Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023)

Physics-Based Task Generation through Causal Sequence of Physical Interactions

Chathura Gamage [1], Vimukthini Pinto [1], Matthew Stephenson [2], Jochen Renz [1]

[1] School of Computing, The Australian National University, Canberra, Australia
[2] College of Science and Engineering, Flinders University, Adelaide, Australia
chathura.gamage@anu.edu.au, vimukthini.inguruwattage@anu.edu.au, matthew.stephenson@flinders.edu.au, jochen.renz@anu.edu.au

Abstract

Performing tasks in a physical environment is a crucial yet challenging problem for AI systems operating in the real world. Physics simulation-based tasks are often employed to facilitate research that addresses this challenge. In this paper, first, we present a systematic approach for defining a physical scenario using a causal sequence of physical interactions between objects. Then, we propose a methodology for generating tasks in a physics-simulating environment using these defined scenarios as inputs. Our approach enables a better understanding of the granular mechanics required for solving physics-based tasks, thereby facilitating accurate evaluation of AI systems' physical reasoning capabilities. We demonstrate our proposed task generation methodology using the physics-based puzzle game Angry Birds and evaluate the generated tasks using a range of metrics, including physical stability, solvability using intended physical interactions, and accidental solvability using unintended solutions. We believe that the tasks generated using our proposed methodology can facilitate a nuanced evaluation of physical reasoning agents, thus paving the way for the development of agents for more sophisticated real-world applications.

Introduction

Physics-based puzzles are often utilized to assess the physical reasoning abilities of humans (Diezmann and Watters 2000; Cheke, Loissel, and Clayton 2012), animals (Emery and Clayton 2009), and AI systems (Xue et al. 2023; Bakhtin et al. 2019). While humans develop the capacity to perform physical reasoning tasks from their infancy (Valenza et al. 2006; Baillargeon and DeVos 1991), this has proven to be a challenge for AI systems (Xue et al. 2023; Bakhtin et al. 2019). With the expanding use of AI systems in physical environments, there is a need for suitable testbeds to enable the advancement of these systems. Therefore, researchers have developed a range of testbeds in physics-based environments to enable experimentation and evaluation of AI agents' physical reasoning capabilities.

When evaluating the physical reasoning capabilities of an AI system based on its performance on a task, it is essential to have a thorough understanding of the physics mechanics required to solve that task. This enables a rigorous assessment of the AI system's physical reasoning weaknesses based on the physics mechanics necessary to solve the tasks. In some existing benchmarks, this task analysis is not performed at all, or it is done at a high level where the tasks are mainly categorized into families of physical scenarios/events, such as stacking blocks, picking and placing, creating domino effects, physically supporting objects, dropping objects, etc. (Ahmed et al. 2021; Allen, Smith, and Tenenbaum 2020; Bear et al. 2021). Such manual categorization by developers does not provide a clear understanding of the physics mechanics that distinguish tasks and their associated physical scenarios/events.

Inspired by a widely used approach employed by infant physics researchers in studying physical scenarios (Baillargeon et al. 2012, 2009; Bliss and Ogborn 1994), this study proposes a method for defining physical scenarios in an environment based on the causal interactions between objects. The definition of physical scenarios is carried out in a granular fashion, considering the causal sequence of physical interactions between objects that are necessary to solve tasks associated with the scenario. The task generation process considers the impact of force and motion on the interactions between objects, establishing the process on the grounds of dynamic physics. This method also paves the way towards a systematic classification of tasks according to the associated physical interactions, filling a gap in the current physical reasoning testbeds and benchmarks.

We have selected the physics-based game Angry Birds as our demonstration domain, as it offers a realistic 2D physical environment and is a popular choice in physical reasoning AI research. To begin, we introduce a grammar that we use to describe tasks and object layouts in physical environments. Using this grammar, we define example physical scenarios as a causal sequence of physical interactions, along with a set of corresponding physical restrictions. We then introduce our task generation process that takes a defined scenario as input and produces a feasible task within our demonstration domain (i.e., an Angry Birds game level). This generation process involves constructing and solving qualitative spatial relationship graphs between objects and satisfying constraints through physics simulations. We evaluate the generated tasks for their physical stability, intended solvability, and accidental solvability, which analyzes whether the tasks are solvable using unintended so-

lutions. Additionally, we analyze the generation time of the task generator as the number of physical interactions in the input physical scenarios increases.

Background and Related Work

In AI research, the generation of physics-based content is most commonly discussed in physical reasoning testbeds and benchmarks that are used to evaluate AI systems, as well as in physics-based video games when developing game content. In this section, we investigate these two research areas in the context of content generation and position our work in the current literature, emphasizing its contributions.

Physical Reasoning Benchmarks and Testbeds

In recent years, researchers have developed various environments as benchmarks and testbeds to evaluate the physical reasoning capabilities of AI agents. These environments are mainly based on tasks that involve taking actions in a physical environment (Xue et al. 2023; Bakhtin et al. 2019), reasoning about images of physical scenarios (Hong et al. 2021; Wolf 2020), and reasoning about videos of physical scenarios (Riochet et al. 2020; Yi et al. 2020). Among them, task generation methods used in action-based environments are related to this work as we also focus on generating tasks that an agent can interact with and take actions to solve them. Examples of recent action-based environments include Phy-Q (Xue et al. 2023), PHYRE (Bakhtin et al. 2019), Virtual Tools (Allen, Smith, and Tenenbaum 2020), OGRE (Allen et al. 2020), CausalWorld (Ahmed et al. 2021), and RL-Bench (James et al. 2020).

The design and development of tasks in these works are predominantly done manually by developers, with most utilizing automated generation techniques to create simple variations of handcrafted task templates. For example, Phy-Q testbed has handcrafted task templates for 15 physical scenarios, and tasks are generated from those templates by slightly varying the locations of objects in the template and adding distracting objects. PHYRE, Virtual Tools, and OGRE follow a similar procedure. They all have predesigned task templates and tasks are generated by either varying the shape, size, and location of the original template. CausalWorld is a robotic benchmark with eight types of tasks, such as pushing, picking, picking and placing, and stacking. They have separate generators to generate tasks for each of those task types. The generators have hard-coded templates, and the tasks are generated by sampling new tasks from those templates.

Despite the use of automated generation techniques to some extent in these benchmarks and testbeds, task generation is still heavily reliant on pre-defined templates. The process of creating these templates can be tedious and time-consuming, as the template developer must ensure that the template is stable under gravity and other physical constraints, solvable, and flexible enough to allow for modifications in task generation. In contrast, our proposed generation method does not require any initial task templates. Instead, the input to the generator is a minimal description of the physical scenario, defined as a sequence of physical interactions between objects that leads to solving the task. This approach eliminates the need for pre-designed templates and simplifies the task generation. Furthermore, tasks generated from this method consider the physical interactions involved in solving the task, allowing for systematic categorization of tasks into distinct physical scenarios based on the involved physical interactions. This feature is particularly valuable for physical reasoning testbeds and benchmarks, as it can be used to classify tasks systematically and evaluate AI agents' strengths and weaknesses more comprehensively. The current task classifications on those benchmarks do not justify why a given task belongs to its class.

