nature machine intelligence

Article

https://doi.org/10.1038/s42256-025-00981-4

Goals as reward-producing programs

Received: 14 May 2024

Accepted: 5 January 2025

Published online: 21 February 2025

Check for updates

Guy Davidson[1,4] ✉, Graham Todd[2,4], Julian Togelius[2], Todd M. Gureckis[3,5] & Brenden M. Lake[1,3,5]

People are remarkably capable of generating their own goals, beginning with child's play and continuing into adulthood. Despite considerable empirical and computational work on goals and goal-oriented behaviour, models are still far from capturing the richness of everyday human goals. Here we bridge this gap by collecting a dataset of human-generated playful goals (in the form of scorable, single-player games), modelling them as reward-producing programs and generating novel human-like goals through program synthesis. Reward-producing programs capture the rich semantics of goals through symbolic operations that compose, add temporal constraints and allow program execution on behavioural traces to evaluate progress. To build a generative model of goals, we learn a fitness function over the infinite set of possible goal programs and sample novel goals with a quality-diversity algorithm. Human evaluators found that model-generated goals, when sampled from partitions of program space occupied by human examples, were indistinguishable from human-created games. We also discovered that our model's internal fitness scores predict games that are evaluated as more fun to play and more human-like.

Understanding how humans create, represent and reason about goals is crucial to understanding human behaviour. Goals are pervasive throughout psychology[1–3], having been studied from perspectives such as motivation[4–6], personality and social psychology[7,8], and learning and decision-making[9,10]. But what is a goal? Elliot and Fryer offer the workable, albeit simplified, definition: a representation of a future object to be approached or avoided (see also refs. 3,10). Reinforcement learning offers another formulation, operationalizing goals as maximizing cumulative reward over a series of steps[11]. Typical goals in reinforcement learning tasks include reaching a target location, winning in a video or board game[12], or placing an object in a specified position (for example, Fig. 1a), such that success can be characterized by reaching a target state.

By contrast, people routinely create novel, idiosyncratic goals with richness beyond these common modelling settings. Chu et al.[13] report the example of Gareth Wild, who set an unusual goal for himself to park in every spot in a particular grocery store's parking lot (Fig. 1b). Children routinely devise fun and compelling goals without external guidance, such as creating a 'truck carrier truck' (Fig. 1c) or stacking as many blocks as possible in a single tower (Fig. 1d). Beyond being fun, these playful goals serve a crucial role in learning to structure and solve arbitrary problems[14–16]. Indeed, it has been argued that autonomously setting and achieving goals is a core component of human intelligence[13,17].

We propose a framework for modelling human goal generation as synthesizing reward-producing programs (Fig. 1, bottom row). There are several advantages to representing goals as symbolic programs, which map an agent's behaviour to a reward score indicating the degree of success. First, a structured language facilitates the compositional reuse of motifs across disparate goals. Such reuse makes capturing the wide range of human creativity in goal creation substantially more tractable: In Fig. 1e, we illustrate a simple ball-throwing game (in black) and four distinct variants (in red, blue, pink and brown) composed in part from shared components: balls being thrown (highlighted in yellow), the thrown ball hitting something (orange) and the thrown ball landing somewhere (green). Second, our choice of representation makes goal semantics explicit. The particular grammatical elements of our representation each fulfil particular roles, such as predicates

[1]Center for Data Science, New York University, New York, NY, USA. [2]Department of Computer Science and Engineering, New York University, New York, NY, USA. [3]Department of Psychology, New York University, New York, NY, USA. [4]These authors contributed equally: Guy Davidson, Graham Todd. [5]These authors jointly supervised this work: Todd M. Gureckis, Brenden M. Lake. ✉e-mail: guy.davidson@nyu.edu

https://doi.org/10.1038/s42256-025-00981-4

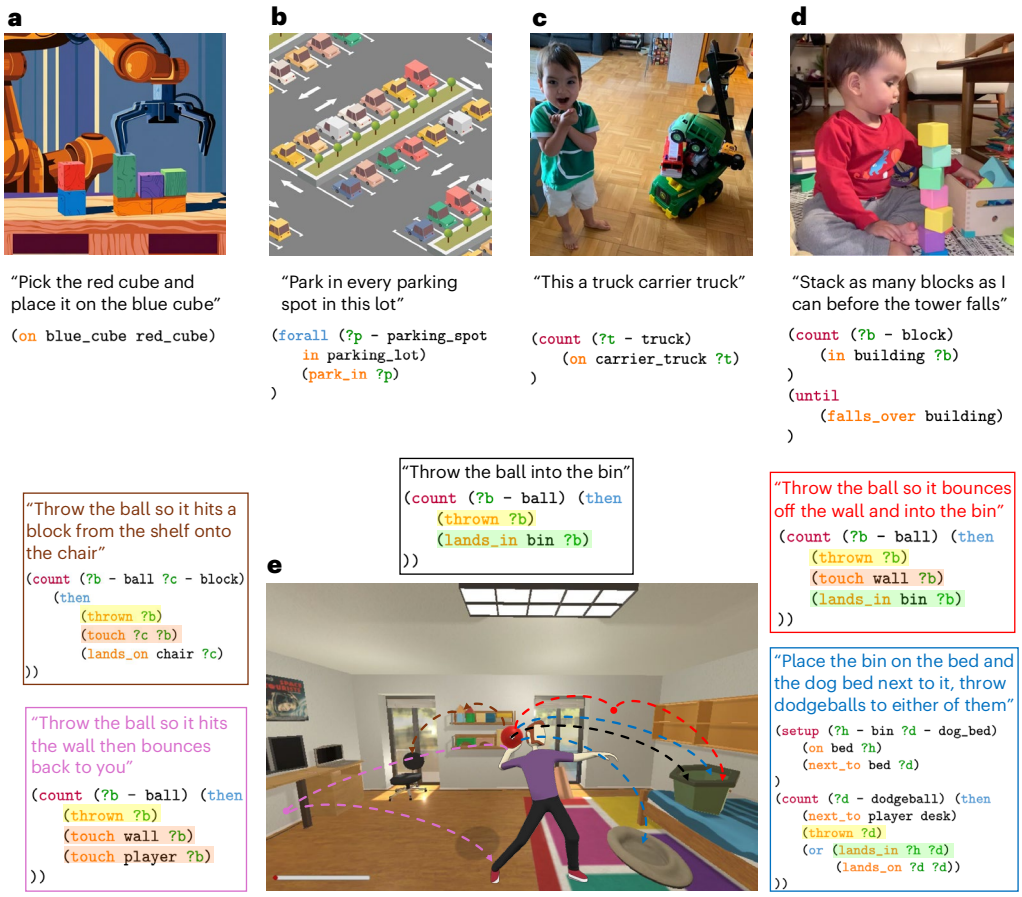


Fig. 1 | Goals as reward-producing programs. a–**d**, Different goals, presented in natural language and mapped to pseudo-code in a program-like representation. Panel **a** depicts a pick-and-place task of the form often studied in reinforcement learning and robotics, presented in contrast to human-created goals: a self-imposed challenge to park in every spot in a parking lot (**b**), creating a 'truck carrier truck' (**c**), and stacking blocks until a tower falls (**d**). **e**, A set of varied yet related goals in our experiment environment, of which the blue and pink were created by participants in our experiment. Each goal is represented by a throw trajectory (dashed line in the illustration) matching a description of the goal (whose text is the same colour as the line). We highlight shared compositional components between programs in yellow, orange and green. Our program representations are reward-producing, that is, run on sequences of agent interactions with an environment (state–action pairs) and emit a score with respect to the specified goal. Our pseudo-code and DSL both use syntax inspired by the LISP (list processing) programming language, where function calls have the function name as the first token inside the parentheses. Participants in our experiment created some of these goals; see Supplementary Fig. 1 for representations of the blue and pink programs in our DSL. Credit: car park in **b**, Vecteezy.com.

(that is, specific and evaluable relations between objects, coloured orange in the programs in Fig. 1) and temporal modals (that is, relationships in time between goal components, such as 'until' and 'then' in Fig. 1). Finally, goals-as-programs are executable; that is, they can be computationally interpreted to detect when a goal is entirely or partially achieved (Fig. 1e, each program would be interpreted and provide a score only when the matching throw trajectory is completed).

In this Article, we demonstrate that programs can capture real human-created goals in a naturalistic domain and build a model capable of generating new programs representing human-like goals. We devised a rich experimental environment for goal generation and asked human participants to generate playful goals in the form of single-player scoreable games (see ref. 18 on the relationship between games and goals). We translated these games into programs in a domain-specific language (DSL) that explicitly models the core semantics of the participants' creations. We also developed a goal program generator (GPG) model to generate new goals in this representation, learning a fitness metric over programs to capture human likeness and sampling diverse goal programs to maximize fitness. We found that the model succeeds in generating novel games distinct from examples in the training dataset. Human raters evaluated several characteristics of model-generated games, including how human-like they were. Model games from sections of program space closer to participant-created games were judged indistinguishably from the real games, but model samples further away were not rated as highly on average. Analyses revealed that our learned fitness function predicts several human judgement questions, including how human-like games are rated. These results demonstrate that our goal representations and model capture important aspects of how people creatively construct new goals, generating plausible, diverse goals and predicting understandability and fun ratings. We conclude with a discussion of the scope of our representational hypothesis (capturing goals as programs), the relationship to prior work, some limitations of our model and avenues for future work.

Behavioural results

Although goals play a crucial role in psychological theory, there are few, if any, empirical paradigms for eliciting wide-ranging goals from study participants. We created an experimental setting that aims to capture the rich, playful and creative nature of how children (and adults) create everyday goals. We used AI2-THOR[19] (an embodied, three-dimensional environment simulation) to set up a room resembling a child's bedroom, filled with toys and other common objects (Fig. 2a; see Extended Data Fig. 1 for a larger version). In our task, we asked participants to

https://doi.org/10.1038/s42256-025-00981-4

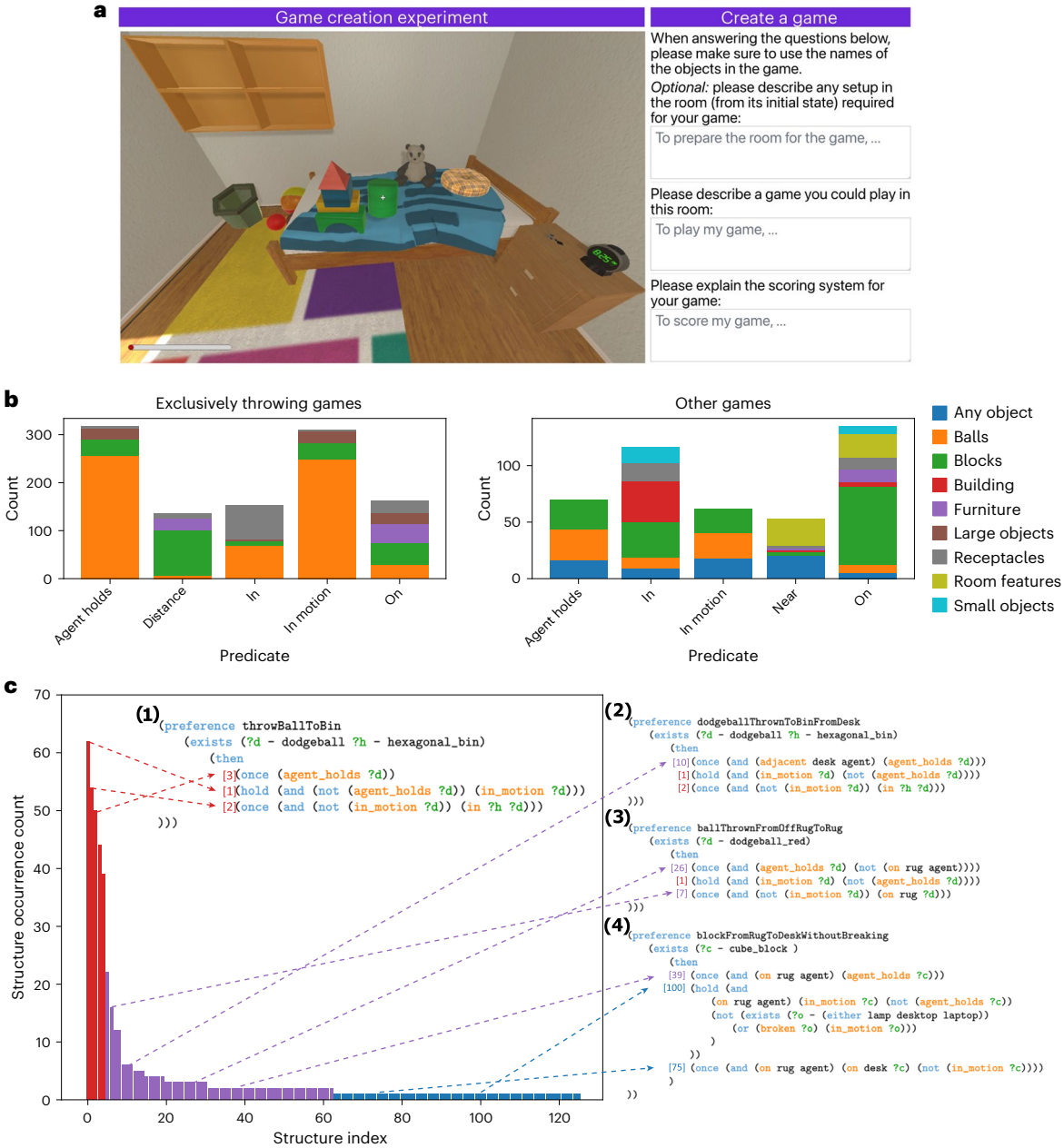


Fig. 2 | Participants in our behavioural experiment create diverse games reflecting common sense and compositionality. a, Our online game creation experiment (see full interface in Extended Data Fig. 1). **b**, Participants showcase intuitive common sense. Left: in games involving exclusively throwing, participants use balls (orange) far more often than any other object type. Right: in other games, participants refer to blocks or 'any object' more often, most often checking where objects are placed (using the in and on predicates). We most often observe balls being thrown and blocks being stacked, and while a few participants specified block-throwing games, no participant created a game involving ball-stacking. Participants also rarely specified throwing large or cumbersome objects (such as the chair or laptop), and only used buildings to specify stacking objectives (as opposed to moving or throwing them). See Extended Data Fig. 2 for an extended version of this panel (including additional object categories and predicate). **c**, We analyse the occurrence of various abstract structures in our programs (see 'Game dataset analysis methods' section in the Methods for details). Red: the five most common structures cover almost half (47.5%) of total occurrences, showing extensive compositional reuse. The three most common structures combine into simple ball-to-bin throwing preference ((1), structure indices in square brackets). Purple: other structures are reused fewer times, covering most remaining occurrences (another 40.5%). These rarer structures allow for creating more complex throwing elements, constraining where the player throws the ball from (2,3) or to (3). Blue: exactly half of the structures (63/126) appear only once—this long tail of expressions offers evidence of creativity. The last throwing preference (4), specifying throwing a block from the rug onto the desk without moving off the rug or breaking any of the objects on the desk, uses two unique structures.

propose a single-player game to be played in the room. This design allowed participants to imagine and propose a wide range of playful goals, with the aim of game generation helping to make the resulting goals more concrete. We collected a dataset of 98 games, described by participants in natural language. In addition, we recorded full state–action traces of each participant's interactions with the environment, which we leveraged in later experiments (see 'Dataset collection methods' section in the Methods for additional details).

