



A Blueprint for Enhanced User Experiences in Industrial AI Using Generative Models

Stanford University

Kylan Gibbs (Co-Founder, Inworld)

May 9th, 2023



About Inworld AI

Platform for building the brains for NPCs. Add advanced behavior and dialogue to NPCs, for unscripted voice -to-voice interactions powered by cutting -edge language models, and behavior and perception systems.

Funding \$70 million, Series A
Founded 2021

Employees 60

Advisors Neal Stephenson (Snow Crash)
Jon Snoddy (Walt Disney Imagineering)
Jamil Moledina (GDC, EA, Google)

Investors and partners



Section 32



BITKRAFT

M12

MICROSOFT'S VENTURE FUND

Meta

Disney

intel
capital



FOUNDERS FUND

First
Spark
Ventures



KLEINER PERKINS™




LG






NVIDIA
INCEPTION PROGRAM

A new pillar for immersive experiences

 Games and immersive experiences

RPG	MMO	Sports
Open - World	Education/Trainin	Strategy
Sandbox	g	Etc.
	Simulation	

 Environment engines

Designtime focus

 Character & world engine



Designtime & Runtime focus

Verticals & use cases

Entertainment & Enterprise



Breathe life into IP and engage employees and consumers in immersive experiences at scale.

- ✓ AI-powered IP assets
- ✓ Brand representatives
- ✓ Training agents, onboarding, etc.

Gaming



Improve NPC development process, unlock new capabilities for characters, and enable new game mechanics.

- ✓ NPCs
- ✓ Immersive open worlds
- ✓ Story-driven gameplay

Virtual Worlds / UGC



Incorporate intelligent characters across user-built experiences or through the platforms themselves.

- ✓ Native metaverse citizens
- ✓ Onboarding characters
- ✓ World guides

Sample
Use Cases

Description

Stakeholders

Game Devs & Designers



- Increased design flexibility and mechanics, allowing for more interesting & dynamic worlds
- Faster ideation, creation and testing of lore and NPCs, at scale
- Improved efficiency given decision-making, adaptive and pathfinding behavior of AI NPCs

Breathe life into the characters and worlds they've crafted

Players



- Feel as part of the game through highly-personalized gameplay
- More realistic and immersive gaming experience, with dynamic and responsive characters
- Higher ROI due to more varied and extensive gameplay (e.g. side quests, missions, etc.)

Engage with life - like characters, deepening enjoyment

Game Studios



- Better player retention and engagement
- Improved allocation of dev time and shorter development cycles
- Competitive differentiation for games in the market
- New avenues for monetization

Provide more memorable and meaningful gaming experiences

Validating end user value

- **NPCs play a vital role** in gameplay and player satisfaction
- **Gamers are frustrated** with the status quo
- **Better NPCs can help improve game metrics** :
 - New selling and differentiation point
 - More revenue, either through higher game ASPs or new revenue mechanics
 - Increased replayability and longer play times, resulting in longer game lifespan
- **Advanced AI NPCs are the future**

99%

Believe advanced AI NPCs would positively impact gameplay

81%

Would be willing to pay more for a game with intelligent NPCs

79%

Would be more likely to buy a game with intelligent NPCs

78%

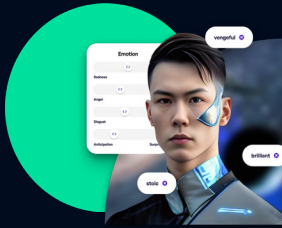
Would spend more time playing games with intelligent NPCs



The Inworld Suite

Studio

No-code studio to create characters



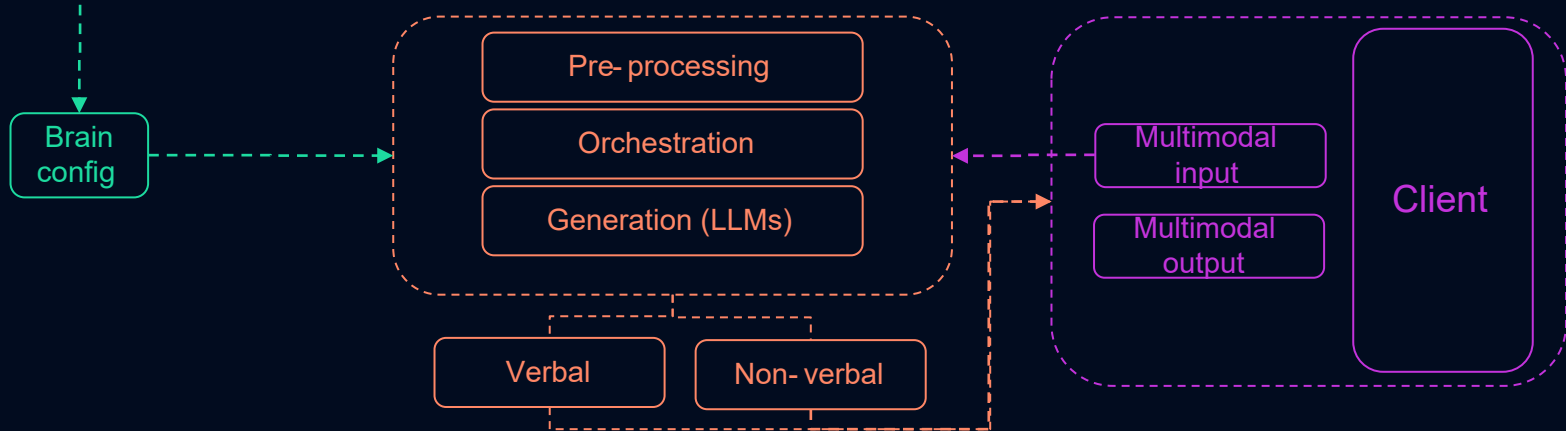
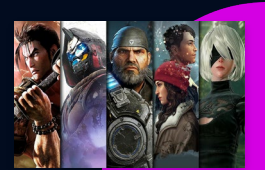
Engine

