Novelty Generation Framework for AI Agents in Angry Birds Style Physics Games

Chathura Gamage
School of Computing
The Australian National University
Canberra, Australia
chathura.gamage@anu.edu.au

Vimukthini Pinto
School of Computing
The Australian National University
Canberra, Australia
vimukthini.inguruwattage@anu.edu.au

Cheng Xue
School of Computing
The Australian National University
Canberra, Australia
cheng.xue@anu.edu.au

Matthew Stephenson
Dpt. of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
matthew.stephenson@maastrichtuniversity.nl

Peng Zhang
School of Computing
The Australian National University
Canberra, Australia
p.zhang@anu.edu.au

Jochen Renz
School of Computing
The Australian National University
Canberra, Australia
jochen.renz@anu.edu.au

***Abstract*—Handling novel situations is a critical capability of Artificial Intelligence (AI) agents when working in open-world physical environments. To develop and evaluate these agents, we need realistic and meaningful novelties, that is, novelties that are detectable and learnable. However, there is a lack of research in the area of creating novelties for AI agents in physical environments. Physics-based video games are popular among AI researchers due to the ability to create realistic and controllable physical environments. In this paper, we present a systematic novelty generation framework for physics-based video games. This framework allows the user to define a specific objective when generating novel content that ensures detectability. We instantiate the proposed framework for the video game Angry Birds and conduct experiments to show that the generated novel content is consistent with the user-defined objectives. Furthermore, we use a reinforcement learning agent to experiment with the learnability of the generated novel content.**

***Index Terms*—open-world learning, novelty generation, physics based video games, Angry Birds**

I. Introduction

Detecting and reacting to novel and unforeseen situations is a key feature of human intelligence and is still a challenge for modern AI systems. Open-world learning is an emerging research area that attempts to address this challenge [1]. With the increasing reliance on autonomous systems operating in open-world environments, such as self-driving vehicles, underwater exploration robots, and drone swarms, embedding novelty detection and novelty adaption capabilities to AI is becoming more important. For example, a trained self-driving car knows to stop at the red stop signs, but it may fail to respond if it approaches a blue stop sign, which is not included in its training data [2].

To proceed with research in open-world learning, we need agents that can detect and adapt to novelty and environments where novelty can be easily introduced in a controllable way. As real-world environments offer limited opportunities to inject novelty and conduct controlled experiments [1], we need test environments that simulate realistic novelties. Physics simulation games, video games where the game world simulates real-world physics, offer simplified and controlled environments for developing and testing AI agents [3]. Therefore, physics simulation games are ideal platforms to introduce realistic novelties and conduct controlled experiments of open-world learning systems.

While novelty generation has been investigated in domains such as Monopoly [4] and Polycraft [5], to the best of our knowledge, there are no existing automated approaches to generate novelties in physics-based environments. In this paper, we present the first systematic novelty generation framework for physics-based environments. The framework generates meaningful novelties for AI agents. We consider a novelty to be meaningful if the novelty is detectable and learnable. The detectability and learnability are to ensure that agents can find and adapt to the novelty and improve performance over tasks. The pipeline of the proposed framework is shown in Fig. 1. The framework consists of two components which can, (1) generate novel game objects for a given user-defined objective and (2) inject generated novel objects to the game content preserving the compatibility of the content for the game. We use Angry Birds as the test environment to introduce our framework as it is a popular physics-based game for developing AI agents, with the long-running AI competition as part of the IJCAI conference [6].

The rest of the paper is structured as follows. We start by providing the definition of novelty in the context of AI and discuss related work in the areas of AI agent experimentation and video games. Next, we present the overview of the proposed novelty generation framework followed by the application to Angry Birds with the tests used to generate requested novel content. We then present examples of generated novel content from our framework followed by experiments with AI agents in Angry Birds to evaluate if the generated novel content meets the user-defined requirements.

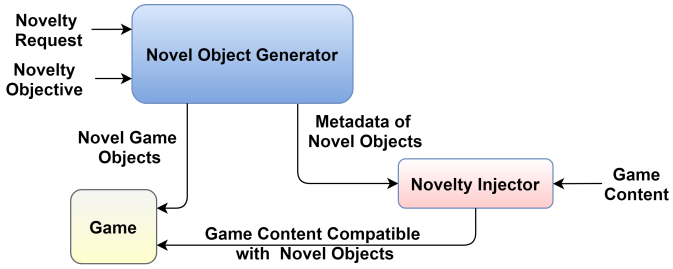


Fig. 1: Overview of the proposed novelty generation framework. The *Novel Object Generator* generates novel game objects according to a *Novelty Request* satisfying a user-defined *Novelty Objective*. The *Novelty Injector* places the generated novel objects in game content ensuring the compatibility of the content in the context of the game.

