DesignFlux

UI/UX Case Study — A creative design workspace for the Godot game-dev pipeline

PRODUCT	DesignFlux — part of the Flux Studio suite by FunLogic Studios
PLATFORM	Desktop application · Windows 10/11 (64-bit)
ROLE	UI/UX Designer & Researcher — Ali Ahmad Khan
TIMELINE	Concept, product site & interface design for Flux Studio v1.0.0
TOOLS	Figma-style component thinking, dark-neon design system, web (Netlify) product site

Overview

DesignFlux is the design-focused half of Flux Studio, a two-app creative suite built for the Godot 4 game development ecosystem. Where its sibling app CodeFlux handles scripting, DesignFlux gives artists and designers a dedicated space to build UI mockups, compose game color palettes, organize icon and sprite frames, and preview typography — with export-ready packaging that plugs straight back into a Godot scene. The goal of this case study is to walk through the UX thinking, problem framing, and interface decisions behind the product.

The Problem

Godot developers who also handle their own art and UI work were bouncing between general-purpose design tools never built for a game engine's asset model, and Godot's own editor, which isn't meant for freeform visual composition. Palettes, sprite sheets, and UI mockups lived outside the project, so nothing was ever quite "in sync" with the actual scene files. The brief was to design a lightweight, native, distraction-free tool that speaks Godot's language natively rather than bolting design work on as an afterthought.

Design Goals

- Keep the tool fast and native — instant launch, no bloat, minimal chrome around the canvas.
- Make every design artifact (palette, mockup, sprite set) exportable in a Godot-ready format.
- Establish one consistent visual language shared with CodeFlux so switching tools feels seamless.
- Reduce eye fatigue during long working sessions with a considered dark, neon-accented theme.

Research & Process

Research centered on how solo and small-team Godot developers currently handle design work, and where friction shows up most. Three recurring pain points shaped the direction of DesignFlux.

Key Findings

- **Fragmented toolchains** — designers exported assets from generic tools, then manually re-imported and re-aligned them inside Godot, losing time and fidelity at every handoff.
- **No shared visual language** — teams pairing a coder and a designer had no common theme between their tools, which made pair sessions and reviews feel disjointed.
- **Palette and typography guesswork** — in-engine colors and fonts often looked different once rendered in an actual game build, with no reliable way to preview them beforehand.

Process

The process followed four light, fast-moving stages, matching the "no bloat" philosophy of the product itself:

- **Discover** — mapped the end-to-end asset workflow of a typical Godot project, from mockup to shipped scene.
- **Define** — scoped DesignFlux around five core jobs: mockups, palettes, sprite/icon organization, typography preview, and packaging.
- **Design** — built the modular workflow and dark neon system shared with CodeFlux, then translated it into the product site and app screens.
- **Validate** — reviewed the four-step install-to-first-project flow to keep onboarding under a few minutes.

At a Glance

2	5	4
apps, one shared design system	core DesignFlux capabilities	steps from install to first project

Information Architecture

The product site and the app follow the same top-level structure, so a user's mental model carries over from the moment they land on the page to the moment they open the app:

- **Why Flux Studio** — the pitch: fast, Godot-first, one visual system, free.
- **The Suite** — CodeFlux and DesignFlux presented as two equal, standalone-or-combined tools.
- **Get Started** — a single, numbered path from download to first launch, repeated identically for both apps.

Visual Design System

DesignFlux shares one cohesive "dark neon" aesthetic with CodeFlux, tuned specifically to stay comfortable during long working sessions. The palette leans on a near-black base with a single confident accent color, so attention is drawn to the canvas and the work — not the chrome around it.

Core Principles

- **One system, two apps** — identical spacing, type scale, and accent logic across CodeFlux and DesignFlux so the suite reads as a single product, not two bolted-together tools.
- **Purposeful minimalism** — every panel earns its place; the interface gets out of the way of the mockup, palette, or sprite sheet being worked on.
- **Feature cards over text walls** — capabilities are presented as scannable, icon-led cards (■ ■ ■ ■ ■ ■) rather than long paragraphs, matching how developers actually skim documentation.
- **Consistent iconography** — a lightning bolt for CodeFlux, a spark for DesignFlux — small marks that make each tool instantly identifiable at a glance.

Core DesignFlux Capabilities

UI MOCKUP CANVAS	A dedicated space for composing UI mockups mapped directly to Godot scenes.
PALETTE GENERATOR	Purpose-built color palette creation tuned for in-game use, not generic web design.
SPRITE / ICON ORGANIZER	Keeps icon and sprite frames organized as first-class, browsable assets.
TYPOGRAPHY PREVIEW	Lets designers preview how in-game fonts will actually render before committing.
EXPORT PACKAGING	One-step, export-ready packaging so finished assets drop straight into a Godot project.

Interaction & Onboarding Flow

Onboarding was designed to be finished before a user's attention wanders — a single numbered sequence, identical in shape to CodeFlux's, so the second app in the suite needs zero relearning:

- **1. Run the installer** — `DesignFlux_Setup.exe` walks through a standard, no-admin-wizardry setup.
- **2. Locate the install** — the app lands in a predictable default folder, kept consistent with CodeFlux's pattern.
- **3. Launch** — a desktop shortcut or direct executable launch, no extra configuration screens.
- **4. Start a project** — create new, or import an existing Godot 4 assets folder, to begin designing immediately.

Outcomes

The result is a focused desktop companion that treats design work as a first-class part of the Godot pipeline rather than an external detour. By sharing its visual system and onboarding shape with CodeFlux, DesignFlux reduces the cognitive cost of using both tools together, while its five core features map directly onto the real jobs designers reported doing before every scene ships.

- A shared design language across CodeFlux and DesignFlux turns two separate downloads into one cohesive studio.
- The four-step install flow keeps time-to-first-project short and friction-free.
- Export-ready packaging closes the loop between "designed" and "in the scene," the core problem the research surfaced.
- A completely free, no-subscription model keeps the tool accessible to solo developers and small teams alike.

Reflection

The biggest design challenge was resisting feature creep — it would have been easy to let DesignFlux grow into a generic image editor. Anchoring every decision to "does this serve a Godot scene" kept the tool lightweight and kept the interface honest about what it's actually for. The shared system with CodeFlux also proved that consistency, not just individual polish, is often the highest-leverage UX decision in a multi-app product.

Case study prepared by **Ali Ahmad Khan**, UI/UX Researcher & Designer.

Product site: studio-flux.netlify.app