A GAMES

We present the initial scenes of all the 40 games in Figs. A2 to A5. The games vary in different positional setups. We keep invariant the dynamic properties like density and friction to emphasize the interactive aspects of the task.

B GAME DIFFICULTY IN TERMS OF SAMPLING

To evaluate the game difficulty in `I-PHYRE`, we measure the number of iterations required to gather 50 successful action sequences that do not repeat for each game. As an approximate estimation, the more iterations random sampling needs, the more difficult it is to solve; see Fig. A1 for results. The compositional games show significant difficulty compared with those in the other three splits, especially the seesaw angle and activated pendulum. These games have stringent requirement on execution timing: Players and agents may miss the perfect timing easily. Another interesting discovery is that the games related to the seesaw show higher difficulty regardless of the split. The analysis of game difficulty demonstrates a clear alignment with human performance, as humans tend to achieve lower scores in games that are more challenging to sample. See Tab. A4 for human scores in detail.

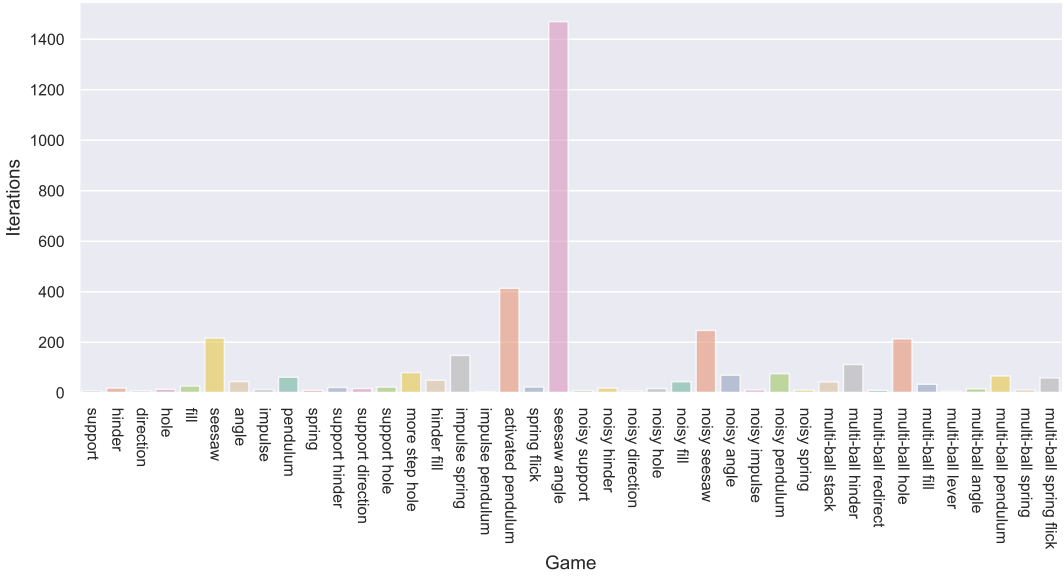
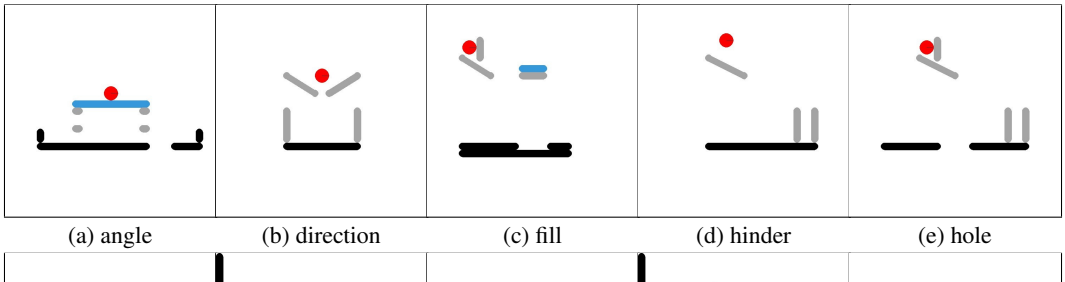
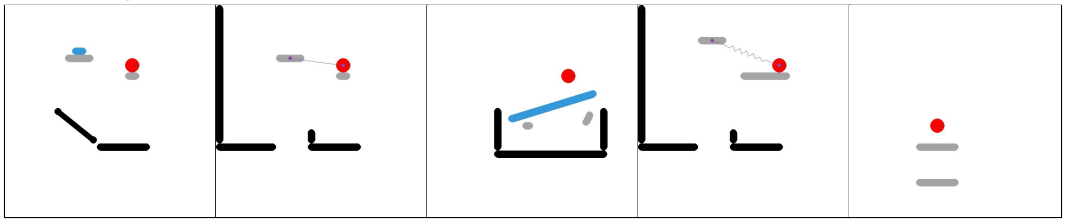