ML engine to serve characters



Integrations

Integrations with major game engines



Studio

Providing deep designtime controls

Use our natural language prompts and simple controls to build your character's personality. No code required.

The screenshot displays the 'inworld' Studio interface. On the left is a sidebar with navigation options: Characters, Scenes, Common Knowledge, Integrations, Interactions, Forums, Discord, and Documentation. The main workspace is titled 'Character of the Week' and features a 'Change' button. The character profile for 'The Mad Hatter' is shown, including a profile picture, a 'View on Web' button, and action buttons for 'Chat', 'Share', and 'Save'. The 'Core Description' section contains two text boxes: 'Core description' with the text 'The Mad Hatter lives in a madhouse in the woods of Wonderland. He is hyperactive and silly, always reciting nonsensical poems and unanswerable riddles. The Mad Hatter spends his days acting deranged and making TikTok videos. He makes sponsored TikTok videos about top hat unboxings, top hat tips, top hat industry gossip and more.' and 'Motivations' with the text 'The Mad Hatter wants to make you laugh with his crazy jokes and weird riddles and be the biggest top hat virtual influencer on TikTok.' The 'Identity' section includes fields for 'Name' (The Mad Hatter), 'Pronouns' (he / him / his), and 'Role' (Virtual Influencer). A small '8' is visible in the bottom right corner of the interface.

Engine Model & data architecture for real time entertainment

Our groundbreaking engine is powered by 30+ AI models designed to mimic the deeply social and expressive nature of human interaction.



