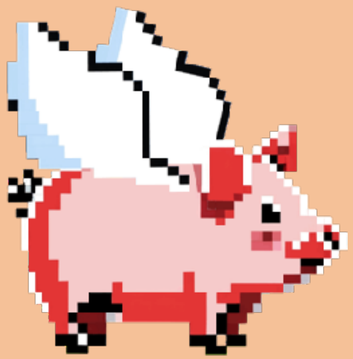
“chili”

“chili”

Input		Context		Output
	+	 Cincinnati	=	
	+	 Anywhere else	=	

Non-determinism

the same input can produce different outputs

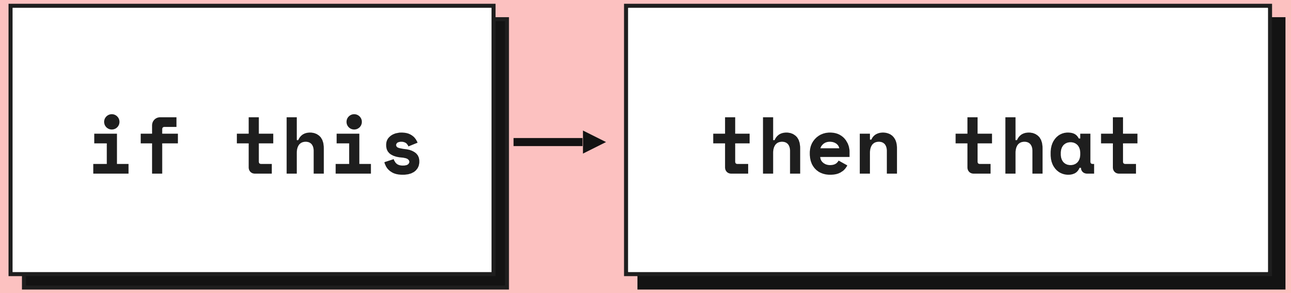


Large Language Models are

probabilistic

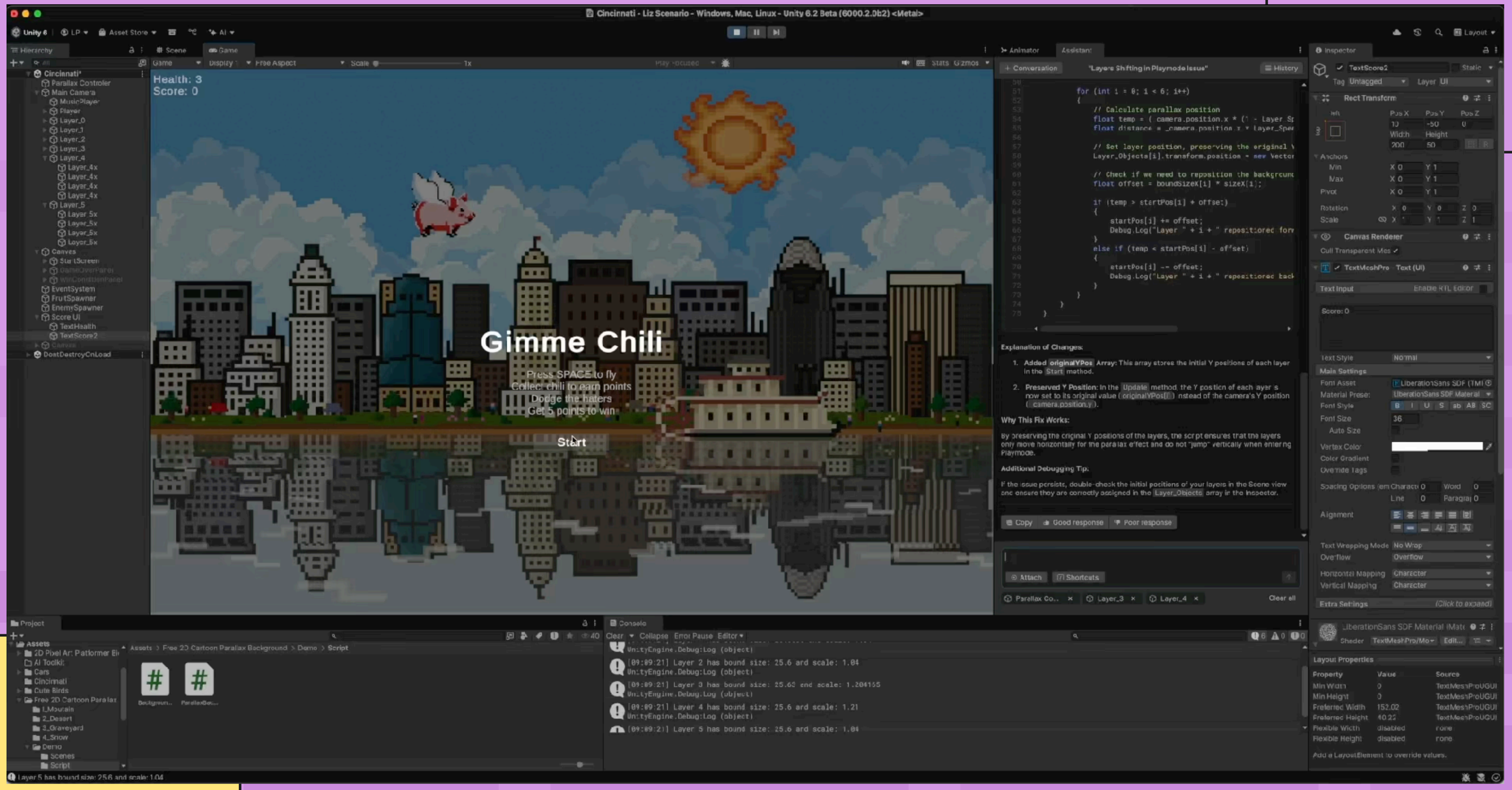
Deterministic

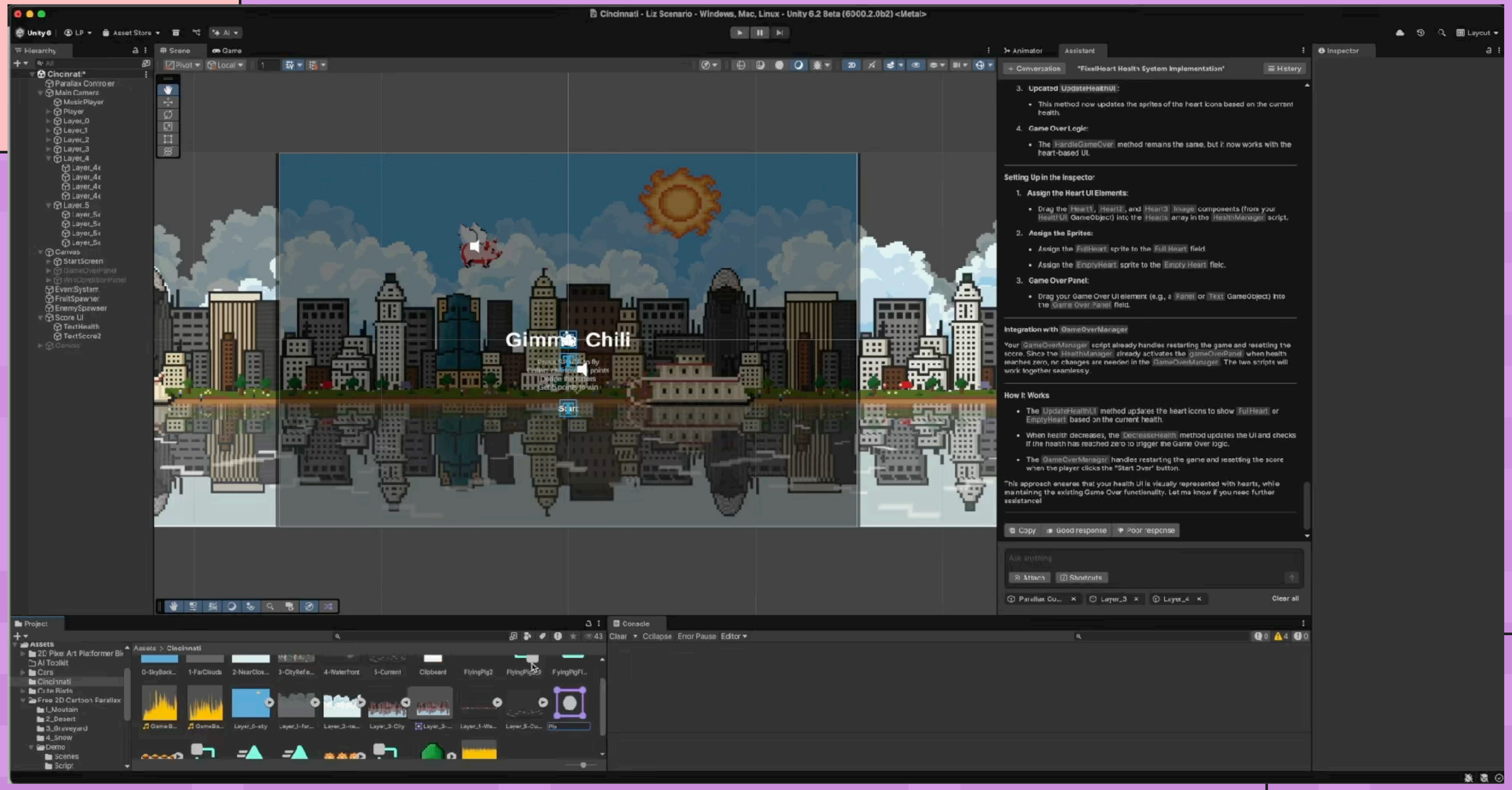
systems have linear flows
& controlled outcomes



