

CASE STUDY · GAME DEVELOPMENT

APOSTLE

Grace Remains

One Unity game shipped to iPhone, iPad, Mac, Android phones and Android tablets in eleven weeks, with a privacy footprint of a single permission and every level proven finishable before each build.

AT A GLANCE

The project in one page

APOSTLE: Grace Remains is a first-person adventure game with no attack button. The player is an apostle in training who receives what is thrown at them and spends it healing. We wanted a respectful, faith-based game that plays well on every screen people actually own, collects nothing about its players, and that our small team could maintain.

5
form factors shipped: iPhone, iPad, Mac, Android phone, Android tablet

1
Android permission requested (vibration). No network access in the Google Play build

0
ads, purchases, accounts or data types collected

18
scenes generated from code by a 5,084-line level builder

76
objectives machine-checked as reachable before every build, 0 blocked

178
countries and regions on Google Play; all 175 App Store territories

PRODUCT	APOSTLE: Grace Remains, our own game, published by ProjectThunder.com, Inc.
PLATFORMS	App Store (iPhone, iPad, Mac), Google Play (phones and tablets), direct Android download
ENGINE AND STACK	Unity 6 (6000.5.7f1), Universal Render Pipeline, IL2CPP, C#; native Objective-C plugins for iOS audio, haptics and window controls; Python and shell release tooling
TIMELINE	First commit 6 August 2026. App Store 1.0 submitted 20 August; 1.1 live on iPhone, iPad and Mac; iOS 27 build on TestFlight 7 October; Google Play submitted 22 October and approved
SCOPE	Game design, level generation, cross-platform engineering, release automation, store listings, privacy and ratings, website, marketing video
RATINGS	ESRB Teen · PEGI 12 · USK 16 · ClassInd 10+ (IARC); App Store 12+

THE CHALLENGE

A faith game that respects its players

Most games built around a story of faith fall into one of two traps. Some turn the sacred into a power-up and sell it. Others become a quiz with a coat of paint. We wanted neither, and set firm lines before a single scene was built:

- **Jesus is never shown and never voiced,** and no words are put in his mouth. The player is a follower, not the Messiah.
- **No spiritual mechanic is ever for sale.** No paid miracles, no paid streak repairs, no grace in a store.
- **No miracle can be failed on purpose for drama**, and no scripture is used as character dialogue.
- **No ads, no accounts, no tracking.** The game must work entirely offline.
- **Teen and adult positioning**, not the Kids category, so the content can carry real weight: occupation, an arrest, a wound.

The engineering challenge was just as specific. Our small team had to ship one codebase to five form factors, from a 2.17:1 iPhone to a 4:3 iPad window to a Mac desktop, and keep every one of them working through an operating-system generation change (iOS 27) that landed mid-project.

> *"But he passing through the midst of them went his way." Luke 4:30 (KJV), the verse the game's Caesarea finale is built on.*



The telling: a tent revival frames each round.



Caesarea: healing under the enforcers' watch.

DESIGN

One verb: receive

The design replaces the attack button with a single, counter-intuitive action. Hostile people throw stones. Pressing RECEIVE at the right moment means the stone lands *in* you instead of *on* you, and what was thrown in anger becomes Grace. Grace powers one aimed beam that heals the suffering, quiets accusers and cleanses what blocks the road. Receive enough of a thrower's stones and his anger runs out: the player breaks him by absorbing him, not by aiming at him.