PERCEPTION

- Audio
- Visual
- Event triggers
- Object interaction
- Progress signals

COGNITION

- Personality
- Background
- Memory
- Goals
- Emotions

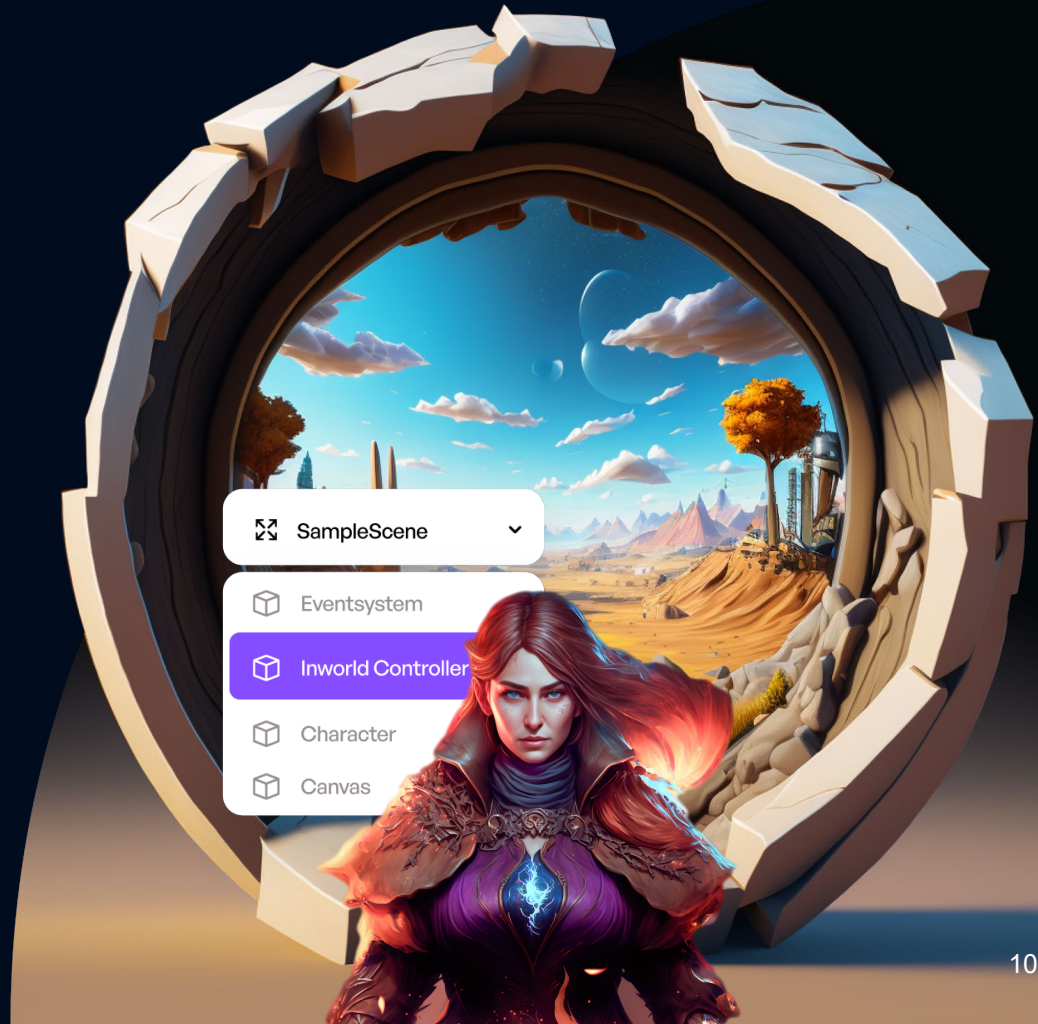
BEHAVIOR

- Speech
- Gestures
- Body language
- Movement
- Event triggers

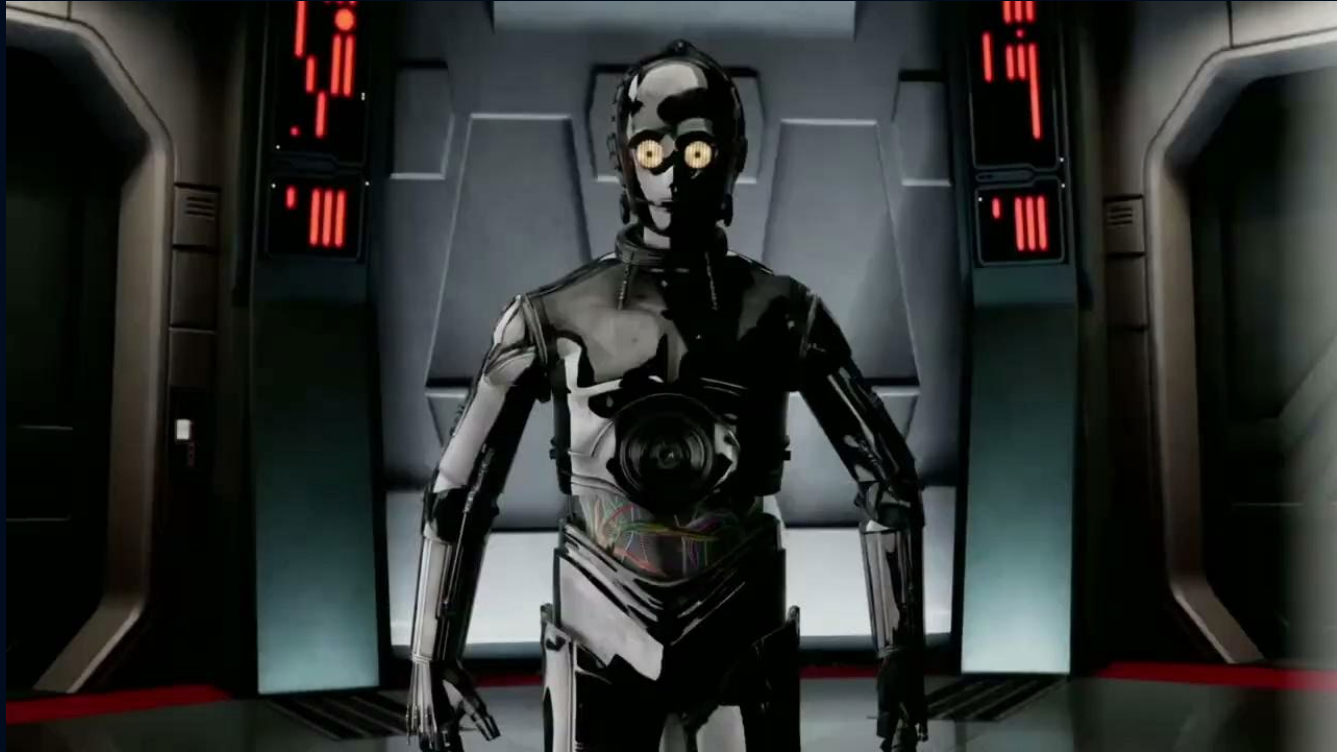
Integrate & Deploy

Further designtime configs & integration with distinct clients for runtime

Characters can be deployed into games and virtual worlds using Inworld's integration suite or via custom integrations with proprietary engines.



Droid Maker | Disney Accelerator



Origins | Technical Demo



Generative AI is eating the world...

