex to SGF coordinates
vertexToSgf(vertex: Vertex): string
// Example: vertexToSgf([15, 3]) => "pd"
```

sgfLoader

Load SGF files from the app bundle.

```
loadSgfFromAssets(category: string, filename: number): Promise<string>
```

boardTransformation

Transform vertices for board rotation/mirroring.

```
transformVertex(
  vertex: Vertex,
  transformation: BoardTransformation,
  boardSize: number,
  inverse?: boolean
): Vertex
```

Types

Board Types

```
type Stone = 0 | 1 | -1;
type BoardState = Stone[];
type Vertex = [number, number];
type Sign = -1 | 0 | 1;

interface BoardRange {
  startX: number;
  startY: number;
  endX: number;
  endY: number;
}

interface BoardTransformation {
  rotate: number;
```

```
  flipX: boolean;
  flipY: boolean;
  invertColors: boolean;
}
```

SGF Types

```
interface SGFProblem {
  id: string;
  filename: string;
  category?: string;
  difficulty?: string;
  description?: string;
}
```

Navigation Routes

The app uses Expo Router with the following main routes:

- `/` - Home screen
- `/problems` - Problem category selection
- `/problems/[category]` - Problems in a category
- `/problems/problem/[id]` - Individual problem viewer
- `/daily` - Daily problems
- `/daily/problem/[id]` - Daily problem viewer
- `/settings` - App settings
- `/about` - About page

Best Practices

1. **Context Usage**: Always use contexts within their providers
2. **Type Safety**: Use TypeScript types for all props and state
3. **Performance**: Use React.memo and useCallback for expensive renders
4. **Error Handling**: Wrap components in ErrorBoundary
5. **State Updates**: Use Redux actions for settings, contexts for game state

Examples

Placing a Stone

```
const { placeStone, isValidMove } = useGame();

const handleMove = async (vertex) => {
  if (isValidMove(vertex)) {
    const success = await placeStone(vertex);
    if (success) {
      console.log('Move placed successfully');
    }
  }
};
```

Navigating Problems

```
const { navigate, canNavigate } = useGameTree();

if (canNavigate.forward) {
  navigate.forward();
}

navigate.first();
```

Accessing Settings

```
const dispatch = useDispatch();
const showCoordinates = useSelector(state => state.settings.showCoordinates);

dispatch(toggleShowCoordinates());
```