

NEON STRIKERS

Wish-list, Future Work & Testing Plans

Post-Showcase Bug Log Review, Proposed Testing Events,
and Feature Wishlist for Continued Development

Authored by: Louis Flores
Technical Lead & Documentation Neon Strikers
Florida International University Spring 2026

1. Introduction

This document covers three interconnected areas: the bug backlog identified at and after the capstone showcase, proposed structured testing plans to address those bugs in a rigorous and engaging way, and a wishlist of features and improvements for future development iterations of Neon Strikers.

The showcase was a successful event that demonstrated a playable, functional game to a real audience. It also revealed, as real-user testing always does, a set of bugs and usability issues that internal testing had not surfaced. This is expected and valuable. This document is the team's commitment to not leaving those findings unaddressed.

2. Post-Showcase Bug Analysis

2.1 How Bugs Were Found

The capstone showcase involved a significant number of participants playing the game in an unstructured way — pressing buttons the development team would not have pressed, combining inputs in unexpected sequences, and interacting with the game at a pace and variety that no internal testing session had matched.

During the showcase, team members observed gameplay, noted anomalies, and spoke with participants after their sessions. Following the showcase, additional bugs were reported asynchronously. All identified issues have been logged on the project GitHub repository as Issues. This section summarizes and categorizes the known bugs as of this document's authoring.

2.2 Known Bug Inventory

The following bugs are tracked on the GitHub Issues tab and constitute the current backlog:

Bug ID	Category	Description	Severity	Status
BUG-01	Physics	Player clips through thin platforms at high fall velocities	High	Open
BUG-02	Combat	Hitbox remains active for one extra frame after attack animation ends, causing phantom hits	Medium	Open
BUG-03	UI	Health bar does not animate smoothly on rapid successive hits; jumps instead of interpolates	Low	Open
BUG-04	Game Logic	Simultaneous death of both players produces undefined round result; game hangs	Critical	Open
BUG-05	Controls	Wall jump direction inverted when jump input is pressed on the exact frame of wall contact	Medium	Open
BUG-06	State Machine	Game over screen occasionally fails to load on first trigger; retrying succeeds	High	Open
BUG-07	Audio	Audio continues briefly after scene transitions due to missing AudioSource cleanup	Low	Open
BUG-08	UI	Control mapping not displayed anywhere in game; first-time players have no in-game reference	Medium	Open
BUG-09	Physics	Landing on platform edge at certain angles causes momentary stutter in horizontal movement	Low	Open
BUG-10	Combat	Heavy attack animation plays at slightly	Low	Open

		wrong offset when facing left vs. right		
--	--	---	--	--

2.3 Severity Definitions

Severity	Definition
Critical	Causes a crash, hang, or game-breaking undefined state that prevents play from continuing
High	Significantly degrades player experience or causes incorrect game outcomes
Medium	Noticeable and annoying but does not block gameplay or corrupt game state
Low	Minor visual, audio, or polish issue with minimal functional impact

2.4 Priority Fix Order

Based on severity and estimated implementation complexity, the recommended fix order for the next development sprint is:

1. BUG-04 (Critical) — Define a clear tiebreaker behavior for simultaneous deaths. Simplest fix: last player to enter the death state loses. Add a guard in `GameManager.OnPlayerDeath()` to handle the case where `RoundEnd` has already been triggered.
2. BUG-06 (High) — Add a guard in the `GameManager` state machine transition to prevent double-triggering of scene loads. Likely cause is an event being fired twice from `HealthManager`.
3. BUG-01 (High) — Implement continuous collision detection on the player `Rigidbody2D`. Set `CollisionDetectionMode2D.Continuous`. Test with all platform thicknesses in the current level.
4. BUG-05 (Medium) — Add a one-frame cooldown to wall jump direction resolution to eliminate the edge case.
5. BUG-02 (Medium) — Add an explicit hitbox disable call at the start of the hitbox-active window's final frame rather than relying on animation event timing alone.
6. BUG-08 (Medium) — Add a semi-transparent control overlay to the main HUD. This should be accessible via a pause menu toggle and visible briefly at match start.
7. Remaining low severity bugs — defer to a dedicated polish sprint.

3. Proposed Live Testing Event

3.1 Concept

The single most effective testing activity for Neon Strikers was the showcase itself. Structured player testing with diverse participants surfaced more actionable bugs in two hours than weeks of internal testing had produced. The Live Testing Event is a proposal to replicate and formalize this approach as a deliberate testing methodology rather than an incidental byproduct of an academic event.

The format draws on the game development industry practice of playtest sessions but combines it with an informal, social atmosphere designed to maximize both participant engagement and data collection. The event is structured around a mini tournament format with pizza, which creates natural incentives for participants to play multiple sessions, escalate their engagement, and push the game harder than they might in a passive testing session.