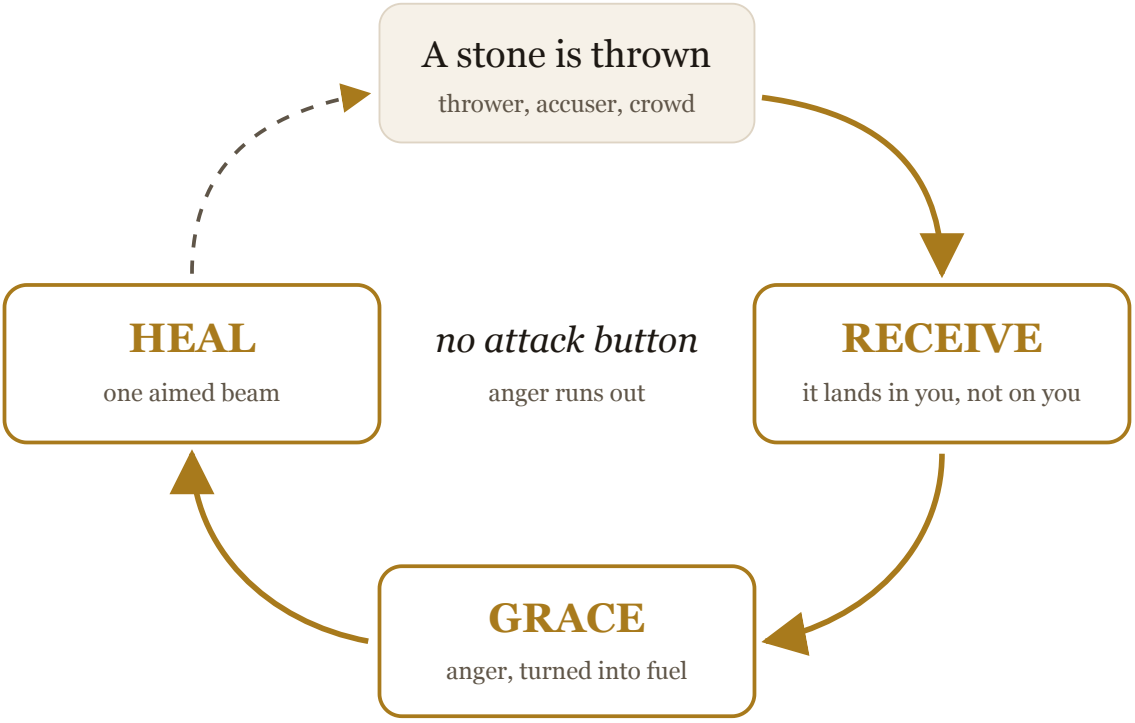


Figure 1. The core loop. The dashed edge is the only "combat": the world keeps throwing; the player keeps receiving.

Six rounds, each worth walking

The campaign runs through the Jordan, Capernaum, the Decapolis, Caesarea, the Temple courts and the Jericho Road. Each round is large enough to explore without being led: accusers drain Grace while they talk, waystones restore it, stones arriving close together can be chained, weather crosses open ground, and fear is a meter of its own. A hidden seventh round, a wedding at Cana, rewards players who look carefully.

Restraint as a feature

The one drawn blade in the game belongs to someone defending the player, and the round will not end until the player heals the man it wounded. We treated every content decision this way: the moment is allowed to be heavy, but the player's only tool is mercy.

ENGINEERING

Levels written in code, proven finishable

Hand-placing six arenas for five screen shapes would have made every change expensive. Instead, a single editor script (5,084 lines of C#) builds all 18 scenes from scratch on demand: terrain, buildings, every character, beacon, waystone, lighting setup and UI. A design change is a code change, reviewed like any other, and the whole game regenerates in minutes.

Generated levels need a safety net, so every build runs a **completability audit**. It flood-fills each arena with the player's own collision capsule, then proves that every person who needs healing can be reached and has a clear line for the beam. It also flags detours: a target that is 17 metres away in a straight line but 60 metres away on foot fails the audit, even though it is technically reachable.



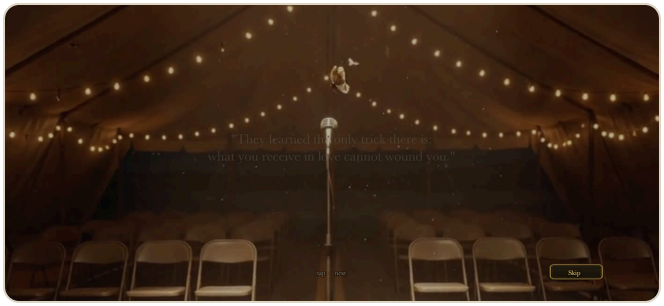
Figure 2. Round VI, the Jericho Road, plotted from the audit's own log: an 85 × 177 m grid, 7,561 reachable cells, 12 of 12 people and one darkness node servable. Across all seven arenas, 76 objectives were checked with zero blocked.

What the audit caught

Before the audit existed, three real progression blockers turned up in playable builds: an NPC authored inside a building's shell, a harbour spawn sealed into a corner, and paired towers whose colliders met mid-road and cut off half of the final round. Each was invisible in screenshots. After the audit, the build fails instead of the player.

PLATFORMS

Five screens, one codebase



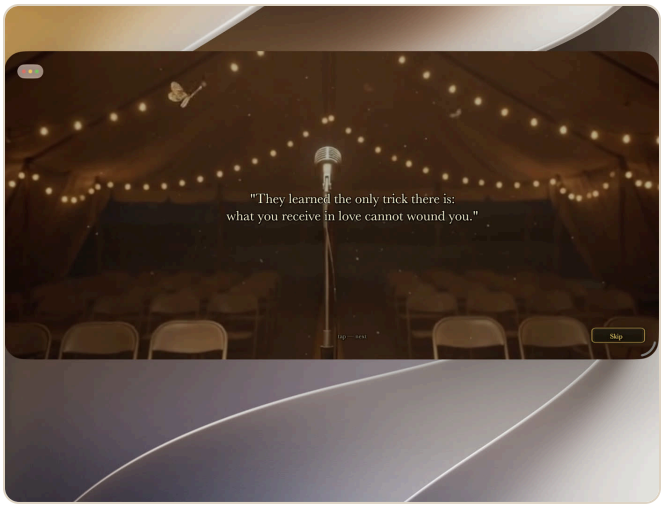
iPhone: 2.17:1, edge to edge.