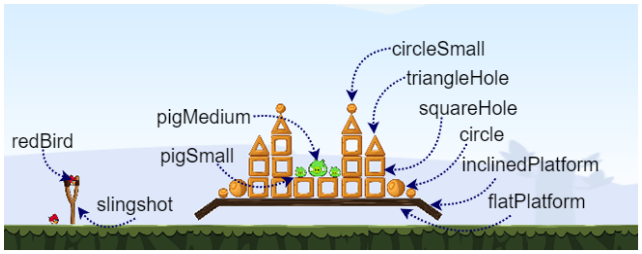


Figure 1: A game level from Angry Birds.

Physics Based Puzzle Games

Physics-based puzzle games have gained popularity in the field of Procedural Content Generation (PCG) owing to the intriguing physics-related challenges that are encountered while generating content for such games, which are applicable to real-world physics challenges. Popular physics puzzle games, such as Angry Birds (Stephenson et al. 2018) and Cut the Rope (Shaker, Shaker, and Togelius 2013), have been used as test domains by researchers to investigate physics-based content generation techniques. Of these games, Angry Birds has received considerable attention in the research community, particularly in the fields of PCG (Stephenson et al. 2018) and AI game-playing agent development (Renz et al. 2019). Consequently, in this study, we utilize Angry Birds as a test domain to showcase our proposed task generation methodology.

Angry Birds is a 2D physics simulation game wherein players must destroy pigs in a given game level by launching a designated number of birds from a slingshot. The levels in the game consist of dynamic objects (birds, blocks, and pigs) that adhere to Newtonian physics and static objects (platforms) that are not influenced by external forces. The dynamic objects possess health points that decrease upon collision and are destroyed when their health points are completely depleted. In this study, we use Science Birds (Ferreira and Toledo 2014), a research clone of the game developed in Unity with the Box2D physics engine. A modified version of Science Birds, with adjusted physics parameters and health points of objects, is employed to effectively showcase the proposed methodology. The modifications address limitations in Science Birds' object dynamics and fragility, ensuring a more accurate demonstration of the methodology. Figure 1 displays a game level from Angry Birds in Science Birds, showcasing the primary game objects utilized in this study.

Numerous research studies have explored various methods for creating game levels in Angry Birds. Prior investigations have tackled this problem from diverse angles, proposing solutions to intriguing challenges, such as generating stable structures within the physical environment (Stephenson and Renz 2017), dynamically adjusting the game levels' difficulty (Stephenson and Renz 2019), creating block structures based on hand-drawn sketches (Stephenson et al. 2021), generating deceptive levels to deceive AI agents (Gamage et al. 2021a), generating novel scenarios for physics environments (Gamage et al. 2023, 2021b) and, most recently, using prompt engineering to create prompts for level generation (Taveekitworachai et al. 2023). None of the previous studies has treated the level generation for Angry Birds as a task generation problem that incorporates the physical interactions between the game objects. This paper proposes a novel approach that defines a task as a causal sequence of physical interactions and generates tasks whose solution follows this sequence. The generation process considers the effect of force and motion on the interactions of objects, resulting in a more scientifically grounded approach in terms of dynamic physics. By generating tasks based on the physical interactions present in the solution, agents can be evaluated and compared based on their performance in specific interaction sequences of interest. Therefore, the proposed generation method enables a rigorous evaluation of AI agents' physical reasoning abilities, similar to the evaluation conducted in the Phy-Q testbed that uses handcrafted Angry Birds levels.

Grammar and Defining Scenarios

This section presents the proposed grammar and discusses how the grammar can be used to define physical scenarios.

Grammar for Angry Birds

The introduced grammar is used to describe objects, interactions, restrictions, and object layouts in a physical environment. The grammar consists of four components, namely, object grammar, interaction grammar, restriction grammar, and layout grammar. The first three grammar components are utilized to define physical scenarios, while the last grammar component is utilized in the generation process for defining the object layouts within the physical environment.

Object Grammar The Object grammar component is used to define various types of objects that may exist in a physical environment. Each grammar term represents a class of objects that share similar properties, such as the ability to roll or slide. These terms are listed in Table 1. In Angry Birds, the objects include birds that the player shoots (a single type is used in this work, named redBird), pigs that the player has to destroy (two types with different sizes are used, named pigSmall and pigMedium), blocks with different shapes and sizes (including circle, circleSmall, squareHole, and triangleHole), and platform objects that can vary in size and rotation (mainly used as surfaces).

Interaction Grammar Interaction grammar defines fundamental physical interactions between objects. When defining the interaction terms, we take into account the impact of force and motion, drawing inspiration from experiments that were conducted to understand the physical reasoning capabilities of infants (Bliss and Ogborn 1994). Specifically, we focus on the fundamental physical interaction where one object applies a force on another object; in Angry Birds, which commonly occurs by one object hitting another object. We then analyze the consequent effects of the force, which can lead to the dynamics of the object being affected, causing it to roll, fall, slide, or bounce, or even making the object deform or get destroyed as seen in Angry Birds. The interaction grammar terms are listed in Table 2.

Grammar Term	Game Objects Represented
bird	redBird
pig	pigSmall∨pigMedium
rollableBlock	circleSmall∨circle
fallableBlock	circleSmall∨circle∨squareHole∨ triangleHole
slidableBlock	squareHole∨triangleHole
horizontalSurface	flatPlatform
inclinedSurface	inclinedPlatform
surface	flatPlatform∨inclinedPlatform

Table 1: Object grammar.

Grammar Term	Description
hit(a)(b)(d)	a collides with b from direction d of b $d \in \{$*left, right, above, below, any*$\}$
roll(a)(b)(d)	a rolls on b towards direction d $d \in \{$*left, right*$\}$
fall(a)(b)	a falls towards b
slide(a)(b)(d)	a slides on b towards direction d $d \in \{$*left, right*$\}$
bounce(a)(b)(d)	a bounces off b towards direction d $d \in \{$*left, right, above, below*$\}$
destroy(a)(b)	a destroys b in the collision with b

Table 2: Interaction grammar. The parameters a and b represent objects, and d represents a direction.

Restriction Grammar The restriction grammar defines the restricted interactions between objects. Since an object in a physical environment has many possibilities of interaction, restriction grammar narrows down the interactions of interest of an object. Our current restriction terms, as shown in Table 3, prevent objects from hitting each other and from falling in their movements. These restrictions are put in place as hitting is the primary way of force transferring in Angry Birds, and falling is a natural phenomenon that occurs when an object is placed in the physical environment. It is essential to note that these restrictions only apply to the solution interaction sequence of the task and not to any possible interaction of the relevant object. For instance, an object with a restriction of cannotFall in a task implies that the object cannot fall when the solution interactions of the task are being executed, but not that it cannot fall by any means when someone is interacting with the environment.

