iGGi Your future in games research

MAP-Elites to illuminate game space in CaveSwing

Martin Balla, Adrián Barahona-Ríos, Adam Katona, Nuria Peña Pérez & Ryan Spick

UNIVERSITY of York

EPSRC Engineering and Physical Sciences Research Council

Queen Mary University of London

Search space

Search Space = Game Space → All possible combinations of game parameters (e.g. gravity, distances…)

Search algorithms aim to find **optimal game parameters** for a particular game evaluation function (fitness function) by **exploring the game space**

Some search algorithms:

Optimization Algorithms **VS** **Quality Diversity Algorithms**

Optimization Algorithms

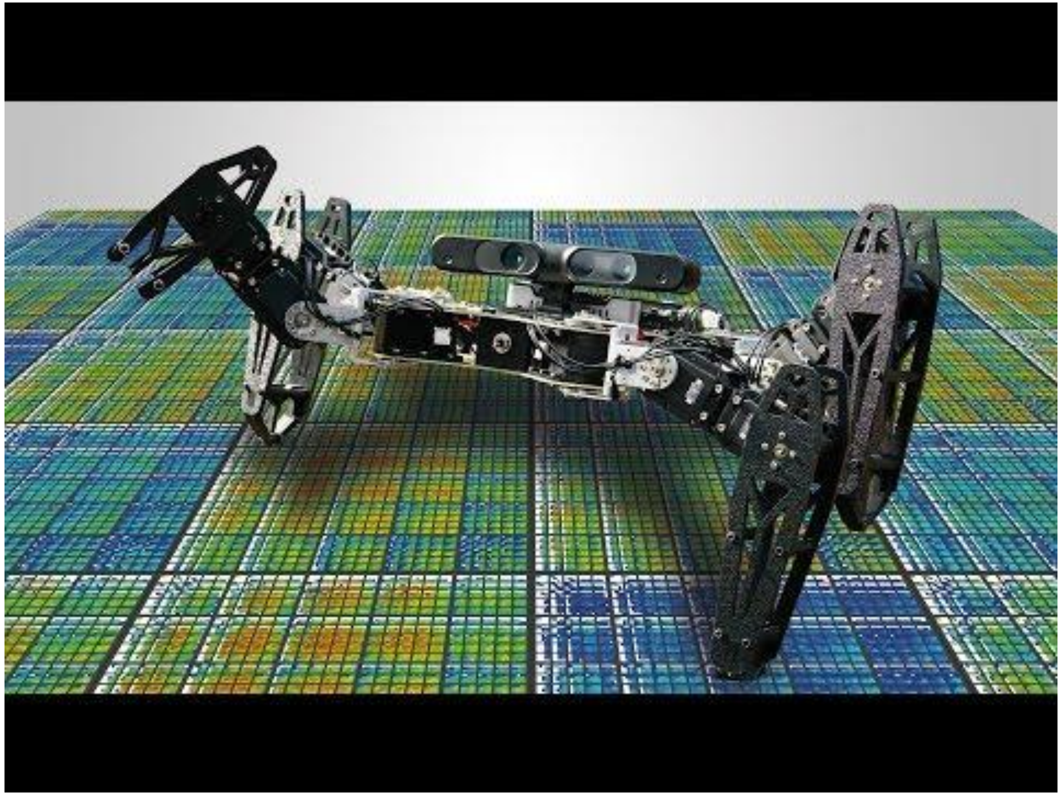
- Find best solution for a given set of parameters
- Optimizes fitness function (usually score)

Quality Diversity Algorithms

- Find a set of good quality solutions
- Optimize performance and diversity based on user-defined behaviour characterization

