Game design tools

For when spreadsheets and flowcharts aren't enough

Katharine Neil
Independent developer

What are game design tools?

Game design tools help you solve design problems without having to build playable experiences in order to test out your ideas.

- Conceptual models, notation systems, software tools
- We generally don't use them!
- We've been working on this problem for a while…
  (See "Formal Abstract Design Tools", Doug Church, *Game Developer Magazine*, 1999)

Design tools

- Main goal: support thinking
- Support ambiguous, evolving ideas
- Unresolved conflicts and issues are expected
- Messy, visual
- Support an anarchic, even private process (like a designer's notebook)
- "But how does this help us build the game?" <– wrong question

Production tools

- Main goal: support production
- Create usable, bug free assets
- Technical mistakes and logical errors break things
- Clean, optimal, efficient
- Easy to measure and demonstrate progress and outcomes
- Make sense to programmers and producers

Tools we typically use for game design

- Documenting
  - GDDs, spreadsheets, flowcharts, diagrams
- Prototyping and iterative development
- Player metrics





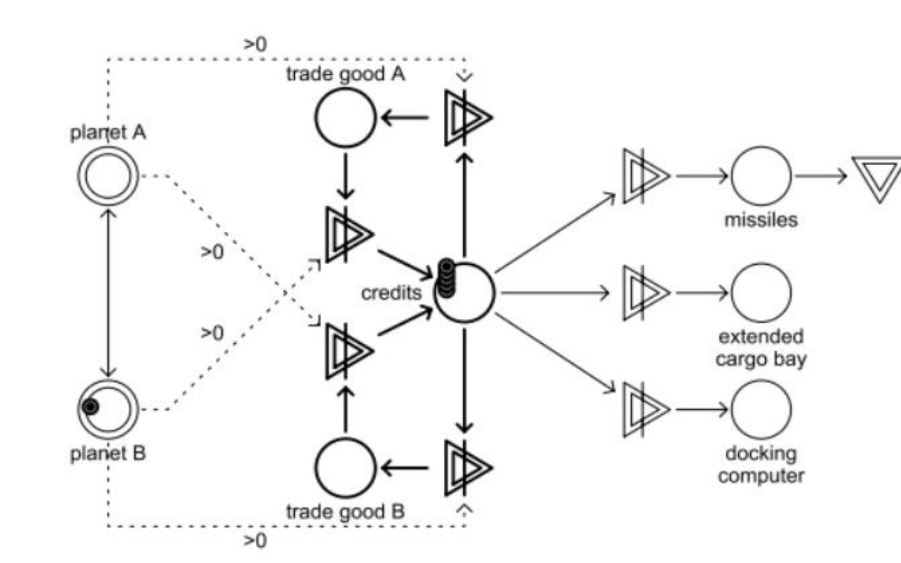
What design tools can help with

- System design and balancing
- Narrative design
- Progression design
- Level and mission design
- Design for procedural content generation