II. Background and Related Work

A. What is Novelty?

Consider an unmanned underwater vehicle trained for near-surface coastal areas. The vehicle has expertise for missions in the trained setting. But when it enters a region with new types of rocks or an environment with extremely hot water, the vehicle may fail to respond as the vehicle has not experienced these situations before [1]. "Novelty" is described as situations that violate implicit or explicit assumptions about the agents, the environment, or their interactions [7]. Following this, [1] and [8] explain different types of novelties that may occur in open-world environments. In [5] novelty is explained as a relative property to an agent's past experience and cognitive capabilities. When an agent encounters an entity, if the agent cannot recall the entity from prior experience, or the agent cannot infer the entity through cognition, the encountered entity is considered novel for the agent.

Following the novelty categorization in [5], we define two types of parameters available in the game objects of physics-based games as appearance parameters and physics parameters. Appearance parameters are the shape, size, and colour of an object which affects the visual appearance of the object. Physics parameters are the mass, friction, bounciness, and etc., which determine the physics characteristics of the object. In this paper, we consider two categories of novel objects which can be encountered in physics-based games,

- **Category 1:** Objects with new appearance parameter values to the already encountered objects in the game. Compared to already encountered objects, these objects do not have additional parameters and they may or may not have the same physics parameter values.
- **Category 2:** Objects with the same appearance parameter values to the already encountered objects in the game, but different physics parameter values.

B. Related Work

There is prior work on novelty generation such as generating new types of symbols for digits [9] and designing molecules that can be used as drugs [10]. As open-world learning is an emerging research area [1], [8], there is only a limited number of prior work in generating novelties for AI agents. As discussed previously, Monopoly [4] and Polycraft [5] domains have been used to create novelties for open-world agents. However, their approaches to novelty generation are not publicly available.

Fig. 2: A level from the Angry Birds game.

In the context of content generation for video games, Procedural Content Generation through Quality Diversity (PCG-QD), is a sub-branch of search-based procedural content generation [11], which generates a substantial number of high quality varied artifacts. In PCG-QD, the underlying mechanism is evolutionary computation for multidimensional optimization. For the divergent search, evolution is not optimized using a fitness function which satisfies the eventual goal, but rewarded for the diversity of the solutions. This concept has been termed by various researchers as increasing generality [12], surprise [13], curiosity [14], or novelty [15]. PCG-QD has been used in video games to generate 3D objects [16], weapons [17], and dungeons [18]. One of the major drawbacks of PCG-QD is that it has to explore multiple solutions in a multi-dimensional space which can affect the performance of the algorithm. Also, it is strenuous to find diversity from all the solutions obtained across the search space [19]. In our proposed work, the diversity that is introduced to the game from the novel content is guaranteed through a series of tests.

C. Angry Birds

Angry Birds is a physics-based puzzle game in which the player shoots birds from a slingshot at pigs. These pigs are often covered by different physical structures and the goal is to kill all the pigs using the provided birds. A sample Angry Birds level is shown in Fig. 2. Instead of the real Angry Birds game environment, we used an open-source research clone of Angry Birds developed in Unity by Ferreira [20]. The main objects that are available in this game can be split into four types: birds (red, blue, yellow, white, and black), blocks (ice, wood, and stone), pigs, and explosives.

While Angry Birds seems simple and easy to play for human players, it is particularly challenging for AI agents due to its large and effectively continuous action space, as well as unknown consequences for actions in advance [21]. A successful AI agent not only needs to learn the physical properties of game objects to correctly predict the outcome of an action but also needs to choose the desired action from the action space. To achieve the same or better performance as compared to the best human players, agents need to address a number of issues from different areas of AI. Therefore, the impact of successfully addressing these issues goes far beyond

Angry Birds and will be an indispensable part of intelligent agents that operate in the physical world.

Together with its popularity and importance in AI, the ability to mimic the real-world environment in Angry Birds makes it an ideal platform to add realistic novel content to facilitate open-world learning research. Therefore, we instantiate our novelty generation framework in the Angry Birds domain.

III. Overview of the Proposed Novelty Generation Framework

In this section, we present a high-level overview of the proposed novelty generation framework. As shown in Fig. 1 the framework consists of two main modules *Novel Object Generator* and *Novelty Injector*.

A. Novel Object Generator

Novel Object Generator module takes two types of inputs, a *Novelty Request* and a *Novelty Objective*. The *Novelty Request* contains the category of the novel object that is needed to be generated and the amount of novel game content to be generated. The *Novelty Objective* is a request made by the user that describes the expected change to the game by introducing the novel object. *Novelty Objective* includes a set of game objects to be converted to novel objects, the physics parameters of the objects to adjust, and the expected outcome after introducing the novel objects. A *Novelty Objective* example is *reducing the agent's task performance by 10% by changing the mass of the game object A.*

Novel Object Generator generates game objects that belong to one of the two novel object categories defined in Section II-A. The most critical task of this module is determining the appropriate values of the physics parameters and this is done through a series of automated tests. The steps of the testing procedure are summarized below:

- **Input Analysis:** Analyze the *Novelty Objectives* and obtain the game objects and the physics parameters to adjust.
- **Value Range Selection:** Determine the number of steps and the step size to increase and decrease the parameter values from the original values.
- **Constraint Satisfaction:** Apply constraints to the selected parameter values to ensure they are realistic.
- **Test Content Generation:** Create game content by using the selected values of the parameters.
- **Test Formulation:** Formulate tests to quantify the effect due to the change of the parameters.
- **Test Execution:** Perform the tests based on the specified *Novelty Objectives*.
- **Result Analysis:** Based on the test results, determine appropriate values for the physics parameters.

