
The background features stylized tree branches in black, orange, and light beige. Small black birds are perched on various branches. The title text is centered in the upper half of the image.

# Reinforcement Learning in Gaming

The background features a solid tan color. On the left side, there are several thick, black, stylized tree branches. A small white bird is perched on one of these branches near the top left. Another white bird is perched on a lower branch further down. In the center, there is a single, thin, white vertical branch. A white bird is perched on a horizontal branch that extends from this central white branch towards the right. On the right side, there are more thick, black, stylized tree branches. A white bird is perched on one of these branches near the top right.

# Article 1: Applications of Reinforcement Learning in Mobile Games

# Intro

- To create a more engaging experience and entertainment, mobile games developers implement artificial intelligence tools in their works. The main idea is to create an algorithm of actions and opportunities which will make a game more absorbing and satisfying but do not outsmart players. There has to be space for imperfections, but more importantly — new challenges to tackle.
- Using the reinforcement learning method, mobile apps development companies create games where the player will be rewarded for the right behaviour and punished for the wrong one. The agent will be doing their best to take fewer wrong steps and make more right decisions. In addition to that, a player looks for a lasting and most optimal award.



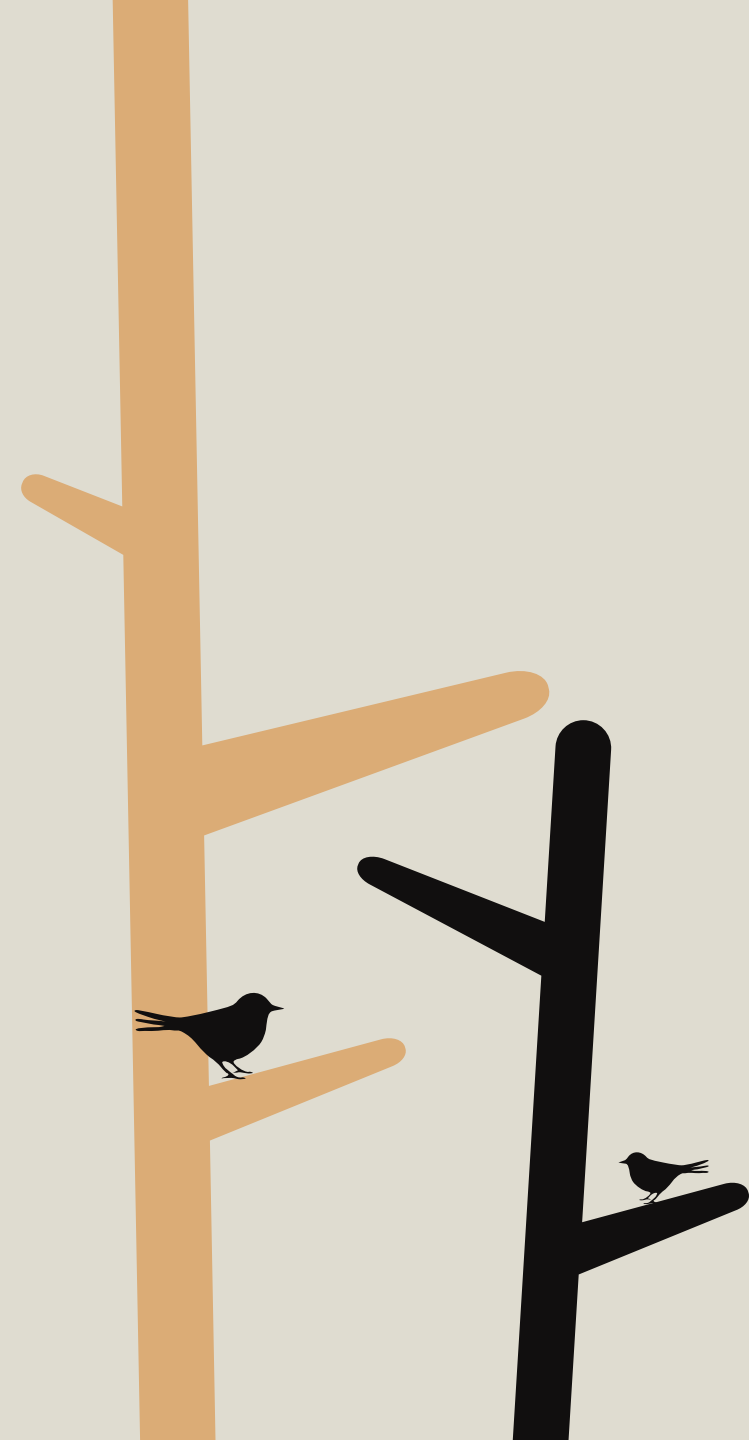
# Examples of Reinforcement Learning Method in Mobile Games

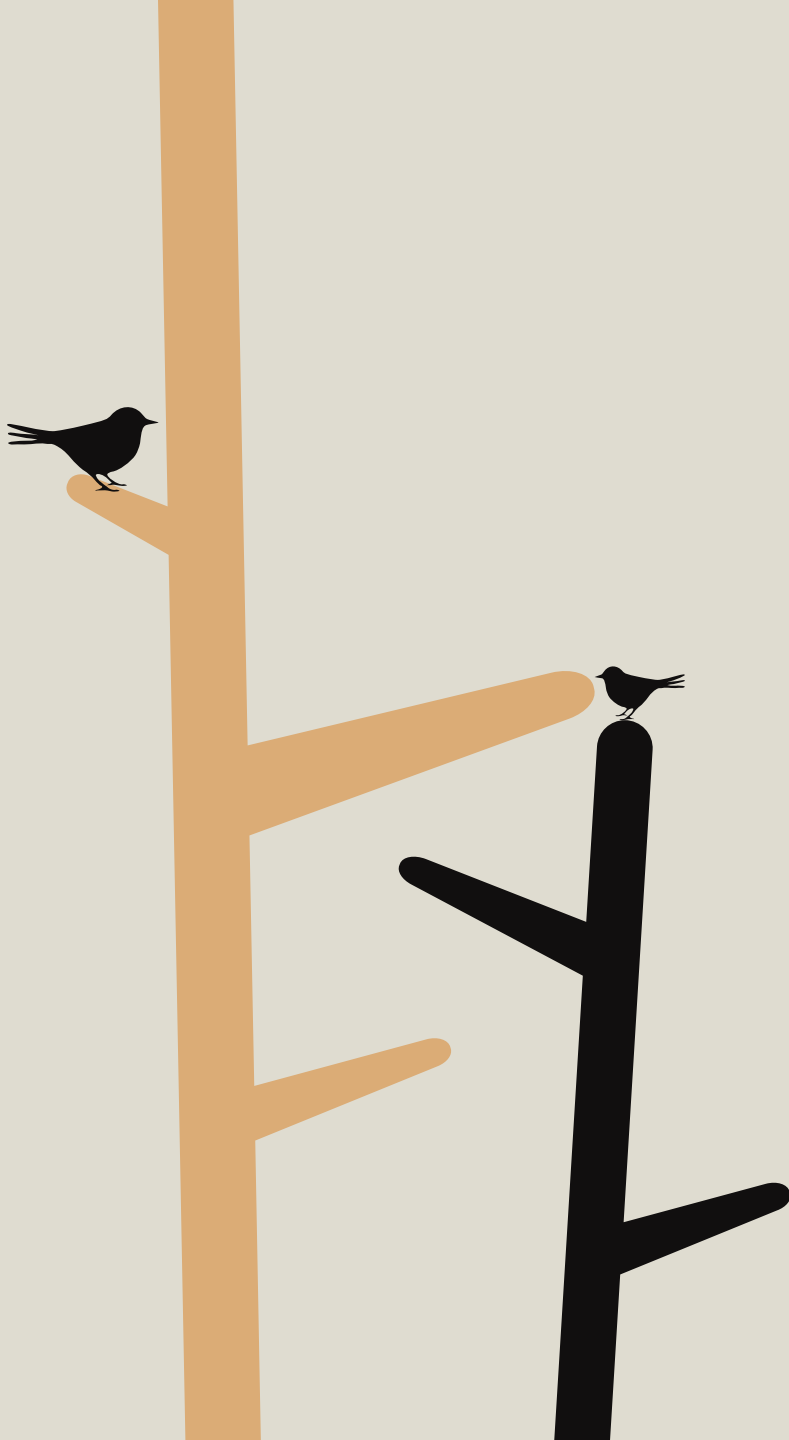
- As mentioned above, machine learning algorithms are successfully applied for training in modern mobile games. Due to the slow process and high resource cost, the most efficient algorithm would be the one that can adjust well to multiple environments.
- Q-learning type of reinforcement learning is a perfect match, as it is necessary to learn to discredit action spaces with near-zero tolerance for errors. That is why value-based methods like q-learning fit better than stochastic algorithms.
- The following slides illustrate the implementation of reinforcement learning to mobile video games such as Flappy Bird and Subway Surfers.