We then manually translated each game from natural language to programs in a DSL, inspired by language-of-thought models in computational cognitive science[20–24]. The DSL is used to model the semantics of games in our dataset, independent of the exact natural language

phrasing. Although the translation from natural language to DSL is unlikely to be lossless, we aim to capture the core semantics of the rich and generative structure of human goals with these relatively simple programs. This DSL was derived from the planning domain definition language (PDDL[25]), which offers a basic representation for specifying goals (that is, end states of plans) and preferences (that is, other costs to optimize while planning). Each program in the DSL contains two mandatory sections: gameplay preferences describing how a game is played, and scoring rules specifying how to determine a player's score based on the satisfaction of the game's preferences. Game programs may also contain optional setup instructions and terminal conditions (see Supplementary Information L for the full DSL).

Our choice to represent games as programs allows us to quantitatively analyse their structure and fundamental components. We found that people recruit an intuitive physical common sense when creating games (Fig. 2b; see Extended Data Fig. 2 for a detailed breakdown and 'Game dataset analysis methods' section in the Methods for details). For instance, if an object is thrown, it is probably a ball, and if an object is stacked, it is probably a block—and although a few participants specified games involving throwing blocks, none attempted to stack balls. Similarly, participants did not specify throwing cumbersome objects (such as the laptop or chair), and a participant who specified throwing a large 'beach ball' clarified that it should land on the bin (as the ball does not fit within the bin). We also observed evidence of both compositionality (common structure reuse) and creativity (preponderance of unique structures) across our participants, summarized in Fig. 2c (see 'Game dataset analysis methods' section in the Methods for details). Counting occurrences of grammatical structures while abstracting over the identities of individual objects—that is, treating the modal expressions 'the agent holds a block' (once (agent_holds block)) and 'the agent holds a ball' (once (agent_holds ball)) the same—we find the five most common structures cover almost half of the total observations, showing how representing goals as programs can reveal shared, compositional substructure. At the other end of the distribution, we also observe a long tail emblematic of creativity, as one-half of the unique structures we count appear exactly once. Despite not being explicitly prompted to generate novel or creative games, many participants proposed entirely unique gameplay ideas, encouraging us that our experimental paradigm elicits rich and creative goal creation.

Modelling results

We next develop a computational model to synthesize human-like goals. Guided by insights from our behavioural analyses, we design our model to explicitly leverage cognitive capacities that people seem to recruit in creating goals. Our GPG model (illustrated in Fig. 3) operates over a high-dimensional program space and learns how to generate goals maximizing a fitness measure. Upon entering a new environment, people can create goals without extensive data-driven demonstrations; therefore, we aim for a model that can similarly generate goals without a large number of examples. The GPG consists of two main elements: a fitness function and a search procedure. The fitness function (learned from data) attempts to quantify human likeness over the space of goal programs (Fig. 3a), such that a higher score indicates a better generated goal (Fig. 3b). The search procedure generates diverse samples that maximize this fitness function (Fig. 3c). As a framework, the GPG model is committed to the idea of evaluating the quality of goals-as-programs with a learned objective function and less so to the specific algorithms used for optimization and search.

The fitness function $f(g) = \boldsymbol{\theta} \cdot \boldsymbol{\varphi}(g)$ maps $f : \mathcal{G} \to \mathbb{R}$ from a game $g \in \mathcal{G}$ to a real-valued score that aims to encode its human-likeness (Fig. 3b). We transform each game into an 89-dimensional vector of features that capture properties relating to structure (for example, the size and depth of its syntax tree), logic (for example, whether any expressions are redundant) or goal semantics (for example, the extent to which different parts of the goal are interrelated). We leverage our programmatic representation of goals to automate this feature extraction process (see 'Fitness function methods' section in the Methods for details). In this implementation, parameter learning of feature weights $\boldsymbol{\theta}$ proceeds in a contrastive fashion[26,27] by optimizing for the difference in scores between our set of human-generated games and a substantially larger set of corrupted (that is, lower quality) games obtained through random tree regrowth[21] on our dataset (see Fig. 3b and details in 'Fitness function methods' section in the Methods).

This learned fitness function then guides an evolutionary search procedure to generate novel games (Fig. 3c). Broadly inspired by work in genetic programming, we use a quality-diversity algorithm[28,29] called MAP-Elites[30] to generate a set of samples that widely cover the space of programs in addition to optimizing the fitness function. The details of our implementation, including the particular behavioural characteristics used for maintaining sample diversity and structure our search of program space, are available in 'MAP-Elites methods' section in the Methods.

Our model includes several components that explicitly proxy cognitive capacities, such as features representing physical common sense (estimating predicate feasibility from play data) and recombination operators that explicitly leverage compositionality (the crossover operation that recombines programs). We describe a few of these components and how we ablated their contribution to our model in 'Ablation methods' section in the Methods.

Generated games

GPG produces a variety of outputs that range from variants of simple games in our reference dataset to games in entirely new regions of program space. In Fig. 4, we show examples of model outputs alongside the human-generated games that occupy the same 'niche' as defined by the MAP-Elites algorithm (see 'MAP-Elites methods' section in the Methods for details). We call generated games that occupy the same niche as a human game matched and those that do not unmatched. In the first pair (Fig. 4, left), the model proposes an original block-stacking objective: where the human participant created a tower, the model asks to stack three blocks all on the same taller block. The second and third pairs (Fig. 4, middle and right) demonstrate the model's ability to propose throwing games. In both cases, the model proposes interesting detailed objectives, some unseen in our training set (for example, throwing balls onto the top shelf or desk), that match the niche of the participant games by having the same high-level configuration. However, the purpose of certain minor elements in generated games tends to be less intuitively obvious (for example, the scoring condition in the left-most generated game, which arbitrarily multiplies the number of satisfactions by 0.4). Our model also produces unmatched games that occupy niches without corresponding human games (Fig. 5). These include unusual combinations of throwing and block-stacking (Fig. 5, left), a game that combines ball throwing and small object placement (Fig. 5, middle) and a game that offers a collection of varied block-stacking objectives (all-on-one, a T-shape and a tower; Fig. 5, right). Although these programs represent creative goals, with preferences that are each individually sensible, their components sometimes fail to combine into a coherent whole (for example, the golf ball throwing and block placement elements in Fig. 5, left, which do not intuitively form a cohesive game).

Quantitatively, Extended Data Fig. 3 shows that the GPG quickly produces games with fitness scores in the range of human samples and does so across many of the niches defined by our search procedure. Of the 2,000 programs we report, 1,889 programs (94.45%) exceed the fitness score of the least fit real game, and exactly half (1,000) exceed the fitness of the median human game. This demonstrates that our search procedure successfully finds high-fitness samples across much of the range of variation defined by our behavioural characteristics. To the extent that our fitness function captures human likeness, our model produces human-like games; we next use human evaluators to extrinsically test our model.

https://doi.org/10.1038/s42256-025-00981-4

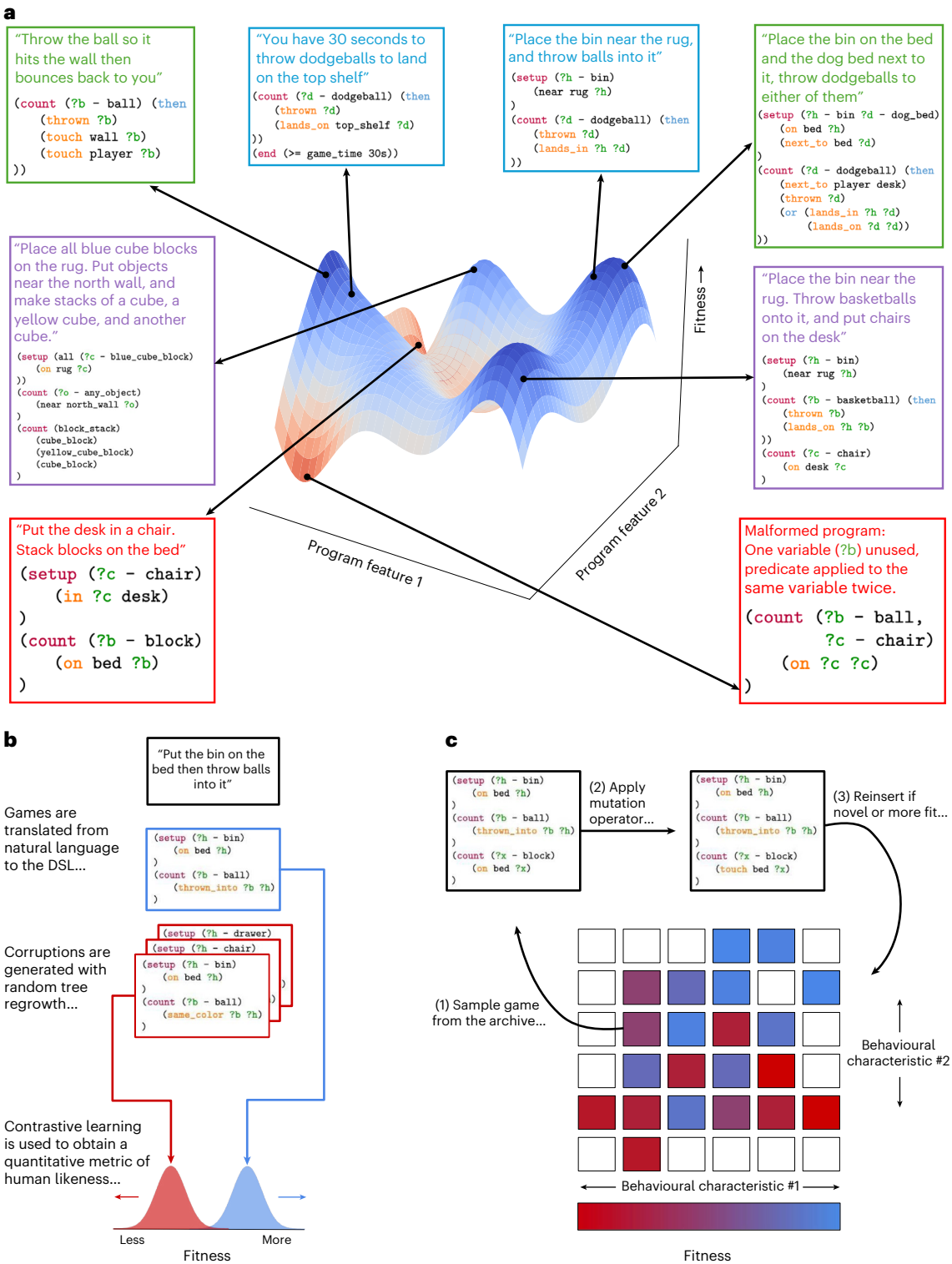


Fig. 3 | GPG model. a, Overview. Our model operates on programs in a high-dimensional space (visualized in two dimensions). We learn a fitness metric (*Z* axis) capturing desirable aspects of programs using a dataset of human-created goals (highlighted in green). Our model then generates diverse new samples maximizing the fitness measure, some matched to participant-created goal programs on diversity criteria (in blue) and other unmatched novel goals (in purple). These programs stand in contrast to potential failure modes, such as generating programs that are malformed or semantically incoherent (in red). All (non-red) goals in this figure were created by participants in our experiment or our model; see Supplementary Fig. 1 for their full representations in our DSL. **b**, Parameter learning. We contrastively learn a quantitative measure of fitness (the *Z* axis in **a**) by maximizing the distance between human-generated exemplar games and a set of corruptions obtained through random tree regrowth. **c**, Search. This measure is then used as the basis for quality-diversity optimization using MAP-Elites. The algorithm maintains an archive of games that differ across phenotypic 'behavioural characteristics'. At each step, a game is randomly sampled from the archive (1), randomly mutated (2) and reevaluated for fitness and its position in the archive. It is added to the archive only if it would occupy a previously empty position or if it is more fit than the current occupant (3).

https://doi.org/10.1038/s42256-025-00981-4

Participant game #14 (36.491)

Gameplay: stack a flat block on a bridge block, then place a tall cylindrical block on the flat block, followed by a cube block on the tall cylindrical block, and finally a pyramid block on the cube block.

Scoring: you get 10 points for each different set of blocks you have successfully stacked in this order by the end of the game

```
(define (game game-14) (:domain medium-objects-room-v1)
(:constraints
  (and
    (preference castleBuilt
      (exists (?b - bridge_block ?f - flat_block ?t -
          tall_cylindrical_block ?c - cube_block ?p -
          pyramid_block)
        (at-end
          (and
            (on ?b ?f)
            (on ?f ?t)
            (on ?t ?c)
            (on ?c ?p)
))))))
(:scoring
  (* 10 (count-once-per-objects castleBuilt))
))
```

Participant game #31 (37.338)

Gameplay: throw a ball so that it touches a wall and then either catch it or touch it

Scoring: you get 1 point for each time you successfully throw the ball, it touches a wall, and you are either holding it again or touching it after its flight

```
(define (game game-31) (:domain many-objects-room-v1)
(:constraints
  (and
    (preference ballThrownToWallToAgent
      (exists (?b - ball ?w - wall)
        (then
          (once (agent_holds ?b))
          (hold-while (and (not (agent_holds ?b)) (in_motion
              ?b)) (touch ?w ?b))
          (once (or (agent_holds ?b) (touch agent ?b)))
)))))
(:scoring
  (count ballThrownToWallToAgent)
))
```

Participant game #40 (36.152)

Setup: Place a green golf ball near the door and ensure it remains there for the entire game. During the game, place at least one dodgeball near the green golf ball.