Game design tools you can use now

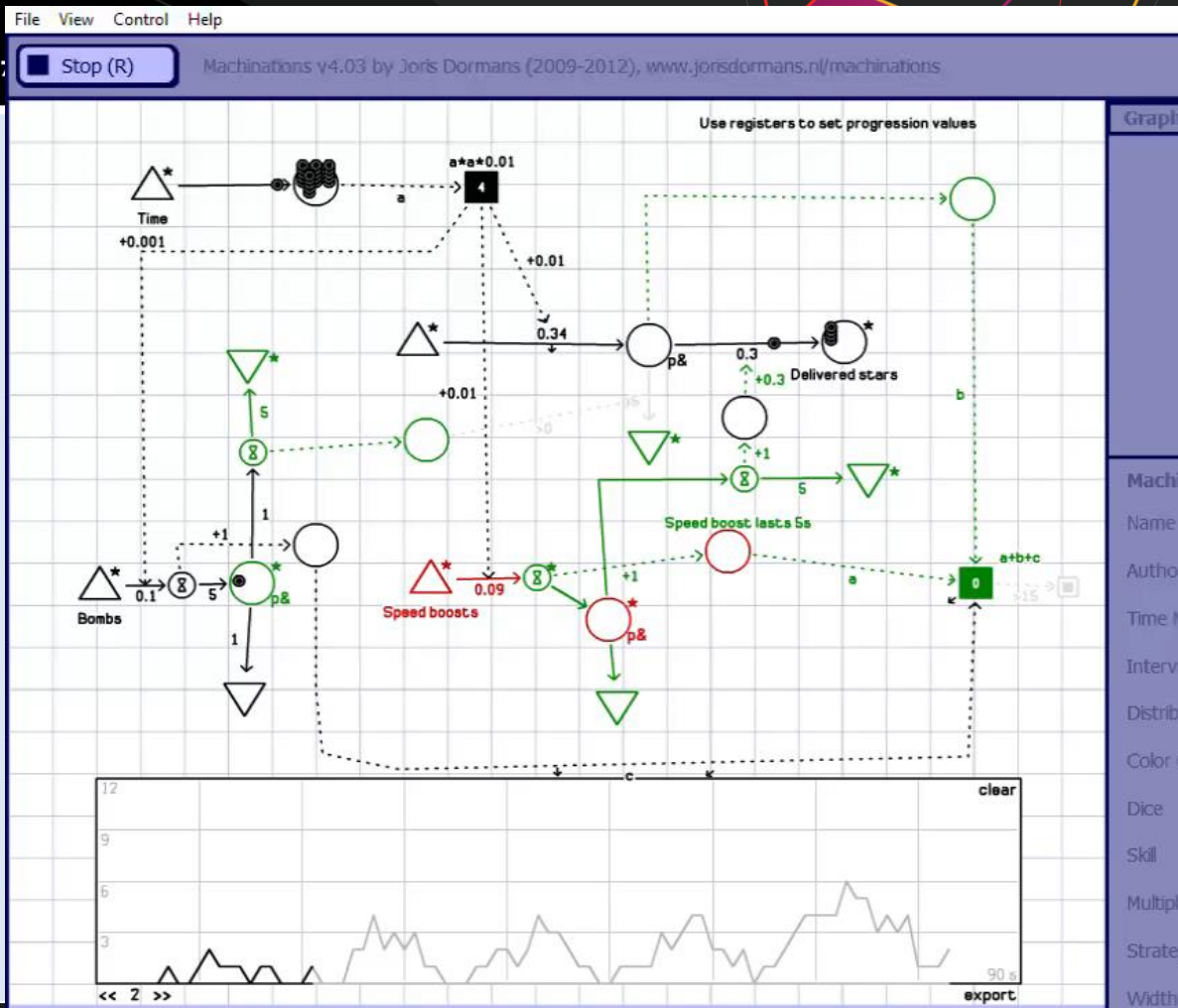
Machinations

- Notation system for modelling game systems
- Models game economies as resource flows
- Provides interactive simulation of game dynamics



Prototype

Machinations model



Machinations is useful for:

- Modelling, balancing systems – especially for games with emergent dynamics
- Quickly sketching out ideas
- Analysing, learning from other games

Limitations:

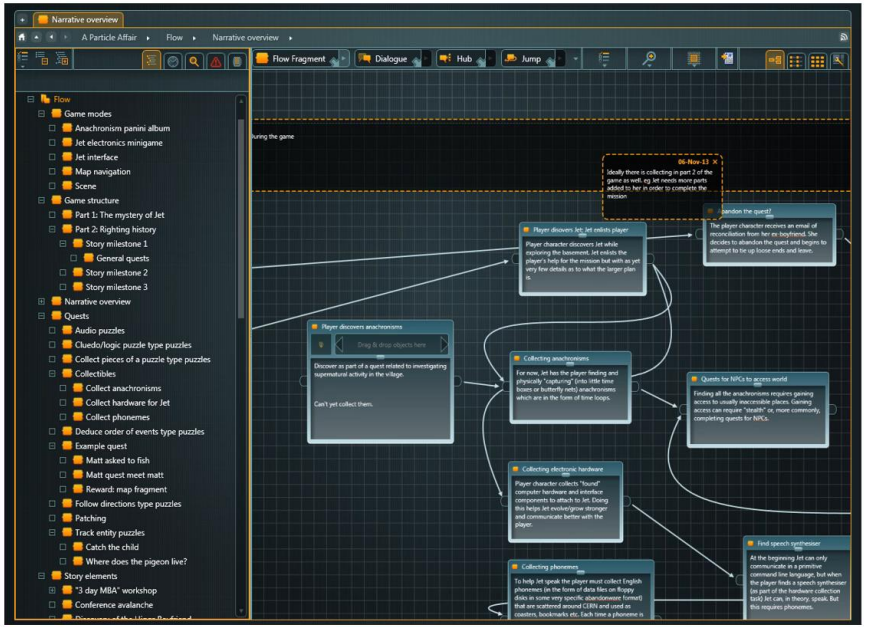
- Not suited to data-heavy, scripted gameplay
- Can’t factor in « game feel » or topography

Where to get it:

- http://www.jorisdormans.nl/machinations/
  Book: “Game Mechanics: Advanced Design”, Adams & Dormans

Articy:Draft

- Narrative design and mission design
- Flow diagram-style interface supports branching, graph-based, nested structures
- Also serves as an authoring tool



Articy:Draft is useful for...

