Unity Machine Learning Agents ではじめる強化学習

株式会社インフィニットループ
技術研究グループ
波多野 信広

Twitter: @nobuhatano
数学勉強会@札幌

infinite loop
インフィニットループ

本セッションの内容

- 自己紹介
- どんな人が何を目的に Unity で機械学習するのか
- 強化学習と Unity Machine Learning Agents 概要
- Ver 0.3 のインストールからビルド、実行まで
- 状態と報酬でどのように学習を導くか

自己紹介

- 元商用DBサーバーのサポートエンジニア
- （株）インフィニットループ
  - インフラエンジニア
    - クラウド構築運用 & MySQL 担当
  - 技術研究グループ テクニカルディレクター
    - インフラしながら技術調査や研究など

Data Analytics, Machine Learning 履歴書

- 2015 : MySQL クラウド向け InnoDB チューニング
- 2017 : Jupyter Notebook と MySQL でゼロからはじめるデータサイエンス
- 2018 : インフィニットループ技術ブログ： 入門 Keras (1) - (7)
- 2018 : Unity Machine Learning ではじめる強化学習

どんな人が何を目的に
Unity で機械学習をするのか

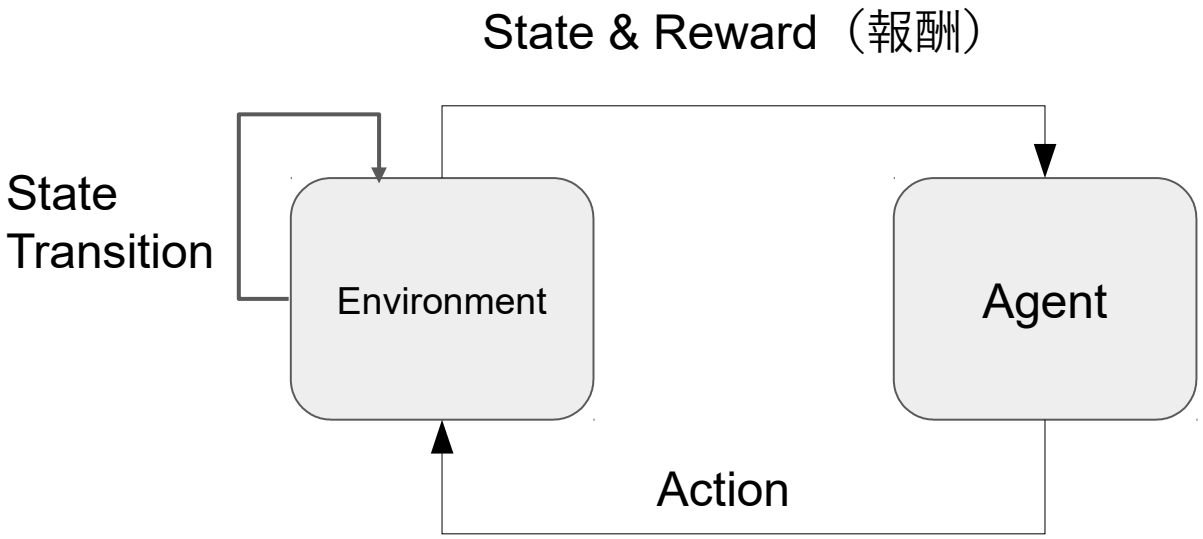
Unity Machine Learning Agents の目的

- ゲーム開発者
  - Machine Learning をゲームに活用
- アーティスト
  - プロシージャルコンテンツ生成
  - インタラクティブアート等で AI を活用
- コンピューターサイエンス、機械学習研究者
  - AI 研究、ロボットや自動運転
  - シミュレータープラットフォームとして
- 学生／ホビイスト
  - Unity と Machine Learning で何か楽しいことしたい

