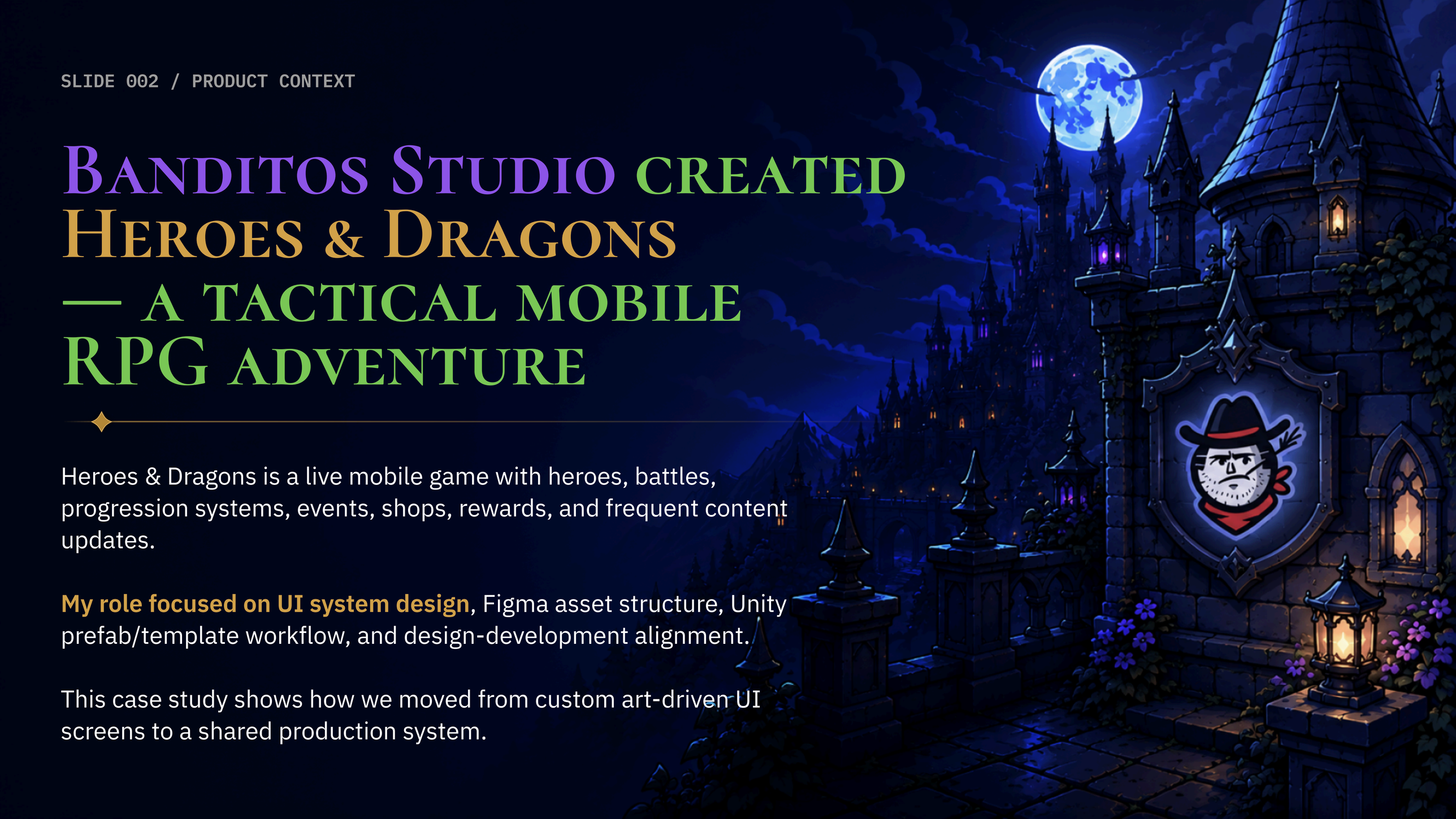
BUILDING A SCALABLE UI SYSTEM
FOR A LIVE MOBILE GAME

Vitali Levit

Senior Product Designer
https://vitali.design

BANDITOS

Banditos Studio created Heroes & Dragons — a tactical mobile RPG adventure



Heroes & Dragons is a live mobile game with heroes, battles, progression systems, events, shops, rewards, and frequent content updates.