- Writers who want a tool like Scrivener – but for games
- Organising and visualising design and narrative materials
- Narrative-based, dialog-heavy games

Limitations

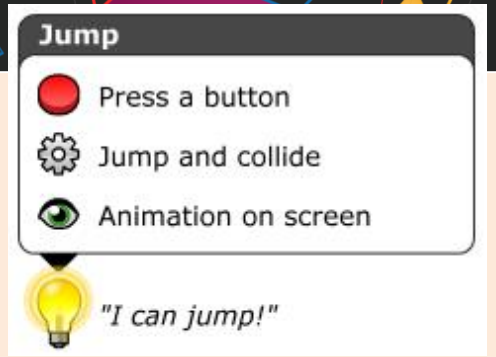
Heavily tailored towards RPGs, adventure games

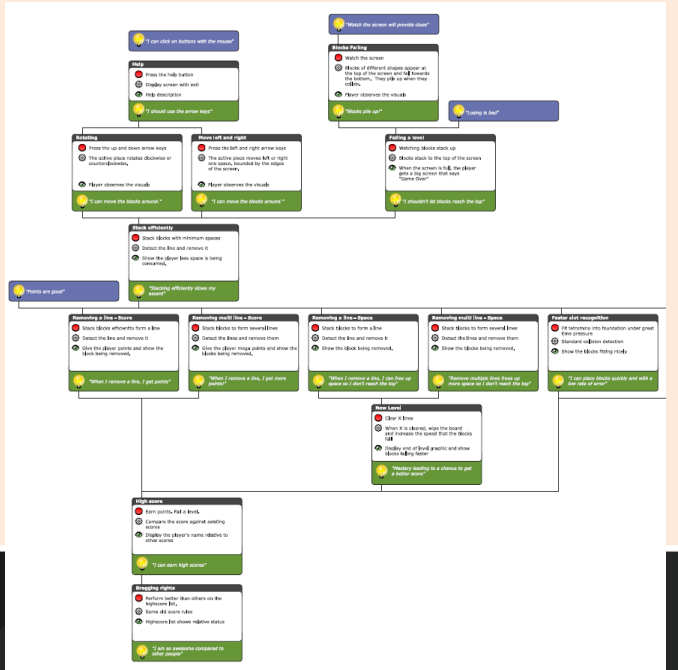
Where to get it

http://www.nevigo.com

Skill atoms

- A diagramming system
- A *skill atom* describes how the player gains a new skill
- A *skill chain* describes the evolution of the player's skill set
- Progression design using the lens of player skills





Skill atoms are useful for…

- Focusing on player experience
- Onboarding/scaffolding

Limitations

- Skill chain graphs can become large, hard to read/use

Where to get it

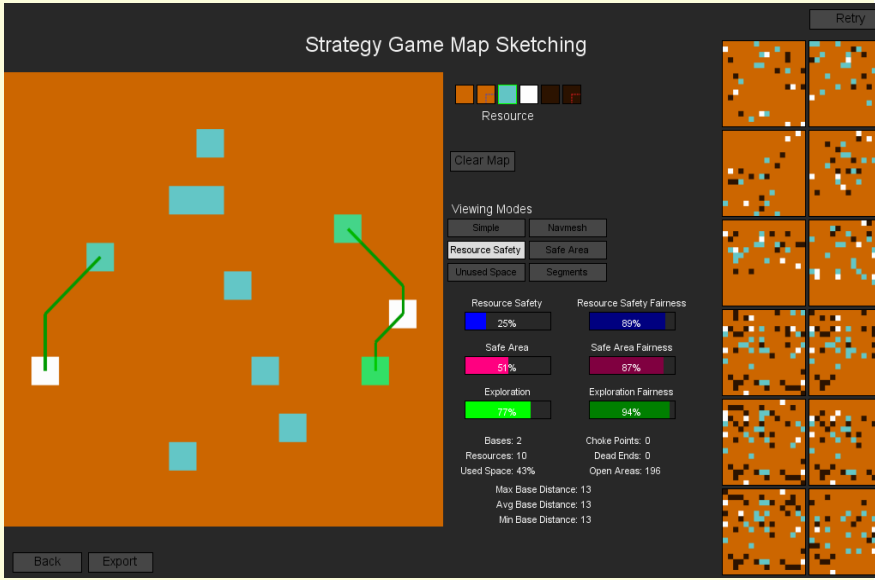
- Read the article here: http://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php