Grammar Term	Description
cannotHit(a)(b)(d)	a cannot collide with b from direction d of b, $d \in \{$*left, right, above, below, any*$\}$
cannotFall(a)	a cannot fall in its motion

Table 3: Restriction grammar. The parameters a and b represent objects, and d represents a direction.

Grammar Term	Description
inDirection(a)(b)(d)	a is in direction d of b $d \in \{$*left, right, above, below*$\}$
onLocation(a)(b)(l)	a is on top of b at location l $l \in L$, $L = \{$*left, centre, right*$\}$
locatedFar(a)(b)(d)	a is far from b in direction d of b $d \in \{$*left, right, above, below*$\}$
touching(a)(b)(l)	a touches b at location l of b, $l \in \{$*upperLeft, centreLeft, lowerLeft*$\}$
pathObstructed(a)(b)(d)	there is an obstacle in the path from a and b, in the direction d to b, $d \in \{$*left, right, above, below, all*$\}$
liesOnPath(a)(b)	a lies on b's moving path

Table 4: Layout grammar. The parameters a and b represent objects, d represents a direction, and l represents a location.

Layout Grammar Layout grammar encompasses a set of terms that characterize spatial relationships between objects in a physical environment. This grammar is utilized by the generator to specify the configuration of objects during task generation. The layout grammar terms deemed appropriate for defining spatial relationships within the context of Angry Birds are displayed in Table 4.

Defining Scenarios

As stated previously, the object grammar, interaction grammar, and restriction grammar components are used to define the scenarios. The initial step of defining a scenario involves identifying the objects that are relevant to the scenario, which are characterized using the object grammar. The next step is to determine the intended sequence of physical interactions that must take place between these objects to obtain a solution for the scenario. These interactions are described using the interaction grammar and are arranged sequentially based on their causality. If any interaction needs to be restricted between the objects in the scenario, the restriction grammar is employed to add them.

Table 5 displays 16 sample scenarios, which were defined for illustrative purposes in this study. Scenarios one through seven were created to replicate the physical scenarios of Single Force (1 to 3), Rolling (4), Falling (5), Sliding (6), and Bouncing (7) in the Phy-Q testbed. For instance, in the first scenario, the bird collides with the pig from any direction, causing the pig to be destroyed, and there are no associated restrictions. In the second and third scenarios, the bird cannot collide with the pig from above or from the left, respectively, due to imposed restrictions. In the fourth scenario, the bird hits a rollable block, causing it to roll on a surface and then collide with a pig, leading to its destruction. The restrictions in this scenario state that the bird cannot collide directly with the pig (i.e., the player cannot shoot the bird directly at the pig to solve the task) and that the rollable block must not fall during its movement. Scenarios eight to 12 combine two out of Rolling, Falling, Sliding, and Bouncing, while scenarios 13 to 16 combine three of them.

Task Generation Process

The proposed methodology for generating tasks is discussed in detail in this section as a six-step process. Firstly, the layout constraints between objects are inferred and represented using a layout constraint graph based on the task definition. Secondly, Qualitative Spatial Relations (QSRs) between objects are inferred and the layout constraint graph is converted into a QSR graph. To verify consistency and solve constraints to obtain the initial positions of the objects, the spatial constraints are projected into X and Y dimensions separately and dimension graphs are generated and solved in the third step. Next, non-QSRs of the objects, such as restrictions related to object destruction, are satisfied using simulations and the final positions of the objects are determined. In the next step, distractions are added to the tasks, and finally, the solvability of the tasks is verified.

Generating the Layout Constraint Graph

The generation process begins by determining the objects required to be present in the scenario and inferring the layout constraints between the objects. To achieve this, the scenario definition is used to identify all objects required for the scenario. Then, the layout constraints are inferred based on the interactions and restricted interactions between objects and the constraints are described using the layout grammar. The layout constraints that can be inferred are shown in Table 6. The layout constraint graph is then generated with the objects in the scenario as nodes and the directed edges representing the layout constraints between the objects. In this process, a layout optimization is conducted to remove redundant layout constraints and ensure physical stability. This stability enhancement involves confirming that dynamic objects are supported by static ones and adding static objects beneath those not adequately supported, ensuring equilibrium under gravity assuming that the task is initially stable. Figure 2 presents the layout graphs of scenarios 6 and 9.

Transforming Layout Constraints into Spatial Relations

In this phase, the graph of layout constraints is transformed into a spatial relationship graph among objects in 2D Euclidean space. To accomplish this, we draw inspiration from various QSR calculi, including cardinal direction relations (Frank 1991), interval relationships (Allen 1983), and topological relationships (Clementini and Di Felice 1997). To meet the requirements of this study, we introduce a range of spatial predicates that are relevant to objects in 2D Euclidean space. Usually, in QSR literature, 2D objects are represented using their Minimum Bounding Rectangle (MBR) (Chen et al. 2013). However, this approach has limitations when determining the actual corner positions of an object if