My role focused on UI system design, Figma asset structure, Unity prefab/template workflow, and design-development alignment.

This case study shows how we moved from custom art-driven UI screens to a shared production system.

A live mobile RPG product with frequent updates

Growing UI complexity created a need for a more scalable production model

The product needed **frequent updates**: new screens, events, offers, modes, and ongoing changes after release.

My main contribution focused on the company’s new game, where we had an opportunity to **rethink how UI was created** from the beginning. As the UI grew, a custom screen-by-screen production model became harder to maintain and slower to scale.

UI WAS TREATED AS CUSTOM ART, NOT AS A REUSABLE SYSTEM

Each screen was designed individually, and many decisions were made visually and subjectively – “make it bigger”, “move it a bit”, “try another version”.

That worked while the product was small, but **became harder to scale as the game grew**.

There was no reusable structure, no clear system, and no strong connection between design and implementation. Every new screen or change could become a custom effort. That created repeated rework, duplicated assets, and a process where many UI decisions had to be reviewed again from scratch.

A shift from static screens to implementation-aware UI

We shifted UI from art-driven screens to a system-driven workflow

I brought product and design-systems thinking into the process, based on my previous experience at **Autodesk**.

Together with engineering, we translated that thinking into reusable structures, implementation-aware components, and a **workflow that could support live production**.

The shift was not only about visual consistency. It changed how we thought about UI: from static screens to reusable structures, patterns, and production-ready components.