Gameplay: While standing next to the green golf ball and the door, throw dodgeballs with the goal of getting them to stop inside a hexagonal bin.

Terminal: The game ends when you have thrown the same dodgeball and it has stopped moving more than once, or when you have thrown and stopped at least three different objects.

Scoring: You earn 10 points for each different object that you successfully throw into the hexagonal bin.

```
(define (game game-40) (:domain many-objects-room-v1)
(:setup
  (exists (?g - golfball_green)
    (and
      (game-conserved
        (near door ?g)
      )
      (forall (?d - dodgeball)
        (game-optional
          (near ?d ?g)
)))))
(:constraints
  (and
    (forall (?d - dodgeball)
      (and
        (preference dodgeballThrownToBin
          (exists (?h - hexagonal_bin ?g - golfball_green)
            (then
              (once (and (adjacent ?g agent) (adjacent door
                  agent) (agent_holds ?d)))
              (hold (and (in_motion ?d) (not (agent_holds ?d))
                  ))
              (once (and (not (in_motion ?d)) (in ?h ?d)))
          )
        )
      )
      (preference throwAttemptFromDoor
        (exists (?g - golfball_green)
          (then
            (once (and (adjacent ?g agent) (adjacent door
                agent) (agent_holds ?d)))
            (hold (and (in_motion ?d) (not (agent_holds ?d))
                ))
            (once (not (in_motion ?d)))
)))))))
(:terminal
  (or
    (> (external-forall-maximize (count throwAttemptFromDoor)
        ) 1)
    (>= (count-once-per-objects throwAttemptFromDoor) 3)
))
(:scoring
  (* 10 (count-once-per-objects dodgeballThrownToBin))
))
```

Matched model game (36.994)

Gameplay: stack three blocks on top of a tall rectangular block, with two of the stacked blocks being the same type as the tall rectangular block.

Scoring: your score is 1.4 times the number of such stacks you have at the end of the game.

```
(define (game gpg-8176-22-0) (:domain many-objects-room-v1)
(:constraints
  (and
    (preference preference0
      (exists (?v0 - block ?v1 - block ?v2 -
          tall_rectangular_block ?v3 - block)
        (at-end
          (and
            (on ?v2 ?v0)
            (on ?v2 ?v1)
            (on ?v2 ?v3)
            (same_type ?v0 ?v2)
            (same_type ?v1 ?v2)
))))))
(:scoring
  (+ (* 0.4 (count preference0))
    (count preference0)
)))
```

Matched model game (37.324)

Gameplay: throw dodgeballs so that they land and come to rest on the top shelf

Terminal: the game ends after 30 seconds

Scoring: you get 1 point for each dodgeball that is resting on the top shelf at the end of the game

```
(define (game gpg-8176-22-0) (:domain many-objects-room-v1)
(:constraints
  (and
    (preference preference0
      (exists (?v0 - block ?v1 - block ?v2 -
          tall_rectangular_block ?v3 - block)
        (at-end
          (and
            (on ?v2 ?v0)
            (on ?v2 ?v1)
            (on ?v2 ?v3)
            (same_type ?v0 ?v2)
            (same_type ?v1 ?v2)
))))))
(:scoring
  (+ (* 0.4 (count preference0))
    (count preference0)
)))
```

Matched model game (37.020)

Setup: Place a hexagonal bin near the rug and ensure it remains there for the entire game.

Gameplay: Throw dodgeballs aiming to land them on the desk or inside the hexagonal bin.

Scoring: You earn points for each dodgeball that comes to rest either on the desk or inside the hexagonal bin. Your final score is the sum of these points.

```
(define (game gpg-8178-6-0) (:domain medium-objects-room-v1)
(:setup
  (exists (?v0 - hexagonal_bin)
    (game-conserved
      (near rug ?v0)
)))
(:constraints
  (and
    (preference preference0
      (exists (?v0 - dodgeball)
        (then
          (once (agent_holds ?v0))
          (hold (and (not (agent_holds ?v0)) (in_motion ?v0)))
          (once (and (not (in_motion ?v0)) (on desk ?v0)))
      )
    )
    (preference preference1
      (exists (?v0 - hexagonal_bin ?v1 - dodgeball)
        (then
          (once (agent_holds ?v1))
          (hold (and (not (agent_holds ?v1)) (in_motion ?v1)))
          (once (and (not (in_motion ?v1)) (in ?v0 ?v1)))
)))))
(:scoring
  (+ (count preference1) (count preference0))
))
```

Fig. 4 | GPG model produces simple, coherent, human-like games. Each pair of games in a column has the same set of MAP-Elites behavioural characteristics (a real participant-created game and the corresponding matched model-generated one). The fitness score assigned by the model to each game is shown in parentheses. Natural language descriptions are generated through automated back-translation from programs (see Supplementary Information F for details). To ascertain that the model-generated programs are distinct from training set examples, we also provide in Supplementary Fig. 2 the most similar real exemplar using an edit distance; see Supplementary Information G for details.

https://doi.org/10.1038/s42256-025-00981-4

Unmatched model sample (36.066)

Setup: place a hexagonal bin near the north wall and make sure it stays there throughout the game.

Gameplay: throw golfballs aiming to have them stop on and inside the hexagonal bin, and stack blocks so that each has three cube blocks on top, with one cube block being the same type as the block it's on.

Scoring: you score points based on the number of correctly stacked blocks minus four times the number of golfballs that stop on and in side the hexagonal bin.

```
(define (game gpg-8172-48-1) (:domain many-objects-room-v1)
(:setup
  (exists (?v0 - hexagonal_bin)
    (game-conserved
      (near north_wall ?v0)
)))
(:constraints
  (and
    (preference preference0
      (exists (?v0 - hexagonal_bin ?v1 - golfball)
        (then
          (once (agent_holds ?v1))
          (hold (and (not (agent_holds ?v1)) (in_motion ?v1)))
          (once (and (on ?v0 ?v1) (not (in_motion ?v1)) (in
              ?v0 ?v1)))
      )
    )
    (preference preference1
      (exists (?v0 - cube_block ?v2 - cube_block ?v3 - block
          ?v4 - cube_block)
        (at-end
          (and
            (on ?v3 ?v0)
            (on ?v3 ?v2)
            (on ?v3 ?v4)
            (same_type ?v4 ?v3)
))))))
(:scoring
  (+ (* -4 (count preference0))
    (count preference1)
)))
```

Unmatched model sample (36.881)

Gameplay: throw dodgeballs and place credit cards or CDs into a hexagonal bin

Scoring: you get 40 points for each dodgeball that ends up in the hexagonal bin multiplied by the number of credit cards or CDs in the bin, plus 1 point for each dodgeball thrown regardless of where it lands.

```
(define (game gpg-8189-236-1) (:domain many-objects-room-v1)
(:constraints
  (and
    (preference preference0
      (exists (?v0 - hexagonal_bin ?v1 - dodgeball)
        (then
          (once (agent_holds ?v1))
          (hold (and (not (agent_holds ?v1)) (in_motion ?v1)))
          (once (and (not (in_motion ?v1)) (in ?v0 ?v1)))
        )
      )
    )
    (preference preference1
      (exists (?v1 - dodgeball)
        (then
          (once (agent_holds ?v1))
          (hold (and (not (agent_holds ?v1)) (in_motion ?v1)))
          (once (not (in_motion ?v1)))
        )
      )
    )
    (preference preference2
      (exists (?v2 - (either credit_card cd) ?v3 -
          hexagonal_bin)
        (at-end
          (in ?v3 ?v2)
)))))
(:scoring
  (+ (* 40 (count preference0) (count preference2))
    (count preference1)
)))
```

Unmatched model sample (35.872)

Gameplay: Stack blocks in specific configurations

Scoring: You get 1 point for each stack where one cube block is on top of another cube block with a tall rectangular block on the same cube block. You also get 1 point for each stack where a cube block is on top of a tall rectangular block, which is on top of another cube block, with an additional cube block on top of the tall rectangular block, provided the bottom cube block is the same type as the tall rectangular block. Additionally, you get 1 point for each stack where a tall rectangular block is on top of a cube block, which is on top of another cube block, and the top cube block is the same type as a third cube block. Your final score is the sum of points from these three configurations.

```
(define (game gpg-8127-244-0) (:domain many-objects-room-v1)
(:constraints
  (and
    (preference preference0
      (exists (?v0 - cube_block ?v1 - cube_block ?v2 -
          tall_rectangular_block ?v3 - cube_block)
        (at-end
          (and
            (on ?v3 ?v0)
            (on ?v3 ?v2)
            (on ?v3 ?v1)
          )
        )
      )
    )
    (preference preference1
      (exists (?v0 - cube_block ?v1 - cube_block ?v2 -
          tall_rectangular_block ?v3 - cube_block)
        (at-end
          (and
            (on ?v2 ?v0)
            (on ?v1 ?v2)
            (on ?v2 ?v3)
            (same_type ?v1 ?v2)
          )
        )
      )
    )
    (preference preference2
      (exists (?v0 - cube_block ?v1 - cube_block ?v2 -
          tall_rectangular_block ?v3 - cube_block)
        (at-end
          (and
            (on ?v3 ?v2)
            (on ?v0 ?v3)
            (same_type ?v3 ?v1)
))))))
(:scoring
  (+ (* 1 (count preference1) (count preference0))
    (count preference2)
)))
```

Fig. 5 | GPG model produces interesting, novel goals. Each of the three games below has high fitness and fills an unmatched cell in the MAP-Elites archive, with no corresponding human game in our dataset. The fitness score assigned by the model to each game is shown in parentheses.

Human evaluations

To systematically and extrinsically evaluate our model, we performed human evaluations using a second set of human participants ($n = 100$; see Extended Data Fig. 4 for the evaluation interface and 'Human evaluation methods' section in the Methods for details). Evaluated games belonged to one of three different categories mentioned above: real participant-created games from our behavioural experiment, or matched or unmatched model-generated games (see Fig. 3 for category definitions; games in Figs. 4 and 5 were included; see 'Human evaluation methods' section in the Methods for details). Participants evaluated three games in each category above (without knowing their categories) in a randomized order and provided Likert scale ratings on each game for seven measures, including human likeness, fun and creativity. Our final dataset includes 892 participant-game evaluations, each with a rating for all seven measures.

To analyse these results, we performed a mixed-effects regression analysis (we provide the raw score means and non-parametric statistical tests in Extended Data Table 1). We fit independent models using each of the seven attributes we asked our human evaluators to judge as the dependent variables. We examine two questions: (1) Are there any systematic differences between game categories? (2) Does our fitness function, learned from corrupting samples in program space, capture any human-evaluated qualities of the games? For both questions, we fit mixed-effects models that include a fixed effect for membership in the real and matched groups (treating the unmatched group as a baseline) and random effects for the participants and individual games. For the second question, we also include a fixed effect for the fitness score (see 'Human evaluation methods' section in the Methods and Supplementary Table 5 for full details).

To answer the first question, we use the method of estimated marginal means to compare the difference in scores between each pair of categories, averaging out the random effects (Table 1, visually summarized in Extended Data Fig. 5; see Supplementary Information I.3 for details). Participants respond similarly to the real and matched games, with no statistically significant differences in the estimated mean scores across all seven attributes. Meanwhile, the unmatched games differ on several attributes. Participants judge them to be less easily understood and fun to play than real games and less human-like and fun to watch than both matched and real samples. We observe similar results using non-parametric statistical tests (Extended Data Table 1). One potential explanation for the apparent similarity between matched and real games is that the former simply replicate the latter in form and function. We examined this question and found that matched and real games have substantial functional differences (see summary in Extended Data Fig. 6, details in Supplementary Information I.4 and methodological details in 'Sample similarity comparison methods' section in the Methods).

Next, we analyse the mixed-effect models fit with a fixed effect of fitness scores. First, we replicate the effects of the fitness-less regressions; we continue observing no significant differences between the real and matched groups, and several significant differences between both of those and the unmatched group (Supplementary Table 6). Next, we examine the fitted coefficients in these regressions (summarized in Extended Data Table 2 and visualized in Extended Data Fig. 7). We find that our fitness function captures many of the evaluated

Table 1 | Mixed-model marginal means comparison summary

	Real – Matched			Real – Unmatched			Matched – Unmatched		
Attribute	**Difference±SE**	***Z***	***P* value**	**Difference±SE**	***Z***	***P* value**	**Difference±SE**	***Z***	***P* value**
Understandable↑	−0.001 ± 0.331	−0.003	1.000	1.042 ± 0.332	3.138	4.837×10^{-3}**	1.042 ± 0.333	3.133	4.927×10^{-3}**
Fun to play↑	0.143 ± 0.266	0.538	0.853	1.020 ± 0.274	3.722	5.791×10^{-4}***	0.877 ± 0.273	3.210	3.791×10^{-3}**
Fun to watch↑	0.135 ± 0.250	0.542	0.850	0.892 ± 0.259	3.446	1.650×10^{-3}**	0.757 ± 0.257	2.944	9.076×10^{-3}**
Helpful[a]	0.016 ± 0.159	0.097	0.995	0.251 ± 0.165	1.521	0.281	0.236 ± 0.165	1.426	0.328
Difficult$^{\downarrow}_{\uparrow}$	−0.200 ± 0.357	−0.559	0.842	−0.194 ± 0.361	−0.538	0.853	0.006 ± 0.361	0.016	1.000
Creative↑	0.228 ± 0.310	0.736	0.742	0.489 ± 0.316	1.548	0.269	0.261 ± 0.314	0.832	0.683
Human-like↑	0.199 ± 0.274	0.727	0.748	1.396 ± 0.283	4.927	2.495×10^{-6}***	1.197 ± 0.283	4.225	7.088×10^{-5}***

Evaluators do not distinguish between participant-created real and matched model games but do distinguish unmatched games from real (and marginally from matched ones). Participants responded to seven Likert questions on a 5-point scale, one for each attribute in the first column (see 'Human evaluation methods' section in the Methods). We found fairly low inter-rater agreement (Supplementary Information I), and so we centre our analysis on our fitted mixed-effects models (see 'Human evaluation methods' section in the Methods). We use the method of estimated (least-squares) marginal means to compare the three groups of games, accounting for the random effects fitted to particular games and human evaluators. We report two-sided *Z* significance tests adjusted using the Tukey method to control for the multiple difference tests within each attribute, as implemented in the emmeans package, with standard errors (SE) computed on the basis of the pooled residual standard deviations. *$P < 0.05$, **$P < 0.01$, ***$P < 0.001$ [a]The full measure description is 'helpful for interacting with the simulated environment'. In most measures, higher scores are better, indicated by the ↑, other than 'Difficult$^{\downarrow}_{\uparrow}$', in which 3 means 'appropriately difficult' and scores below and above indicate too easy and too hard, respectively.

attributes: higher fitness predicts higher ratings of understandability, fun to play and human likeness ($\beta_{\text{fit}} > 0$); conversely, higher fitness also predicts lower ratings of helpfulness, difficulty and creativity ($\beta_{\text{fit}} < 0$). Our positive findings are promising: they indicate that our fitness function, learned to maximize human likeness in a symbolic program space, also captures intuitive human notions of understandability and fun. Conversely, we view the negative relations as evidence of some degree of mode seeking: our fitness measure probably assigns the highest scores to the games most representative of the dataset at large. These modal games are plausibly neither particularly creative nor difficult, which means that participants might also find them less helpful for learning the details of the environment. To explore this, we highlight the highest-fitness games generated both by our model and by human participants in Supplementary Fig. 3 and observe the type of mode seeking we suggest above (see Supplementary Information H for details).

