Misleading Level Age for Ninja Dogs

Table of Contents

[1.Abstract 2](#_Toc112411559)

[2.0 Introduction 2](#_Toc112411560)

[2.1 Purpose 2](#_Toc112411561)

[2.2 Action 2](#_Toc112411562)

[3.0ACKGROUND AND RELATED WORK 3](#_Toc112411563)

[3.1 Ninja Dogs 3](#_Toc112411564)

[3.3 Deceptive Games 3](#_Toc112411565)

[3.4 Process the game 3](#_Toc112411566)

[3.5 TERMINOLOGY 4](#_Toc112411567)

[4.0WORKING 5](#_Toc112411568)

[4.1 PROPOSED PROCEDURE 6](#_Toc112411569)

[4.2 Module for Physical Entity Extraction 6](#_Toc112411570)

[4.2 Module for Physical Entity Analysis 7](#_Toc112411571)

[4.3 Level Validation Module 8](#_Toc112411572)

[5 UML Model 9](#_Toc112411573)

[5.1 Use Case Diagram 9](#_Toc112411574)

[5.2 Diagram 10](#_Toc112411575)

[6. Game Screen 10](#_Toc112411576)

[7. Conclusions and future work 13](#_Toc112411577)

[Bibliography 13](#_Toc112411578)

# 1.Abstract

The Ninja Dogs AI competition has been going on for a while now to encourage the development of artificial intelligence experts that can play Ninja Canine's levels better than human gamers. Over the course of the opposition's existence, a broad variety of experts with various techniques have been employed to do this task. Despite the fact that these specialists' presentations have essentially grown over the last few years, they actually exhibit major disadvantages while playing deceptive levels. This is due to the fact that most practicing professionals focus on selecting the greatest photo to take next rather than planning a workable sequence of images. We provide a computerized method to create challenging game levels for Ninja Canines in order to facilitate advancements in such professionals. Although there are several material generators for Ninja Canines already, they don't focus on creating deceptive levels. In addition, we describe metrics to assess the stability, resolvability, and degree of difficulty of the developed levels.

In this project, we mainly focus on developing a vision module for agents playing the Ninja dogs computer game. [1] To achieve this, a range of common methods in image processing and computer vision was employed. The result is a fast, reliable vision module that outputs accurate descriptions for most of the known objects. The game features a single level, parallax backgrounds, animations, and pinch-to-zoom ability.

***Index Terms:*** *Ninja Dogs, misleading games, level generation, and game-playing agents.*

# 2.0 Introduction

## 2.1 Purpose

A crucial field of research called Procedural Level Age (PLG) focuses on the algorithmic age of game levels in computer game exploration. PLG has benefited game-playing professional development since it may be used to quickly create a large amount of prepared information. Most of the continuing learning-based techniques used by computer-based intelligence professionals demand a significant amount of information preparation in order to be effective.

Artificial intelligence (AI) researchers regularly employ video games as testbeds for their research. One such example of combining PLG and agent development is Ninja Dogs. Ninja Dogs is a physics-based puzzle game that presents AI agents with amusing difficulties when completing the levels. Due to the physics limitations in Ninja Dogs, PLG is likewise not a simple problem. To guarantee that the anticipated results may be attained while playing, any PLG algorithm should conform to the game's physical restrictions and the locations of the game objects should be established with better precision. Ninja Dogs playing agents have a continuous action space, similar to an agent in the real world. Therefore, creating levels that can only be completed by predetermined actions is highly challenging, especially when the levels are intricate and call for sophisticated reasoning.

## 2.2 Action

An aspect of a job that "tricks" an AI agent into taking undesirable actions by taking advantage of its biases or constraints is known as deception. A deceptive game level features a reward system that could sway the agent from the best course of action. Six sorts of deceptions for the current Ninja Dogs playing agents were described in a prior study on Ninja Dogs . They exposed the weaknesses of the cutting-edge agents by using hand-crafted levels for certain deception categories. There is still an opportunity for agent capabilities to be improved, as seen by the shortcomings of current agents in managing deceptions. This suggests that in order to meet the data needs of present learning systems, there must be created a sufficient number of difficult misleading levels.