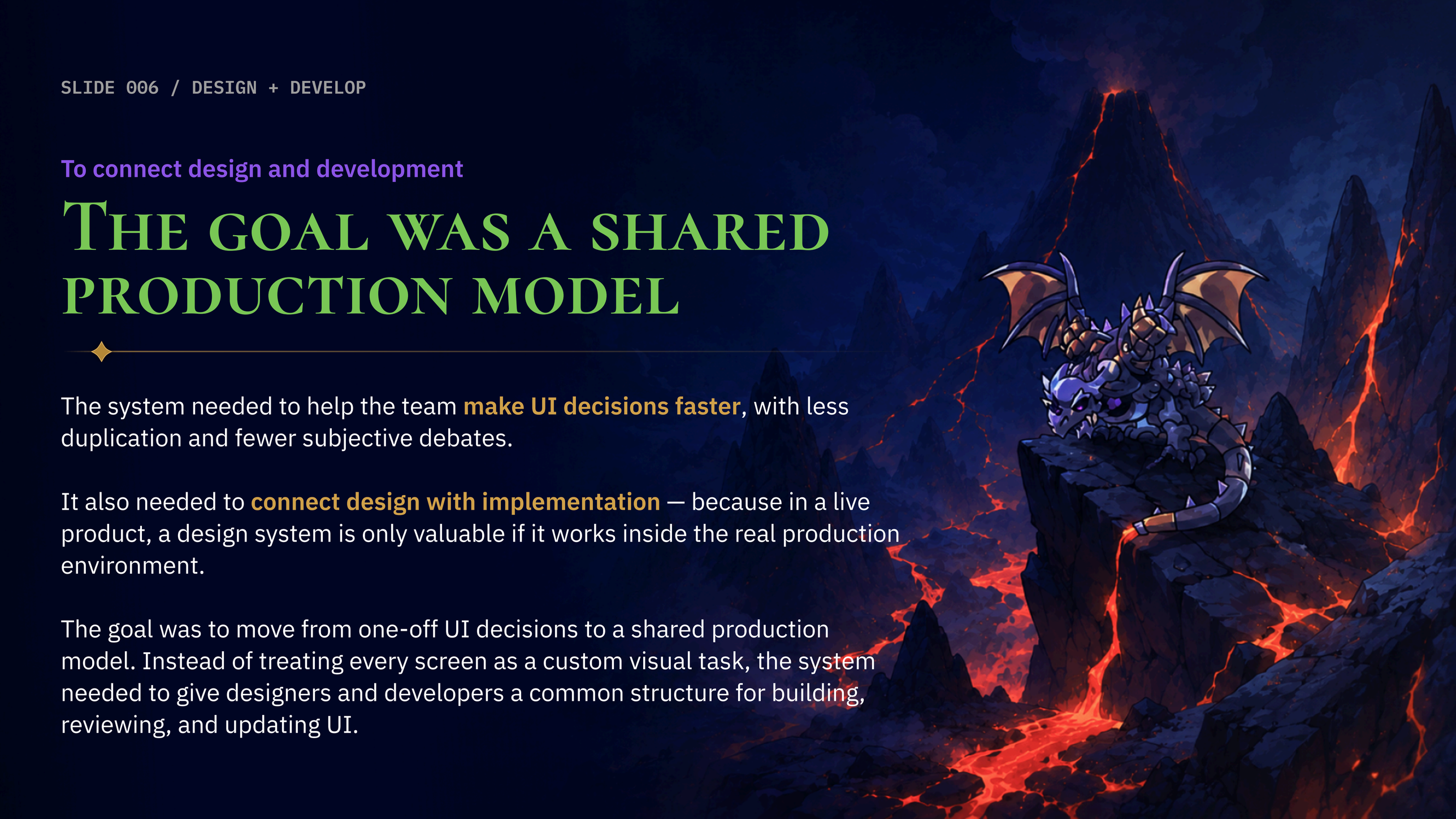
To connect design and development

The goal was a shared production model

The system needed to help the team **make UI decisions faster**, with less duplication and fewer subjective debates.

It also needed to **connect design with implementation** — because in a live product, a design system is only valuable if it works inside the real production environment.

The goal was to move from one-off UI decisions to a shared production model. Instead of treating every screen as a custom visual task, the system needed to give designers and developers a common structure for building, reviewing, and updating UI.



From screen-by-screen design to reusable structures

Core building blocks created consistency, scalability, and reuse

Instead of designing each screen as a separate final image, I defined **reusable building blocks** that could support multiple screens and features.