We also performed ablations of key model components that explicitly proxy some cognitive capacities we found our participants recruited (see details in 'Ablation methods' section in the Methods). To ablate physical common sense, we remove from our fitness function the two features that estimate the feasibility of a game's preferences by leveraging our database of participant–environment interactions. Analogously, we ablate the intuitive coherence we observe in human goals by removing the features that capture the coordination of gameplay elements between different sections. Ablating compositionality is more difficult, as our programmatic representation is inherently compositional. We do so by removing the crossover mutation operator used to generate new samples during MAP-Elites, which most explicitly leverages the compositional structure of games. In these three ablations, model performance degrades substantially, either in sample fitness scores or in goal plausibility, as estimated using our database of participant–environment interactions. We also report two other comparisons, one to a model sampling only from the probabilistic context-free grammar (PCFG) over our DSL (which performs much worse) and one to a model optimizing a fitness function trained on a subset of our full dataset (which performs comparably and generalizes to the held-out regions of program space). See Supplementary Information J for further details.

Discussion

Goals are a critical aspect of human cognition and, in fact, the starting place for many models of human behaviour. However, the representation of goals is often impoverished. In this Article, we proposed a new framework for understanding a particular class of human goals as reward-producing programs, as a stepping-stone towards a broader understanding of goal representation and generation. To evaluate this framework, we developed an interactive experiment in which participants created playful goals, operationalized as games to be played in a virtual environment. By analysing the program-based translation of these games, we highlighted several cognitive capacities recruited by our participants, such as physical common sense and compositionality. These capacities, in turn, informed our modelling efforts. We then built a computational model that learns from a small dataset of games and generates coherent, novel goals, where those sampled from partitions of program space occupied by human examples were deemed human-like according to human evaluators.

This work unites various strands of research in cognitive science, artificial intelligence and game design. First, we build on substantial literature studying the psychology of goals[1–3,10,13] by offering a specific representational hypothesis, in contrast with previous approaches to defining goals. We emphasize open-ended goal creation given that generating new exemplars is a core capacity of human conceptual representations[31] and the utility of games in the study of cognition[32]. Our work also relates to goal-conditioned reinforcement learning[33], and we aim to improve on the goal representations used for such agents that tend to lack the variety and richness of human-created goals (ref. 34, chapter 7). In this respect, our proposal attempts to abstract from the reward functions and simpler goals used in many reinforcement learning tasks. Our goal program interpreter conceptually draws on the notion of reward machines introduced in ref. 35. Finally, we are inspired by the automatic game design literature, such as synthesizing board-game variants[36–38] or simple video games[39–42]. Unlike our approach, these efforts often optimize program synthesis for some heuristic notion of fun[38,39] rather than explicitly modelling human-like game generation.

Our framework is committed to the representation of goals as reward-producing programs: computationally executable mappings from behaviour to indications of progress towards a goal, which we term reward. We find it crucial that these programs capture the rich, temporally extended nature of goals people create and that they facilitate the flexible and compositional creation that people seem to engage in[31,43]. We hope that this proposal is useful to understanding goal representation and generation, not that it losslessly explains every source of variation in human-created and reported goals. We note that we currently study goal generation through game creation, and while many games have players take on goals[18], not all goals are fully equivalent or

isomorphic to games. We believe that our representational hypothesis also has merit for additional kinds of goals, such as the goals created in joint play between multiple children or adults (such as tag or dodgeball) or the objectives a person exploring a new environment might set for themselves (for instance, how to turn on the light at an AirBnB without bumping into anything). While we expect the general GPG framework to accommodate such goals in different domains, doing so would certainly entail changes to the specific syntax and semantics of the programmatic representations. There are, however, types and aspects of goals that might complicate the general procedure of translation into programs. For instance, subjectivity, which may be modelled as listeners forming different representations of the same utterance, might require breaking the assumption that each natural language goal corresponds to a single program. Similarly, it is not obvious how to represent truly abstract goals such as 'I want to do well in school' as a well-formed program. In both cases, an avenue forward might be grounding not to a single program but to distributions over programs or programs with stochastic elements (for example, as suggested in the Rational Meaning Construction framework[24]). We are excited for future work to continue studying open-ended goal generation in other domains and explore how readily other types of goals map onto programs.

There are limitations of the current work that we hope to address in future work. Our model strongly relies on its approach to sample diversity, which arises from the choice of behavioural characteristics that define the axes along which the MAP-Elites algorithm maintains diversity. In this work, we select behavioural characteristics based on notable gameplay components observed in our human dataset; future work could explore other techniques for maintaining diversity, including the automated selection of behavioural characteristics[44,45]. In addition, our fitness function ablation (reported in Supplementary Information J.5) reveals evidence of potential overfitting in our fitness function and highlights the limitation of fitting a single objective function to all participants. Future work could explore tuning versions of the objective to the individual preferences of particular participants. Our model currently does not account for any resource constraints people face in creating goals that may make compact goals easier to propose or maintain; we leave to future work the question of whether adding any length-related penalty recovers more human-like proposed goals. Finally, our compositionality ablation (reported in Supplementary Information J.2) is limited—as we collect a single playful goal from each participant, we can only compare compositionally between participants, rather than measuring component reuse within a single participant's goals. Within-participant reuse may offer further evidence of how humans creatively recombine components to compose novel goals.

Our current features approximating intuitive physical common sense are indirect, using participant interactions with the environment to estimate feasibility. Future approaches could integrate planning or physical simulation to improve our model's understanding of physics[46,47]. Our model is currently limited to a single kind of common sense, the intuitive physical one; other environments may require leveraging similar knowledge from other domains, such as intuitive social models of agency and theory of mind. Finally, our model is inherently coupled to the environment and dataset we collected—particularly given the engineering effort to instantiate various types of knowledge. This approach has some distinct advantages: we can isolate various cognitive capacities, interpret their contribution to our fitness measure (Supplementary Information C.1) and ablate their roles (Supplementary Information J). Simultaneously, some of the challenges our model faces (such as coherence between program components) might be alleviated by incorporating natural language or by leveraging the capabilities of large language models to write code and adapt to in-context instructions. Language models could also alleviate our current reliance on manual translations from participant game descriptions to the proposed mental language of goal programs (see ref. 24 for a discussion on using language to construct meaning through programs, and ref. 48 for building programs to act as world models).

We see two particularly promising ways in which our representational framework could be used going forward. First, there is increasing interest in building artificial agents that can flexibly explore and generalize across environments[49,50]. The autotelic perspective argues that empowering agents to propose and pursue self-generated goals is a fruitful way to improve their ability to generalization[34]. However, goals in such systems are often derived from agent or object positions[51,52], short natural language descriptions[53,54] or limited temporally aware mechanisms[55,56]—all impoverished when compared with the diverse goals humans flexibly create. Closely related to our notion of representing goals by programs, recent work proposes to directly synthesize reward functions[57] or environment descriptions[58] using code-generation models. We are excited for future work to empower artificial agents with richer goals that reflect human-like novelty and difficulty, for two specific reasons. First, we believe that access to complex and varied goals would enable agents to learn flexible representations of their environments that support higher behavioural adaptability[13]. Second, we view compositional goal production as facilitating effective exploration of unseen goals[59] (see ref. 60 for a discussion of generalization and exploration). We also note that our current approach estimates goal fitness without considering additional higher-level objectives that might guide goal generation. Prior literature offers curiosity[61,62], empowerment[63–65], information gain[66,67], novelty[62,68] and learning progress[69,70] as compelling potential objectives. Future work could instantiate goal generators that consider these objectives as auxiliary terms to the fitness function and compare the behaviours that arise in artificial agents through pursuing them.

If we are to understand goals as programs, our proposed framework may also help advance our understanding of intuitive psychology and goal inference[71–73]. Previous work proposed that our ability to understand other people's goals, as part of our theory of mind, operates through inverse reinforcement learning: inferring an agent's reward from observing their behaviour[74]. Many prior approaches eschew goals entirely, using some function approximator (for example, a neural network) to estimate reward, resulting in an uninterpretable estimator that can struggle to generalize[75]. We envision leveraging our goal programs as a prior distribution for a Bayesian theory of mind[76] approach, scaling up previous approaches that relied on a small number of predefined goals[77], to create models that would parse an agent's behaviour and provide an interpretable, semantically explicit estimate of their goal[78]. Applying our framework to either of these proposed problems would offer a substantial long-term challenge building on the work we present in this Article. Nevertheless, we see an exciting prospect to leverage this approach to improve the understanding of human goals and endow machines with human-like goal concepts and capabilities.

Methods

Dataset collection methods

Experimental design. After an informed consent form and instructions quiz, participants completed a tutorial designed to familiarize them with the controls for our environment. After successfully completing the tutorial, participants were randomly assigned to one of three variations of the main experiment room, with the same structure but different amounts of available toys and objects. Participants were then free to explore this new room until they had a game ready, and could freely reset it to its initial state in the meantime. Participants were asked to create games with the following restrictions: single-player, requiring no additional space or objects that they do not see in the room, and including a scoring system. Although the latter constraint may seem limiting, we note that any arbitrary goal can be scored by rewarding the achievement of the goal.

Ethics oversight. The study was performed under the New York University institutional review board study titled 'Active learning in dynamic task environments', IRB-FY2016-231, under the principal investigator T.M.G.

Dataset collection. Participants then reported their game in natural language in three text boxes, one of which was optional (Extended Data Fig. 1). The optional first one allowed specifying whether there was any setup or preparation required to get the room from its default initial state to one that would allow playing the game (for example, placing the bin on the bed). The second text box allowed participants to describe the game's gameplay, and the third offered space to describe the scoring rules. To encourage participants to imagine playing their game, they were also asked to report their perceived difficulty level and how many points they thought they might score if they played it. Participants then had a chance to play their game and revise it should they want to; if participants opted to revise their games, we analysed the revised ones. We contacted 192 participants via Prolific[79] of whom 114 finished the experiment and another 12 were paid due to technical difficulties. Participants were paid a base rate of US$10 and received a US$2 bonus if their game satisfied the required constraints. Successful participants took 44.4 min on average, with a standard deviation of 23.3 min. We then excluded eight games that did not satisfy the constraints we posed on participants, six duplicates (including some due to technical difficulties from participants who restarted the experiment) and six other games that were unclear or underspecified. After accounting for 2 other games we opted to avoid modelling owing to their complexity (one referring directly to the game interface and controls, and another describing several games or levels in the single description we collected), we arrived at our final dataset of 98 games. We acknowledge the potential arbitrariness of manually translating from natural language to our program representations; we attempted to be maximally faithful to the descriptions and excluded participants whose games required too much subjectivity or interpretation.

Interaction traces. In addition to the game descriptions in natural language, we record traces of participants' interactions with the environment. We record state–action traces to allow us to replay and examine how participants interact with our environment. We record separate traces for each different segment of the experiment (before creating the game, while reporting their game, playing their game and after editing their game) and for each time the participant resets the environment within each segment. We end up with 382 total such traces. Our primary use for them is in implementing a reward machine, an interpreter for our goal programs, which parses a goal program into a state machine and iterates through a trace to emit the score of that trace under the goal. We use a limited version of this in our fitness features (see 'Fitness function methods' section for additional details) and in some of our model evaluations and ablations (see Supplementary Information J for additional details).

Natural language to DSL translation. We manually translated the games we collected from participants to programs in a DSL we created. We examined the natural language descriptions our participants provided to identify recurring semantic components, which we then mapped onto elements in our DSL, iterating between translating more programs and updating the DSL grammar. We began by attempting to translate directly into PDDL[25], which offers a basic representation for specifying planning problems, but deviated from it as we encountered game elements our participants specified with no clear PDDL analogues. We assume that the translation process is not lossless, as there are probably multiple natural language descriptions for each underlying set of game semantics and multiple programmatic encodings of vague natural language descriptions; however, we aimed to develop representations that capture the core semantics of the rich, generative and creative structure in goals. We also perform some analyses to validate the extent to which these translated programs capture semantic concepts that were intended by participants, which we report in Supplementary Information B.

Goal program interpreter. Inspired by the reward machine proposed by ref. 80, we similarly implement an interpreter for the goal programs in our DSL. The interpreter parses a program in our DSL into a state machine. This state machine enumerates over environment states and participant actions emitted as a participant plays in our experiment (see 'Interaction traces' section above), tracks the participant's progress with respect to each program component (setup conditions, gameplay preferences and terminal conditions) and emits a reward according to the scoring conditions defined in the goal program. This allows us to ground programs to participant interactions and evaluate partial or complete fulfilment of the specified goal. We use this reward machine as part of our feature set (see 'Fitness function methods' section), to analyse functional similarity between programs our model generates and participant-created games (Extended Data Fig. 6) and to assess our manual translations of participant-provided descriptions (Supplementary Information B). Our current implementation of the interpreter covers the vast majority of predicates and grammar elements; we omitted grounding a few rarely used predicates owing to their complexity and lack of frequency. In these cases, we attempted to ensure that our implementation would be biased towards false negatives rather than false positives—we would rather fail to count an interaction that occurred than count interactions that did not occur.

