

Installation manual AirplaneAI

Research Project Jarne Demoen 2022-2023

Content table

Contents

Intro2

Technologies.....2

Creating the flight simulator in Unity3

Training your own AI using ML-Agents4

 Creating your own virtual environment4

Reward system7

IntelliSense Issues?8

Intro

For my Research Project I made an airplane AI. It's a Unity project containing a scene where the airplane takes off and flies towards a random sphere in the scene.

Technologies

The first thing that came to mind was to use the **"Unity" game engine** for my research project. This engine uses the **C#** language, which I already learnt at school. I have some experience regarding this game engine, I even have some "Udemy" courses explaining everything about the game engine so my choice was made quickly.

Unity is a cross-platform game engine and development environment for creating 2D, 3D, AR, and VR games and experiences. It was first released in 2005 and has since become one of the most popular game engines in the industry. Unity supports a wide range of platforms, including Windows, Mac, Linux, iOS, Android, and various VR and AR devices. It also has a large and active community of developers, who have created a wide range of tools and assets that can be used to speed up game development.

Additionally, Unity has a built-in visual editor, scripting API, and physics engine, making it a powerful tool for creating interactive experiences.

To install Unity, you need to download **Unity Hub**: <https://unity.com/download> . In Unity Hub you can manage all of your Unity projects and your **Unity Editor installs**. And you can also add Unity projects that you downloaded online. Keep in mind that your system needs to have some basic requirements:

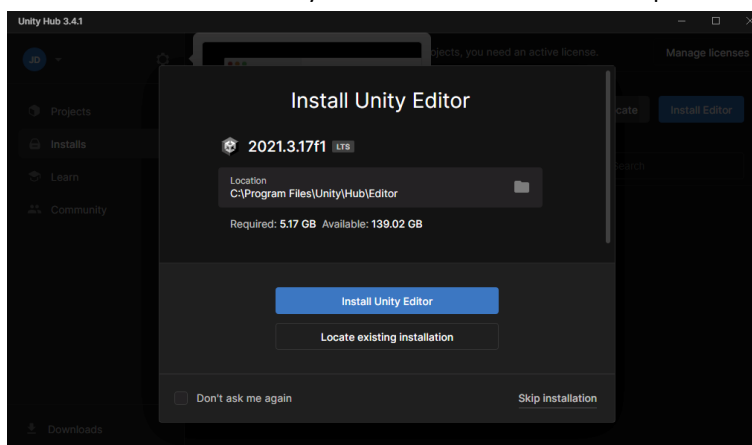
OS:

- ✓ Windows 7 SP1+, 8, 10, 64-bit versions only
- ✓ Mac OS X 10.13+
- ✓ Ubuntu 16.04, 18.04
- ✓ CentOS 7

GPU:

- ✓ Graphics card with DX10 (shader model 4.0) capabilities.

Once you went through the installer, you go to the Unity Hub. You will be prompted to create a Unity account or to login. I recommend you to do that. After you have created an account or logged in, you can install a version of the Unity Editor. I recommend you to install the latest LTS version of the Unity Editor. To install this editor you need to have a license. The personal edition license is sufficient.



The installation of the editor can take a while. Now that we have installed our Unity Editor, you can install you favorite **IDE (Integrated Development Environment)**.
An IDE...

- ✓ Helps us write code to tell the game engine (Unity) what to do
- ✓ Auto-complete, color coding, syntax error checking...

Mine is **Visual Studio Code**: <https://code.visualstudio.com/> .
Click on the installer and follow the instructions.