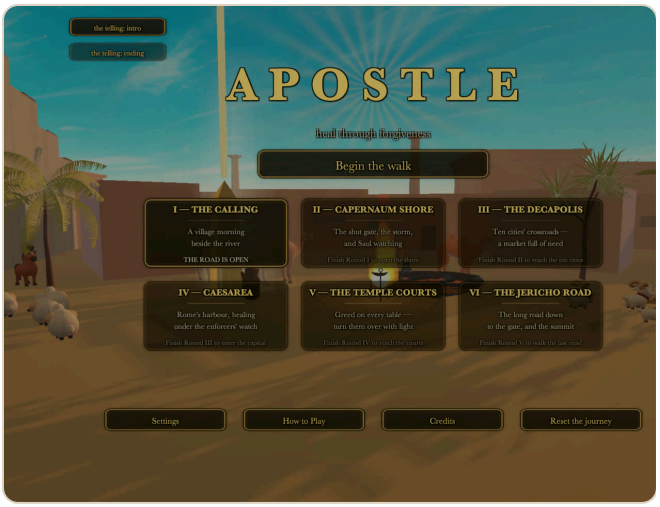
Android phone: the same HUD, thumb-reachable.

iPhone and iPad, ready for iOS 27 on day one

- **Resizable iPad windows.** With the iOS 27 SDK, iPad apps can no longer opt out of free-form windowing. Left alone, the game would have drawn sideways in a portrait window. A custom viewport system now letterboxes anything narrower than 4:3 into a centred band and pillarboxes anything wider than 2.2:1. Cameras, canvas scaling, safe areas and touch zones all follow the band, and the interface keeps clear of the new window-control pill.
- **iPhone Duo.** The folding iPhone needs the iOS 27.1 SDK to fill its screen. We built with the 27.1 release candidate alongside the stable toolchain, without disturbing other apps on the same machine.
- **Audio and input.** iOS 27 flags synchronous audio-session calls as hang risks; moving to the asynchronous API cut the warnings from 13 to 2 per launch. New iPhone models report screen density the engine does not know yet, so the touch stick now falls back to a measured value instead of running 41% too sensitive.



iPadOS 27: a free-form window, letterboxed instead of stretched.



iPhone Duo, outer screen.

Mac

The Mac version ships through the Mac App Store, plus a Developer ID–signed, notarized installer for direct distribution. We found and documented two traps along the way. Unity's optional receipt validation compares Apple's certificate chain against today's date and silently quits the app once Apple's signing certificate expires, so it is off (Apple does not require it). And switching Unity's active platform rewrites render-pipeline settings, so a Mac build made right after an iOS build renders a black world; every build script now pins its target first and never switches away.



Mac: keyboard and mouse, the same world.

Android: Google Play and a direct download

Android ships two ways from one project. The **Google Play** build is an App Bundle targeting API 36 with 16 KB page-size native libraries, and its manifest removes internet access entirely. The **direct download** is a signed APK with a small update check that reads one version file once a day, waits for the menu before prompting, and logs every decision so failures are visible.

Both are signed with the same key, which is also the Play app-signing key, so a player can move between the two and keep their progress. One company key now signs every Android app the studio ships, with a documented rotation path for older apps.



Android tablet: 16:9, the same layout rules.



The Decapolis on tablet.

Windows

A Windows player builds from the same project. It has not been released: it has not yet been tested on Windows hardware, and an untested build is not a shipped one.

QUALITY

Found on real devices, fixed before release

Every platform was exercised on simulators and emulators, and Android and Mac on physical hardware as well: an iPhone 18 Pro Max simulator, an iPad Pro 13" simulator in every multitasking mode, phone and tablet emulators, a Samsung Galaxy Tab, a nubia gaming phone and a Mac. A selection of what that testing turned up:

ISSUE	CAUSE	FIX
Haptics silently failing on Android	The game drives the vibration motor directly for precise amplitudes, so the engine never declared the permission; every buzz threw a security exception inside the Receive and Heal handlers.	Permission added at build time; haptics fail soft and switch off after one error.
Settings buttons overlapping	The panel was laid out before platform-specific rows existed.	Rows stack at runtime and the frame grows only when needed.
Hints cut off mid-sentence	Phones scale reading text 1.6× for legibility; the banner stayed sized for one line.	Wider banner with best-fit text on two lines.
Pause button too small	28 × 23 points, under Apple's 44-point minimum, and on top of the minimap.	72 × 72 units, clear of the map, in every round.
Cut-scene video stretched	Videos filled the screen's aspect, 75% wide on iPad and 123% on iPhone.	Cropped to cover; verified frame-by-frame against the source clip.