Game dataset analysis methods

Common sense through predicate role-filler analysis. We analyse predicate role-filler occurrences, coarsening individual objects to higher-level categories (see the legend on the right of Fig. 2b). To split between the two panels of Fig. 2b, we categorize each game by whether it includes the following motifs: throwing (for example, balls into a bin), stacking (for example, blocks in a building), organizing (for example, placing objects in specified places) or other. We split the figure into games involving only throwing motifs (left panel) and games involving any other motifs, potentially in addition to throwing (right panel). In games involving only throwing (left panel), participants most often refer to balls, primarily checking whether or not the agent holds a ball or a ball is in motion (as part of quantifying the act of throwing). Other predicates are often used to specify some additional conditions on throwing (such as specifying the bin being on the bed or the agent being next to the desk) and are used with a variety of object categories. Conversely, in games involving other elements (right panel), we see blocks and the generic 'any_object' being used far more often, mostly in various placement and stacking constraints.

Compositionality and creativity through abstract structure occurrence. We analyse how often participant games make use of various grammatical structures to showcase both compositional reuse and long-tail creativity. Each structure involves a temporal modal (such as once or hold) and the predicate expression nested under it, such as (once (agent_holds ?b)), where ?b is a variable quantified earlier. We count structures, abstracting away specific variables and their types—so the expression above would be coarsened as (once (agent_holds <obj>)) and would be counted together with any other expression coarsened to this pattern. We encounter a total of 126 unique expressions in our dataset, the most common one with 62 occurrences being (hold (and (not (agent_holds <obj>)) (in_motion <obj>))), which maps loosely to 'find a sequence of states where an object is not held and is in motion'—that is, it is currently moving with the agent touching it, for instance while being thrown or rolled. Of the 126 expressions, exactly half (63) occur only once.

Fitness function methods

Fitness function form. We use the most direct mapping from feature values to a real-valued score as our fitness function: a learned, weighted linear combination of a set of features extracted programmatically from each game that is optimized to assign high scores to 'human-like' games and low scores to everything else. It is a function $f : \mathcal{G} \to \mathbb{R}$ that maps individual games $g \in \mathcal{G}$ to real-valued scores: $f(g) = \boldsymbol{\theta} \cdot \boldsymbol{\varphi}(g)$, where $\boldsymbol{\theta}$ is a learned vector of weights and $\phi : \mathcal{G} \to [0,1]^F$ is a feature extractor.

Feature extractor and feature set. The feature extractor $\boldsymbol{\varphi}$ represents each game as an 89-dimensional vector (that is $F = 89$). Each entry in the vector corresponds to a particular structural or semantic property of the game, from the size and depth of the syntax tree to the apparent feasibility of the game's preferences. We normalize the values of each property to fall within the unit interval by using the observed range of values in our dataset. Many features used in the fitness function are directly computable from the DSL representation of a game (for instance, properties of its syntax tree or the misuse of particular grammatical structures). While these features represent the majority of the 89 features used, we also implement two important sets of features that require additional computation.

The first of these are n-gram features that capture the mean log score of the game under a simple n-gram language model trained over the set of human-generated syntax trees. We fit n-gram models using stupid backoff[81] to account for missing n-grams, using the default discount factor of 0.4 reported in ref. 81. We compute these scores separately for each game section (that is, setup, preferences, terminal and scoring) and also for the game overall, resulting in five features.

The second set consists of two features that make use of an interpreter that parses game programs into reward machines[80]: finite-state machines that process a trace of player inputs and emit a reward whenever the particular scoring conditions of the game are met. The interpreter programmatically implements each of the predicates in the DSL, which allows us to construct a dataset of which objects were used to satisfy which predicates across our dataset of 382 human play traces. The two features query this database to get an approximate common sense measure of a game's feasibility, computing the proportion of a game's predicate–argument combinations that have been satisfied by human players in our dataset (one feature does this for individual predicates, while the other does this for Boolean logical expressions over predicates). Although these feasibility measures give a sense of whether the objectives of a game can be completed in the physical reality of the simulation, the limited nature of our play trace dataset means they are far from perfect proxies.

We developed our feature set starting from features used in similar prior work (for example, features representing the length and depth of the syntax tree[82]). We then fit a fitness function using the procedure described below and inspected the fittest games from our set of negative examples. We iteratively added features to account for mistakes our model made (flawed negatives with high fitness) and removed features that our fitness function seemed to ignore (by learning a weight with a low magnitude). The complete set of features used (and accompanying descriptions) is available in Supplementary Information C, with the most important features (by their learned weights) highlighted in Supplementary Information C.1.

Fitness function learning algorithm. To learn the weight vector $\boldsymbol{\theta}$, we take inspiration from the contrastive learning of energy-based models[26] with the objective of separating a set of positive examples (our set of human-generated games) from a set of negative examples (see a summary in Fig. 3b). To learn an effective fitness function, these negatives must be qualitatively worse than our set of human games without being trivially distinguishable from them. We generate a set of plausible negatives by corrupting games from our positive set. To corrupt a game, we select a random node in its syntax tree, delete the node and its children, and randomly resample a subtree according to the DSL grammar (illustrated in red in Fig. 3b). This tree-regrowth approach[21] generally produces subtrees that are syntactically valid but semantically out of place, with the severity of the corruption tending to correspond to the height of the resampled node in the syntax tree. To account for the variance in the difficulty of distinguishing between a given positive and negative example, we generate a large set of negatives: 1,024 for each of the 98 positives, for a total of 100,352 negatives.

We train the fitness function (that is, optimize $\boldsymbol{\theta}$) using a softmax loss, not unlike the minimum empirical error (MEE) loss used to train energy-based models[83] or the InfoNCE loss[84]. For a positive example g^+ and a set of negative examples $\{g_k^-\}, k \in \{1, 2, \cdots, K\}$,, we assign the loss

$$\mathcal{L}(g^+, \{g_k^-\}_1^K; \boldsymbol{\theta}) = -\log \frac{\exp(f_\theta(g^+))}{\exp(f_\theta(g^+)) + \sum_{k=1}^{K} \exp(f_\theta(g_k^-))}. \quad (1)$$

This loss encourages the model to assign higher fitness scores to the real games than the negative examples. Simultaneously, this loss provides a diminishing incentive to push negative fitness scores down as the distance between the positives and negatives increases, intuitively assigning higher loss to negative examples with fitness closer to the positive example's fitness. See Supplementary Information D for full details of our training and cross-validation setups.

Final fitness function. Note that, while we perform cross-validation for hyperparameter selection, once we fixed a set of fitness features and hyperparameters, we fit a final fitness function using our entire dataset (98 participant-created examples and their corresponding negatives). Given the minuscule human dataset we collected, we opted against holding out data from the final objective function to best guide our model's search process (but see Supplementary Information J.5 for a comparison with fitness function trained on a subset of our dataset).

MAP-Elites methods

MAP-Elites overview. MAP-Elites is a population-based, evolutionary algorithm that works by defining a set of behavioural characteristics: discrete-valued functions over genotypes (in our case, game programs in the DSL) that form the axes of a multidimensional archive of cells (see the overview in Fig. 3c). At each step, a game g is selected uniformly from among the individuals in the archive (Fig. 3c, step 1) and mutated to form a new game g' (Fig. 3c, step 2). The mutated g' is evaluated under both the fitness function f and each of the n behavioural characteristics $b_i : \mathcal{G} \to \{0, \ldots, k_i\}$ to determine which cell $c = [b_1(g), \ldots, b_n(g)]$ it occupies. If the cell is unoccupied, then g' enters the archive. Otherwise, it enters the archive (and replaces the previous occupant) only if its fitness is greater than the current occupant of the cell (Fig. 3c, step 3). In this way, the algorithm maintains an 'elite' for each possible combination of values under the behavioural characteristics.

Behavioural characteristics. Inspired by prior work on using MAP-Elites for procedural content generation[85], we define a set of integer-valued behavioural characteristics that each indicate how many preferences in each archive game match one of nine archetypal exemplar gameplay preferences (illustrated as the axes of the grid in Fig. 3c). These include several types of ball-throwing preferences, as well as ones capturing block-stacking, object-sorting and other miscellaneous activities. We also include two other features: one that captures whether the game includes a setup component and one that captures the total number of preferences. For additional details and descriptions of the exemplar preferences, see Supplementary Information E. We use nine exemplar preferences, in addition to these two other features, as a trade-off between covering many behaviours that participants demonstrate and avoiding exploding the size of the archive: as it is, the 11 total behavioural characteristics result in a total archive size of 2,000 games.

The 98 participant-created games in our dataset map onto 47 different archive cells; conversely, most archive cells (1,953, or 97.65%) have no corresponding participant-created exemplar.

Auxiliary coherence check. We include an auxiliary pseudo-behavioural characteristic that explicitly captures a few general coherence properties of games, which we use to help our model search the space of programs. This characteristic computes a conjunction of the values of 21 features, ones that we expect either all plausibly human-generated games to exhibit or none of them to exhibit (indeed, all participant-created programs in our dataset pass this check). These include features such as checking that all quantified variables are referenced at least once, that all game preferences defined are mentioned in the terminal or scoring conditions and that no logical expressions are tautological or redundant. This check does not use any information beyond the fitness features and serves as a mild additional inductive bias and structure for our search process.

We keep two copies of the 2,000-sample archive from the behavioural characteristics using the exemplar preferences above, one with samples passing this auxiliary check and the other with samples failing it. During the search process, we sample uniformly from both archives. Intuitively, this accomplishes two desiderata: (1) it forces the model to generate a sample in each archive cell that passes this check, and simultaneously, (2) it allows the model to better search the space of programs by also exploring high-fitness samples that fail this check. We consider as outputs of our final model only goals from the archive copy that pass this check, and those are the only ones we report in fitness-based and human evaluations. See Supplementary Information E for additional details.

Mutation operators. To mutate a game, we randomly select an operator from among the following: regrowing a random node and its children in its syntax tree; inserting and deleting the child of a node with multiple potential children; crossing over with the syntax tree of another randomly selected game; resampling the variables, initial conditions or final conditions used by a preference; and resampling the optional game sections (that is, setup and terminal conditions). We seed the initial archive by naively sampling candidates from the PCFG—not with real, human-participant-created games or corruptions thereof that were used to train the fitness function. Further details of the algorithm are available in Supplementary Information E.

Archive initialization. Our search process is not seeded from any real participant-created examples. Instead, we initialize the MAP-Elites archive with examples generated by sampling from the PCFG defining our DSL. We generate 1,024 initial samples, sort them by their fitness scores and add at most 128 of them to the archive. See Supplementary Information E for additional details.

Ablation methods

We ablate several components of our model that leverage cognitive capacities people appear to use when creating goals. We describe the components and briefly elaborate on their respective cognitive capacities below. We report the full ablations in Supplementary Information J.

Common sense. We offer evidence in Fig. 2 and our discussion of the behavioural results that participants seem to leverage (physical) common sense reasoning in their goal creation. The DSL we use to represent goals is underconstrained with respect to this type of common sense and allows one to generate expressions that are physically improbable or entirely impossible. To aid our model in generating physically plausible expressions, we include two fitness features that query a dataset of participant interactions with our environment (see 'Dataset collection methods' section) and score predicate expressions on whether or not any participants ever satisfied them in their play behaviour. We report the results of this ablation in Supplementary Information J.1, where we find that these features are crucial for our model.

Compositionality. We offer evidence of the way participants appear to recombine simple elements to create diverse games in Fig. 2. Compositionality is core to our DSL, as programs naturally offer the ability to compose expressions of the same type. We ablate this ability by removing the mutation operators that implement compositions. We first remove some of our custom resampling operations and then remove the crossover operation, which explicitly composes two programs in our archive to create two new candidates (see 'MAP-Elites methods' for additional details). We report the results of this ablation in Supplementary Information J.2, where we observe that the crossover operation is crucial for our model and that our custom operators offer a smaller but measurable effect.

Coherence. We observe that most participants create coherent goals that fit together without any explicit prompting to do so: different components of a goal tend to refer to one another and avoid disjointedness. After earlier versions of our model struggled with this type of higher-level coherence, we included several fitness features that attempt to measure it at different degrees of abstraction (see Supplementary Information J.3 for additional details and the full results). We find that including these features substantially improves the model-generated games.

PCFG-sampling-only baseline. To illustrate the necessity of a complex search process over the space of programs in our DSL, we created a baseline that repeatedly samples from the PCFG representing our grammar, with rule and terminal counts fitted to our human datasets. We match the total number of samples to the total number of candidates our full model generates in its search. We find both low occupancy rates (sampling from this prior fails to explore the space) and low fitness scores. See Supplementary Information J.4 for additional details and the full results.

Held-out data ablation. We perform a held-out evaluation of our model to evaluate how robust our procedure is to unobserved data. We split our dataset of 98 games into 20 test games and 78 training games and fit the fitness function only using those games (with the same set of fitness features as our full model). We then run our search to optimize the fitness function fitted to the partial data. We find the results comparable to our full model, both in overall fitness scores and when particularly examining the model-generated games corresponding to the held-out samples. See Supplementary Information J.5 for additional details and the full results.

Human evaluation methods

Evaluation dataset. We select games to be evaluated using the following procedure:

(1) Real: We include 30 participant-created games, each with a different set of behavioural characteristics—that is, each being considered different according to how our model searches through the space of games (see 'MAP-Elites methods' section for additional details).
(2) Matched: For each of the real games included above, we include the model-generated game from our final model from the corresponding MAP-Elites archive cell. Each of these games includes the same number of gameplay preferences as the corresponding real participant-created games, matching the same exemplar preferences.
(3) Unmatched: We also include 30 additional games from our model's archive. We sample these in a fashion that aims to be balanced across the different preference counts and usage of the different exemplar preferences. That said, given that human

games cover only 47 out of the 2,000 archive cells, that leaves 1,953 potential unmatched games to sample; it is difficult to know how representative our set of 30 (which is about 1.5%) is. We initially sampled 40 unmatched games and had participants evaluate 4. We then discovered that some of these model samples have drastically lower fitness scores from the real and matched samples. We therefore excluded evaluations of the ten lowest-fitness unmatched samples from our analyses to reduce the degree to which fitness scores confound our analyses.