Creating the flight simulator in Unity

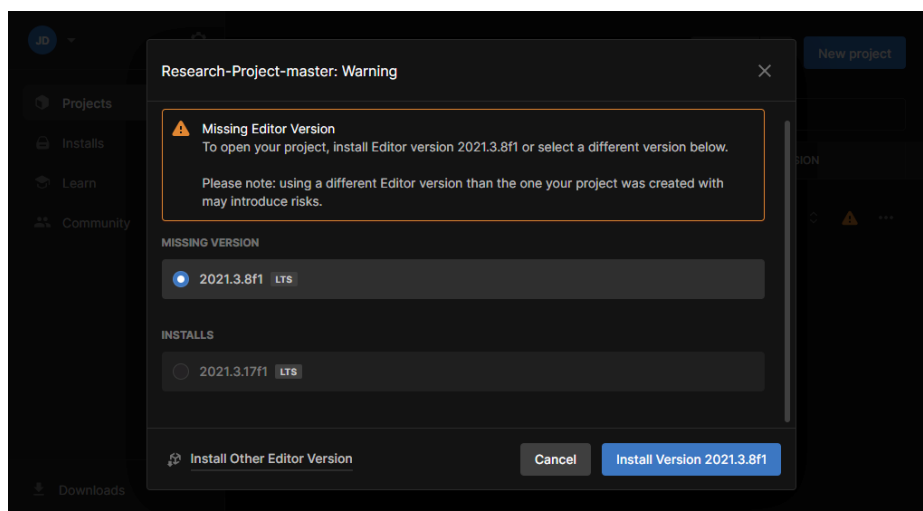
Now we have everything ready to set up our Unity project. As mentioned earlier we want to create an airplane AI that can fly to a given point (sphere) in the environment space.

Creating a flight simulator all by yourself is really hard and can be time consuming. In the beginning of the project I tried to create my own flight simulator but after a few days, I realized I'm wasting my time. After researching for a while I found this **Github** repository that contains a really good and realistic flight simulator: https://github.com/skiwee45/Airplane_Tutorial based on **Ivan Pensionerov's aircraft physics system**: <https://www.youtube.com/watch?v=p3jDJ9FtTyM&t=134s>.

Ivan Pensionerov's aircraft physics system is a Unity plugin that provides realistic flight physics and aerodynamics for aircrafts. The system is designed to be easy to use, and can be integrated into Unity projects with little to no programming experience. The system simulates various physical phenomena such as **thrust, drag, lift, and weight**, to provide a realistic and immersive flight experience. It includes a variety of customizable settings and parameters to allow users like me to fine-tune the flight behavior of their aircraft.

My project is based on Ivan Pensionerov's aircraft system, I made my own scene and I already trained a lot of models. To access my project you need to clone my repository or download the zip-file:
<https://github.com/JarneDemoen/Research-Project>

Now go to the Unity Hub and click **Open → Add project form disk**. This will open your explorer so you can choose the location of the project you cloned or downloaded. Now our project will be visible in the Unity Hub and we can click on it to open the project. When trying to open the project this message may pop up:



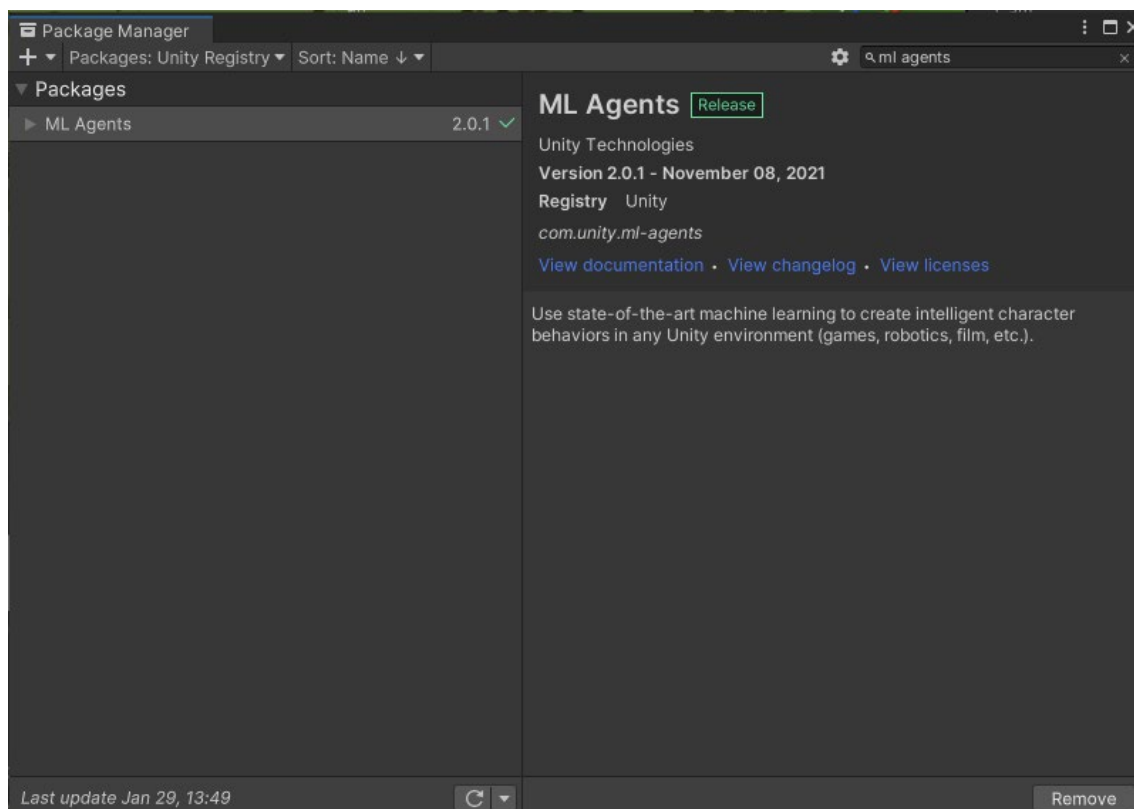
You could open the project with your current installation of the Unity Editor but to be sure I recommend you to install the same version of the project so click on **Install Version 2021.3.8f1**. This can take a while. After the installation you can successfully open my Unity project.