# Flappy Bird

- This is an arcade-style game where a player controls the bird Faby. It moves continuously to the right and goes through sets of pipes. The tubes have an even size and breaks are placed randomly. A player has to touch the screen to make a bird soar. Otherwise, it automatically goes down.
- The successful pass through the pair of pipes is awarded in the form of one point. In case Faby collides with the pipe or ground, the game is over. In general, a player can win a prize of a bronze medal for reaching ten or more points, a silver one from twenty points, and a gold medal from thirty points. The highest award is a platinum medal, which could be achieved by earning forty points.

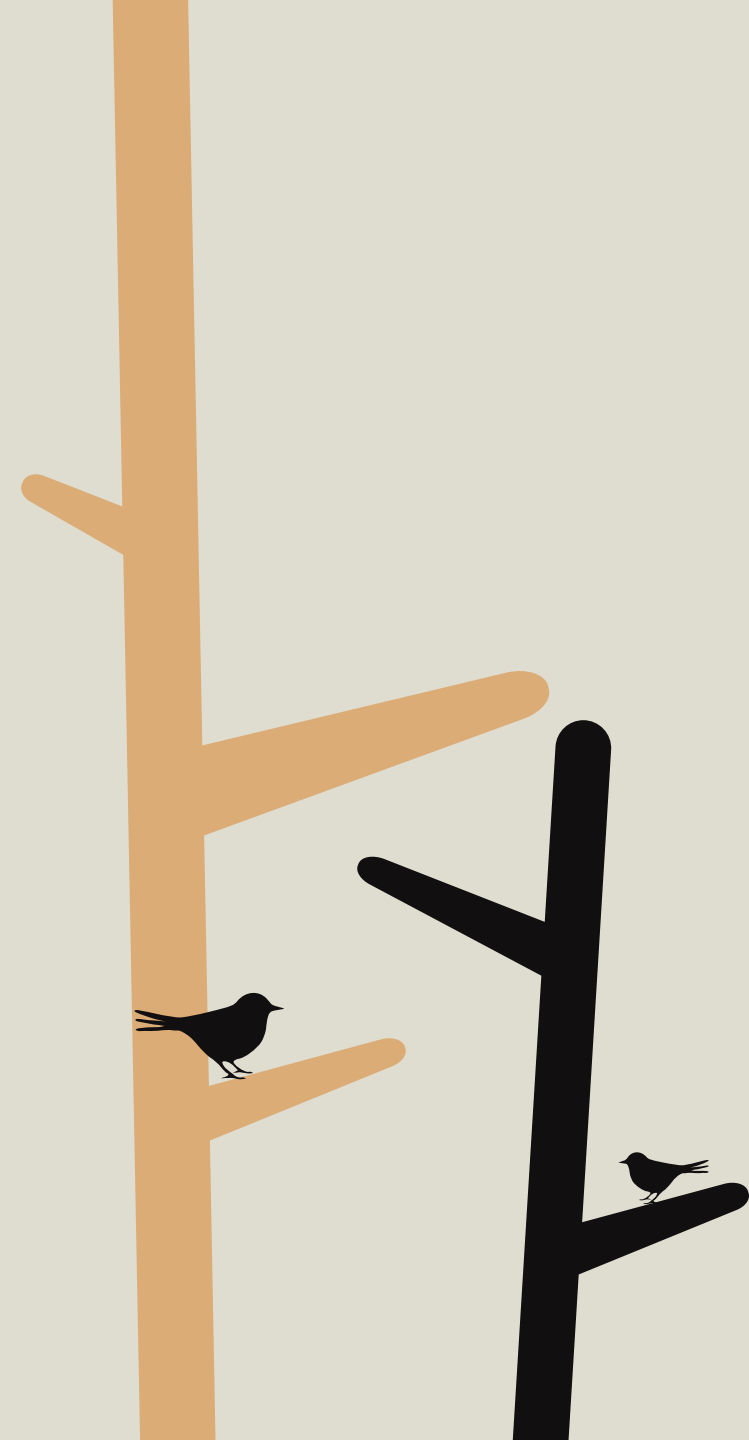


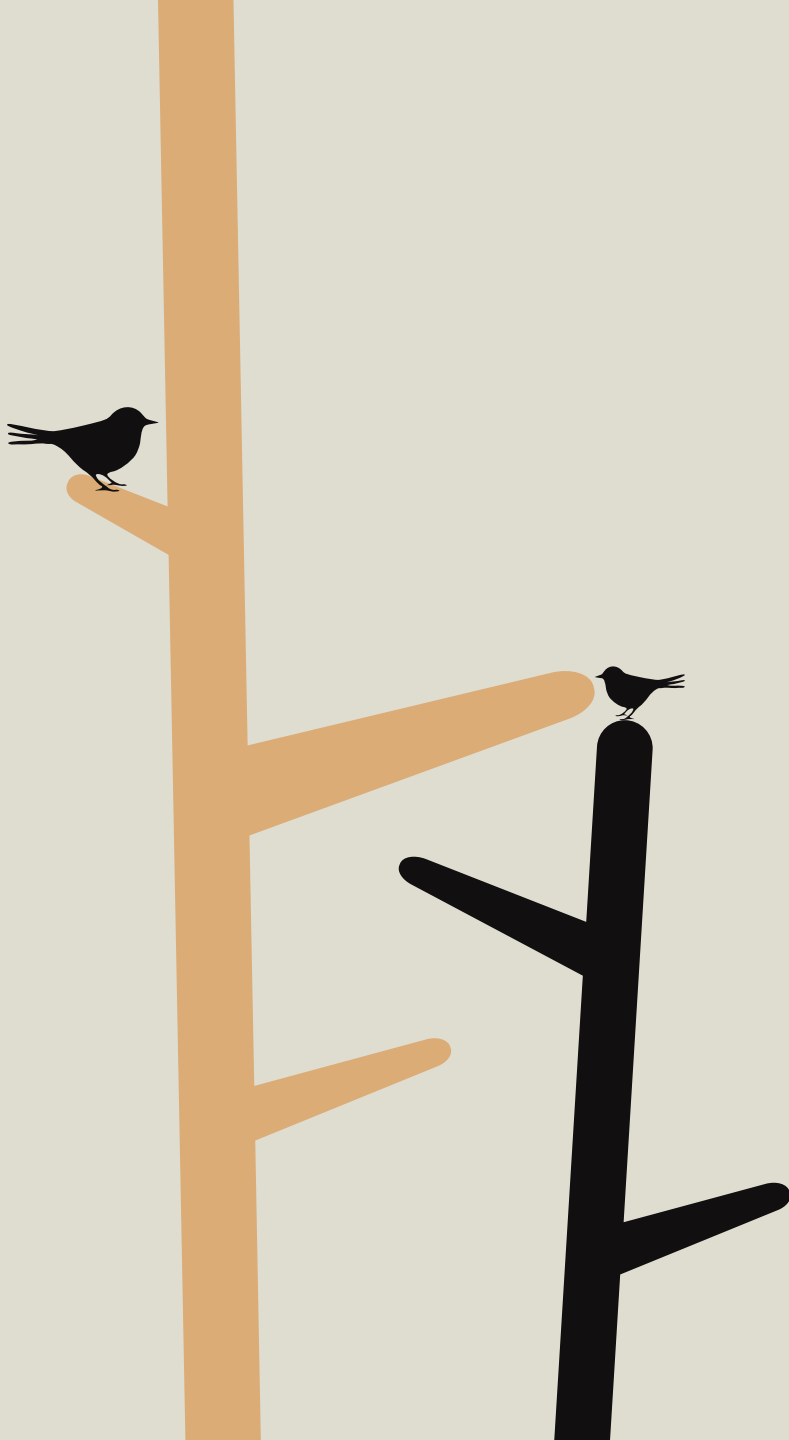


- To get the initial reward, the agent has to pass the first pipe. This action can be tricky for an inexperienced player using random exploration. However, its simple and clear visuals make the overall policy easy to learn.
- There could be numerous initial steps made without any rewards afterward. This means that the agent gets no feedback and consequently no way to learn. After experiences of passing the first pipe, the agent can finally learn how to improve its tactics.
- Reinforcement learning aims to maximize the expected value of the total payoff or the expected return. It was used the Q-learning approach, in which a neural network is used to approximate the Q-function. According to the test, Flappy Bird was trained at 30FPS with a frame-skip of 2 (15 Steps-Per-Second) for a total of 25M steps. This takes around 40 hours to train using 12 emulators. Evaluation Score: Average 420 points, Max 1363 (10 Eval Episodes).

# Subway Surfers

- Subway Surfers is a runner mobile game where a player controls a graffiti artist who creates a piece of art on a railway wall and eventually has to run away through the railroad tracks and tries to escape from the inspector and his dog. The aim of the game is not to get caught in a metro railway and grab as many coins as it's possible.
- A player has to swipe a screen in different directions to dodge objects along the way. Certain gestures such as swiping rapidly serve as a specific option to increase the running speed. The faster the player runs, the more coins can be collected. Moreover, among the coins, a runner can pick up bonus items.






- In this case, coin collection count is not an integral part of growth, but it serves as a necessary trigger for the learning process. While adding a negative reward which equals -1 coin, the agent is motivated to play without failures. By adding a small negative reward on each action taken, the agent won't misuse bonus items. During the training process, the game is trained at 30FPS with a frame-skip of 4 (7.5 steps-per-second), for 25M steps. This takes around 90 hours to train using 12 emulators. The action space varies in 4 actions (swipe up/down/left/right) and the "noop action". Evaluation Score: Average 142, max 434 (30 Eval Episodes).



# More Results to Come

- Reinforcement learning is still being improved and is widely applied in different areas. Significant progress has been made to improve the field of the mobile gaming industry to enhance players' engagement and gaming experience overall.
- Nevertheless, the outcome cannot be described as surpassing human abilities. With continuous training, a reasonable amount of time, frames, and resources, the reinforcement learning algorithm can enhance modern mobile games in real-time.



The background features a solid tan color. On the left, there are thick, dark brown tree branches. A small white bird is perched on one of these branches near the top left. Another white bird is perched on a thinner, light-colored branch that extends from the left towards the center. On the right, there is another dark brown tree branch with a small white bird perched on it. The text is centered in the middle of the image.

# Article 2: Reinforcement learning improves game testing

# Intro

- As game worlds grow more vast and complex, making sure they are playable and bug-free is becoming increasingly difficult for developers. And gaming companies are looking for new tools, including artificial intelligence, to help overcome the mounting challenge of testing their products.
- A new paper by a group of AI researchers at Electronic Arts shows that deep reinforcement learning agents can help test games and make sure they are balanced and solvable.