Game design tools of the future:

"Mixed initiative" (procedural content generation and AI-assisted) design tools

The Sentient Sketchbook

- AI-assisted top-down sketching of game levels – particularly strategy game maps
- Place bases, resources, rough collision scene, goals
- Evaluates maps for player balance and gameplay pacing
- Shows navigable paths, choke points
- Fleshes out details and suggests alternative designs while you sketch



The Sentient Sketchbook is useful for…

- The grey-boxing stage of level design
- Strategy games, FPS

Limitations

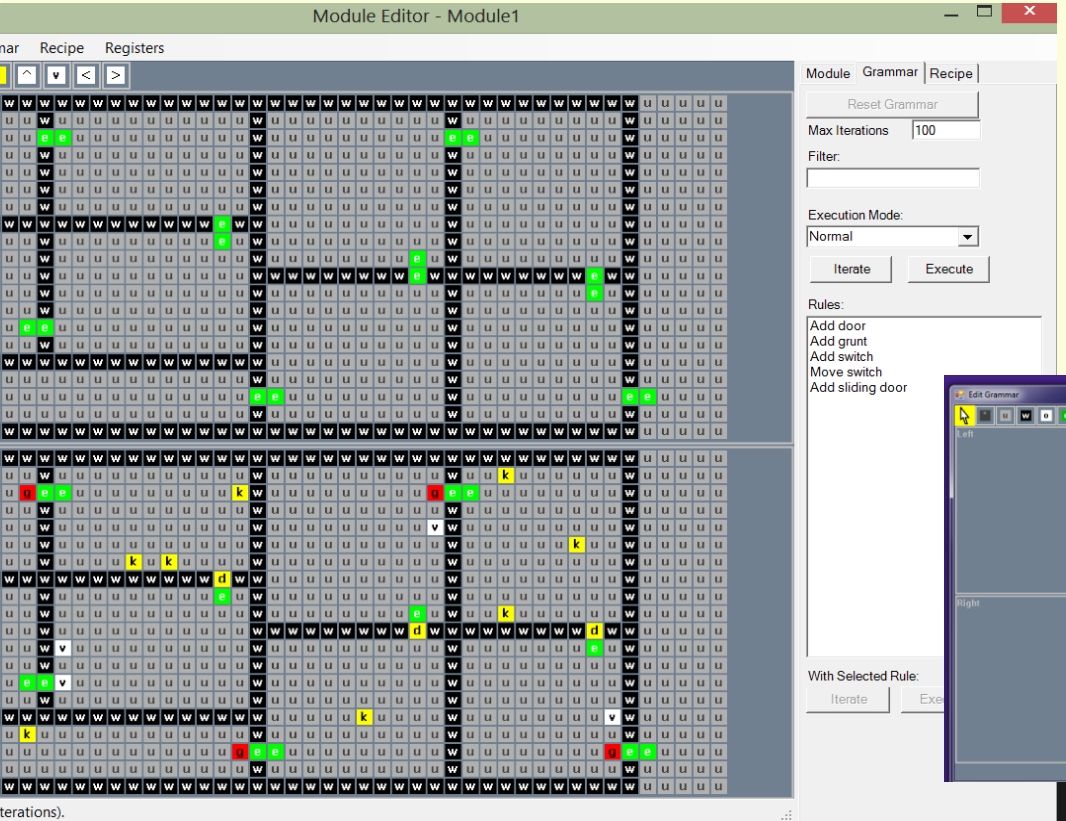
- Quite genre-specific
- Preset map sizes
- High level, approximate