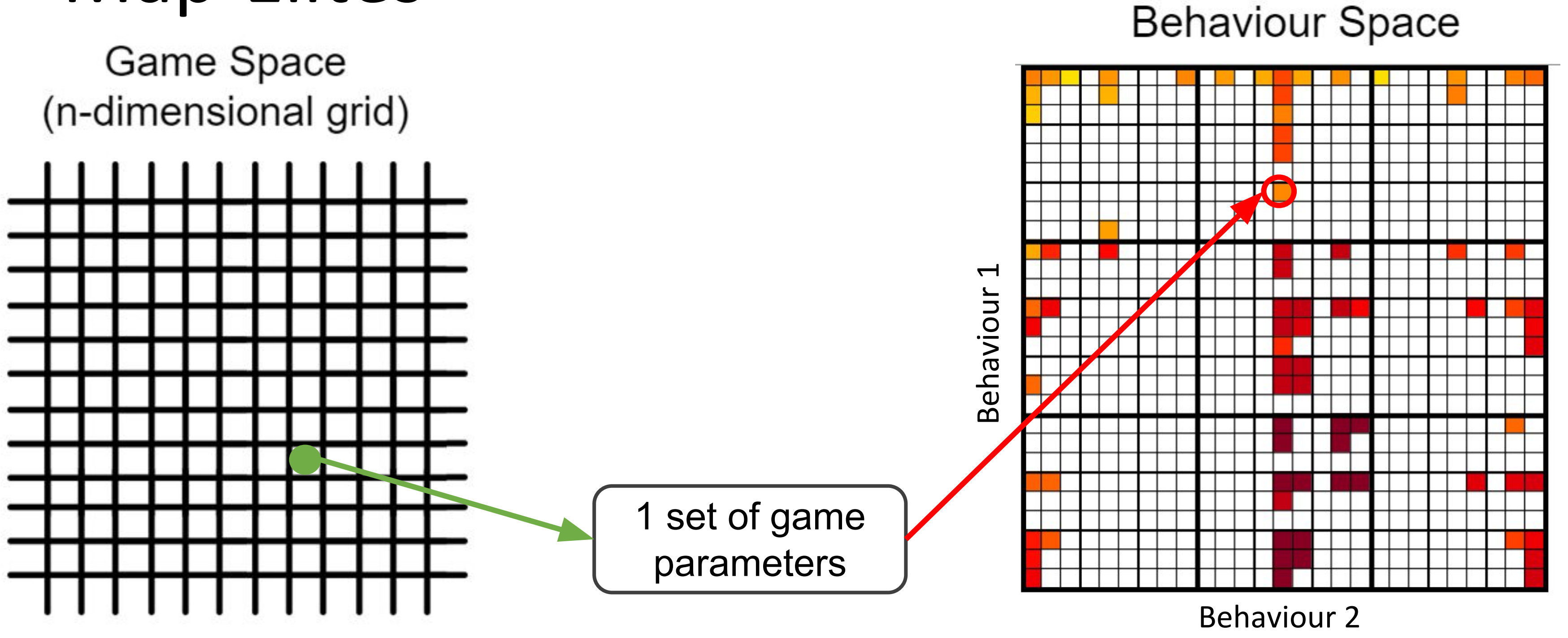
MAP-ELITES

Map-Elites



```
$(\mathcal{P} \leftarrow \emptyset, \mathcal{X} \leftarrow \emptyset)$    ▷ Create an empty, $N$-dimensional map of elites: {solutions $\mathcal{X}$ and their performances $\mathcal{P}$}
for iter $= 1 \rightarrow I$ do    ▷ Repeat for $I$ iterations.
    if iter $< G$ then    ▷ Initialize by generating $G$ random solutions
        $\mathbf{x}' \leftarrow$ random_solution()
    else    ▷ All subsequent solutions are generated from elites in the map
        $\mathbf{x} \leftarrow$ random_selection($\mathcal{X}$)    ▷ Randomly select an elite $x$ from the map $\mathcal{X}$
        $\mathbf{x}' \leftarrow$ random_variation($\mathbf{x}$)    ▷ Create $x'$, a randomly modified copy of $x$ (via mutation and/or crossover)
    $\mathbf{b}' \leftarrow$feature_descriptor($\mathbf{x}'$)    ▷ Simulate the candidate solution $x'$ and record its feature descriptor $\mathbf{b}'$
    $p' \leftarrow$performance($\mathbf{x}'$)    ▷ Record the performance $p'$ of $x'$
    if $\mathcal{P}(\mathbf{b}') = \emptyset$ or $\mathcal{P}(\mathbf{b}') < p'$ then    ▷ If the appropriate cell is empty or its occupants's performance is $\leq p'$, then
        $\mathcal{P}(\mathbf{b}') \leftarrow p'$    ▷ store the performance of $x'$ in the map of elites according to its feature descriptor $\mathbf{b}'$
        $\mathcal{X}(\mathbf{b}') \leftarrow \mathbf{x}'$    ▷ store the solution $x'$ in the map of elites according to its feature descriptor $\mathbf{b}'$
return feature-performance map ($\mathcal{P}$ and $\mathcal{X}$)
```