Here, we outline a way for creating de-captive stages in Ninja Dogs . We take into account the six kinds of deception listed in and create level templates that make it possible for each of these categories to automatically generate a variety of levels. The solution to the level is also created and may be used by the agents in the learning process, which is another notable aspect of this technique in addition to the level itself. We establish criteria to gauge the created levels' stability, solvability, and deceitfulness in order to assess our process. Furthermore, we assess AI agents' behavior on both produced levels and handmade levels that need similar capabilities to solve in order to determine the features of the generated levels in comparison to human-created levels.

# 3.0ACKGROUND AND RELATED WORK

## 3.1 Ninja Dogs

In the 2D physics simulation game Ninja Dogs, players use slingshots to kill cats by shooting Dogs at them. The items in the game levels can move when forces are applied, including dynamic elements like cats, Dogs, bricks, and TNT bombs, as well as static ones like platforms. Dynamic objects have health points that are depleted through collisions, and when their health points are zero, they are destroyed and vanish. Blocks come in three different material varieties: ice, stone, and wood. Structures are examples of physical objects formed from different arrangements of these building components. [2]There are also five other breeds of Dogs, some of which have unique abilities and resistance to specific materials:

* Rumo dog: Has no unique abilities.
* Ninja\_Man dog: When tapped, splits into three Dogs and is effective against ice.

The level-based distribution of the Dogs cannot be altered by the player. The release point of the dog from the slingshot and the touch time to activate the power of the dog, if available, combine to form the player's action. Once all Cats have been eliminated with the assigned Dogs, a level is finished. We utilized a Ninja Dogs research clone created in Unity as the original Ninja Dogs game is not open-sourced.

*3.2 PLG in Ninja Dogs*

For procedurally creating Ninja Dogs game levels, researchers have previously employed a variety of different techniques. The yearly Ninja Dogs level generation competition [6] encourages study into creating Ninja Dogs level generators. This earlier effort mostly focuses on maintaining the integrity of the levels' physical structures, altering the levels' challenge and fun factor, and making sure the levels are solvable. None of them concentrates on creating tricky gaming puzzles that demand complex planning and thinking skills to accomplish. This effort focuses on creating game levels that AI agents cannot resolve using straightforward intuitive methods.

## 3.3 Deceptive Games

The research by Anderson et al. was when the idea of misleading games was first introduced. For the General Video Game AI (GVGAI) framework, they develop a number of deceptive arcade-style games and investigate how such games affect game-playing agents. Based on that concept, a methodology to assess agents on false levels is described in and a mechanism to construct deceptive levels for games in the GVGAI platform is discussed.

The study on deceptive Ninja Dogs levels by Stephenson and Renz identifies six potential sorts of deceptions that can fool or abuse the present state-of-the-art Ninja Dogs playing agents in the setting of sophisticated physics simulation games, as already indicated in Section I. While not all Ninja Dogs agents are equally affected by various deceptions, no agent was able to handle every type of deceit. In our generating process, we take into account the same six deception areas. The six kinds of deception are outlined here, and Fig. 1 provides instances for each category.

## 3.4 Process the game

**1) Rolling/falling articles:** This misdirection utilizes the way that objects of one substance can fall or be moved on to one more element to make an effect. A specialist needs to comprehend that an item can fall or be rolled and that article can be utilized to hit an objective.

**2) Making ways:** This stunt is utilized when there are hindrances that should be wiped out or eliminated to get to an objective. A specialist should understand that to arrive at an objective, it should initially clear the street there to adapt to this double-dealing.

Substance strength investigation is expected for this misdirection, which requires a specialist to assess the level's elements' assets. The material, structure, and size of an article all influence areas of strength for how is. To finish the level, a specialist should have the option to perceive the actual constraints and qualities of different animals and collaborate with them fittingly.

**4) Non-ravenous activities:** In this stunt, an activity that initially appears to be less worthwhile than another choice will wind up helping more over the long haul. Rather than acting in a ravenous manner that expands the transient result, the specialist ought to prepare and utilize how it might interpret the environmental elements.