It is not always guaranteed that any given *Novelty Objective* can be satisfied. From the results of the testing procedure, the framework determines whether there are suitable values, which satisfy the provided *Novelty Objectives*, for the given parameters of the novel game objects. If so, the generation proceeds. If not, the generation aborts and concludes that the *Novelty Objective* is not achievable.

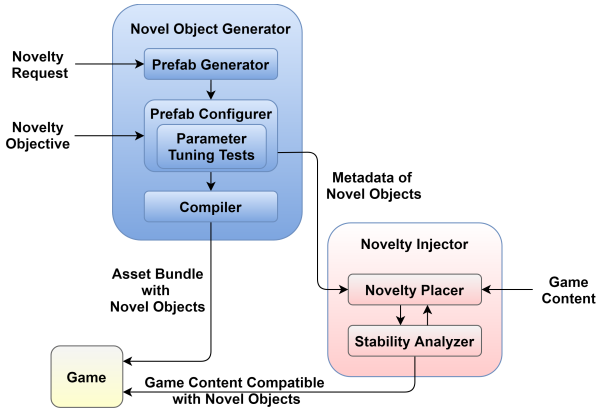


Fig. 3: Architecture of the novelty generation framework in Unity.

The outputs of this module are the generated novel game objects, which are sent to the game and metadata, which contains properties of novel objects and is used by the *Novelty Injector*.

B. Novelty Injector

Novelty Injector takes two inputs as (1) metadata of the generated game objects from the *Novel Object Generator* and (2) game content to place novel game objects. The game content can be already available content for the game or content generated from content generators. *Novelty Injector* places the generated novel objects in the game content. Metadata of the generated novel objects are taken as an input to this module. The spatial constraints of an object should be satisfied to successfully place it in game content. Also, the novel content should satisfy the constraints of the game. Therefore, the usability of the game content after placing the novel objects is validated through a game specific feasibility analysis. This analysis ensures that the output game content from this module is compatible to load to the game.

Finally, the novel objects from the *Novel Object Generator* and game content compatible with the novel objects created from the *Novelty Injector* are ready to be loaded into the game in real-time.

IV. Instantiating Novelty generation Framework to Angry Birds

In this section, we discuss how the proposed novelty generation framework is instantiated to Angry Birds. Fig. 3 shows the overall architecture of the framework in Unity. Subsequent sections discuss the modules of the framework in detail.

A. Novel Object Generator

Unity has an inbuilt mechanism, which allows storing objects within the game, including the objects' attached components and parameters. Game objects created from this mechanism are termed as prefabs. Prefabs act as templates

that can be instantiated multiple times and can be configured individually in the game scenes. We create novel objects as Prefabs. The *Novel Object Generator* module comprises three submodules, namely the *Prefab Generator*, *Prefab Configurer*, and *Asset Bundle Compiler*.

1) Prefab Generator: This module gets a *Novelty Request* as an input which specifies the required category of the novel object and the number of novel game levels that is needed to generate. According to the category of the novel object, *Prefab Generator* submodule either creates a new prefab or selects an existing prefab from the game.

- The *Prefab Generator* creates a new prefab if a category 1 novel object is to be introduced. The visual image (sprite) of the prefab is selected from an existing sprite sheet provided beforehand. The type of the game object (e.g., a bird, a block) that needs to be generated is obtained through the *Novelty Request*. To initialize the physics parameters of the new prefab, the available range of the parameter values of existing same-type game objects are considered. Then, each parameter is set by randomly selecting a value from the available range. The modular architecture of the framework facilitates a user to replace this module with other prefab generation tools.
- If the generation is a category 2 novel object, which has the same appearance of an existing object in the game, then the submodule selects the prefab that corresponds to the object received from the *Novelty Request*.

2) Prefab Configurer: This submodule obtains a *Novelty Objective* as the input. Then *Prefab Configurer* conducts automated tests to determine the physics parameter value to accomplish the given *Novelty Objective*. For Angry Birds, we considered four example expected effects of introducing novelty. These effects, example *Novelty Objectives* from these effects, and the testing procedure are discussed in Section V in detail. For a given effect of introducing novelty, a game object, and a physics parameter, the tests in the testing procedure need to be conducted only once and the data can be stored and reused, which allows the framework to perform in real-time.

After determining the parameter value, the novel prefab is configured with the selected value. The outputs of the *Prefab Configurer* are a set of configured novel prefabs and a message with metadata of the configured prefabs.