This meant thinking about **UI at multiple levels**: visual style, component structure, layout behavior, implementation constraints, and reuse across different features.

This is where the work became a **design systems case** — not only designing what screens looked like, but defining how screens should be built.

The design system was built across multiple production layers

Typography: Reusable text styles for regular, outlined, and heavy outlined UI text.

Colors: Named Unity color styles, so colors could be reused semantically instead of copied by hex code.

Components: Reusable prefabs for buttons, bars, cells, tabs, timers, badges, progress bars, shop panels, offer cards, and currency displays.

Templates: Reusable structures for dialogs, popups, empty screens, hero selection flows, and play mode panels.

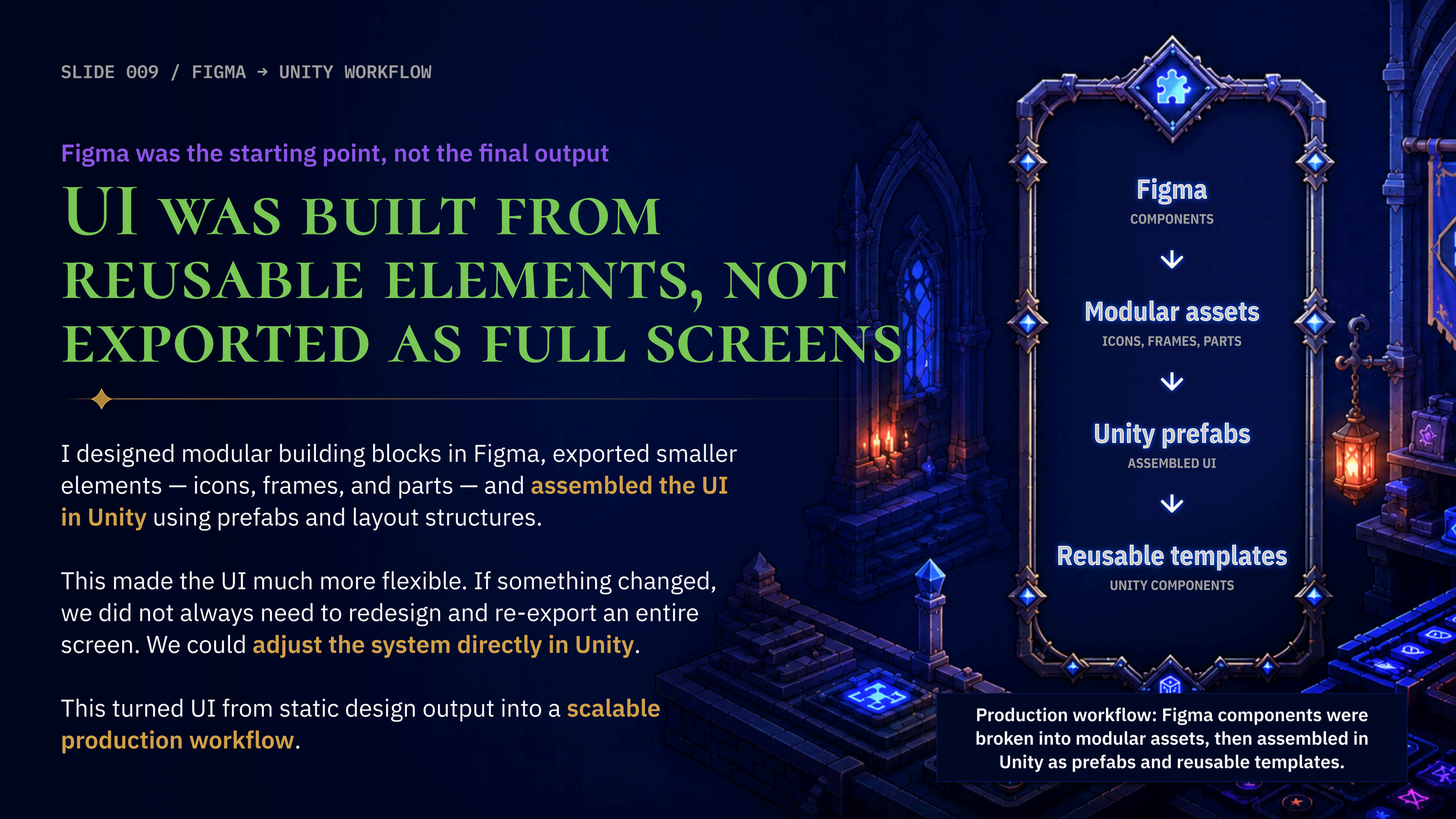
Figma was the starting point, not the final output