**5) Non-fixed tap time:** The Canines ' extraordinary capacities are opened by tapping them while they are flying. In this trick, a specialist should utilize the Canines ' extraordinary capacities at irregular stretches as opposed to tapping at a certain, foreordained second.

**6) Dynamite:** When a player hits a stick of dynamite dangerous in the game, it detonates, harming or moving neighboring things. In this trick, either a specialist should utilize Explosives to kill felines, or Explosives are utilized to redirect the specialist from the killing of felines, contingent upon the circumstance.

## 3.5 TERMINOLOGY

In this work, the term "environment" refers to the non-particular area where an agent can detect and act.

Offline Stage

Level Template Matching Module

Candidate Deceptive Level and Solution

Level Validation Module

Repository of Physical Entities and Metadata

Collection of Self-Contained Physical Entities

Physical Entity Extraction Modul

Physical Entity Analyzer

Module

Content from Existing Generation

Fig. 2: Deceptive level generation methodology.

A physical entity is a standalone, self-contained collection of physically stable items in the environment. Deception is seen as a trait of a task that takes advantage of an agent's cognitive biases and influences its decision-making to be less than ideal [3].A plan of action for an agent that involves interacting with the environment (e.g., dog release timings and tap intervals) is called a strategy. A solution strategy is a particular approach that, when used, clears a specified level. An agent uses a technique when trying to solve a level, to sum up (e.g. shooting Dogs targeting at Cats). There may be several different approaches that, depending on the method, might either complete the level or not.

# 4.0WORKING

Class diagrams show the system's structure as a hierarchy of classes and objects. A class diagram is a form of a static structure diagram that demonstrates the system's classes, their properties, actions, and connections among objects to illustrate the structure of the system. Classes are shown in the diagram as boxes with three sections each: The class name is located in the top compartment. The initial letter is capitalized, it is printed in bold, and it is centered. The class property is located in the center compartment, which is positioned to the left. The class's operations are located in the bottom compartment. They also align to the left. The fact that there may be many more configurations of a software product than there are problems in it presents a challenge for software testing. Testing makes it challenging to uncover bugs that don't happen very often. A general rule of thumb is that a system must have been tested for at least the amount of time that it will be expected to operate faultlessly. This has negative effects for initiatives to create durable, dependable software. Associations stand for a network of connections. Typically, a Line is used to depict a binary relationship.

## 4.1 PROPOSED PROCEDURE

We outline the suggested Ninja Dogs misleading level generating process in this section. The major elements of the generating process are depicted in Fig. 2. The four components in red correspond to the inputs and outputs of the four components in blue, which are represented as the procedure's modules. The generating process removes physical things made by already-existing content generators rather than constructing them from scratch. This gives users access to a variety of information that was created using different techniques. Customized handcrafted material may be utilized in a similar way to utilizing current content generators. There are two steps to the generating process: an offline stage and an online stage. The first row's grey box depicts the offline stage, and the second row's purple box depicts the online stage (second row).

## 4.2 Module for Physical Entity Extraction

This module retrieves physical entities from already created game instances from Ninja Dogs content generators. There are two types of game instances: full game levels and game levels with physical things. We create a series of game instances using a variety of Ninja Dogs level generators that are already available [6]. A qualitative reasoning procedure that iteratively extracts entities copies the entities from these examples. The physical entity is displayed in Algorithm 1.

**Algorithm 1 Extract physical entities**

Input: Game instance with physical entities

Output: Set of extracted physical entities

entities extracted = {}

objects remaining = all objects in the instance

while objects remaining is not empty do

topObject = topmost object of objects remaining

entityExtracting = topObject and supporters of topObject

while True do

boundingBox = bounding box of entity Extracting

entityExtracting = objects inside boundingBox

if boundingBox size not increased then

Break

add entity Extracting to entities extracted

remove objects of entityExtracting from objects remaining

return entities extracted

a method of extraction that may be used with a game instance that has several physical things. We use the support graph to define the word "supporter" when developing the method, much like the authors did in [19]. The support graph has an edge pointing from object Oi to object Oj if the bottom horizontal edge of one item Oi is in touch with the top horizontal edge of another object Oj (i.e., Oi is resting on top of Oj). If there is a path from object Oi to object Oj in a particular support graph, then object Oj is regarded as a supporter of the object Oi (Oj supports Oi). The support graph for objects put on the ground is empty since there are no supporters.