Map-Elites



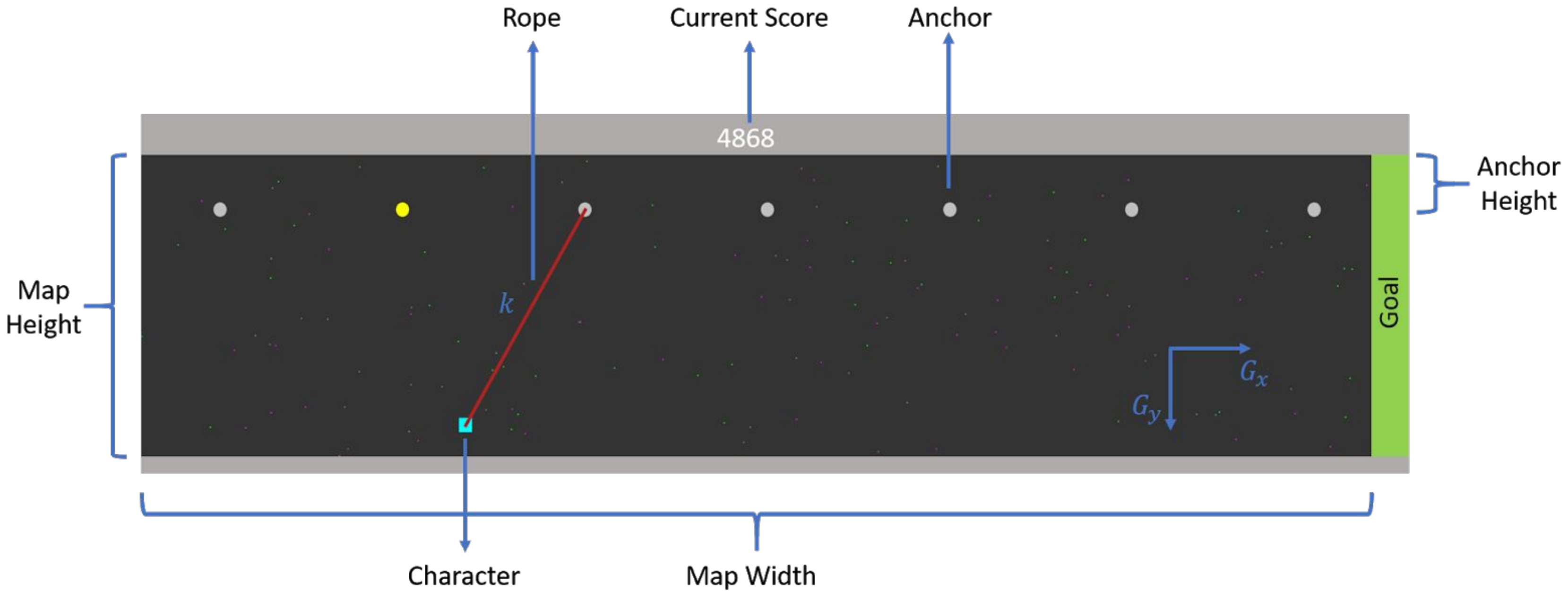
Maps game parameters to behaviour space

Find possible ways a game can be played: diversity and quality

What do we need?

1. Select a game:
   a. Understand what parameters are part of the game space
   b. Characterize the desired behaviours to explore diversity
2. Find a way to evaluate the quality for each combination of behaviours.

