

Godot Gym API

Фреймворк для обучения RL-агентов в 3D средах из Godot

Константин Черняк

ИТМО

Магистр

Об авторе 🗨️

Второй курс магистратуры ИТМО

Работаю над системой помощи водителю в АТОМ

Мечтаю об автопилоте на VA3-2105, основанном исключительно на RL



План

1. Для чего?
2. Почему обратить внимание?
3. А что еще есть на Github?
4. Демо без регистрации и СМС
5. Дальнейшее развитие
6. Ответы на вопросы

Описание

Godot Gym API - фреймворк для обучения RL-агентов на Python в 2D- и 3D-средах, реализованных на игровом движке Godot 3.

Фреймворк предоставляет:

- Расширяемый функционал для написания среды обучения
- Транспорт данных между Godot и Python по протоколу TCP

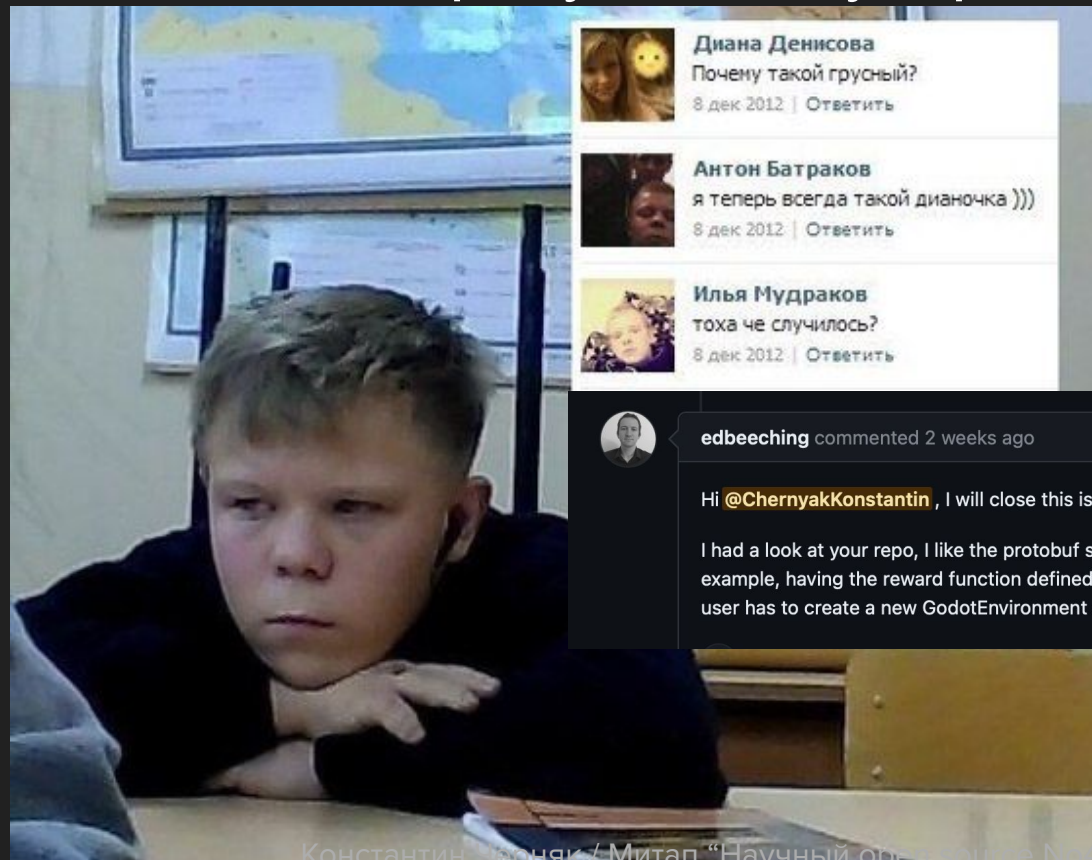
Почему фреймворк должен быть интересен? 🤔

1. Быстрая передача любых данных через формат protobuf, в том числе:
 - a. Облака точек ☁️
 - b. Изображения / последовательность изображений 🖼️
2. Возможность писать мало кода 🐛
3. Документация с примерами ☐

