Описание среды

Параметры среды

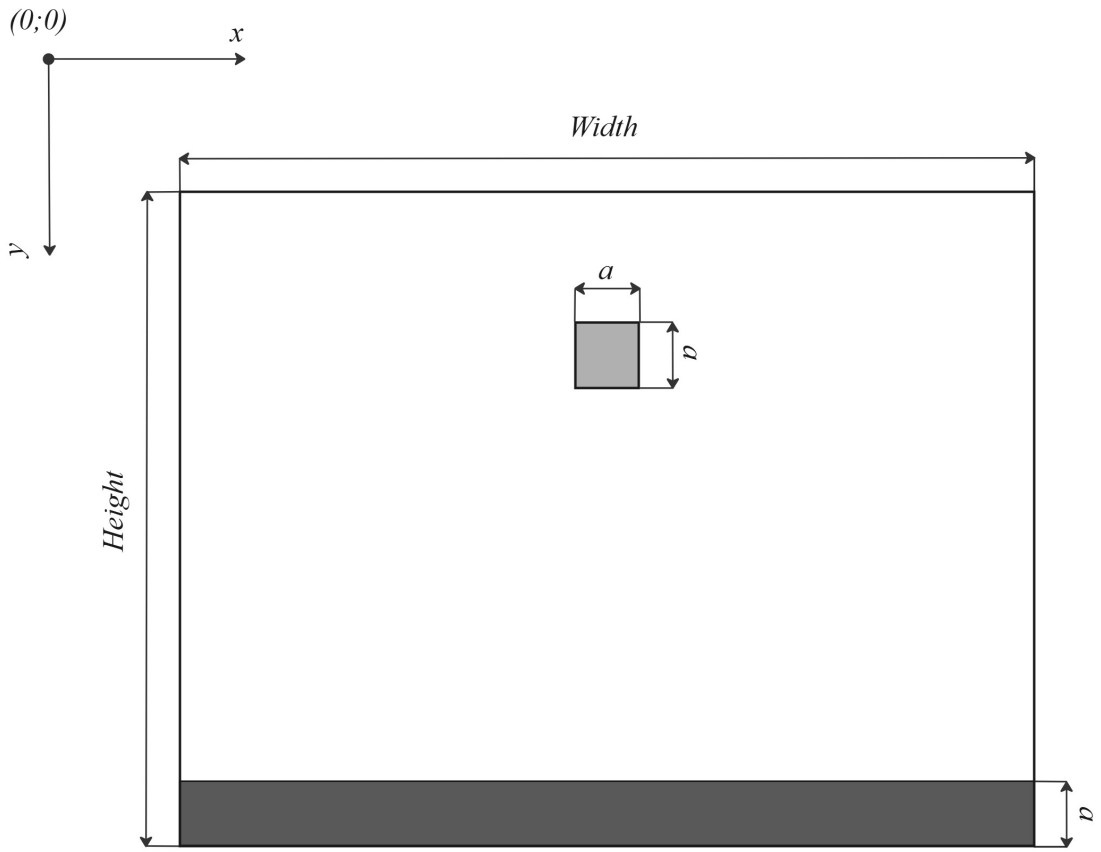


Рисунок 1 — Параметры среды

- Окно размером $Width \times Height$
- Кубики размером $a \times a$
- Платформа размером $Width \times a$

На вход модель получает Q-таблицу. Состояния — координаты X и Y центра масс башни ($CenterX$ и $CenterY$).

Действия агента

Создание кубика. Координата Y фиксирована. Координата X выбирается из кортежа дискретных координат:

$$\{x_0, x_1, \ldots, x_n\}, \quad \text{где } n = \frac{Width}{a}$$

Шаг $\Delta x = a$. Координата X выбирается случайно с вероятностью ε, иначе выбирается наилучшее действие из Q-таблицы.

На выходе — индекс действия, соответствующий выбранной координате X.

Целевая функция

Цель — максимизировать суммарную награду.

Условия

Каждый новый блок с координатами x_{new} и y_{new} проверяется условиями:

– Блок не выходит за пределы допустимой ширины по X:

$$|x_{\text{right}} - x_{\text{new}}| < 3a \quad \text{или} \quad |x_{\text{left}} - x_{\text{new}}| < 3a,$$

где x_{right} и x_{left} - правая и левая координаты башни.

– Блок не упал ниже игрового окна:

$$y_{\text{new}} < Height$$

Если условия не выполняются:

$$Reward = Reward - 20$$

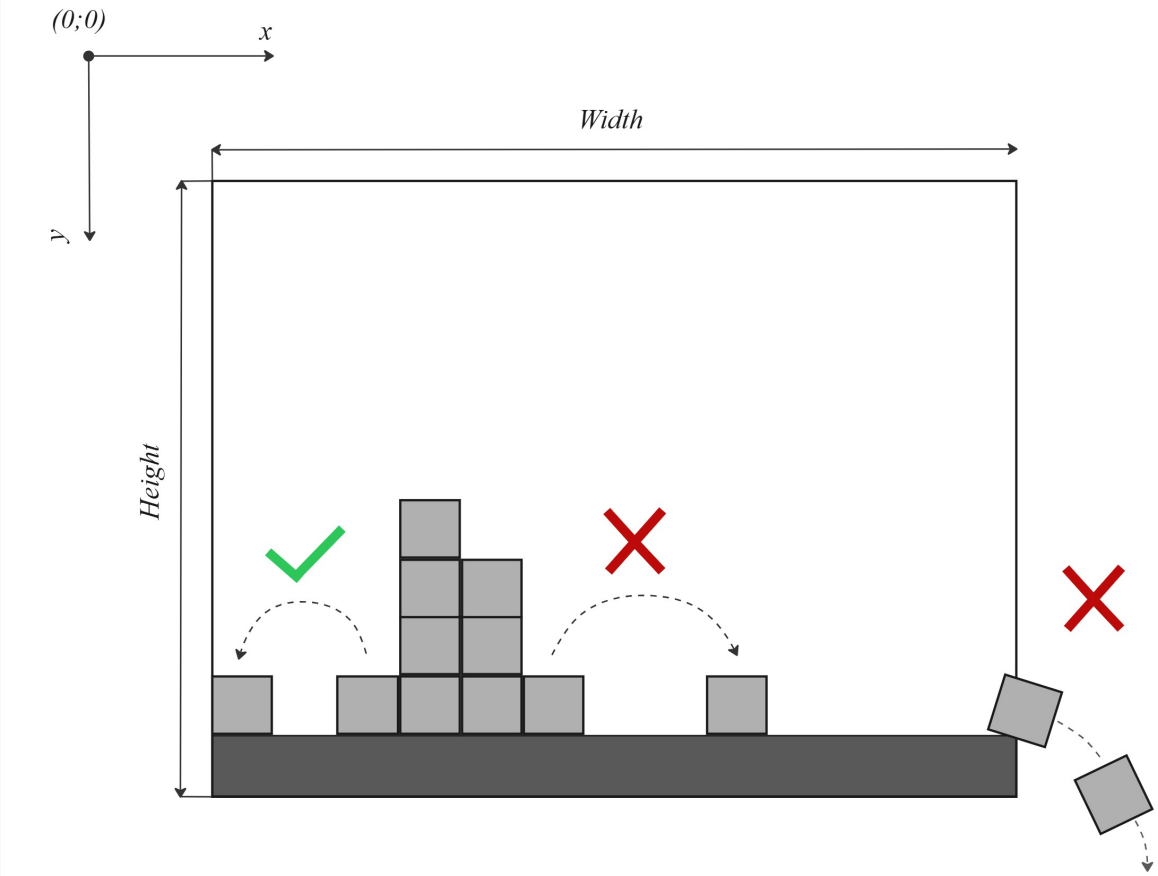


Рисунок 2 — Условия завершения игры

Награда

$$Reward = 10 + tower\_height + pyramid\_score$$

где:

– $tower\_height$ — вознаграждение за высоту башни, определим как разность $Height$ и наименьшей координаты Y среди всех блоков.

– $pyramid\_score$ — вознаграждение за “похожесть” формы башни на пирамиду. Как известно, у “идеальной” пирамиды центр масс делит медиану в отношении 2:1, считая от вершины. Отталкиваясь от этого факта определим вознаграждение $pyramid\_score$ как точность попадания в желаемый центр масс умноженный на 10 для весомости награды.

$$pyramid\_score = \left(\frac{Height - C_y^t}{tower\_height/3}\right) \cdot 10 + \left(\frac{C_x^t - x_{\min}}{(x_{\max} - x_{\min})/2}\right) \cdot 10$$

Описание решения

Состояние S

Состояние среды определяется центром масс башни:

$$S_t = (C_x^t, C_y^t)$$

где

$$C_x^t = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad C_y^t = \frac{1}{n}\sum_{i=1}^{n} y_i$$

n — количество блоков в башне на шаге t.

Действия A

Агент выбирает позицию x для следующего блока из дискретного множества:

$$A_t \in \{x_0, x_1, \ldots, x_n\}, \quad x_i = i \cdot a, \quad n = \frac{Width}{a}$$

Стратегия выбора действия

Агент выбирает действие по ε-жадной стратегии:

$$A_t = \begin{cases} \text{случайное } a \in A, & \text{с вероятностью } \varepsilon \\ \arg\max\limits_{a \in A} Q(S_t, a), & \text{с вероятностью } 1 - \varepsilon \end{cases}$$

Награда

Если блок размещён с нарушением условий:

- $|x_{\text{new}} - x_{\text{left/right}}| \geqslant 3a$
- или $x_{\text{new}} \geqslant Height$

то агент получает штраф:

$$Reward = -20$$

Иначе награда рассчитывается как:

$$R = 10 + tower\_height + pyramid\_score$$

где

$$tower\_height = Height - \min\{y_1, y_2, \ldots, y_n\}$$

$$pyramid\_score = \left(\frac{Height - C_y^t}{tower\_height/3}\right) \cdot 10 + \left(\frac{C_x^t - x_{\min}}{(x_{\max} - x_{\min})/2}\right) \cdot 10$$

Обновление Q-таблицы

После выполнения действия A_t и получения нового состояния S_{t+1} и награды R_t, агент обновляет таблицу Q следующим образом:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \cdot \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right]$$

где:

- α — коэффициент обучения (обычно $\in [0.1, 0.5]$)
- γ — коэффициент дисконтирования будущих наград (обычно ≈ 0.95)

Обучение

Обучение происходит по эпизодам. В каждом эпизоде:

1. Башня строится по одному блоку за раз, пока не достигнуто ограничение или не нарушены условия.
2. После завершения эпизода агент получает награду и обновляет таблицу.
3. Значение ε уменьшается:

$$\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot 0.999)$$

Цель обучения

Агент стремится максимизировать общее вознаграждение за эпизод:

$$\sum_{t=0}^{T} Reward_t$$

где T — количество шагов (блоков) в эпизоде.

Оптимальная стратегия — строить устойчивую, высокую и симметричную башню, избегая штрафов.

Взаимодействие компонентов

Общая постановка задачи

Требуется разработать систему, в которой агент с использованием классического Q-обучения обучается размещать падающие блоки таким образом, чтобы построить устойчивую башню, похожую на пирамиду. Физическая симуляция осуществляется с использованием библиотеки Pymunk, визуализация — через Pygame.

Декомпозиция задачи

1. Физическая и визуальная среда

- **Инициализация среды:**
  - Создание окна Pygame;
  - Инициализация физического мира Pymunk;
  - Установка гравитации.
- **Платформа:**
  - Горизонтальная поверхность у нижней границы окна;
  - Фиксированный объект в симуляции.
- **Блоки:**
  - Квадратные тела с массой, моментом инерции и трением;
  - Создаются по координате X, определяемой агентом;
  - Падают под действием гравитации.
- **Отображение:**
  - Отрисовка платформы и блоков;
  - Визуальное отображение прогресса агента.

2. Агент (QAgent)

- **Структура агента:**
  - Q-таблица: $dict[state][action]$;
  - Параметры: скорость обучения α, дисконт γ, исследование ε, шаг дискретизации.
- **Состояние:**
  - Центр масс башни: (C_x, C_y).
- **Действия:**
  - Дискретные позиции по x с шагом $\Delta x = a$;
  - Агент выбирает координату x, по которой будет создан следующий блок.
- **Алгоритмы:**
  - Выбор действия: ε-жадная стратегия;
  - Обновление Q-таблицы по правилу Беллмана.

3. Игровой цикл и логика (Game)

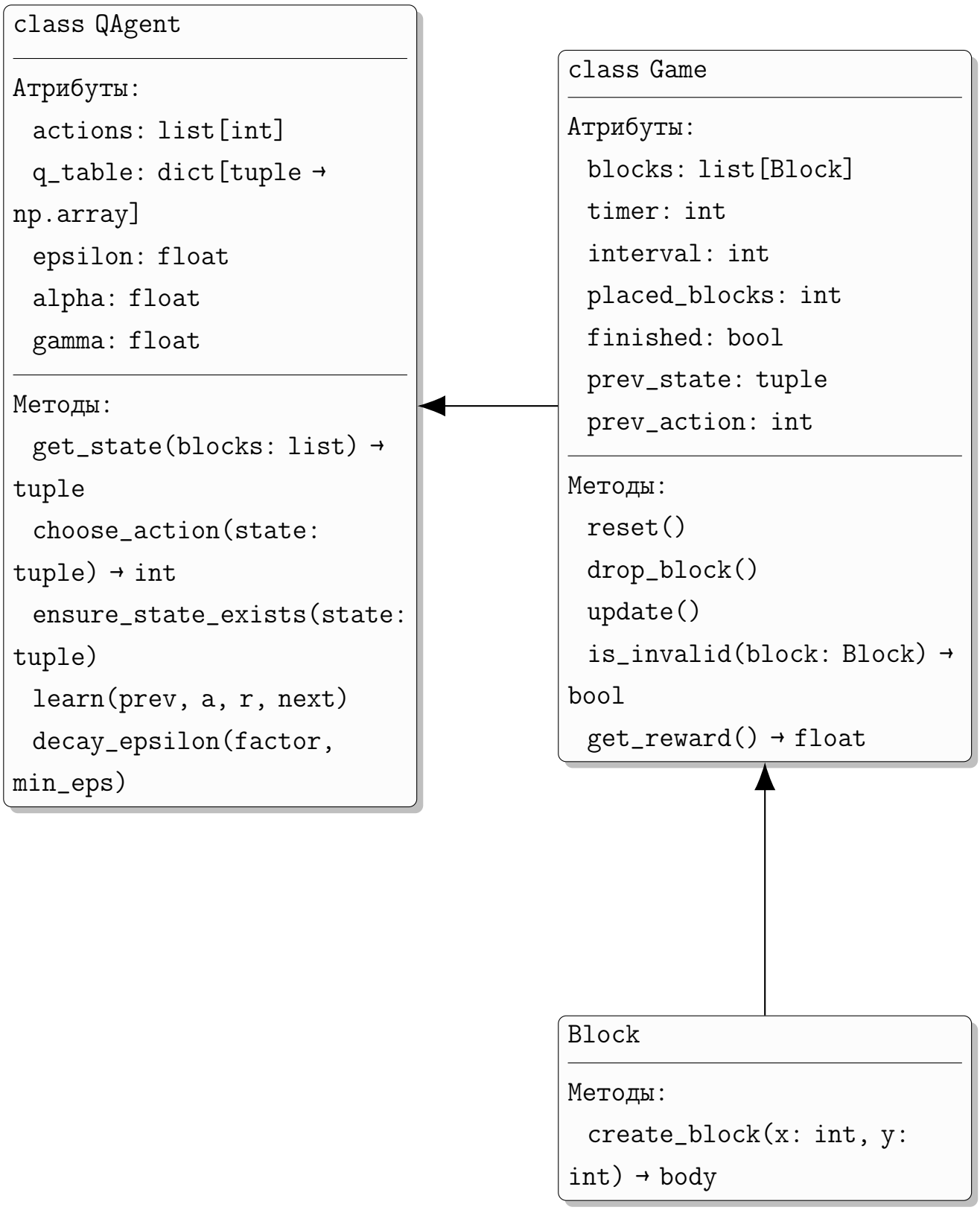
- **Цикл симуляции:**
  - На каждом шаге агент выбирает действие;
  - Блок создаётся и падает на платформу;
  - После стабилизации блоков оценивается результат.
- **Награда:**
  - За падение или неустойчивую конструкцию — штраф;
  - За высокую и устойчивую башню — положительная награда;
  - Дополнительная оценка симметричности (форма пирамиды).
- **Переход между эпизодами:**
  - Полный сброс физической сцены;
  - Очистка блоков и пространства;
  - ε обновляется (уменьшается).

4. Функции оценки и анализ стабильности

- **Определение стабильности:**
  - Проверка движения блоков;
  - Проверка отклонений центра масс;
  - Проверка выхода за границы экрана.
- **Оценка башни:**
  - Высота башни $tower\_height$;
  - Центр масс (C_x, C_y);
  - Ширина основания $[x_{\text{left}}, x_{\text{right}}]$;
  - Вычисление "похожести на пирамиду".

Сценарий эпизода

1. Сброс симуляции: платформа остаётся, блоки удаляются;
2. Агент получает начальное состояние (C_x, C_y);
3. Агент выбирает действие — координату X для следующего блока;
4. Блок создаётся и падает;
5. После стабилизации всех блоков:
   - Рассчитывается новое состояние и награда;
   - Q-таблица обновляется;
6. Эпизод завершается при падении или превышении лимита блоков;
7. Начинается следующий эпизод.


class QAgent
Атрибуты:
actions: list[int]
q_table: dict[tuple → np.array]
epsilon: float
alpha: float
gamma: float
Методы:
get_state(blocks: list) → tuple
choose_action(state: tuple) → int
ensure_state_exists(state: tuple)
learn(prev, a, r, next)
decay_epsilon(factor, min_eps)
class Game
Атрибуты:
blocks: list[Block]
timer: int
interval: int
placed_blocks: int
finished: bool
prev_state: tuple
prev_action: int
Методы:
reset()
drop_block()
update()
is_invalid(block: Block) → bool
get_reward() → float
Block
Методы:
create_block(x: int, y: int) → body