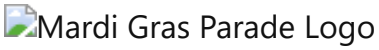
Game Design Document (GDD)



Date: 2025-12-09

Project: Mardi Gras Parade Simulator — Repository

Vision

An accessible, browser-based 3D Mardi Gras parade simulator where players catch throws from parade floats while competing with AI bots. The game aims to be family-friendly, playable on desktop and mobile, and emphasize fun mechanics with light strategy via color-matching and power-ups.

▶ Test now — Open in browser

Quick start (dev)

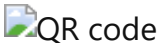
1. Install dependencies:

```
npm install
```

2. Start the dev server (Vite + Express dev integration):

```
npm run dev
```

3. Open the game in your desktop browser: http://localhost:5000
4. To test from a mobile device on the same network: scan the QR code below (or regenerate with `npm run qr`):



Core mechanics

- Movement: WASD / arrow keys, click-to-move, and touch controls for mobile.
- Catching: Move near a collectible to catch it. Trajectory physics determine where throws land.
- Color match: Each player gets an assigned color. Catching matching-color items yields 3x points.
- Combos: Consecutive catches within a 3-second window increase combo multiplier.
- Power-ups: Speed boost (temporary movement increase) and Double Points (score multiplier).
- King Cake: Rare special collectible worth 5 points.

Scoring

- Base item = 1 point.
- Color match = 3x points for that catch.
- Double Points power-up multiplies current catch points by 2.
- King Cake = flat 5 points.
- Coins awarded per catch for cosmetics economy.

Levels & difficulty

- Each level: a set number of parade floats pass by (10 floats per level base).
- Early levels (1-3): gentle learning curve. After level 3 difficulty ramps up: faster throws, more bots, more obstacles.
- Obstacles break combos if hit.

AI Competitors (Personas)

- King Rex — aggressive: fast, prioritizes nearest throws, high risk.
- Queen Zulu — focused: efficient pathing, targets highest-priority items.
- Saint Mardi — playful: wanders often, sometimes prioritizes low items.
- Jester Jo — sneaky: nimble, quick direction changes.
- Voodoo Vee — lucky: biases toward low-y items and enjoys rare catches.
- Mambo Mike — steady: balanced behavior for testing and stability.

Balancing notes

- Start target score low (e.g., 5) and increase per level.
- Throw frequency starts slow (3500ms at level 1) and decreases over levels.
- Adjust bot speed and number to tune challenge.

Assets & budgets

- Floats & characters: GLB (gltf binary), aim <10k tri per float; characters <2k tri.
- Collectibles: simple low-poly models (<1k tri).
- Textures: WebP preferred; max 2048×2048; use compressed formats for production.
- Audio: SFX short (0.1–1s), background loop ~60–120s compressed for web.

UI/UX & tutorial

- First-level tutorial overlay with concise instructions.
- Enable Audio button required to unlock audio on browsers.
- HUD: Score, Level, Power-ups, Bot catches, Persona toggle in debug.

Telemetry suggestions

- Capture catches per item type, combos, level completion time, player deaths (float collisions), and bot catches for balancing.

Additional ideas

- Cosmetic shop: purchase skins with coins.
- Instancing for crowd/crowd props to reduce draw calls.
- Accessibility: large UI, reduce motion, colorblind support.

Development checklist for content

- [ ] Finalize float and character models
- [ ] Create low/med/high LOD versions
- [ ] Prepare textures and MIP maps
- [ ] Create audio SFX and background music files

- [ ] QA asset integration in dev server