Game: CaveSwing



Game Space in CaveSwing

Fixed Parameters

Map Dimensions

Max Duration

Anchor's Height

Score Parameters

Explored Parameters

Gravity (x and y)

Hooke's constant

Number of Anchors

Loss Factor

For this particular example not all gamespace is explored

Behaviour characterization in CaveSwing

Average height of the agent's trajectory

Game duration (ticks it took to finish the game)

What do we need?

1. Select a game:
   a. Understand what parameters are part of the game space
   b. Define what arising behaviours are interesting
2. Find a way to evaluate the quality for each combination of behaviours.

Evaluating the quality of each combination

Rolling Horizon Evolutionary Algorithm (RHEA)

1. Execute random sequence of actions in the forward model
2. Calculate a score of the state reached → based on game score
3. Mutate current action sequence and evaluate it
4. If better, overwrite previous solution
5. Continue until solution is good enough or maximum time is elapsed

Evaluating the quality of each combination

Evaluate performance of RHEA agent using game score

$$Score = xP_x + yP_y - tP_t$$

1. How much the agent has progressed in the horizontal (x) direction
2. How high it is on the vertical (y) direction
3. How fast it is completing the level

Score is calculated for every time point (t)

Project implementation

Java Server:

- CaveSwing implementation
- RHEA agent

Python Client:

- Map-Elites implementation
- Visualization tools

Project implementation: Java server

To measure performance every set of parameters was evaluated 20 times in the game → performance = agent’s score per run

The observed behaviour features (average height for the game run and time for the game run) were also stored

These values were averaged over the 20 runs and returned to the client

Project implementation: Python client

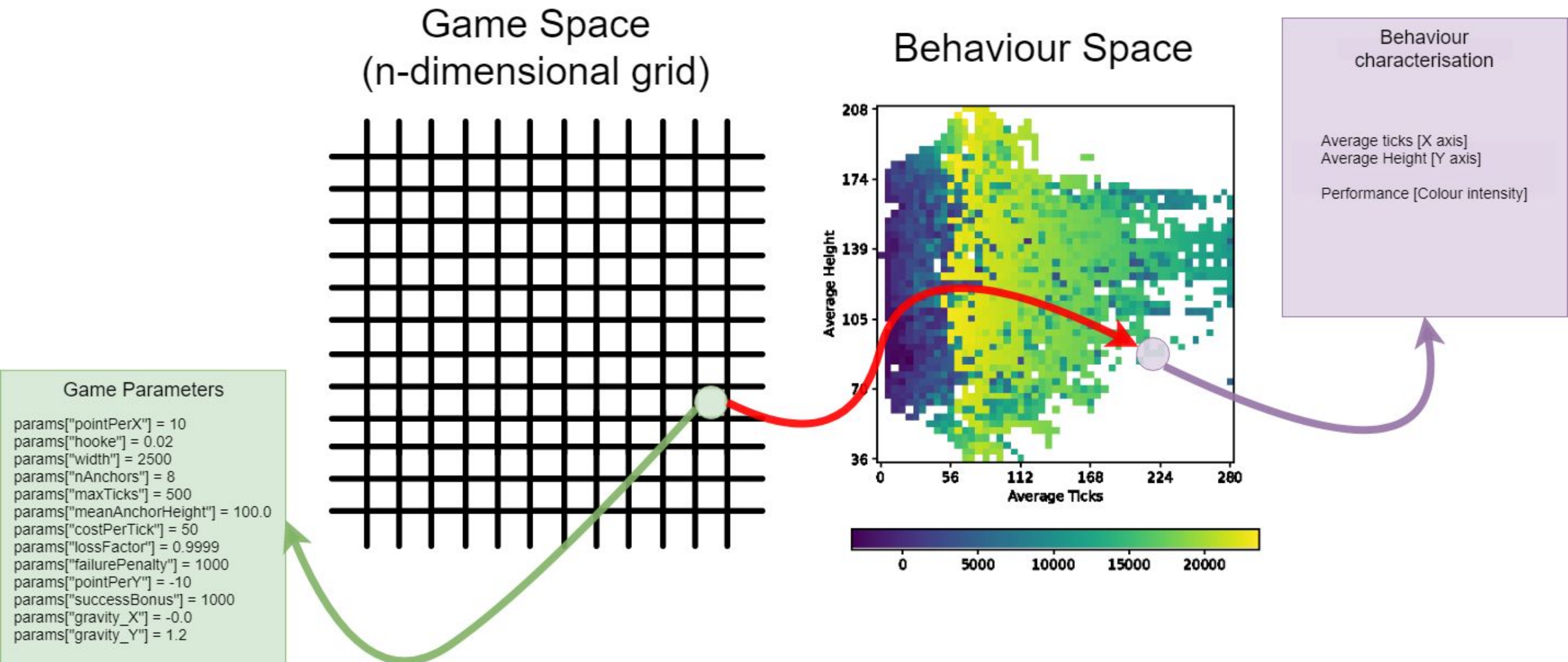
In the client MAP-Elites created a grid with 2500 cells, each corresponding to a set of behaviours and with its associated performance value

