

Building a game with multiple AI services

How a Designer, an AI planning assistant, an AI coding agent, generative AI services and 3D asset libraries work together to build a point-and-click adventure Game on the Inferno engine, and where each MCP and API touchpoint sits.

LEGEND

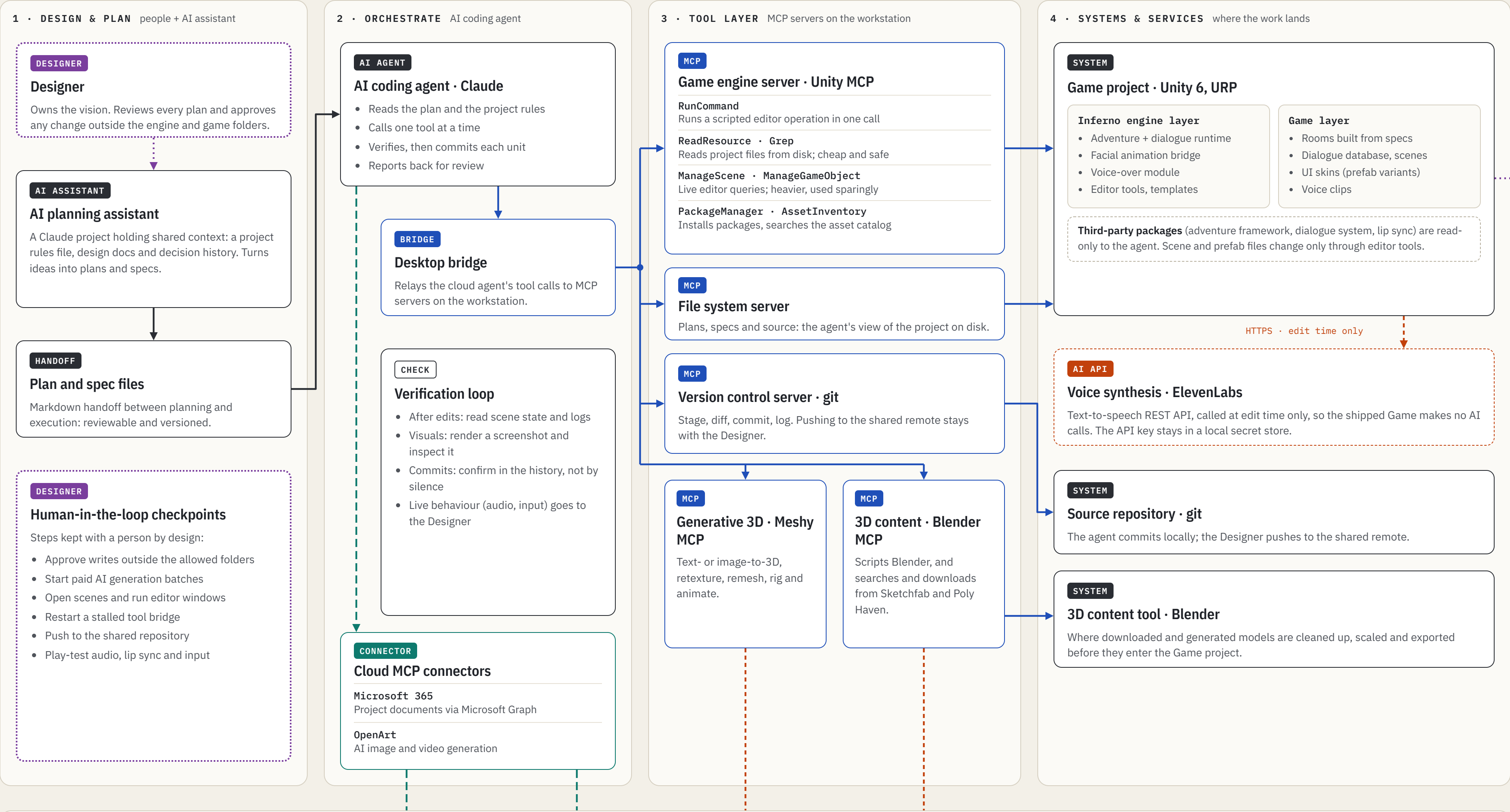
MCP tool call, via the desktop bridge

Third-party API over HTTPS

Handoff or file flow

Cloud MCP connector (OAuth)

Manual step by the Designer



3 · TOOL LAYER

MCP servers on the workstation

MCP

Game engine server · Unity MCP

RunCommand

Runs a scripted editor operation in one call

ReadResource · Grep

Reads project files from disk; cheap and safe

ManageScene · ManageGameObject

Live editor queries; heavier, used sparingly

PackageManager · AssetInventory

Installs packages, searches the asset catalog

MCP

File system server

Plans, specs and source: the agent's view of the project on disk.

MCP

Version control server · git

Stage, diff, commit, log. Pushing to the shared remote stays with the Designer.

MCP

Generative 3D · Meshy MCP

Text- or image-to-3D, retexture, remesh, rig and animate.

MCP

3D content · Blender MCP

Scripts Blender, and searches and downloads from Sketchfab and Poly Haven.

4 · SYSTEMS & SERVICES

where the work lands

SYSTEM

Game project · Unity 6, URP

Inferno engine layer

- Adventure + dialogue runtime
- Facial animation bridge
- Voice-over module
- Editor tools, templates

Game layer

- Rooms built from specs
- Dialogue database, scenes
- UI skins (prefab variants)
- Voice clips

Third-party packages

(adventure framework, dialogue system, lip sync) are read-only to the agent. Scene and prefab files change only through editor tools.

AI API

Voice synthesis · ElevenLabs

Text-to-speech REST API, called at edit time only, so the shipped Game makes no AI calls. The API key stays in a local secret store.

SYSTEM

Source repository · git

The agent commits locally; the Designer pushes to the shared remote.

SYSTEM

3D content tool · Blender

Where downloaded and generated models are cleaned up, scaled and exported before they enter the Game project.

5 · CLOUD SERVICES

Generative AI, asset libraries and documents

Everything generated or downloaded passes through Blender or the editor, and gets a licence-ledger row, before it ships in the Game.

DOCUMENTS

OneDrive / SharePoint

Decision log, licence ledger, design docs.

AI API

OpenArt

Concept art, textures and video from prompts.

AI API

Meshy

3D models from text or a concept image.

ASSET API

Sketchfab

3D model library; licence set per model.

ASSET API

Poly Haven

CC0 HDRIs, textures and models.

Licensing differs by source

Poly Haven is CC0. Sketchfab licences vary per model. Rights to AI output from OpenArt, Meshy and ElevenLabs depend on the plan tier at generation time.

ARCHITECTURE PRINCIPLES

MCP is the integration layer

Each tool exposes typed functions. Add a service by adding a server, not by changing the agent.

Plan and execution are separate

Spec files are the contract; the agent executes what was reviewed.

Least-privilege writes

Engine and game folders only; third-party code is read-only.

Secrets stay out of source

API keys live in a local secret store, never in the repository.

AI at edit time, not run time

Voice, art and models are baked into assets; the shipped Game makes no AI calls.

Provenance for every asset

Source, licence tier and attribution are logged before an asset ships.

One call at a time, fail fast

Tool calls are serialized; on a timeout the agent stops and reports.

Verify and audit

Every action is checked; commits and the decision log form the trail.

OUTPUT

Game

A point-and-click adventure built on Inferno. Engine modules are copied in unchanged; rooms, dialogue, art and voice belong to the Game.

Four workflows, step by step

Each step is tagged with the MCP server, connector or AI API it uses. Dotted cards are the Designer's checkpoints.