3.2 Event Format

3.2.1 Overview

Attribute	Detail
Event Name	Neon Strikers Dev Testing Night
Target Attendees	12-20 developers, CS students, and engaged gamers
Duration	3-4 hours
Venue	FIU computer lab or similar space with multiple stations
Equipment	4-6 paired controllers per station, development builds loaded on each machine
Format	Open play + structured tournament + live bug fixing sessions
Incentives	Pizza, informal prizes for tournament winner, recognition in release notes

3.2.2 Session Structure

The event is broken into four phases:

Phase	Duration	Activity	Testing Goal
Phase 1: Orientation	20 min	Welcome, brief game overview, control explanation. Participants are NOT given detailed mechanics info — this is intentional to test discoverability.	Identify onboarding pain points and control confusion
Phase 2: Open Play	45 min	Unstructured free play at all stations. Team members observe and take notes. No intervention unless participant is	Surface edge-case bugs and unexpected behaviors

		fully stuck.	
Phase 3: Tournament	60-90 min	Single or double elimination bracket. 1v1 matches. Team members continue observing.	Surface competitive-scenario bugs; generate high-intensity play data
Phase 4: Live Fixes	30-45 min	Team reviews notes, deploys hotfix builds addressing the most impactful bugs found during phases 1-3. Participants test the new builds.	Validate fixes in real time; introduce physics tweaks and observe response

3.3 Live Bug Fixing Protocol

Phase 4 is the most distinctive element of the proposed event. Rather than treating bugs as post-event action items, the team will have a developer at a fix station during the event — a laptop separate from the play stations — monitoring the bug log and preparing small targeted fixes in real time.

When a bug is confirmed by multiple independent observations during phases 2 or 3, it becomes a candidate for a Phase 4 fix. The process:

8. Bug confirmed — at least two independent observations or one Critical/High bug observed once.
9. Fix scoped — developer at fix station assesses the change required. If it can be done safely in under 15 minutes, it proceeds. If not, it is logged for post-event sprint work.
10. Fix implemented — change made in a local branch. Build deployed to a dedicated test station.
11. Hotfix version issued — the build is labeled with an incremented version number (e.g., v0.8.1, v0.8.2). Participants at the test station play the hotfix build alongside the previous build for direct comparison.
12. Observation recorded — team notes whether the fix resolved the issue and whether it introduced any regressions.

Importantly, Phase 4 is not just about bug fixes. It is also an opportunity to deploy physics tuning adjustments between rounds of the tournament. Changing values like jump force, movement speed, or attack knockback and observing how participant reactions shift round by round generates direct data on what game feel changes are improvements versus regressions. This turns the event into a rapid iteration loop.

3.4 Physics Tuning Protocol

The following physics parameters are identified as candidates for live tuning at the event:

Parameter	Current Value	Proposed Test Variants	Goal
Jump Force	14.0	12.0, 16.0	Test whether lower jump feels more grounded or worse; test whether higher

			jump opens aerial play
Move Speed	8.0	7.0, 9.0	Test if faster movement feels more exciting or harder to control
Gravity Scale	3.5	3.0, 4.0	Test lighter vs. heavier gravity; affects jump arc feel significantly
Attack Knockback	6.0	4.0, 8.0	Test whether less knockback makes combat feel more tactical; more knockback more chaotic
Hitstun Duration	0.15s	0.10s, 0.20s	Test whether less hitstun feels more action-oriented; more hitstun increases punish window

Each physics variant is assigned a build version number. Participant reactions during tournament rounds using different builds are recorded. At the end of the event, the team reviews which variants generated the most positive participant feedback and uses that as input for the next sprint's physics configuration.

3.5 Observation and Data Collection

During phases 2 and 3, team members circulate through play stations with the following data collection responsibilities:

- Bug Observer — watches for unexpected game behavior, notes the action sequence that triggered it
- Reaction Observer — watches player facial expressions and verbal reactions; notes moments of confusion, frustration, or delight
- Control Observer — specifically watches how participants use the controls, particularly on first contact with the game

A simple paper observation sheet is used at each station. At the end of the event, all sheets are compiled and the bugs are cross-referenced against the GitHub Issues list. New issues not previously tracked are added with the "event-discovered" label.

3.6 Success Criteria

The event is considered successful if:

- At least 5 bugs from the existing backlog are reproduced and confirmed by independent participants
- At least 2 new bugs not previously known are discovered and logged
- At least 1 physics variant generates a measurably more positive participant reaction than the baseline
- At least 1 hotfix build is successfully deployed and validated during Phase 4

- All participants complete at least 2 full matches of the tournament

4. Feature Wishlist

4.1 Overview

The following features represent the team's vision for what Neon Strikers could become with additional development time. They are organized by category and annotated with rough implementation complexity estimates. This wishlist is intended as a starting point for any future team continuing this project, not as a committed roadmap.

4.2 Gameplay Features

4.2.1 Control Overlay and Tutorial

Priority: High. Complexity: Low to Medium.

An in-game control mapping display, accessible from the pause menu and shown briefly at match start, would dramatically reduce onboarding friction for new players. The showcase confirmed this is the single most requested addition. An optional interactive tutorial or training mode where players practice against a stationary dummy would further help new players understand the combat system.

4.2.2 Expanded Attack System

Priority: Medium. Complexity: Medium to High.