The only thing left to do is to go to **Assets/AirplaneAI/Scenes/AirplaneAI** and press **Play!**

Now you see the airplane in action, it constantly tries to fly towards a random sphere in the environment space.

Training your own AI using ML-Agents

Training an AI in Unity requires the **ML-Agents package**. To install this package you go to **Window → Package manager → Select Packages**: Unity Registry and type in ml agents



Make sure it is version 2.0.1.

Creating your own virtual environment

To train an mlagent you need to create your own **virtual environment**. I used **miniconda** for that. You can install miniconda via this link: <https://docs.conda.io/en/latest/miniconda.html>. Click on the installer and follow the instructions. Next open the **anaconda prompt** and go to the location of the Unity Project and type **conda create -n mlagents python=3.8**


Now that our virtual environment is installed and our python install is complete we can activate the environment with this command: **conda activate mlagents**. The most important package we need to install is the mlagents package: **pip install mlagents**.

Next up is **Pytorch**. Go to the Pytorch website (pytorch.org), select the requirements of your device and run the command that is presented to you.

PyTorch Build	Stable (1.10.2)	Preview (Nightly)	LTS (1.8.2)
Your OS	Linux	Mac	Windows
Package	Conda	Pip	LibTorch
Language	Python	C++ / Java	
Compute Platform	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)
Run this Command:	conda install pytorch torchvision torchaudio cudatoolkit=10.2 -c pytorch		

Previous versions of PyTorch >

This installation can take a while. To check if your installation is correct, you can use the **mlagents-learn** command and if it throws no errors and displays the Unity logo, you're good to go.



```

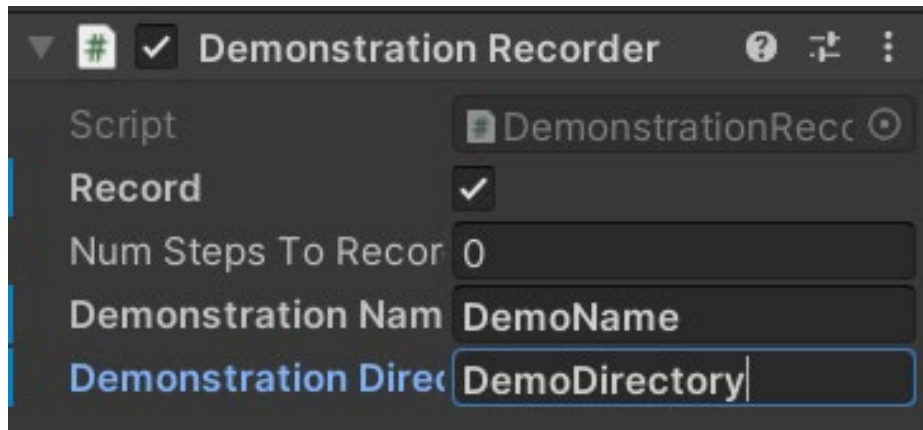
Version information:
ml-agents: 0.28.0,
ml-agents-envs: 0.28.0,
Communicator API: 1.5.0,
PyTorch: 1.10.2
[INFO] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.

