Overview

This is a small Phaser 3 puzzle game where players drag numbered tiles from a queue on the right side onto a 4×4 grid. They can divide or match numbers to clear or combine tiles. This repository includes a refined version that improves UI spacing, z-ordering (fix for the drag jump bug), previews for the two-number queue, better typography, a timer, hints, an undo stack, and handling for responsive containers.

Approach

Iterative polish: Begin with the playable prototype and focus on the biggest issues first, such as the jumping tile, alignment, and z-ordering. After that, enhance the visuals, like fonts and timer placement, and improve the user experience with animations and hints.

Separation of concerns: Keep game logic, including grid state, merging and dividing rules, scores, and the undo feature, distinct from rendering code. Use small helper methods for creating tiles, rendering the queue, and placing logic.

Safe drag-layer pattern: Display the draggable top queue tile as a world object on a dedicated dragLayer instead of inside the queue container. This prevents issues with re-parenting and coordinate conversion that caused the tile to jump to the top left.

Progressive refinement: After fixing the main issues, add finishing touches like landing animations, floating feedback texts, highlighted hints, and a two-tile queue layout to match the requested design.

Decisions Made

Top queue tile placed on dragLayer: This prevents snapping or jumping and ensures dragged tiles are always displayed above the UI.

Two-number queue: This shows the top draggable number and a preview below it to enhance gameplay clarity.

Timer centered under the title: This matches the requested visual layout and improves visibility.

Fonts: A playful display font (Luckiest Guy) is used for the title and badges, with fallback fonts provided for readability if the webfont is unavailable.

Fixed-size canvas: The canvas remains at 1440×1024 for a consistent layout. Phaser's Scale.FIT is applied to fit the canvas into the container without CSS transforms that previously caused zoom or scale issues.

Layered drawing order: The order is title, timer, cat, subtitle, badges, grid, right panel, and drag layer. This explicit ordering prevents elements from overlapping incorrectly.

Undo stack: A snapshot approach, using deep copies of arrays and primitive fields, is used for a simple undo feature, limited by CONFIG.MAX_UNDO.

Challenges

Drag snapping and reparenting issues: Changing a display object from a container to the main scene, or vice-versa, can lead to incorrect world or local coordinates, causing the top tile to appear at the top left when dragging starts. The solution is to create the top tile directly in world coordinates on a top layer.

Layer and z-order consistency: Ensuring the cat image is visually behind the badges but in front of the background required careful draw ordering. The subtitle must also remain above the cat for readability.

Responsive vs fixed layout: The game art and layout are designed for a fixed 1440×1024 canvas. Making it responsive without losing spacing and clean visuals is challenging; using Phaser's scaling with a fixed base size strikes a good balance.

Asset placement and naming: Visual assets need to match their filenames exactly, as they are case sensitive. During changes, mismatched names can lead to blank or misaligned elements.

Touch and mouse parity: Ensuring consistent drag functionality across different pointer types involved testing and simplifying the drag logic to make it portable.

What I Would Improve (Next Steps)

Responsive layout modes

Provide multiple layout presets for desktop, tablet, and phone, or scalable UI anchors, so the experience is uniform across devices without losing art proportions.

Tile pooling

Implement object pooling for tile sprites to reduce garbage collection churn and enhance performance on lower-end devices.

Better animations and audio

Add subtle sound effects for the UI and richer animations for placements and clears to increase feedback and satisfaction.

Accessibility

Introduce keyboard controls and ARIA-compliant labels for non-mouse users, along with visual accessibility options like color-blind mode.

Configurable difficulty and goals

Add level objectives, progressively increasing difficulty, and an in-game store for cosmetics and trash items.

Unit tests for core logic

Separate grid, merging, and dividing logic into a small pure JavaScript module and include unit tests for the accuracy of the merging and dividing rules.

Asset optimization

Combine sprite sheets, compress PNGs and WebP files, and create a manifest to allow for faster loading and a smaller bundle size.

Persistent progress

Enhance localStorage use to save the full game state, track level progress, and optionally sync to the cloud.

Polish UI

Replace the emoji-based timer with a small SVG or hourglass sprite, and add a subtle shadow or blur behind the badges for improved contrast.

How to Run Locally

Ensure the assets folder is present next to index.html and contains:

Desktop_JustDivide_Game_2.png, Cat.png, Levels and Score.png, Placement_Box.png, trash.png, blue.png, orange.png, pink.png, purple.png.

Use a local server (recommended):

VS Code Live Server or:

python -m http.server 8000, then open http://localhost:8000/.

Open index.html in a modern browser like Chrome, Edge, or Firefox.

Make sure the browser zoom is set to 100% for the designed 1440×1024 layout.

Controls:

Drag the top queue tile to the grid, keep it in a slot, or trash it.

Press R to restart, Z to undo, G to toggle hints, and 1/2/3 to change difficulty.

Code Notes and Important Gotchas

The top draggable tile is created each time the renderQueue() function is called as a world object on this.dragLayer. If you change the container transforms or parenting, follow the same approach to avoid coordinate problems.

The undo snapshot is implemented as a shallow copy for arrays and primitive fields; it works for the current state shape. If nested objects are added to the state, the snapshot strategy will need to be updated.

Fonts: The code uses Google Fonts for Luckiest Guy. If offline, the fallback is Arial; adjust as needed.

Ensure asset filenames are exact, including case sensitivity on case-sensitive file systems.