これです！

強化学習と Unity Machine Learning Agents の概要

Reinforcement Learning （強化学習）

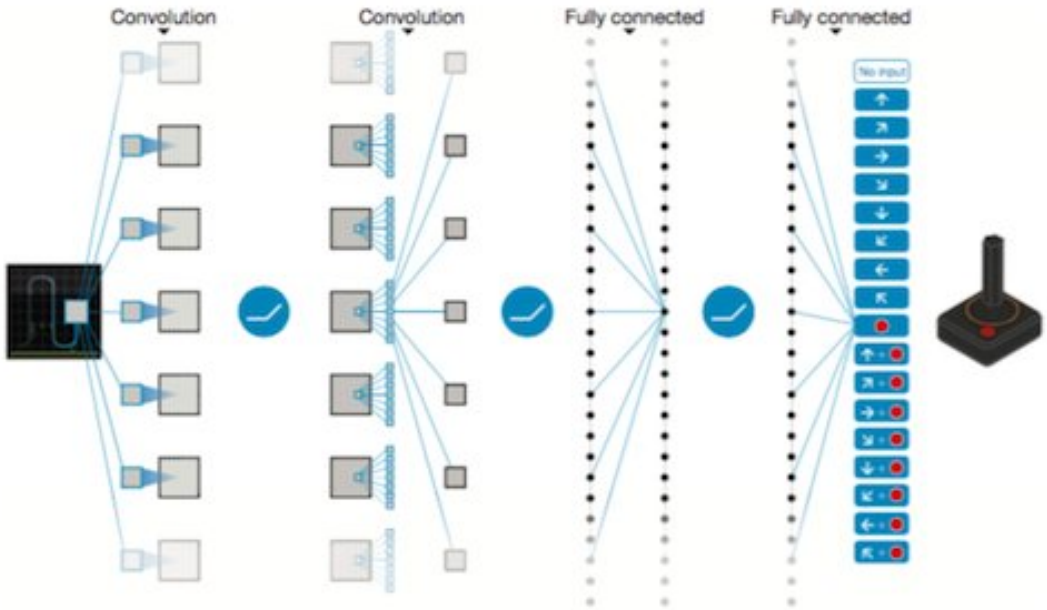


- ある状態における最も**将来の報酬**を期待できるアクションを選択
- 新しい状態に遷移し、そのときの報酬を計算
- 戦略やルールなど事前知識なしで学習可能
- Q (state, action) 関数の最適化問題
- 状態と行動は数値記号的だが、将来の報酬まで計算するには莫大な組み合わせ
- 確率的に探索など
- 無茶な計算を無理やり解く定番⇒Genetic Algorithms か Neural Network

Deep Reinforcement Learning (Deep Q-Network)

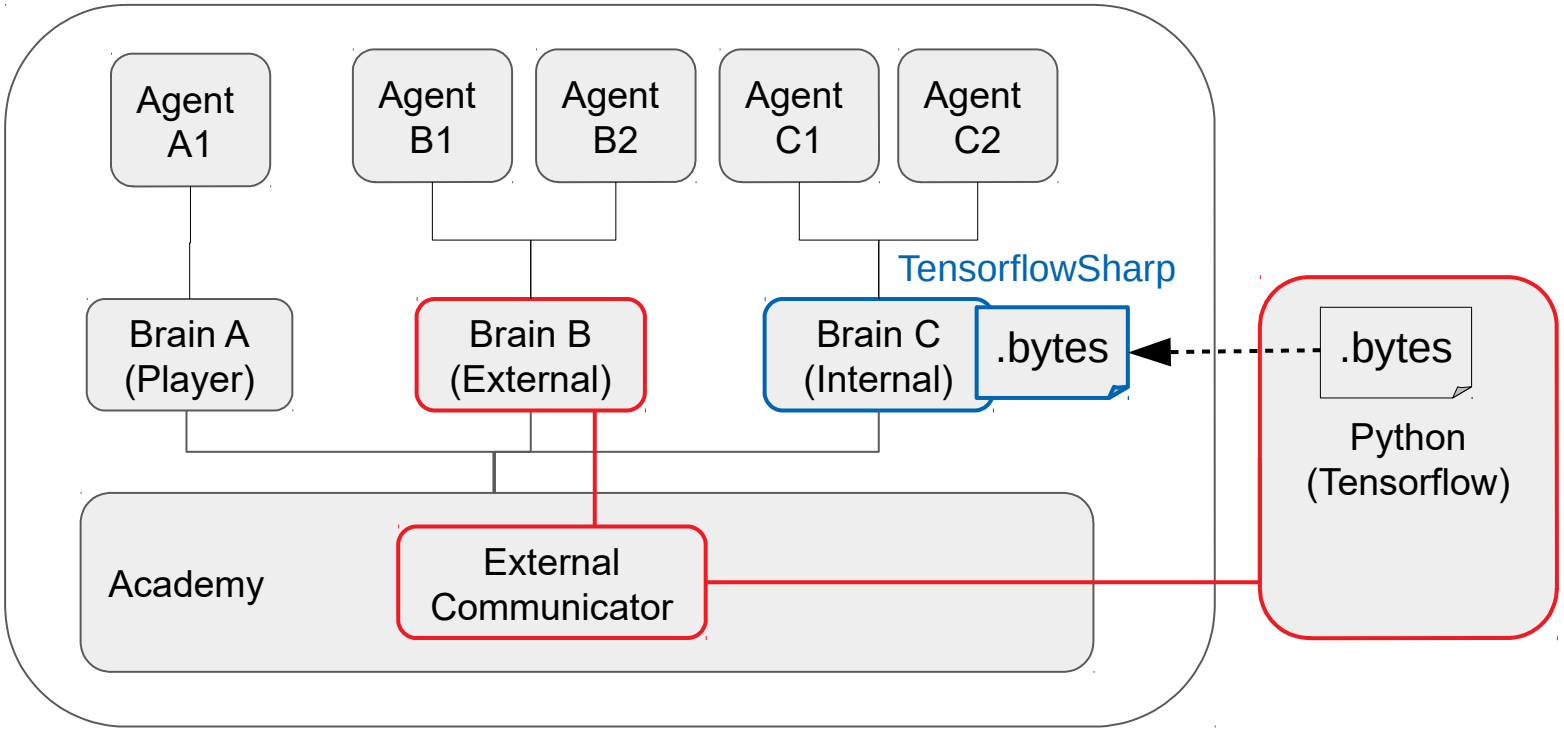
- DeepMind 社が開発 AlphaGo
- 深層ニューラルネットワーク(DL)　入力：画像⇒出力：操作
- 強化学習への DL 応用は 2008 柴田 大分大 など
- DQN は以下の手法で汎用化（パラメータチューニングだけで対応可能）
  - Experience Replay
  - Separate Target Network
  - Clipping Rewards
  - Skipping Frames