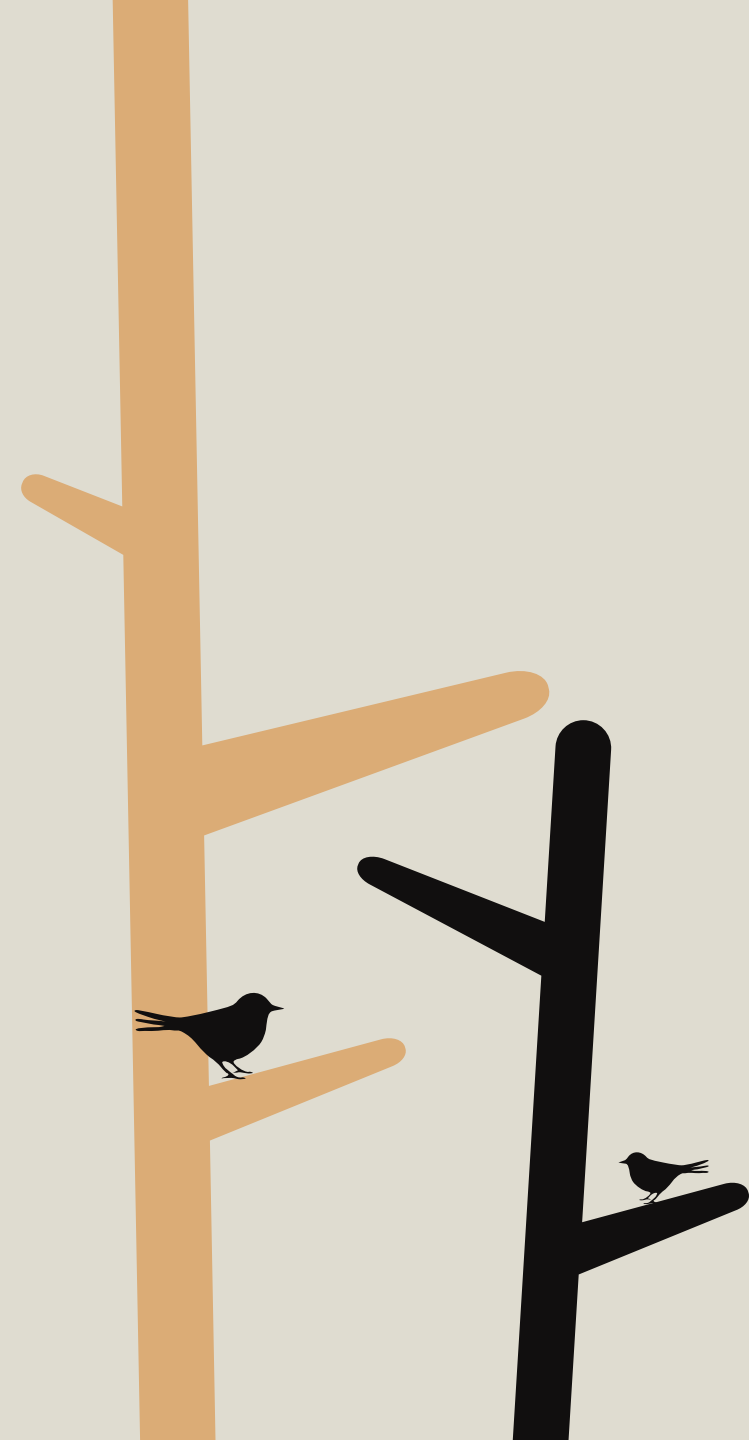
# Testing large game environments

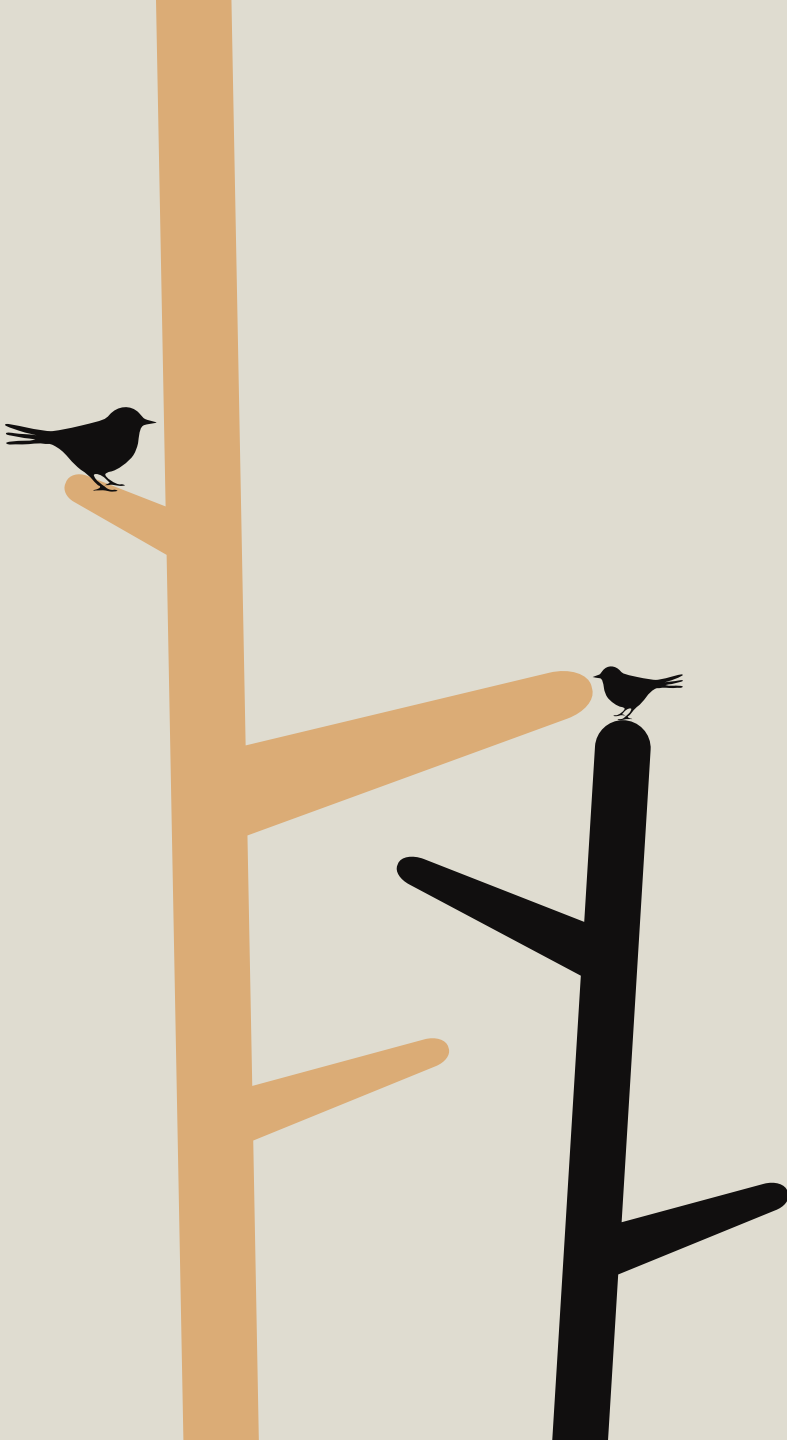
- Developers have currently two main tools at their disposal to test their games: scripted bots and human play-testers. Human play-testers are very good at finding bugs. But they can be slowed down immensely when dealing with vast environments. They can also get bored and distracted, especially in a very big game world. Scripted bots, on the other hand, are fast and scalable. But they can't match the complexity of human testers and they perform poorly in large environments such as open-world games, where mindless exploration isn't necessarily a successful strategy.
- The goal is to use reinforcement learning (RL) as a method to merge the advantages of humans (self-learning, adaptive, and curious) with scripted bots (fast, cheap and scalable).



