"Create a top-down multiplayer racing betting game where players cannot control the race but instead bet on AI-controlled races ,which is using this github : https://github.com/MonAmiral/CustomHorseRaceTest . a retro pixel art style using 32x32 or 64x64 resolution sprites. a custom foreground by putting a PNG file named "Foreground" in there, with dimensions of 4000x2000. "

Game play :

Players create custom characters that compete in fully automated races [implement random "blind horse" movement instead of pathfinding. ] while spectators place bets and earn leaderboard rankings based on both win rate and total score accumulation. All player started at one point in the each map. The target is to obtain the M token in the map. The time is 5 mins per game. If no one get the M token, they all loss. M token will be another direction of the starting point.

Referece : https://callmemonamiral.itch.io/custom-horse-race-tests

**Visual Style & Resolution:**

* Use extremely low-resolution pixel art graphics with 32x32 or 64x64 sprite dimensions
* Create a retro 8-bit or 16-bit aesthetic with limited color palettes
* Design simple, blocky car sprites that are easily distinguishable at low resolution
* Make all UI elements and text large enough to be readable at the pixelated resolution
* Use bright, contrasting colors to ensure visibility during fast-paced gameplay

**Multiplayer Racing Mechanics:**

* Support 6 players racing simultaneously on the same track

**Random Speed Booster System:**

* Spawn random speed boost power-ups on the map and spawn in random interval.
* Create different types of boosters: anatena booster, Twitter post , Memex Tag , Poo , Toilet paper, Toilet , Banana , King Kong .
* Make boosters appear as glowing pixel icons that players can drive over to collect
* Each booster should provide a 4-7 second speed enhancement
* Add visual effects when boosters are activated (screen shake, particle effects, speed lines)

**Player Skill System:**

* 5 random skills that drop from air in the game.
* Skills :
  + Thunder : temporarily paralyze 3 random opponents for 3 secs
  + Fire : will burn the other 2 charterers and slow down .
  + **Bubble Protection :** The player can bounces off other players if encounter in 8 seconds
  + **Magnetized Shoes**  
    For 5 seconds, the player will attract to nearby racers and sticker together . if found the M coin during the stage, the play with magnetized shoes win.
  + **Random Teleport Twitch**  
    all port players teleport randomly to a place in the map at once.
* The paralysis effect should prevent other players from accelerating or steering for 2-3 seconds
* Make this ability have a cooldown period of 15-20 seconds to prevent spam
* Add a charging system where players build up paralysis energy by driving well or collecting specific items
* Show a clear visual indicator when a player is paralyzed (flashing sprite, different color)
* Include a brief immunity period after being paralyzed to prevent chain-stunning

**Betting & Leaderboard System:**

* Implement a dual-factor leaderboard that weighs both **win rate percentage** and **total accumulated points**
* Use a weighted ranking algorithm where players need both consistency (high win rate) and volume (high total score) to reach the top
* Each individual race is completely separate with independent betting pools
* Players earn points based on correct predictions and betting accuracy
* Include betting confidence multipliers: higher confidence bets yield more points but cost more to place
* Add seasonal leaderboard resets to prevent score inflation over time

**Ultra-Randomization Framework:**

**Character Performance Randomization:**

* Use **Gaussian randomness** instead of uniform randomness for character stats to create natural performance variations
* Implement **biorhythm systems** where each character's performance fluctuates using sine waves with different periods (30-120 second cycles)
* Add **chaos theory elements** where tiny initial condition differences create vastly different race outcomes

**Multi-Layer Random Event System:**

* **Environmental Chaos Events** (every 15-45 seconds):
  + Sudden weather changes affecting all racers differently
  + Track surface modifications (ice patches, oil spills, boost strips)
  + Gravitational anomalies that alter physics for 5-10 seconds
  + Visual interference events (fog, blinding flashes, inverted controls)
* **Individual Character Events**:
  + Random skill activation with unpredictable timing and effectiveness
  + Equipment malfunction or super-performance bursts
  + Temporary character personality shifts affecting driving style
  + Random paralysis effects targeting random players for random durations

**Procedural Track Generation:**

* Use **L-Systems** and **cellular automata** to generate completely unique track layouts for each race
* Implement **distributed generation** using multiple noise functions to create natural-looking but unpredictable terrain
* Add **fractal geometry** to create self-similar track sections with chaotic properties
* Include **Poisson disk sampling** for optimal placement of track hazards and boost zones

**Multi-Source Randomization:**

* Combine **multiple pseudorandom generators** with different algorithms for different game systems
* Use **filtered randomness** to ensure fairness while maintaining unpredictability
* Implement **variable rate random numbers** that adjust probabilities based on previous outcomes
* Add **human-generated randomness** by incorporating player betting patterns as entropy sources

**Dynamic Probability Systems:**

* Create **adaptive random events** where probabilities shift based on race progression
* Use **weighted randomization** with dynamic weight adjustments throughout each race
* Implement **anti-pattern systems** that detect and counter any emerging predictable sequences

**Recommended Tech Stack Combination**

**For Maximum Simplicity:**

1. **Phaser.js** for game engine
2. **Mersenne Twister** for professional randomness
3. **Socket.io + Express** for multiplayer (~200 lines total)
4. **Simple JSON/LocalStorage** for data persistence Real-time game state broadcasting . Perfect for betting updates

**Easy Leaderboard Systems**

**Simple Implementation Approach**

Research shows the most effective leaderboard structure:

* **Small groups** (100 players max) for meaningful rankings
* **Dual-factor scoring**: Win rate + total points (as you specified)
* **Real-time updates** through WebSocket connections