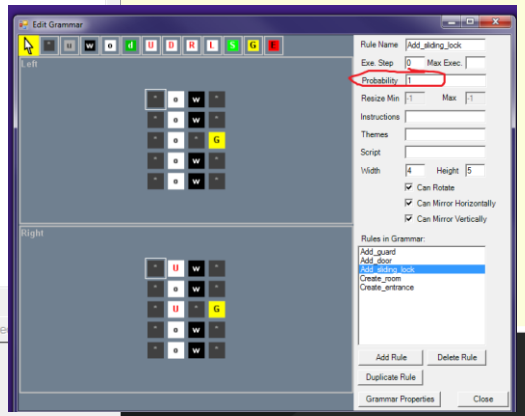
Where to get it

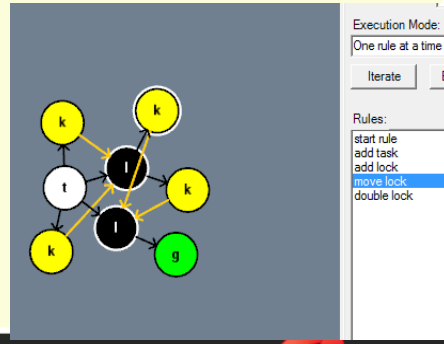
http://www.sentientsketchbook.com/

Ludoscope

- AI-assisted mission and level design
- Design-time procedural generation with designer-friendly approach
- Can transform abstract mission structures into level designs







Ludoscope is useful for:

- Generating or fleshing out level designs
- Devising procedural level generation rules without scripting/code
- Freaky new ways to think about your level design process

Limitations

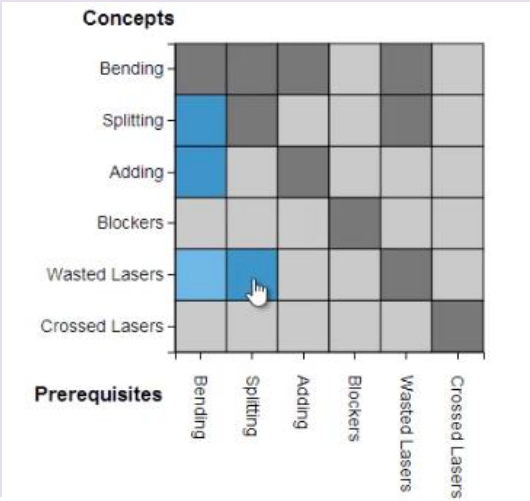
- Requires hard work in abstract thinking
- Can be challenging to figure out what to use it for
- Highly experimental and still in development

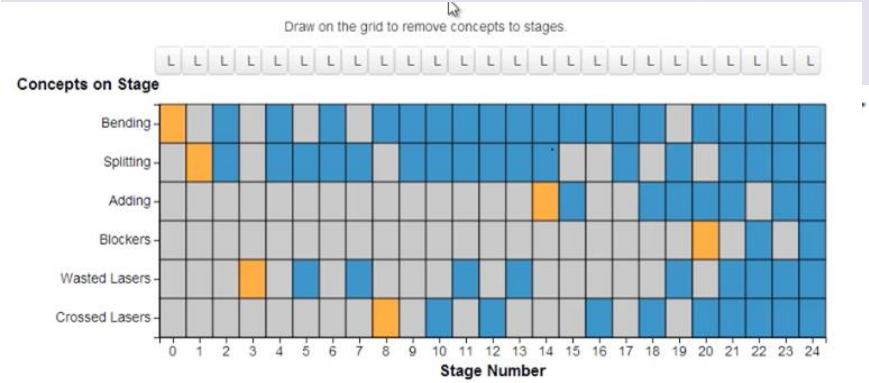
Where to get it

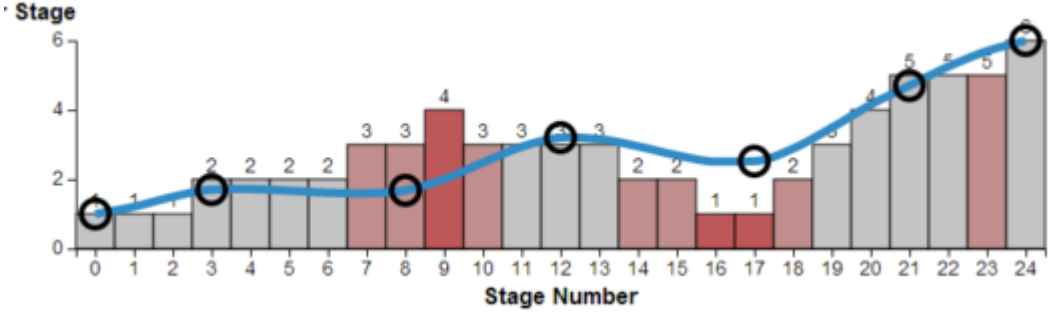
Ask Joris Dormans about beta testing: jd@jorisdormans.nl

PCG-based level editor for the game *Refraction*

- Mixed-initiative design of puzzle game levels
- Computer-aided progression design
- Helps the designer (and the procedural generation) stick to the progression rules and structure they've defined







Refraction's level editor is useful as:

- Proof of concept of how we can embed computational assistance into level editing/world building tools

Limitations

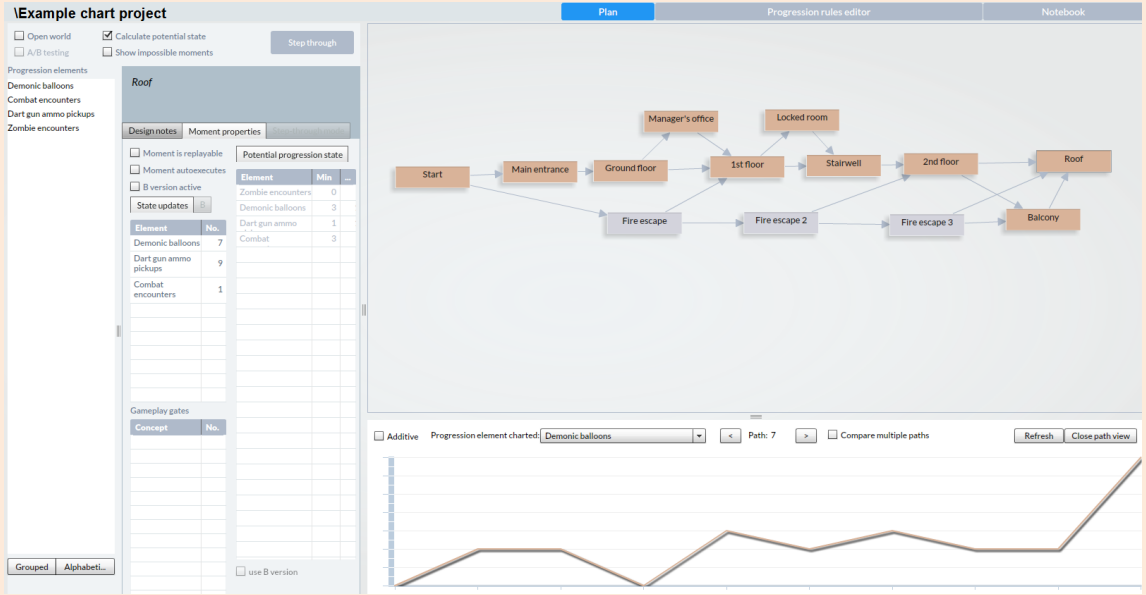
- You can't use it - it's specifically for building Refraction game levels
- Built to handle design for a linear game with a small feature set

Where to read about it

https://adamsmith.as/papers/uist2013_progression.pdf

Let’s make design tools!

Progressimo



- Standalone progression design environment inspired by the *Refraction* tool
- Branching and open-world progression structures
- Calculates game state, mission unlocking
- (Disclosure: I made this tool!)

Progressimo is useful for:

- Progression-heavy games e.g. levels and missions, game-as-a-service, adventure games, action RPGs, open world games
- Visualising and walking through missions and how they fit together in the game
- High-level content planning for procedural generation

Limitations

- Not suitable for games where progression is driven primarily by emergent system dynamics

Where to get it

- Contact me about joining the beta: katharine.neil@gmail.com

Benefits of using game design tools

- Adds structure to a design process and makes design visible
- Can provide a safe space in which to attempt ambitious, complex designs
- Lessens design's reliance on production
- Learn new ways of thinking that impact the way you design even when you're *not* using tools

Limitations and pitfalls

- Can take a while to learn and be hard to use
- Hard to tell what fun looks like in abstract form
- Not great for modelling game feel and interaction
- A tool acts like a filter on your ideas. It has its own agenda!
- False positives, false negatives (fun in the tool but not in the game & vice versa)