# Testing game content with adversarial reinforcement learning

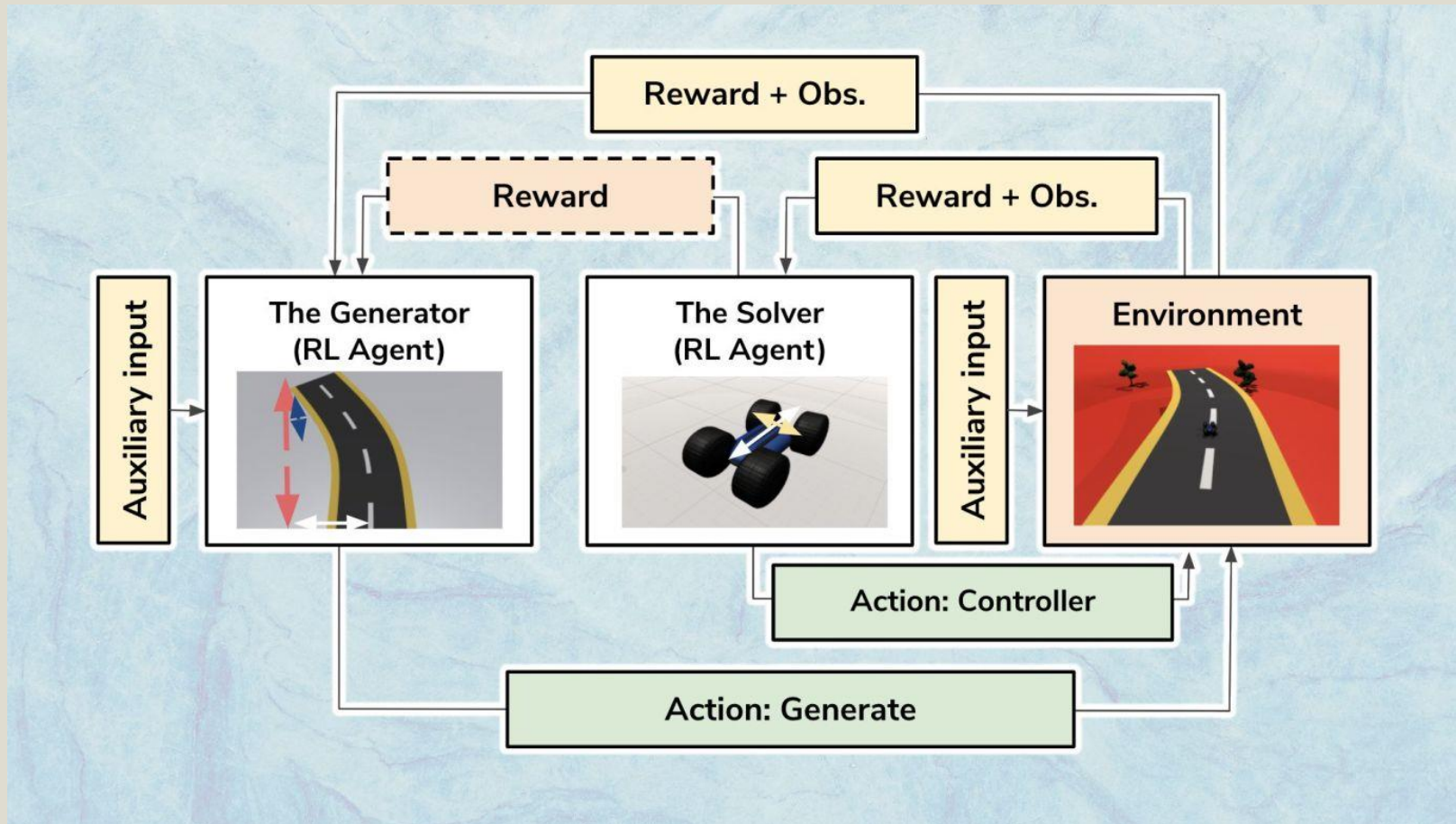
- More recently, gaming companies have become interested in using reinforcement learning and other machine learning techniques in the game development lifecycle.
- For example, in game-testing, an RL agent can be trained to learn a game by letting it play on existing content (maps, levels, etc.). Once the agent masters the game, it can help find bugs in new maps. The problem with this approach is that the RL system often ends up overfitting on the maps it has seen during training. This means that it will become very good at exploring those maps but terrible at testing new ones.