Embracing Uncertainty

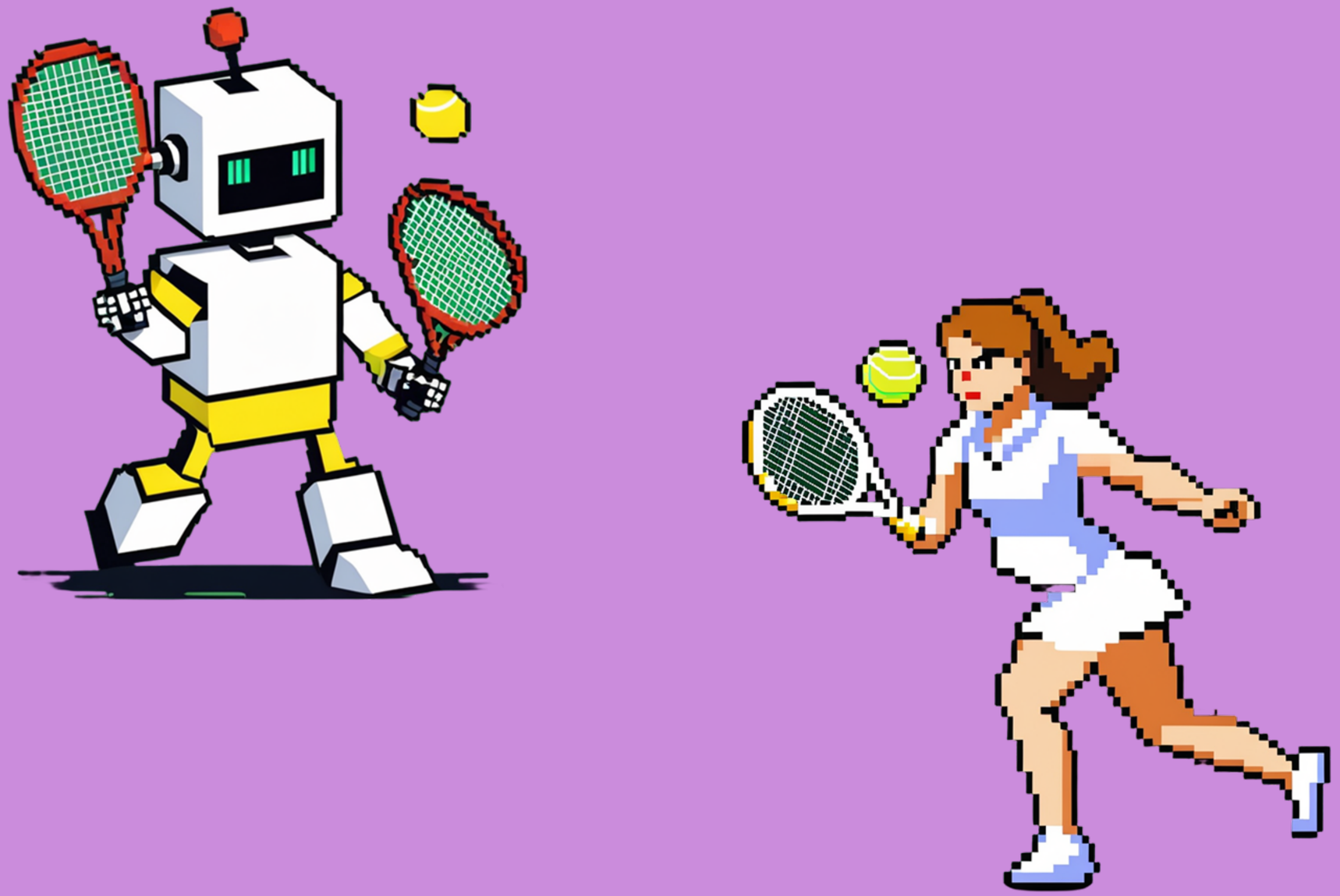
when designing for AI


Cincinnati - Liz Scenario - Windows, Mac, Linux - Unity 6.2 Beta (6000.2.0b2) <Metal>
Health: 3
Score: 0
Gimme Chili
Press SPACE to fly
Collect chili to earn points
Dodge the haters
Get 5 points to win
Start
Explanation of Changes:
Why This Fix Works:
Additional Debugging Tip:


Cincinnati - Liz Scenario - Windows, Mac, Linux - Unity 6.2 Beta (6000.2.0b2) <Metal>
Hierarchy
Scene
Game
Animator
Assistant
Inspector
Project
Console
Gimme Chili
Start
3. Updated UpdateHealthUI:
This method now updates the sprites of the heart icons based on the current health.
4. Game Over Logic:
The HandleGameOver method remains the same, but it now works with the heart-based UI.
Setting Up in the Inspector
1. Assign the Heart UI Elements:
Drag the Heart1, Heart2, and Heart3 Image components (from your HealthUI GameObject) into the Hearts array in the HealthManager script.
2. Assign the Sprites:
Assign the FullHeart sprite to the Full Heart field.
Assign the EmptyHeart sprite to the Empty Heart field.
3. Game Over Panel:
Drag your Game Over UI element (e.g., a Panel or Text GameObject) into the Game Over Panel field.
Integration with GameOverManager
Your GameOverManager script already handles restarting the game and resetting the score. Since the HealthManager already activates the gameOverPanel when health reaches zero, no changes are needed in the GameOverManager. The two scripts will work together seamlessly.
How It Works
The UpdateHealthUI method updates the heart icons to show FullHeart or EmptyHeart based on the current health.
When health decreases, the DecreaseHealth method updates the UI and checks if the health has reached zero to trigger the Game Over logic.
The GameOverManager handles restarting the game and resetting the score when the player clicks the "Start Over" button.
This approach ensures that your health UI is visually represented with hearts, while maintaining the existing Game Over functionality. Let me know if you need further assistance!
Copy
Good response
Poor response
Ask anything
Attach
Shortcuts
Clear all

AI is an incredible technology, but great tools require work at the intersection of technology, design, and understanding people and the world.