UI WAS BUILT FROM REUSABLE ELEMENTS, NOT EXPORTED AS FULL SCREENS

I designed modular building blocks in Figma, exported smaller elements — icons, frames, and parts — and **assembled the UI in Unity** using prefabs and layout structures.

This made the UI much more flexible. If something changed, we did not always need to redesign and re-export an entire screen. We could **adjust the system directly in Unity**.

This turned UI from static design output into a **scalable production workflow**.



Production workflow: Figma components were broken into modular assets, then assembled in Unity as prefabs and reusable templates.

From baked images to editable UI parts

Modular, layered UI graphics replaced fully baked images

Another important part was rethinking how UI graphics were created and used.

Instead of relying on fully baked UI images, I broke **visual elements into smaller reusable parts** — shapes, shadows, highlights, outlines, and strokes.

These elements could be **layered and reused inside Unity**. That made the UI more editable, adaptable, and easier to update during production.

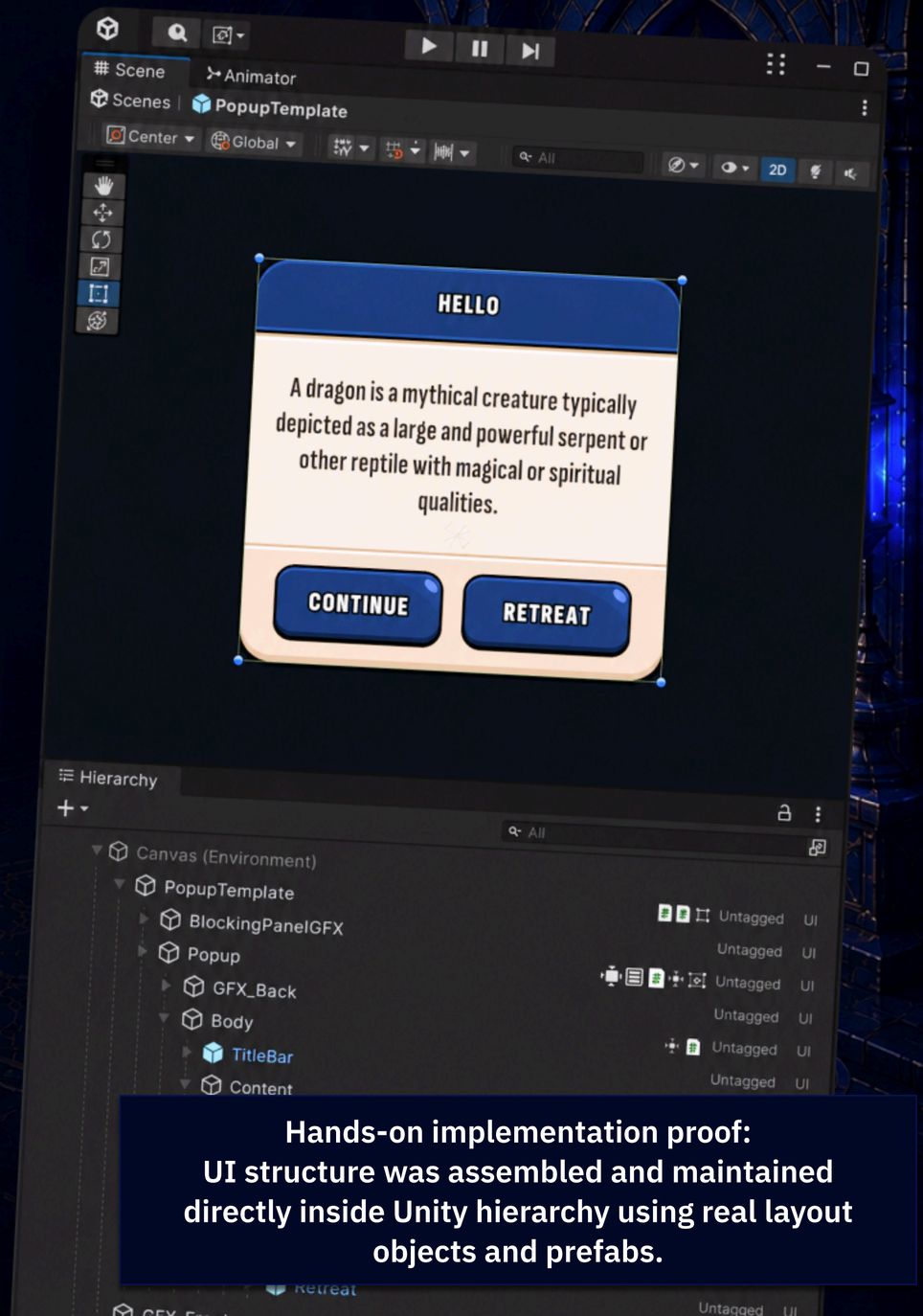
The system had to work where the product was built

Unity became part of the design system workflow

Because the UI was assembled in Unity, design decisions had to work with real layout structures, prefabs, and production constraints.

This helped keep the system practical, maintainable, and connected to how the game was actually built.

The goal was not to create a design-only library, but a system that could survive inside the real production environment.



Hands-on implementation proof:
UI structure was assembled and maintained directly inside Unity hierarchy using real layout objects and prefabs.

Reusable components on the implementation side

Unity prefabs became real design system components

Elements like title bars, buttons, and bottom bars became reusable UI components instead of custom pieces created again and again.

This gave us a more **maintainable UI architecture**. When a component needed to change, we could update and reuse it across different parts of the product.

The design system was not only a Figma library — it became a real implementation system.

To make the system usable in real production

Templates turned the system into production infrastructure

A design system is only useful if people can actually use it during production.

I introduced reusable **UI templates that gave developers a structured starting point**. They could start building new screens from empty screen templates or use panel templates that already had the right structure.

This helped development **move forward before the final UI was fully polished**, while still staying aligned with the system.

Templates turned repeated screen patterns into ready-to-use starting points for production.

From static output to adjustable production system

UI BECAME EDITABLE, REUSABLE, AND SCALABLE

Because UI was assembled from reusable parts, changes could happen inside the system **(visual changes directly in Unity)** instead of restarting the whole asset pipeline.

That improved iteration speed in a fast-paced production environment.

The main shift was simple: **UI was no longer a static image**. It became something the team could adjust, reuse, and scale.

This changed how design and development worked together

The system gave design and development a shared structure.

Developers could start implementation earlier from reusable templates and prefabs, before the final UI was fully polished. We could **work more in parallel, reduce blockers**, and make changes without restarting the whole asset pipeline.