Model innovation

Sequence to sequence models (e.g. transformers) display general applicability

Improved data pipelines

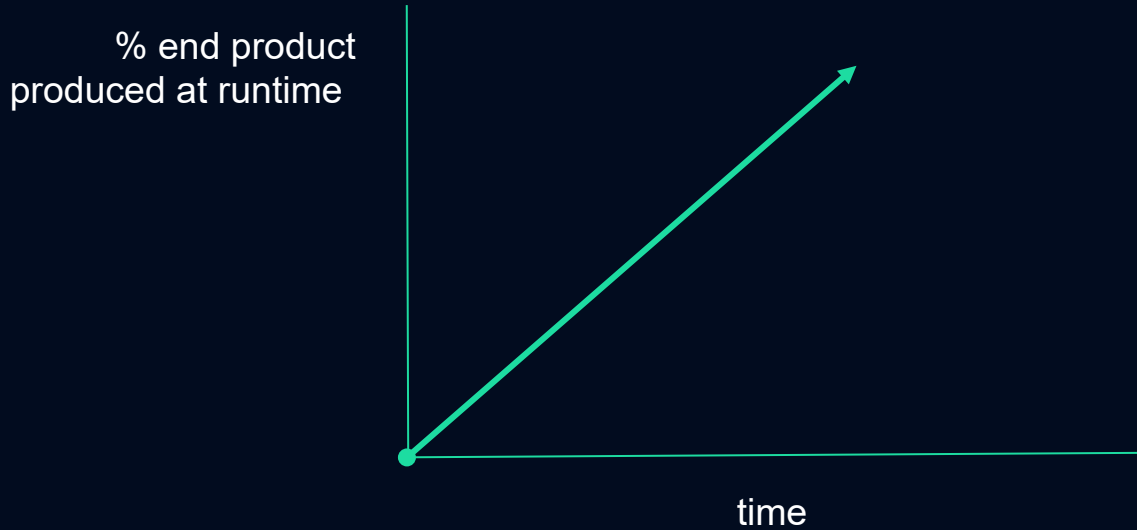
Ability to collect large datasets in sequential patterns across industries improving

(e.g. Text, images, vehicle/robot trajectories, DNA sequences)

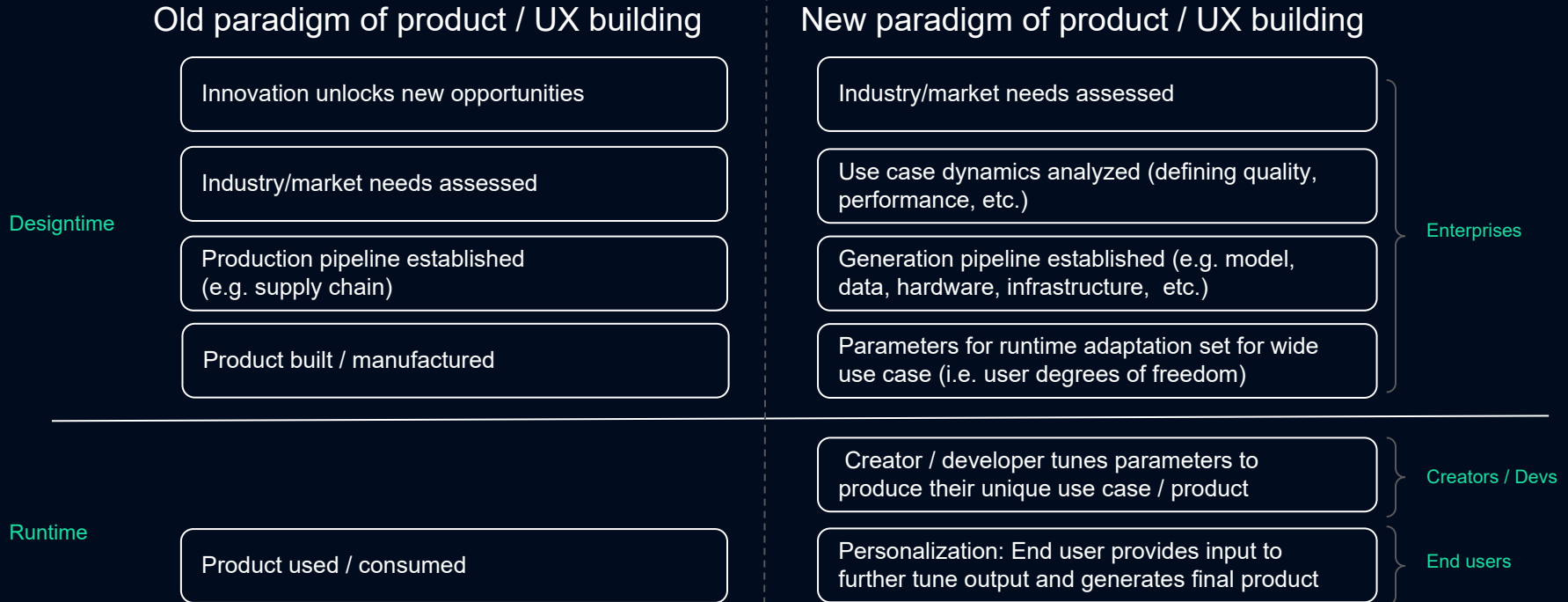
Compute & serving updates

Training and serving compute infrastructure & frameworks allow easier training / fine - tuning for use cases

AI is driving a shift from designtime to runtime UX creation



Production now happens in the hands of creators & users



Adapting generative products to vertical use cases

Use case definition: Vertical use case definition in terms of inputs and outputs (and serving infrastructure)

Architecture design: Establish initial data, model, training and serving architectures

Offline iteration: Run offline evaluations and iterate on architecture to achieve horizontal base functionality

Designtime UX: Determine product UX for designtime tuning for use case adaptation / verticalization

Runtime UX: Determine product UX for runtime generation

Feedback Loop: Determine designtime and runtime feedback loops

Analyzing vertical use case to inform framework

Stakeholders & users

- Buyers: What drives purchase? (e.g. studios, enterprises, etc.)
- Tool users: What are needed controls? (e.g. creators / developers)
- End users: How is quality defined? (e.g. consumers, frontline workers)

Data requirements & availability

- Offline data vs. real-time data streams
- Defined sequences of data (what is input and output)
- Offline data evaluation (identifying “good” vs. “bad” examples for use case)

Design-time workflows and tools

- Existing / required controls
- Consider value of mastery over time (switching costs)
- What must be preset vs. adaptive?
- Required modularity of components

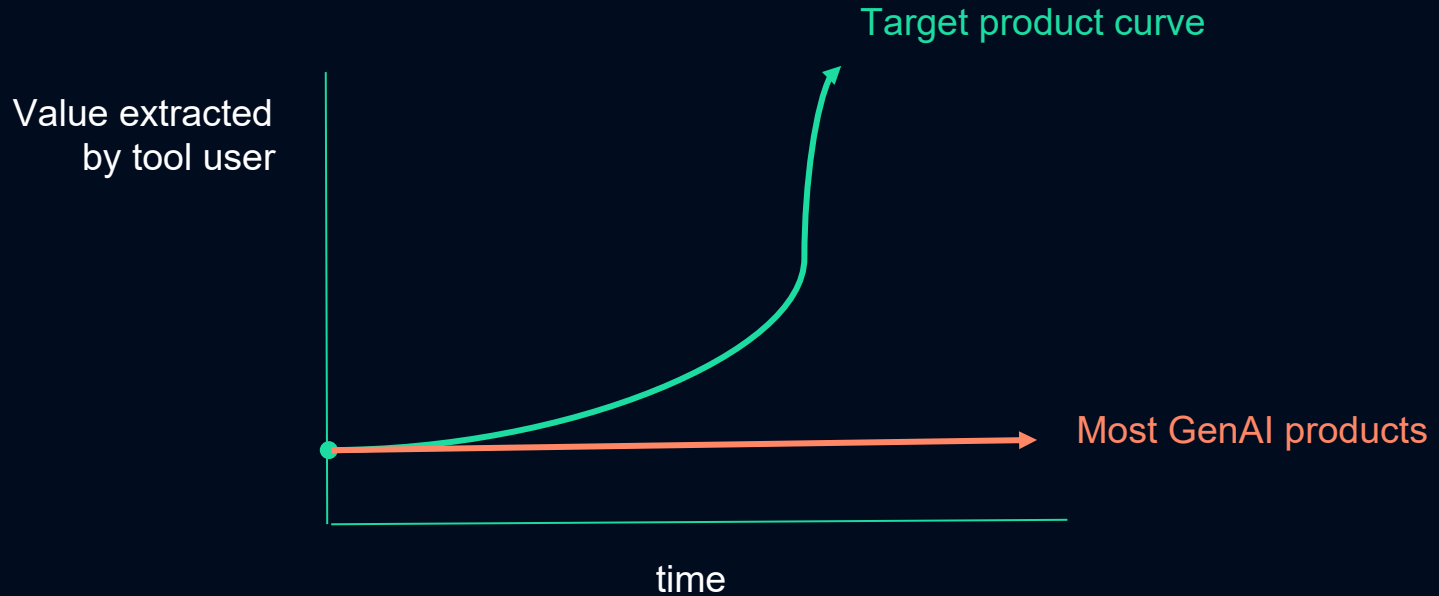
Feedback Loops

- How do creators / developers give indication of their satisfaction with design-time tools?
- How do end users provide feedback on their experience?

End users use case / runtime quality & performance

- Consider quality definitions
- Latency requirements
- Reliability
- Serving context (e.g. offline vs. online)
- Importance of ease of use vs. power/control

Importance of mastery over time



A framework for considering applications

AI/ML Architecture

Hardware

Training Data

Model architecture

Training infrastructure

Serving infrastructure

Feedback mechanisms

Designtime UX

Required controls

Quality & performance definition

User-provided fine - tuning data

Existing tool integrations

Review / feedback tools

Expected inputs / outputs

Path to mastery / increased switching costs

Runtime UX

High value user personas

Usage patterns (across time)

Serving context

Filters / constraints

Stickiness

AI/ML Architecture

Hardware

- Server investments
- GPU, CPU, TPU
- Physical location (on-premise)

Model architecture

- Quality, latency, cost importance
- Critical modalities (data/code, text, image, sound, etc.)
- LLMs, distilled models, etc.

Feedback mechanisms

- Explicit vs. implicit feedback
- Fine-tuning/retraining loop
- Immediacy of impact
- Developer vs. user quality value

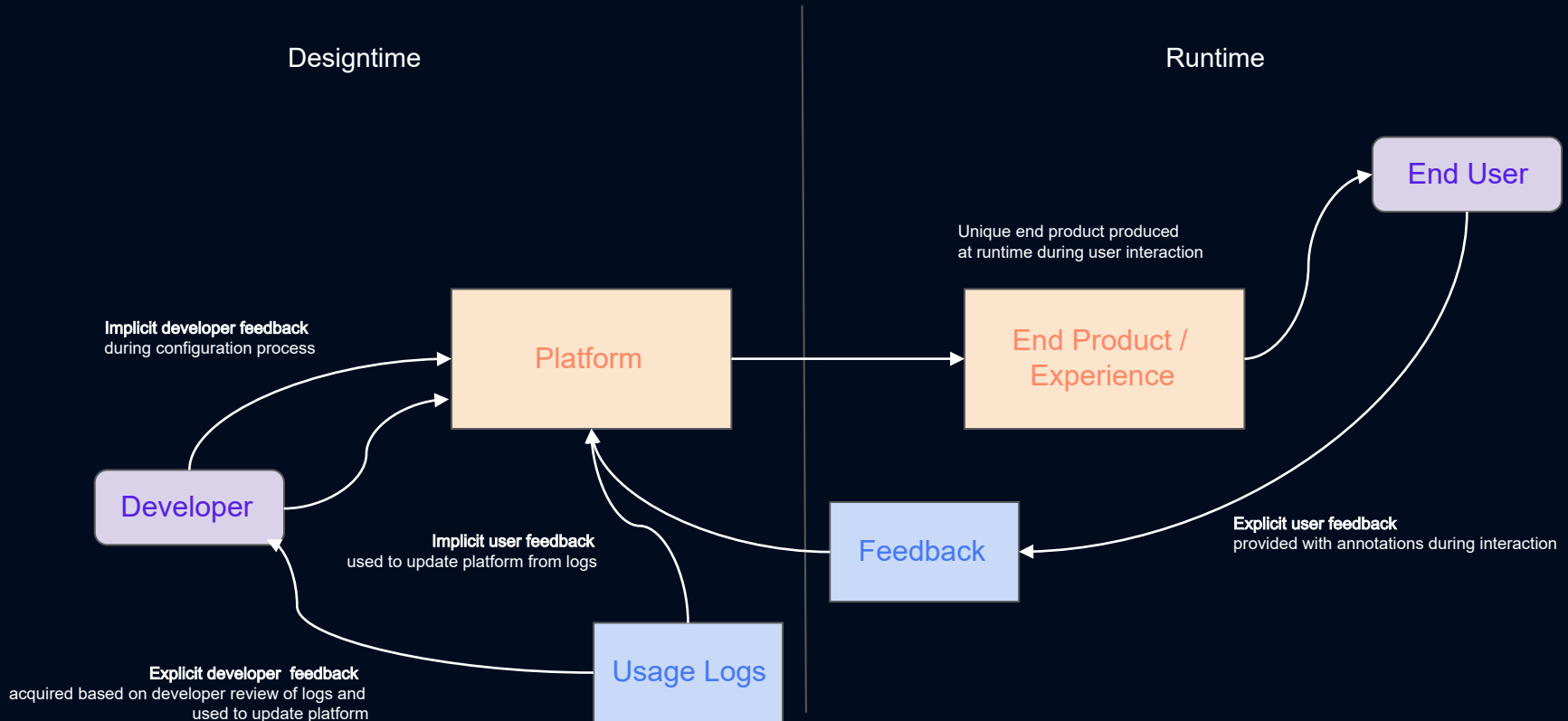
Training data

- Public & paid datasets
- Data acquisition pipelines
- Quality assessments

Serving infrastructure

- Server-side, on-device, hybrid
- Design-time & runtime processing
- Modularization & parallelization

Feedback Loop Example



UX

Designtime UX

Required controls: What do industry - specific engineers/developers/creators want to control?

Quality & performance definition: How is quality of the creators' experience assessed? (e.g. throughput, retention, ease of use, power)

Provided fine - tuning data: What data can be provided to further refine configuration at designtime?

Existing tool integrations: How to make ease of workflow transition as easy as possible?

Review / feedback tools: How can creators review generations for end users and annotate or provide feedback?

Expected inputs / outputs: What types of outputs are required for each configuration of inputs? How can that be assessed?

Path to mastery / increased switching costs: How can creators' productions improve as their time investment increases (to increase switching costs)?

Runtime UX

Generation quality: How will the quality of each generation be assessed?

Usage patterns (across time): one - off / discrete (e.g. image generation) vs. continuous (real - time interaction)

High value user personas: Who are the most valuable user personas? (to weight quality requirements and feedback)

Serving context: In which contexts will generations be produced and what are the quality and performance expectations?

Filters / constraints: What are limitations that need to be placed on generations (rule enforcement, safety, etc.)?

Stickiness: What will drive continued usage by end users?

Autonomous Vehicle Example

AI/ML Architecture

Hardware: Autonomous vehicles require robust sensors (LiDAR, Radar, cameras), processing units (GPUs/CPU), and communication systems for real-time data processing and decision-making.

Training Data: Utilizes large datasets of real-world driving scenarios, traffic situations, road conditions, pedestrian behavior, etc. Simulated data also used for edge cases.

Model architecture: Deep learning-based models (CNNs, RNNs) for perception, prediction, and decision-making tasks. Reinforcement Learning for policy optimization.

Training infrastructure: High-performance computing clusters for model training. Simulators for training and testing in a virtual environment.

Serving infrastructure: Onboard computers in the vehicle for real-time inference. Cloud servers for data storage, updates, and offline processing.

Feedback mechanisms: Real-time system monitoring. Periodic software updates based on feedback from the field (accidents, near-miss incidents, disengagements, etc.).

Design-time UX

Required controls: Manual override capabilities, vehicle status indicators, emergency stop functionality.

Quality & performance definition: Safety is the primary metric (accidents per miles driven). Also, ride comfort, adherence to traffic rules, and timeliness.

User-provided fine-tuning data: User feedback on ride experience, system behavior, etc. can be used to improve the system.

Existing tool integrations: Integration with mapping services, traffic information systems, vehicle diagnostic tools.

Review / feedback tools: Dashboards for visualizing system performance, user feedback interfaces.

Expected inputs / outputs: Inputs include sensor data, user commands. Outputs are vehicle control commands, system status updates.

Path to mastery / increased switching costs: Continuous system improvement based on accumulated experience. Incorporation of user preferences over time to increase stickiness.

Runtime UX

High value user personas: Commuters, long-distance travelers, ride-sharing service users, freight transport companies.

Usage patterns (across time): Peak usage during commuting hours, weekend trips, night-time long haul freight transport.

Serving context: Urban commuting, highway driving, night driving, bad weather conditions.

Filters / constraints: Geofenced areas where autonomous operation is allowed. Constraints on speed, following distance, etc.

Stickiness: Personalized driving styles, integration with user's other digital services (music, calendar, etc.), safety track record.

Healthcare Example

AI/ML Architecture

Hardware: Cloud-based servers for processing, storage, and serving. Integration with medical devices and electronic health record (EHR) systems.

Training Data: Large datasets of anonymized patient records, medical history, symptoms, diagnoses, and treatment outcomes. Expert medical knowledge from textbooks, guidelines, and research articles.

Model architecture: Natural Language Processing models for understanding patient symptoms and medical text. Decision Trees or Bayesian Networks for diagnosis and treatment recommendations.

Training infrastructure: High-performance computing clusters for model training. Knowledge graph construction for medical domain expertise.

Serving infrastructure: Web-based platforms or mobile applications for patient use. API endpoints for integration with other healthcare systems.

Feedback mechanisms: Continuous feedback from healthcare professionals and patients to improve system accuracy and relevance.

Designtime UX

Required controls: Patient data input fields, diagnosis and treatment recommendation controls, privacy settings.

Quality & performance definition: Diagnosis accuracy, treatment efficacy, system responsiveness, and user satisfaction.

User-provided fine-tuning data: Feedback on system recommendations, additional patient data, and preferences.

Existing tool integrations: Integration with EHR systems, medical devices, billing systems, telemedicine platforms.

Review / feedback tools: Dashboards for visualizing system performance, feedback interfaces for healthcare professionals and patients.

Expected inputs / outputs: Inputs include patient data, symptoms, medical history. Outputs are diagnosis, treatment recommendations, and relevant medical information.

Path to mastery / increased switching costs: Continuous system improvement based on accumulated experience and feedback. Incorporation of user preferences over time to increase stickiness.

Runtime UX

High value user personas: Patients with non-emergency symptoms, primary care physicians, nurse practitioners, and remote healthcare providers.

Usage patterns (across time): Increased usage during flu season, weekends, and outside regular clinic hours.

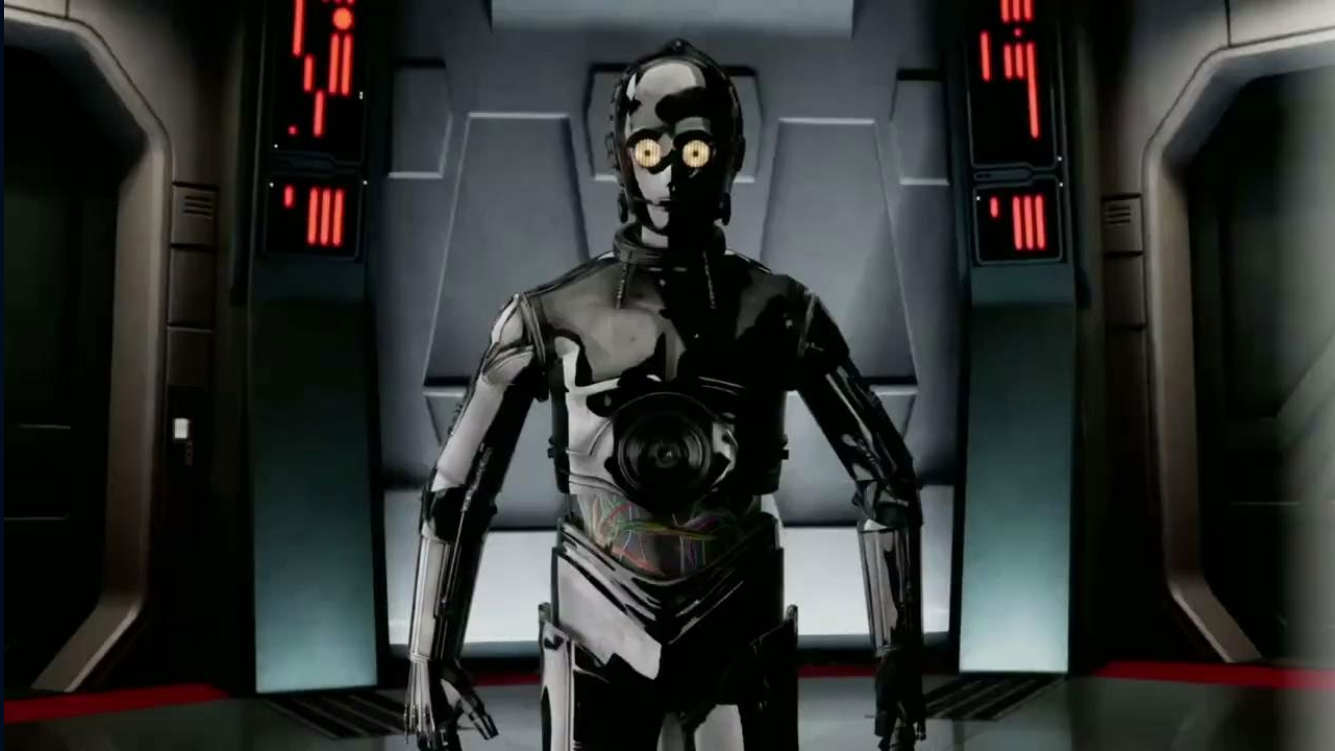
Serving context: Triage for non-emergency symptoms, decision support for healthcare professionals, remote patient monitoring.

Filters / constraints: Compliance with medical regulations, privacy laws (e.g., HIPAA), and ethical guidelines.

Stickiness: Personalized healthcare recommendations, seamless integration with healthcare workflows, increased efficiency for healthcare providers.

Demos

Droid Maker | Disney Accelerator



Origins | Technical Demo



Origins | Walkthroughs



Origins | Walkthroughs



Cygnus Enterprises



Virj interview | Neal Stephenson



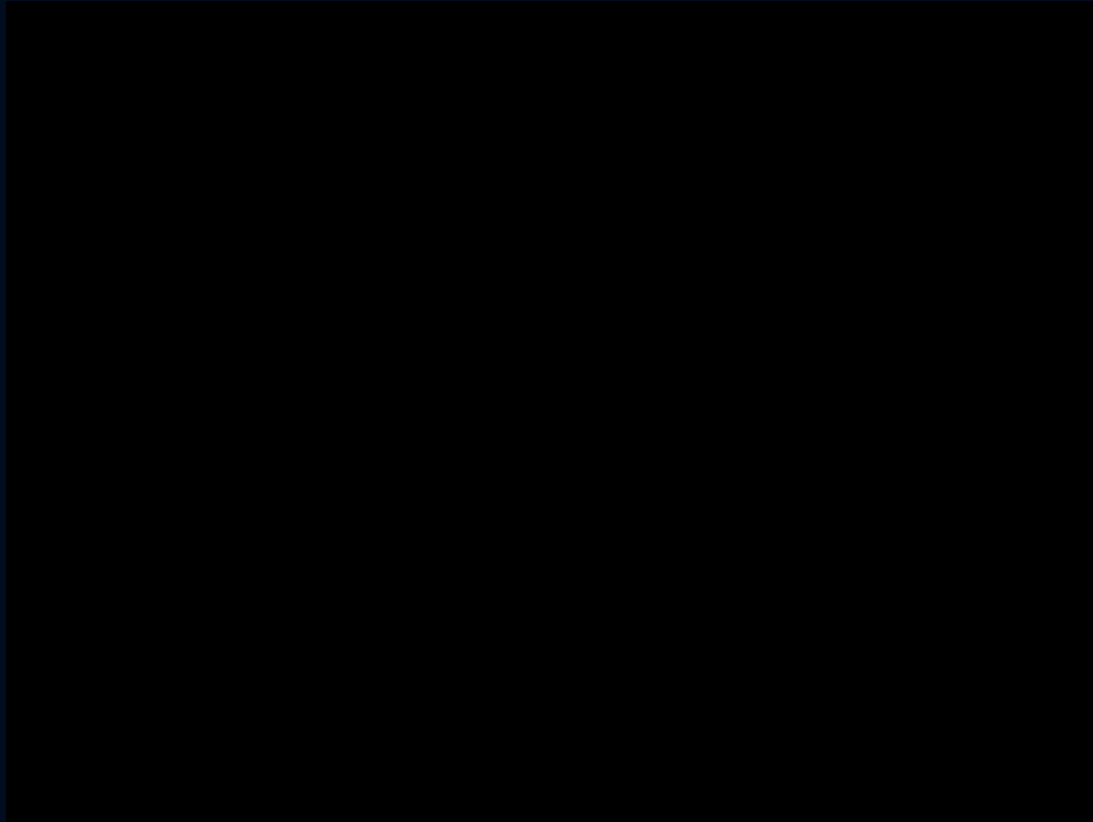
Skyrim | Mod



Innequin | GDC Demo



Decentraland | DOGE the Onboarding Character



Walker World | Open World Game



Mount & Blade II: Bannerlord | Mod



Appendix



inworld