3) Asset Compiler: The final step of the *Novel Object Generator* is compiling the generated novel objects from the *Prefab Configurer* and make them ready to send to the game. This is done by the *Asset Compiler* submodule. It compiles all the configured novel objects into a single file named *Asset Bundle*. An *Asset Bundle* is an archive file that contains assets that can be loaded at the runtime of the game.

B. Novelty Injector

The two tasks of the Novelty Injector module are (1) introducing the generated novel game objects to existing game content and (2) guaranteeing the game is stable after introducing the novelty. Novelty Injector module comprises two submodules for the above two tasks as *Novelty Placer* and *Stability Analyzer*.

1) Novelty Placer: This module is used to place novel objects in existing game content. It processes already available game content (game levels or scenes) and performs a feasibility study to place the novel objects. In this initial version of our framework, *Novelty Placer* only consider the spatial constraints of the object as the feasibility criteria. If it finds a candidate object with matching dimensions in the game, then that object is replaced by the novel object. Otherwise, the novel object is placed in an unoccupied region in the game scene which satisfies the spatial requirements.

2) Stability Analyzer: Novel objects introduced to the game should satisfy physical stability requirements to prevent the game scene from collapsing. Since the *Novelty Placer* submodule does not consider parameters such as friction, mass, etc., when placing the novel object, the game levels can still be unstable although the spatial constraints are satisfied. To ensure the stability of the scene, *Stability Analyzer* simulates the underlying physics simulator in Unity that is used for the game to determine whether the new scene is stable in a physical sense. If the game scene is not stable, that means the added novel objects to the game level is not compatible. Then the novel game object injection is retried by replacing another game object or with a different placement.

V. Angry Birds Novelty Objectives

In this section, we discuss the *Novelty Objectives* that we consider in the Angry Birds domain. As discussed under Section III-A, our focus in this work is to generate meaningful novelties that achieve a pre-defined *Novelty Objective*, which allows us to ensure detectability of novelty. Novelty Objective includes the expected outcome after introducing the novelty (novelty outcome), the game objects to be converted to novel objects (novelty objects), and the parameters of the objects to tune (novelty parameters). In the context of Angry Birds, we assume a reasonable agent should be able to detect the changes in at least one of the following four effects:

- The game score
- The level passing rate
- The dispersion of the game objects after a bird shot
- The velocity of the game objects after a collision

Therefore, we consider the expected outcome of introducing novelty from these four effects when defining the *Novelty Objectives*. In Angry Birds, there are multiple object types and multiple physics parameters associated with each object. There is a large space of *Novelty Objectives* that can be defined with the availability of multiple game object types and physics parameters. Therefore, even with these four expected effects, the framework can handle a substantially large set of *Novelty Objectives*. Possible examples of *Novelty Objectives* in Angry Birds are (1) reducing the mean score of game levels by 10% by changing the friction of ice blocks and (2) reducing the level passing rate by 15% by changing the mass of pigs.

The *Novelty Objectives* should be defined such that the novelties are detectable to AI agents. The thresholds to ensure

the detectability of novelty is determined using our prior knowledge in the domain. For example, the game score should deviate by at least 5% to make a detectable change for the agents. A user's *Novelty Objective* input to the framework is validated using these threshold values.

A. Novelty Outcomes

In this section, we describe how to calculate parameter values for a given novelty outcome associated with any of the four novelty effects mentioned in the previous section. For the calculation, data should be recorded using an Angry Birds playing agent. In this calculation, for category 1 novel objects, we assume that an agent is capable of detecting the type of the novel object introduced (e.g., a bird, a block). Hence, for category 1 novel objects, we only consider the impact of the physics parameter changes without the impact of the appearance change.

A novelty outcome associated with any of the above four effects is presented as a null hypothesis. The framework tests this hypothesis for each novelty parameter value across a fixed number of levels, both with and without novelty added, by using a paired sample t-test (except for the level passing rate which uses a two-sample proportion test). Results are obtained from the t-test for each novelty parameter value at 5% level of significance. The novelty parameter values which do not reject the null hypothesis are the values that satisfy the *Novelty Objective*. Out of the novelty parameter values which satisfy the *Novelty Objective*, the value which has the highest p-value is selected as the output of the test. If none of the novelty parameter values satisfy the *Novelty Objective*, the framework considers the provided *Novelty Objective* as incompatible.

1) Impact on the Game Score:

$$H_0 : \frac{\mu_{base} - \mu_{novel}}{\mu_{base}} = x\% \tag{1}$$

where, μ_{base} is the population mean score of the base levels (levels without novelty) and μ_{novel} is the population mean score of the levels with novelty.

2) Impact on the Level Passing Rate:

$$H_0 : P_{base} - P_{novel} = x\% \tag{2}$$

where, P_{base} is the population proportion of passing the base levels and P_{novel} is the population proportion of passing the levels with novelty.

3) Impact on the Dispersion of the Game Objects:

Dispersion is referred to as the scattering of objects when a force is applied. In Angry Birds, agents can interact with the game objects only by shooting birds. The location of all the blocks are recorded after a single bird shot and when all the game objects become stationary. The following formulation is used to measure the dispersion of objects and find the novelty parameter value which satisfies the *Novelty Objective*.