In a live game environment, that mattered because **updates and features moved quickly**. The system helped the team move faster with more structure.

The system improved speed, consistency, and team alignment

Production speed

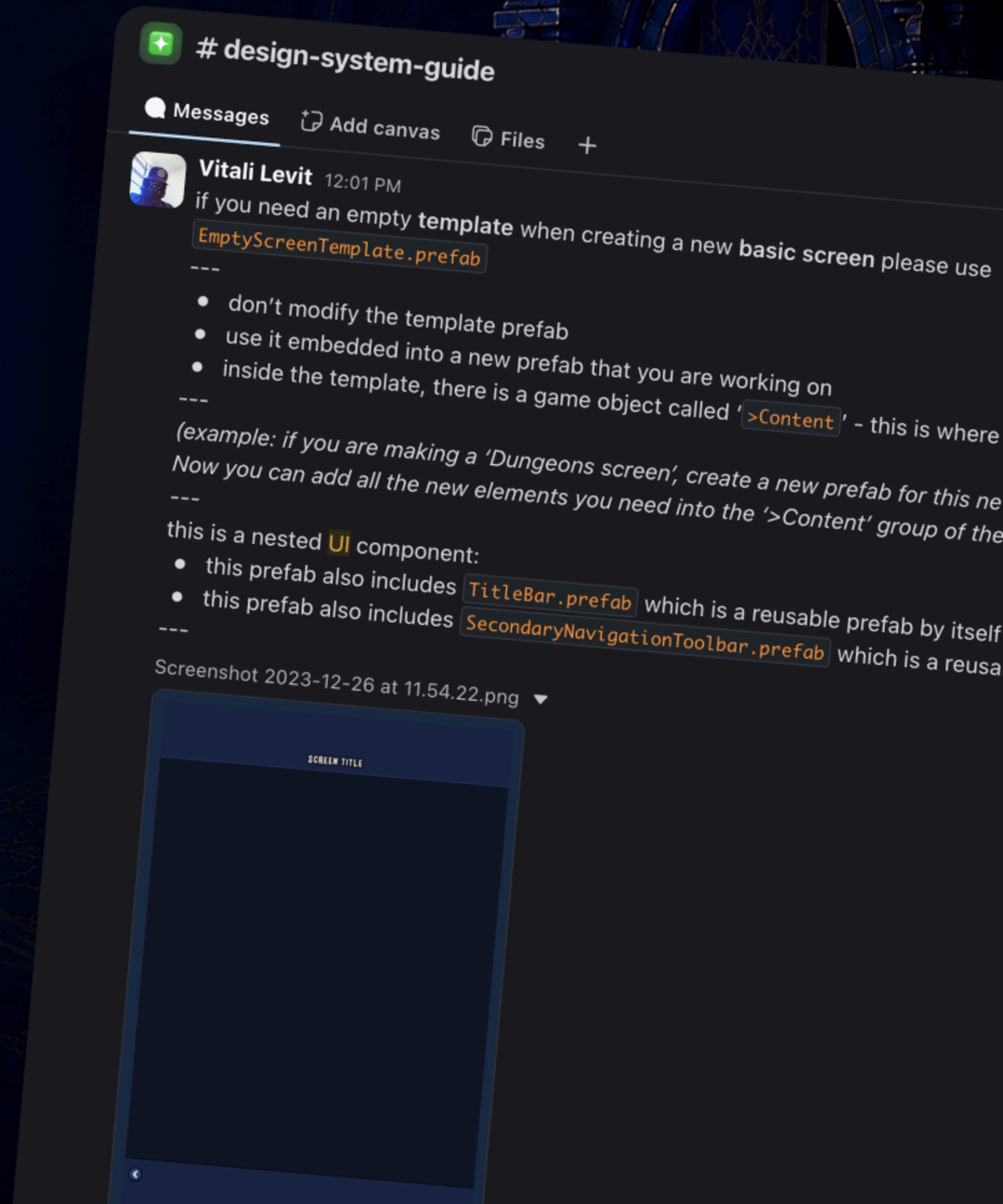
Reusable templates and prefabs helped developers start earlier and reduced waiting on finalized screen assets.

Maintainability

Modular assets and Unity prefabs reduced duplication and made repeated UI structures easier to update.

Team alignment

The team gained a shared structure for UI decisions, reducing subjective debates and improving consistency across screens.

