STRETCHLEARN · WORKBOOK

Game UI Design — Workbook

This workbook turns the course into the decisions, sketches, and specs you need to design a real game interface. Work each section as you go: classify your elements with the diegesis matrix, lay out a HUD around the focal point and safe zones, design legible indicators and navigable menus, plan your feedback and juice, and prepare an engine-ready handoff. The templates are built to be filled in for your own game and reused as the working documents you take into Unity or Unreal.

Foundations: What Game UI Is and the Diegesis Matrix

Inventory every piece of information the player needs, classify each one with the diegesis matrix, and sketch a HUD that respects safe zones, hierarchy, and the player's focal point.

Worksheet: Element Inventory and Diegesis Matrix

List every piece of information your player needs to see, then classify each with the two matrix questions and decide its treatment. Do this before you draw anything; it is the plan the rest of the UI follows.

Information element (e.g., health, ammo, objective, map, interaction prompt)

Is it part of the story/fiction? (yes / no)

Does it live in the 3D world space? (yes / no)

Resulting quadrant (non-diegetic / diegetic / spatial / meta)

Priority for this game (immersion-first or clarity-first)

Chosen treatment (e.g., corner bar, glowing armor, screen-edge effect)

Is it safety-critical / fast-read? (yes / no)

If fast-read, does the chosen treatment keep it legible? (yes / no)

Exercise: Sketch Your HUD Layout

On paper or in any tool, draw your game's frame with a 5-10 percent safe-zone margin marked, then place each element from your inventory. Justify every position against the focal point and your priority order.

• Where is the player's focal point during normal play (crosshair, character, interaction target), and which urgent elements did you place near it?

• Did you keep all readable UI inside the title-safe margin (about 5-10 percent from every edge), and where did you mark that boundary?

• What is your priority order (e.g., health, then ammo, then objective, then map), and how does size, contrast, or position reflect it?

• Which conventional positions did you honor (health bottom-left, ammo bottom-right, minimap a top corner) and where did you deliberately break convention and why?

Checklist: Foundations Readiness Check

[ ] I listed every information element the player needs before designing any visuals
[ ] I classified each element into a diegesis quadrant on purpose, not by default
[ ] I chose immersion-first or clarity-first treatments deliberately per element
[ ] All readable UI sits inside a 5-10 percent safe-zone margin from every edge
[ ] The most urgent elements are near the focal point or carry the strongest visual signal
[ ] My HUD is quiet by default and escalates only when an element becomes urgent

Designing the HUD and Core Indicators

Specify your health and resource meters, define the feedback states for ammo and cooldowns, and tune the navigation layer so guidance helps without clutter.

Worksheet: Indicator Spec Sheet

Define each core indicator so it is legible at speed and at distance. Fill one row per indicator (health, shield/mana, ammo, key ability) and make sure no signal relies on color alone.
Indicator name (health / shield / ammo / ability)

Representation (continuous bar / segmented pips / numeric / diegetic-meta)

Primary instant-read cue (length / count / color / position)

Secondary cue so it does not rely on color alone

Danger / low-state treatment (color change, pulse, flash, sound)

Change feedback (chip/ghost damage, number pop, etc.)

Readable in peripheral vision at game speed? (yes / no)

Readable on a TV at couch distance? (yes / no)

Exercise: Map the States of One Ability

Pick one ability and define every state it can be in and the visual for each, then describe the transitions. The test is that no two states can be confused in a glance at full game speed.
• What are all the states this ability can be in (ready, charging, on cooldown, disabled)?

• What distinct look (color, brightness, overlay like a radial sweep) does each state get so none overlap?

• How is the moment it becomes usable marked (flash, sound) to recapture the player's eye?

• When you watch the transitions at 60 frames per second, does each state still read, or does a transition

disappear?

Worksheet: Navigation Layer Plan

Decide how much guidance your game gives and which tools deliver it. Aim for just enough direction at the right moment without burying the world under icons.

Navigation tool (minimap / compass strip / world waypoint / edge arrow / object highlight)

What it shows and what it deliberately omits

When it appears and when it fades (distance, relevance)

Genre fit reason (why this tool suits your game)

Maximum on-screen markers allowed before it becomes noise

Immersion vs coordination setting (sparse/diegetic ... rich/tactical)

Checklist: HUD Legibility Check

- [ ] My health bar passes the frightened-player test: status is clear without looking directly at it
- [ ] No critical indicator relies on color alone; each pairs color with length, count, position, or pattern
- [ ] Bars and text hold contrast over both bright and dark backgrounds (outline or shadow)
- [ ] Ammo shows magazine prominently with reserve smaller, and warns when nearly empty
- [ ] Every ability state (ready, cooldown, charging, disabled) is visually distinct in motion
- [ ] Navigation markers are capped and fade with distance so the world is not buried in icons

Menus, Inventory, and Interaction Flow

Lay out an inventory grid that scales, design menus and radial wheels that work on a gamepad, and lower the interaction cost of your framing and onboarding screens.

Worksheet: Inventory Grid Designer

Decide your inventory structure and grid dimensions, trading overview against legibility for your target screen and input. Define what the selection panel shows so every comparison is effortless.