Что уже существует? 🧐

GodotRLAgents	GymGodot	GodotAIGym
<ul style="list-style-type: none">+ Самое популярное репо+ Поддержка Godot 4+ Запуск моделей в ONNX- Поддержка только Godot4- JSON для передачи данных- Недостаточно документирован- Частично работающие примеры	<ul style="list-style-type: none">- JSON для передачи данных- Негибкий- Нет документации- Развитие прекращено	<ul style="list-style-type: none">+ Shared memory для передачи данных+ Документация- Только для Linux- Только Godot 3.2- Необходима сборка- Необходимо писать много кода- Кажется, развитие прекращено


Может законтрибуть в популярное решение? 🙄



 **Диана Денисова**
Почему такой грустный?
8 дек 2012 | Ответить

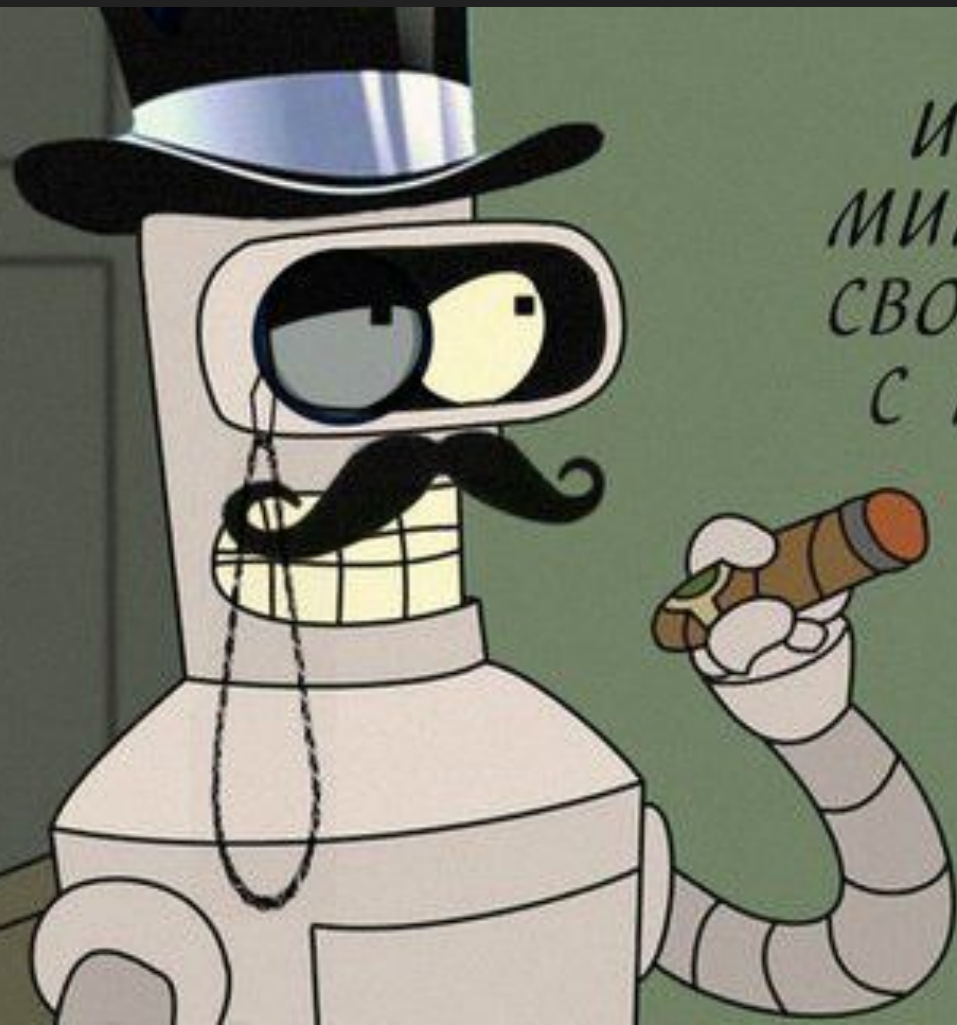
 **Антон Батраков**
я теперь всегда такой дианочка)))
8 дек 2012 | Ответить

 **Илья Мудраков**
тоха че случилось?
8 дек 2012 | Ответить

 **edbeeching** commented 2 weeks ago Owner ...

Hi **@ChernyakKonstantin**, I will close this issue unless you have any more details.