Figure A1: **The iteration numbers required to generate 50 different successful action sequences.**

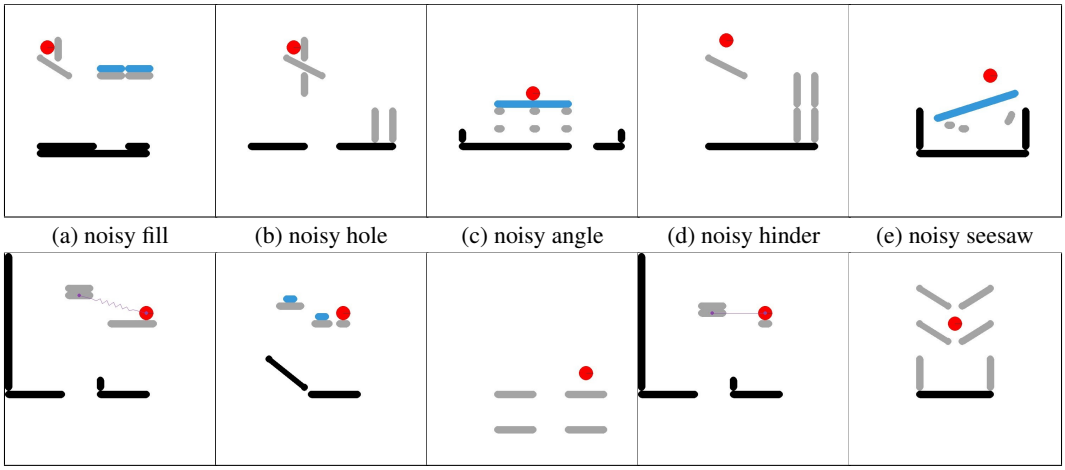


(a) angle (b) direction (c) fill (d) hinder (e) hole



(f) impulse (g) pendulum (h) seesaw (i) spring (j) support

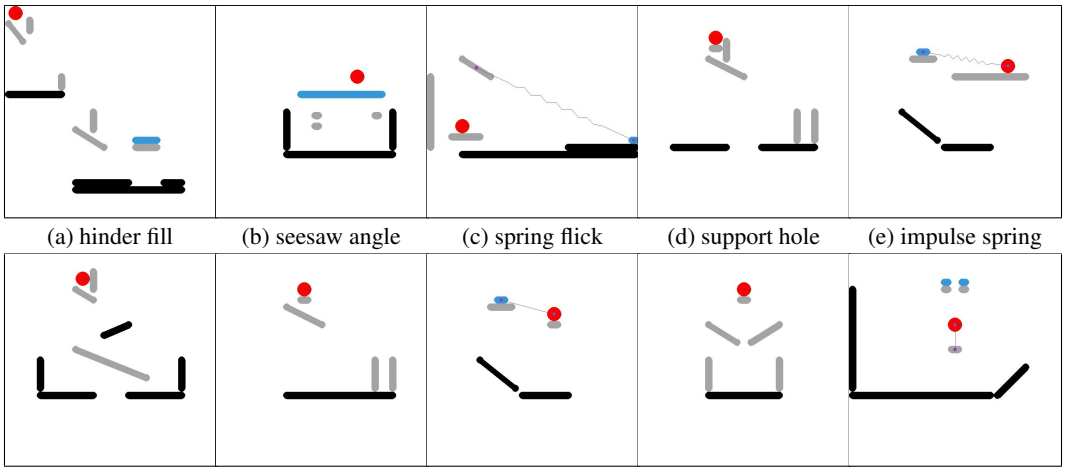
Figure A2: **The initial scenes of basic games.**