-Sam Altman, OpenAI



fixed flows with
predictable outcomes

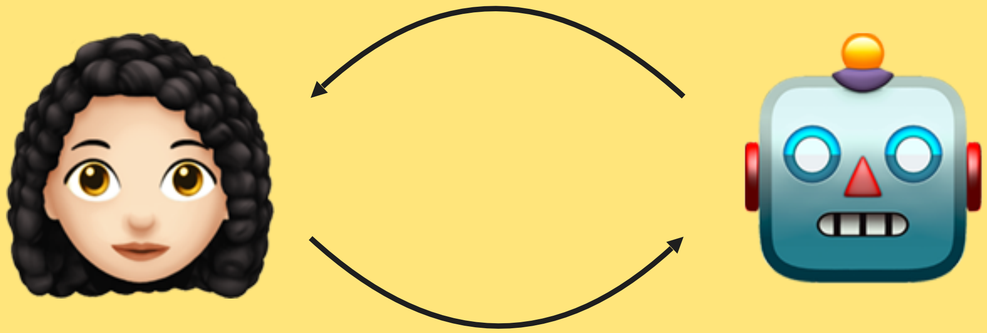
↓

iterative systems that
suggest, generate, adapt

2 challenges within the human-computer interaction

Mental models don't match

- impossible expectations
- don't anticipate a learning curve

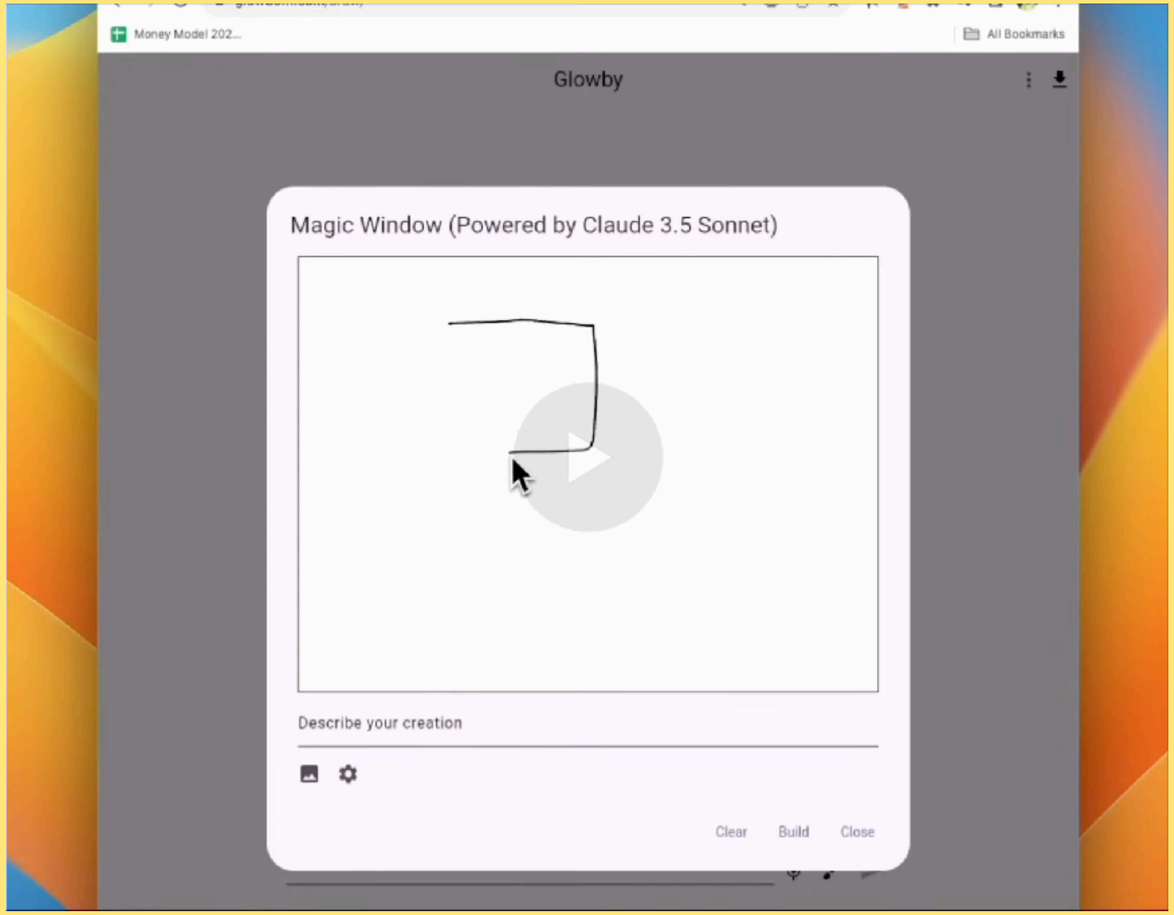


Non-deterministic output creates curveballs

- needs the *right* context
- and the right *amount* of context

When users expect magic, even good results can feel like failure.

When users expect magic, even good results can feel like failure.



Too subjective * too vague * too complex

Could you reorganize our level to have a better game experience?

Can you do it yourself please?