Name	Scenario Definition
1. SF	{[hit(*bird*)(*pig*)(*any*)] > [destroy(*bird*)(*pig*)]}, {}
2. SFTB	{[hit(*bird*)(*pig*)(*any*)] > [destroy(*bird*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*above*)]}
3. SFLB	{[hit(*bird*)(*pig*)(*any*)] > [destroy(*bird*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*left*)]}
4. R	{[hit(*bird*)(*rBlock*)(*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*pig*)(*left*)] > [destroy(*rBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock*)]}
5. F	{[hit(*bird*)(*fBlock*)(*left*∨*above*)] > [fall(*fBlock*)(*pig*)] > [hit(*fBlock*)(*pig*)(*above*)] > [destroy(*fBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]}
6. S	{[hit(*bird*)(*sBlock*)(*left*)] > [slide(*sBlock*)(*hSurface*)(*right*)] > [hit(*sBlock*)(*pig*)(*left*)] > [destroy(*sBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*sBlock*)]}
7. B	{[hit(*bird*)(*iSurface*)(*any*)] > [bounce(*bird*)(*iSurface*)(*below*)] > [hit(*bird*)(*pig*)(*above*)] > [destroy(*bird*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]}
8. RF	{[hit(*bird*)(*rBlock*)(*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*fBlock*)(*left*)] > [fall(*fBlock*)(*pig*)] > [hit(*fBlock*)(*pig*)(*above*)] > [destroy(*fBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock*)]}
9. RS	{[hit(*bird*)(*rBlock*)(*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*sBlock*)(*left*)] > [slide(*sBlock*)(*hSurface*)(*right*)] > [hit(*sBlock*)(*pig*)(*left*)] > [destroy(*sBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock*)]∧[cannotFall(*sBlock*)]}
10. FR	{[hit(*bird*)(*fBlock*)(*left*∨*above*)] > [fall(*fBlock*)(*rBlock*)] > [hit(*fBlock*)(*rBlock*)(*above*∨*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*pig*)(*left*)] > [destroy(*rBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock*)]}
11. SR	{[hit(*bird*)(*sBlock*)(*left*)] > [slide(*sBlock*)(*hSurface*)(*right*)] > [hit(*sBlock*)(*rBlock*)(*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*pig*)(*left*)] > [destroy(*rBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*sBlock*)]∧[cannotFall(*rBlock*)]}
12. BF	{[hit(*bird*)(*iSurface*)(*any*)] > [bounce(*bird*)(*iSurface*)(*below*)] > [hit(*bird*)(*fBlock*)(*left*∨*above*)] > [fall(*fBlock*)(*pig*)] > [hit(*fBlock*)(*pig*)(*above*)] > [destroy(*fBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]}
13. SRF	{[hit(*bird*)(*sBlock*)(*left*)] > [slide(*sBlock*)(*hSurface*)(*right*)] > [hit(*sBlock*)(*rBlock*)(*left*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*fBlock*)(*left*)] > [fall(*fBlock*)(*pig*)] > [hit(*fBlock*)(*pig*)(*any*)] > [destroy(*fBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*sBlock*)]∧[cannotFall(*rBlock*)]}
14. SFR	{[hit(*bird*)(*sBlock*)(*left*)] > [slide(*sBlock*)(*hSurface*)(*right*)] > [hit(*sBlock*)(*fBlock*)(*left*)] > [fall(*fBlock*)(*rBlock*)] > [hit(*fBlock*)(*rBlock*)(*left*∨*above*)] > [roll(*rBlock*)(*surface*)(*right*)] > [hit(*rBlock*)(*pig*)(*any*)] > [destroy(*rBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*sBlock*)]∧[cannotFall(*rBlock*)]}
15. RRF	{[hit(*bird*)(*rBlock1*)(*left*)] > [roll(*rBlock1*)(surface1)(*right*)] > [hit(*rBlock1*)(*rBlock2*)(*left*)] > [roll(*rBlock2*)(surface2)(*right*)] > [hit(*rBlock2*)(*fBlock*)(*left*)] > [fall(*fBlock*)(*pig*)] > [hit(*fBlock*)(*pig*)(*any*)] > [destroy(*fBlock*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock1*)]∧[cannotFall(*rBlock2*)]}
16. RRR	{[hit(*bird*)(*rBlock1*)(*left*)] > [roll(*rBlock1*)(surface1)(*right*)] > [hit(*rBlock1*)(*rBlock2*)(*left*)] > [roll(*rBlock2*)(surface2)(*right*)] > [hit(*rBlock2*)(*rBlock3*)(*left*)] > [roll(*rBlock3*)(surface3)(*right*)] > [hit(*rBlock3*)(*pig*)(*any*)] > [destroy(*rBlock3*)(*pig*)]}, {[cannotHit(*bird*)(*pig*)(*any*)]∧[cannotFall(*rBlock1*)]∧[cannotFall(*rBlock2*)]∧[cannotFall(*rBlock3*)]}

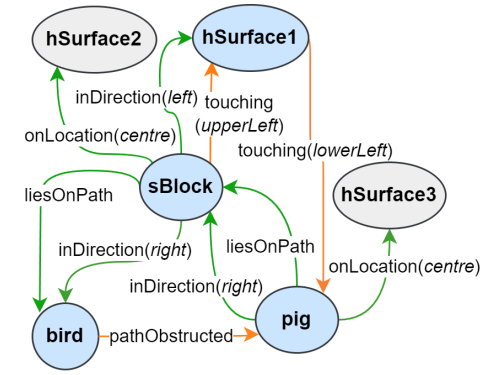
Table 5: Definitions of 16 example physical scenarios. In the definition, a sequence of physical interactions (order denoted by >) is followed by a set of restrictions. The abbreviations in the names of scenarios 1-7 are SF (Single Force), SFTB (Single Force Top Blocked), SFLB (Single Force Left Blocked), R (Rolling), F (Falling), S (Sliding), and B (Bouncing). The remaining scenario names are formed by combining R, F, S, and B (e.g., RF represents Rolling Falling and SRF represents Sliding Rolling Falling). The object grammar terms rollableBlock, fallableBlock, slidableBlock, horizontalSurface, and inclinedSurface are abbreviated as rBlock, fBlock, sBlock, hSurface, and iSurface. For overloaded parameter values, any of the overloaded values can be used (e.g., hit(*bird*)(*fBlock*)(*left*∨*above*) represents the bird collides with the *fBlock* from *left* or from *above*).

it is rotated, which is important for determining the precise placement of neighbouring objects, especially when satisfying the layout terms such as touching. To overcome this, we suggest a 5-point representation for a 2D object using the positions of 5 points lower left (ll), centre (c), upper right (ur), upper left (ul), and lower right (lr). When defining QSRs, we assume that objects have a maximum of 90 degrees of rotation. The defined QSRs are illustrated in Figure 3. The layout constraints are translated into spatial constraints using these QSRs. The QSRs in our defined set that can be inferred for the layout constraints are shown in Table 7.

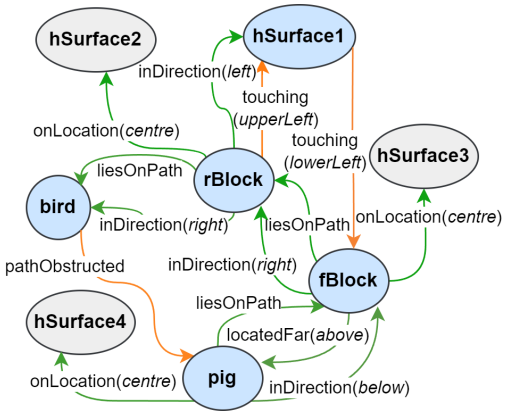
As can be seen from Table 7, a single layout constraint can be mapped to various QSRs. Additionally, the generator can leverage the flexibility in defining the scenario by

Predicate	Inferred Layout Constraints
hit(a)(b)(d)	liesOnPath(b)(a)∧inDirection(b)(a)(-d)
roll(a)(b)(d)	inDirection(a)(b)(-d)
fall(a)(b)(d)	locatedFar(a)(b)(d)
slide(a)(b)(d)	inDirection(a)(b)(-d)
bounce(a)(b)(d)	inDirection(a)(b)(d)
cannotHit(a)(b)(d)	pathObstructed(a)(b)(d)
cannotFall(a)	touching(a)(p)(lp)∧touching(p)(q)(lq),...

Table 6: Inferred layout constraints from the interactions and restrictions. The parameter -d refers to the opposite direction of d. In the term cannotFall(a), p, q, and so on denote the objects that a moves on top of in the specified order, while lp, lq, and so on signify the locations where those objects are connected, forming a continuous path for a to move along.