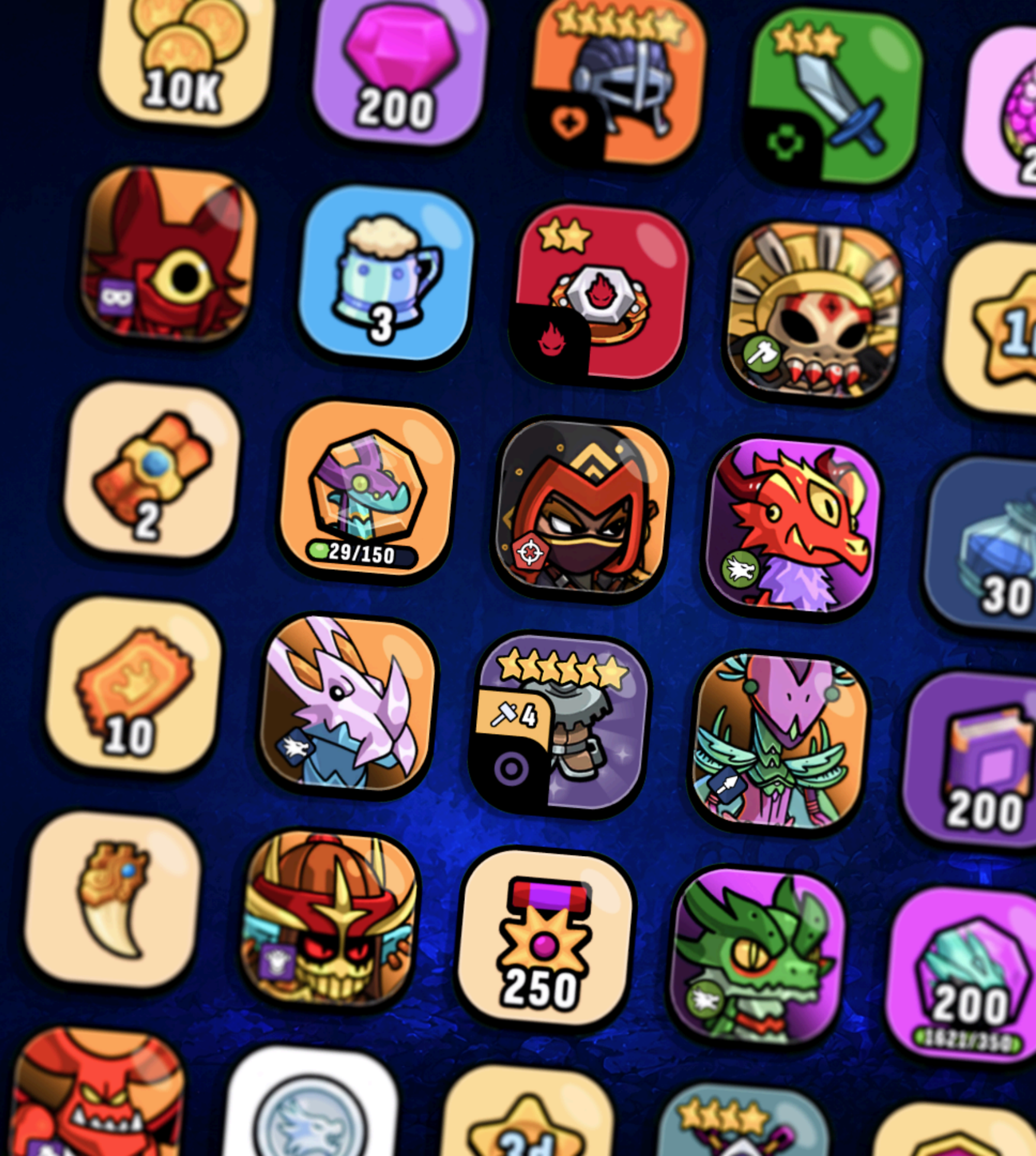


UI IS NOT JUST VISUALS — IT'S A PRODUCTION SYSTEM

The main lesson from this project is that UI is not only what players see. It is also how teams build, maintain, and scale the product.

This case shows how I **connect design systems with production reality**: consistency, reuse, speed, maintainability, and collaboration with engineering.

I helped the team move from custom art-driven UI screens to a shared production system — where design, assets, Unity prefabs, templates, and engineering **workflow worked together**.