For the level index j, location of the blocks in the j^{th} base level after a single bird shot and when all the game objects become stationary can be represented as a list of vectors.

$$[(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k), \ldots, (x_n, y_n)]$$

where, $j = \{1, \ldots, N\}$, N = total number of levels used for one parameter value, n = total number of blocks in j^{th} level, $k = \{1, \ldots, n\}$, and (x_k, y_k) are the x and y location coordinates of the k^{th} block.

Similarly, for the corresponding j^{th} novel level, the list of location vectors of dispersed blocks is,

$$[(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_k^*, y_k^*), \ldots, (x_n^*, y_n^*)]$$

The list of Euclidean distances between the two location vectors for each object can be represented as,

$$[d_1, d_2, \ldots, d_k, \ldots, d_n]$$

where,

$$d_k = \sqrt[2]{(x_k - x_k^*)^2 + (y_k - y_k^*)^2} \tag{3}$$

Thus, for i^{th} parameter value where $i = \{1, 2, \ldots, m\}$, m = number of parameter values ($m = 22$ in this example) and for the j^{th} game level, the mean Euclidean distance of all the objects across the level is,

$$d_{ij}^* = \frac{\Sigma_{k=1}^{n} d_k}{n} \tag{4}$$

Thus, the *dispersion factor* for parameter value i,

$$\hat{D}_i = \frac{\Sigma_{j=1}^{N} d_{ij}^*}{N} \tag{5}$$

Based on this *dispersion factor* definition, the null hypothesis:

$$H_0 : \frac{D_{base} - D_{novel}}{D_{base}} = x\% \tag{6}$$

where D_{base} is the *dispersion factor* of the base levels and D_{novel} is the *dispersion factor* of the levels with novelty.

4) Impact on the Velocity of the Game Objects:

The change in the velocity of the game objects after a collision is a detectable change to the dynamics in a physical environment.

The same procedure used to calculate the *dispersion factor* in the above test is used to calculate the *velocity factor*. The velocity data of the blocks are obtained after 0.5 seconds from the first collision of the bird that is shot. With the (x_k, y_k) components as the velocity across x and y directions of the k^{th} block, the same calculation can be used to compute the *velocity factor* $\hat{V}_i$ of the i^{th} novelty parameter value. Based on this *velocity factor* definition, the null hypothesis is,

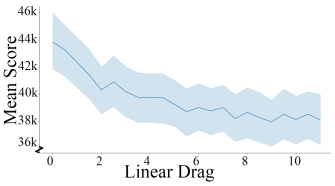
$$H_0 : \frac{V_{base} - V_{novel}}{V_{base}} = x\% \tag{7}$$

where V_{base} is the *velocity factor* of the base levels and V_{novel} is the *velocity factor* of the levels with novelty.

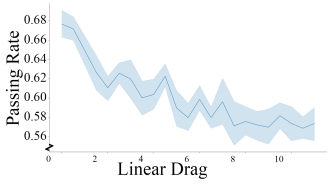
B. Novelty Objective Example

To illustrate our proposed testing procedure on a specific example, we consider the *Novelty Objective*: reduce the game score by 10% by changing the linear drag of wood blocks. Linear drag is the tendency of an object to slow down due to friction with the air or water surrounding it [22]. For the tests, we used an Angry Birds playing agent that shoots randomly either to a pig or an explosive, and the game levels were generated using a state-of-the-art Angry Birds level generator [23]. The automated procedure in Section III-A, applied for this specific *Novelty Objective* in Angry Birds, is as follows:

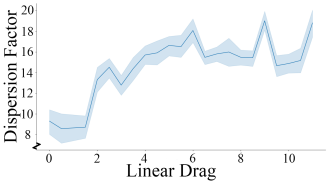
- **Input Analysis:** the game object and the physics parameter provided in the *Novelty Objective* are wood blocks and linear drag respectively.



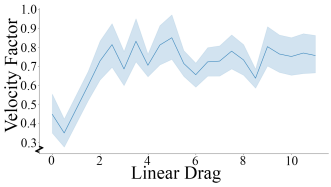
(a) Mean score of game levels



(b) Passing rate of game levels



(c) Dispersion factor of blocks



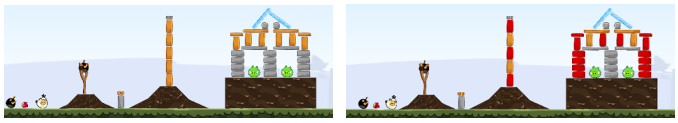
(d) Velocity factor of blocks

Fig. 4: Variation in our four novelty outcomes as the linear drag of wood blocks changes.