Improve my lighting

Fix it

Do it

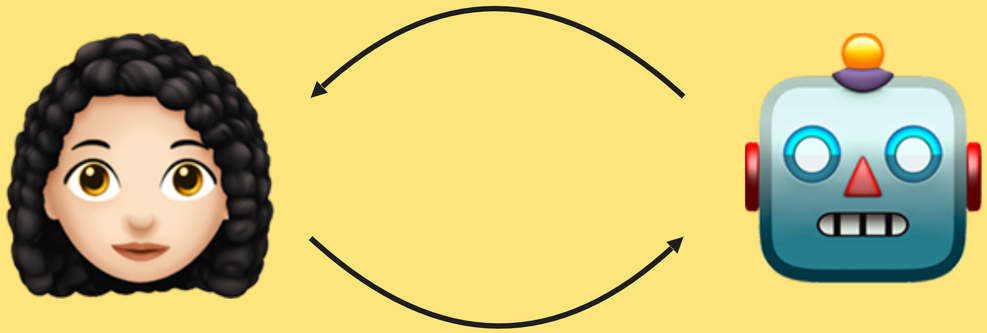
Add a spawning system to my game

Generate a 3D landscape

2 challenges within the human-computer interaction

Mental models don't match

- impossible expectations
- don't anticipate a learning curve



Non-deterministic output creates curveballs

- needs the *right* context
- and the right *amount* of context

The AI designer's toolbox

Research

Deeply understand your users' mental models

UX

Help users recover from imperfect results

UI

Design for ambiguity

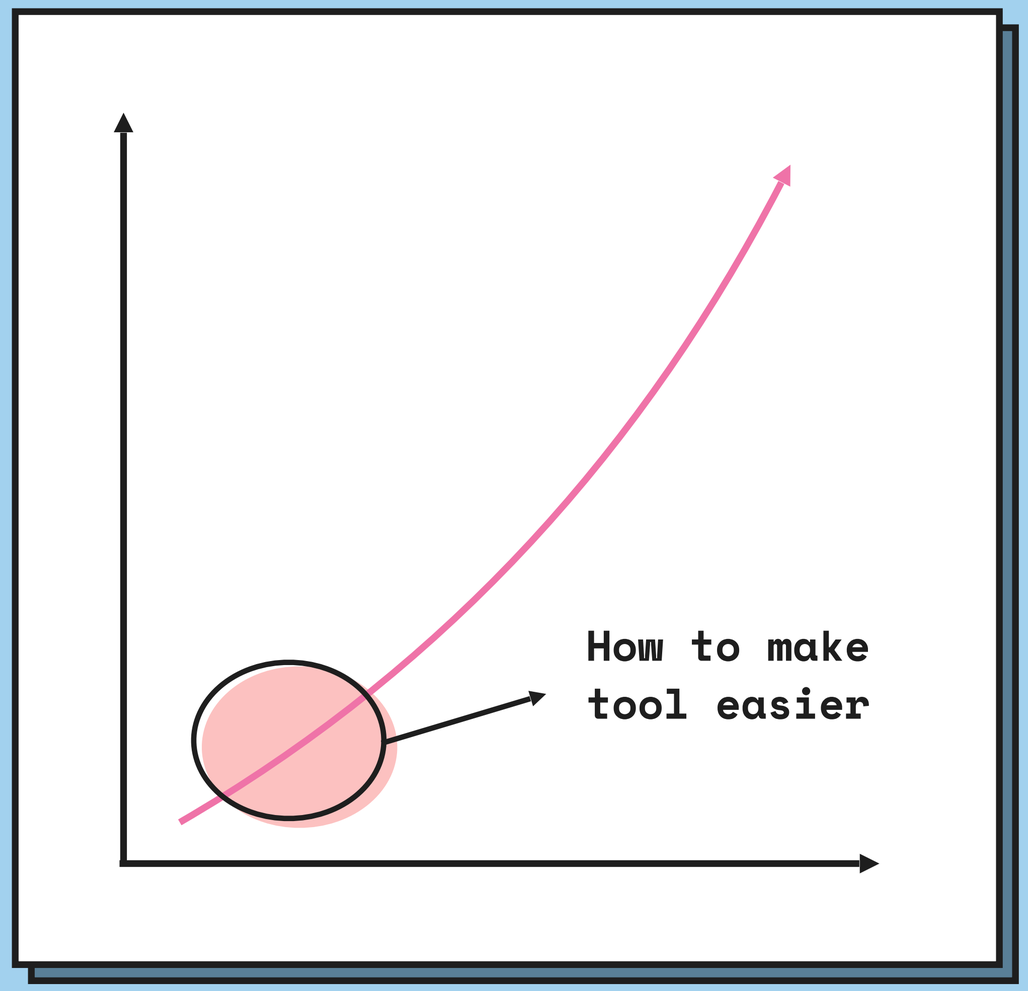
Implementation

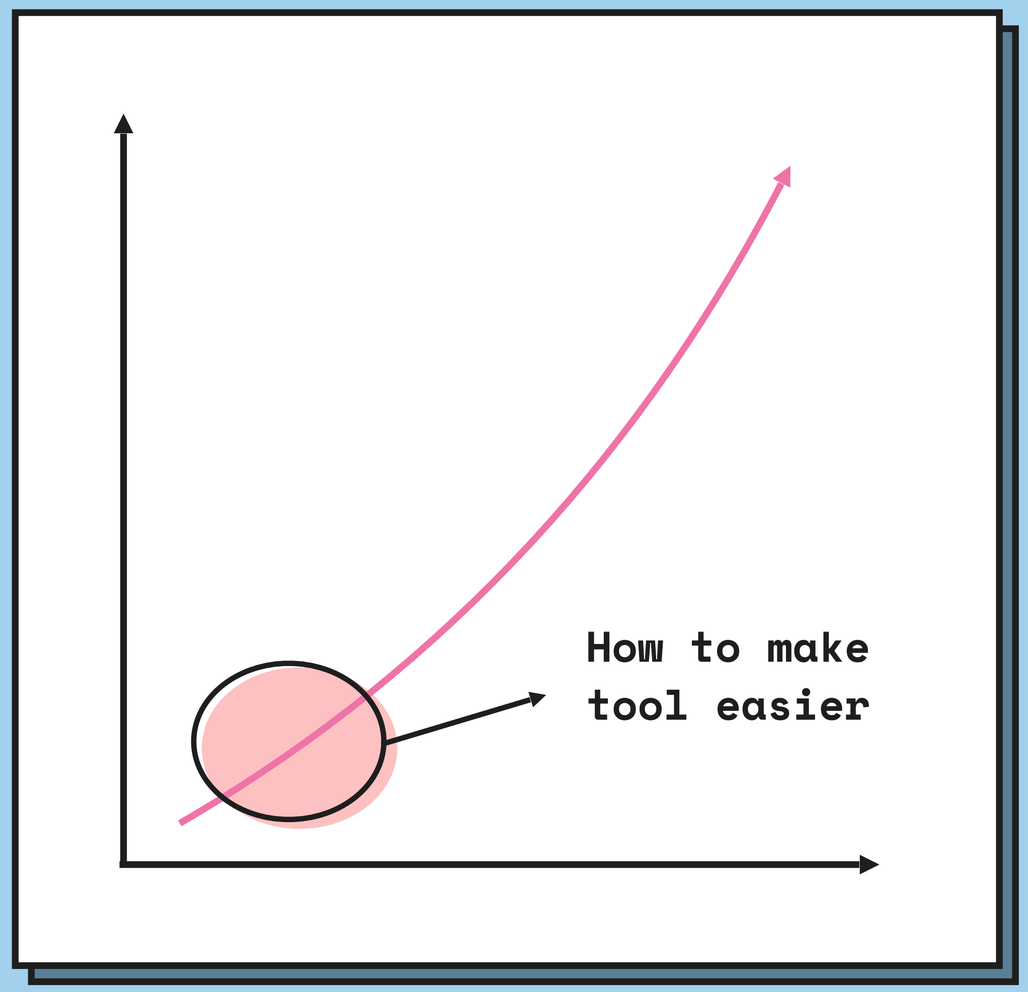
Find opportunity in the back-end

Research | UX | UI | Implementation

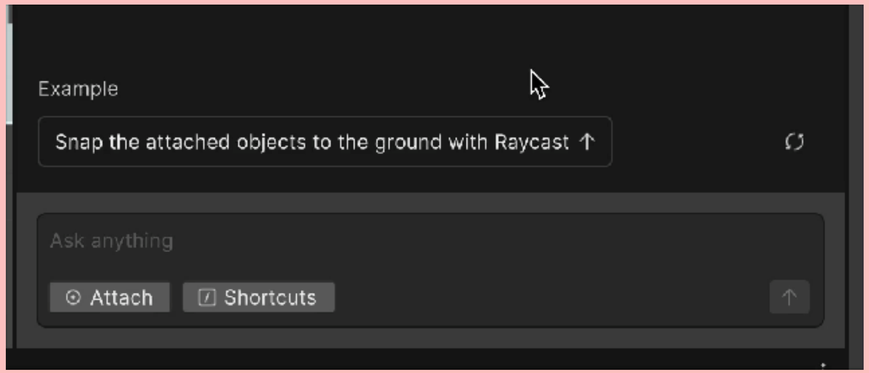
Deeply understand your users' mental models