全て Unity が Tensorflow でやってくれます！



出典：https://www.nature.com/articles/nature14236

Unity Machine Learning Agents の構成



Agent（シミュレーター）
- 状態観測
- アクション実行
- 報酬計算

Brain（意思決定）
- Agent □ □
- アクション決定

Academy
- Brainを包含
- 全体設定
- 外部の Python と通信

インストールからビルド、実行まで

Unity と Tensorflow のインストール

- Unity をデフォルトでインストール （今回使っている Unity のバージョンは 2018.1.0f2）
- Miniconda (https://conda.io/miniconda.html) 64bit Python 3 系をデフォルトでインストール
- 環境名を決めて（例: UnityML）tensorflow 他をインストール
  - Anaconda Prompt を起動
    - conda create -n UnityML
    - activate UnityML
    - conda install tensorflow Pillow matplotlib numpy notebook pytest docopt
    - pip install pyyaml
    - 参照：https://github.com/Unity-Technologies/ml-agents/blob/master/python/requirements.txt

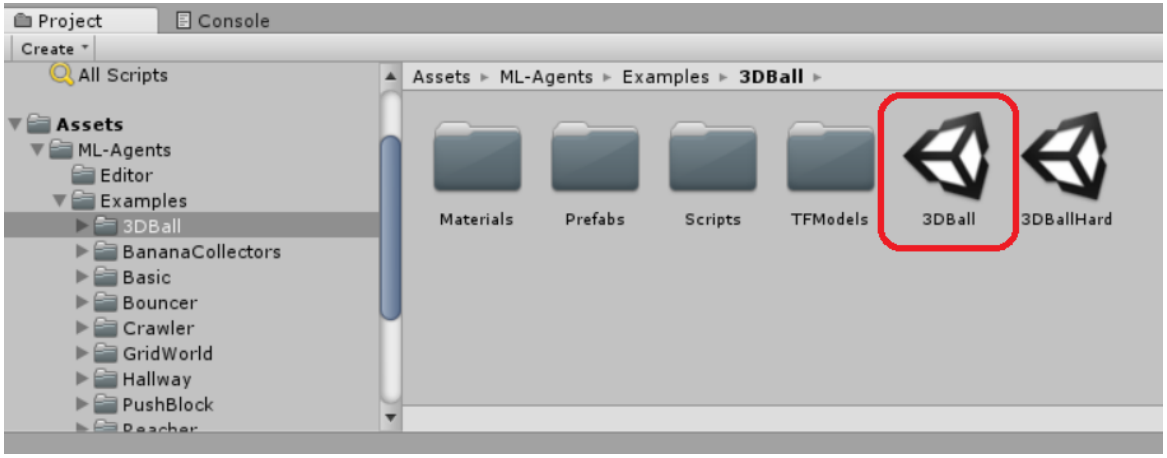
- Tensorflow動作確認
    - jupyter-notebook コマンドでブラウザに起動
    - New で新規 Python3 ページを作成

```
In [5]:
1 import tensorflow as tf
2 print(tf.Session().run(tf.constant('Hello TensorFlow!'))) # 定数でハローを print

b'Hello TensorFlow!'
```

ML-Agents (ver 0.3) のインストール

- github から ml-agents を clone し、Tensorflowsharp をダウンロードしておく
  - git clone git@github.com:Unity-Technologies/ml-agents.git
  - https://s3.amazonaws.com/unity-ml-agents/0.3/TFSharpPlugin.unitypackage
- Unity を起動し clone 先の ml-agents\unity-environment を Open
- TFSharpPlugin.unitypackage をダブルクリックして Project に追加
- Project ウィンドウの Asset/ML-Agents/Examples に各サンプル
- 試したいシーンファイルをクリック
- デフォルトは Brain (Player) になっていて↑↓←→で操作できます