- **Value Range Selection:** with the original value of linear drag 1.0, consider the number of steps as 40 (i.e., 20 each upwards and downwards), and step size as 0.5. The set of selected values P for the parameter is: $P = \{1.0 \pm 0.5 \times n \mid n \in \mathbb{Z}^+, \text{ and } n \leq 20\}$.
- **Constraint Satisfaction:** the realistic values for linear drag $\in [0, \infty)$. Therefore, the range is truncated as P_T: $P_T = \{p_i \mid \forall p_i \in P \text{ and } p_i \geq 0\} = \{0, 0.5, 1.5, 2.0, 2.5, \ldots, 10.5, 11\}$
- **Test Content Generation:** there are 22 novel parameter values (P_T) to be tested. A set of 200 random base levels are obtained using the level generator. Then the linear drag value of wood blocks in each base level is adjusted for the 22 selected parameter values separately (i.e., a set of 200 levels for each selected parameter value).
- **Test Formulation:** the test is formulated to measure the impact of the game score as presented in Section V-A1: $H_0 : (\mu_{base} - \mu_{novel})/\mu_{base} = 10\%$.
- **Test Execution:** conduct the test using the game playing agent. The agent plays each level 10 times and the mean score of each level is calculated. This is done to minimize the effect of possible random variation in scores.
- **Result Analysis:** select the most suitable value that satisfies the given *Novelty Objective* (i.e., the best value that reduces the score by 10% at 5% level of significance).

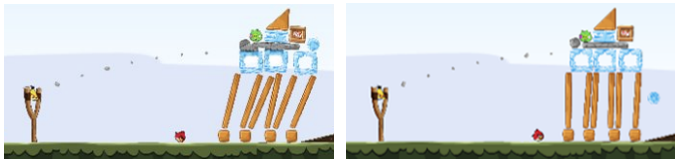
After this procedure, it is identified that changing linear drag of wood blocks to 2.0, most satisfies our *Novelty Objective*.

To give a more complete representation of this testing procedure on all novelty outcomes mentioned in Section V-A, Fig. 4 shows the recorded variation in the mean score, level passing rate, dispersion factor, and velocity factor with the change in linear drag. From this, we can see that each of our novelty outcomes is affected by changing the linear drag of wood blocks. Please note, that there is nothing special about the linear drag of wood blocks, and our *Novelty Objective* could involve any game object or physics parameter, this is just one specific example to demonstrate our procedure.



(a) Base level (b) Novel level

Fig. 5: A base game level (left) and the corresponding game level with category 1 game objects generated from the framework (right) which has objects (dark red coloured blocks) with different appearance to the existing game objects.



(a) Base level (b) Novel level

Fig. 6: A base game level (left) and the corresponding game level with category 2 novel game objects (right) after the same bird shot. The novel game objects have the same appearance to the wood blocks (brown-coloured blocks), but increased linear drag value.

VI. Results and Evaluations

A. Examples of Generated Novelties

Screenshots of game levels generated from our framework with the two categories of novel objects are shown in Fig. 5 and Fig. 6. A game scene with category 1 novel red-coloured blocks is shown in Fig. 5. The *Novelty Objective* used for the generation was to reduce the dispersion of blocks by 15% by adjusting the linear drag of the new block type. Fig. 6 shows a level with category 2 novel objects. The novelty objective was reducing the velocity factor of blocks by 12% by changing the linear drag value of wood blocks. The effect on the wood blocks after the same bird shot is shown for the base level and its corresponding novel level. Because the linear drag is increased in the novel level, the structure with wood blocks does not collapse for the same shot, which collapsed the structure in the base level.

B. Evaluating the Generated Novelties

As an evaluation for generated novelty from the framework, we performed experiments using three AI agents which competed in the AIBirds competition to show that generated novel content satisfies the corresponding novelty objective.

The three agents used for the experiment are,

- *Naïve Agent*: Naïve agent uses the strategy of shooting directly at the pigs. It fires the given bird to any pig selected at random using either a low or high trajectory, which is also selected at random [24].
- *Datalab*: The winner of 2014 and 2015 AIBirds competitions uses four strategies to solve game levels. The strategies are to destroy pigs, physical structures, explosives, and round blocks. The decision of which strategy

TABLE I: Results of the three agents for the two experiments with two *Novelty Objectives*. (1) reduce the mean score of game levels by 20% by adjusting the linear drag parameter of wood blocks and (2) reduce the passing rate of game levels by 20% by adjusting the linear drag parameter of wood blocks.

Agent	Exp 1: mean score			Exp 2: passing rate		
	Base	***Novel***	***Redu.***	***Base***	***Novel***	***Redu.***
Naive Agent	39,987	32,241	19%	0.44	0.28	16%
Datalab	53,042	42,101	21%	0.76	0.54	22%
Eagle's Wing	43,261	32,796	24%	0.48	0.30	18%

to use is based on the game level, possible trajectories, currently selected bird, and remaining birds [25].

- *Eagle's Wing*: Eagle's Wing, which is the winner of 2017 and 2018 competitions makes its shooting decision based on five strategies. The five strategies are shoot pigs, destroy explosives, destroy most blocks, shoot high round objects, and destroy structures strategy [26].