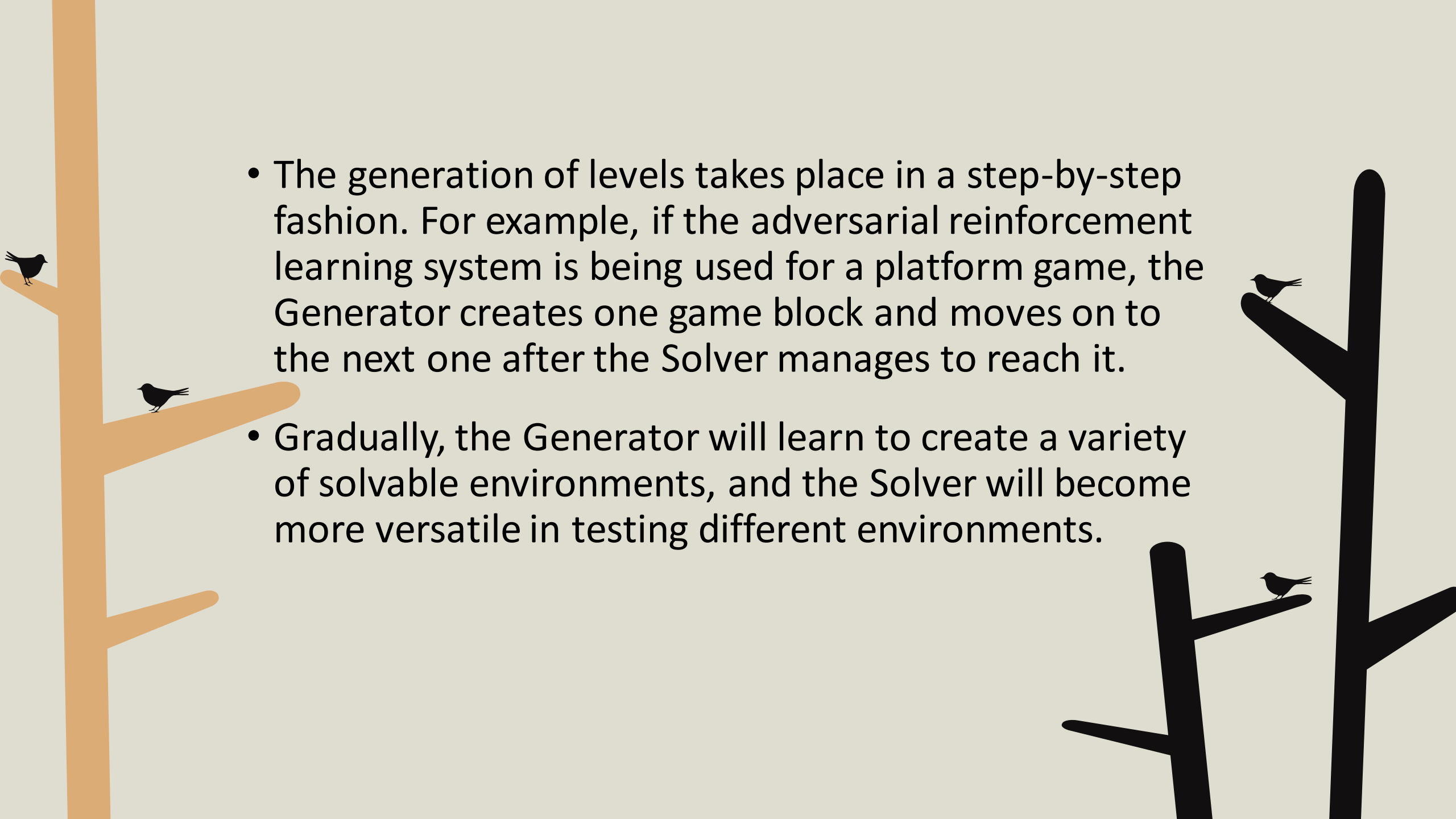




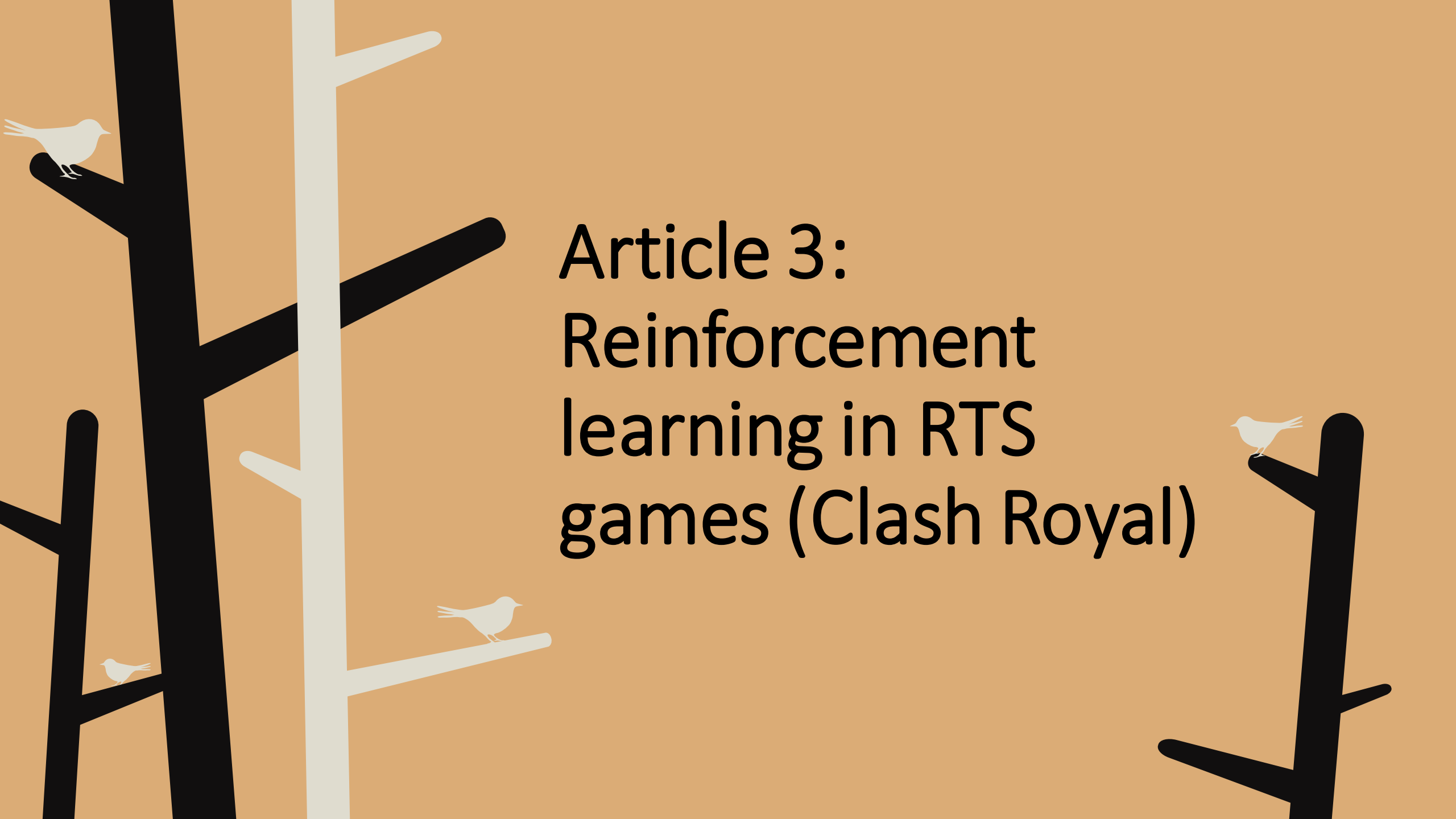
- The technique proposed by the EA researchers overcomes these limits with “adversarial reinforcement learning,” a technique inspired by generative adversarial networks (GAN), a type of deep learning architecture that pits two neural networks against each other to create and detect synthetic data.
- In adversarial reinforcement learning, two RL agents compete and collaborate to create and test game content. The first agent, the *Generator*, uses procedural content generation (PCG), a technique that automatically generates maps and other game elements. The second agent, the *Solver*, tries to finish the levels the Generator creates.