Start with dogfood

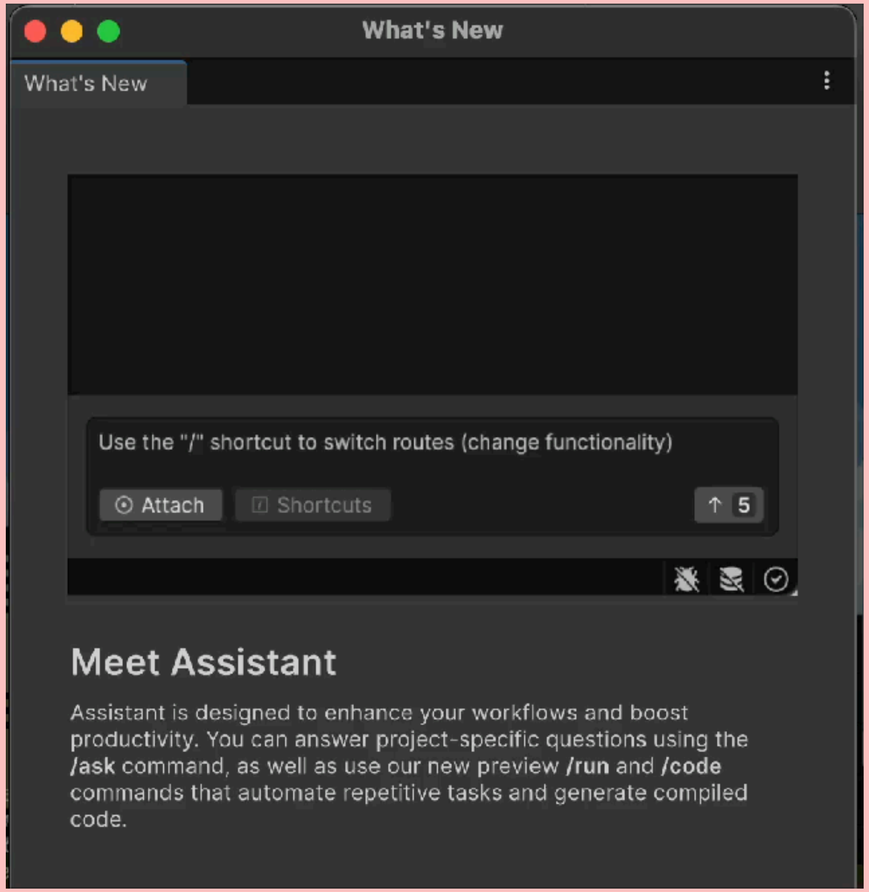

How to make
tool easier

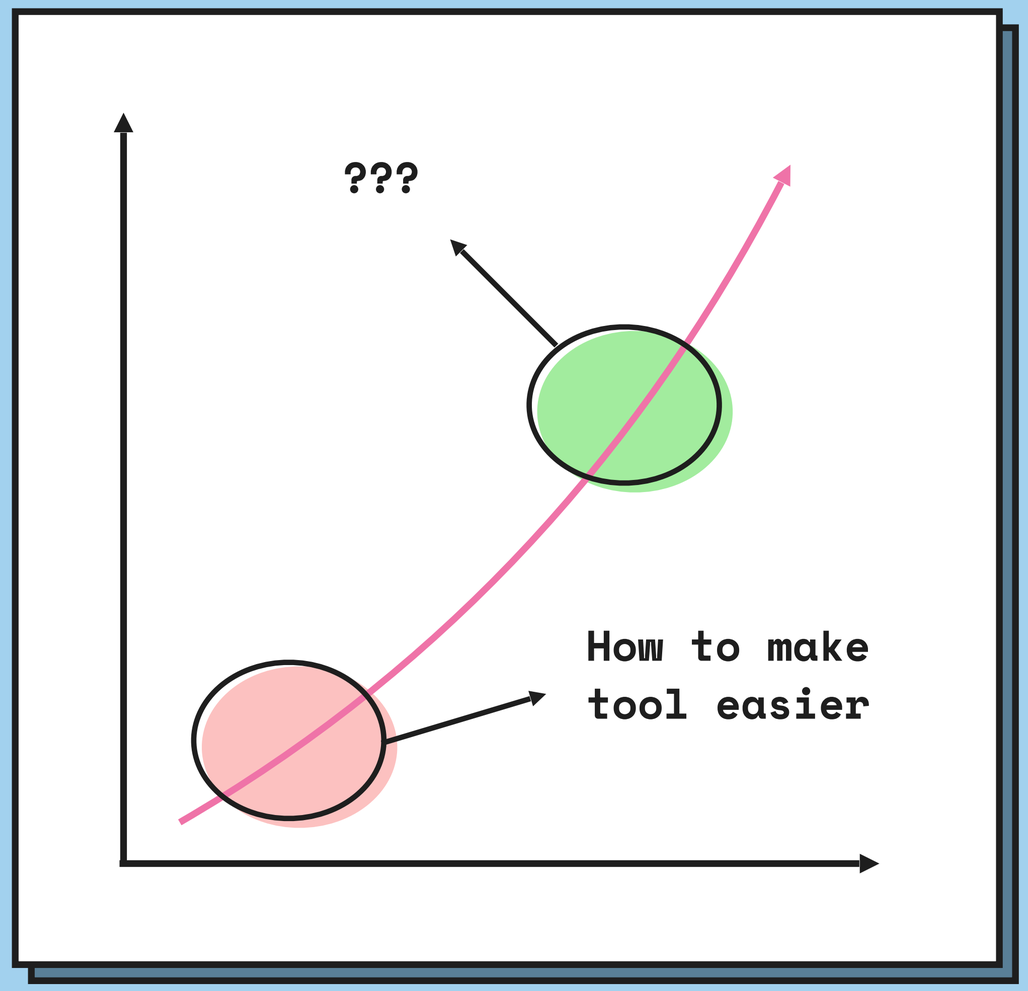


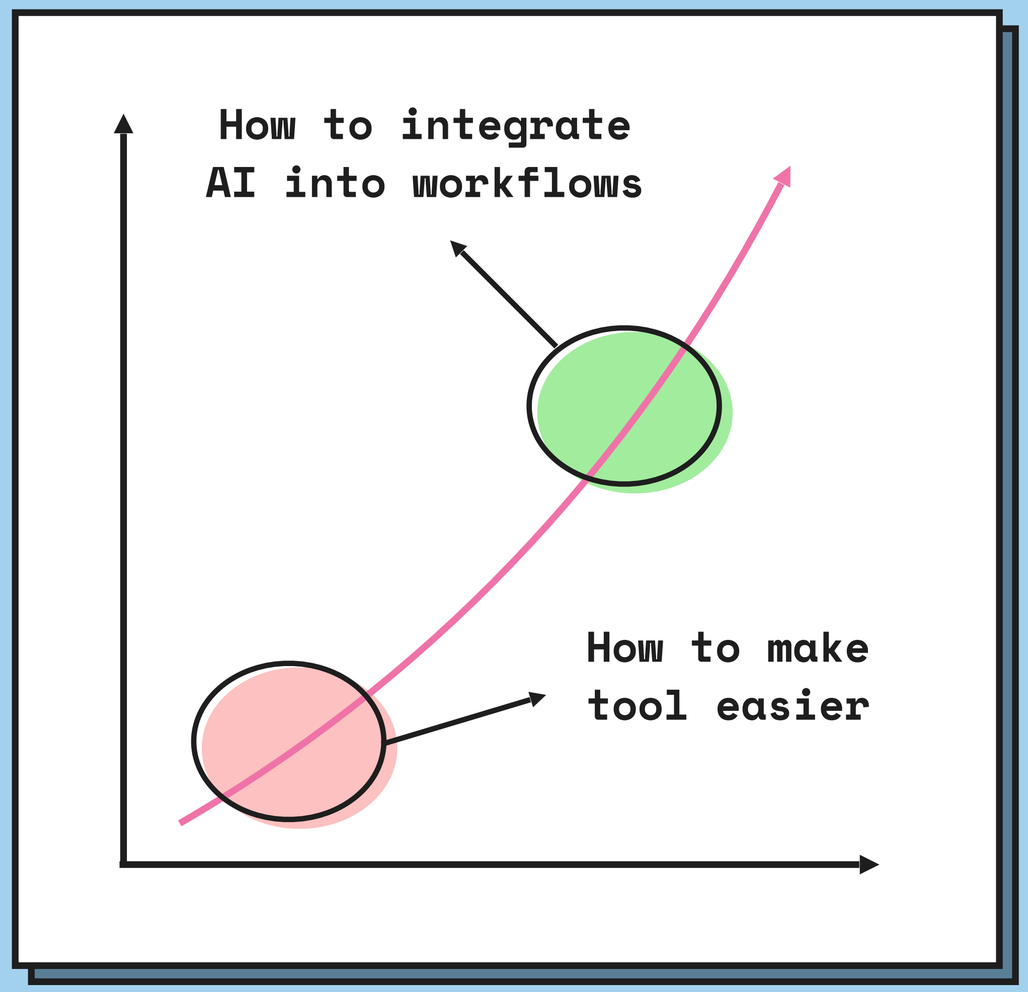
Inspiration
show realistic
examples



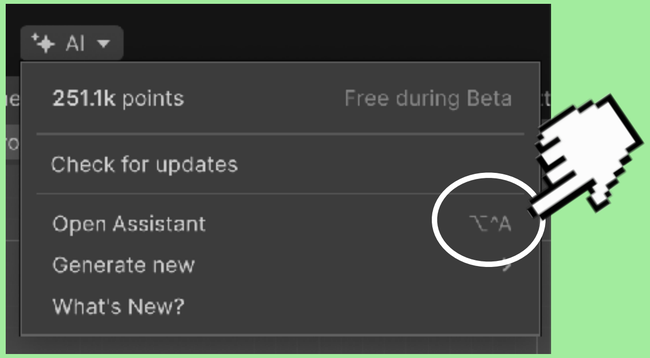
Tutorials
teach valuable
use-cases



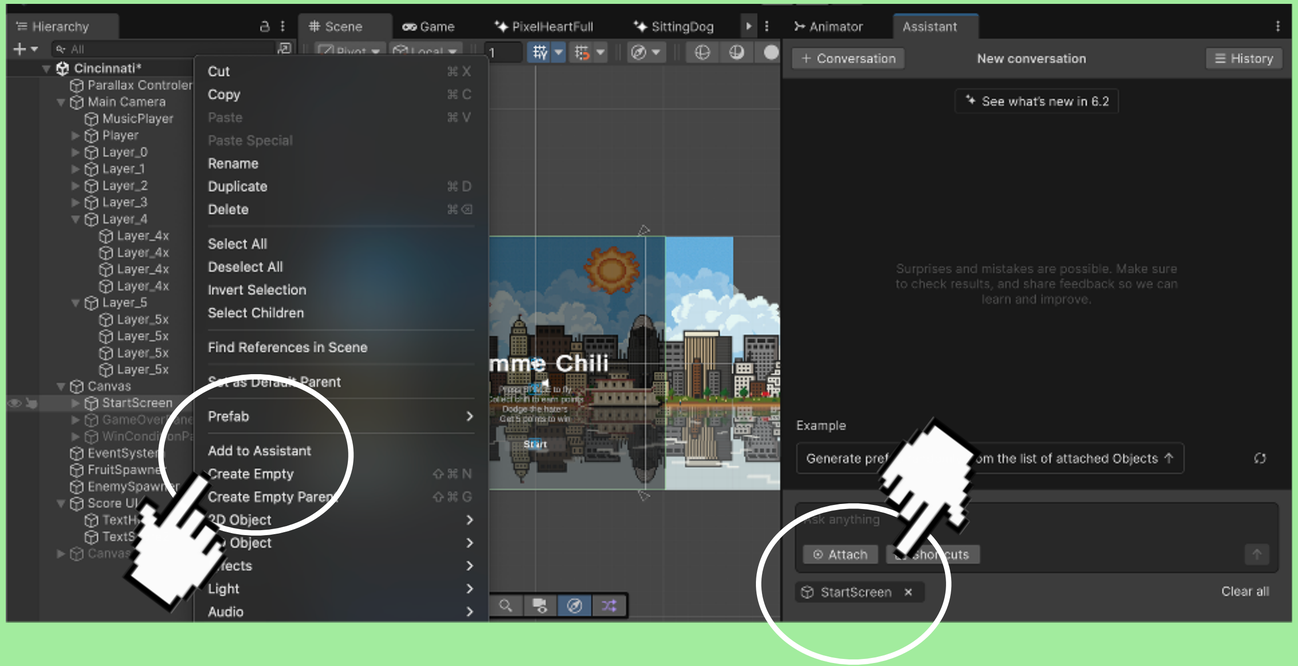

???
How to make
tool easier



Keyboard hot-key
for fast access



Right-click shortcuts
to quickly add context



Research | UX | UI | Implementation

Deeply understand your users' mental models

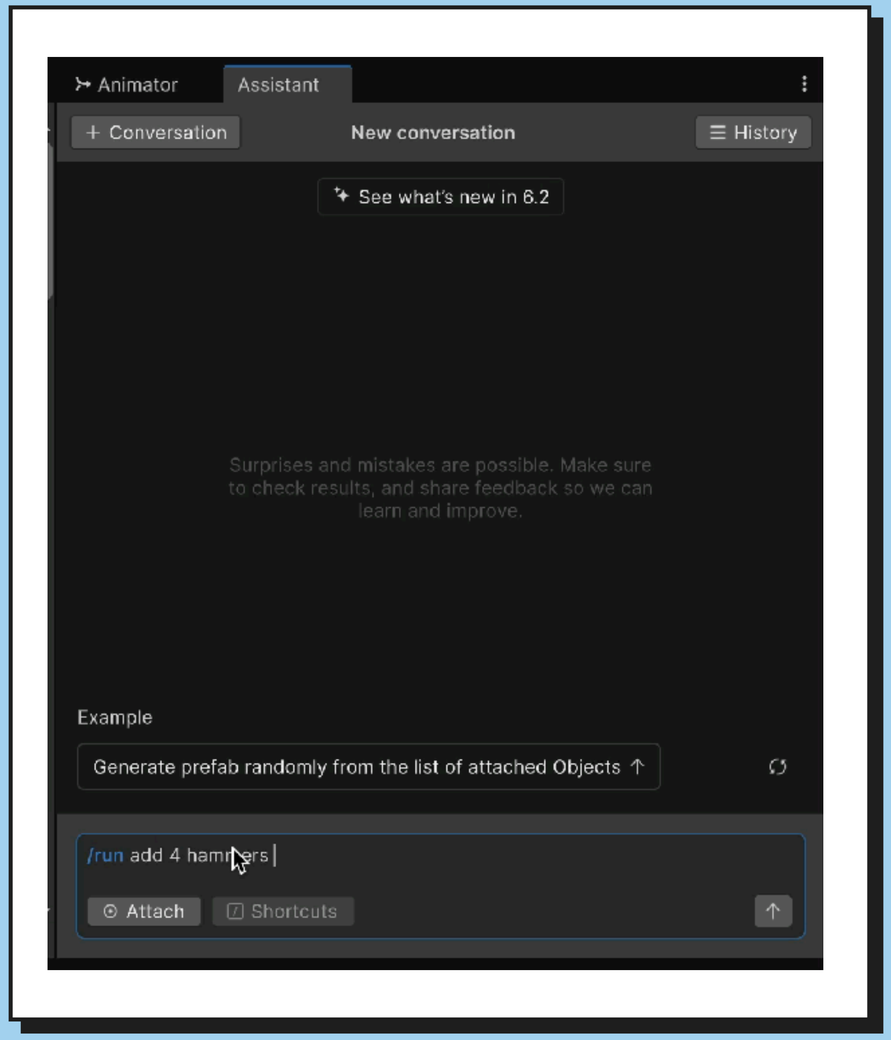
Start with dogfood

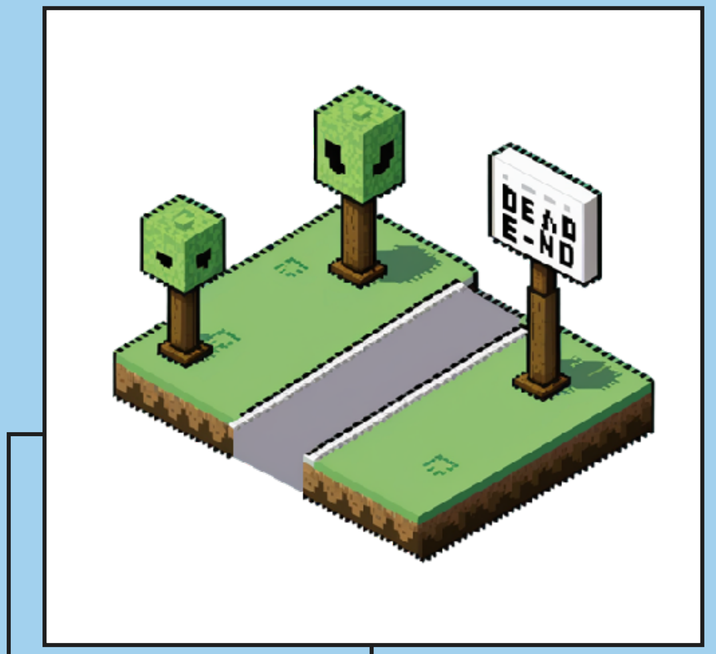
Research | UX | UI | Implementation

Help users recover from imperfect results

Graceful degradation

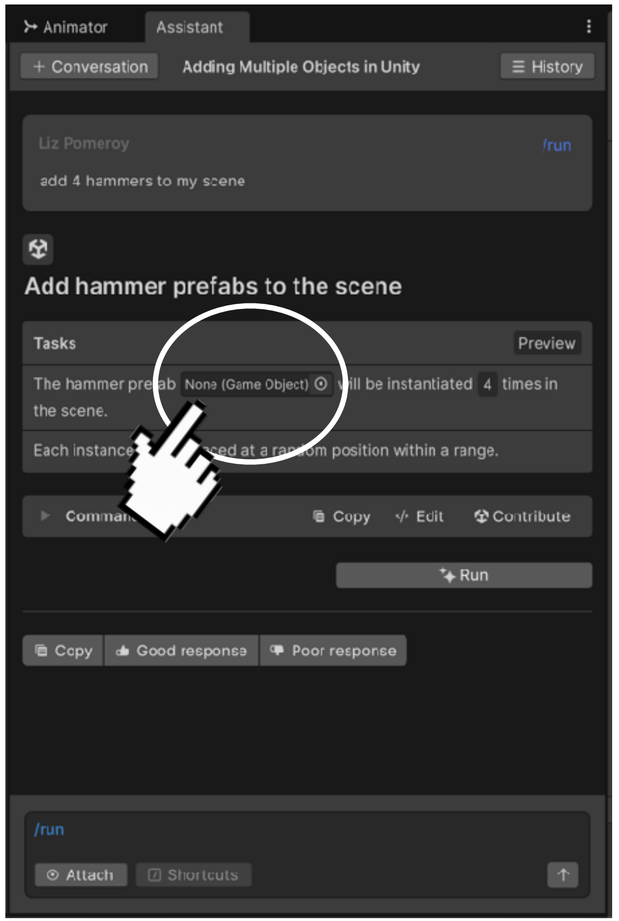