(a) 6. S (Sliding) scenario.



(b) 9. RF (Rolling Falling) scenario.

Figure 2: The layout constraint graphs for scenarios 6 and 9. Blue nodes represent objects in the scenario definition, while grey nodes are the objects introduced to ensure stability under gravity. Green edges represent layout constraints inferred from interactions, while orange edges represent those inferred from restricted interactions.

overloading the parameters of the grammar terms, such as in scenario 10, allowing the bird to hit the fBlock from *either the left or the above* and enabling the fBlock to hit the rBlock from *either the above or the left*. These opportunities

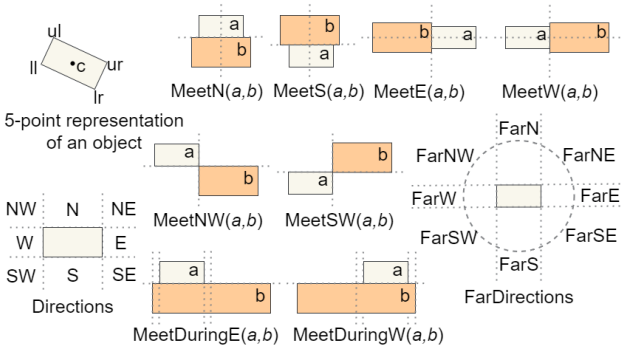


Figure 3: Illustrations of qualitative spatial relations.

Layout Predicate	Inferred QSRs
inDirection(a)(b)(*left*)	W(a,b)∨NW(a,b)∨SW(a,b)
inDirection(a)(b)(*right*)	E(a,b)∨NE(a,b)∨SE(a,b)
inDirection(a)(b)(*above*)	N(a,b)∨NE(a,b)∨NW(a,b)
inDirection(a)(b)(*below*)	S(a,b)∨SE(a,b)∨SW(a,b)
onLocation(a)(b)(*left*)	MeetDuringW(a,b)
onLocation(a)(b)(*centre*)	MeetN(a,b)
onLocation(a)(b)(*right*)	MeetDuringE(a,b)
locatedFar(a)(b)(*left*)	FarW(a,b)∨FarNW(a,b)∨ FarSW(a,b)
locatedFar(a)(b)(*right*)	FarE(a,b)∨FarNE(a,b)∨ FarSE(a,b)
locatedFar(a)(b)(*above*)	FarN(a,b)∨FarNE(a,b)∨ FarNW(a,b)
locatedFar(a)(b)(*below*)	FarS(a,b)∨FarSE(a,b)∨ FarSW(a,b)
touching(a)(b)(*upperLeft*)	MeetNW(a,b)
touching(a)(b)(*centreLeft*)	MeetW(a,b)
touching(a)(b)(*lowerLeft*)	MeetSW(a,b)

Table 7: Inferred QSRs from the layout constraints. The abbreviated notation of the directions signifies their conventional representations.

for flexibility in the generation process enable the generation of a varied range of layout configurations for a given scenario definition.

Generating a Plausible Spatial Configuration for QSR Constraints

At this stage of task generation, we have a set of spatial constraints among objects in the 2D Euclidean space. These spatial constraints are denoted as point connections using the above-mentioned 5-point representation of the objects. Now, the task at hand is to examine the consistency of these Euclidean spatial constraints among points. The purpose of consistency checking is to ensure the existence of a plausible spatial configuration that satisfies the constraints. To tackle this issue, we use an approach based on the dimension graph representation for managing the spatial constraints between objects (Liu, Shekhar, and Chawla 2001). In this technique, the spatial constraints are projected onto the X and Y dimensions, and the constraints that need to be met for each dimen-

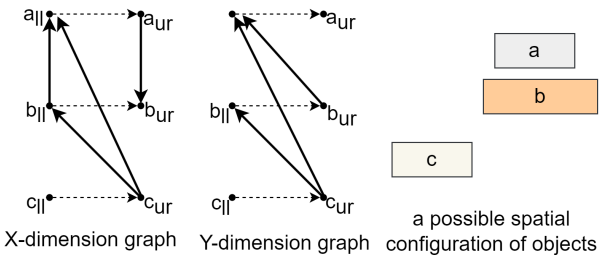


Figure 4: Example dimension graphs and a spatial configuration for the conjunctive constraint N(*a,b*)∧NE(*b,c*)∧SW(*c,a*). The objects are represented by their MBR using lower left (ll) and upper right (ur) points. Continuous arrows represent $\leq$ constraints, and dotted arrows represent intrinsic constraints between ll and ur of an MBR (Liu, Shekhar, and Chawla 2001).

sion are stored in separate graphs. From this technique, the problem of checking constraint consistency is reformulated as a graph cycle detection problem on the dimension graph. The constraints are considered consistent if both the X and Y dimension graphs are free of cycles. Figure 4 illustrates the dimension graphs of a sample conjunctive constraint. It is noteworthy that the illustration in this example employs the MBR of the objects for the sake of simplicity, whereas in this study, a more complex 5-point representation is used.

During the construction of dimension graphs, constraints in a general format are converted to the Disjunctive Normal Form (DNF), which is a disjunction of conjunctions without any disjunction within a conjunction. For instance, a constraint graph with N(*a,b*)∨NE(*a,b*) and NE(*b,c*) and SW(*c,a*) can be transformed to (N(*a,b*)∧NE(*b,c*)∧SW(*c,a*))∨(NE(*a,b*)∧NE(*b,c*)∧SW(*c,a*)) in DNF format. Consequently, for a given constraint graph, there exists a pool of dimension graph pairs (for X and Y dimensions) that can be generated from all its conjunctions. The pool size is equivalent to the number of conjunctions in the DNF. In the generation process, a random pair of dimension graphs is selected, and its cycles are checked. If there are no cycles, the generation proceeds to the next steps; otherwise, another pair is chosen and tested. In case no pair in the pool without cycles (i.e., all possible combinations of constraints are inconsistent) is found, the scenario defined is considered unfeasible to accomplish in the 2D Euclidean space. Once a consistent set of constraints is determined, the forward checking technique is employed to solve them, and a set of possible values for the points (i.e., object positions) is obtained.

Satisfying Constraints through Simulation

Although qualitative methods can provide a useful starting point to narrow down the extensive and unbounded generative space, they may not be sufficient to fully address the challenges posed by physical environments. The physical environments present a multitude of challenges that are difficult to overcome through purely qualitative methods. Interactions between objects in these environments are complex and even slight variations in these interactions can result in significant changes in the overall outcome. As such, researchers have relied on simulation-based techniques when generating physics-based tasks (Stephenson and Renz 2017; Gamage et al. 2021a).