Before: Close sits on the HAPTICS row.



After: rows stacked, links and Close inside the frame.

RELEASE ENGINEERING

Shipping without surprises

With five targets and two stores, release mistakes are the expensive kind. We automated the checks that a tired person skips:

- **A gate that can fail.** One rendering bug, a post-processing shader compiled to zero programs, produces a completely black world while the menus still draw, and it shipped twice before it was understood. The release script now refuses to archive unless the build log proves that shader compiled a non-zero number of programs. An earlier version only looked for the failure text and passed silently when the log format changed.
- **A build-number ledger.** App stores reject a reused version or build number only after a 20-minute archive and upload. Every successful upload is recorded in the repository, and the next build is checked against it before anything is compiled.
- **Store APIs, not clicking.** App Store Connect and the Google Play Developer API handle uploads, track assignments, listings, screenshots and keyword changes from scripts, so the listing text lives in the repository next to the code it describes.
- **Robot testing before production.** Android builds go to an internal track first so Google's pre-launch report can exercise them on real devices before a production release.

Privacy you can verify

The privacy promise is checkable. The Google Play build's manifest lists exactly one permission, vibration, and no internet access. The Data safety declaration is "no data collected". The direct-download build adds only the daily update check. Ratings questionnaires were answered against the shipping build, not the pitch: stones are thrown, one sword is drawn, wine appears at a wedding, there are a few frightening moments, and there is no blood, profanity or purchase.

RATING BODY	REGION	RATING
ESRB	North America	Teen
PEGI	Europe	12
USK	Germany	16
ClassInd	Brazil	10+
App Store	Worldwide	12+

LAUNCH

Website, store presence and video

We built the launch kit from the same source material as the game, so every claim matches the build:

- **playapostle.com**: a static, script-free site with App Store, Mac App Store and Google Play links, screenshots, the six rounds and the Android download page. It loads no third-party code, sets no cookies, and deploys through the studio's CI pipeline to its own IIS server.
- **Share cards**: a branded 1200 × 630 image with complete Open Graph and X tags on every page.
- **Store listings**: descriptions, phone and tablet screenshots and a feature graphic, kept in the repository and pushed by API. A search-term review added the words players actually type ("Christian", "Bible", "faith", "game") without keyword stuffing.
- **Vertical trailers**: two 1080 × 1920 spots cut from real device recordings, captioned only with what is on screen, mixed to −14 LUFS.



The Jericho Road.



The Decapolis.



Caesarea.

Where it stands

APOSTLE is live on the App Store for iPhone, iPad and Mac and on Google Play for Android phones and tablets. Version 1.2, with the iOS 27 work and the device fixes above, is built and on TestFlight. The game is new and its audience is just forming; this study reports what was built and verified, not projected downloads.

TAKEAWAYS

What we learned

Generate, then prove. Code-generated levels are cheap to change and easy to break. Pair the generator with an audit that walks the level the way the player does, and validate the audit against a level you know is broken.

Write checks that can fail. A gate that only searches for an error string passes the day the string changes. Require positive proof of the good state.

Test on the screen shape, not the device name. A 4:3 iPad window, a 2.17:1 phone and a 16:9 tablet broke different things. A viewport rule fixed all of them at once.

Make the privacy claim verifiable. "Nothing collected" is stronger when the manifest has no internet permission than when the policy says so.

Let the content rules shape the mechanics. "Never sell a spiritual mechanic" and "no attack button" were constraints. They became the design.

About ProjectThunder

ProjectThunder.com, Inc. is a custom software studio. We build web, commerce and game software, and APOSTLE is one of our own products, designed, built and published in-house.

projectthunder.com · support@projectthunder.com · The game: **playapostle.com**

Sources

Line counts, scene counts and commit history: the APOSTLE git repository. Audit figures: the build's [AUDIT] log. Build numbers: the release ledger in the repository. Permissions: `bundletool dump manifest` on the 1.2 (2) App Bundle. Ratings and Data safety: Google Play Console (IARC) and App Store Connect. Territory counts: App Store Connect availability and the Play production track. Haptic values: `Haptics.cs`. All screenshots are unedited captures from simulators, emulators and physical devices. Vision cut-scenes and parts of the soundtrack were AI-assisted; the game credits disclose this.

APOSTLE: Grace Remains © 2026 ProjectThunder.com, Inc. App Store is a service mark of Apple Inc. Google Play is a trademark of Google LLC. This document may be shared freely in its unmodified form.