The *Novelty Request* we considered was to generate 50 novel game levels with category 2 novel objects. The base game levels that were used by the framework were 50 game levels generated from the state-of-the-art Angry Birds level generator. The agent used within the initialization of the framework was the agent mentioned in Section V-B. In our first experiment, the *Novelty Objective* was reducing the mean score of the game levels by 20% by adjusting the linear drag value of wood blocks, while the *Novelty Objective* of the second experiment was reducing the passing rate of the game levels by 20% by adjusting the linear drag value of wood blocks. For the first experiment, we ran the three agents and recorded the scores they obtained for the 50 base game levels and for the 50 novel game levels. For the second experiment, we ran the three agents and recorded the passed game level count for the 50 base levels and for the 50 novel levels.

The results of the three agents for the two experiments are shown in Table I. From the results, it can be seen that the framework could achieve the expected *Novelty Objective* consistently across different agents.

C. Experiments Using a Learning Agent

We now test our second requirement for meaningful novelty: learnability. We, therefore, conducted an additional experiment using an Angry Birds reinforcement learning agent, called DQ-Birds [27] to show that the generated novel content is learnable, even though the learnability is not explicitly guaranteed. An AI agent that can reasonably learn to perform in a physical environment should also be able to learn to perform in the same environment with a different physical parameter configuration. Since the novelties generated by the framework can be seen as a new physical parameter configuration of the environment, we expect a learning agent can learn to adapt to the novel environment.

Firstly, we created six data sets for the experiment:

- Set A - 300 randomly generated base levels for training
- Set B - 50 randomly generated base levels for validation

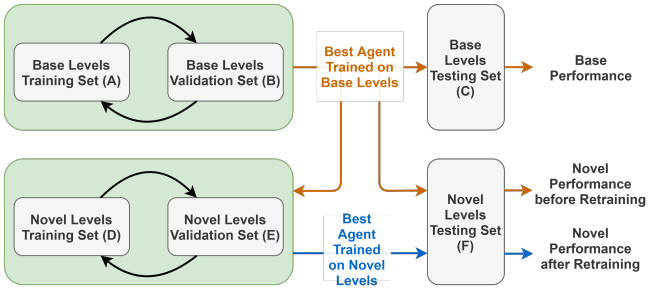


Fig. 7: Evaluation process of the DQ-Birds agent.

TABLE II: Mean scores for different versions of the DQ-Birds agent.

Agent Status	Mean Score
Base Performance	37,133
Novel Performance before Retraining	30,834
Novel Performance after Retraining	34,613

- Set C - 50 randomly generated base levels for testing
- Set D - 300 novel levels by adding novelty to the set A with the *Novelty Objective* of reducing the mean score of the game levels by 20% by adjusting the linear drag value of wood blocks
- Set E - 50 novel levels by adding novelty to the set B with the same novelty objective as in set D
- Set F - 50 novel levels by adding novelty to the set C with the same novelty objective as in set D

Secondly, we conducted the experiment. The experimentation process is shown in Fig. 7. We trained the agent on the set A while evaluating on the set B. We obtained the best performing agent model, which is the model with the highest mean score in the set B. The trained agent was tested on the set C (*Base Performance*). Then, we tested the agent on the set F (*Novel Performance before Retraining*). Next, we trained the agent on set D while evaluating on set E. We obtained the best performing agent model, which is the model with the highest mean score in the set E. Finally, the trained agent was tested on set F (*Novel Performance after Retraining*).

Base Performance is the mean score of the best performing agent on the base game levels. *Novel Performance before Retraining* is the mean score of the agent for the novel levels without training on novel levels. The difference between these two results is the impact on the agent by the introduced novelty. *Novel Performance after Retraining* is the mean score of the agent after training on novel game levels. The difference between the *Novel Performance after Retraining* and *Novel Performance before Retraining* quantifies the effect of learning. If this difference is positive we claim the generated novel content is learnable.

Table II shows the results of our third experiment. After training on the base-level training set, the agent achieved a mean score of 37,133 for the base-level test set and it was reduced by 16.96% when the same agent was tested on the novel test-set. Next, the agent could increase the mean score

from 30,834 to 34,613, representing a 12.25% improvement after training on the novel levels set. This shows that the novel content generated from the framework is learnable.

D. Discussion

The results of these experiments show that the framework can generate novel content satisfying the novelty objective for three agents with different strategies. However, the agent used for the novelty generation can be switched with an agent that deploys multiple strategies representing the strategies of the existing suite of Angry Birds agents. This will make the initialization of the framework more capable to generate novel content that satisfies the novelty objective of a broader range of agents. The second set of experiments done using the reinforcement learning agent is a preliminary analysis conducted to show that the generated novel content is learnable. As the agent we used is based on the standard deep Q-learning algorithm, we expect that the state-of-art reinforcement learning agents should at least achieve similar performance. Future experiments can be done with different learning agents and different novelty objectives to study and compare the learning capabilities of the agents.