## 4.2 Module for Physical Entity Analysis

After each physical entity has been retrieved, it is studied using the Physical Entity Analyzed Module by interacting with it and seeing the results. The only way to interact with the animals in Ninja Dogs is by launching Dogs from a slingshot and tapping to activate the dog's unique skill. To predict the result of physical interactions, we can use qualitative methods, which are often faster but less precise, or simulation-based approaches, which are frequently more accurate but slower. The qualitative physical predictions made by Ninja Dogs have shown to be less accurate and robust for really complex items. As a consequence, we perform interactions and document the outcomes using an approach based on simulation. [4]

Using four different methods from past AI agents that could play Ninja Dogs, we developed a portfolio agent that can play the game. The first involves shooting Cats in the entity with Dogs without special abilities, TNT in the entity with Dogs without special abilities, reachable blocks in the entity with Dogs without special abilities, and Cats in the entity with Dogs with special abilities (activated at various times) in the fourth. The numerous iterations (i.e., tactics) of the aforementioned approaches depend on the order of the targets (if there are many targets), the firing angle, and the length of the dog's special power's activation time. [5]

• Information about the entity that existed before the contact, such as how many items are included within it and how big its bounding box is.

Info about encounters (e.g. the magnitude and the location of the force applied on the entity due to the interaction).

• Dynamic data when the objects in the entity are moving as a result of the interaction, such as the location and velocity of objects that have left the entity's original bounding box.

• Entity data after the interaction, after all the objects have stopped moving (such as the number of items in the entity and the size of the entity's bounding box).

The data obtained through this analysis is added to the entity as metadata and stored in a repository of entities with metadata. The offline portion of the producing process is complete; the remaining stages take place online.

To start the online step of the physical deceptive level generating process, a generation request for the relevant deception category is delivered to the Level Template Matching Module. This module creates a candidate level for that deception category using pre-defined level templates. Entity requirements and level construction rules are included in a level template for deception. When building a level, the level template considers conceivable interactions that an agent could engage in as well as the outcomes of those interactions as inferred from the metadata of the entities. According to the generation rules, the level is built using entities that abide by the limits of the template.

**Algorithm 2 Rolling/falling objects**

Input: Sender entity, Receiver entity  
Output: A level with rolling/falling objects deception

if sender has suitable objects that can be rolled/fallen then

if receiver has an OSS then

while generation unsuccessful do

get sender’s rolling/falling object’s trajectory

get OSS of receiver

generate level by matching sender and receiver

verify reachability of targets

if generation successful then

break

if maximum generation attempts reached then

return none

allocate Dogs to level

generate solution strategy

return generated level

Level Template Matching Module uses the interaction data (dog shots) contained in the entity's metadata to produce levels while also producing the level's solution. The interaction data necessary to finish the level is in the solution (i.e., the solution strategy of the level). We try to make the trick level solvable exclusively with the generated strategy while creating the level templates. If an agent applies the solution technique to complete the level, we assume that the agent was aware of the trick. By only allowing the created answer, the agents are prevented from completing the levels using alternative strategies without recognizing the trick. [5]

## 4.3 Level Validation Module

The final module of the generation process verifies the created levels' physical stability and solvability. [5] All of the items in Ninja Dogs should be immobile at the beginning of the level. Using the Box2D physics engine, the Level Validation Module first confirms the stability of the level. After replicating the level for two seconds, the velocities of the items are measured to assess whether the level is stable. The Level Validation Module first determines whether the level is stable, and then uses its created solution method to determine whether the level is solvable. This approach results in a consistent misleading level and its proven solution as its end product.