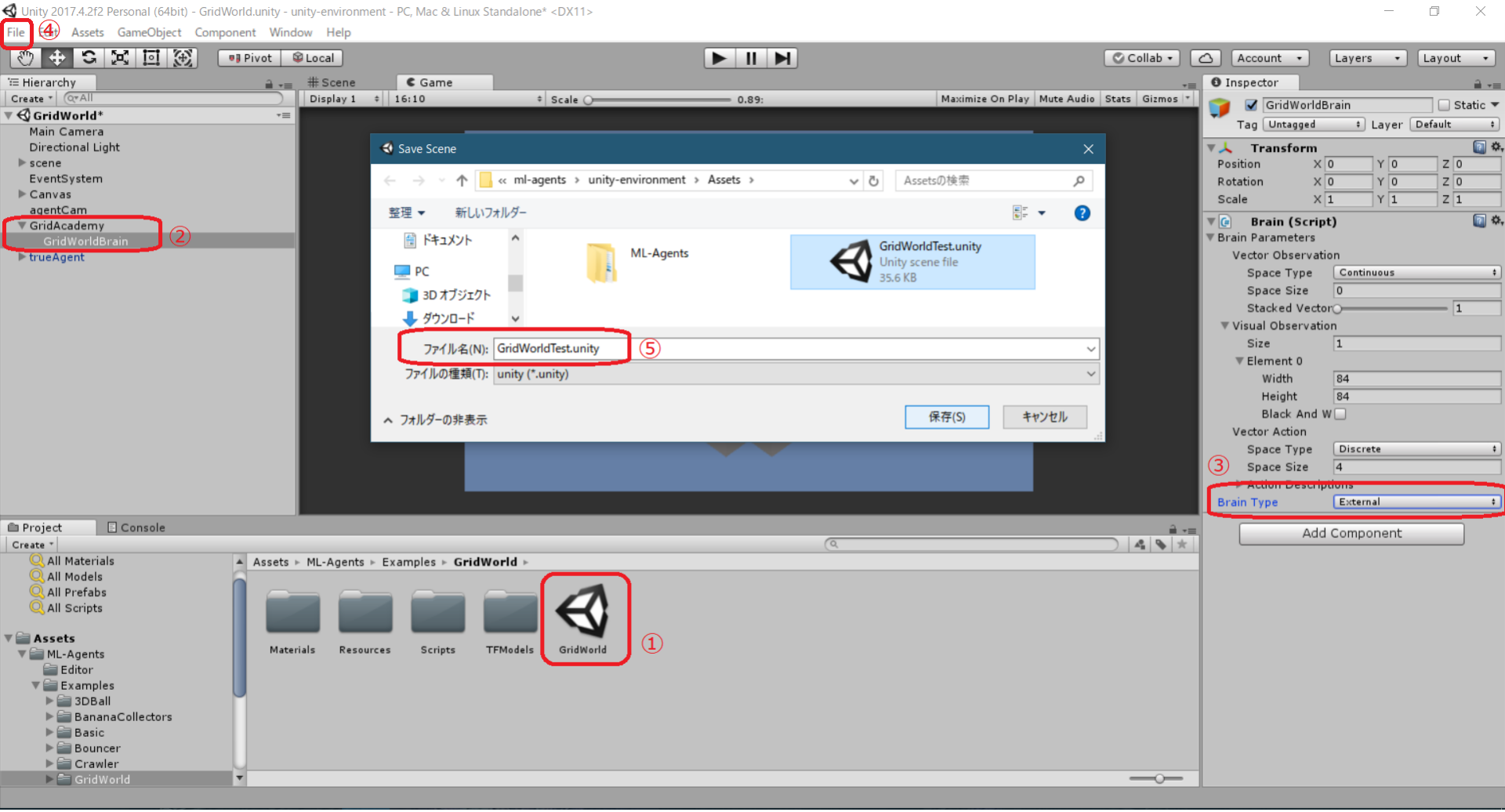


インストールから 3DBall ビルドのデモ

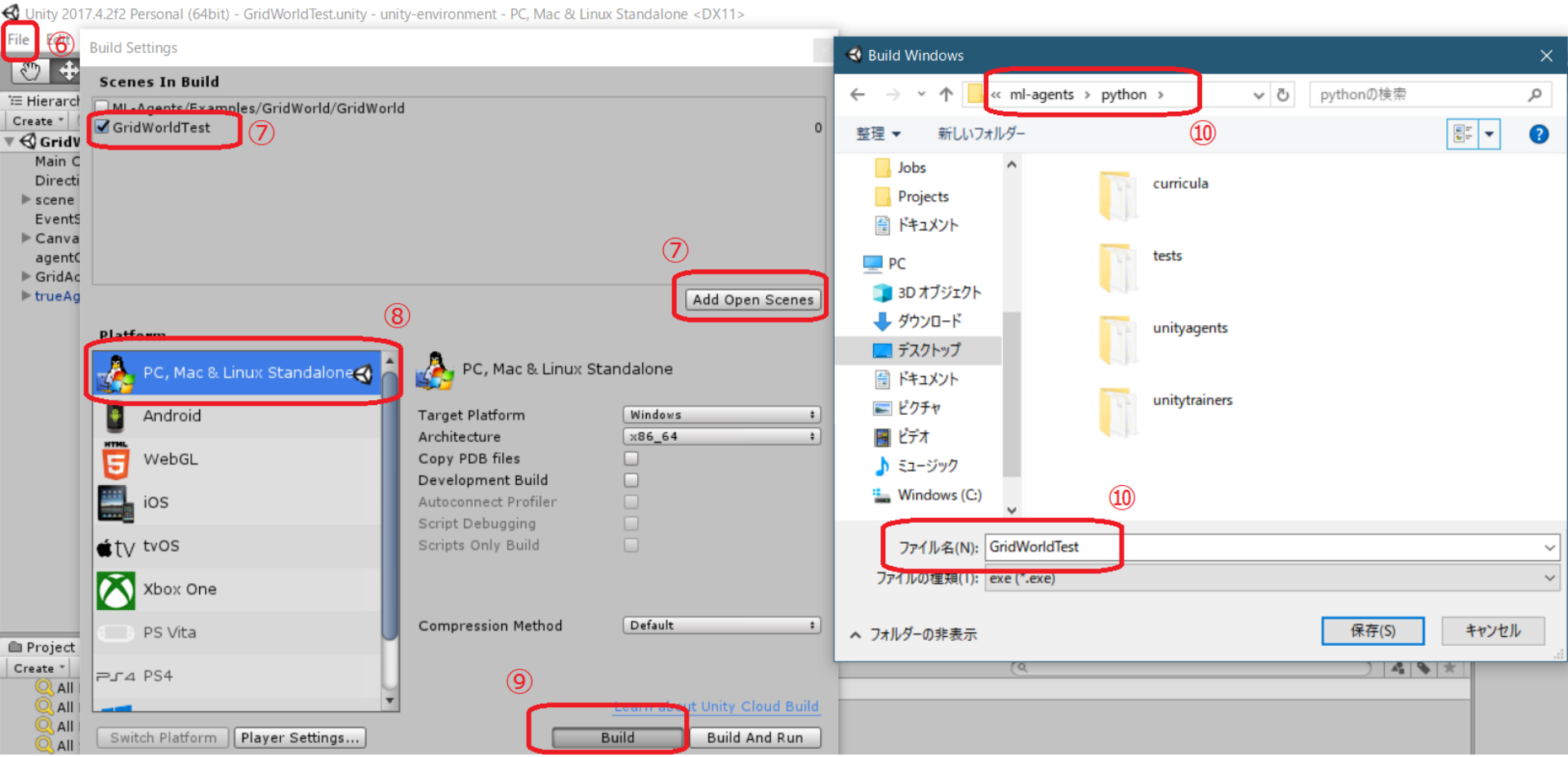
サンプルを動かすまでの流れ（資料ページ）

1. ml-agents\unity-environment\Assets\ML-Agents\Examples\GridWorld シーンを開く
2. Hierachy で GridAcademy\GridWorldBrain を選択
3. Inspector で Brain Type を External に変更
4. File メニューで Save Scene as を選択
5. ml-agents\unity-environment\Assets に名前を付けてシーンを保存 (例：GridWorldTest)
6. File メニューで Build Setting を選択
7. Add Open Scenes し、不要なシーンのチェックは外して、Scenes In Build に GridWorldTest だけが含まれるようにする
8. Target Platform に StandAlone を選んで
9. Build
10. (2017まで）GridWorldTest.exe ファイルの置き場所は ml-agents\python へ
11. (2018から）Build 時はフォルダ選択のみで生成される exe はそのフォルダ名と同じ名前でフォルダ内に生成されます。運用としては ml-agents\python\GridWorld のように python 直下に Build したい名前のフォルダを作ってから Build しましょう

Brain を External に変更しシーンを保存（資料ページ）



Build する（資料ページ）



Python から learn.py と exe を実行

Anaconda Prompt で

- activate UnityML （tensorflow をインストールした環境名、例 UnityML）
- cd [git clone した場所]\ml-agents\python\ビルドディレクトリ
- python ..\learn.py ビルドしたexe --オプション

--train が学習させるオプション、つけない場合は Brain (Internal) と同じで推論のみ

--run-id= で学習済みのモデルを保管&識別する名前を指定。デフォルトは ppo

--load モデルを初期化せず --run-id で指定のモデルに学習を追加

--slow をつけると実際の実行時間で観察可能、デフォルトは表示フレームをスキップした加速学習

--save-freq= モデルを保存するステップ （デフォルト 50000）

ml-agents\python\trainer_config.yml で設定された max_steps は超えられないので長く学習させるには設定変更が必要（例）サンプルの Tennis は trainer_config.yml に記載されていない。Tennis の Unity Editor 上のオブジェクト名は AgentBrain なので

AgentBrain:
    max_steps: 5.0e5 （例：50万ステップ）

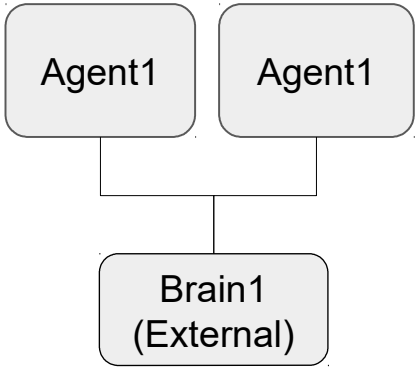
と記述を追加すると50万ステップまでの学習が可能

3DBall 実行と学習のデモ

状態と報酬でどのように学習を導くか

3DBall : Simultaneous Single-Agent

- 1個の Brain に独立したアクションと報酬を受ける多数の同じ Agent
- 複数同時にトレーニングを実行する。ロボットのトレーニング。学習の加速に
- 落下してくるボールを受けて落とさない



状態ベクトル Vector Observation

- Z軸（正面）角度
- X軸（横）角度
- ボールの板上座標
- ボールの板上速度
- 過去3状態

報酬 (Reward)