Graceful degradation

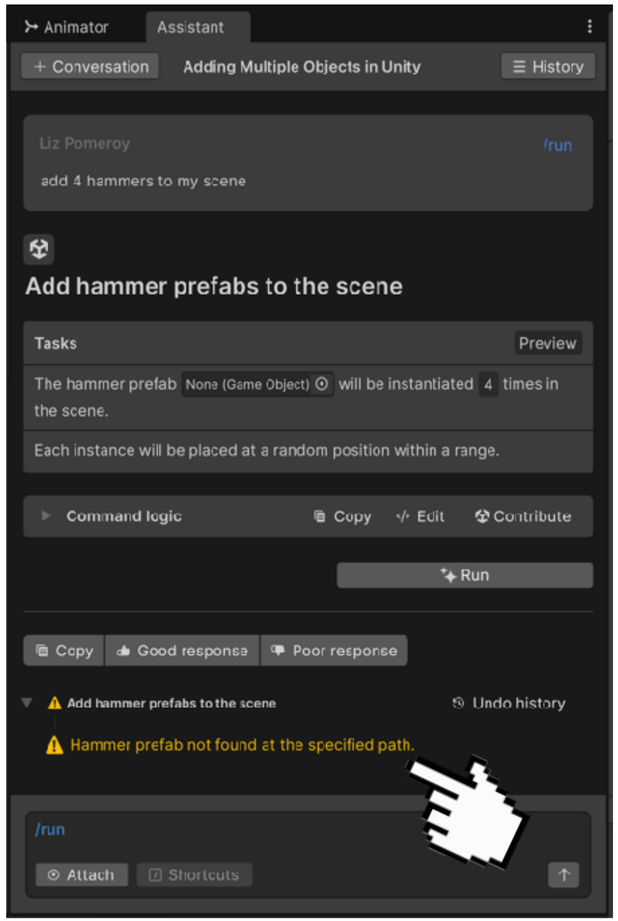




Graceful degradation

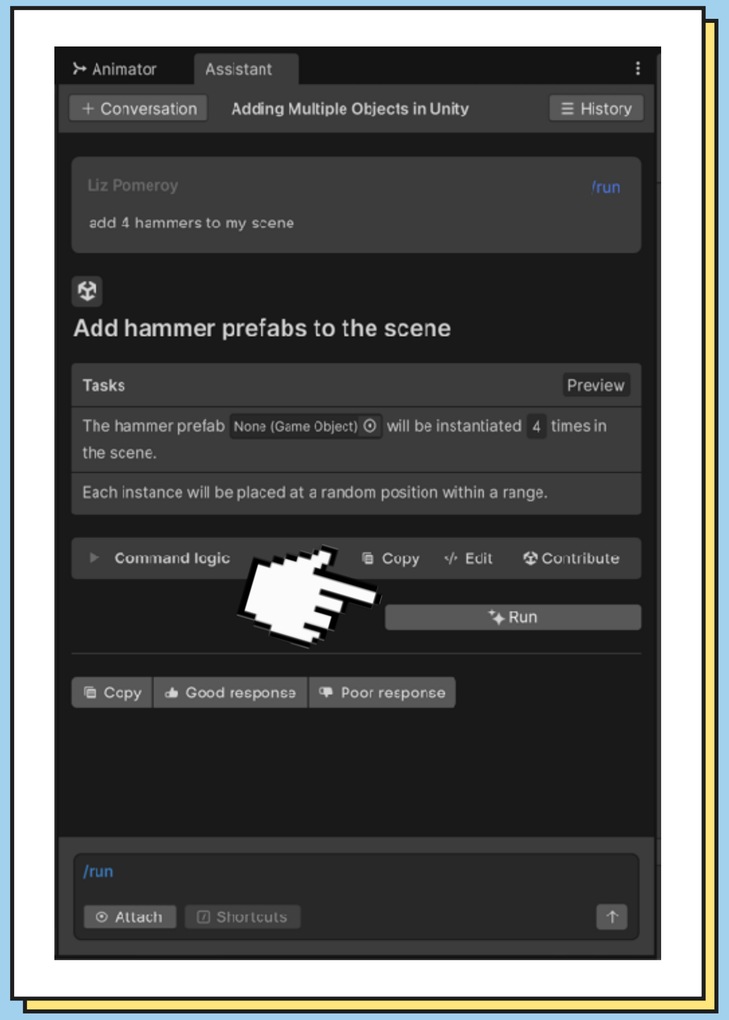
keeps the user moving toward their goal

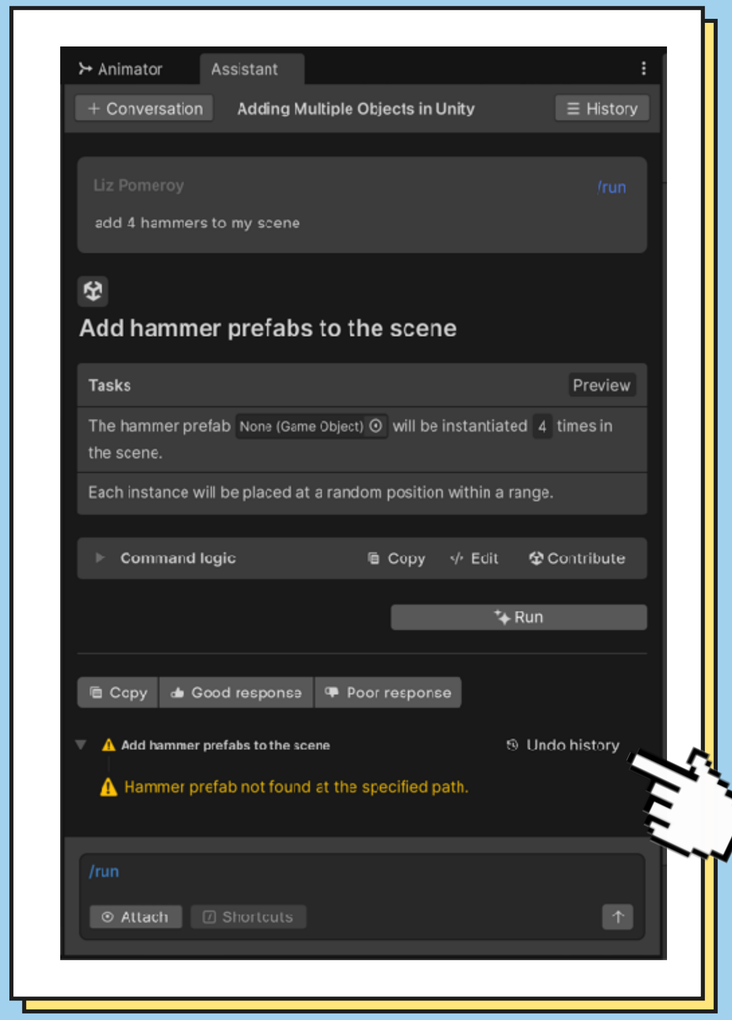




Graceful degradation

keeps the user moving toward their goal





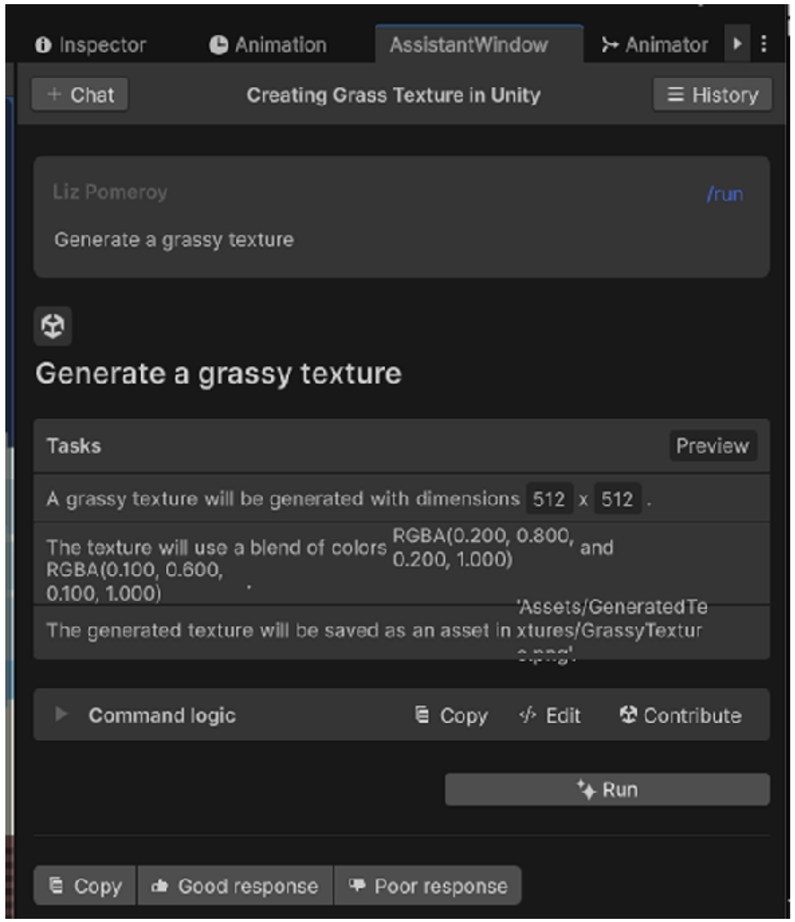
Research | UX | UI | Implementation

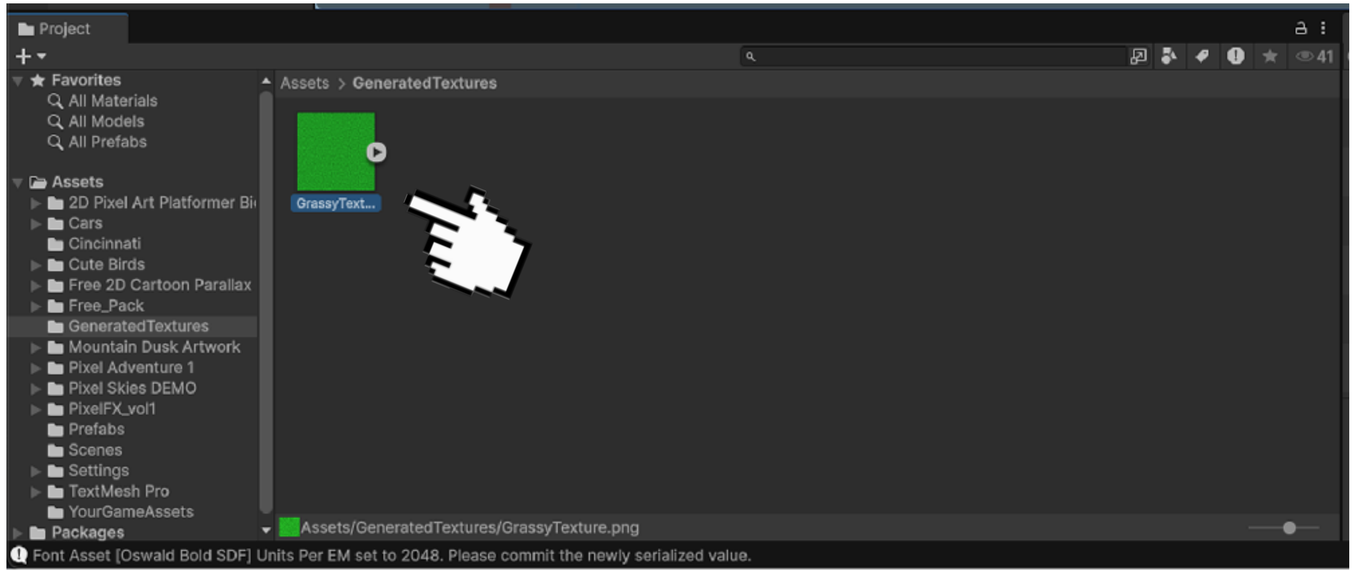
Design for ambiguity

Research | UX | UI | Implementation

Design for ambiguity

Generate a texture >


Inspector
Animation
AssistantWindow
Animator
+ Chat
Creating Grass Texture in Unity
History
Liz Pomeroy
/run
Generate a grassy texture
Generate a grassy texture
Tasks
Preview
A grassy texture will be generated with dimensions 512 x 512 .