There is a symbiosis between the two agents. The Solver is rewarded by taking actions that help it pass the generated levels. The Generator, on the other hand, is rewarded for creating levels that are challenging but not impossible to finish for the Solver. The feedback that the two agents provide each other enables them to become better at their respective tasks as the training progresses.



- 
- The slide features a light beige background. On the left side, there is a stylized tree branch in a light orange color. Two small black birds are perched on this branch: one near the top and one further down. On the right side, there is a darker, black stylized tree branch. Two more small black birds are perched on this branch: one near the top and one further down. The text is centered in the middle of the slide.
- The generation of levels takes place in a step-by-step fashion. For example, if the adversarial reinforcement learning system is being used for a platform game, the Generator creates one game block and moves on to the next one after the Solver manages to reach it.
  - Gradually, the Generator will learn to create a variety of solvable environments, and the Solver will become more versatile in testing different environments.



The background features a solid tan color. On the left, a large, dark brown tree branch structure extends vertically, with a small white bird perched on a horizontal branch near the top. Another dark brown branch extends from the bottom left towards the center. On the right, a dark brown tree branch structure extends vertically, with a small white bird perched on a horizontal branch near the top. A light beige branch extends from the center towards the bottom right, with a small white bird perched on it. The text is centered in the middle of the image.

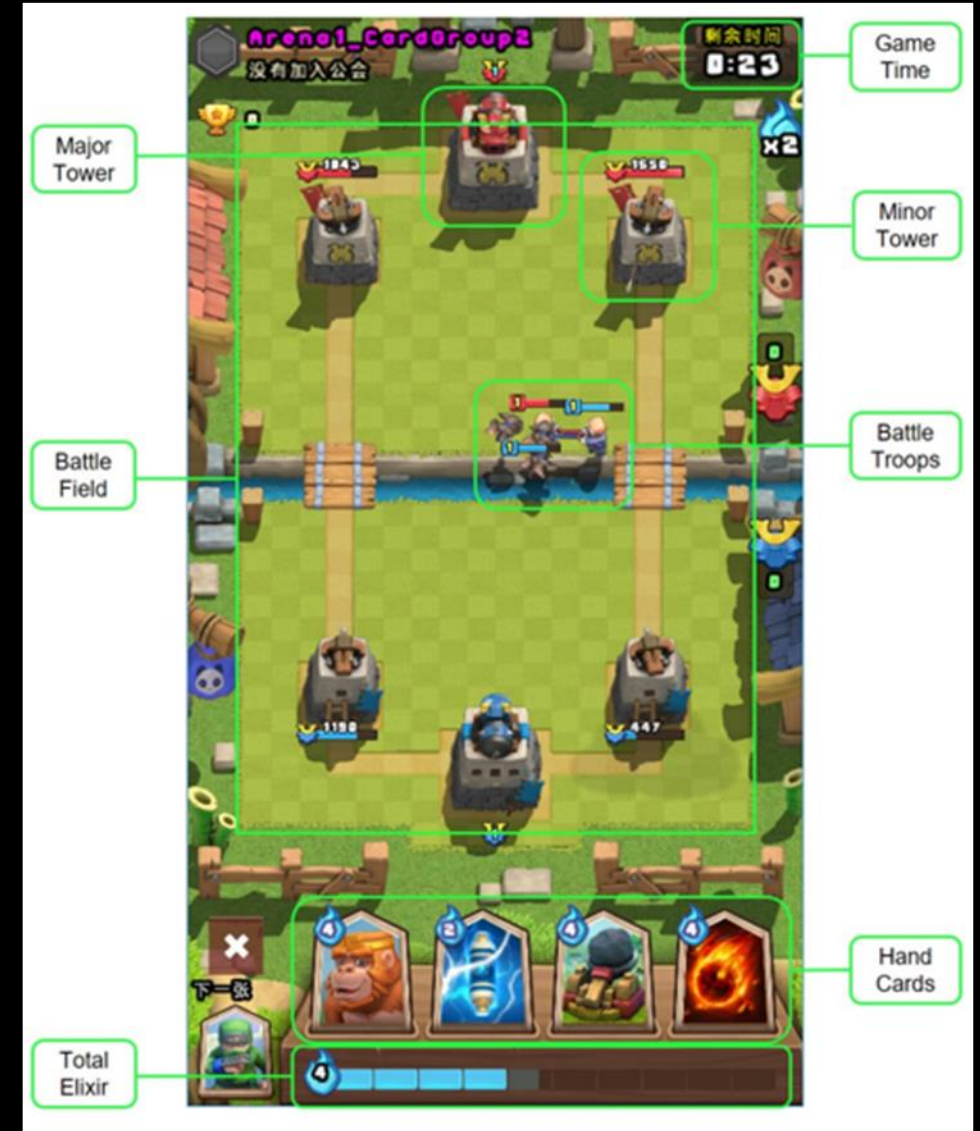
# Article 3: Reinforcement learning in RTS games (Clash Royal)

# Real-time strategy games

- Real-time strategy games have the characteristics of huge state space, imperfect information, sparse rewards and various strategies, which make it difficult to design intelligent agents for playing this kind of games.
- Card games RTS are that type of RTS games that all decision from a round is taken only by your cards from your hand and is harder than a normal card game. An example of that type of game is Clash Royals. That types of games are hard, because every single card can produce a new game state, and the same time, the enemy can use a card that produce another state space, so the space will be huge.