(a) noisy fill (b) noisy hole (c) noisy angle (d) noisy hinder (e) noisy seesaw

(f) noisy spring (g) noisy impulse (h) noisy support (i) noisy pendulum (j) noisy direction

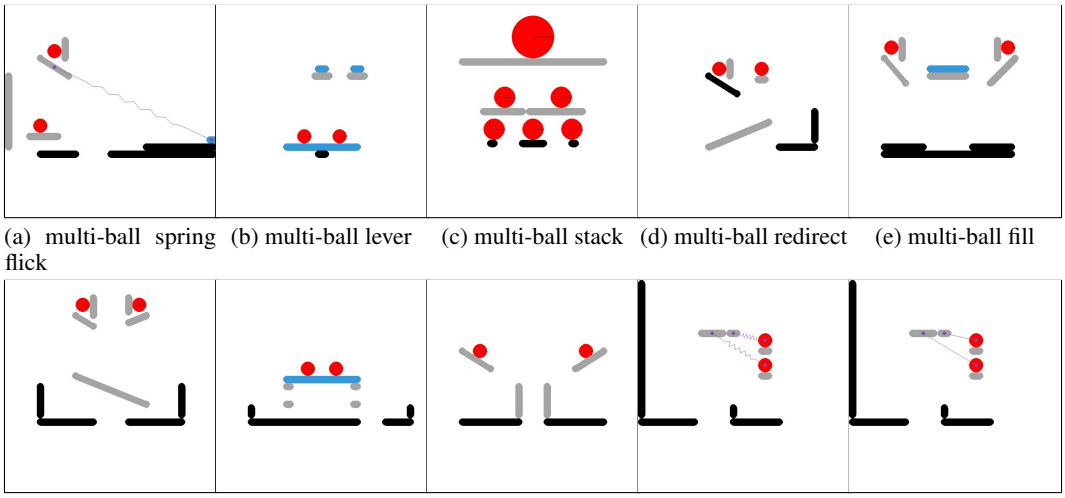
Figure A3: **The initial scenes of noisy games.**



(a) hinder fill (b) seesaw angle (c) spring flick (d) support hole (e) impulse spring

(f) more step hole (g) support hinder (h) impulse pendulum (i) support direction (j) activated pendulum

Figure A4: **The initial scenes of compositional games.**



(a) multi-ball spring flick (b) multi-ball lever (c) multi-ball stack (d) multi-ball redirect (e) multi-ball fill

(f) multi-ball hole (g) multi-ball angle (h) multi-ball hinder (i) multi-ball spring (j) multi-ball pendulum

Figure A5: **The initial scenes of multi-ball games.**

C TRAINING DETAILS OF MODEL-FREE LEARNERS

We run all our experiments on RTX 3090 GPUs. The simulator produces the next state, reward, and termination indicator from the action per time step.

Architectures We adopt three distinct strategies in training: planning in advance, planning on-the-fly, and the combined strategy.

In the planning in advance strategy, the combination of initial observation of the game and the entire action space serves as the model's input, while the output consists of a continuous-valued vector whose dimension equals to the maximum number of actions (6 in our case). Each vector element represents the normalized action timing. We implement model-free reinforcement learning algorithms, namely Proximal Policy Optimization (PPO-I), Advantage Actor-Critic (A2C-I), Soft Actor-Critic (SAC-I), and Deep Deterministic Policy Gradient (DDPG-I), to generate continuous action values in accordance with this setup.

In the planning on-the-fly strategy, at each time step, the model's input comprises the current observation combined with the entire action space, while the output consists of the probability for each possible action, including no action. The action with the highest probability is executed during inference time at each step. Following this approach, we implemented model-free reinforcement learning algorithms Proximal Policy Optimization (PPO-O), Advantage Actor-Critic (A2C-O), Soft Actor-Critic (SAC-O), and Deep Q-Network (DQN-O).

For the combined strategy, the model's input is the fusion of the current game observation and the entire action space, while the output is the same as that of the planning in advance strategy. This single-step procedure is analogous to the planning in advance strategy; however, after executing the first action, the entire action distribution is updated based on the current observation. Employing this framework, we implemented model-free reinforcement learning algorithms Proximal Policy Optimization (PPO-C), Advantage Actor-Critic (A2C-C), and Soft Actor-Critic (SAC-C).