The texture will use a blend of colors RGBA(0.200, 0.800, 0.200, 1.000) and RGBA(0.100, 0.600, 0.100, 1.000) .
The generated texture will be saved as an asset in 'Assets/GeneratedTextures/GrassyTexture.png'.
Command logic
Copy
Edit
Contribute
Run
Copy
Good response
Poor response

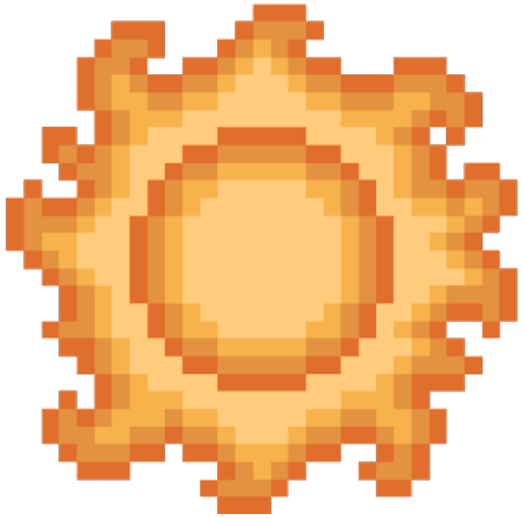

Project
Favorites
All Materials
All Models
All Prefabs
Assets
2D Pixel Art Platformer Bi
Cars
Cincinnati
Cute Birds
Free 2D Cartoon Parallax
Free_Pack
GeneratedTextures
Mountain Dusk Artwork
Pixel Adventure 1
Pixel Skies DEMO
PixelFX_vol1
Prefabs
Scenes
Settings
TextMesh Pro
YourGameAssets
Packages
Assets > GeneratedTextures
GrassyText...
Assets/GeneratedTextures/GrassyTexture.png
Font Asset [Oswald Bold SDF] Units Per EM set to 2048. Please commit the newly serialized value.