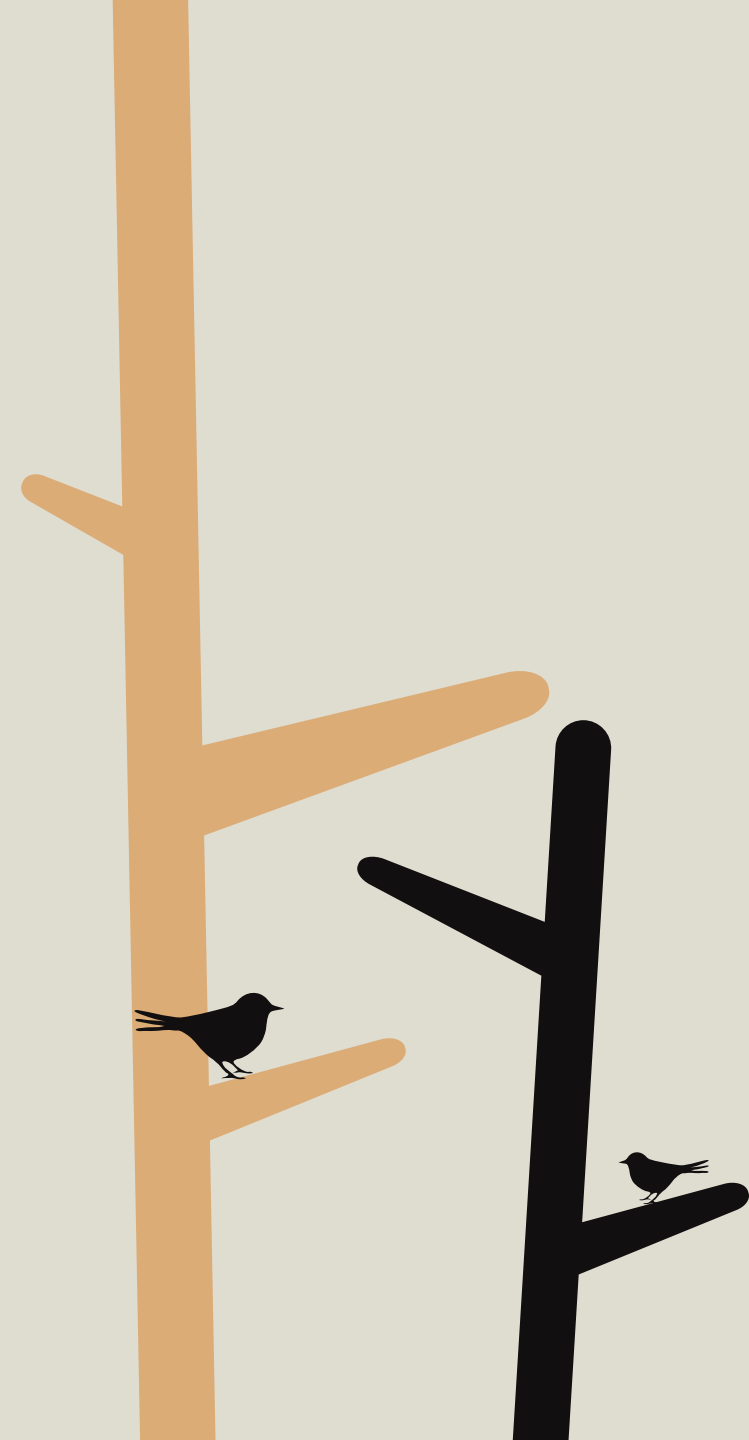


- The rewards during playing are hard to define because the game results can only be known after battles end. Strategies like offense, defense, rush and flexibility make the games with great uncertainty.
- For RTS games is used DRL, Deep reinforcement learning, that combine RL and deep learning to reduce state space and to have more intelligent agents. In that type of RL functions of decisions or policy  $\text{Policy}(\text{action} \mid \text{state})$  will be neural networks. That helps us to take decisions from a bigger space of input, faster and smarter.



# Selection-Attention Model

- In this article will speak about SEAT (selection-attention model) that has 2 parts, one for card selection and one for card choosing. The selection part transforms battle field states into state images, and makes decisions on choosing which hand card to use. The attention part processes enemy states by trained convolutional layers, and makes decisions on where to use the selected card.
- The game is composed in big part by troops and towers (those have 4 states), I will present next the states. The states are: Position (x,y), with  $1 \leq x \leq 18000$  and  $1 \leq y \leq 32000$ , position represent the position of a tower or of a troop ( I will name both of them unit, one static, tower and one dynamic, troop). Health, that differ from a unit to another one, but in this study has normalized to be 0 -> 10, from death to max life. Class that represent type of unit, every class has specials attributes, attacks, damage area and the last one Faction ( 0 or 1 that represent if the troops is yours or no).



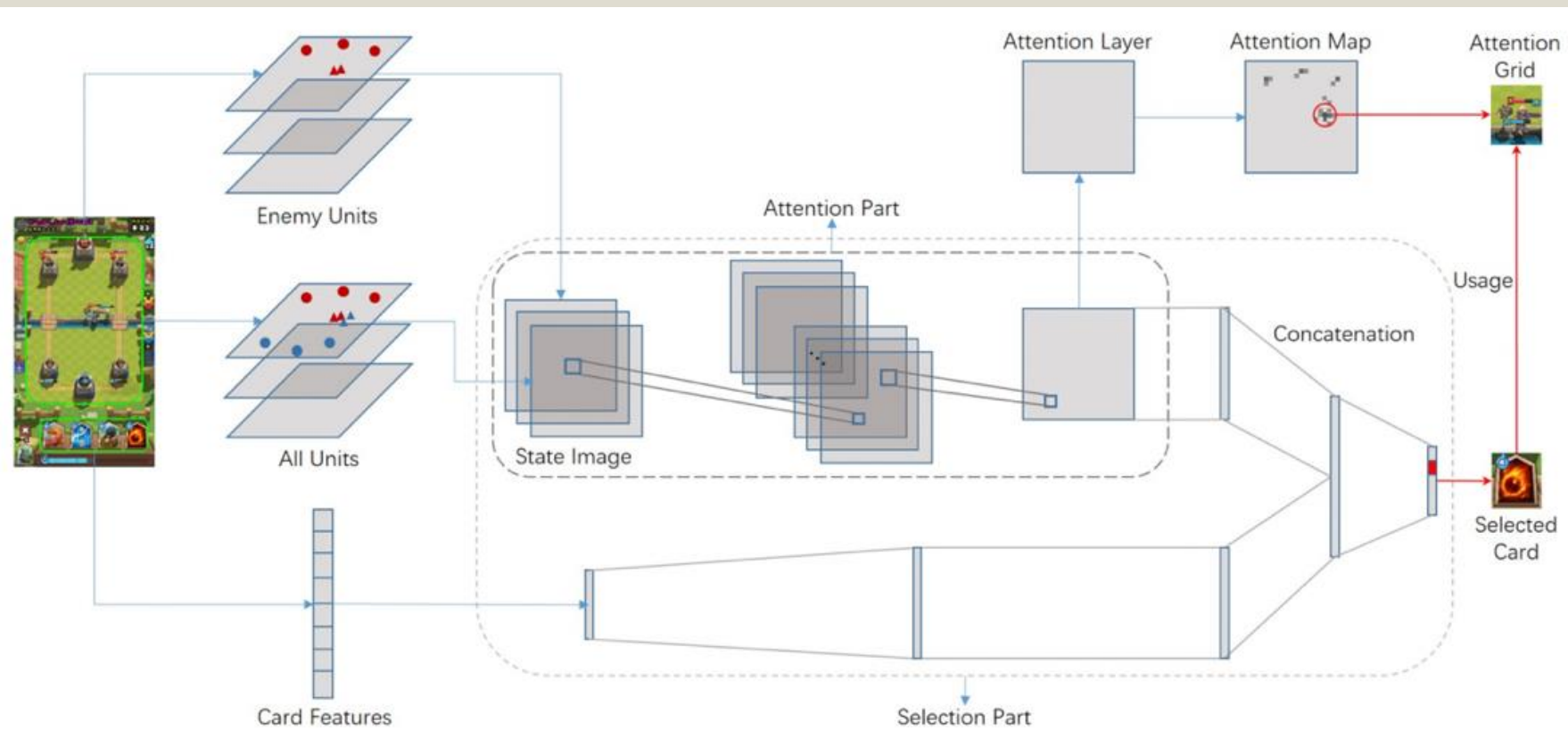
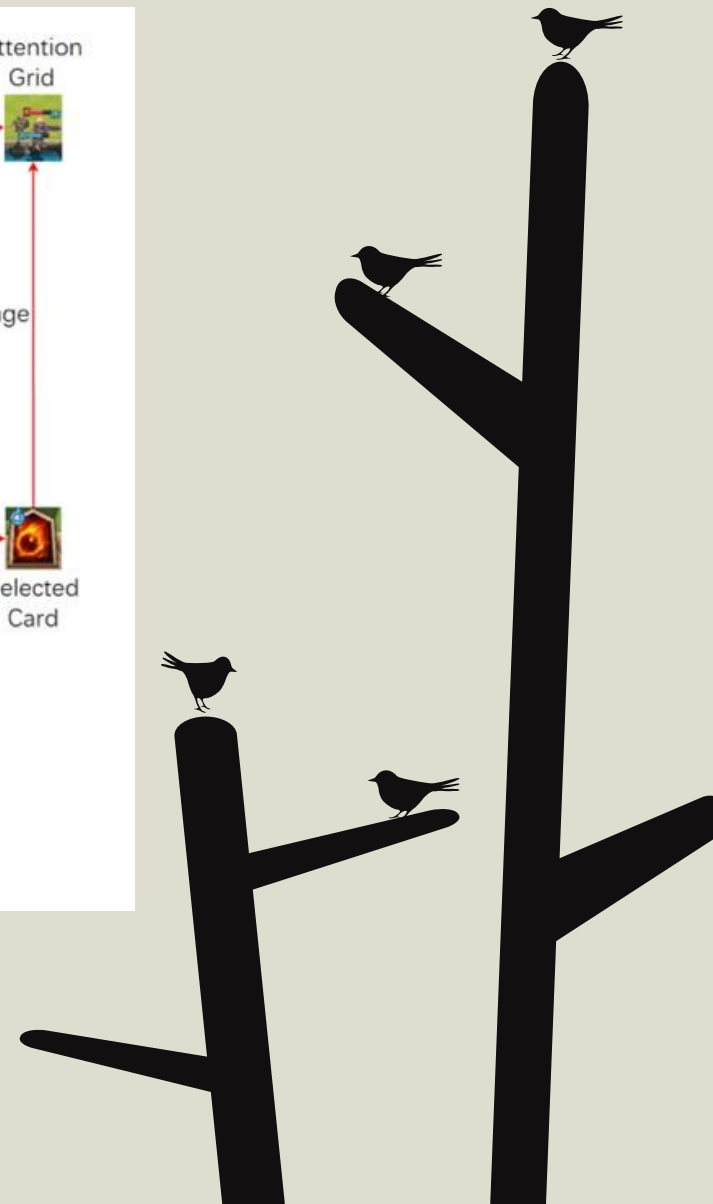