The policy architecture of model-free learners is MLP with two hidden layers of size 256 and the activation function is tanh.

Learning The observation for a scene is processed in the symbolic space, represented as a 12×9 matrix. Each row denotes one object's features in the scene. We use the symbolic representation of objects instead of visual input since symbolic representation consists of all the necessary components to do reasoning and visual information does not contribute any additional useful information for this particular task but introduces uncertainty and noise into the planning process.

The object feature vector consists of all the essential components for interactive physical reasoning:

- **Object Position**: The spatial coordinates of the objects with four scalars representing the two endpoints for bars or two duplicate centers for balls.
- **Object Size**: The radius of the object.
- **Eliminable Indicator**: An indication of whether the object can be eliminated in the given context.
- **Fixed Object Indicator**: The identification of whether the object is stationary and can not be moved due to gravity.
- **Joint Indicator**: An indicator of whether the object is connected to a joint.
- **Spring Indicator**: An indicator of whether the object is connected to a spring.

The action space is the concatenation of the positions of the objects that can be removed from the scene, padded to the maximum number of 6. Since different games have different action spaces, we concatenate the action space with the observation space as input to enable generalizable reasoning with a single agent.

The same object features and action space are used in the other learning paradigms as well.

A2C-I, A2C-C, and SAC-C are trained with a learning rate of 1×10^{-5}. All other models are trained with a learning rate of 1×10^{-6}. SAC-I is trained for 57k steps. SAC-O and SAC-C are trained for 80k steps. A2C-I are trained for 426k steps. Other models are trained for 800k steps.

Results Please refer to Sec. 4.2 for analysis. The detailed rewards are listed in Appx. J.

D SUPERVISED LEARNERS

A commonly used metric in evaluating physical reasoning (Qi et al., 2021; Bakhtin et al., 2019; Girdhar et al., 2020; Li et al., 2022) is the accuracy of a classifier model to correctly predict the outcome of an action. In this part, we follow the same protocol: whether a supervised classifier can predict the outcome of an action sequence based on the initial scene.

Architectures We formulate the problem as a binary classification task, wherein a model predicts whether an action sequence will succeed. We consider three different general architectures: Global Fusion, Object Fusion, and Vision Fusion. Of note, this paradigm falls into the category of planning in advance. Each model takes as input the action sequence and the initial scene configuration and outputs the success probability. The Global Fusion model embeds the entire action sequence and all object states and fuses them together using multiple MLPs. The Object Fusion model embeds each action and symbolized object independently and fuses them, both using MLPs. The Vision Fusion model is similar to the Global Fusion model except that it takes the pre-trained ViT (Dosovitskiy et al., 2020) features of the initial scenes as extra inputs.

Learning We generate 50 successful and 50 failed action sequences for each game by random sampling. The generation iteration is regarded as an approximate estimate of the difficulty of games. To simplify the action space, we discretize 15 seconds into 150 time steps. Models are trained on the basic split and are tested on the other three splits. The object features are the same as the ones in Appx. C. The supervised agents are trained for 200 epochs with a batch size of 16, with the learning rate annealed from 1×10^{-3} to 1×10^{-6} using a cosine scheduler.

Results Tab. A1 tabulates the performance of each supervised learning model, measured by mean accuracy across the test games. Of note, a random classifier should reach about 50% accuracy due to an equal number of positive and negative samples. However, all supervised learning agents fail to show notably improved performance compared with a random guess. Among the models, the Object Fusion model shows the best generalization compared to others due to fine-grain embeddings for actions and object states. Adding visual features from a pre-trained ViT does not necessarily improve the overall performance in the Vision Fusion model, though we do find enhanced accuracy when generalized to multiple balls.

Table A1: **Performance of different supervised learning models on `I-PHYRE` measured by mean accuracy (%).** Models are trained on the basic split.

Agent	Bas.	Noi.	Comp.	Mul.
Global Fusion	87.5	59.8	57.0	56.7
Object Fusion	71.4	**60.1**	**60.2**	54.1
Vision Fusion	86.4	57.5	55.8	**59.5**

These experimental results demonstrate that training supervised learning models on naively sampled data cannot endow agents with interactive physical reasoning ability.

E MODEL-BASED AND OFFLINE RL LEARNERS

We apply the model-based World Model (Ha & Schmidhuber, 2018) and the offline Decision Transformer (DT) (Chen et al., 2021) to the `I-PHYRE`.

Architectures In the model-based method, we pre-train an MDRNN to predict the next state given the current state and action. The predicted states are concatenated with the current states as guidance for learning policies. We use PPO, SAC, and A2C as controllers. In the offline method, we use GPT-2 as the backbone to learn the mapping from states to actions autoregressively.

Learning We used 50 successful and 50 failed actions per game to train the model-based and offline models. Models are trained on the basic split and tested on the other three splits. For the model-based learner, the prediction module of MDRNN is trained for 50 epochs with a learning rate of 1×10^{-3}, a batch size of 128, and a sequence length of 64. The prediction module is then fixed and served as an additional part of the observation. The offline RL learner is trained for 100 epochs with a batch size of 128, with the learning rate annealed from 1×10^{-3} to 1×10^{-6} using a cosine scheduler.

Results As shown in Fig. A6, the results indicate that the model-based method doesn't improve the performance of PPO, SAC, and A2C, potentially due to the fact that RNN cannot learn an appropriate

physical dynamics model to accurately predict the next state. The offline method is also unsatisfactory; the Decision Transformer planning on-the-fly learns a conservative strategy that takes no action at all. Although the performance may improve by increasing the data size, we argue that delicate modeling of physics is essential to learning better dynamics and emerging interactive physical reasoning.

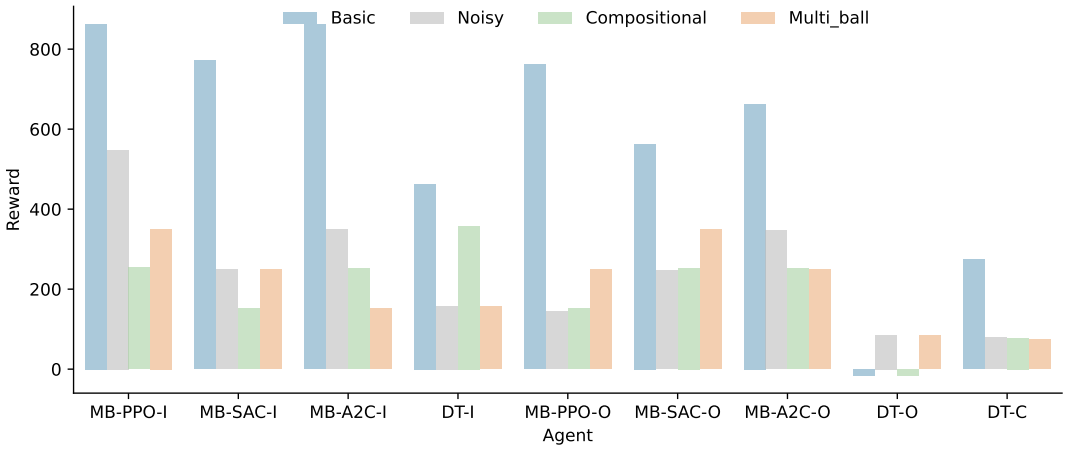


Figure A6: **Performance of model-based and offline RL learners.**

F WITHIN-TEMPLATE GENERALIZATION

To see the generalization capabilities of current intelligent agents in games with similar structures, we developed an unseen basic split consisting of an additional 10 games. These games resemble the basic ones but differ in object positions and angles. As with the previous setting, we evaluate A2C-I, PPO-I, A2C-C, PPO-C, A2C-O, and PPO-O that are trained on the basic split. The agents' performance in these modified games is detailed in Tab. A2. The performance nearly matches the basic split but exceeds the performance of the noisy, compositional, and multi-ball splits. These results indicate that current agents are adept at generalizing similar tasks but struggle with generalization across different game templates. Thus, we focus specifically on the challenge of generalizing to entirely new tasks in `I-PHYRE`.

Table A2: **Performance of agents on 10 similar tasks to basic games, measured by average rewards.**

Split	Random	A2C-I	PPO-I	A2C-C	PPO-C	A2C-O	PPO-O
Basic	360.33	862.47	861.56	765.26	862.78	461.2	663.91
Unseen basic	360.17	862.25	763.13	662.72	660.51	561.75	662.12

G LARGE LANGUAGE MODEL

In this interactive physical reasoning environment, GPT-4 is tasked not only to do physical reasoning but also to plan and take actions at specific times. This is quite a challenge for GPT-4 with only the initial states as input configuration. We prompt GPT-4 with detailed game rules and object features. The results show that GPT-4 can successfully finish some preliminary tasks like support and noisy support but fail on other tasks. The average rewards of different folds are in Tab. A3.

Table A3: **Performance of GPT-4 on `I-PHYRE` measured by average rewards.**

Agent	Bas.	Noi.	Comp.	Mul.
GPT-4	75.17	64.17	-29.10	77.49

The preliminary results suggest that the large language model cannot perform well in this interactive reasoning task, although they may be good at understanding physical concepts and utilizing physical heuristics. The interactive physical reasoning scenarios challenge GPT-4 to reason on both spatial information and precise action timing. Studies need to combine quantitative methods to further extend its capability to do physical reasoning, especially in terms of interactive reasoning.

H ANALYSIS ON FAILURE CASES

To gain more insight into the interactivity in physical reasoning, we delve into failure cases specifically related to action order and timing. We assume that, to solve the game in an oracle way, one should first decide the correct action order and then the precise action timing. The situations when actions in the wrong order can still solve the game are beyond our discussion. Thus, the failure may come from two sources: (i) wrong action order and (ii) wrong action timing with the correct order. We want to explore to what extent the agents failed due to those two reasons respectively. We benchmark against established baselines: PPO, A2C, and SAC. We count three terms:

- **Success Number (SN):** The number of scenes in a split where agents could solve puzzles.
- **Right Order Number (RON):** The number of scenes in a split where agents aligned with the Oracle in terms of action order.
- **Success from Right Order Number (SRON):** The number of scenes that the agent solved and also matched the Oracle's action order.

With a total number of games $Total$, the percentage in failure cases of wrong action timing with the correct order, denoted as $P(T \mid F)$, can be calculated as:

$$P(T \mid F) = \frac{RON - SRON}{Total - SN}. \tag{A1}$$

The percentage in failure cases with wrong action order, denoted as $P(O \mid F)$, can be calculated as:

$$P(O \mid F) = 1 - P(T \mid F). \tag{A2}$$

The percentage of two failure sources in PPO, A2C, and SAC are shown in Fig. A7.

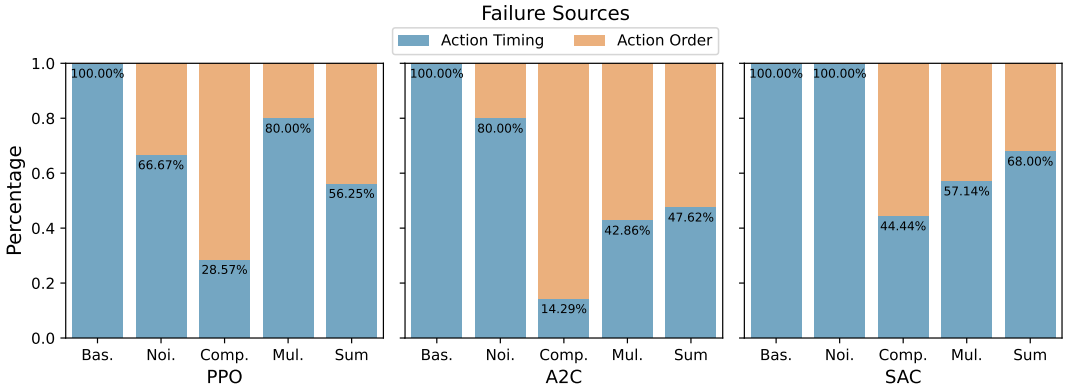


Figure A7: **The percentage of two failure sources in `I-PHYRE`.**

Our findings suggest that, of the cases solved, many of them are executed with the optimal action order. For cases not solved, about half of them are due to correct action order but wrong action timing on average (56.25%, 47.62%, and 68.00%), suggesting that both action order and action timing are important. When examining basic games, we find that learning action order is much easier than learning action timing since all of the failures come from wrong action timing. Additionally, for compositional games, the main challenge lies in executing actions in a more reasonable order, while in noisy games, the primary hurdle is enhancing the accuracy of action timing. In summary, our observations indicate that while action order significantly influences puzzle-solving success, the nuances of action timing prove particularly challenging for RL agents to get better performance. We hope this analysis can provide some of the future directions for model design from the temporal aspect.

I DETAILED DISCUSSION ABOUT PHYSICS MODELING

Physics modeling serves as a crucial tool for comprehending and engaging with the physical realm, and its integration is critical in the progression of machine learning agents (Zhu et al., 2020; Duan

et al., 2022). The primary branches of physics modeling include physics-based simulation, physics-informed methodologies, and intuitive commonsense modeling.

The implementation of physics modeling can be achieved through physics-based simulations (Battaglia et al., 2013; Kubricht et al., 2016). Due to the recent developments in computational capabilities and efficiency, it is now possible to accurately simulate an array of physical phenomena in real time. These physics engines offer a consistent and comprehensive platform where agents can employ these principles to foresee future states. Despite its practicality, physics-based simulation presents its own challenges, especially regarding the computational demands for precisely depicting complex physical systems and grasping latent variables in partially observed environments (Ludwin-Peery et al., 2021). Moreover, it necessitates expert knowledge to construct, and it's not capable of learning from experience.

Another method is incorporating explicit physics constraints directly into the agent by employing physics-informed neural networks (Raissi et al., 2019; Cuomo et al., 2022). Physical laws and principles are used as constraints or guides in the learning process. It can help in building more robust models that are less prone to overfitting and ensure that the model's predictions adhere to established physical laws, making them more interpretable and reliable. This approach helps mitigate accumulated estimation errors. However, it might not generalize well to unseen scenarios due to its reliance on predefined physical laws.

Supplementing learning of physical dynamics with commonsense reasoning is another method. Agents learn not just how object states evolve, but also the commonsense or intuitive understanding of how these objects behave under certain circumstances (Piloto et al., 2022; Kubricht et al., 2017; Dasgupta et al., 2021; Weihs et al., 2022). This involves leveraging the vast amount of implicit knowledge humans possess about the world and encoding it into agents and could potentially lead to more robust and generalizable models.

However, each method has its challenges and limitations, and further research is required to develop more efficient and effective ways of incorporating physics modeling into agents.

J Complete results of humans and RL agents

We show the detailed game rewards from RL agents, human participants, and a random agent. For straightforward comparison, we average the rewards in 40 games; see Tab. A4 for details.

Table A4: **All results of humans and different RL agents on `I-PHYRE` benchmark, measured by rewards of gameplay.**

Game	Human	Random	DDPG-I	DQN-O	A2C-I	A2C-O	A2C-C	PPO-I	PPO-O	PPO-C	SAC-I	SAC-O	SAC-C
support	975.96	977.6	968	970.3	973.6	970.6	974.3	973.6	977.6	973	969.6	977.6	975.9
hinder	971.57	-45	-25	972.5	962.5	962.5	962.5	962.5	962.5	962.5	962.5	-45	972.5
direction	971.89	953.3	-55	947.3	958.4	963.3	959.7	949	953.3	948.9	963.3	953.3	970
hole	966.08	-55	955.1	952.5	950.1	-55	-55	949.7	962.5	960	959.6	953.9	966
fill	970.89	-45	-35	-45	957.1	-45	-35	958.4	-45	958.6	-45	-45	969.5
seesaw	436.36	-35	-25	-35	-35	-35	-35	-35	-35	-35	-35	-35	-35
angle	770.47	-55	-45	948.2	952.9	949.3	962.9	950	949.4	950	958.3	951.6	-45
impulse	970.47	972.3	966.5	973.5	967.5	971.3	967.8	967.8	972.3	967.7	966.9	972.1	-35
pendulum	972.56	-35	969	-35	966.9	-35	971.7	968.7	-35	971.2	968.2	-35	-35
spring	973.84	970.1	969.6	-35	970.7	-35	970.6	970.9	976.5	970.9	984.2	-35	-35.1
noisy_support	977.09	957.6	949	947.9	-45	957.6	-45	952.5	954.4	953.8	952.9	957.6	960.3
noisy_hinder	967.33	-65	-55	952.5	951.9	942.3	962.5	938.9	942.5	952.5	-65	-65	-45
noisy_direction	972.81	928.7	-45	-25	940.6	-75	950.2	928.5	-75	938.6	940.1	-75	-55
noisy_hole	966.43	-65	955.4	-55	950.3	946	959.5	941.3	942.5	951.5	-65	-65	-65
noisy_fill	971.47	-55	-45	-15	-45	-55	-35	-45	-55	-55	-55	-55	-55.1
noisy_seesaw	455.88	-45	-25	-45	-45	-45	-45	-45	-45	-45	-45	-45	-35
noisy_angle	565.88	-75	-45	-25	-75	-75	931.4	935.3	-75	941.3	-75	-75	-45
noisy_impulse	968.36	-45	-35	-35	956.7	-45	957	956.2	962.3	956.9	-45	-35	-45
noisy_pendulum	972.59	-45	-35	-15	-45	-45	-45	961.4	-45	-45	-45	-45	-45.1
noisy_spring	974.47	-45	-35	983.1	960.8	-45	-45	961.9	966.6	963.3	966.1	-45	984.3
support_hinder	961.54	-55	-55	-25	946.1	-55	-45	946.8	-55	945.9	-45	-55	-35
support_direction	963.07	-65	956	-45	-35	-65	951.3	-55	-65	936.2	-65	-65	-55
support_hole	957.88	-65	-45	-25	-65	945	-45	-65	-65	-65	-65	-65	-45.1
more_step_hole	969.51	-45	-45	-25	-45	965	-45	973.5	963.6	-45	-45	-45	-45
hinder_fill	954.2	-75	-55	-35	-65	-75	-55	-65	-75	-65	938.3	-75	-55
impulse_spring	281.96	-35	-35	-25	-35	-35	-35	-35	-35	-35	-35	-35	-35
impulse_pendulum	975.79	972.8	966.3	-25	966.3	-35	969.2	966.5	973.4	966.2	967.5	972.3	968.7
activated_pendulum	577.84	-45	-45	-15	-45	-45	-45	-45	-45	-45	-45	-35	-45
spring_flick	935.83	-45	966	966	-45	974.9	-35	-45	-45	-45	958.6	963.9	975.1
seesaw_angle	409.22	-45	-35	-15	-45	-45	-45	-45	-45	-45	-45	-45	-35
multi_ball_stack	612.59	-45	-35	-25	-35	969.9	-35	958.5	-45	-45	-45	957.3	-45.1
multi_ball_hinder	487.67	948	-45	-25	-45	-55	-35	-55	-55	-55	-55	-45	-55
multi_ball_redirect	919.63	964.7	-35	-15	961.1	962.4	961.2	-45	964.5	962.4	955.5	964.5	-35
multi_ball_hole	673.51	-65	-45	-45	-65	-65	-55	-55	-65	-55	-65	-65	-65
multi_ball_fill	957.84	-65	-45	-45	936.8	-65	-55.1	-65	936.1	-45	-65	-65	-65
multi_ball_lever	980.33	969.7	990.8	990.8	968.9	969.9	970.8	969.1	974.5	970.7	980.8	976.9	980.8
multi_ball_angle	927.3	-55	-45	-45	-55	951.5	-45	-55	-55	-55	-55	-55	969.7
multi_ball_pendulum	928.52	-55	-45	-35	-55	-55	-45	-55	-55	-55	-55	-55	-45
multi_ball_spring	896.18	-55	-45	-25	-45	-55	960.5	947.9	-55	950.9	960.6	-55	-45.1
multi_ball_spring_flick	624.17	-55	956	-25	-55	945.9	-45	-55	-55	-45	-55	-55	-55
average reward	844.17	200.87	260.19	242.99	429.36	352.69	383.45	528.10	377.74	504.33	378.45	227.15	233.93
success rate	87.55%	25.00%	30.00%	27.50%	47.50%	40.00%	42.50%	57.50%	42.50%	55.00%	42.50%	27.50%	27.50%