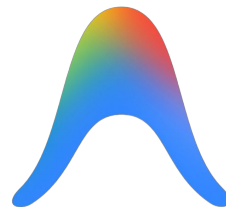




Building a simple Chess Game with Google Antigravity

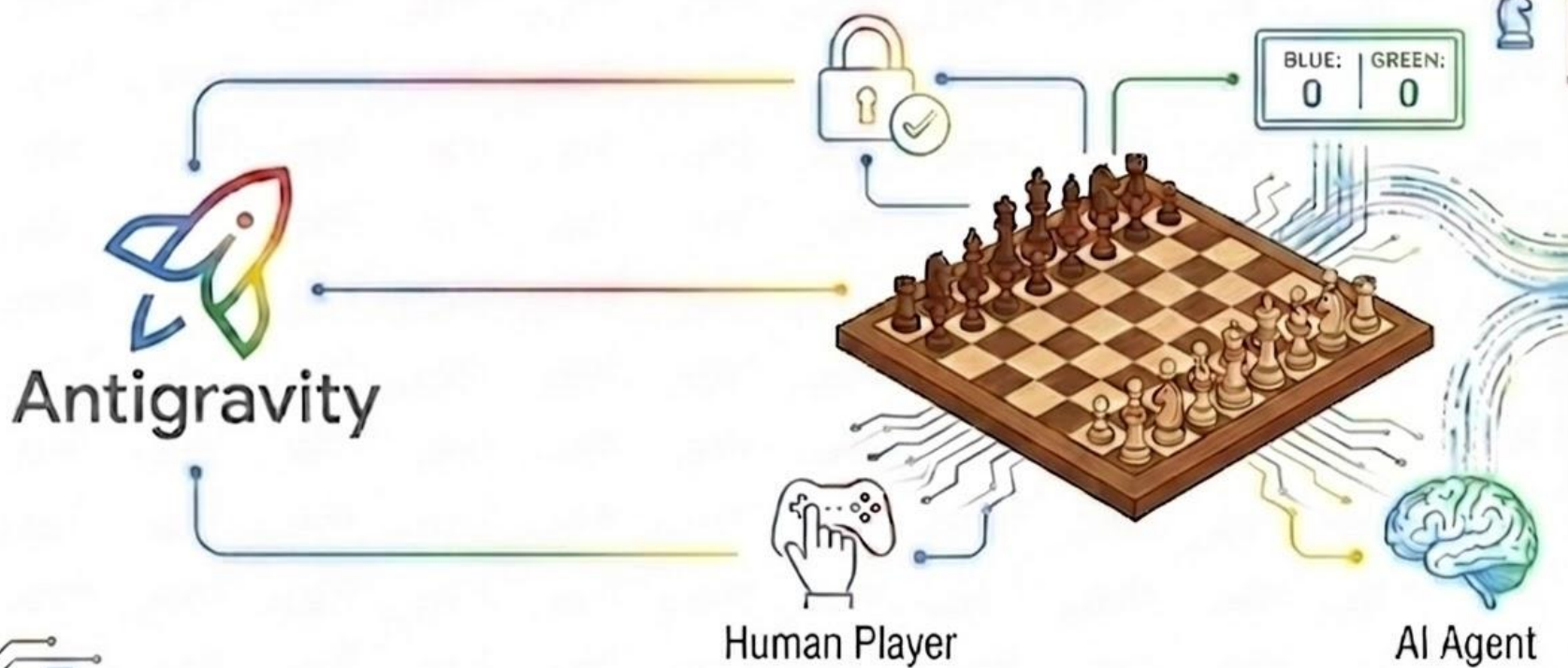


Nov 2025

acourbin@google.com

BUILDING AN AI CHESS GAME

With Google Cloud's Antigravity (Agent-1st coding Dev App)



Why developers care?