Suggested approaches

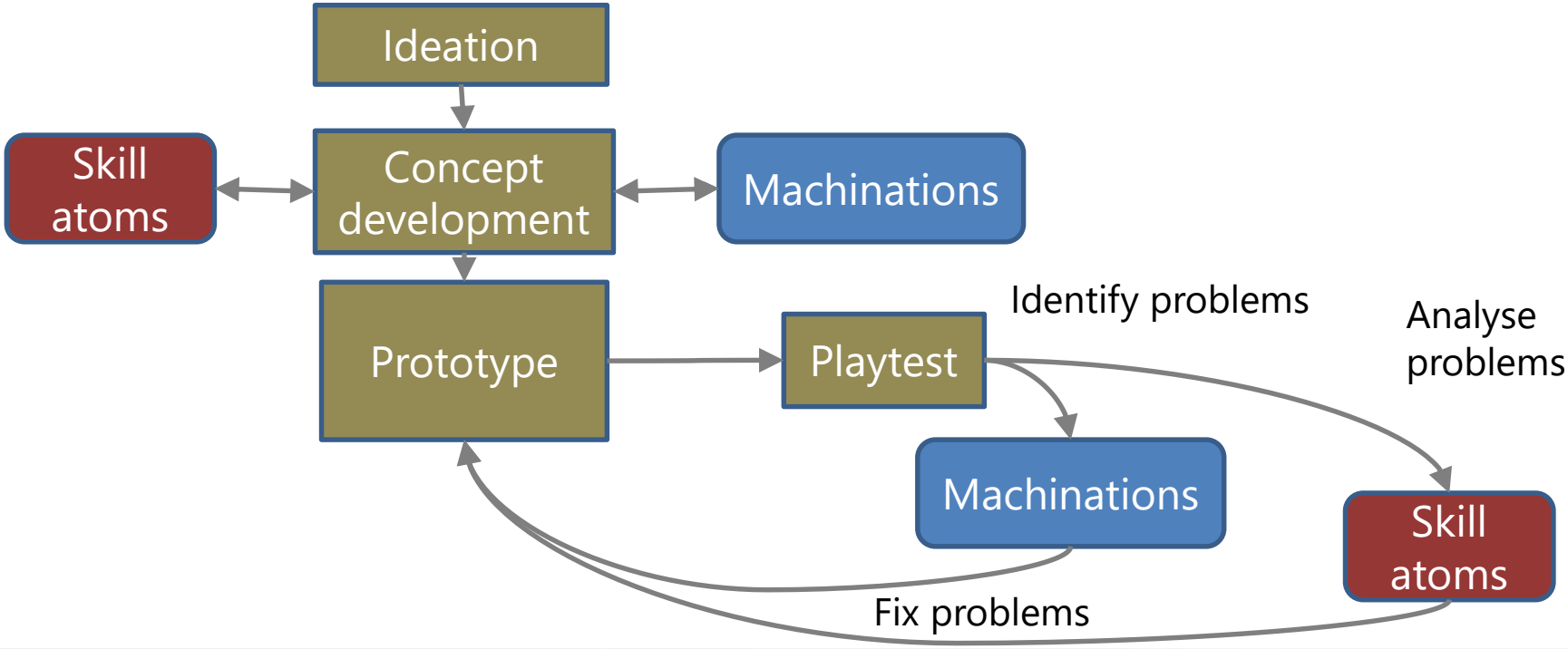
Use a “toolbox” approach

- Have a range of tools to hand (no “one tool to rule them all”)

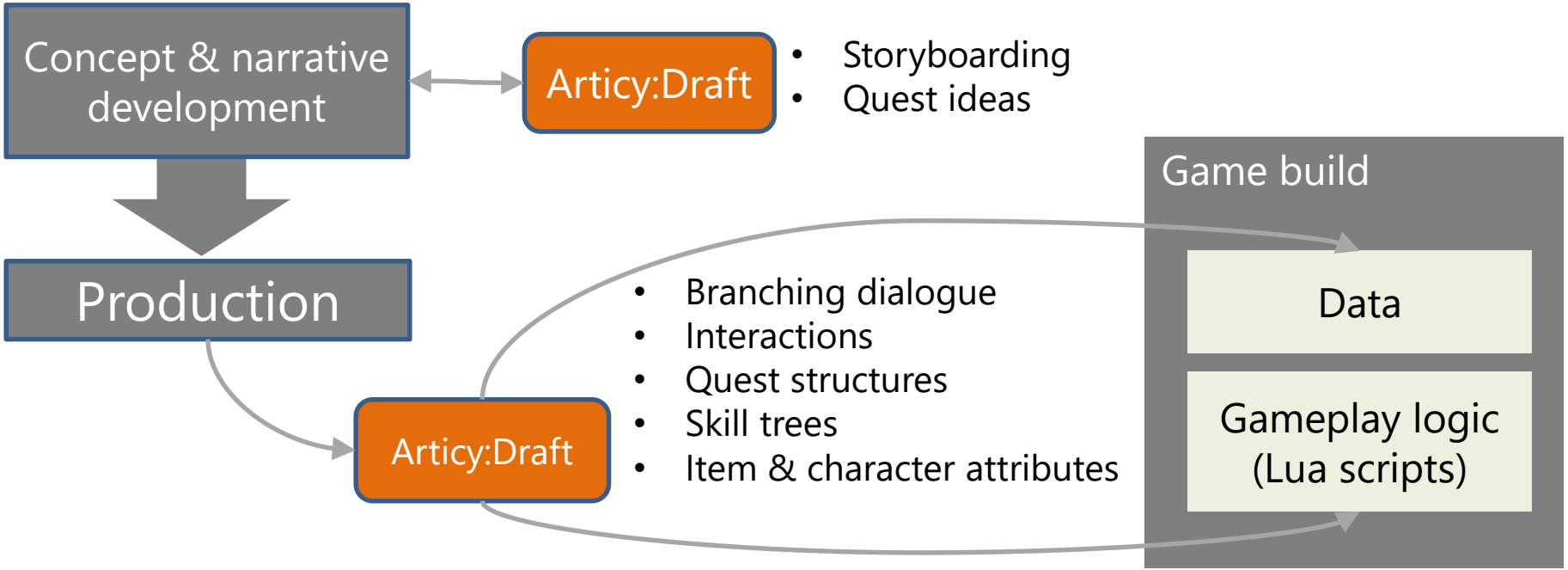
Use game design tools to complement other methods