In this step of the generation process, the destroy interaction and the constraints of liesOnPath and pathObstructed are satisfied by simulating the physics engine of the game. The simulation begins by setting up the objects in the game space according to their predetermined positions from the previous steps. Then, the behaviours of objects are observed for constraint satisfaction by executing potential solution actions (i.e., shooting the bird at the object that initiates the interaction sequence). However, due to the continuous action space in Angry Birds (where shooting angles can be chosen from a continuous range), there exist numerous actions with slight variations. Therefore, the possible solution space is discretized by taking into account the target points and the stretch of the slingshot when shooting the bird. Specifically, only certain points of interest on the target object (such as the ll, ul, and ur) are selected, and it is assumed that the bird is always shot with the full stretch of the slingshot, resulting in only two possible trajectories to reach a target point. Shooting with the full stretch reflects the current capability of Angry Birds playing agents (Xue et al. 2023).

The object destruction model in the game is based on the health points which diminish through collisions and is mainly reliant on the materials of the colliding objects and the relative velocities they attain during the collision. In order to ensure that the destroy interaction is fulfilled, the health points of the objects of interest are closely monitored at the relevant juncture of the interaction sequence.

To ensure that the liesOnPath and pathObstructed constraints are satisfied, precise consideration of the moving paths of the objects is required. The simulation results are used to observe the paths of the objects associated with these constraints. To satisfy liesOnPath(*a*)(*b*), the position of the object *a* is adjusted to the location that intersects the most paths of *b* for different actions from the solution actions. After repositioning, the new object configuration is verified by checking whether it still satisfies all the spatial constraints considered in the previous steps. If not, the process is repeated by repositioning object *a* to the next position that cuts the most paths, and so on. Once the liesOnPath constraints are satisfied, the pathObstructed constraints are handled. The pathObstructed(*a*)(*b*) constraint is satisfied by blocking the paths of *a* that lead to *b*, by adding platform objects as obstacles. The simulation results and QSRs between the objects are used to determine the paths of the object *a* that lead to *b*. It is ensured that the added obstacles do not interfere with the defined interaction sequence of the task.

If any of the simulation satisfactions fail during the generation process, the approach retries with a different dimension graph pair from the pool of dimension graphs, which is associated with a different conjunctive constraint in the DNF. This process is repeated until a feasible dimension graph pair is found. If the entire pool of dimension graphs is exhausted without finding a feasible solution, it is concluded that the defined scenario is infeasible to achieve.

Incorporating Distractions

In this work, the main purpose of generating these tasks is to use them to assess the physical reasoning capabilities of AI agents. But when solving the tasks, agents may rely on spurious patterns instead of reasoning about the underlying physics. To prevent the exploitation of such patterns, we introduce a random number of distracting objects at arbitrary positions within the generated tasks. This approach aims to reduce the likelihood of agents exploiting specific patterns, such as 'shoot the bird at the path unobstructed block', and encourages agents to engage in genuine physical reasoning.

Ensuring the Intended Solvability

In the final step, the solvability of the task is verified by executing the intended solution, which entails shooting the bird at the target object that initiates the interaction sequence. Upon successful completion, the output of the generator is a game level that possesses a definite solution consistent with the defined causal sequence of physical interactions.

Results and Evaluations

In this section, we assess the performance of our proposed method. We begin by presenting Figure 5, which showcases 16 levels that were generated for the sample scenarios provided in Table 5.

To evaluate the generated tasks, we conducted a series of analyses related to physical stability, solvability using intended solutions, and accidental solvability using unintended solutions. We generated a total of 480 tasks, with 30 tasks for each of the 16 sample scenarios. The evaluation results are presented as an aggregate, taking into account the number of physical interactions defined in each scenario. Specifically, scenarios 1 to 3 have two interactions, 4 to 7 have four interactions, 8 to 12 have six interactions, and 13 to 16 involve eight interactions. Furthermore, we analyzed the runtime of the generator to provide insights into the efficiency of our proposed method.

Physical Stability

In physical environments, ensuring the physical stability of the tasks is crucial. Despite the absence of complex physical structures in the tasks generated through the proposed method, ensuring the physical stability of all the dynamic objects in a task is essential for overall stability. As discussed previously, when creating the layout constraint graph, the physical stability of dynamic objects is ensured by introducing supports. Additionally, during the last stage of generation, distracting objects are placed only in locations that can sufficiently support their stability. The generation process strictly enforces these stability checks, resulting in a 100% physical stability rate for tasks across all scenarios.

Intended Solvability

In this evaluation, we assess the solvability of the generated tasks using the intended sequence of physical interactions defined in the scenario used to generate the task. To determine the solvability rate, the solution of each task (i.e., the shooting angle of the bird) was recorded during the task generation process and then executed using a playtest agent. Notably, the final step of the generation process that usually validates solvability was skipped in this experiment to evaluate how often the proposed methodology generates a solvable task. The solvability rate was calculated as the percentage of tasks that can be solved using their intended solutions. The results, shown in Table 8, demonstrate that the solvability rate decreases as the number of interactions in the scenario increases. Specifically, the highest rate is 100% for tasks with two interactions, while the lowest rate is 82% for tasks with eight interactions.

Accidental Solvability

There is a potential for unintended interactions leading to solving the tasks in a continuous physical environment, which can be exploited by AI agents, thereby limiting the credibility of the evaluations conducted using those tasks. We propose the accidental solvability evaluation to assess the vulnerability of the generated tasks to unintended solutions. The evaluation involves the deployment of a heuristic agent, that systematically shoots at all dynamic blocks in the game level, excluding the target block for the solution. The heuristic of shooting at different types of objects located at different positions is widely used by many Angry Birds agents (Stephenson et al. 2018).

$$AS_i = \frac{1}{N_i} \sum_{n=1}^{N_i} \frac{1}{P_n} \sum_{p=1}^{P_n} (S_{np}) \qquad (1)$$