```

The next thing we need is a **config file**, you can find the config file in my repository (learnToFly.yaml).

This config file is used to train an agent in Unity using mlagents behaviors, specifically for an airplane AI. The trainer type used is **Proximal Policy Optimization (PPO)** and several hyperparameters are set for the training, including **batch size, buffer size, learning rate, beta, epsilon, lambda, and number of epochs**. The network settings are also specified, including normalization, number of hidden units and layers, and various encoding and conditioning types. The reward signals for extrinsic and **General Adversarial Imitation Learning (GAIL)** are also set, including strength and path for demonstration data. **Behavioral cloning** is also utilized with a specified strength and demonstration path. The agent's progress is checked every 500000 steps and a maximum of 50000000 steps are set for training, with a summary frequency of 50000 steps. The agent is also set to run in a threaded mode. The agent's progress is checkpointed every 500000 steps and the last 5 checkpoints are kept.

GAIL and Behavioral cloning are very important because it speeds up the training a lot! As mentioned you need to specify a path to the directory of the demo file. You can create a demo on your own by adding the **demonstration recorder** to the agent:



Press play and the recording has started. As soon as you stop the scene, the demonstration file will be stored in the directory you specified. After creating the demo file, you can uncheck the **record** option.

General Adversarial Imitation Learning (GAIL) is a method for training agents in which a discriminator is trained to distinguish between the actions of the agent and a set of expert demonstrations. The agent is then trained to maximize the likelihood that the discriminator classifies its actions as those of the expert. One benefit of GAIL is that it allows the agent to learn from expert demonstrations without the need for explicit supervision or reward signals. This can be useful in situations where it is difficult or expensive to obtain labeled data or define a reward function. GAIL can help to improve the sample efficiency of the learning process, as the agent can learn from the expert's experience rather than having to explore the environment on its own.

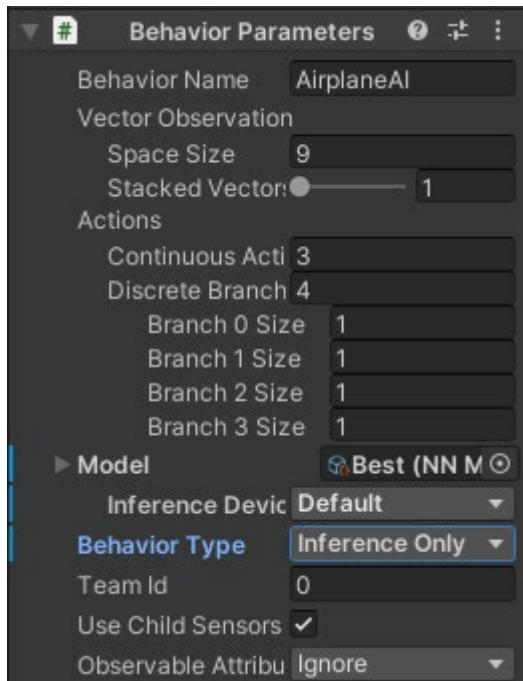
Behavioral Cloning is a method for training agents by learning from expert demonstrations. The idea behind Behavioral Cloning is that if an agent can mimic the behavior of an expert, it will be able to perform the task well. The agent is trained by providing it with expert demonstrations and it learns to replicate the expert's actions based on the input-output pairs of the expert. One benefit of Behavioral Cloning is that it can be very sample efficient as it only requires expert demonstrations to train the agent. Another benefit is that it can be used in domains where it is difficult or expensive to obtain labeled data or define a reward function. Behavioral Cloning can work well in environments where the expert's actions are deterministic and the agent's actions can be replicated with a high degree of accuracy.

But how do we train the agent with this configuration file?

