Machine Learning Agents

Реализация интеллектуальных агентов для применения в игре жанра гонки на самолётах

Выпускная работа
Дюдюкина Кирилла

Научный руководитель
Ксемидов Борис

Цели и задачи

- Создание игры – гонки на самолётах
  - Необходимо облететь большее чем оппонент количество случайно генерируемых в начале сцены контрольных точек за отведённое время
- Создание и тренировка интеллектуального агента с применением машинного обучения
  - Агент должен быть способен управлять самолётам и выполнять поставленные игровые задачи, собирать как можно больше шаров
  - Данный агент будет реализован с помощью глубокого обучения с подкреплением на игровом движке Unity и с помощью инструментария Unity ML-Agents
  - Unity был выбран для данного проекта из-за высокого абстрактного уровня, что позволяет писать логику игры и агента, а не реализовывать движок с нуля

Аналоги агентов машинного обучения

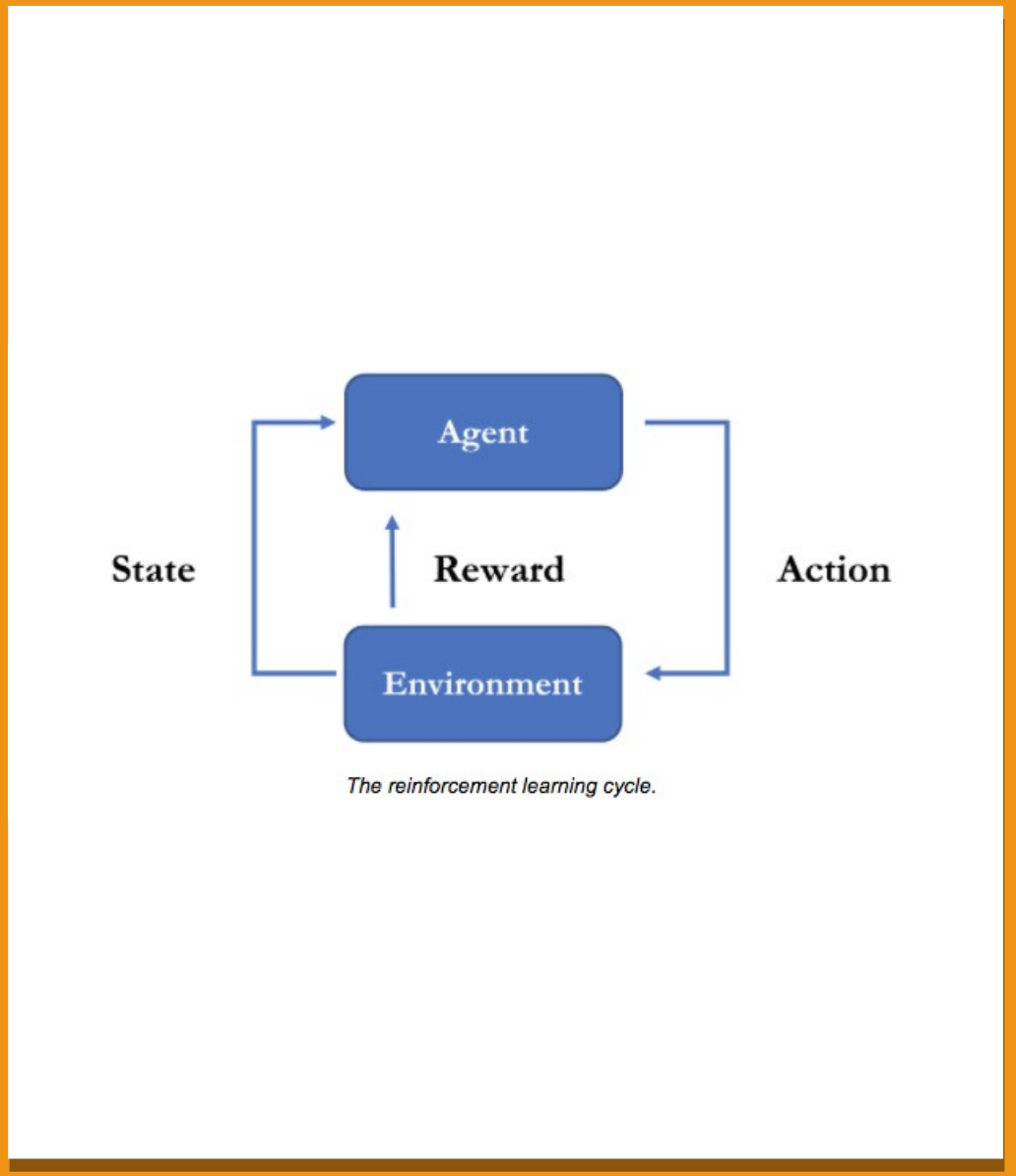
- Аналогом интеллектуального агента, реализованного с помощью машинного обучения, является программируемый разработчиком неигровой персонаж (NPC)

```
List<Signal> signals =
Player.getSignals(SignalHandler.REACTION_REQUIRED);

if(signals.size() > 0){
        AIHandler.sortPriority(signals);
        this.react(signals[0]);
} else {
        this.say(getVoiceLines()[Math.random(VOICE_LINES)]);
}
```