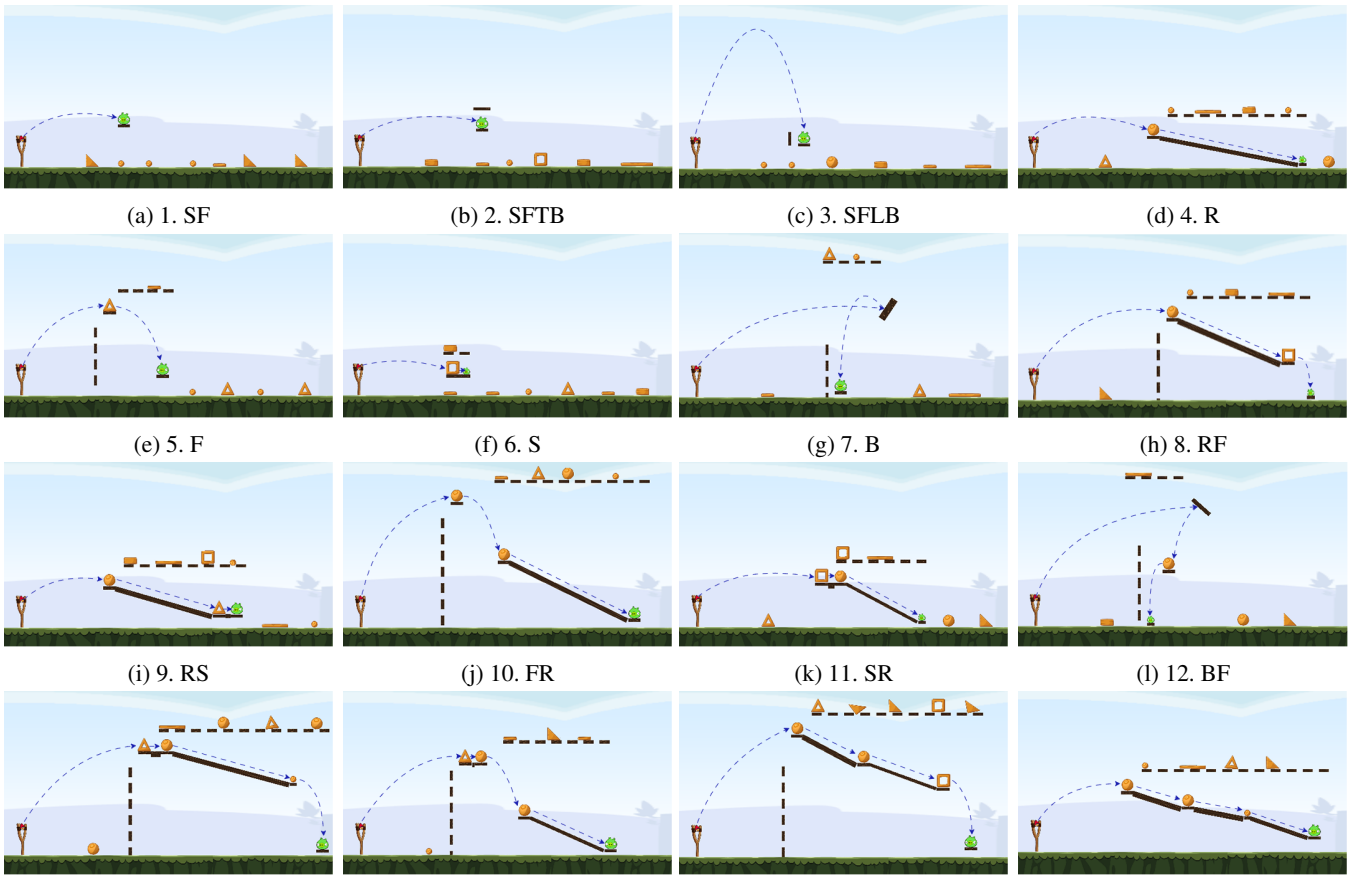
The accidental solvability rate (AS_i) of a scenario i is calculated using Equation 1, where N_i is the total number of levels tested, P_n is the total number of plays used to test the n^{th} level, and S_{np} is 1 if the n^{th} level is solved by the p^{th} strategy, or 0 otherwise. The AS_i value ranges between 0 and 1, and a higher value indicates a higher vulnerability to unintended solutions. The results shown in Table 8 demonstrate a decreasing trend in accidental solvability rate as the interaction count of the scenario increases. Specifically, the highest rate is 12% for tasks with two interactions, while the lowest is 3% for tasks with eight interactions.

Generation Time

The task generator is implemented in C# within the Unity game engine and integrated into the Angry Birds clone, Science Birds. The simulation-based components of the generator were executed by accelerating the physics engine by a factor of five. The experiments were performed on a Windows 10 desktop computer with an i9-9900KS CPU and 64GB RAM. The runtime of the generator was measured by recording the time taken to produce tasks for each sample scenario. The last row in Table 8 presents the results of the runtime analysis, indicating that the generation time increases linearly with the complexity of the scenario.

Conclusion and Future Work

This research presented a novel method for generating physical tasks using a systematic approach that leverages physical scenarios defined as a causal sequence of physical interactions between objects. We first introduced a grammar con-



(a) 1. SF (b) 2. SFTB (c) 3. SFLB (d) 4. R

(e) 5. F (f) 6. S (g) 7. B (h) 8. RF

(i) 9. RS (j) 10. FR (k) 11. SR (l) 12. BF

(m) 13. SRF (n) 14. SFR (o) 15. RRF (p) 16. RRR

Figure 5: Generated tasks for the 16 example scenarios. Arrows show the object trajectories when the solution is executed.

Evaluation Metric	Physical Interaction Count in the Scenario			
	Two	Four	Six	Eight
Intended Solvability Rate	1.00 ± 0.00	0.91 ± 0.11	0.87 ± 0.10	0.82 ± 0.07
Accidental Solvability Rate	0.12 ± 0.05	0.08 ± 0.03	0.07 ± 0.01	0.03 ± 0.01
Generation Time (Seconds)	1.83 ± 0.10	2.37 ± 0.27	3.16 ± 0.86	4.68 ± 1.03

Table 8: The results of the intended solvability, accidental solvability, and generation time evaluations (Mean±SD).

sisting of object, interaction, restriction, and layout grammars to support defining scenarios and object configuration in physical environments. The physical scenarios defined, adhering to this grammar, were used as input to the task generation process. In the task generation phase, the qualitative spatial relationships between objects were inferred, and a spatial constraint graph was constructed. For the narrowed-down generative space using spatial constraints, simulation-based constrain satisfactions were performed to determine the precise positions of the objects. We conducted evaluations on the generated tasks, focusing on physical stability, intended solvability, and accidental solvability. Additionally, we analyzed the task generation time. The results indicate that the proposed method is competent in generating tasks for a given causal sequence of physical interactions, achieving satisfactory performance across the evaluated metrics.

Moving forward, we foresee several potential paths of improvement for this research. One such direction would be to expand the current limitations of the process, allowing the generation of tasks that consider multiple interaction sequences with interdependent effects. Additionally, the proposed methodology can be directly applied to physics-based action domains like PHYRE, OGRE, and Virtual Tools, where the agent's action involves a one-time event, similar to shooting a bird in Angry Birds. Furthermore, investigating the applicability of this approach in domains such as Animal-AI testbed (Crosby et al. 2020), where the agent performs continuous interventions in the environment to solve tasks, holds promise for future exploration. Overall, this research opens up a new line of inquiry in the field of physics-based task generation, and its outcomes will be useful in developing AI with better physical reasoning capabilities.

References

Ahmed, O.; Träuble, F.; Goyal, A.; Neitz, A.; Bengio, Y.; Schölkopf, B.; Wüthrich, M.; and Bauer, S. 2021. CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning. In *9th International Conference on Learning Representations (ICLR)*.

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11): 832–843.

Allen, K. R.; Bakhtin, A.; Smith, K.; Tenenbaum, J. B.; and van der Maaten, L. 2020. OGRE: An Object-based Generalization for Reasoning Environment. In *NeurIPS Workshop on Object Representations for Learning and Reasoning*.