I had a look at your repo, I like the protobuf stuff, but I think it requires the user to code too much on the python side. For example, having the reward function defined in python is far less flexible than defining it in Godot. Also for each new env, the user has to create a new GodotEnvironment child class in python.



И БУДЕТ У МЕНЯ,
МИЛОСТЛИВЫЕ СУДАРИ,
СВОЙ ИГОРНЫЙ ДОМЪ
С МАДЕМУАЗЕЛЯМИ
И ПРЕФЕРАНСОМЪ...

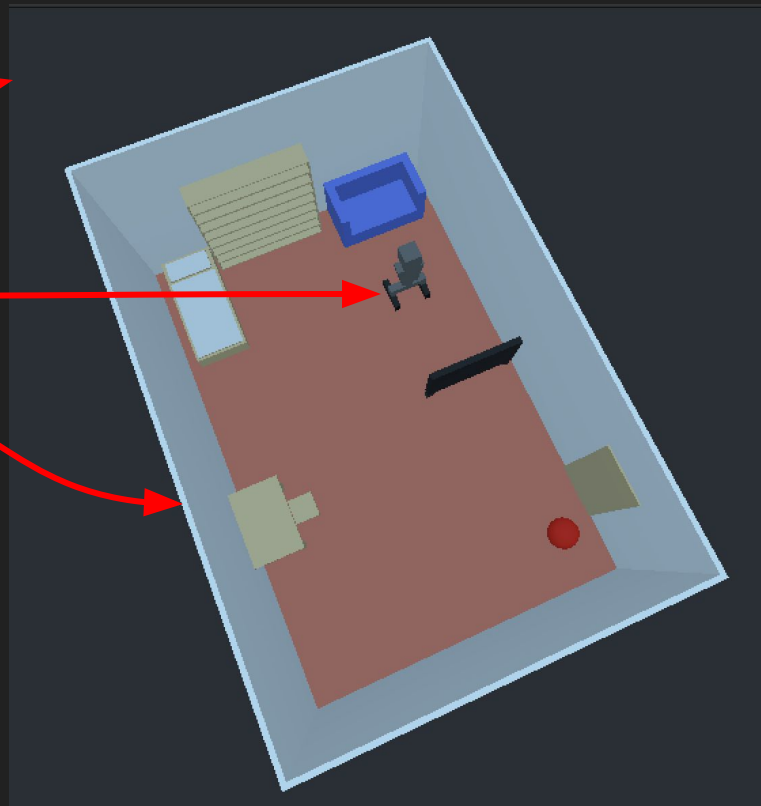
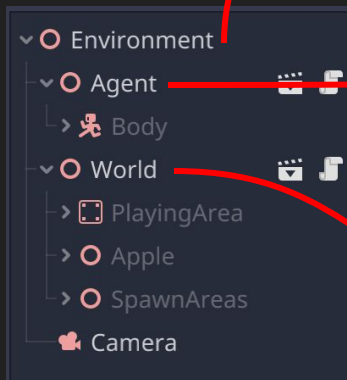
Демо

Есть готовая игра.

Робот управляется стрелками на клавиатуре.

При касании роботом красной сферы, сфера респавнится в случайной точке.

Что нужно сделать, чтобы она стала пригодной для обучения?



Демо

Задача агента: Найти красную сферу.

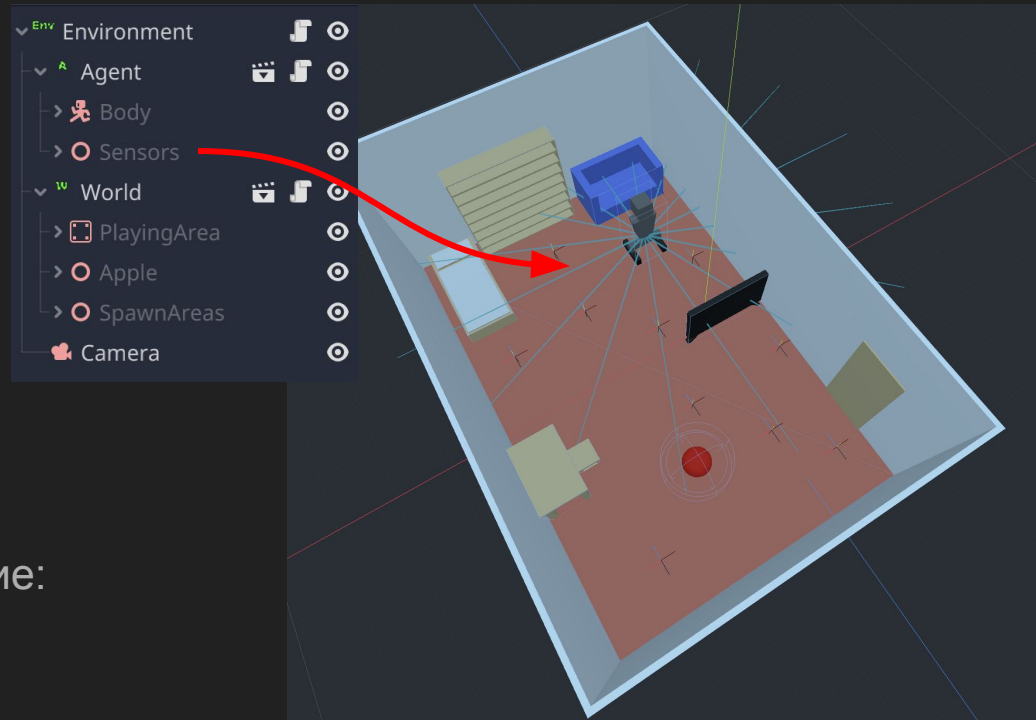
Агент смотрит на мир через датчики.

Датчики сообщают:

- Расстояние до препятствий
- Расстояние до цели

Направления движения агента прежде:

- Вперед
- Назад
- Влево
- Вправо



Демо . Изменим World

```
1 extends Spatial
2
3 onready var apple = $Apple
4 onready var spawn_areas = $SpawnAreas
5
6 > func _ready():
10 >|
11 > func reset():
14
15 > func _on_catch_apple(_body):
17
18 > func _sample_initial_position() -> Vector3:
```

```
1 extends RLEnvWorld
2
3 onready var apple = $Apple
4 onready var spawn_areas = $SpawnAreas
5 onready var apple_caught: bool = false
6
7 > func _ready():
11 >|
12 > func reset():
16
17 > func _on_catch_apple(_body):
19 >|
20 > func get_data(observation_request, storage) -> void:
22
23 > func _sample_initial_position() -> Vector3:
```

Демо . Изменим Agent

```
1 extends Spatial
2
3 # How fast the agent moves in meters per second.
4 var speed = 5
5 # Current velocity of the agent.
6 var velocity: Vector3 = Vector3.ZERO
7
8 onready var body = $Body
9
10 > func move_body(): ☺
28
29 > func _ready(): ☺
31
32 > func _physics_process(delta): ☺
```

```
1 extends RAgent
2
3 export var target_node_path: NodePath
4
5 # How fast the agent moves in meters per second.
6 var speed = 5
7 # Current velocity of the agent.
8 var velocity: Vector3 = Vector3.ZERO
9 # The maximum distance of the agent sensors.
10 var max_sensor_distance = 5
11 var current_action: int = -1
12 var target
13
14 onready var body = $Body
15 onready var sensors = $Sensors
16
17 > func get_data(observation_request, storage) -> void: ☺
29 >
30 > func set_action(action): ☺
32 >
33 > func reset(new_position): ☺
37
38 > func move_body(): ☺
56
57 > func _ready(): ☺
59
60 > func _physics_process(delta): ☺
```

Демо . Изменим Environment

```
1  extends RLEnvironment
2
3  func _ready():
4    >| world = $World
5    >| agent = $Agent
6    >| communication.start_server(9090, "127.0.0.1")
7
8  func _reset():
9    >| world.reset()
10 >| agent.reset(world.sample_initial_position())
```

Демо . Оформим структуру сообщения

```
1  syntax = "proto3";
2
3  message WorldData {
4      |   bool apple_caught = 1;
5  }
6
7  message AgentData {
8      |   repeated float distances_to_obstacle = 1;
9      |   repeated float distances_to_target = 2;
10 }
11
12
13 message Message {
14     |   AgentData agent_data = 1;
15     |   WorldData world_data = 2;
16 }
```

В пару кликов по инструкции в репозитории компилируем сообщение в формат Python и Godot.

Демо . За кадром

1. Пишем `gym.Environment` в Python (наследуем `Environment` из фреймворка).
2. Пишем свой / используем из любимой библиотеки цикл обучения агента.
3. Запускаем среду в Godot.
4. Запускаем обучение.

Демо 🤖. Картинки в качестве наблюдений

```
syntax = "proto3";
```

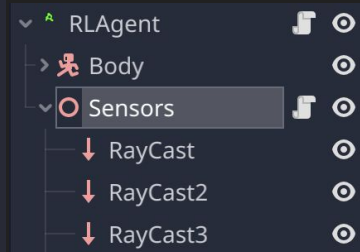
```
message CameraFrame {  
    int32 height = 1;  
    int32 width = 2;  
    string format = 3;  
    bytes data = 4;  
}
```

```
message WorldData {  
    bool apple_caught = 1;  
}
```

```
message AgentData {  
    CameraFrame camera_image = 1;  
}
```

```
message Message {  
    AgentData agent_data = 1;  
    WorldData world_data = 2;  
}
```

Меняем формат сообщения, тип сенсора и код отправки



```
func get_data(_observation_request, storage) -> void:  
    var data = sensors.get_data()  
    for distance in data["distances_to_obstacle"]:  
        storage.add_distances_to_obstacle(float(distance))  
    for distance in data["distances_to_target"]:  
        storage.add_distances_to_target(float(distance))
```



```
func get_data(_observation_request, storage) -> void:  
    var data = sensors.get_data()  
    storage.set_height(int(data.get_size().y))  
    storage.set_width(int(data.get_size().x))  
    storage.set_format("rgb")  
    storage.set_data(data.get_data())
```


Демо . Минутка лукавства

За кадром остались мелкие неизбежные правки

```
10 ▾ func move_body():
11 >| var direction = Vector3.ZERO
12 >|
13 ▾ >| if Input.is_action_pressed("ui_right"):
14 >| >| direction.x -= 1
15 ▾ >| elif Input.is_action_pressed("ui_left"):
16 >| >| direction.x += 1
17 ▾ >| elif Input.is_action_pressed("ui_up"):
18 >| >| direction.z += 1
19 ▾ >| elif Input.is_action_pressed("ui_down"):
20 >| >| direction.z -= 1
21
22 ▾ >| if direction != Vector3.ZERO:
23 >| >| direction = direction.normalized()
24
25 >| velocity.x = direction.x * speed
26 >| velocity.z = direction.z * speed
27 >| velocity = body.move_and_slide(velocity, Vector3.UP)
```

```
38 ▾ func move_body():
39 >| var direction = Vector3.ZERO
40 >|
41 ▾ >| if current_action == 0: # MOVE_RIGHT
42 >| >| direction.x -= 1
43 ▾ >| elif current_action == 1: # MOVE_LEFT
44 >| >| direction.x += 1
45 ▾ >| elif current_action == 2: # MOVE_UP
46 >| >| direction.z += 1
47 ▾ >| elif current_action == 3: # MOVE_DOWN
48 >| >| direction.z -= 1
49
50 ▾ >| if direction != Vector3.ZERO:
51 >| >| direction = direction.normalized()
52
53 >| velocity.x = direction.x * speed
54 >| velocity.z = direction.z * speed
55 >| velocity = body.move_and_slide(velocity, Vector3.UP)
```

Дальнейшее развитие проекта

- Расширение встроенной во фреймворк конфигурации
 - Сделать внутриигровое время быстрее реального
- Улучшению функционала по мере предложений пользователей
- Shared memory
 - Обеспечивает наименьшее время передачи данных
 - Работает на Windows, Linux, MacOS
- Добавить поддержку Godot 4

Заключение

Godot gym API - первый фреймворк, позволяющий обучать RL-агентов в средах из Godot на любых данных.

Фреймворк стремится минимизировать усилия на оформление передачи данных, высвобождая ресурсы на разработку самой среды или алгоритма.

Буду рад видеть ваши звездочки на репозитории и предложения по улучшению в issues.



Репозиторий



Канал с анонсами



Мой Telegram

Спасибо за внимание

Константин Черняк
chernyakonstantin@gmail.com