Inventory philosophy (abstract one-slot grid / spatial tetris grid)

Total item capacity

Visible grid dimensions (e.g., 6x10 all visible, or 5x4 scrolling)

Reason for those dimensions (full overview vs larger legible icons)

What the selection panel shows (name, stats, compare-to-equipped)

Sorting and filtering options (type, rarity, newest)

Rarity color ramp used (e.g., grey/green/blue/purple/orange)

How the current selection is highlighted for gamepad traversal

Exercise: Design a Radial Menu and Its Focus Order

Design one radial/wheel menu and one standard menu screen for the gamepad first. Define the segments and, for the standard menu, the exact highlight movement so there are no dead ends.

• How many segments does your radial menu have (keep it 4-8), and what clear icon marks each?

• How is the currently aimed segment highlighted so it is unmistakable with a thumbstick?

• For your standard menu, where does the highlight move on up/down/left/right from each item, with no traps or dead ends?

• Which button glyphs do you show for confirm and back, and do they swap to match the connected controller?

Worksheet: Interaction Cost Audit

List the most common player tasks in your framing screens and count the steps each one takes. Anything frequent that is buried is a candidate to move closer to the surface.

Common task (resume, restart, change volume, remap a control, save, quit)

Number of steps/clicks to complete it today

Is the current value shown without opening it? (yes / no)

Is it appropriately confirmed if destructive? (yes / no)

Convention honored (Esc/Start to pause, bottom face button to confirm)? (yes / no)

Target steps after redesign

Checklist: Gamepad and Onboarding Check

[ ] I can navigate the entire UI with a controller alone, with no unreachable element or trapped highlight
[ ] Selection state is always strongly visible since there is no cursor on a gamepad
[ ] Radial menus have 4-8 segments with clear icons and a strong aimed-segment highlight
[ ] Targets and text are sized generously for a TV at couch distance
[ ] Onboarding teaches one mechanic at a time with contextual prompts, not a wall of text
[ ] Accessibility options exist: UI scale/text size, color-blind support, remapping, subtitles

Game Feel, Feedback, and the Engine Handoff

Plan the juice that makes the UI feel alive, lock a consistent visual style, and produce a handoff with anchors, 9-slice sprites, states, and specs that build cleanly.

Worksheet: Juice and Feedback Plan

For each key action and event, define the feedback that confirms it. Match intensity to the event and keep frequent actions short, and make sure no effect hides the information underneath.

Action or event (button hover, button press, take damage, pickup, critical hit)

Visual feedback (scale, glow, flash, shake, number pop)

Easing curve (ease-out, slight overshoot/bounce, linear) and duration in ms

Sound and/or haptic paired with it

Intensity level (subtle for frequent, loud for rare)

Does it ever obscure critical info at the wrong moment? (yes / no)

Is there a reduced-motion / screen-shake toggle for it? (yes / no)

Exercise: Define Your UI Style Guide

Pin down the art direction of your interface so it belongs to your game's world and stays consistent across every screen. Then test legibility at the real target size and distance.

• What is your type system (display face for headers, legible face for body/numbers), and does it read at the target viewing distance?

• What is your tight color palette, and which saturated colors are reserved for critical signals (damage red, ready green)?

• What is your shape language (sharp vs rounded, heavy vs thin panels), and your icon set's consistent stroke and perspective?

• When you view the whole UI in grayscale, does the hierarchy and contrast still hold without color doing the work?

Worksheet: Engine Handoff Spec

Fill one row per UI element so an engineer can build it in Unity or Unreal without guessing. Anchors, slices, states, and exact specs are what keep your design from becoming a rough approximation.

Element name

Anchor (which corner/edge/center it pins to)

Is it 9-slice? If yes, where do the slice lines fall

States exported (default, hover/focus, pressed, disabled)

Export resolution, format, and naming

Exact size, spacing, and color values

Font and text size

Animation easing and duration; updates every frame or static?

Checklist: Handoff Readiness Check

[ ] Every key action has feedback matched in intensity to the event, kept short for frequent actions
[ ] No flash, shake, or effect hides critical information at the moment it matters
[ ] The UI style (type, color, shape, icons) is consistent across HUD, menus, and inventory
[ ] All text and icons were tested at the real target size and distance, and survive grayscale
[ ] Every element has an anchor defined and 9-slice lines marked where panels/buttons stretch
[ ] All interactive states and icons are exported at target resolution with clean alpha and naming
[ ] Specs are documented: sizes, spacing, color values, fonts, and animation timings
[ ] Performance intent is noted: which elements update each frame and which effects cost draw calls

Your Action Plan

1. Inventory every information element the player needs, then classify each with the diegesis matrix and choose its treatment.
2. Sketch the HUD with safe-zone margins marked, placing urgent elements near the focal point and honoring position conventions.
3. Spec each core indicator (health, shield, ammo, abilities) for instant legibility, never relying on color alone, and map every ability state so ready and not-ready can never be confused.
4. Decide your navigation layer on the immersion-versus-coordination dial and cap on-screen markers.
5. Design the inventory grid dimensions and selection panel to balance overview against legibility for your screen and input.
6. Design menus and radial wheels for the gamepad first, defining focus order and selection state with no dead ends.
7. Audit interaction cost on framing screens and pull frequent tasks closer to the surface; plan contextual onboarding.
8. Plan juice and feedback for every key action with easing and durations, plus a reduced-motion toggle.
9. Lock a UI style guide for type, color, shape, and icons, and test legibility at the real size and distance.
10. Produce the engine handoff (anchors, 9-slice lines, exported states, full specs for Unity or Unreal), then navigate the whole UI with only a controller to catch unreachable elements before you ship.