# 5 UML Model

## 5.1 Use Case Diagram

PHYSICS LAWS AND PROJECTILE (TRAJECTORY) MOTION

Store the played game

scene, the shot, and

the score achieved

Compare the score achieved

Yes

with the score corresponding

to the most similar scene

Similar game

scene found?

Search for the most similar

game scene to the played one

Play a random shot and

observe the score

Observe a game

Start

Achieved score >

score of the most

similar scene?

Fig: 2-D AND 3-D DIFFRENCES

Replace the shot and the score

of the most similar scene with

those of the new one

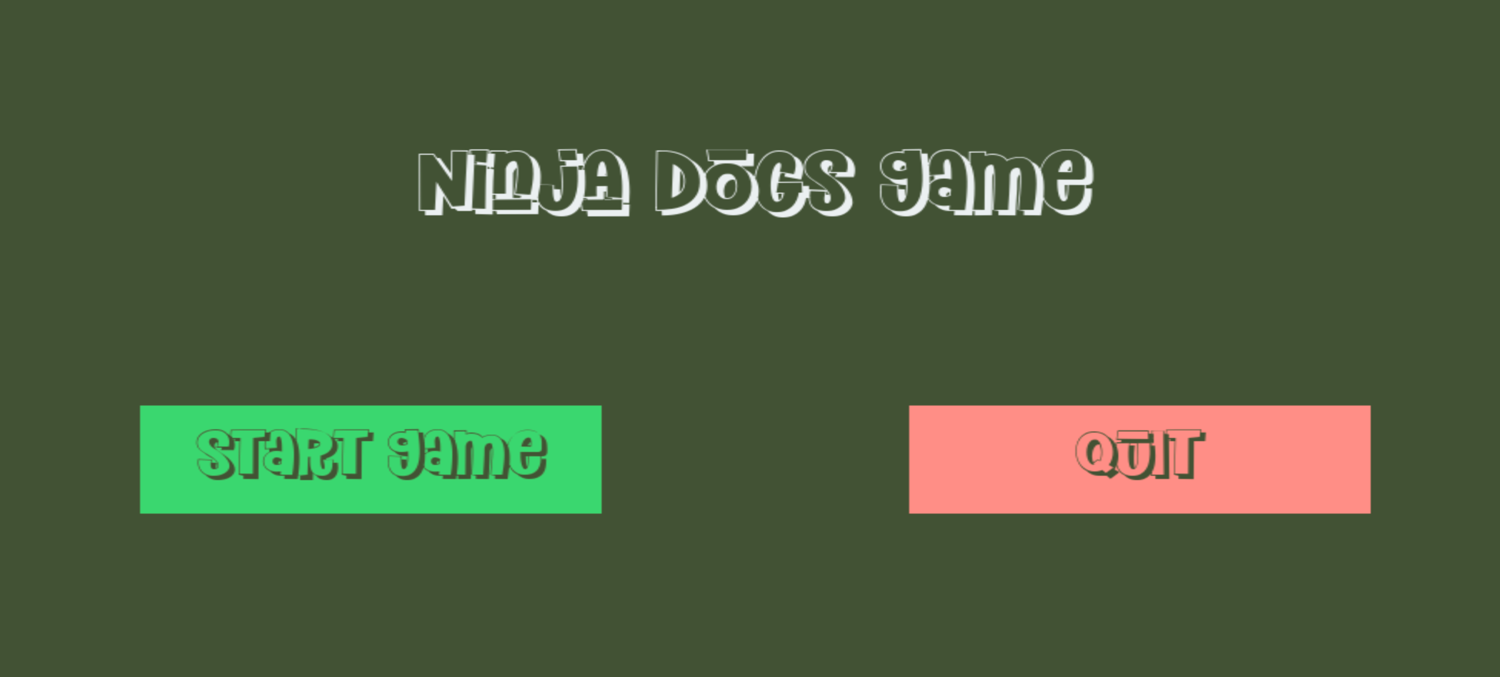
Fig. 2. The steps of the case base construction process.

