AI Game Builder — Getting Started Guide

From zero to a playable game in 10 minutes.

Prerequisites

Before you begin, install these three tools:

Tool	Version	Install
Godot Engine	4.3+	godotengine.org/download — download the **Standard** version
Node.js	20+	nodejs.org — use the LTS version
Claude Code	Latest	`npm install -g @anthropic-ai/claude-code`

Verify everything is installed:

```
godot --version      # Should show 4.3.x or higher
node --version       # Should show v20.x or higher
claude --version     # Should show the Claude Code version
```

Step 1: Get the AI Game Builder

Clone or download the AI Game Builder to your machine:

```
git clone <your-repo-url> ~/ai-game-builder
```

Or if you received it as a zip, unzip it anywhere you like. This is a **Claude Code plugin** — it contains skills, hooks, MCP config, and more:

```
ai-game-builder/
├── .claude-plugin/    # Plugin manifest (makes this a Claude Code plugin)
├── skills/            # 14 game generation skills
├── hooks/             # Stop hook (build guard)
├── .mcp.json          # MCP server configuration
├── mcp-server/        # MCP bridge (Claude ↔ Godot)
├── godot-plugin/      # Godot editor plugin
├── knowledge/         # GDScript reference, patterns, etc.
├── setup.sh           # Godot editor plugin installer
└── README.md
```

Step 2: Create a New Godot Project

Open the Godot editor. You'll see the **Project Manager**.

1. Click **"New Project"**
2. Set the **Project Name** (e.g., `my-first-game`)

3. Choose a **Project Path** (e.g., `~/games/my-first-game`)
4. Select **Renderer**: choose **Forward+** (default) or **Compatibility** for 2D games
5. Click **"Create & Edit"**

Godot opens the editor with an empty project. This is fine — Claude will create all the files.

> **Tip**: You can close the Godot editor after creating the project. The setup script only needs the project folder to exist with a `project.godot` file. You'll reopen Godot in Step 4.

Step 3: Install the Godot Editor Plugin

The setup script installs the HTTP bridge plugin into your Godot project:

```
cd ~/ai-game-builder
./setup.sh ~/games/my-first-game
```

Replace `~/games/my-first-game` with the actual path to your Godot project.

```
================================================
  AI Game Builder — Godot Editor Setup
================================================

Godot project: /Users/you/games/my-first-game

Installing MCP server dependencies...
✓ MCP server ready
Installing Godot editor plugin...
✓ Plugin installed to /Users/you/games/my-first-game/addons/ai_game_builder/
Installing knowledge base...
✓ Knowledge base installed

================================================
  Godot editor setup complete!
================================================
```

What just happened: - The Godot HTTP bridge plugin was copied into your project's `addons/` folder - MCP server dependencies were installed (Node.js) - A knowledge base was added with GDScript references and game patterns

---

Step 4: Enable the Plugin in Godot

1. Open your project in the **Godot editor** (double-click it in the Project Manager)
2. Go to **Project → Project Settings**
3. Click the **Plugins** tab at the top
4. Find **"AI Game Builder"** in the list
5. Check the **Enable** checkbox

You should see a new **"AI Game Builder"** panel appear in the bottom dock. It shows: - A green dot when the HTTP bridge is active (port 6100) - A log of all commands received from Claude Code

> **Important**: Keep Godot open. Claude Code communicates with it in real-time through the plugin.

Step 5: Open Claude Code with the Plugin

Open a new terminal and navigate to your Godot project:

```
cd ~/games/my-first-game
claude --plugin-dir ~/ai-game-builder
```

> **Alternative**: If you installed via marketplace (`/plugin install godot-ai-builder`), just run `claude` — the plugin loads automatically.

Claude Code starts with full awareness of: - The Godot project structure - All 14 game generation skills - The MCP tools to control the Godot editor - The Stop hook that keeps it focused during builds

---

Step 6: Tell Claude to Build Your Game

Type a game prompt. Here are some examples:

Simple (good for first try)

```
Create a top-down shooter where I move with WASD, aim with mouse,
and shoot enemies that chase me. Keep score.
```

Medium complexity

```
Build a 2D platformer with a player that can jump and double-jump,
coins to collect, spikes that kill you, and 3 levels of increasing
difficulty. Include a main menu and game over screen.
```

Full game request

```
Create a complete tower defense game with:
- A winding path that enemies follow
- 3 tower types (basic, splash, slow)
- Wave system with 10 waves of increasing difficulty
- Gold economy (earn from kills, spend on towers)
- Lives system (enemies reaching the end cost lives)
- Full UI: main menu, HUD with gold/lives/wave, game over
- Polish: particles, screen shake, smooth animations
```

What Happens Next

Claude follows the **Game Director** protocol — a 6-phase build process:

Phase 0: PRD (Product Requirements Document)
Claude writes a detailed game design doc and asks for your approval before writing any code.

Phase 1: Foundation
Creates the project structure, player movement, camera, and background. Runs the game to verify it works.

Phase 2: Player Abilities
Adds shooting, jumping, or whatever the player's actions are. Tests everything.

Phase 3: Enemies & Challenges
Creates enemy types, spawn systems, collision, health, scoring, and difficulty ramping.

Phase 4: UI & Game Flow
Builds the main menu, HUD, game over screen, pause menu, and screen transitions. Wires up the full game loop: menu → play → die → retry.

Phase 5: Polish
Adds game "juice" — screen shake, hit flash, death particles, spawn animations, floating score numbers, smooth camera, and visual effects.

Phase 6: Final QA
Runs the game multiple times, checks for errors, edge cases, and memory leaks. Reports the final status.

The Stop hook keeps Claude locked in until all 6 phases are complete. You don't need to worry about it quitting halfway through.

During the Build

While Claude is working, you'll see activity in two places:

In the terminal (Claude Code): - Claude reports progress after each phase - Shows quality gate results (e.g., "Phase 3 complete. 6/6 gates passed.") - Asks for your input if needed (e.g., PRD approval)

In the Godot editor (AI Game Builder dock): - Shows bridge activity (reload, run, error check commands) - The game runs automatically in the editor when Claude tests it - You can see the game come to life in real-time

You can interact during the build: - Approve or modify the PRD when Claude presents it - Ask for changes ("make the enemies faster", "change the color palette") - Claude will adjust and continue

---

After the Build

When all 6 phases complete: 1. Claude reports what was built (file list, features, quality gates) 2. The Stop hook disengages (Claude can finish normally) 3. Your game is ready to play in the Godot editor 4. Press **F5** in Godot (or the play button) to run it anytime

Your project folder now contains:

```
my-first-game/
├── project.godot         # Game settings + input mappings
├── scripts/              # All game logic
│   ├── main.gd           # Game loop, spawning, score
│   ├── player.gd         # Player controller
│   ├── enemy.gd          # Enemy AI
│   ├── bullet.gd         # Projectile logic
│   └── ui/               # HUD, menus, game over
├── scenes/               # Scene files
├── assets/               # Generated sprites
├── docs/PRD.md           # The game design document
├── addons/               # Godot editor bridge plugin
└── knowledge/            # Reference docs
```

Troubleshooting

"Godot bridge not connected"

- Make sure the Godot editor is open with your project
- Check that the AI Game Builder plugin is enabled (Project → Project Settings → Plugins)
- The plugin runs an HTTP server on port 6100 — make sure nothing else uses that port

"No project.godot found"

- Run setup.sh with the path to a valid Godot project (the folder containing `project.godot`)
- If you haven't created a project yet, do Step 2 first

Claude stops unexpectedly

- If Claude hits a rate limit, wait a moment and type "continue" — it will resume
- If the build lock is stuck, manually remove it: `rm .claude/.build_in_progress`
- Re-run Claude with `claude` and tell it to continue the build

Game has errors after build

- Open Claude Code and say: "Fix the errors in my game"
- Claude will call `godot_get_errors`, read the problematic files, and fix them

Want to start over

- Delete the generated files: `rm -rf scripts/ scenes/ assets/ docs/`
- Tell Claude: "Create a new game from scratch"

Quick Reference

Action	How
Build a complete game	"Create a [genre] game with [features]"
Add a feature	"Add enemies that shoot back"
Fix errors	"Fix the errors" or "Check for errors and fix them"
Polish the game	"Add screen shake, particles, and juice"
Change something	"Make the player faster" / "Change the background to dark blue"
Cancel mid-build	`rm .claude/.build_in_progress` then Claude can stop
Run the game	Press F5 in Godot, or Claude runs it automatically
See game errors	Claude checks automatically, or say "Get errors"

Built with AI Game Builder — Claude Code + Godot 4