Initialize the grid: obtain the range for each behaviour by running a thousand evaluations (measure min, max and create grid with uniform sized cells)

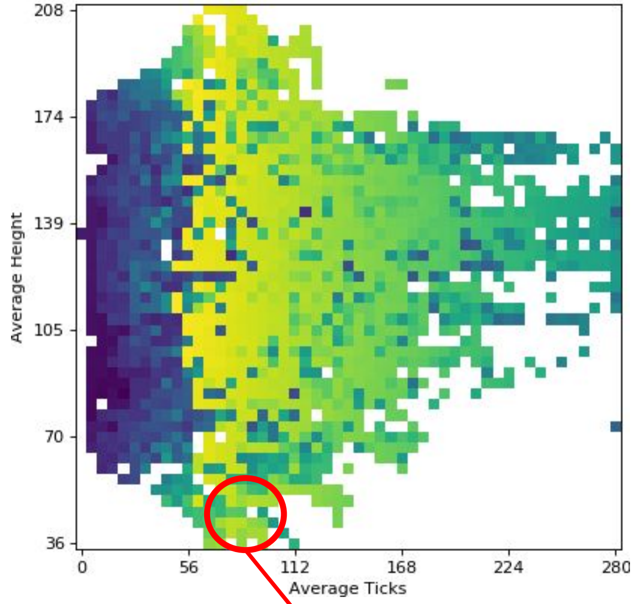
The selection and evaluation process was repeated:

- Initial random evaluations: While all the cells in the behaviour map were empty the algorithm randomly picked parameters and evaluated them
- Later, the algorithm continued by randomly mutating populated cells

The entire process:

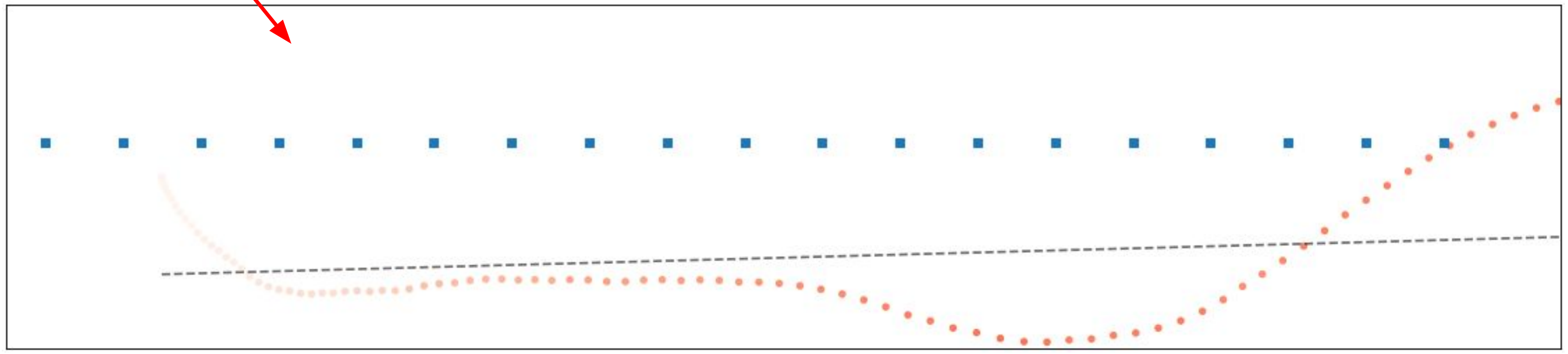


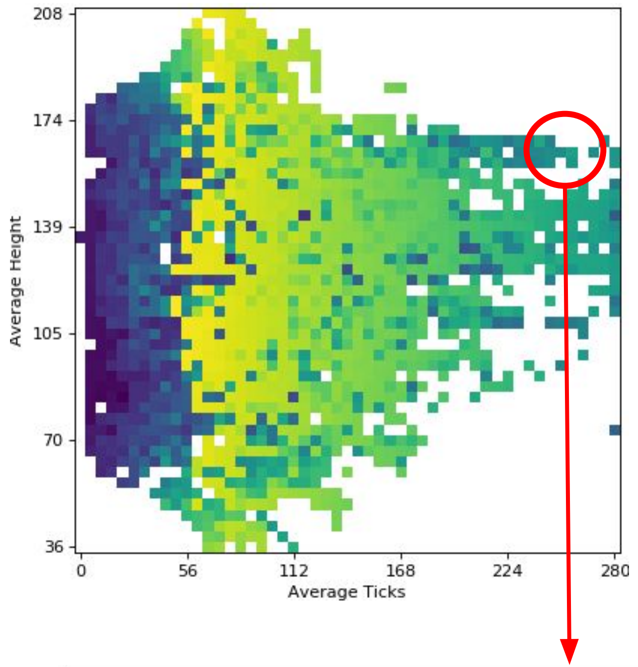
```
params["pointPerX"] = 10
params["hooke"] = 0.02
params["width"] = 2500
params["nAnchors"] = 8
params["maxTicks"] = 500
params["meanAnchorHeight"] = 100.0
params["costPerTick"] = 50
params["lossFactor"] = 0.9999
params["failurePenalty"] = 1000
params["pointPerY"] = -10
params["successBonus"] = 1000
params["gravity_X"] = -0.0
params["gravity_Y"] = 1.2
```



Fast gameplay

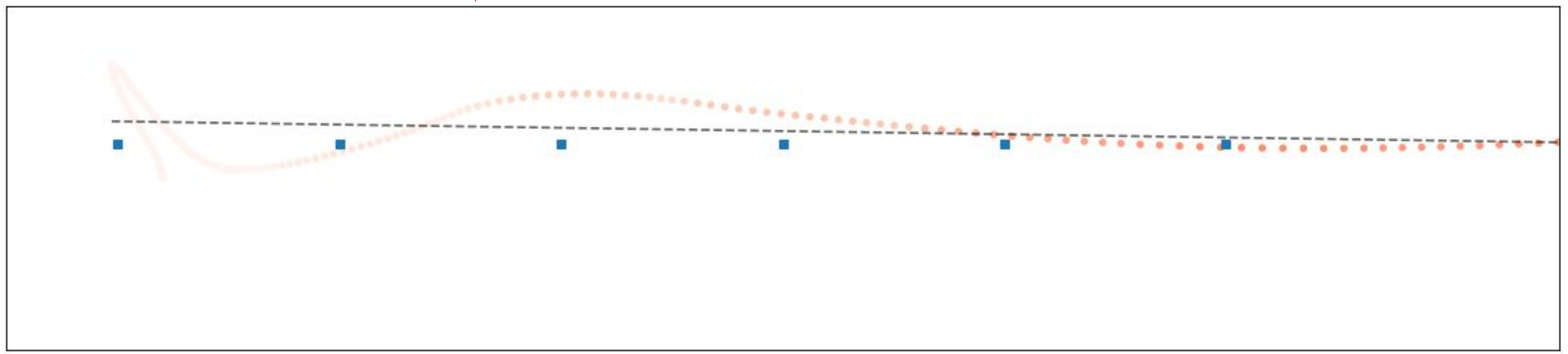
Low height





Slow gameplay

Medium-high height



Some instructions...

https://bit.ly/2lK0oau