We collected evaluations from $n = 100$ human participants, and our final dataset includes 892 participant-game evaluations, of which 300 are in the real category, 300 in the matched category and 292 in the unmatched category (due to the exclusions mentioned above).

GPT-4-based back-translation. Rather than ask participants to interpret our DSL, we use the GPT-4 (ref. 86) language model to perform a multistep back-translation from programs in our DSL to structured natural language. For fairness and consistency, we use this procedure on the real games in addition to the model-generated matched and unmatched games. We first apply a rule-based system to apply templates, translating expressions in the DSL to natural language sentence fragments. We then use GPT-4 to first map the templated fragments to a more natural language and then combine the description of each game component (setup, gameplay preferences, terminal conditions and scoring rules) to a short coherent description. See Supplementary Information F for full details and prompts used.

Human evaluations structure. Extended Data Fig. 4 shows our human evaluation interface. Following instructions and an understanding quiz, participants evaluated nine total games: three real ones, the corresponding three matched ones and three unmatched ones. Participants were presented one game at a time and provided two short textual responses, one explaining the game in their own words and one providing a short overall impression of the game. Participants also answered seven Likert-type questions on 5-point scales, answering the following questions about the italicized attributes:

(1) Understandable: 'How confident are you that you understand the game described above?' (1, not at all confident; 3, moderately confident; 5, very confident).
(2) Fun to play: 'How fun would it be to play the game yourself?' (1, not at all fun; 3, moderately fun; 5, extremely fun).
(3) Fun to watch: 'How fun would it be to watch someone else play this game?' (1, not at all fun; 3, moderately fun; 5, extremely fun).
(4) Helpful: 'Imagine that you played this game for several minutes. How helpful would it be for learning to interact with the virtual environment?' (1, not at all helpful; 3, moderately helpful; 5, extremely helpful).
(5) Difficult: 'Imagine that you played this game for several minutes. Do you think it would be too easy, appropriately difficult, or too hard for you?' (1, far too easy; 3, appropriately difficult; 5, far too hard).
(6) Creative: 'How creatively designed is this game?' (1, not at all creative; 3, moderately creative; 5, extremely creative).
(7) Human-like: 'How human-like do you think this game is?' (1, not at all human-like; 3, moderately human-like; 5, extremely human-like).

Evaluation statistical analyses. For each attribute and each game category (real, matched and unmatched), we report the mean score assigned by all participants to games in that category for that attribute. We then also aggregate these attribute scores by category and report a non-parametric Mann–Whitney U test[87] for differences in outcomes, as appropriate for ordinal data. See Supplementary Table 2 for the full table including test statistics and P values. Significance results were highly similar when computing two-sample t-tests as an alternative statistical test. We do not perform any adjustment for multiple comparisons but note that most effects discussed would remain significant at the $\alpha = 0.05$ level under a standard Bonferroni correction. We report extended analyses, including inter-rater reliability[88], in the supplemental information.

Mixed-effects models. We are interested in modelling the relationship between the scores predicted by our fitness function and the attributes human evaluators predicted. To that end, we set up mixed-effects regression models[89,90]. We fit separate models for each measure as the dependent variable, regressing a continuous latent score (for example, s^i_{fp} for the fun-to-play measure, equation (2) below). We include fixed effects for membership in the real ($\mathbb{1}^i_{\text{real}}$) and matched ($\mathbb{1}^i_{\text{matched}}$) groups, treating the unmatched group as a baseline. In our second analysis, we also include a fixed for the fitness score (x^i) (which is the full form reported in equation (2) below). We include random effects for the individual participants ($\epsilon_p^{p_i} \sim \mathcal{N}(0, \sigma_p^2)$) and evaluated games ($\epsilon_g^{g_i} \sim \mathcal{N}(0, \sigma_g^2)$). We also fit a sequence of cut-points (equation (3)) that transform the latent score to the observed ordinal rating y^i_{fp} (equation (4)). We suppress the subscript for each measure below:

$$s^i = \beta_{\text{fit}} x^i + \beta_{\text{real}} \mathbb{1}^i_{\text{real}} + \beta_{\text{matched}} \mathbb{1}^i_{\text{matched}} + \epsilon_p^{p_i} + \epsilon_g^{g_i} + \epsilon^i, \qquad \epsilon^i \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

$$-\infty \equiv c_0 < c_1 < c_2 < c_3 < c_4 < c_5 \equiv \infty \tag{3}$$

$$c_{k-1} < s^i \leq c_k \Rightarrow \text{observe } y^i = k. \tag{4}$$

Models without either random effect performed worse than the full model, so we report results including both random effects. We fit cumulative link models for ordinal regression[91,92] using the ordinal package version 2023.12-4 (ref. 93) in R version 4.3.2 (2023-10-31)[94], and produce plots using the jtools package version 2.3.0 (ref. 95). We report coefficient significance estimates using the two-sided Wald test, as implemented in the ordinal package. The fitted coefficients of these mixed-effects models are summarized in Extended Data Table 2 and Extended Data Fig. 7 (see Supplementary Information I.3 and Supplementary Tables 5 and 6 for additional details).

Marginal means comparisons. To compare between the three categories we evaluate (real, matched, and unmatched games), we use the method of estimated (least-square) marginal means. This allows us to account for variations in the random effects fitted to individual evaluation participants and evaluated games. In the models fitted with fitness scores, these similarly allow accounting for variations in observed fitness scores between game categories and their predicted effect on the ratings. Intuitively, the method simulates the marginal means of the dependent variable as though we had observed each combination of fixed effect (fitness score) and random effects (for individual raters and games) for all values of the group of interest (game type), allowing us to compare its effect most directly. We use the emmeans package version 1.1.0 (ref. 96) to estimate the mean score for each attribute in each category. We also report standard errors (of the differences in estimated means) using the emmeans package and two-sided significance tests adjusted using the Tukey method (to control for the multiple difference tests within each attribute). We summarize the marginal means comparison in Table 1, Extended Data Fig. 5, Supplementary Table 6 and Supplementary Fig. 4.

Sample similarity comparison methods

For each real game and its corresponding matched game from those included in the human evaluations, we examine which of our recorded

participant interactions (see 'Dataset collection methods' section above) fulfils one or more gameplay elements. We treat the setup (if specified) and each gameplay preference as a gameplay element; our aim here is to quantify which participant interaction traces play a part of the game. We do this using our reward machine—our implementation of an interpreter for goal programs in this DSL (see 'Dataset collection methods' section). For each pair of games, we then check which particular interactions either (1) play parts of both games, (2) fulfil components only in the real game or (3) fulfil components only in the matched game. We colour these proportions in purple, green and blue, respectively, in Extended Data Fig. 6.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All data for our study, including raw participant responses in the behavioural experiment, their translations to programs in our DSL and the specification for the DSL, are available via GitHub at https://github.com/guydav/goals-as-reward-producing-programs/ or via Zenodo at https://doi.org/10.5281/zenodo.14238893 (ref. 97).

Code availability

All code for our study, including code used to analyse and generate figures for our behavioural experiment, and the full implementation of our GPG model, are available via GitHub at https://github.com/guydav/goals-as-reward-producing-programs/ or via Zenodo at https://doi.org/10.5281/zenodo.14238893 (ref. 97). Our behavioural data collection experiment is publicly accessible at https://game-generation-public.web.app/. Code for the behavioural experiment is available via GitHub at https://github.com/guydav/game-creation-behavioral-experiment. Our human evaluation experiment is publicly accessible at https://exps.gureckislab.org/e/expert-caring-chemical/#/welcome. Code for the human evaluation experiment is available via GitHub at https://github.com/guydav/game-fitness-judgements.

References

1. Dweck, C. S. Article commentary: the study of goals in psychology. *Psychol. Sci.* **3**, 165–167 (1992).
2. Austin, J. T. & Vancouver, J. B. Goal constructs in psychology: structure, process, and content. *Psychol. Bull.* **120**, 338–375 (1996).
3. Elliot, A. J. & Fryer, J. W. in *Handbook of Motivation Science* Vol. 638 (ed. Shah, J. Y.) 235–250 (The Guilford Press, 2008).
4. Hyland, M. E. Motivational control theory: an integrative framework. *J. Pers. Soc. Psychol.* **55**, 642–651 (1988).
5. Eccles, J. S. & Wigfield, A. Motivational beliefs, values, and goals. *Annu. Rev. Psychol.* **53**, 109–132 (2002).
6. Brown, L. V. *Psychology of Motivation* (Nova Science Publishers, 2007); https://books.google.com/books?id=hzPCuKfpXLMC
7. Fishbach, A. & Ferguson, M. J. in *Social Psychology: Handbook of Basic Principles* Vol. 2 (eds Kruglanski, A. W. & Higgins, E. T.) 490–515 (The Guilford Press, 2007).
8. Pervin, L. A. *Goal Concepts in Personality and Social Psychology* (Taylor & Francis, 2015); https://books.google.com/books?id=lIXwCQAAQBAJ
9. Moskowitz, G. B. & Grant, H. *The Psychology of Goals* Vol. 548 (Guilford Press, 2009).
10. Molinaro, G. & Collins, A. G. E. A goal-centric outlook on learning. *Trends Cogn. Sci.* **27**, 1150–1164 (2023).
11. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MIT Press, 2018).
12. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
13. Chu, J., Tenenbaum, J. B. & Schulz, L. E. In praise of folly: flexible goals and human cognition. *Trends Cogn. Sci.* **28**, 628–642 (2024).
14. Chu, J. & Schulz, L. E. Play, curiosity, and cognition. *Annu. Rev. Dev. Psychol.* **2**, 317–343 (2020).
15. Lillard, A. S. in *Handbook of Child Psychology and Developmental Science* Vol. 3 (eds Liben, L. & Mueller, U.) 425–468 (Wiley-Blackwell, 2015).
16. Andersen, M. M., Kiverstein, J., Miller, M. & Roepstorff, A. Play in predictive minds: a cognitive theory of play. *Psychol. Rev.* **130**, 462–479 (2023).
17. Oudeyer, P.-Y., Kaplan, F. & Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE Trans. Evol. Comput.* **11**, 265–286 (2007).
18. Nguyen, C. T. *Games: Agency as Art* (Oxford Univ. Press, 2020).
19. Kolve, E. et al. AI2-THOR: an interactive 3D environment for visual AI. Preprint at https://arxiv.org/abs/1712.05474 (2017).
20. Fodor, J. A. *The Language of Thought* (Harvard Univ. Press, 1979).
21. Goodman, N. D., Tenenbaum, J. B., Feldman, J. & Griffiths, T. L. A rational analysis of rule-based concept learning. *Cogn. Sci.* **32**, 108–154 (2008).
22. Piantadosi, S. T., Tenenbaum, J. B. & Goodman, N. D. Bootstrapping in a language of thought: a formal model of numerical concept learning. *Cognition* **123**, 199–217 (2012).
23. Rule, J. S., Tenenbaum, J. B. & Piantadosi, S. T. The child as hacker. *Trends Cogn. Sci.***24**, 900–915 (2020).
24. Wong, L. et al. From word models to world models: translating from natural language to the probabilistic language of thought. Preprint at https://arxiv.org/abs/2306.12672 (2023).
25. Ghallab, M. et al. *PDDL—The Planning Domain Definition Language* Tech Report CVC TR-98-003/DCS TR-1165 (Yale Center for Computational Vision and Control, 1998).
26. Chopra, S., Hadsell, R. & LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 539–546 (IEEE, 2005).
27. Le-Khac, P. H., Healy, G. & Smeaton, A. F. Contrastive representation learning: a framework and review. *IEEE Access* **8**, 193907–193934 (2020).
28. Pugh, J. K., Soros, L. B & Stanley, K. O. Quality diversity: a new frontier for evolutionary computation. *Front. Robot. AI* https://doi.org/10.3389/frobt.2016.00040 (2016).
29. Chatzilygeroudis, K., Cully, A., Vassiliades, V. & Mouret, J. B. Quality-diversity optimization: a novel branch of stochastic optimization. *Springer Optim. Appl.* **170**, 109–135 (2020).
30. Mouret, J.-B. & Clune, J. Illuminating search spaces by mapping elites. Preprint at https://arxiv.org/abs/1504.04909 (2015).
31. Ward, T. B. Structured imagination: the role of category structure in exemplar generation. *Cogn. Psychol.* **27**, 1–40 (1994).
32. Allen, K. R. et al. Using games to understand the mind. *Nat. Hum. Behav.* https://doi.org/10.1038/s41562-024-01878-9 (2024).
33. Liu, M., Zhu, M. & Zhang, W. Goal-conditioned reinforcement learning: problems and solutions. In *Proc. 31st International Joint Conference on Artificial Intelligence: Survey Track* (ed. De Raedt, L.) 5502–5511 (IJCAI, 2022).
34. Colas, C., Karch, T., Sigaud, O. & Oudeyer, P.-Y. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *J. Artif. Intell. Res.* **74**, 1159–1199 (2022).
35. Icarte, R. T., Klassen, T. Q., Valenzano, R. & McIlraith, S. A. Reward machines: exploiting reward function structure in reinforcement learning. *J. Artif. Intell. Res.* **73**, 173–208 (2022).
36. Pell, B. *Metagame in Symmetric Chess-Like Games* UCAM-CL-TR-277 (Univ. Cambridge, Computer Laboratory, 1992).