VII. Conclusion

In this paper, we proposed the first systematic novelty generation framework for physics-based video games. We discussed the modules of the novelty generation framework in general and we instantiated it for a research clone of Angry Birds. We conducted two experiments to show that the generated novelties are consistent with user-defined objectives. We conducted another experiment using a learning agent to show that the generated novelties are learnable. In future, we plan to extend the framework to generate novelties, which ensures a change to the agent's strategy. We believe that this work will facilitate advancements in future AI agent developments in open-world environments.

Acknowledgments

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) and was accomplished under Cooperative Agreement Number W911NF-20-2-0002. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the DARPA or ARO, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] P. Langley, "Open-world learning for radically autonomous agents," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 13539–13543, 04 2020.

[2] K. Horner, "Polycraft team to lay groundwork for smarter AI." https://news.utdallas.edu/science-technology/polycraft-world-darpa-2020. [Accessed: Mar. 08, 2021].

[3] J. Renz and X. Ge, *Physics Simulation Games*, pp. 77–95. Singapore: Springer Singapore, 2017.

[4] M. Kejriwal and S. Thomas, "Generating novelty in open-world multi-agent strategic board games." https://nips.cc/Conferences/2020/Schedule?showEvent=20778, 2020. [Accessed: Mar. 23, 2021].

[5] M. Faizan, S. Vasanth, T. Gyan, G. Shivam, G. Saurav, G. Mateo, S. Jivko, and S. Matthias, "A Novelty-Centric Agent Architecture for Changing Worlds," *20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, vol. abs/1612.01608, 2021.

[6] J. Renz, X. Ge, S. Gould, and P. Zhang, "The Angry Birds AI competition," *AI Magazine*, vol. 36, pp. 85–87, Jun. 2015.

[7] DARPA-SAIL-ON, "Science of artificial intelligence and learning for open-world novelty (SAIL-ON)." https://iresearch-cms.tau.ac.il/sites/resauth.tau.ac.il/files/DARPA%20SAIL-ON_HR001119S0038.pdf. [Accessed: Apr. 10, 2021].

[8] T. Boult, P. Grabowicz, D. Prijatelj, R. Stern, L. Holder, J. Alspector, M. Jafarzadeh, T. Ahmad, A. Dhamija, C. Li, *et al.*, "A unifying framework for formal theories of novelty: Framework, examples and discussion," *arXiv preprint arXiv:2012.04226*, 2020.

[9] M. Cherti, B. Kégl, and A. Kazakçı, "Out-of-class novelty generation: an experimental foundation," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1312–1319, IEEE, 2017.

[10] M. Cherti, B. Kégl, and A. Kazakçı, "De novo drug design with deep generative models: an empirical study," in *International Conference on Learning Representations*, 2017.

[11] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, "Procedural content generation through quality diversity," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019.

[12] S. Risi and J. Togelius, "Increasing generality in machine learning through procedural content generation," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 428–436, 2020.

[13] D. Gravina, A. Liapis, and G. N. Yannakakis, "Surprise search: Beyond objectives and novelty," in *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, 2016.

[14] C. Stanton and J. Clune, "Curiosity search: Producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime," *PLoS ONE*, 2016.

[15] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, 2011.

[16] A. Khalifa, A. Nealen, S. Lee, and J. Togelius, "Talakat: Bullet hell generation through constrained map-elites," in *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, 2018.

[17] D. Gravina, A. Liapis, and G. N. Yannakakis, "Constrained surprise search for content generation," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2016.

[18] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius, "Empowering quality diversity in dungeon design with interactive constrained MAP-Elites," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019.

[19] A. Nguyen, J. Yosinski, and J. Clune, "Innovation engines: Automated creativity and improved stochastic optimization via deep learning," in *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, 2015.

[20] L. Ferreira and C. Toledo, "A search-based approach for generating Angry Birds levels," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2014.

[21] J. Renz, R. Miikkulainen, N. R. Sturtevant, and M. H. M. Winands, "Guest editorial: Physics-based simulation games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 2, pp. 101–103, 2016.

[22] U. Technologies, "Unity - scripting api: Rigidbody2d.drag." https://docs.unity3d.com/ScriptReference/Rigidbody2D-drag.html. [Accessed: Mar. 29, 2021].

[23] M. Stephenson and J. Renz, "Generating varied, stable and solvable levels for angry birds style physics games," in *2017 IEEE Conference on Computational Intelligence and Games, CIG 2017*, 2017.

[24] M. Stephenson, J. Renz, X. Ge, and P. Zhang, "The 2017 AIBIRDS competition," *ArXiv*, vol. abs/1803.05156, 2018.

[25] T. Borovička, R. Špetlík, and K. Rymeš, "Datalab Angry Birds AI." http://aibirds.org/2014-papers/datalab-birds.pdf. [Accessed: Mar. 31, 2021].

[26] T. J. Wang, "AI Angry Birds Eagle Wing." https://github.com/heartyguy/AI-AngryBird-Eagle-Wing. [Accessed: Mar. 31, 2021].

[27] E. Nikonova and J. Gemrot, "Deep q-network for angry birds," *arXiv preprint arXiv:1910.01806*, 2019.