Product gallery: Live game screens

Product gallery: Dungeons, Character, and progression UI

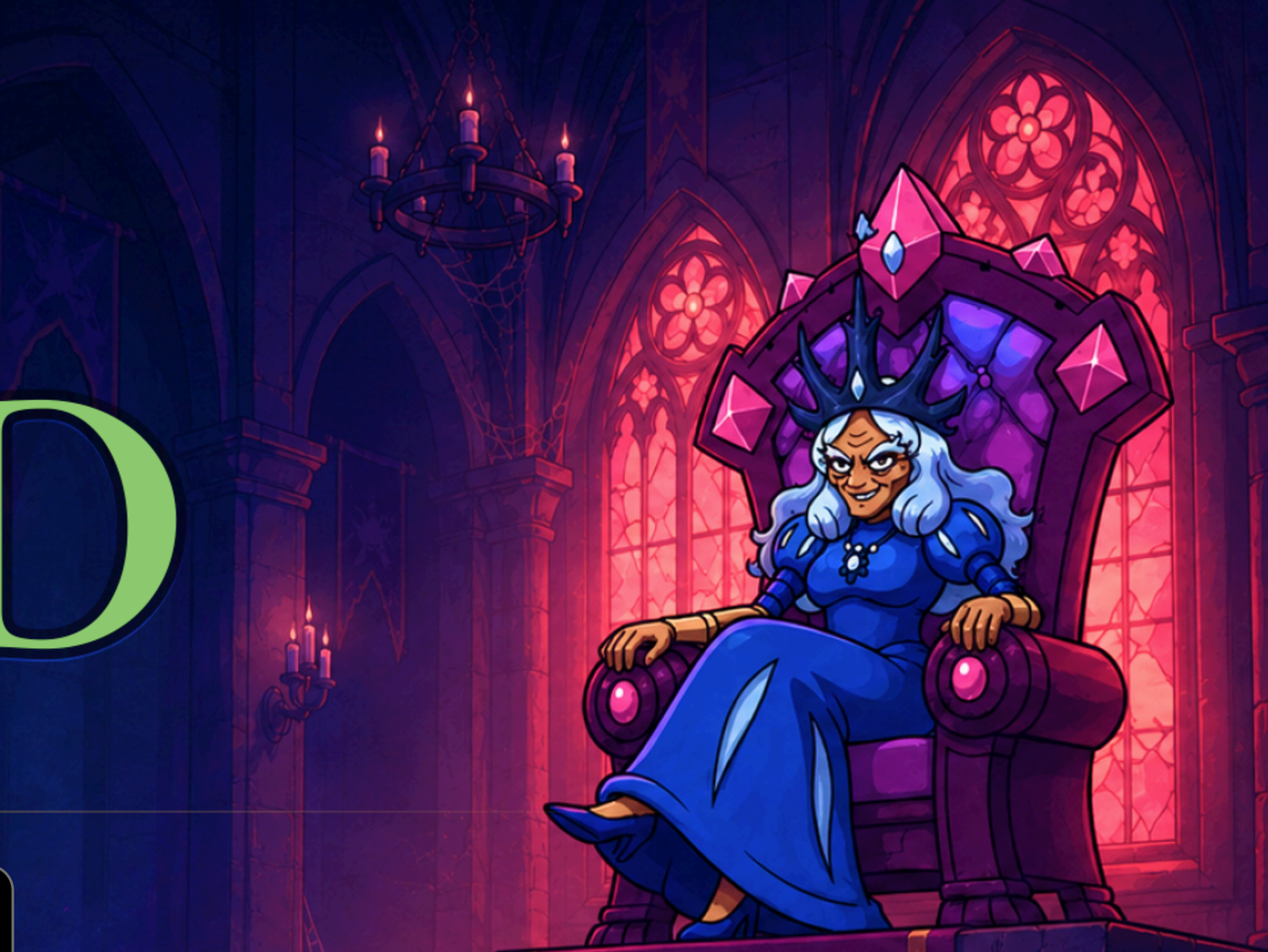
Product gallery: Battle modes and reusable layouts

BUILD YOUR LEGEND

TACTICAL RPG ADVENTURE

Download on the App Store

GET IT ON Google Play



BANDITOS

HEROES & DRAGONS