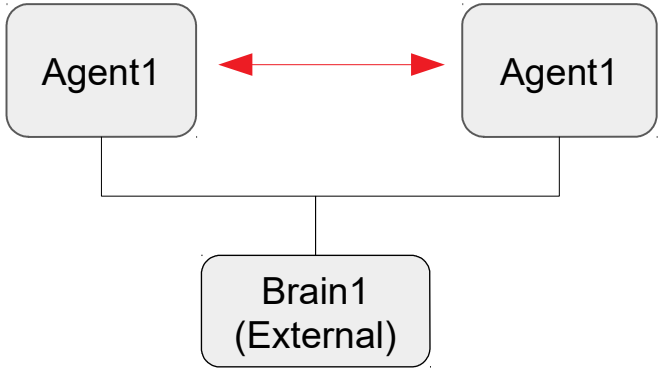
- 処理フレーム毎 +0.1
- 落ちたら -1

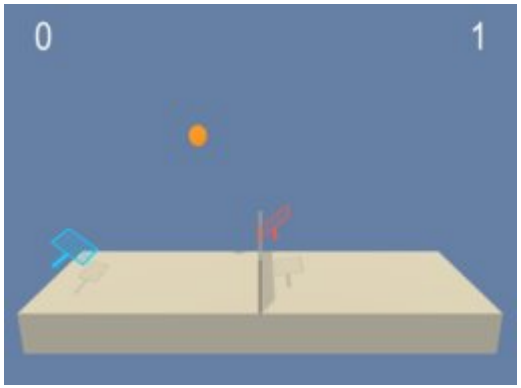
Tennis : Adversarial Self-Play（敵対的自己プレイ）

- 1個の Brain に逆の報酬をうける同じ Agent ２つ
- 互いに対戦して学習する。AlphaGo もこの戦略。

状態ベクトル (Vector Observation)
- ラケットの位置と速度
- ボールの位置と速度
- 過去3状態

報酬 (Reward) 打った側視点で
- ネット上空を通過 +0.1
- 壁に当たって失点 -0.01
- 床に落ちて失点 -0.01
- ネットに当たって失点 -0.01

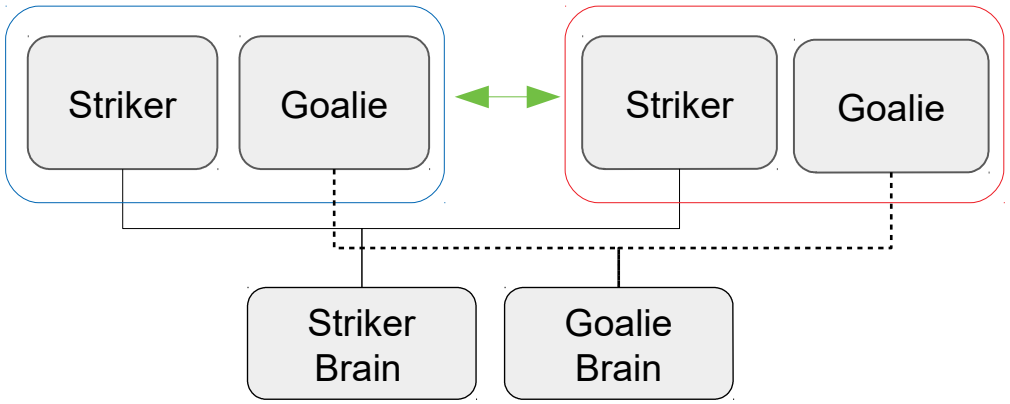




Tennis デモ
初回と 50万ステップ後

SoccerTwos : Competitive Multi-Agent（複数 Agent の対戦）

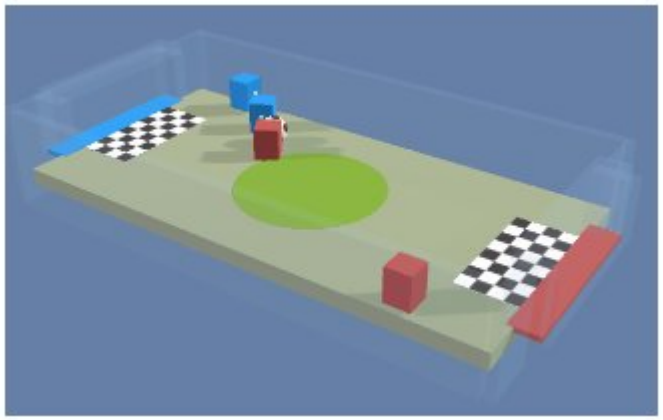
- 複数の Brain に接続した複数タイプの Agent
- Agent は観測範囲が違ったり、可能なアクション、報酬などが違っている
- 相互に逆の報酬。チームスポーツ



状態ベクトル (Vector Observation)
- ボール、壁、ゴール、自軍、敵軍の位置
- 過去3状態

アクション
- 前後左右
- ストライカーは回転

報酬 (Reward)
- 得点ストライカー +1
- 失点ストライカーとキーパー -1
- キーパーであることに +1/3000
- ストライカーであることに -1/3000



SoccerTows デモ
初回と 80万ステップ後

Banana Collector : Selfish Agents Cooperate

- 1Brain に自由な 5 Agent（4面並列）
- 紫のバナナを避け、黄色いバナナを多く取る
- レーザーを当てられると麻痺する
- 床（TeachingArea) の 属性に Num Bananas と Num Bad Bananas
- バナナの数に応じて、対立するか協調するかの解が変化するのを観察する実験
  https://deepmind.com/blog/understanding-agent-cooperation/

状態ベクトル (Vector Observation)
- Agent の前方7方向の光線測定
- Agent の速度
- Agent が麻痺／レーザー発射
- 過去3状態

アクション
- y軸（上から）回転
- 前進
- レーザー発射

報酬 (Reward)
- 黄バナナ +1
- 紫バナナ -1

Banana Collector
(Reinforcement Learning 版)
初回と5万ステップ後

強化学習の利点と課題

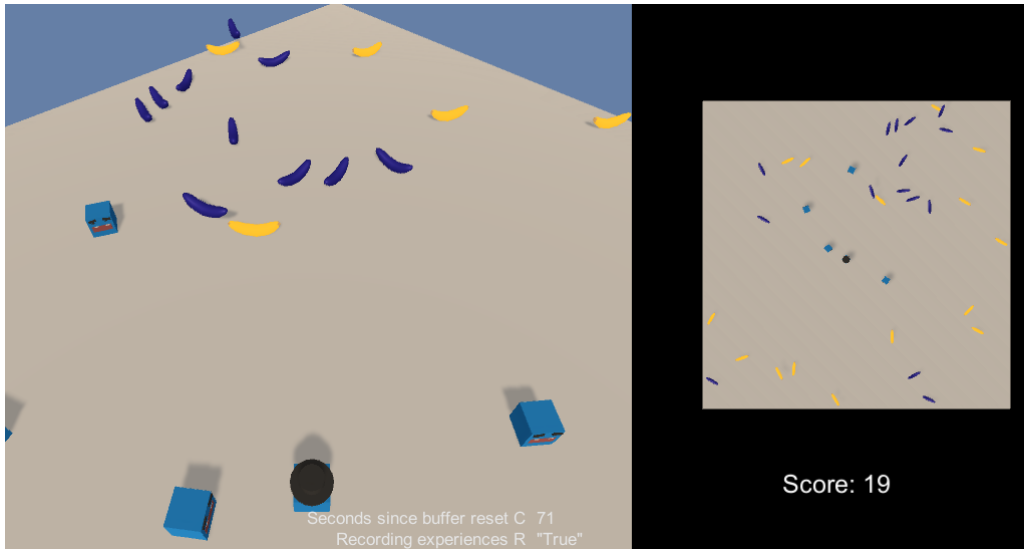
- 利点：未知の戦略の発見も期待出来る
  - AlphaGo、OpenAI での Attari ゲーム解法
  - 例：SoccerTwos での「ボールを壁とはさんでディフェンス」する現象
- 課題：パラメータ・チューニングが必要
  - プレイしながらパラメーターチューニング出来ない
  - パラメータ・チューニングを AI にさせる？そのチューニング AI のチューニングは誰がする？
- 課題：学習をリアルタイムに活用出来る？
  - 突然飛躍的に上手くなる⇒学習過程というより最適化
  - 同じ環境で特色をもった別モデルを作るのが難しい（例：特色のあるストライカー）
- 課題：特にゲームで必要な AI に使えるか？
  - 人や動物に近い挙動が欲しい
  - 個性や多様な学習度が必要

Imitation Learning

- ML-agents ver 0.3 の新機能
- 強化学習と似ているが報酬パラメータが無い
- Behavioral Cloning
  - 強化学習の環境で、教師の操作を学習
  - 画像認識や従来の機械学習の「教師あり学習」と類似
  - 特色を与えるのが簡単
  - 模倣は強化学習と比べて学習が高速
  - マネ以上のことは出来ない

Banana Collector (Imitation Learning 版)

- 1Brain に教師 1 Agent, 生徒 4 Agent（1面のみ）
- 教師の行動をまねることでモデルを学習する
- 学習後のモデル (.bytes) を Unity 内部の Tensorflowsharp で使う



状態ベクトル (Vector Observation)
- Agent の前方7方向の光線測定
- Agent の速度
- Agent が麻痺／レーザー発射
- 過去3状態

アクション
- y軸（上から）回転
- 前進
- レーザー発射

報酬 (Reward)
- なし

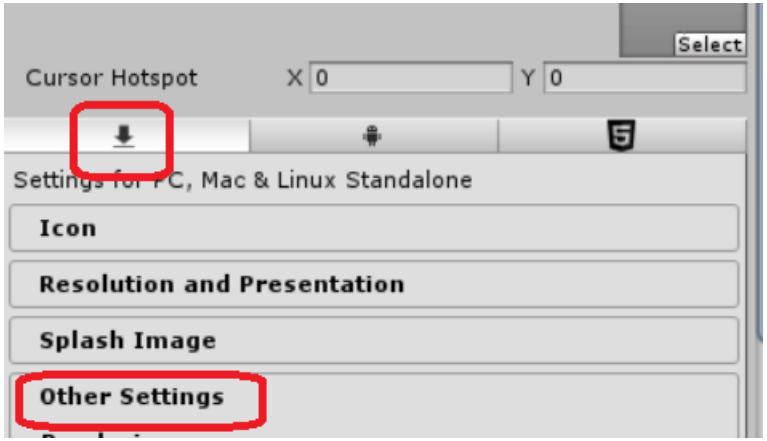
サンプルのままだとあまりに 3D 酔いするので
少し上からの視点に改造してあります