Figure 2: Selection-Attention Model.






- Here we assume that game has 3 types of ending for a player:  $W = \{1, \text{if is equal}, 0 \text{ if you lose and } 2 \text{ otherwise}\}$ . Cards appears fours in hand controlled by a random seed and appears constant until the game is finished. Every card needs elixir to spawn, and elixir max is 10 and will by generate while you are playing (1p at every 2.8 seconds).
- The action of agent could be formalized like  $U_{i,t} \in [0, 1]$ , (2)  $P_{i,t} = (x, y)$ . ( $i = \{1, 2, 3, 4\}$ ) , where  $U$  is probability to choose card  $i$  at position  $P$  at time  $t$ .
- Card choice by selection part is based on both battle field states and complex card features, while card usage by attention part is mainly related to enemy states.

- For every image we will define a vector with 3 dimensions,  $\text{vector} \in [1, 10] \times [1, 10] \times \{0, 1\}$ , those are health, class type and player Faction, we don't take position in computation, because this will be in a different layer analyzed. From now of each episode will update Policy from that moment and will try to find best way to select a card, this episode is for selections.

- We will use DRL to find best card selection because is too big and we need to find a good approximation and, in this way, we use CNN.
- In this game we can get 3 types of reward: troop reward, card reward and tower reward and the final reward is the sum of those 3 types.

The background features a solid tan color. On the left side, there are several thick, black, stylized tree branches. A small white bird is perched on one of these branches near the top left. Another white bird is perched on a lower branch further down. In the center, there is a single, thin, white vertical branch that extends from the bottom to the top of the frame. A white bird is perched on a horizontal branch that extends from this central white branch towards the right. On the right side, there are more thick, black, stylized tree branches. A white bird is perched on one of these branches near the top right.

# Article 4 : Reinforcement learning to play monopoly



# Intro

- Reinforcement learning is that type of machine learning that training a model by the environment and take decisions to maximize reward resulted by actions at each step. This type of IA if used in a lot of games, from normal game, Clash Royal like my previous presentation to Monopoly, a board game that is based on "the Georgist concept of a single land value tax". A board game like monopoly has a big state space.
- For our RL model we need to define Markov decisions process and our single agent.



# State space

- Firstly we define state space, so we can define state like a 3 dimensions vector with next attributes (area, finance, position) every attribute is at time  $t$ .
- The area of the game contains information about game properties and enemy properties. It is a matrix  $10 \times 2$ , where 10 is number of colours tag, and trains and industries, first line represent player and second one is represent by colour tag. Each element  $(x, y)$  specifies the percentage that the player  $y$  owns from group  $x$ , where value 0 indicates that the player does not own anything and 1 that at least one hotel has been built.
- More specifically, the domain of the parameter  $x$  can be described as follows :
  - 0, player does not own anything from the current group
  - $0 < x < 12/17$  player possesses at least one property of the current group



- 12/17 player has all the properties of the current group but has not built anything
- $12/17 < x \leq 1$  player has built at least one house in at least one of the current group's properties (this is taken from article)
- 12 is the cmmmc form 2,3,4 where those are possible number of card from a tag.
- Position represent actual position of player on board in a big relation with tag, so could be 0 or 1.
- The finance represents the report of you number of properties in computation with the others player and the second one would be the maximum amount of money of a player.



# Action space

- Actions for this game are:
- 1) Spending -> You buy something or you pay to an opponent for rent;
- 2) Selling -> selling properties to opponent to make more money;
- 3) doing nothing -> you don't do anything that has an impact to money
- For reward we will use next sigmoid function:

equation :

$$r = \frac{\frac{v}{p} * c}{1 + |\frac{v}{p} * c|} + \frac{1}{p} * m,$$

where:

- $p$  is the number of players
- $c$  is a smoothing factor
- $v$  is a quantity representing the player's total assets value and is calculated by adding the value of all the properties in the possession of the player, minus the properties of all of his opponents
- $m$  is a quantity representing the player's finance and is equal to the percentage of the money the player has to the sum of all players' money

Player	Number of games won	Percentage
Random	20	2%
Fixed Policy	286	28.6%
RL agent	694	69.4%
Total	1000	100%

Table 1. Number of wins for each agent over 1000 games

# Bibliography

- <https://productcoalition.com/applications-of-reinforcement-learning-in-mobile-games-a-neuroscience-approach-1ce35fb5019c>
- <https://bdtechtalks.com/2021/10/04/ea-reinforcement-learning-game-testing/>
- <https://www.ijcai.org/proceedings/2019/0631.pdf>
- <http://doc.gold.ac.uk/aisb50/AISB50-S02/AISB50-S2-Bailis-paper.pdf>



# Thank You!

- Furtuna Theodor
- Lazar Stefania
- Radu Cosmin
- Soare Dennis
- Udroi Laura



GamingRL