The current attack system supports light and heavy attacks with directional variants. A wishlist expansion would add:

- Grab / throw mechanic
- Air-to-ground slam attack
- Dodge / roll with brief invincibility frames
- Special meter that charges with hits taken and spent on a powerful special move

4.2.3 Additional Characters

Priority: Medium. Complexity: High.

Adding a second playable character with a distinct moveset would significantly increase competitive depth and replayability. Even a single alternative character with different speed/weight trade-offs would meaningfully differentiate gameplay experiences.

4.2.4 Multiple Arena Variants

Priority: Medium. Complexity: Medium.

Additional arena layouts with different platform configurations and optional environmental hazards (moving platforms, wind zones, temporary platforms) would increase variety and replayability for repeat players.

4.2.5 Knockback Scaling

Priority: Medium. Complexity: Low.

A common feature in the platformer brawler genre is knockback that scales with accumulated damage — the more damage a player has taken, the farther they fly when hit. This mechanic naturally creates dynamic swings in momentum during matches. Implementing a percentage-based damage accumulation system and scaling knockback against it would add significant strategic depth.

4.3 Technical Improvements

4.3.1 Automated Test Suite

Priority: High. Complexity: Medium.

Implementing a Unity Test Framework automated test suite for the core systems would catch regressions early and reduce manual testing burden. Priority test targets:

- PlayerController movement and jump physics (unit tests)
- CombatSystem hit resolution (unit tests)
- GameManager state machine transitions (integration tests)
- HealthManager edge cases including simultaneous death (unit tests)

4.3.2 Continuous Collision Detection

Priority: High. Complexity: Low.

BUG-01 (platform clipping) is directly addressable by setting `CollisionDetectionMode2D.Continuous` on the player `Rigidbody2D`. This should be treated as a near-term fix rather than a wishlist item.

4.3.3 Data-Driven Physics Configuration

Priority: Medium. Complexity: Low to Medium.

Replace hardcoded physics constants with a `ScriptableObject`-based configuration asset. This would allow physics presets to be swapped at runtime without a code change, making the live tuning event workflow significantly smoother. It would also enable future features like character-specific physics profiles.

4.3.4 Replay System

Priority: Low. Complexity: High.

A lightweight match replay system that records input events and replays them deterministically would be valuable for both debugging and sharing interesting moments. This is a significant engineering investment but would add long-term value for competitive play and bug reproduction.

4.4 Polish and Content

4.4.1 Original Music Score

Priority: Medium. Complexity: External dependency.

The current audio implementation uses placeholder or royalty-free tracks. An original music score composed for the neon aesthetic and pace of the game would substantially elevate the

production value. This requires either external collaboration with a composer or a team member with music production skills.

4.4.2 Particle Effect System Overhaul

Priority: Low. Complexity: Medium.

The current particle effects are functional but minimal. A more comprehensive particle system pass for attacks, hits, jumps, and death events would reinforce the neon visual identity and make the game feel more polished.

4.4.3 Main Menu and Frontend Polish

Priority: Low. Complexity: Low to Medium.

The main menu is functional but sparse. Improvements would include animated menu backgrounds, character preview displays, a settings screen with rebindable controls, and a credits screen.

5. Summary and Priority Matrix

The following matrix summarizes the recommended priorities across bug fixes, testing activities, and wishlist items:

Item	Type	Priority	Estimated Effort	Impact
BUG-04: Simultaneous death	Bug Fix	Critical	Small	Fixes game-breaking hang
BUG-06: Game over screen load	Bug Fix	High	Small	Fixes intermittent crash-adjacent bug
BUG-01: Platform clipping	Bug Fix	High	Small	Fixes high-speed tunneling
BUG-08: Control overlay	Enhancement	High	Medium	Dramatically improves onboarding
Live Testing Event	Process	High	Medium (organizing)	Generates most reliable test data
Automated Test Suite	Technical	High	Medium to Large	Long-term quality infrastructure
BUG-05, BUG-02	Bug Fix	Medium	Small	Gameplay feel improvements
Knockback Scaling	Gameplay	Medium	Small to Medium	Adds competitive depth
Data-Driven Physics Config	Technical	Medium	Small to Medium	Enables live tuning workflow
Expanded Attack System	Gameplay	Medium	Large	Increases player expression
Additional Characters	Content	Medium	Very Large	Major replayability driver
Low severity bugs (BUG-03, 07, 09, 10)	Bug Fix	Low	Small each	Polish
Original Music	Content	Low to Medium	External	Production value
Additional Arenas	Content	Medium	Medium	Variety

6. Closing Notes

This document represents the team's honest assessment of where Neon Strikers stands and what it could become. The bugs are real, the testing event is achievable, and the wishlist is grounded in what the showcase told us players actually want from the game.

The Live Testing Event in particular is a recommendation the team feels strongly about. The single most important thing the project could do with additional time is put the game in front of more real players in a structured environment. The showcase proved the game is fun when it works. More play data will tell us exactly what needs to change to make it work more consistently.

Any future team picking up this project should start by reading the main documentation package, reviewing the GitHub issues list, and attending or organizing a live testing session. The game has a strong foundation. What it needs now is iteration informed by real player behavior — and that starts with getting more people in the room.