37. Hom, V. & Marks, J. Automatic design of balanced board games. In *Proc. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* Vol. 3 (eds Schaeffer, J. & Mateasvol, M.) 25–30 (AAAI Press, 2007).
38. Browne, C. & Maire, F. Evolutionary game design. *IEEE Trans. Comput. Intell. AI Games* **2**, 1–16 (IEEE, 2010).
39. Togelius, J. & Schmidhuber, J. An experiment in automatic game design. In *2008 IEEE Symposium On Computational Intelligence and Games* 111–118 (IEEE, 2008).
40. Smith, A. M., Nelson, M. J. & Mateas, M. Ludocore: a logical game engine for modeling videogames. In *Proc. 2010 IEEE Conference on Computational Intelligence and Games* 91–98 (IEEE, 2010).
41. Zook, A. & Riedl, M. Automatic game design via mechanic generation. In *Proc. AAAI Conference on Artificial Intelligence* Vol. 28, https://doi.org/10.1609/aaai.v28i1.8788 (AAAI Press, 2014).
42. Khalifa, A., Green, M. C., Perez-Liebana, D. & Togelius, J. General video game rule generation. In *2017 IEEE Conference on Computational Intelligence and Games* 170–177 (IEEE, 2017).
43. Lake, B. M., Ullman, T. D., Tenenbaum, J. B. & Gershman, S. J. Building machines that learn and think like people. *Behav. Brain Sci.* **40**, e253 (2017).
44. Cully, A. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proc. Genetic and Evolutionary Computation Conference* (ed. López-Ibáñez, M.) 81–89 (Association for Computing Machinery, 2019).
45. Grillotti, L. & Cully, A. Unsupervised behavior discovery with quality-diversity optimization. *IEEE Trans. Evol. Comput.* **26**, 1539–1552 (2022).
46. Ullman, T. D., Spelke, E., Battaglia, P. & Tenenbaum, J. B. Mind games: game engines as an architecture for intuitive physics. *Trends Cogn. Sci.* **21**, 649–665 (2017).
47. Chen, T., Allen, K. R., Cheyette, S. J., Tenenbaum, J. & Smith, K. A. ' Just in time' representations for mental simulation in intuitive physics. In *Proc. Annual Meeting of the Cognitive Science Society* Vol. 45 (UC Merced, 2023); https://escholarship.org/uc/item/3hq021qs
48. Tang, H., Key, D. & Ellis, K. WorldCoder, a model-based LLM agent: building world models by writing code and interacting with the environment. Preprint at https://arxiv.org/abs/2402.12275 (2024).
49. Reed, S. et al. A generalist agent. *Trans. Mach. Learn. Res.* 1ikK0kHjvj (2022).
50. Gallouédec, Q., Beeching, E., Romac, C. & Dellandréa, E. Jack of all trades, master of some, a multi-purpose transformer agent. Preprint at https://arxiv.org/abs/2402.09844 (2024).
51. Florensa, C., Held, D., Geng, X. & Abbeel, P. Automatic goal generation for reinforcement learning agents. In *Proc. 35th International Conference on Machine Learning* Vol. 80 (eds Dy, J. & Krause, A.) 1515–1528 (PMLR, 2018).
52. Open Ended Learning Team et al. Open-ended learning leads to generally capable agents. Preprint at https://arxiv.org/abs/2107.12808 (2021).
53. Du, Y. et al. Guiding pretraining in reinforcement learning with large language models. In *Proc. of the 40th International Conference on Machine Learning* (eds Krause, A. et al.) 8657–8677 (JMLR, 2023).
54. Colas, C., Teodorescu, L., Oudeyer, P.-Y., Yuan, X. & Côté, M.-A. Augmenting autotelic agents with large language models. Preprint at https://arxiv.org/abs/2305.12487v1 (2023).
55. Littman, M. L. et al. Environment-independent task specifications via GLTL. Preprint at http://arxiv.org/abs/1704.04341 (2017).
56. Leon, B. G., Shanahan, M. & Belardinelli, F. In a nutshell, the human asked for this: latent goals for following temporal specifications. In *10th International Conference on Learning Representations* (OpenReview, 2022); https://openreview.net/forum?id=rUwm9wCjURV
57. Ma, Y. J. et al. *Eureka: Human-Level Reward Design via Coding Large Language Models* (ICLR, 2023).
58. Faldor, M., Zhang, J., Cully, A. & Clune, J. OMNI-EPIC: open-endedness via models of human notions of interestingness with environments programmed in code. In *12th International Conference on Learning Representations* (OpenReview, 2024); https://openreview.net/forum?id=AgM3MzT99c
59. Colas, C. et al. Language as a cognitive tool to imagine goals in curiosity-driven exploration. In *Proc. 34th International Conference on Neural Information Processing Systems (NIPS '20)* (eds Larochelle, H. et al.) 3761–3774 (Curran Associates, 2020).
60. Wu, C. M., Schulz, E., Speekenbrink, M., Nelson, J. D. & Meder, B. Generalization guides human exploration in vast decision spaces. *Nat. Hum. Behav.* **2**, 915–924 (2018).
61. Ten, A. et al. in *The Drive for Knowledge: The Science of Human Information Seeking* (eds. Dezza, I. C. et al.) 53–76 (Cambridge Univ. Press, 2022).
62. Berlyne, D. E. Novelty and curiosity as determinants of exploratory behaviour. *Br. J. Psychol. Gen. Sect.* **41**, 68–80 (1950).
63. Gopnik, A. Empowerment as causal learning, causal learning as empowerment: a bridge between Bayesian causal hypothesis testing and reinforcement learning. *PhilSci-Archive* https://philsci-archive.pitt.edu/23268/ (2024).
64. Addyman, C. & Mareschal, D. Local redundancy governs infants' spontaneous orienting to visual-temporal sequences. *Child Dev.* **84**, 1137–1144 (2013).
65. Du, Y. et al. What can AI learn from human exploration? Intrinsically-motivated humans and agents in open-world exploration. In *NeurIPS 2023 Workshop: Information-Theoretic Principles in Cognitive Systems* (OpenReview, 2023); https://openreview.net/forum?id=aFEZdGL3gn
66. Ruggeri, A., Stanciu, O., Pelz, M., Gopnik, A. & Schulz, E. Preschoolers search longer when there is more information to be gained. *Dev. Sci.* **27**, e13411 (2024).
67. Liquin, E. G., Callaway, F. & Lombrozo, T. Developmental change in what elicits curiosity. In *Proc. Annual Meeting of the Cognitive Science Society* Vol. 43 (UC Merced, 2021); https://escholarship.org/uc/item/43g7m167
68. Taffoni, F. et al. Development of goal-directed action selection guided by intrinsic motivations: an experiment with children. *Exp. Brain Res.* **232**, 2167–2177 (2014).
69. Ten, A., Kaushik, P., Oudeyer, P.-Y. & Gottlieb, J. Humans monitor learning progress in curiosity-driven exploration. *Nat. Commun.* **12**, 5972 (2021).
70. Baldassarre, G. et al. Intrinsic motivations and open-ended development in animals, humans, and robots: an overview. *Front. Psychol.* **5**, 985 (2014).
71. Spelke, E. S. & Kinzler, K. D. Core knowledge. *Dev. Sci.* **10**, 89–96 (2007).
72. Jara-Ettinger, J., Gweon, H., Schulz, L. E. & Tenenbaum, J. B. The naïve utility calculus: computational principles underlying commonsense psychology. *Trends Cogn. Sci.* **20**, 589–604 (2016).
73. Liu, S., Brooks, N. B. & Spelke, E. S. Origins of the concepts cause, cost, and goal in prereaching infants. *Proc. Natl Acad. Sci. USA* **116**, 17747–17752 (2019).
74. Jara-Ettinger, J. Theory of mind as inverse reinforcement learning. *Curr. Opin. Behav. Sci.* **29**, 105–110 (2019).
75. Arora, S. & Doshi, P. A survey of inverse reinforcement learning: challenges, methods and progress. *Artif. Intell.* **297**, 103500 (2021).
76. Baker, C., Saxe, R. & Tenenbaum, J. Bayesian theory of mind: Modeling joint belief–desire attribution. In *Proc. Annual Meeting of the Cognitive Science Society* Vol. 33 (UC Merced, 2011); https://escholarship.org/uc/item/5rk7z59q

77. Velez-Ginorio, J., Siegel, M. H., Tenenbaum, J. B. & Jara-Ettinger, J. Interpreting actions by attributing compositional desires. In *Proc. Annual Meeting of the Cognitive Science Society* Vol. 39 (eds Gunzelmann, G. et al.) (UC Merced, 2017); https://escholarship.org/uc/item/3qw110xj
78. Ho, M. K. & Griffiths, T. L. Cognitive science as a source of forward and inverse models of human decisions for robotics and control. *Annu. Rev. Control Robot. Auton. Syst.* **5**, 33–53 (2022).
79. Palan, S. & Schitter, C. Prolific.ac—a subject pool for online experiments. *J. Behav. Exp. Finance* **17**, 22–27 (2018).
80. Icarte, R. T., Klassen, T., Valenzano, R. & McIlraith, S. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proc. 35th International Conference on Machine Learning* Vol. 80 (eds Dy, J. & Krause, A.) 2107–2116 (PMLR, 2018).
81. Brants, T., Popat, A. C, Xu, P., Och, F. J. & Dean, J. Large language models in machine translation. In *Proc. 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (ed. Eisner, J.) 858–867 (Association for Computational Linguistics, 2007).
82. Rothe, A., Lake, B. M. & Gureckis, T. M. Question asking as program generation. In *Advances in Neural Information Processing Systems 30* (eds Von Luxburg, U. et al.) 1047–1056 (Curran Associates, 2017).
83. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M.'A. & Huang, F. J. in *Predicting Structured Data* (eds Bakir, G. et al.) Ch. 10 (MIT Press, 2006).
84. van den Oord, A., Li, Y. & Vinyals, O. Representation learning with contrastive predictive coding. Preprint at https://arxiv.org/abs/1807.03748v2 7 (2018).
85. Charity, M., Green, M. C., Khalifa, A. & Togelius, J. Mech-elites: illuminating the mechanic space of GVG-AI. In *Proc. 15th International Conference on the Foundations of Digital Games* (eds Yannakakis, G. N. et al.) 8 (Association for Computing Machinery, 2020).
86. *GPT-4 Technical Report* (OpenAI, 2023).
87. Mann, H. B. & Whitney, D. R. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat.* **18**, 50–60 (1947).
88. Castro, S. Fast Krippendorff: fast computation of Krippendorff's alpha agreement measure. *GitHub* https://github.com/pln-fing-udelar/fast-krippendorff (2017).
89. Radenbush, S. W. & Bryk, A. S. *Hierarchical Linear Models. Applications and Data Analysis Methods* 2nd edn (Sage Publications, 2002).
90. Hox, J., Moerbeek, M. & van de Schoot, R. *Multilevel Analysis (Techniques and Applications)* 3rd edn (Routledge, 2018).
91. Argesti, A. *Categorical Data Analysis* 3rd edn (Wiley, 2018).
92. Greene, W. H. & Hensher, D. A. *Modeling Ordered Choices: A Primer* (Cambridge Univ. Press, 2010).
93. Christensen, R. H. B. ordinal—regression models for ordinal data. R package version 2023.12-4 https://CRAN.R-project.org/package=ordinal (2023).
94. R Core Team. *R: A Language and Environment for Statistical Computing* Version 4.3.2 https://www.R-project.org/ (R Foundation for Statistical Computing, 2023).
95. Long, J. A. jtools: analysis and presentation of social scientific data. *J. Open Source Softw.* **9**, 6610 (2024).
96. Lenth, R. V. emmeans: estimated marginal means, aka least-squares means. R package version 1.10.0 https://CRAN.R-project.org/package=emmeans (2024).
97. Davidson, G., Todd, G., Togelius, J., Gureckis, T. M. & Lake, B. M. guydav/goals-as-reward-producing-programs: release for DOI. *Zenodo* https://doi.org/10.5281/zenodo.14238893 (2024).

Acknowledgements

G.D. thanks members of the Human and Machine Learning Lab and the Computation and Cognition Lab at NYU for their feedback at various stages of this project. We thank L. Wong for helpful discussions on which questions to prioritize in the human evaluations of our model outputs. We thank O. Timplaru and https://vecteezy.com for the use of the illustration in Fig. 1a. G.D. and B.M.L. are supported by the National Science Foundation (NSF) under NSF Award 1922658. G.T.'s work on this project is supported by the NSF GRFP under grant DGE-2234660. T.M.G.'s work on this project is supported by NSF BCS 2121102.

Author contributions

G.D. designed and executed the behavioural experiments and analysed their data. G.D. and G.T. jointly designed and implemented the GPG model and designed the human evaluations. G.D. led human evaluation data collection and analysis. G.D.and G.T. led the writing of the paper. J.T. advised on computational modelling and helped write the paper. T.M.G. and B.M.L. jointly advised all work reported in this manuscript and helped write the paper.

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s42256-025-00981-4.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-025-00981-4.

Correspondence and requests for materials should be addressed to Guy Davidson.

Peer review information *Nature Machine Intelligence* thanks Cédric Colas and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

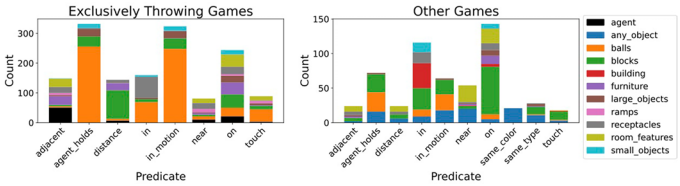
Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature Limited 2025

https://doi.org/10.1038/s42256-025-00981-4

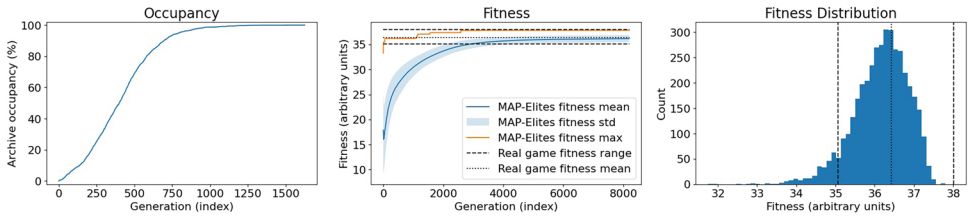
Extended Data Fig. 1 | Online experiment interface. The main part of the screen presents the AI2-THOR-based experiment room. Below it, we depict the controls. To the right, we show the text prompts for creating a new game (fonts enlarged for visualization). Our experiment is accessible online here.