1. Open anaconda prompt
2. Activate the mlagent environment → **conda activate mlagents**
3. Go to the directory of the Unity Project
4. Start the training with the following command: **mlagents-learn path-to-config-file --run-id="RUNNAME" --quality-level=0**

The "**quality-level**" flag in the mlagents-learn command is used to specify the quality level of the agent being trained. It is a value between 0 and 3, with 0 being the lowest quality and 3 being the highest. The quality level determines the amount of computation and resources that will be used to train the agent. At a lower quality level, less computation and resources will be used, which may result in faster training times, but with less accurate results. At a higher quality level, more computation and resources will be used, which may result in slower training times, but with more accurate results.

You can find the results of the training in the **results** directory. You will find a .onnx file. Finally you can use that .onnx file by selecting **model** in the aircraft prefab:



Make sure **Behavior type** is set to **Inference Only**. This way the neural network won't be trained anymore and when you press play you see the airplane in action with the brains you trained.

Reward system

In my project I tried many different reward systems but the most simple one worked out the best:

```
1 reference
private float CalculateReward()
{
    float reward = 0f;
    distanceToTarget = Vector3.Distance(transform.position, targetObject.transform.position);

    if(distanceToTarget < previousDistanceToTarget)
    {
        reward += 300f/(distanceToTarget + 0.001f);
    }
    else
    {
        reward -= distanceToTarget/10000f;
    }

    previousDistanceToTarget = distanceToTarget;

    return reward;
}
```

The agent gets a reward when it flies closer to the target and gets punished when it flies away from the target. The closer the airplane flies to the target, the more rewards it gets per frame. The further the airplane flies from the target, the more it gets punished per frame. The downside to this is that the agent likes to fly nearby the target but not through the target to receive more rewards although it gets significantly more rewards when hitting the target. You can always change the reward system in the **AirplaneController.cs** file.

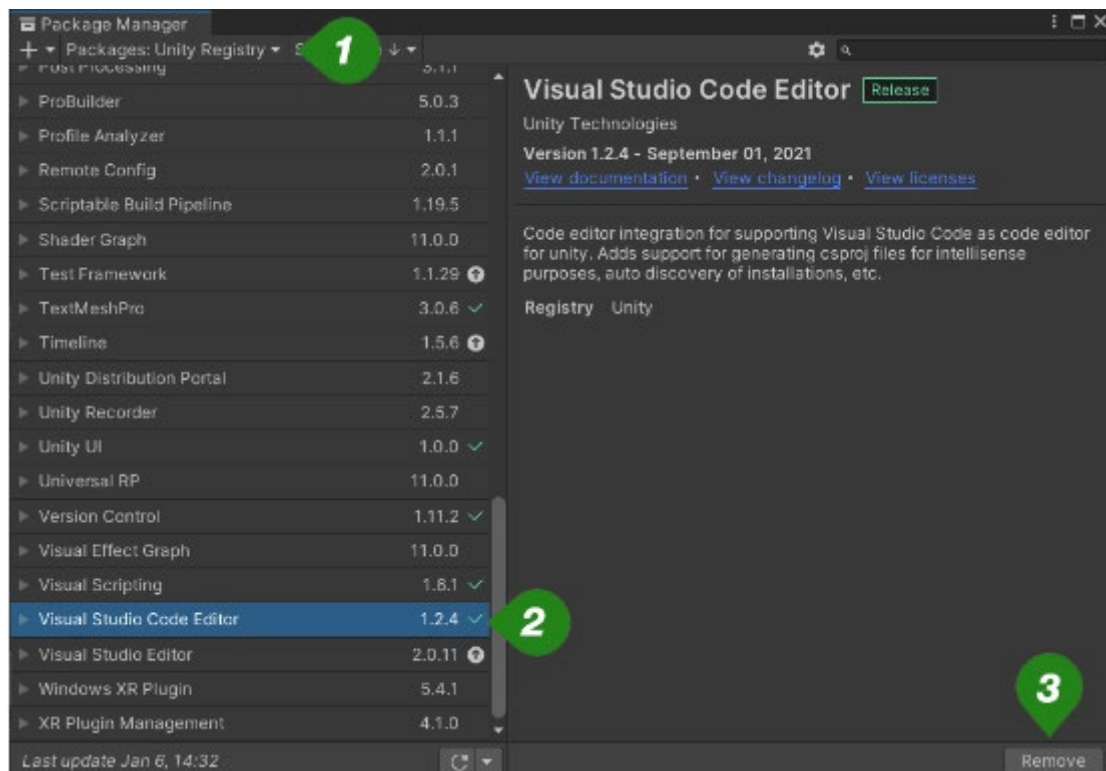
IntelliSense Issues?