## 5.2 Diagram



# 6. Game Screen

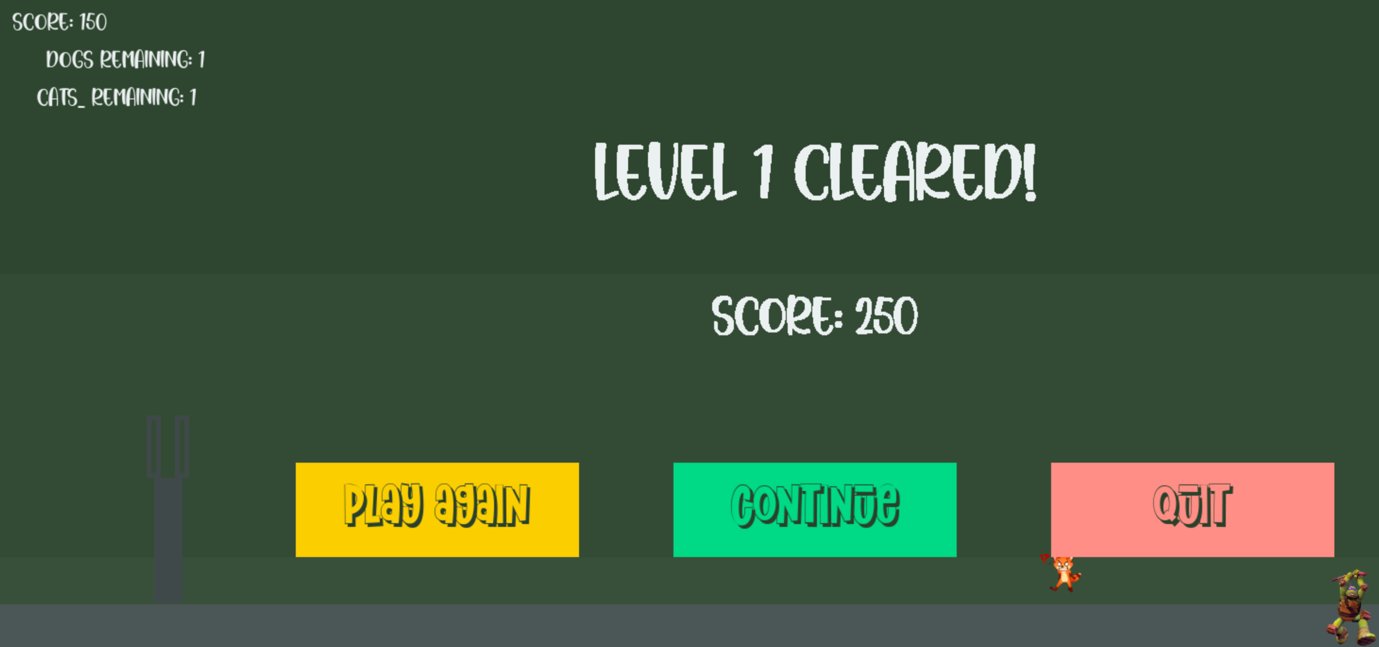
**1**

****

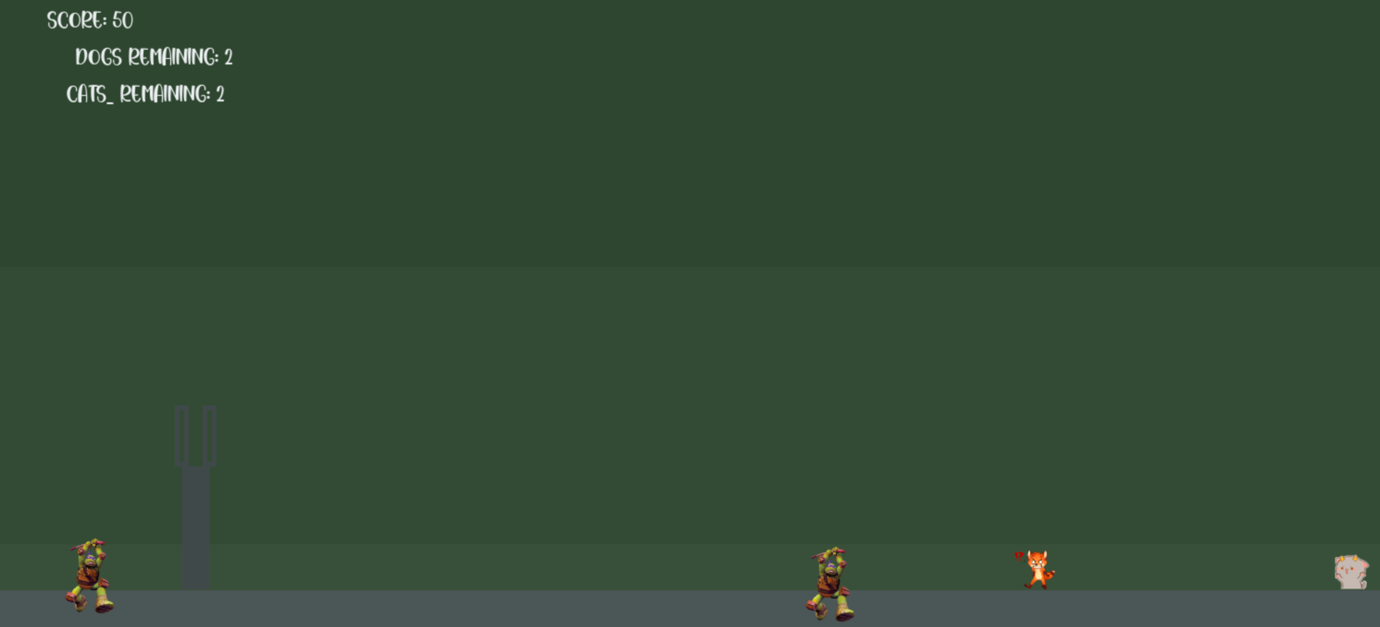
**2**

****

**3**

****

**4**

****

**5**

****

# 7. Conclusions and future work

We have discussed how to make fake levels for the Ninja Dogs game in this essay. The suggested approach may be used to develop the six deception categories that the cutting-edge Ninja Dogs playing agents are sensitive to. Although the idea of manually making them has been considered in the past, this is the first attempt at building false levels for a difficult physics-based game like Ninja Dogs. Our approach also generates solutions to the levels, which can be helpful to learning agents as they practice, unlike any other Ninja Dogs level generators now in use.

This endeavor aims to allow the building of advanced Ninja Dogs playing agents that can function well under deceptions by supplying enough training/testing data. To overcome these tricks, the AI techniques used by the current agents can be improved to expand their ability for generalization, planning, and reasoning. In order to develop levels that are more challenging, future level-generating research may concentrate on combining several deception categories. More deception categories could potentially be proposed if more flaws in the state-of-the-art agents are