Brain (Internal) 設定

Edit / Project Settings / Player /

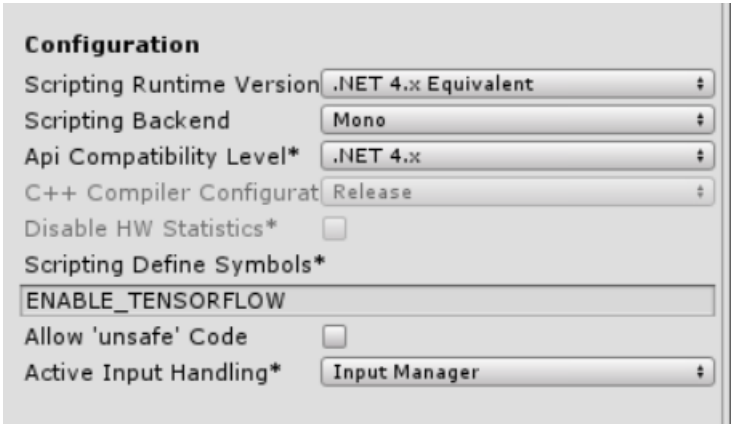
PC Mac & Linux Standalone settings /

Other Settings



Configuration
　Scripting Runtime Version
　　.NET 4.x Equivalent

　Scripting Define Symbols
　　ENABLE_TENSORFLOW

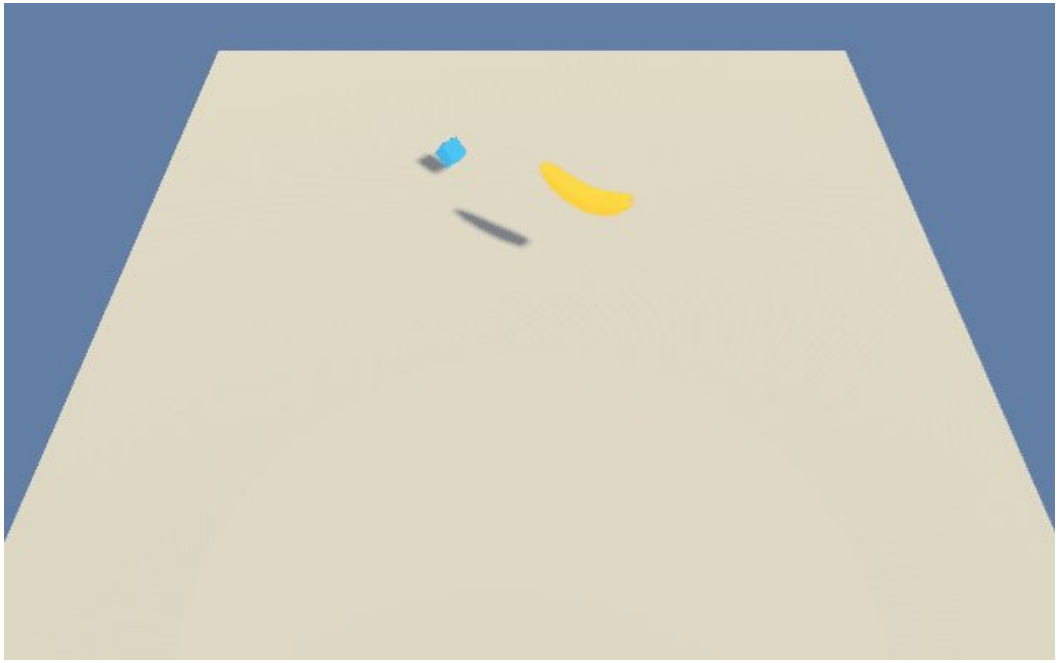


Inspector で Brain の Graph Model に
学習した .bytes ファイルをドラッグする

Banana Collector Imitation Learning 版
初回の学習から
Brain (Internal) での実行デモ

Bouncer : On Demand Decision Making

- 1ゲーム上限20回で常時跳ねている
- 着地時にだけ任意の方向と強さで力を加えてジャンプの方向を変える
- 空中（状態観察と学習のみでアクションしていない時間帯）でバナナを採る
- アクション時のステップ（フレーム）では +1/-1 のゴール報酬は発生しない



状態ベクトル (Vector Observation)
- 自分の位置とバナナの位置
- x,y,z 軸方向にかかる力

アクション
- ジャンプに加える力の方向と強さ

報酬 (Reward)
- x,y,z 方向の力の2乗 * -0.05
  - 力を使う程ペナルティ
- ジャンプ毎にもペナルティ
- バナナを採ると +1
- 外に飛び出すと -1

Bouncer
初回と25万ステップ後

まとめ

- ml-agents で最新の学習手法を Unity でも利用可能に
- ML/AI 実験で必要とされる 3D 箱庭世界を、Unity によって低コストで実装できる
- Unity Machine Learning Agents 是非遊んでみてください！