- NPC, который может ходить, избегать объектов и пытаться стрелять в вас, просто выполняет множество функций. Какая именно информация используется в этом расчете, и какие именно действия выполняются зависит от игры и от заданных разработчиком параметров

Агенты машинного обучения

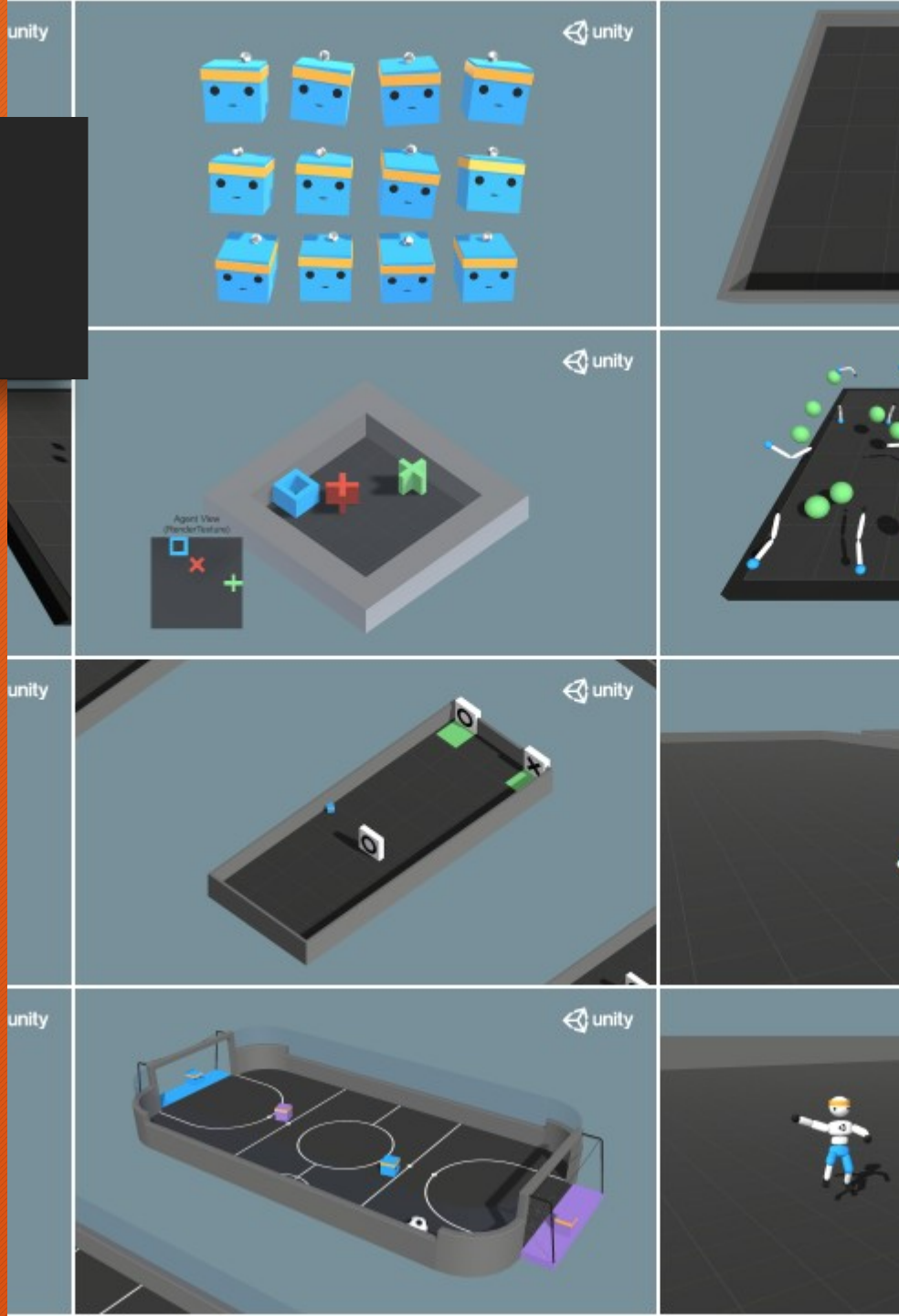
- NPC агенты машинного обучения не ограничены прямой зависимостью получаемой информации и выполняемых действий, заданных заранее. Мы можем обучить агента эффективному выполнению действий. Это достигается путем моделирования среды для множества симуляций, когда агент со временем узнает, какое оптимальное действие следует предпринять для каждого наблюдения, которое он измеряет, максимизируя свою будущую награду.