Research | UX | UI | Implementation

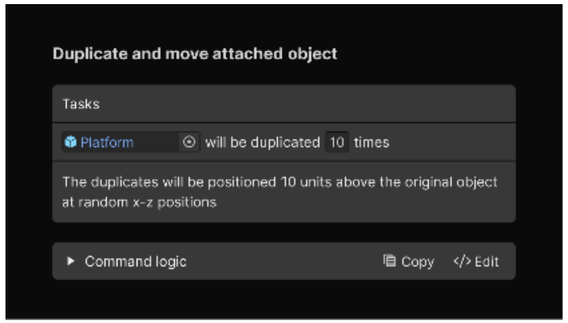
Design for ambiguity

Design an *ecosystem*

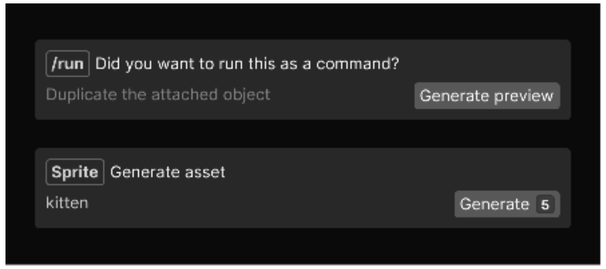
not a flowchart



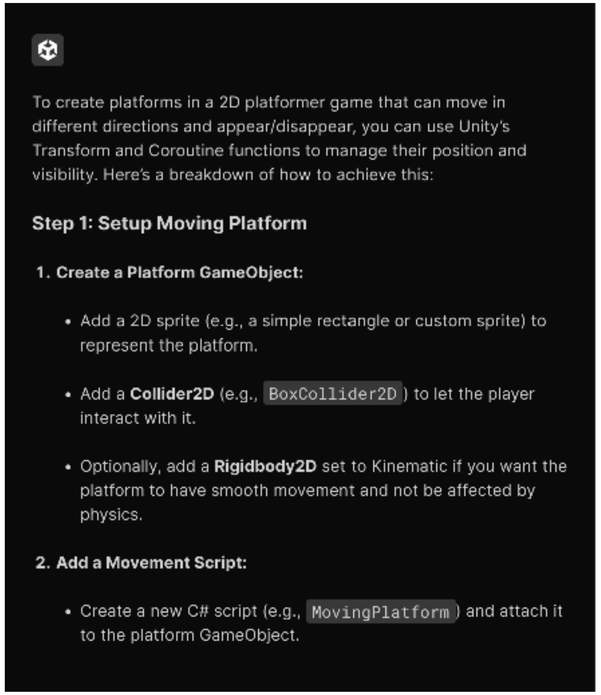
code block



command preview



internal links



To create platforms in a 2D platformer game that can move in different directions and appear/disappear, you can use Unity's Transform and Coroutine functions to manage their position and visibility. Here's a breakdown of how to achieve this:

Step 1: Setup Moving Platform

1. **Create a Platform GameObject:**
   - Add a 2D sprite (e.g., a simple rectangle or custom sprite) to represent the platform.
   - Add a **Collider2D** (e.g., `BoxCollider2D`) to let the player interact with it.
   - Optionally, add a **Rigidbody2D** set to Kinematic if you want the platform to have smooth movement and not be affected by physics.
2. **Add a Movement Script:**
   - Create a new C# script (e.g., `MovingPlatform`) and attach it to the platform GameObject.

natural language

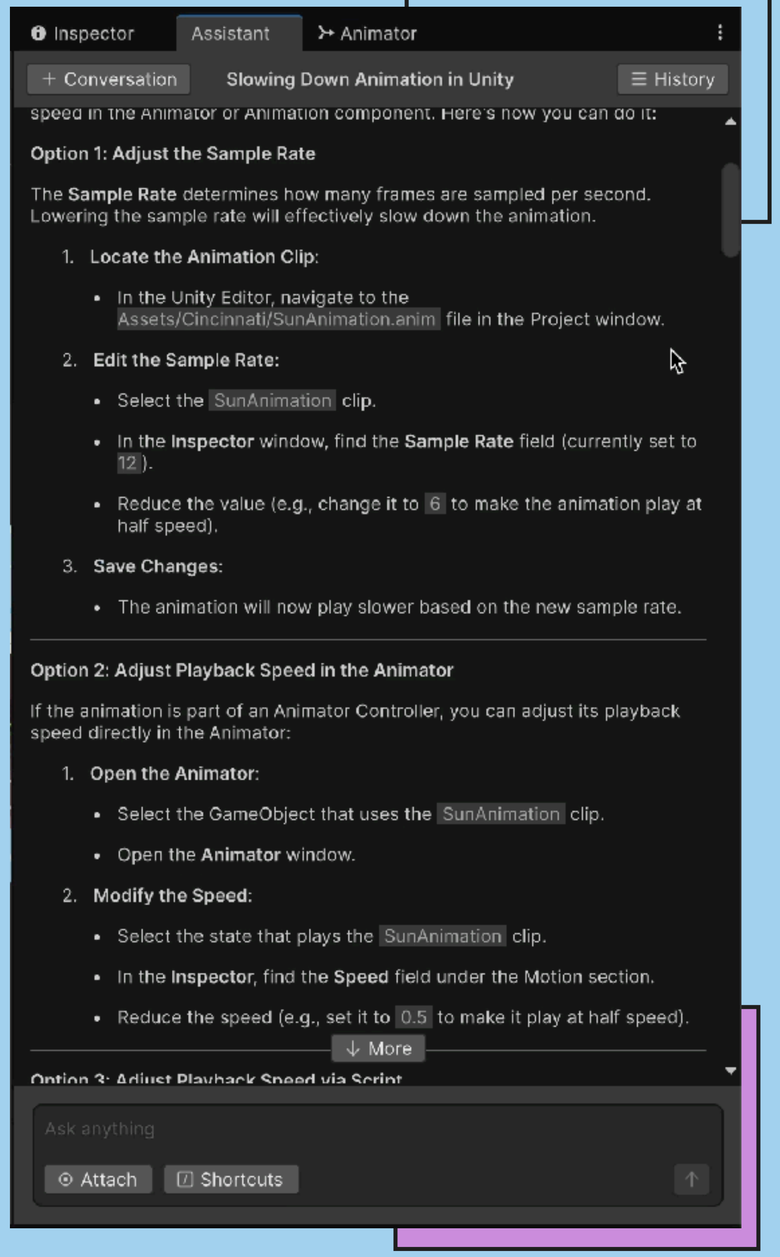
Research | UX | UI | **Implementation**

Find opportunity in the back-end

Research | UX | UI | **Implementation**

Find opportunity in the back-end

The content *is* the experience



AI

The designer's toolbox

Research

Deeply understand your users' mental models

UX

Help users recover from imperfect results

UI

Design for ambiguity

Implementation

Find opportunity in the back-end

AI

The designer's toolbox

Research

Deeply understand your users' mental models

*

Get feedback from the entire learning curve, even if it means dog-fooding

UX

Help users recover from imperfect results

*

Graceful degradation and planned friction add transparency and control

UI

Design for ambiguity

*

An ecosystem of modular components reliably renders unpredictable output

Implementation

Find opportunity in the back-end

*

Sit with engineers to bolster the content experience

AI

The designer's toolbox

📣 Now in beta 📣 unity.com/ai

Research

Deeply understand your users' mental models

*

Get feedback from the entire learning curve, even if it means dog-fooding

UX

Help users recover from imperfect results

*

Graceful degradation and planned friction add transparency and control

UI

Design for ambiguity

*

An ecosystem of modular components reliably renders unpredictable output

Implementation

Find opportunity in the back-end

*

Sit with engineers to bolster the content experience