If IntelliSense isn't working for you in VS Code there are some simple steps you might want to try.

First, have you tried turning it off and on again?! Give your computer a quick restart and see if that helps. Sometimes this is all you need after some new software has been installed.

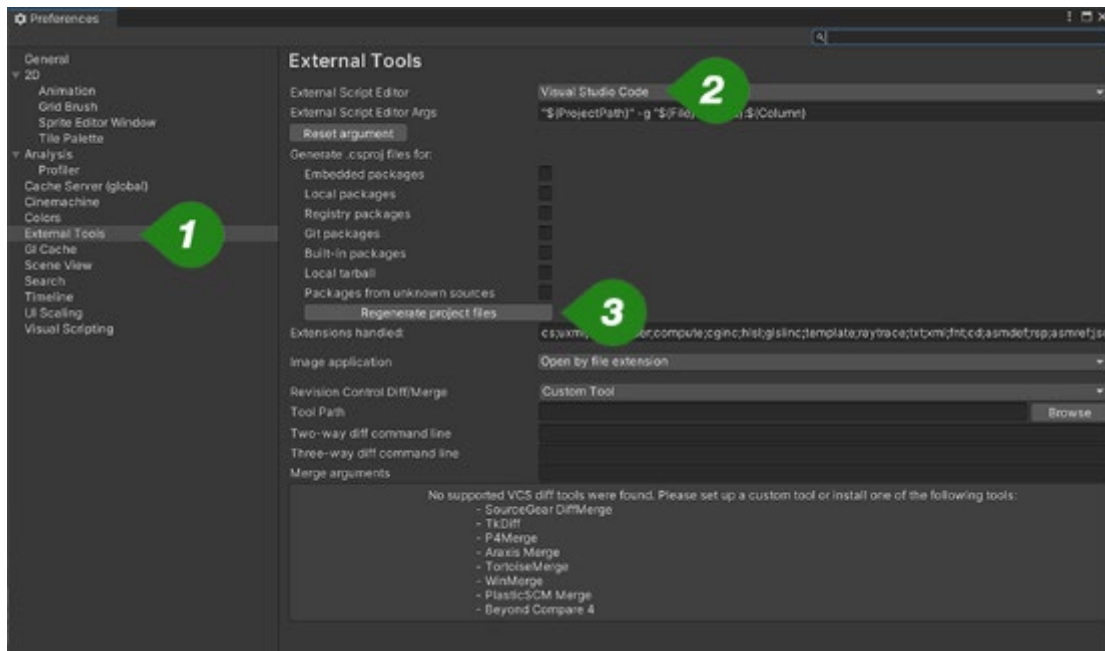
If that didn't work, the next step is to check that you have the Visual Studio Code package installed via the Unity Package Manager. In Visual Studio Code I recommend you to install the **Unity tools extension and Unity Code Snippets**

In Unity, go to: Window > Package Manager. Then, make sure you select "Packages: Unity Registry" from the dropdown menu in the top left and find "Visual Studio Code Editor" in the list. Now make sure this is installed and updated. If you just have a 'Remove' button then you're good to go!



Next, make sure that VSCode is set as the External Script Editor in Unity.

You can check this by going to: Edit > Preferences > External Tools, and then selecting Visual Studio Code from the drop down menu. You may also need to click on the “Regenerate project files” button and including the “Registry packages” option can also help.



If it's still not working the next step is to double check that you have the correct .NET framework installed on your machine.

Start with the latest .NET Core SDK and .NET SDK packages, which you can download from <https://dotnet.microsoft.com/download>.

Once these are installed, give your computer another quick restart.

If the latest release doesn't work for you, you can install the **.NET 4.7.1 (Developer Pack)** as well via this link: <https://dotnet.microsoft.com/en-us/download/dotnet-framework/net471>.

Again, restart your device.

Final solution is to open up VSCode and check that the correct solution workspace is selected.

- Press Ctrl+Shift+P to open the command palette
- Then type “Omnisharp: Select Project”
- Finally, choose the solution workspace (.sln) for the project you're currently working on

howest
hogeschool