discovered. Rolling/falling objects, clearing paths, entity strength analysis, and non-greedy activities are some of the categories of deception that we took into account. These phenomena may also be observed in a physical environment. Therefore, this study might serve as the basis for developing deceptive activities in a physical environment.

# Bibliography

|  |  |
| --- | --- |
| [1] | A. Khan, “Development of Angry Bird Game in Jupyter Notebook Python,” www.researchgate.net, Bannu, October 2020. |
| [2] | I. 6. G. A. E. A. Consumer Relations - Mattel, "SERVICE.MATTEL," Instruction Sheets, 2011. [Online]. Available: https://service.mattel.com/instruction\_sheets/W3969-0920.pdf. |
| [3] | A. Sweigart, "Making Games with Python & Pygame," United States License, 2012. |
| [4] | J. R. PaRkeR, "Game Development UsinG python," mercUry learninG anD information, New Delhi, 2019. |
| [5] | utorial Team Member Julian Meyer, "Beginning Game Programming for Teens with Python," Raywenderlich, 2022. [Online]. Available: https://www.raywenderlich.com/2795-beginning-game-programming-for-teens-with-python. |
| [6] | pygame\_org, "pygame in all the languages," pygame, 16 May 2022. [Online]. Available: https://www.pygame.org/news. |
| [7] | S. Runtime, "Validation Module," 2014. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/validation-module. |