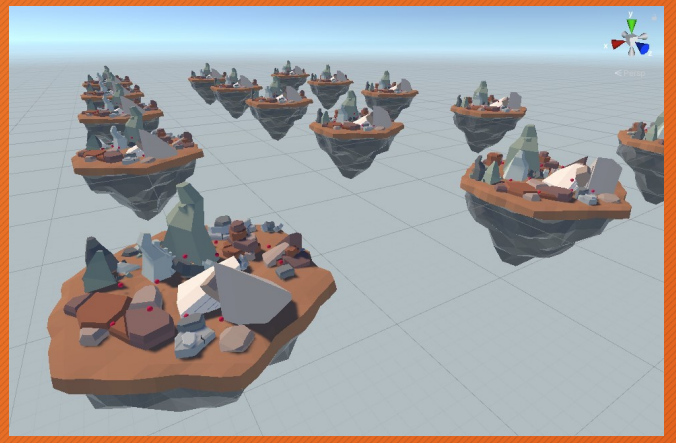
The reinforcement learning cycle.

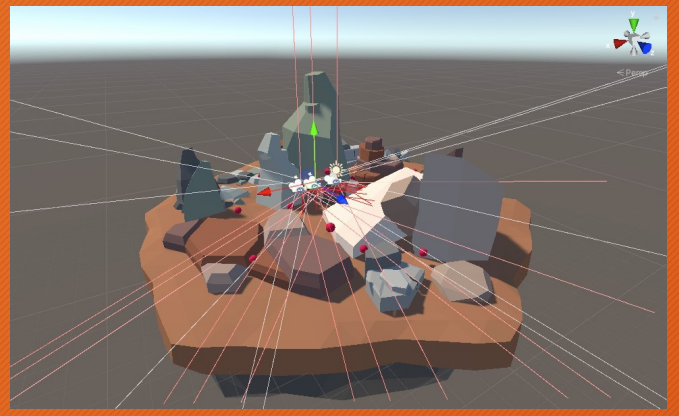
Интеллектуальные агенты в Unity

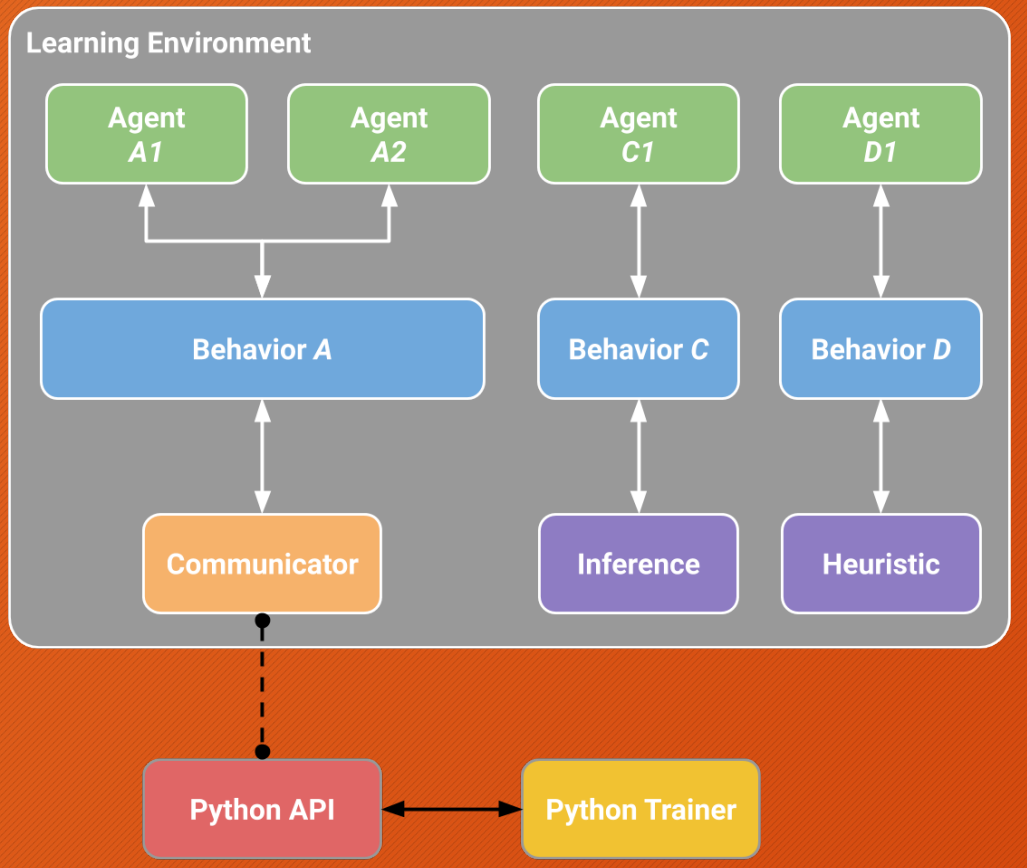
- Unity ML-Agents - open-source проект предоставляющий инструменты для внедрения интеллектуальных агентов в различные игровые проекты
- Unity ML-Agents позволяет избавиться от необходимости программирования поведения агента благодаря обучению интеллектуальных агентов с помощью глубокого обучения с подкреплением или имитационного обучения
- Эффективность интеллектуального агента зависит как от выбранного метода обучения и длительности, так и от корректности заданных алгоритмов взаимодействия агента с окружающей средой и его мотиваторов



Реализация интеллектуальных агентов
Среда обучения

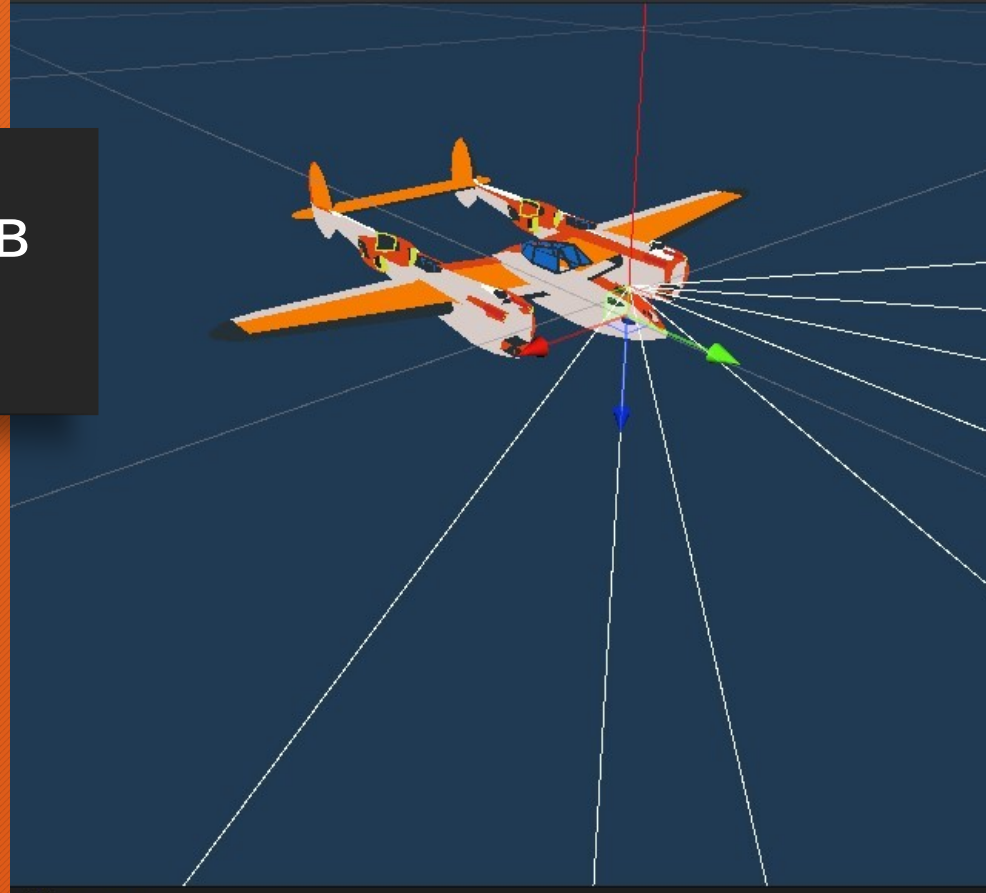






Реализация интеллектуальных агентов Агент, наблюдения и действия

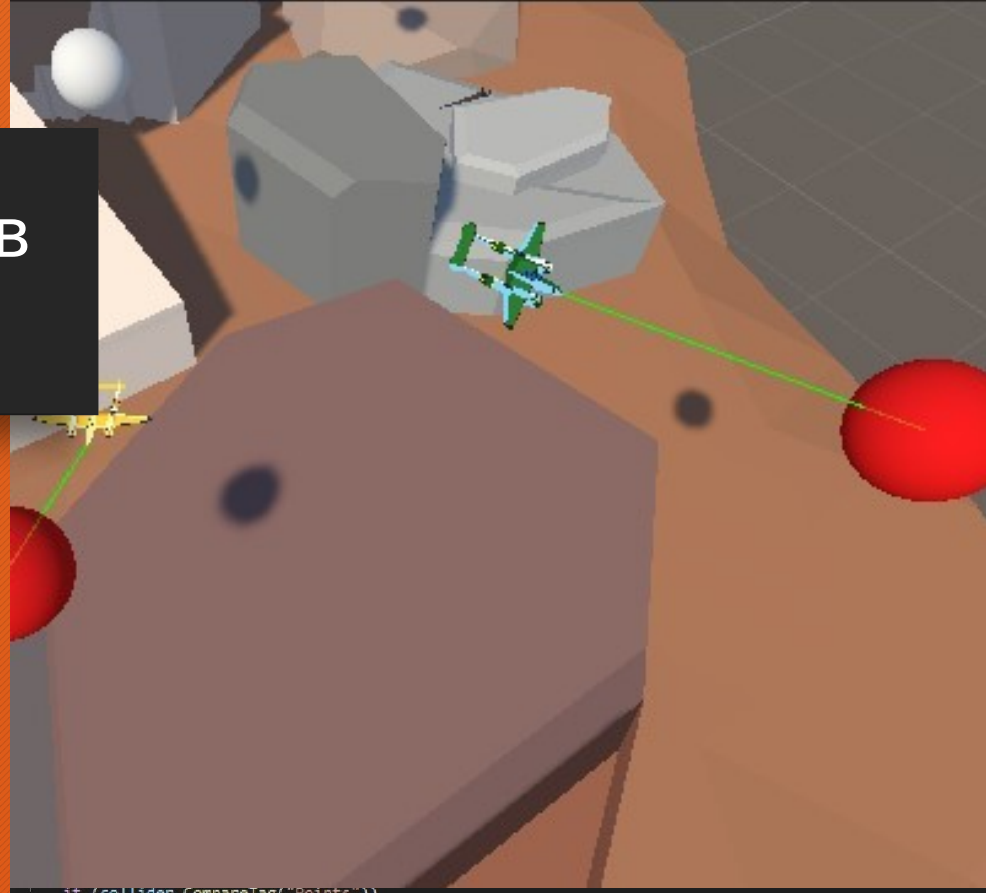
- Агент — это объект, который может наблюдать за средой, выбирать наилучший план действий, используя эти наблюдения, и выполнять действия в окружающей среде.
- Наблюдения собираются сенсорами, добавленными к GameObject агента и в скрипт агента. В проекте реализованы векторные наблюдения положения объекта на сцене и относительно шаров, а также наблюдения в виде добавленных сенсоров-лучей.
- Код агента должен выполнять действие, например, перемещать агента в ту или иную сторону. Параметр действия, передаваемый агенту, представляет собой массив чисел с плавающей точкой. Важно отметить, что алгоритм обучения не знает, что именно означают значения из массива. Алгоритм обучения просто пробует разные значения из списка действий и наблюдает за их влиянием на награды с течением времени и во многих тренировочных эпизодах.



```
/// <summary>
/// Called when action is received from either the player input or the neural network
///
/// </summary>
/// <param name="vectorAction">The actions to take</param>
ссылка: 1
public override void OnActionReceived(float[] vectorAction)
{
    // Don't take actions if frosen
    if (frozen) return;

    rigidbody.AddRelativeForce(0f, 0f, (moveForce * 0.5f));
    float move_forward_backward = Mathf.Clamp(vectorAction[0], -1f, 1f);
    if (move_forward_backward >= 0)
    {
        rigidbody.AddRelativeForce(0f, 0f, move_forward_backward * moveForce);
        rigidbody.AddForce(0f, 0.95f, 0f);
    }
    if (move_forward_backward < 0)
    {
        rigidbody.AddRelativeForce(0f, 0f, move_forward_backward * moveForce * 0.3f);
        rigidbody.AddForce(0f, (0.9f + 0.5f * move_forward_backward), 0f);
    }
```

Реализация интеллектуальных агентов Агент, награды

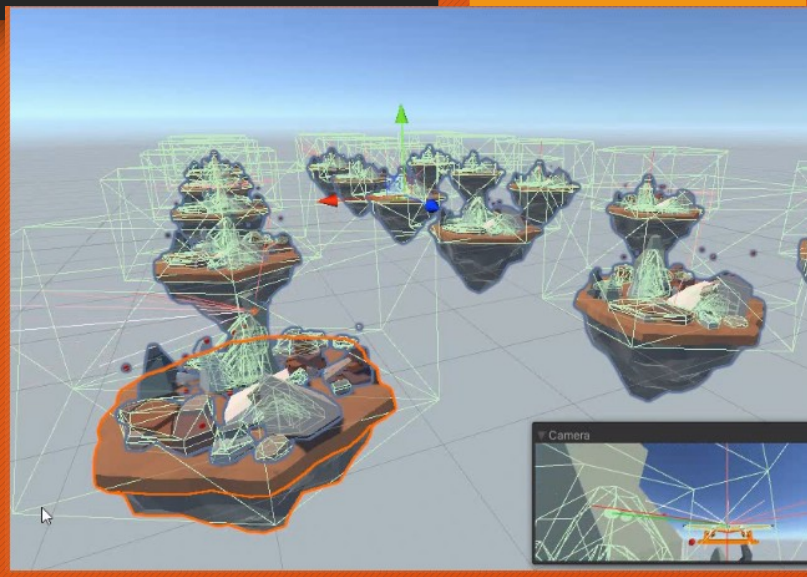
- В обучении с подкреплением награда — это сигнал о том, что агент сделал что-то правильно.
- Применяются как награды за ожидаемое поведение, так и штрафы за некорректное. В рамках проекта агент получает награду за сбор шара и при совпадении вектора направления самолёта и вектора от самолёта до шара. Также агент получает штраф за выход из игровой области сцены.

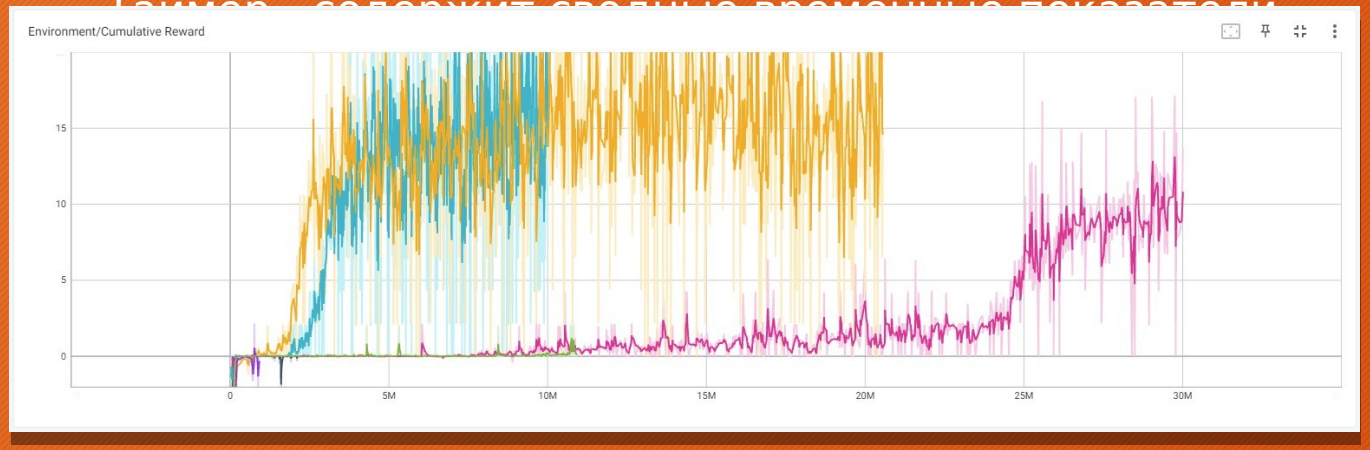


```
if (collider.CompareTag("Points"))
{
    Vector3 closestPointToAirplaneNose = collider.ClosestPoint(airplaneNose.position);

    // Check if the closest collision point is close to the airplane nose
    // Note: a collision with anything but the airplane nose should not count
    if (Vector3.Distance(airplaneNose.position, closestPointToAirplaneNose) < AirplaneNoseRadius)
    {
        // Look up the Checkpoint for this Points collider
        Checkpoint Checkpoint = CheckpointArea.GetCheckpointFromPoints(collider);

        // Attempt to take Points
        float PointsRecieved = Checkpoint.GainPoints(1f);

        // Keep track of Points obtained
        PointsObtained += PointsRecieved;

        if (trainingMode)
        {
            //Calculate reward for getting Points
            float bonus = 0.2f * Mathf.Clamp01(Vector3.Dot(transform.forward.normalized, (nearestCh
            AddReward(2f + bonus);
        }

        // If Checkpoint is empty, update the nearest Checkpoint
        if (!Checkpoint.HasPoints)
```

Реализация интеллектуальных агентов Процесс обучения

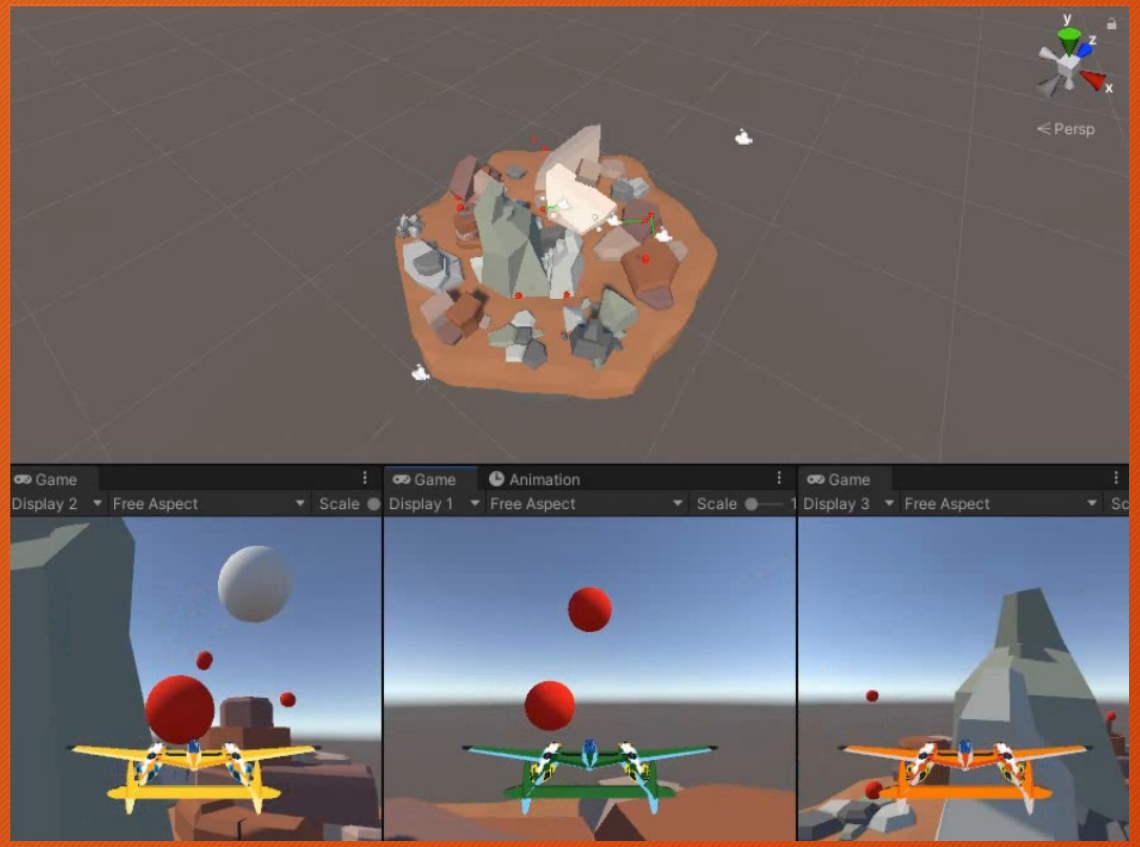
- После того, как среда обучения создана и агент настроен, следующим шагом будет запуск тренировки.
- Процесс обучения всегда будет генерировать три артефакта:
  - Метрика - для наблюдения используется TensorBoard
  - Модель - применяется для обученной модели поведения
  - Таймер - содержит сводные временные показатели

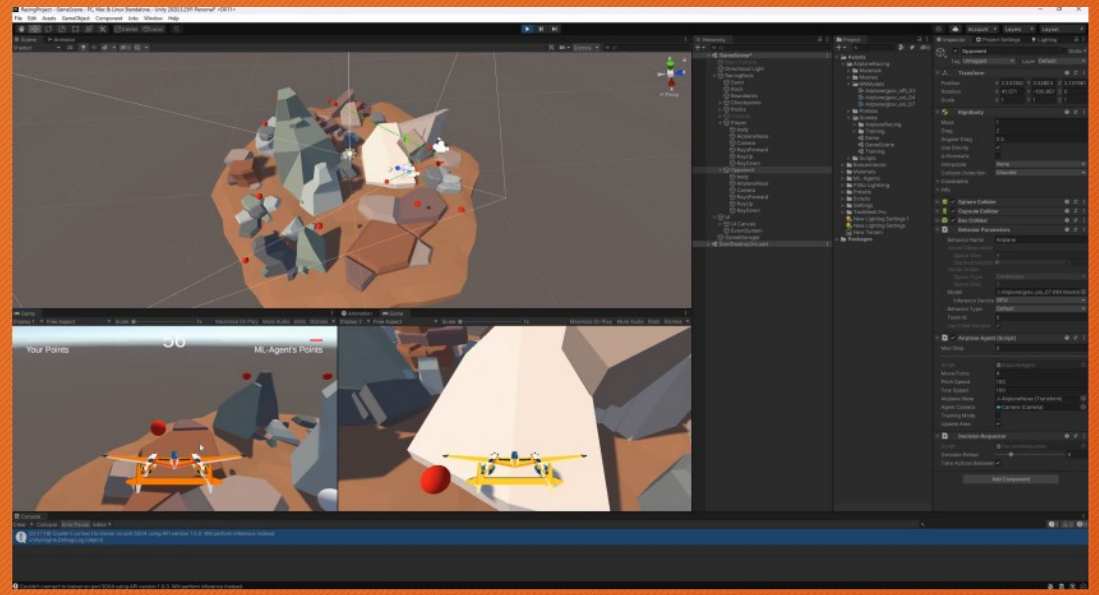




Реализация интеллектуальных агентов Применение агента

- После завершения процесса обучения и сохранения модели её можно добавить в проект Unity и использовать с совместимыми агентами (агентами, сгенерировавшими модель).
- При изменении алгоритмов поведения агентов, а именно действий и наблюдений агента, потребуется переобучение агента.
- При изменении окружающей агента среды, обученный агент будет функционировать корректно, если эти изменения не затрагивают действия и наблюдения агента.




Your Points
ML-Agent's Points

Результаты работы

- В данном проекте были продемонстрированы возможности платформы Unity ML-Agents, путём успешной реализации агента машинного обучения и применения его в игровой сцене.

- Применение инструментария Unity ML-Agents позволило ускорить разработку поведения НПС, по сравнению с классическим кодированием поведения.

- Созданный агент способен успешно функционировать в окружающей среде, отличной от среды обучения, при сохранении алгоритмов действий и наблюдений агента.