https://doi.org/10.1038/s42256-025-00981-4



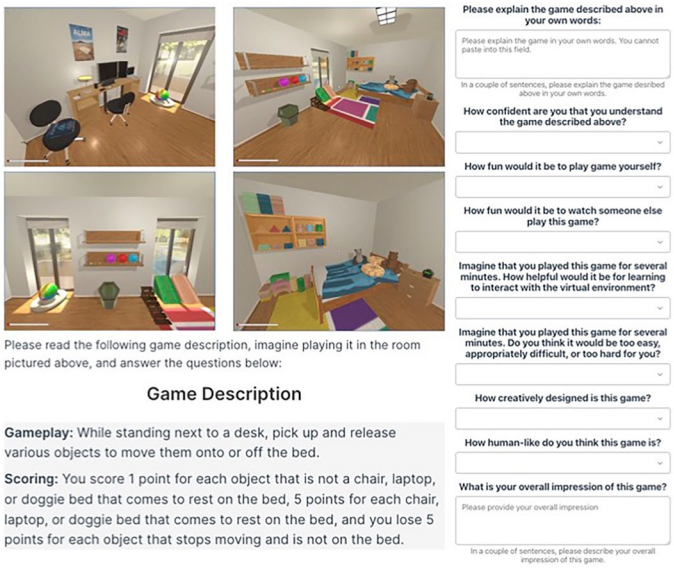
Extended Data Fig. 2 | Common-sense behavioral analyses. We plot similar information to Fig. 2b, but including additional object categories and predicates.

https://doi.org/10.1038/s42256-025-00981-4



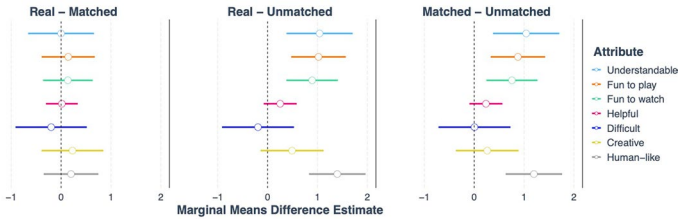
Extended Data Fig. 3 | Our implementation of the Goal Program Generator model fills the archive quickly and finds examples with human-like fitness scores. Left: Our model rapidly finds exemplars for all archive cells (that is niches induced by our behavioral characteristics), reaching 50% occupancy after 400 generations (out of a total of 8192) and 95% occupancy after 794 generations—the archive is almost full 1/10th of the way through the search process. Middle: Our model reaches human-like fitness scores. After only three generations, the fittest sample in the archive has a higher fitness score than at least one participant-created game. By the end of the search, the mean fitness in the archive is close to the mean fitness of human games. Right: Our model generates the vast majority of its samples within the range of fitness scores occupied by participant-created games, though few samples approach the top of the range.

https://doi.org/10.1038/s42256-025-00981-4



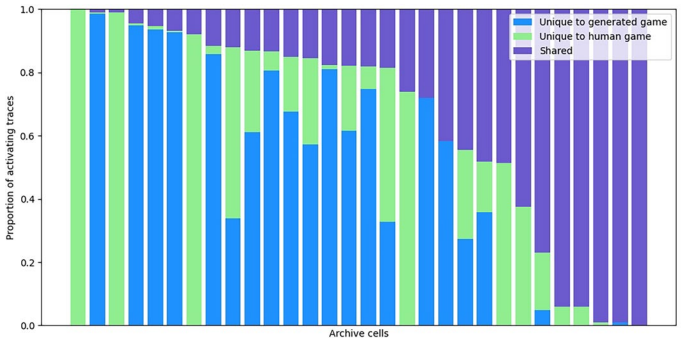
Extended Data Fig. 4 | Human evaluations interface. For each game, participants viewed the same four images of the environment, followed by the GPT-4 back-translated description of the game (see Human evaluation methods for details). They then answered the two free-response and seven multiple-choice questions on the right. In the web page version, the questions appeared below the game description; we present them side-by-side to save space.

https://doi.org/10.1038/s42256-025-00981-4



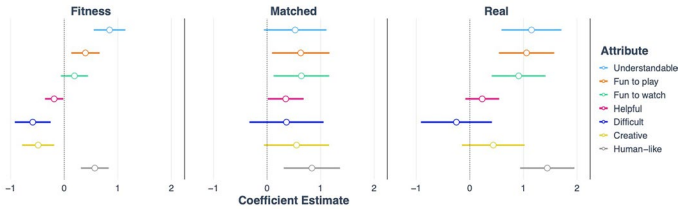
Extended Data Fig. 5 | Mixed model result summary. We summarize the pairwise comparisons made in Table 1. Each panel corresponds to a set of columns in Table 1 and each color to one of the seven human evaluation attributes we consider. We compare the estimated marginal mean scores under the fitted mixed effect models between each pair of game types listed in the panel title. As in Table 1, we use the method of estimated (least-squares) marginal means to compare the three groups of games, accounting for the random effects fitted to particular games and human evaluators.

https://doi.org/10.1038/s42256-025-00981-4



Extended Data Fig. 6 | Proportion of human interactions activating only `matched` and `real` games in the same cell. Each bar corresponds to a pair of corresponding `matched` and `real` games. In each bar, we plot the proportion of relevant interactions (state-action traces) that are unique to the `matched` game (blue), unique to the `real` game (green), or shared across both (purple). A few games (with the bar mostly or entirely in purple) show high similarity between the corresponding games — under 25% (7/30) share more than half of their relevant interactions. Most games, however, show substantial differences between the sets of relevant interactions, with some showing a higher fraction unique to human games and others to matched model games. The average Jaccard similarity between the sets of relevant interactions for the `matched` and `real` game is only 0.347 and the median similarity is 0.180 (identical games would score 1.0, entirely dissimilar games 0).

https://doi.org/10.1038/s42256-025-00981-4



Extended Data Fig. 7 | Mixed model (including fitness) coefficient summary. We summarize the fitted model coefficients listed in Extended Data Table 2. Each panel corresponds to a particular coefficient in Extended Data Table 2 and each color to one of the seven human evaluation attributes we consider. We plot the fitted coefficient value and a standard error estimated using the Hessian as implemented in the `clmm` **R** package. We observe the same effects discussed in Extended Data Table 2.

https://doi.org/10.1038/s42256-025-00981-4

Extended Data Table 1 | Non-parametric significance test results mostly corroborate mixed-model results

	Mean score by category			Real vs. Matched		Real vs. Unmatched		Matched vs. Unmatched	
Attribute	Real $\pm SE$	Matched $\pm SE$	Unmatched $\pm SE$	U-stat	P-value	U-stat	P-value	U-stat	P-value
Understandable↑	3.943 ±0.068	3.923 ±0.070	3.331 ±0.075	45088.0	0.906	55921.5	1.718e−9***	55846.0	3.733e−9***
Fun to play↑	2.522 ±0.066	2.430 ±0.064	2.068 ±0.062	46752.5	0.352	54040.5	3.235e−7***	52539.5	1.826e−5***
Fun to watch↑	2.385 ±0.068	2.313 ±0.066	2.024 ±0.0664	46169.0	0.519	51636.5	8.793e−5***	50515.0	1.027e−3**
Helpful† ↑	2.997 ±0.068	2.987 ±0.066	2.840 ±0.073	44802.0	0.982	47372.5	0.078	47559.0	0.075
Difficult ↓↑	2.582 ±0.055	2.660 ±0.0566	2.676 ±0.066	42921.5	0.326	42218.5	0.419	44081.0	0.947
Creative↑	2.318 ±0.061	2.213 ±0.056	2.143 ±0.064	47036.0	0.282	48286.0	0.025*	46615.0	0.182
Human-like↑	2.813 ±0.066	2.670 ±0.070	2.119 ±0.067	47698.0	0.167	58679.0	1.702e−13***	55434.5	1.333e−8***

Fitness scores show (statistically) significant positive effects on the understandability, fun to play, and human-likeness attributes, and significant negative effects on the helpfulness, difficulty and creativity questions. Accounting for the role of fitness, the matched group membership shows significant effects only the fun to play and watch, helpfulness, and human likeness questions. The real group shows significant effects on understandability, fun to play and watch, and human likeness. Standard errors were estimated using the Hessian as part of model-fitting. We report coefficient significance estimates using the two-sided Wald test. *: P < 0.05, **: P < 0.01, ***: P < 0.001 †: The full measure description is 'Helpful for interacting with the simulated environment.' In most measures, higher scores are better, indicated by the ↑, other than Difficult ↓↑, in which 3 means 'appropriately difficult', and scores below and above indicate too easy and too hard respectively.

https://doi.org/10.1038/s42256-025-00981-4

Extended Data Table 2 | Fitness scores significantly predict several attributes, including understandability and human-likeness

	Variable								
	Fitness			𝟙[Matched]			𝟙[Real]		
Attribute	$\beta_{\text{fitness}} \pm SE$	Z	P-value	$\beta_{\text{matched}} \pm SE$	Z	P-value	$\beta_{\text{real}} \pm SE$	Z	P-value
Understandable ↑	0.846 ±0.150	5.625	1.858e−8***	0.525 ±0.297	1.766	0.077	1.151 ±0.285	4.036	5.431e−5***
Fun to play ↑	0.396 ±0.135	2.936	3.322e−3**	0.629 ±0.274	2.298	0.022*	1.059 ±0.263	4.021	5.797e−5***
Fun to watch ↑	0.191 ±0.130	1.469	0.142	0.641 ±0.266	2.414	0.016*	0.912 ±0.257	3.547	3.901e−4***
Helpful† ↑	−0.189 ±0.087	−2.163	0.031*	0.349 ±0.170	2.048	0.041*	0.232 ±0.161	1.441	0.150
Difficult ↓↑	−0.588 ±0.171	−3.443	5.763e−4***	0.363 ±0.353	1.029	0.304	−0.250 ±0.338	−0.740	0.460
Creative ↑	−0.486 ±0.152	−3.191	1.416e−3**	0.551 ±0.310	1.776	0.076	0.438 ±0.298	1.467	0.142
Human-like ↑	0.570 ±0.132	4.316	1.589e−5***	0.837 ±0.268	3.128	1.762e−3**	1.446 ±0.258	5.597	2.179e−8***

Fitness scores show (statistically) significant positive effects on the understandability, fun to play, and human-likeness attributes, and significant negative effects on the helpfulness, difficulty and creativity questions. Accounting for the role of fitness, the matched group membership shows significant effects only the fun to play and watch, helpfulness, and human likeness questions. The real group shows significant effects on understandability, fun to play and watch, and human likeness. Standard errors were estimated using the Hessian as part of model-fitting. We report coefficient significance estimates using the two-sided Wald test. *: P < 0.05, **: P < 0.01, ***: P < 0.001 †: The full measure description is 'Helpful for interacting with the simulated environment.' In most measures, higher scores are better, indicated by the ↑, other than Difficult ↓↑, in which 3 means 'appropriately difficult', and scores below and above indicate too easy and too hard respectively.

nature portfolio

Corresponding author(s): Guy Davidson

Last updated by author(s): May 30, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☐	☒	A description of all covariates tested
☐	☒	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Our behavioral data collection experiment is publicly accessible at https://game-generation-public.web.app/. Code for the behavioral experiment is available on GitHub: https://github.com/guydav/game-creation-behavioral-experiment. Our human evaluation experiment is publicly accessible here: https://exps.gureckislab.org/e/expert-caring-chemical/#/consent. Code for the human evaluation experiment is available on GitHub: https://github.com/guydav/game-fitness-judgements
Data analysis	Our data analyses were performed in R and Python. All code used for data analysis is available on GitHub: https://github.com/guydav/goals-as-reward-producing-programs.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Data used for our study, including raw, anonymized participant responses on both the game creation and human evaluation studies are available as part of our GitHub repository: https://github.com/guydav/goals-as-reward-producing-programs.

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	We do not report any findings separated by either sex or gender. We do not perform sex- or gender-based analyses as those were not material to our questions of interest.
Population characteristics	See above.
Recruitment	Participants were anonymously recruited through the Prolific platform.
Ethics oversight	The study was performed under the NYU IRB study titled "Active Learning in Dynamic Task Environments", IRB-FY2016-231, under principal investigator Todd Gureckis.

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☐ Life sciences ☒ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Behavioural & social sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	We performed two studies. (1) a qualitative behavioral study asking participants to create games, and (2) a quantitative human-evaluation study asking participants to rate games created by participants in study (1) and by our model.
Research sample	We recruited native English speaker participants using the Prolific online platform. Participants in our sample who completed the game creation experiment have a mean age of 31.7 years (with a standard deviation of 9.35 years), are 73.4% male (26.6% female), and all report being fluent in English. Participants in our sample who completed the human evaluation experiment have a mean age of 36.0 years (with a standard deviation of 10.60 years), are 56.1% male (43.9% female), and all report being fluent in English. These demographic characteristics exclude four participants in the first experiment who have since left the Prolific platform and whose demographic data has expired.
Sampling strategy	We performed no explicit sampling strategy beyond filtering for fluent English speakers (in the first experiment), further limiting to those who reside in the US (for the second experiment, after Prolific made that feature available). Sample sizes were not predetermined. In the first experiment, we collected data from additional participants until it seemed our domain-specific language can describe newly collected data without further modifications. In the second experiment, we collected several hundred evaluations of samples in each group of interest from about one hundred participants.
Data collection	We collected data in interactive online experiments in both studies. These experiments were performed by participants outside of our observation, with no researcher present.
Timing	Data collection in the behavioral game-creation study was conducted from 2021-10-22 to 2022-01-24. Data collection in the game evaluation study was conducted from 2024-01-22 to 2024-02-01.
Data exclusions	In the game creation experiment, we excluded 8 games that did not satisfy the constraints we posed on participants, 6 duplicates (including some due to technical difficulties from participants who restarted the experiment), and 6 other games that were unclear or under-specified. After accounting for two other games we opted to avoid modeling due to their complexity (one referring directly to the game interface, and control, and another describing several games or levels in the single description we collected), we arrived

	at our final dataset of 98 games. In the human evaluation experiment, we selected 30 participant-created games, the 30 model-generated games that 'match' them (under our model's diversity criteria), and randomly sampled 40 additional ('unmatched') model-generated games. We noticed that some of these 'unmatched' games have substantially lower fitness scores than games in the other two categories. We therefore excluded evaluations of the 10 lowest-fitness 'unmatched' samples from our analyses to reduce the degree to which fitness scores confound our analyses
Non-participation	In the game creation experiment, we contacted 192 participants via Prolific of whom 114 finished the experiment and another 12 were paid due to technical difficulties. In the game evaluation experiment, we contacted 169 participants, of whom 107 finished the experiment, though we are missing data from 7 due to technical difficulties.
Randomization	We randomly assigned participants to conditions within each experiment – room variations in the first study, and which games to review in the second study. In the first study, we collected games until we believed we had a reasonably representative sample. We did not perform a power analysis to guide sample sizes in the second study.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging