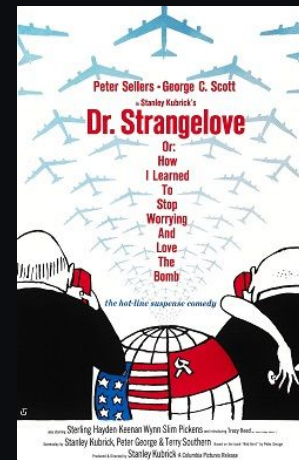


# Dr. Claudelove,

or: How I Learned to Stop Worrying  
and Love the Git Worktree

---

*“Agents, you can't code in here,  
this is the main worktree!”*



## Disclaimer

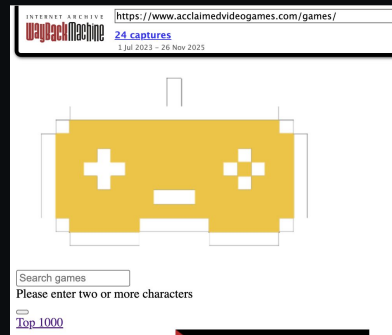
- I am not a Git expert.
- I am not the most sophisticated Git user or an expert in worktrees and how they work / what you can do with them. (Apologies if the title is false advertising.)
- I simply heard they were useful for parallelizing agentic development and asked Claude to spin them up and rebase when needed.

**The following is a “YOLO” experiment in pushing the limits of Claude Code for a personal project. And a deliberate challenge to MAXIMIZE, not minimize, token usage at Anthropic’s promotional rate before New Year’s.**

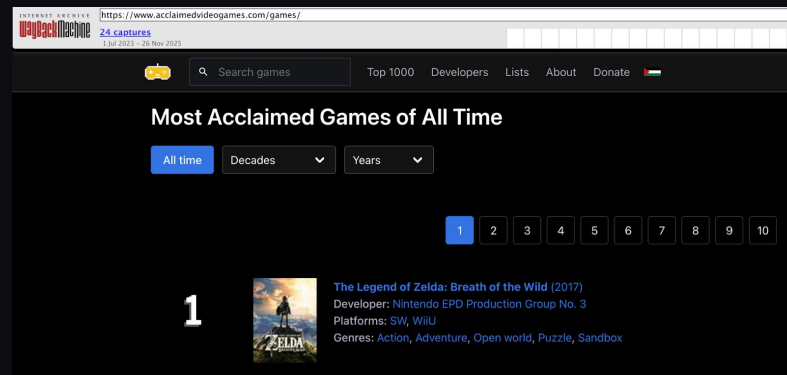
# Project Context

## Personal Project


- [acclaimedvideogames.com](https://www.acclaimedvideogames.com) is a site that aggregates >800 critic lists of the greatest video games. (This has better data for older retro games than something score-based like Metacritic.)
- In the past month and a half it has undergone several major refactors:
  - Migration from heavier frontend single-page app (SPA) on Vue.js to Django + HTMX + Alpine.js for Wayback Machine archivability and simplicity.
  - Completely redoing the layout of pretty much every page and flexibility of filtering the results. (Including custom year heatmap component etc.)
  - Expanding and generalizing the backend to support book lists in the future.



No archive :(





Archive! :)



- Rankings
- Developers
- Source Lists

- Login
- About / FAQ
- Contact Us
















## Most Acclaimed Video Games of All Time

Showing 100 of 1,000

Rank ↑
Game # ↓

1		<b>The Legend of Zelda: Breath of the Wild</b> (2017) <small>Nintendo EPD Production Group No. 3   WiiU, SW   Action-Adventure   ~50h   105 lists</small>
2		<b>Tetris</b> (1985) <small>Alexey Pajitnov   MS-DOS, NES, GB   Puzzle   ~3h   94 lists</small>
3		<b>The Last of Us</b> (2013) <small>Naughty Dog   PS3, PS4   Action-Adventure   ~15h   115 lists</small>
4		<b>The Legend of Zelda: Ocarina of Time</b> (1998) <small>Nintendo EAD   N64   Action-Adventure   ~27h   82 lists</small>
5		<b>Half-Life 2</b> (2004) <small>Valve   WIN, Xbox, X360, PS3   First-Person Shooter   ~13h   85 lists</small>
6		<b>The Witcher III: Wild Hunt</b> (2015) <small>CD Projekt RED   WIN, XB1, PS4   Action RPG   ~52h   108 lists</small>
7		<b>Super Mario 64</b> (1996) <small>Nintendo EAD   N64   Platform   ~12h   83 lists</small>
8		<b>The Legend of Zelda: A Link to the Past</b> (1991) <small>Nintendo EAD   SNES   Action-Adventure   ~15h   77 lists</small>
9		<b>Resident Evil 4</b> (2005) <small>Capcom Production Studio 4   WIN, PS2, GC, Wii   Horror   ~16h   87 lists</small>
10		<b>Super Mario World</b> (1990) <small>Nintendo EAD   SNES   Platform   ~5h   73 lists</small>
11		<b>Shadow of the Colossus</b> (2005) <small>Team Ico   PS2   Action-Adventure   ~10h   72 lists</small>

**Filters** Saved CSV Clear All

☆ **Game Status**

All

☆ Untracked ☆+ Want ☆ Played

🔍 **Search**

📅 **Release Year**

Drag or shift+click for ranges 1 42

2020s	20	21	22	23	24					
2010s	10	11	12	13	14	15	16	17	18	19
2000s	00	01	02	03	04	05	06	07	08	09
1990s	90	91	92	93	94	95	96	97	98	99
1980s	80	81	82	83	84	85	86	87	88	89
1970s	70	71	72	73	74	75	76	77	78	79

📁 **Platform**

- > PC 562
- > PlayStation 429
- > Nintendo 313
- > Xbox 304
- > Arcade, Mobile, & VR 135

Final state :O

(Books changes are backend-only right now)

# Launching the Plane

## Takeoff

- The Script: Wrote a shell script to start eight Git worktrees simultaneously.
- The Command: Integrated as a VS Code workspace task to spin up eight parallel terminals of Claude Code in YOLO mode.
- The Goal: Deliberately doing “the crazy thing” to test massive agent parallelism.

Select the task to run

Launch Swarm CONTROL (Main)

recently used   

configured

W1 CONTROL (Main)

W2 CONTROL (Main)

W3 CONTROL (Main)

W4 CONTROL (Main)

W5 CONTROL (Main)

W6 CONTROL (Main)

W7 CONTROL (Main)

W8 CONTROL (Main)


```
$ start-workers.sh
#!/bin/bash
# Opens VS Code workspace and starts Claude workers in git worktrees

SCRIPT_DIR="$(cd "$(dirname "$0")" && pwd)"
PARENT_DIR="$(dirname "$SCRIPT_DIR")"

# Open VS Code workspace
code "$PARENT_DIR/acclaimedgames-workers.code-workspace"

# Start each worker in its own Terminal with Claude
for i in {1..8}; do
  WORKER_DIR="$PARENT_DIR/acclaimedgames-worker-$i"
  if [ -d "$WORKER_DIR" ]; then
    osascript -e "
      tell application \"Terminal\"
        do script \"cd '$WORKER_DIR' && echo 'Worker $i ready.' && claude --dangerously-skip-permissions\"
      end tell
    "
  else
    echo "Warning: $WORKER_DIR does not exist"
  fi
done

echo ""
echo "Workers started."
```

claude --dangerously-skip-permissions 

# The War Room: This Deserves Its Own Slide

The screenshot displays a multi-pane IDE interface titled "swarm (Workspace)". The interface is divided into several panes:

- Left Pane (SOURCE CONTROL):** Shows a list of worktrees (W1, W2, W3, W4) and a "CHANGES" section with a list of commit messages and actions like "Commit" and "Publish Branch".
- Top Row (W1, W2, W3, W4):** Each pane shows a terminal window for a container. W1 and W2 show error messages related to "SessionStart" and "up hook". W3 and W4 show successful "up hook" execution.
- Bottom Row (W5, W6, W7, W8):** Each pane shows a terminal window for a container. W5 and W6 show error messages. W7 and W8 show successful "up hook" execution.
- Bottom Left Pane (GRAPH):** Shows a list of tasks and their status, including "Fix Alpine.js race condition on page load" and "Add django-extensions for local HTTPS development".

The terminal windows display the following content:

```
W1 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W1
  SessionStart
  up hook
  error
  > Try "create a util logging.py that..."
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W1 [w1
  (worktree: W1)]
  ** bypass permissions on (shift+tab to cycle)

W2 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W2
  SessionSt
  artstart
  up hook
  error
  > Try "fix lint errors"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W2 [w2
  (worktree: W2)]
  start-workers.sh
  ** bypass permissions on (shift+tab to cycle)

W3 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W3
  SessionSt
  artstart
  up hook
  error
  > Try "refactor models.py"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W3 [w3
  (worktree: W3)]
  ** bypass permissions on (shift+tab to cycle)

W4 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W4
  SessionSt
  artstart
  up hook
  error
  > Try "create a util logging.py that..."
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W4 [w4
  (worktree: W4)]
  ** bypass permissions on (shift+tab to cycle)

W5 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W5
  SessionSt
  artstart
  up hook
  error
  > Try "fix lint errors"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W5 [w5
  (worktree: W5)]
  ** bypass permissions on (shift+tab to cycle)

W6 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W6
  SessionSt
  artstart
  up hook
  error
  > Try "how does settings.py work?"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W6 [w6
  (worktree: W6)]
  ** bypass permissions on (shift+tab to cycle)

W7 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W7
  SessionSt
  artstart
  up hook
  error
  > Try "write a test for urls.py"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W7 [w7
  (worktree: W7)]
  ** bypass permissions on (shift+tab to cycle)

W8 (CONTROL (Main)) Task X
* Claude Code v2.8.76
  Opus 4.5 - Claude Max
  ~/avg-swarm/W8
  SessionSt
  artstart
  up hook
  error
  > Try "fix lint errors"
  rchibana@MacBook-Pro-S1:/Users/rchibana/avg-swarm/W8 [w8
  (worktree: W8)]
  ** bypass permissions on (shift+tab to cycle)
```

# Managing the Madness

## Operating realities

- Visual Monitoring: eight terminals looks insane, but monitoring Git state is essential.
- Coordination Failure: tried using “[beads](#)” for issue tracking; failed due to setup + manual tree management.
- The Pivot: fell back to a single Markdown PRD.
- Agent Interaction: agents claim tasks and mark “done” directly in the PRD.

### ### Phase 1: Core Infrastructure

#### #### 1.1 Create Core App Structure

Task ID	Task	Files	Complexity	Risk	Dependencies
1.1.1	Create core/ app skeleton	`core/__init__.py`, `core/apps.py`	S	Low	None
1.1.2	Add to INSTALLED_APPS	`acclaimedgames/settings.py`	S	Low	1.1.1
1.1.3	Create abstract MediaItemBase	`core/models.py`	M	Low	1.1.1
1.1.4	Create abstract CreatorBase	`core/models.py`	S	Low	1.1.1
1.1.5	Create abstract ExternalDataBase	`core/models.py`	S	Low	1.1.1
1.1.6	Create abstract UserTrackingBase	`core/models.py`	S	Low	1.1.1

**\*\*Parallelization\*\*:** 1.1.3–1.1.6 can be done in parallel after 1.1.1–1.1.2

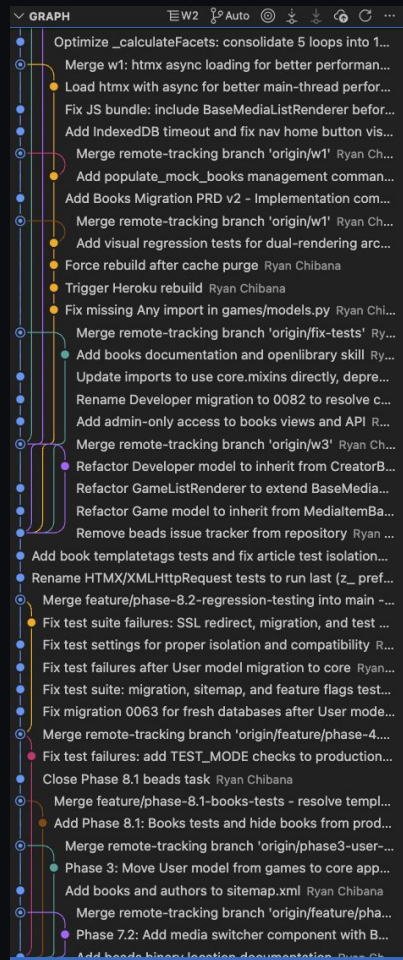
#### #### 1.2 Move Shared Utilities

Task ID	Task	Files	Complexity	Risk	Dependencies
1.2.1	Move RobustPagingMixin to core	`core/mixins.py`, `games/mixins.py`	S	Low	1.1.2
1.2.2	Move HTMXPartialMixin to core	`core/mixins.py`, `games/mixins.py`	S	Low	1.1.2
1.2.3	Add backward-compat re-exports in games	`games/mixins.py`	S	Low	1.1.2

# Mutually Assured Destruction: Drift and Deployment

## What it feels like

- The Drift: eight agents inevitably drift away from each other.
- Rebasing: you must tell agents to rebase regularly (hard at swarm scale).
- The Middle State: you hit moments where you don't understand anything that's happening and briefly consider starting over and throwing out tens of thousands of lines.
- Pushing Through: with intuition + delegation, you emerge into a stable, "very good" state.
- Every now and then, dedicating an agent to reviewing all the code produced and the PRD to determine incompatibilities / gaps helps a lot. I generated a v2 PRD based on this that simplified things immensely.

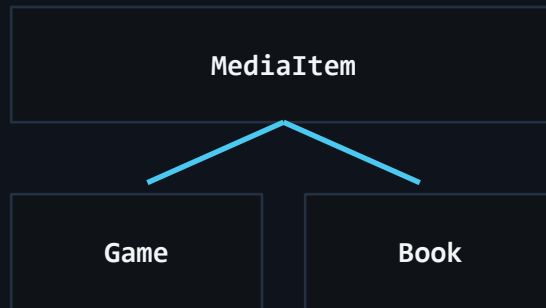


# From Games to Books

## The refactor

- Objective: generalize the data model of acclaimedvideogames.com to support book lists.
- Games now inherit from a general MediaItem class.
- Abstracted templates + pages into a core app with derived versions specific to the games and books apps.
- Scale: involved rewriting tens of thousands of lines of code.
- AI autonomy: agents designed aspects of the new model without it being in the original PRD.
- Outcome: the books refactor was completed and correct **within a couple of hours**.

Data model (mock)



Agents inferred what components could be broken out and which were specific to each app.

## Is 8× Faster Real?

≈4×

...faster than a single  
agent session (in practice)

### What actually happens

- Truth: it isn't eight times faster than a single agent session — there is wasted work.
- Reality: it feels closer to ~4× for large refactors.
- Strategy: clear early and clear often; start new sessions frequently.
- Surprise: despite chaos, the final structure eventually landed on an organized and logically correct state.

## The Post-Atomic Age

### Can traditional 400-line PR reviews survive code generation at this scale?

#### Implications

- Reality: these massive refactors involved zero traditional code review.
- New approach: shift review up a level — validate PRD / intent, then verify outcomes.
- Challenge: we don't have the final answers yet, but the speed forces a rethink.
- Risk vs. speed: correctness wasn't essential for a hobby project — adjusting this approach to corporate / Compliance levels of risk is the hard part.

# Riding the Bomb: From Coder to Fleet Manager

## What changed in my head

- I now see myself as a manager of a fleet of agents, learning to coordinate them effectively.
- The power of iterating across eight agents simultaneously is hard to grasp until you see it. (Okay, eight is too many. Try four just to try it.)
- If things break later, I have an entire team of agents ready to fix them.



# What Now?

## Discussion Questions (Open-Ended)

- What's the best "unit of work" for agents: PRD tasks, files, modules, or end-to-end features?
- If line-by-line review doesn't scale, what should review focus on: intent/PRD, architecture diffs, tests?
- Have you come across any good methods for coordinating swarms of coding agents?
- What are the must-have guardrails to ship safely: test strategy, observability, rollout plan, and automated checks?