- For example, alongside prototyping (to tell you things a prototype can’t tell you)

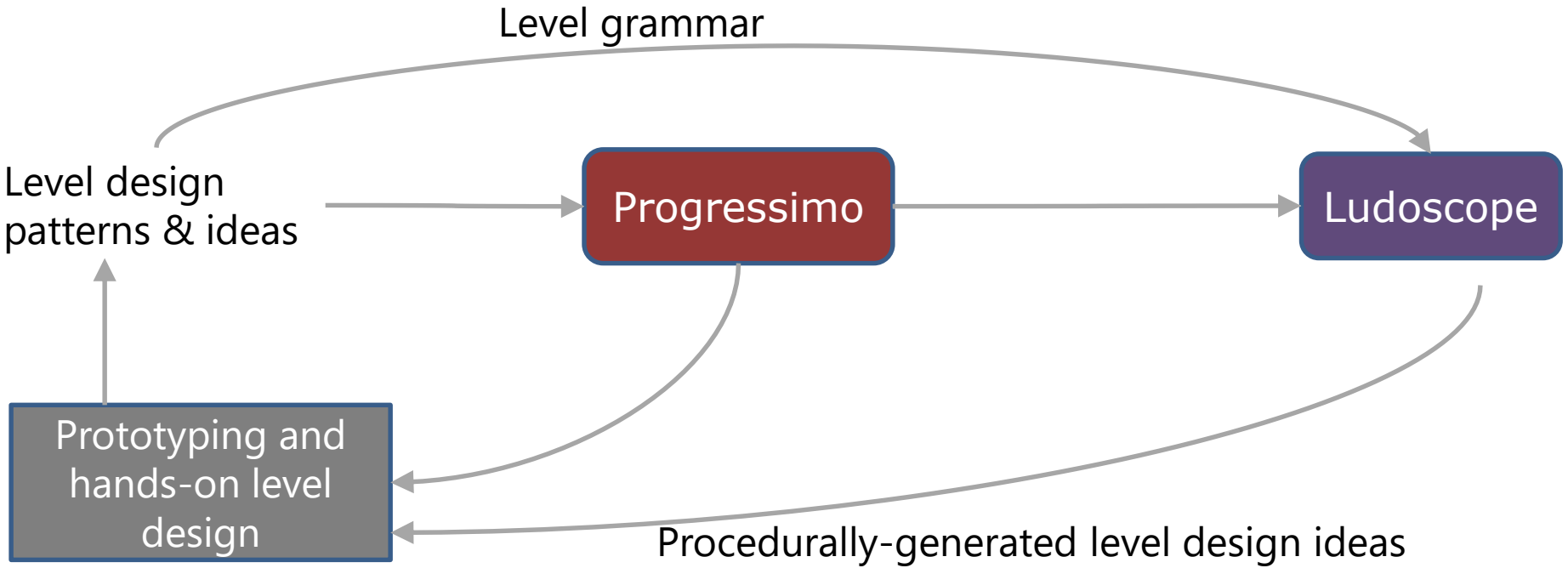
Example: Dan Cook's design workflow



Example: Design workflow for platformer/RPG *Wanderer*



Example: Workflow for my top-down shooter



Thanks for listening!

katharine.neil@gmail.com

@haikus_by_KN