Allen, K. R.; Smith, K. A.; and Tenenbaum, J. B. 2020. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47): 29302–29310.

Baillargeon, R.; and DeVos, J. 1991. Object Permanence in Young Infants: Further Evidence. *Child Development*, 62(6): 1227–1246.

Baillargeon, R.; Li, J.; Ng, W.; and Yuan, S. 2009. An account of infants' physical reasoning. *Learning and the infant mind*, 66: 116.

Baillargeon, R.; Stavans, M.; Wu, D.; Gertner, Y.; Setoh, P.; Kittredge, A. K.; and Bernard, A. 2012. Object individuation and physical reasoning in infancy: An integrative account. *Language Learning and Development*, 8(1): 4–46.

Bakhtin, A.; van der Maaten, L.; Johnson, J.; Gustafson, L.; and Girshick, R. 2019. PHYRE: A New Benchmark for Physical Reasoning. In *NeurIPS*.

Bear, D. M.; Wang, E.; Mrowca, D.; Binder, F. J.; Tung, H.-Y. F.; Pramod, R. T.; Holdaway, C.; Tao, S.; Smith, K.; Sun, F.-Y.; Fei-Fei, L.; Kanwisher, N.; Tenenbaum, J. B.; Yamins, D. L. K.; and Fan, J. E. 2021. Physion: Evaluating Physical Prediction from Vision in Humans and Machines. arXiv:2106.08261.

Bliss, J.; and Ogborn, J. 1994. Force and motion from the beginning. *Learning and Instruction*, 4(1): 7–25.

Cheke, L. G.; Loissel, E.; and Clayton, N. S. 2012. How Do Children Solve Aesop's Fable? *PLOS ONE*, 7(7): 1–12.

Chen, J.; Cohn, A.; Dayou, L.; Wang, S.; Ouyang, J.; and yu, Q. 2013. A survey of qualitative spatial representations. *The Knowledge Engineering Review*, 30: 106–136.

Clementini, E.; and Di Felice, P. 1997. Approximate topological relations. *International Journal of Approximate Reasoning*, 16(2): 173–204.

Crosby, M.; Beyret, B.; Shanahan, M.; Hernández-Orallo, J.; Cheke, L.; and Halina, M. 2020. The animal-AI testbed and competition. In *Neurips 2019 competition and demonstration track*, 164–176. PMLR.

Diezmann, C. M.; and Watters, J. J. 2000. Identifying and Supporting Spatial Intelligence in Young Children. *Contemporary Issues in Early Childhood*, 1(3): 299–313.

Emery, N. J.; and Clayton, N. S. 2009. Tool use and physical cognition in birds and mammals. *Current Opinion in Neurobiology*, 19(1): 27–33. Cognitive neuroscience.

Ferreira, L.; and Toledo, C. 2014. A Search-based Approach for Generating Angry Birds Levels. In *Proceedings of the 9th IEEE International Conference on Computational Intelligence in Games*, CIG'14.

Frank, A. U. 1991. Qualitative Spatial Reasoning with Cardinal Directions. In Kaindl, H., ed., *7. Österreichische Artificial-Intelligence-Tagung / Seventh Austrian Conference on Artificial Intelligence*, 157–167. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-46752-3.

Gamage, C.; Pinto, V.; Renz, J.; and Stephenson, M. 2021a. Deceptive Level Generation for Angry Birds. In *2021 IEEE Conference on Games (CoG)*, 572–579. IEEE.

Gamage, C.; Pinto, V.; Xue, C.; Stephenson, M.; Zhang, P.; and Renz, J. 2021b. Novelty generation framework for AI agents in angry birds style physics games. In *2021 IEEE Conference on Games (CoG)*, 1–8. IEEE.

Gamage, C.; Pinto, V.; Xue, C.; Zhang, P.; Nikonova, E.; Stephenson, M.; and Renz, J. 2023. NovPhy: A Testbed for Physical Reasoning in Open-world Environments. *arXiv preprint arXiv:2303.01711*.

Hong, Y.; Yi, L.; Tenenbaum, J.; Torralba, A.; and Gan, C. 2021. Ptr: A benchmark for part-based conceptual, relational, and physical reasoning. *Advances in Neural Information Processing Systems*, 34: 17427–17440.

James, S.; Ma, Z.; Arrojo, D. R.; and Davison, A. J. 2020. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026.

Liu, X.; Shekhar, S.; and Chawla, S. 2001. Maintaining Spatial Constraints Using a Dimension Graph Approach. *International Journal on Artificial Intelligence Tools*, 10(04): 639–662.

Renz, J.; Ge, X.; Stephenson, M.; and Zhang, P. 2019. AI meets Angry Birds. *Nature Machine Intelligence*, 1.

Riochet, R.; Castro, M. Y.; Bernard, M.; Lerer, A.; Fergus, R.; Izard, V.; and Dupoux, E. 2020. IntPhys 2019: A Benchmark for Visual Intuitive Physics Understanding. *ArXiv*, abs/1803.07616.

Shaker, N.; Shaker, M.; and Togelius, J. 2013. Evolving playable content for cut the rope through a simulation-based approach. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 9, 72–78.

Stephenson, M.; and Renz, J. 2017. Generating varied, stable and solvable levels for Angry Birds style physics games. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 288–295. IEEE.

Stephenson, M.; and Renz, J. 2019. Agent-based adaptive level generation for dynamic difficulty adjustment in angry birds. *arXiv preprint arXiv:1902.02518*.

Stephenson, M.; Renz, J.; Ge, X.; and Zhang, P. 2018. The 2017 AIBIRDS Competition. *ArXiv*, abs/1803.05156.

Stephenson, M.; Renz, J.; Ge, X.; and Zhang, P. 2021. Generating Stable Building Block Structures From Sketches. *IEEE Transactions on Games*, 13: 1–10.

Taveekitworachai, P.; Abdullah, F.; Dewantoro, M. F.; Thawonmas, R.; Togelius, J.; and Renz, J. 2023. ChatGPT4PCG Competition: Character-like Level Generation for Science Birds. *arXiv preprint arXiv:2303.15662*.

Valenza, E.; Leo, I.; Gava, L.; and Simion, F. 2006. Perceptual Completion in Newborn Human Infants. *Child Development*, 77(6): 1810–1821.

Wolf, F. B. N. N. J. M. G. M. C. 2020. Cophy: Counterfactual Learning of Physical Dynamics. In *ICLR*.

Xue, C.; Pinto, V.; Gamage, C.; Nikonova, E.; Zhang, P.; and Renz, J. 2023. Phy-Q as a measure for physical reasoning intelligence. *Nature Machine Intelligence*, 5(1): 83–93.

Yi, K.; Gan, C.; Li, Y.; Kohli, P.; Wu, J.; Torralba, A.; and Tenenbaum, J. B. 2020. CLEVRER: Collision Events for Video Representation and Reasoning. In *International Conference on Learning Representations*.