Why should developers care about Google releasing a new AI developer application

1

All developers always use these **three tools**

IDE, Terminal, Browser (often)

2

Give developers a **unified agentic experience**

Bring IDE + Terminal + Browser into a all-in-one experience

3

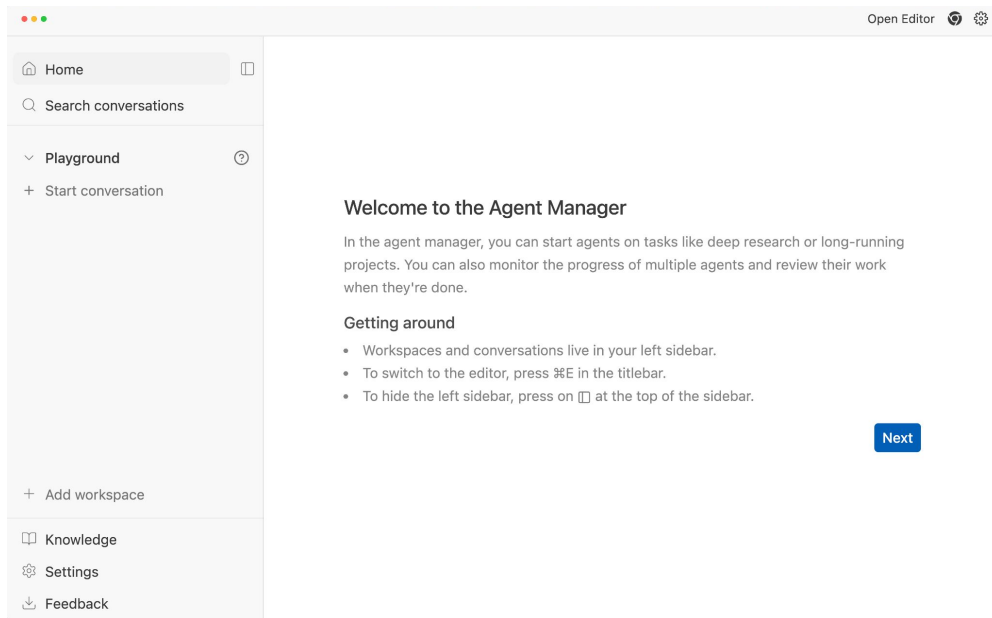
Agent first workflow

Start with goals, not code. Run and manage multiple agents (let agents do heavy lifting)

What is Google Antigravity

Antigravity is Google's AI agent-first development application

- **Agentic development platform**
- Agent first view
- Can orchestrate fleets of agents
- Workflow based instead of coding
- Seamless integration between workflows, editor, terminal and browser
- Supports synchronous (Editor) and asynchronous (Manager) workflows + handoffs
- Built on VS Code



Features & Capabilities

- **Asynchronous, local agents** that can work in parallel on all of your workspaces.
- Agent leverages planning **Artifacts** and **Verification** process to ensure quality and functional results.
- New **Agent Manager** view for an agent-first experience built around planning mode, the conversation UI, and artifact review.
- A multi-window product with a **Manager, Browser(s), and Editor(s)**
- Agent that can browse the web for you and with you to **accomplish dev tasks** like dashboard reads, SCM actions, UI testing, etc. in the Browser.

How do you want to use Antigravity - Insiders?

Select one of the options below

☐ Agent-driven development

☒ Agent-assisted development

Agent will use its own judgement on when to run terminal commands locally and when to request for a review when making changes. In addition, the browser will be pre-configured with an allowlist of developer-focused websites.

Settings can be changed later in the settings window.

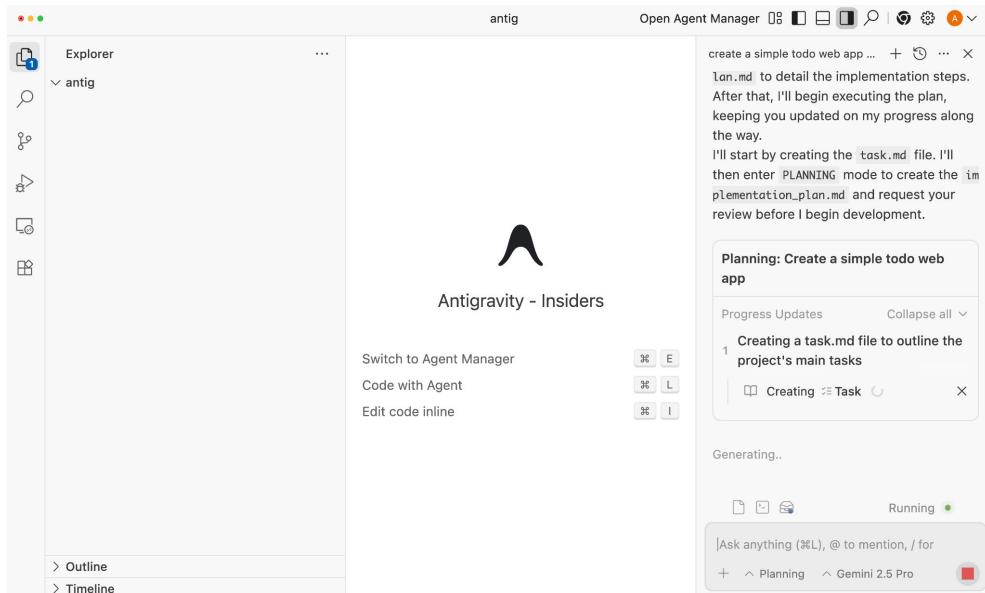
☐ Review-driven development

☐ Custom configuration



Core Surfaces

- **Agent Manager** - an orchestration “no code” view to start and view tasks in a minimalist product focused on the conversation and artifacts
- **Editor** - a “code-view” IDE experience (built on VSCode) for viewing, editing and working with codebase
- **Browser** - browser-use agent capabilities to read & actuate on more surfaces beyond just the IDE
- **Terminal** - Single command and long running terminal sessions can be initiated from agent conversations.



Common use cases

- **End-to-End Software Development** – Accelerate core engineering tasks ranging from complex enterprise refactors and background bug fixes to drafting new 0→90% features.
- **Quality Assurance & Testing** – Automate self-verifying unit test generation, validate mobile UIs across varying screen dimensions, and resolve CI errors asynchronously.
- **UI/UX Design & Prototyping** – Rapidly convert PRDs, Figma designs, or existing application screenshots directly into functional, iterative frontend code.
- **Research & Data Analysis** – Deploy parallel agents to aggregate web information for SRE reports, generate data science visualizations, or create architectural diagrams.
- **Asynchronous Task Automation** – Offload repetitive low-context work, such as clearing inbox tasks or monitoring global artifacts, without disrupting primary workflows.
- And many more ...

Initial Prompt in Antigravity Prompt Panel

I want to create a new chess game with green and blue colors. Can you craft one game with control for authorized moves and counter for points? One player player would be played automatically by an AI agent you will create.

Authorized Moves



Points Counter

BLUE:	GREEN:
0	0



Human Player



AI Agent

Review ▾

With AI

Game with a Green vs. Blue color theme. The game will feature:
Reference to chess movement rules (no sandbox mode).
Advantage or captured pieces.
Mode against an AI agent.

HTML file (or a set of files if preferred, but starting with one

board and UI elements.

Score/captured pieces for Blue and Green.

It is and game state (Check, Checkmate).

i want to create a new chess game with green and blue colors. can you craft one game with control for authorized moves and counter for points. one player would be played automatically by an AI agent you will create.

> Thought for 3s

📁 Analyzed ~/antigravity

📄 Analyzed `chess.html#L1-238`

📄 Analyzed `chess1#L1`

> Thought for 10s

📖 Edited Task

View

Planning Chess Game

Started the task. Created task.md. Now creating implementation plan.

Files Edited

📄 Implementation Plan

Progress Updates

Expand all <

1 Creating implementation plan

I have created a plan to build the Green vs. Blue Chess game with AI. I will

Implementation Plan X

chess_ai.html

Task

Implementation Plan 44 secs ago

Review ▾

Green vs. Blue Chess with AI



Goal Description

Create a fully functional Chess game with a Green vs. Blue color theme. The game will feature:

- **Authorized Moves:** Strict adherence to chess movement rules (no sandbox mode).
- **Point Counter:** Track material advantage or captured pieces.
- **AI Opponent:** A single-player mode against an AI agent.

Proposed Changes

New File: `chess_ai.html`

I will create a new self-contained HTML file (or a set of files if preferred, but starting with one for simplicity) that includes:

HTML Structure

- **Game Container:** Holds the board and UI elements.
- **Scoreboard:** Displays current score/captured pieces for Blue and Green.
- **Status Area:** Shows whose turn it is and game state (Check, Checkmate).
- **Controls:** Reset button, AI difficulty toggle (optional).

CSS (Styles)

- **Theme:** Deep Blue vs. Vibrant Green.
- **Board:** Checkered pattern with thematic colors.
- **Pieces:** Unicode chess pieces styled with CSS (shadows, colors).

Develop Green and Blue Chess

+ ↺ ⋮ ✕

Files Edited

Implementation Plan

Progress Updates

Expand all <

1 Creating implementation plan

I have created a plan to build the Green vs. Blue Chess game with AI. I will create a new file `chess_ai.html` to implement the game logic, AI, and UI. Please review the plan.

Implementation Plan

Open

Plan to create a Green vs. Blue Chess game with AI and authorized moves.



Proceeded with Implementation Plan

Auto-proceeded by the agent under your review policy.

Agent Decides ^

> Thought for 38s

Core Game Logic

Completed planning phase. User approved implementation plan. Now starting execution of core game logic.

Files Edited

chess_ai.html Task

Progress Updates

Collapse all ▾

1 Initializing project structure and board representation



Edited Task

View

Task: Green vs. Blue Chess with AI (1st Artifact : Task)

Planning & Setup

- ✓ Create implementation plan
- ✓ Initialize project structure

Core Game Logic

- ✓ Implement board representation
- ✓ Implement move validation (Authorized moves)
- ✓ Implement game state (Check, Checkmate, Turn)
- ✓ Implement Point Counter

AI Agent

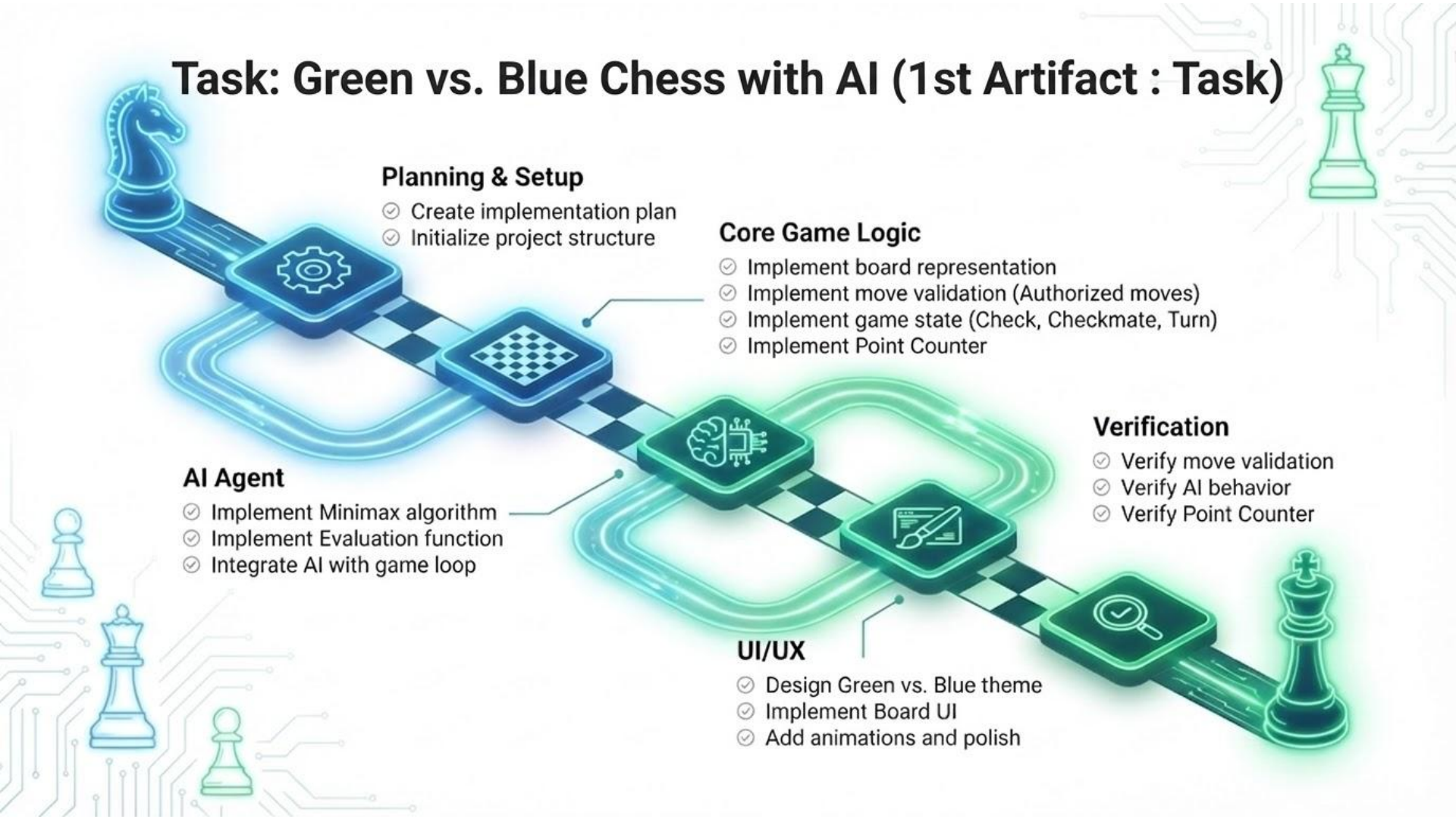
- ✓ Implement Minimax algorithm
- ✓ Implement Evaluation function
- ✓ Integrate AI with game loop

Verification

- ✓ Verify move validation
- ✓ Verify AI behavior
- ✓ Verify Point Counter

UI/UX




- ✓ Design Green vs. Blue theme
- ✓ Implement Board UI
- ✓ Add animations and polish



Green vs. Blue Chess with AI (2nd Artifact : Implementation Plan)

Goal Description

Create a fully functional Chess game with a **Green** vs. **Blue** color theme. The game will feature:





-  **Authorized Moves:** Strict adherence to chess movement rules (no sandbox mode).
-  **Point Counter:** Track material advantage or captured pieces.
-  **AI Opponent:** A single-player mode against an AI agent.

Proposed Changes




New File: `chess_ai.html`

I will create a new self-contained HTML file (or a set of files if preferred, but starting with one for

HTML Structure

-  **Game Container:** Holds the board and UI elements.
-  **Scoreboard:** Displays current score/captured pieces for Blue and Green.
-  **Status Area:** Shows whose turn it is and game state (Check, Checkmate).
-  **Controls:** Reset button, AI difficulty toggle (optional).

CSS (Styles)

-  **Theme:** Deep Blue vs. Vibrant Green.
-  **Board:** Checkered pattern with thematic colors.
-  **Pieces:** Unicode chess pieces styled with CSS (shadows, colors).




Proposed Changes

New File: `chess_ai.html`



CSS (Styles)

-  **Theme:** Deep Blue vs. **Vibrant Green**.
-  **Board:** Checkered pattern with thematic colors.
-  **Pieces:** Unicode chess pieces styled with CSS (shadows, colors).

JavaScript (Logic)

-  **ChessGame Class:** Manage 8x8 board state, validate moves, handle special moves, detect Check/Checkmate.
-  **ChessAI Class:** Minimax algorithm with Alpha-Beta pruning, evaluation function.
-  **UI Controller:** Render board, handle user input, animate moves, update scores.

Verification Plan

-  **Automated Tests:** Small internal test suite for move generation logic.
-  **Manual Verification:** Visual check, move validation, AI behavior, game end, scoreboard.



Green vs. Blue Chess AI - 3rd Artifact : Walkthrough



I have successfully created a new Chess game with a Green vs. Blue theme, featuring an AI opponent and authorized move validation.

Features Implemented



1. Core Game Logic

- **Authorized Moves:** Standard rules enforced
- **Check & Checkmate:** Detection and prevention of illegal moves
- **Special Moves:** Pawn promotion (auto-queen) implemented



2. AI Opponent

- **Minimax Algorithm:** Alpha-Beta pruning
- **Evaluation Function:** Based on material & piece safety
- **Play vs AI:** Blue (You) vs. Green (AI)



3. UI/UX

- **Green vs. Blue Theme:** Custom dark theme
- **Point Counter:** Scoreboard tracks captured pieces
- **Captured Pieces:** Visual display
- **Move Highlights:** Valid moves & last move indicated



How to Play

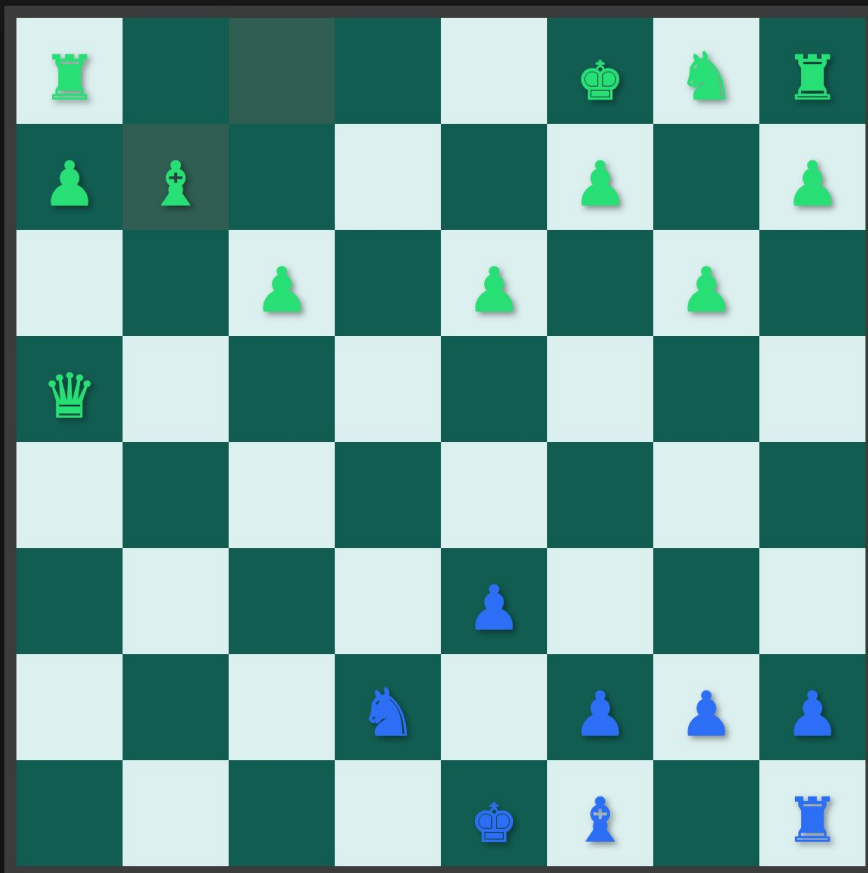
- Open **chess_ai.html** in your browser.
- Open chess_ai.html in your browser.
- Blue (You) starts first. Click to select a piece.
- Click on a highlighted square to move.
- The AI (Green) will automatically respond.
- Game ends on Checkmate or Stalemate.



Verification

- **Move Validation:** Pieces cannot move illegally.
- **Check Safety:** King is protected from check.
- **AI Response:** AI makes valid, strategic moves.
- **Scoreboard:** Scores update correctly.

Green vs. Blue Chess



Turn: **BLUE**

Blue Score: 8

Green Score: 24

Captured by Blue:



Captured by Green:



New Game

Toggle AI (Currently: ON)

Some small improvements : UX first

Color Choice Widget (5 Colors)



- Add widget for board & piece color selection.
- Offer 5 distinct color choices.

Skin Choice Widget (3 Themes)



- Enhance look & feel with skin options.
- Choose from: Game of Thrones, Star Wars, Avengers.

Some small improvements - UX fist : the prompt used

" ready to produce an iteration of version 2 of the game ? i will give you some more instructions to improve some features. First of all you will produce another file for this version based on the current one which we will call version 1. For version 2 you will boost the UI with adding those options for the user : add a widget to choose color for the chess board and the pieces (offer choice of 5 colors). Next you will enhance the look & feel of the pieces : add another widget to let the user choose between 3 possible skins : one inspired from "game of thrones", second from "star wars" and last one from "avengers". "

Task: Version 2 (Themes & Skins)

☒ Setup

- ☒ Create `</>` chess_ai_v2.html from V1

☒ UI Features

- ☒ Implement Color Theme Picker (5 options)
- ☒ Implement Skin Picker (GoT, Star Wars, Avengers)

☒ Implementation

- ☒ Update CSS to support dynamic variables
- ☒ Create Skin/Emoji mappings
- ☒ Wire up UI widgets to Game Class

☒ Verification

- ☒ Verify Theme switching
- ☒ Verify Skin switching

Task: Skin Differentiation

☒ Implementation

- ☒ Update SKINS structure for w/b variants
- ☒ Update render logic to use specific skin variant
- ☒ Update UI previews

Task: UI Readability & Dynamic Names

☒ Implementation

- ☒ Update THEMES with button text color & player names
- ☒ Update CSS to use dynamic button text color
- ☒ Update HTML/JS to render dynamic player names

Chess AI Version 2: Themes & Skins

• Goal Description & UI Additions



• Technical Implementation & Verification



Proposed Changes & CSS Updates

New File: chess_ai_v2.html
Refactor CSS variables & define 5 palettes.



JavaScript Logic

Skin Mapping
(Emojis/Icons), Theme
Switcher & Render Update



Verification Plan

Manual Checks: Theme & Skin
Switching, Gameplay Functionality.

CHESS AI V2: THEMES & SKINS - WALKTHROUGH

I have created Version 2 of the Chess AI game, adding extensive customization options for a more engaging experience.

NEW FEATURES IN VERSION 2

1. Color Theme Picker



NEON (Default):
Green vs. Blue



CLASSIC:
Wood-styled



MIDNIGHT:
Ultra-dark
monochrome



OCEAN:
Blue and teal



LAVENDER:
Soft purple
and yellow

2. Piece Skin Picker



STANDARD:
Classic Unicode



GAME OF THRONES:
Wolves, Dragons,
Swords



STAR WARS:
Robots, Rockets,
Aliens



AVENGERS:
Superheroes,
Shields, Hammers

HOW TO USE & CORE FEATURES

How to Use

- Open `chess_ai_v2.html` in your browser.
- Settings Section** in the right-hand panel.
- Change Theme:** Click colored circles.
- Change Skin:** Click icons (King symbol).
- Game state is preserved when switching!

Core Features (Retained from V1)

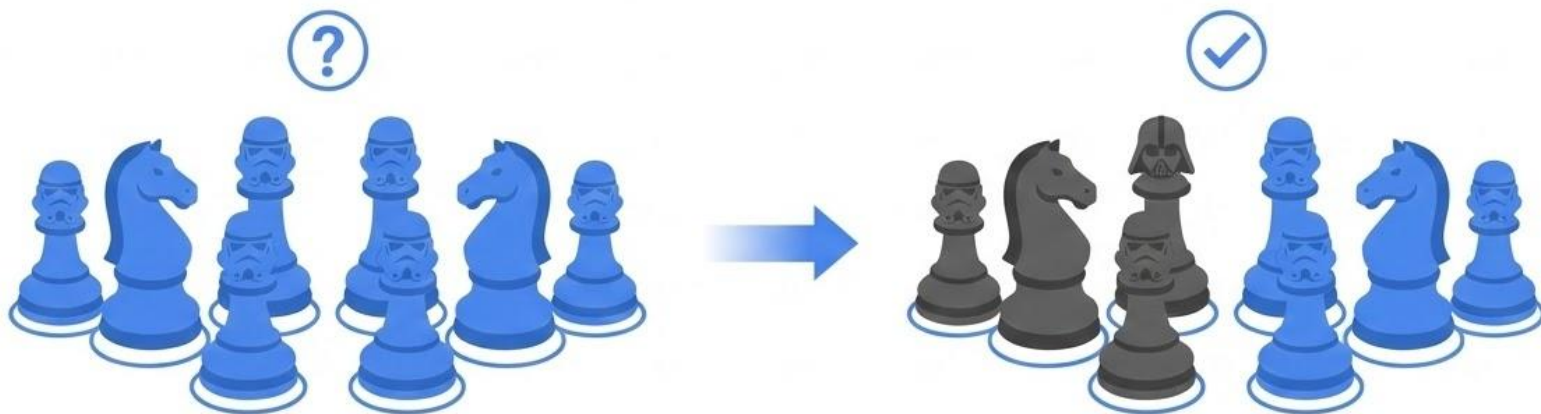
- AI Opponent:** Play against the Green (Black) AI.
- Authorized Moves:** Strict chess rules enforced.
- Scoreboard:** Tracks captured pieces and score.



`chess_ai_v2.html` (Complete V2 game)
`chess_ai.html` (Original V1 preserved)

MORE ADJUSTMENTS

“make a little change : for different skins, be sure we can differentiate between the 2 opponents because now they are looking same with no means to figure out which belongs to which player”



MORE ADJUSTMENTS

“make another adjustment : control that the fonts of words inside widgets are different from the background color so it is easily readable (avoid white font on white background for example, for “new game” button or “toggle AI”. Adapt also the name of players for the color theme chosen when counting score or to indicate whose turn is it to play.

POOR CONTRAST ❌

NEW GAME

TOGGLE AI

PLAYER 1

0

GOOD CONTRAST ✅

NEW GAME

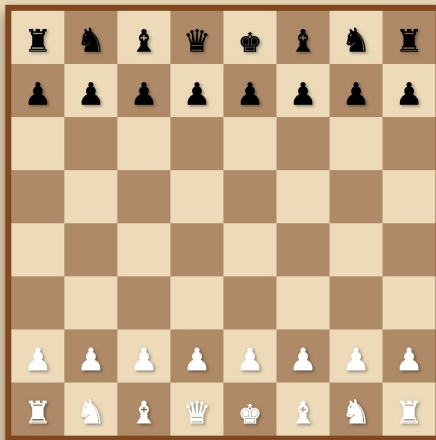
TOGGLE AI

PLAYER 1

0

...and the result is...

Chess AI V2



Turn: **WHITE**

WHITE: 0

BLACK: 0

Captured by WHITE:

Captured by BLACK:

Color Theme



Piece Skin



New Game

Toggle AI (Currently: ON)

Chess AI V2



Turn: **BLUE**

BLUE: 0

GREEN: 0

Captured by BLUE:

Captured by GREEN:

Color Theme



Piece Skin



New Game

Toggle AI (Currently: ON)

Chess AI V2



Turn: **DEEP**

DEEP: 0

TEAL: 0

Captured by DEEP:

Captured by TEAL:

Color Theme



Piece Skin



New Game

Toggle AI (Currently: ON)

Chess AI V2



Turn: **ROYAL**

ROYAL: 0

GOLD: 0

Captured by ROYAL:

Captured by GOLD:

Color Theme



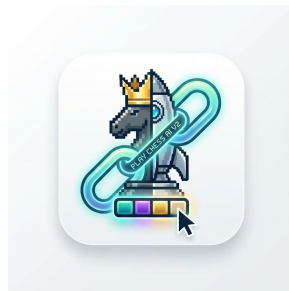
Piece Skin



New Game

Toggle AI (Currently: ON)

- > *Click to play* < -



<https://pandeva72.github.io/chessai/>