AI RAGDOLL BALANCE TRAINING - FRESH START GUIDE

Using Unity Ragdoll Wizard + Improved Training System

This guide combines:

- Unity's built-in Ragdoll Wizard (simpler, more stable)

- Our improved reward/penalty system

- Proper joint control for learning

Hardware: MY HARDWARE: Intel i9-14900KS | NVIDIA RTX 4090 | 64GB RAM

PART 1: INSTALLING EVERYTHING

Step 1.1: Install Unity Hub

1. Open your web browser
2. Go to: https://unity.com/download
3. Click the big blue **"Download Unity Hub"** button
4. Run the downloaded installer (UnityHubSetup.exe)
5. Follow the installation wizard:
    a. Click "I Agree" on the license
    b. Choose install location (default is fine)
    c. Click "Install"
6. When finished, click "Finish" to launch Unity Hub

Step 1.2: Create a Unity Account (if you don't have one)

1. Unity Hub will open
2. Click **"Sign in"** in the top right
3. Click **"Create account"** if you don't have one

4. Fill out the form with your email and password
5. Verify your email
6. Sign in to Unity Hub

Step 1.3: Activate a Unity License

1. In Unity Hub, click the **gear icon** (⚙️) in the top left
2. Click **"Licenses"** in the left sidebar
3. Click **"Add"** button
4. Select **"Get a free personal license"**
5. Click **"Agree and get personal edition license"**
6. You should see "Personal" license appear in the list

Step 1.5: Install Python (Required for ML-Agents Training)

1. Open your web browser
2. Go to: https://www.python.org/downloads/
3. Download **Python 3.10.x** (NOT 3.12 — ML-Agents works best with 3.10)
4. Run the installer
5. **CRITICAL**: Check the box that says **"Add Python to PATH"** at the bottom!
6. Click **"Install Now"**
7. When done, click "Close"

Verify Python Installation:

1. Press `Windows Key + R`
2. Type `cmd` and press Enter
3. In the black command window, type:

```
python --version
```

4. You should see: `Python 3.10.x`
5. If you see an error, restart your computer and try again

Step 1.6: Install PyTorch and ML-Agents Python Package

1. Open Command Prompt (Windows Key + R, type `cmd`, press Enter)
2. **First, upgrade pip** (copy and paste this entire line):

```
python -m pip install --upgrade pip
```

3. **Install PyTorch with CUDA** (for your powerful GPU):

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121
```

(This takes a few minutes)

4. **Install ML-Agents**:

```
pip install mlagents==1.0.0
```

5. **Verify installation**:

```
mlagents-learn --help
```

You should see a help message with many options. If you see an error, something went wrong.

STEP 1: CREATE NEW UNITY PROJECT

Open Unity Hub

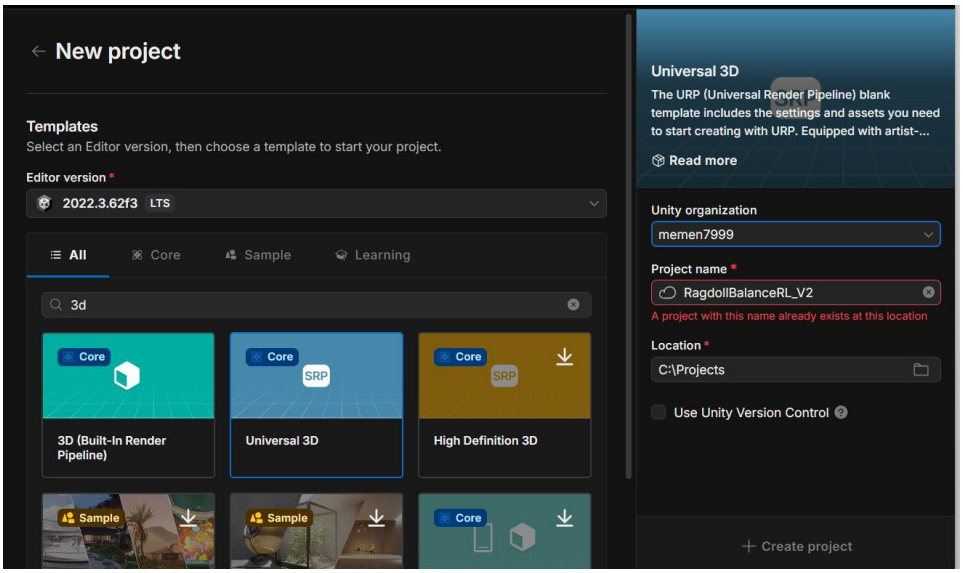
Click "New Project"

Settings: • Template: 3D (URP) - Universal Render Pipeline • Project Name: RagdollBalanceRL_V2 • Location: C:/Projects/RagdollBalanceRL_V2

(you can change name if you want, this is what I did)

Click "Create Project"

Wait for Unity to initialize (~2-3 minutes)



##########

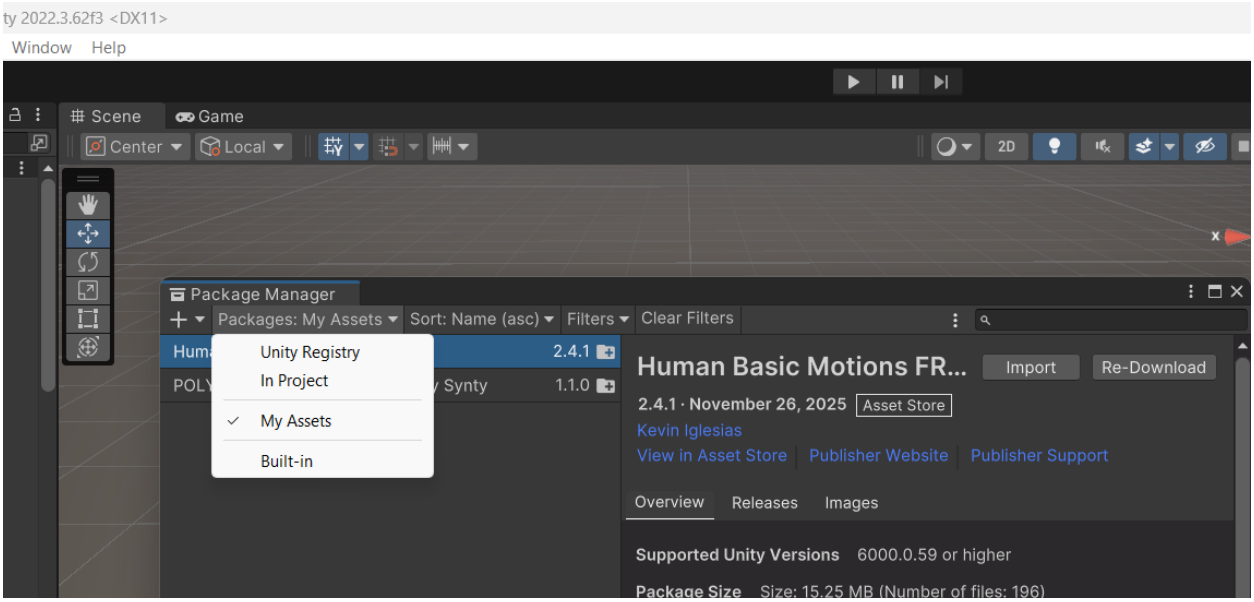
STEP 2: INSTALL ML-AGENTS PACKAGE

##########

In Unity Editor:

1. Window → Package Manager
2. Click "+" (top left) → "Add package from git URL"
3. Enter: com.unity.ml-agents
4. Click "Add"
5. Wait for installation (~2 minutes)

Verify: You should see "ML Agents" listed in Package Manager.



##

STEP 3: IMPORT CHARACTER MODEL

##

Import "Human Basic Motions FREE" from Asset Store:

1. Window → Asset Store (opens in browser) Or go to: [Human Basic Motions FREE | 3D Animations | Unity Asset Store]
2. Click "Add to My Assets" (sign in if needed)
3. Back in Unity: Window → Package Manager
4. Top dropdown: Change "Unity Registry" to "My Assets"
5. Find "Human Basic Motions FREE"
6. Click "Download" (wait ~1 minute)
7. Click "Import"
8. Click "Import" again to confirm

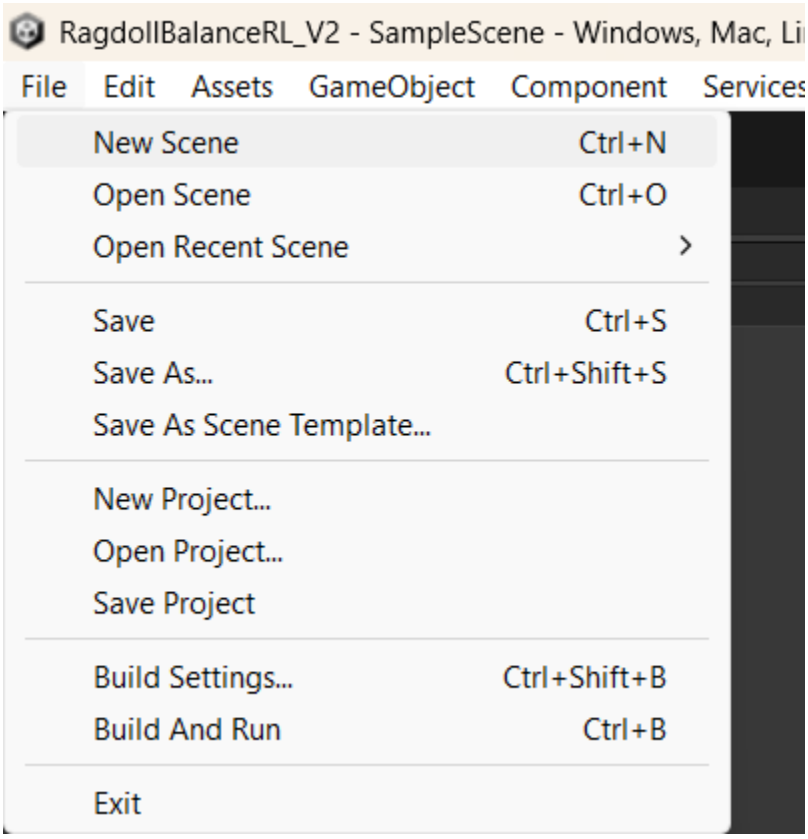
Verify: In Project panel, you should see: Assets/Kevin Iglesias/Basic Motions/

##

STEP 4: SETUP TRAINING SCENE

##

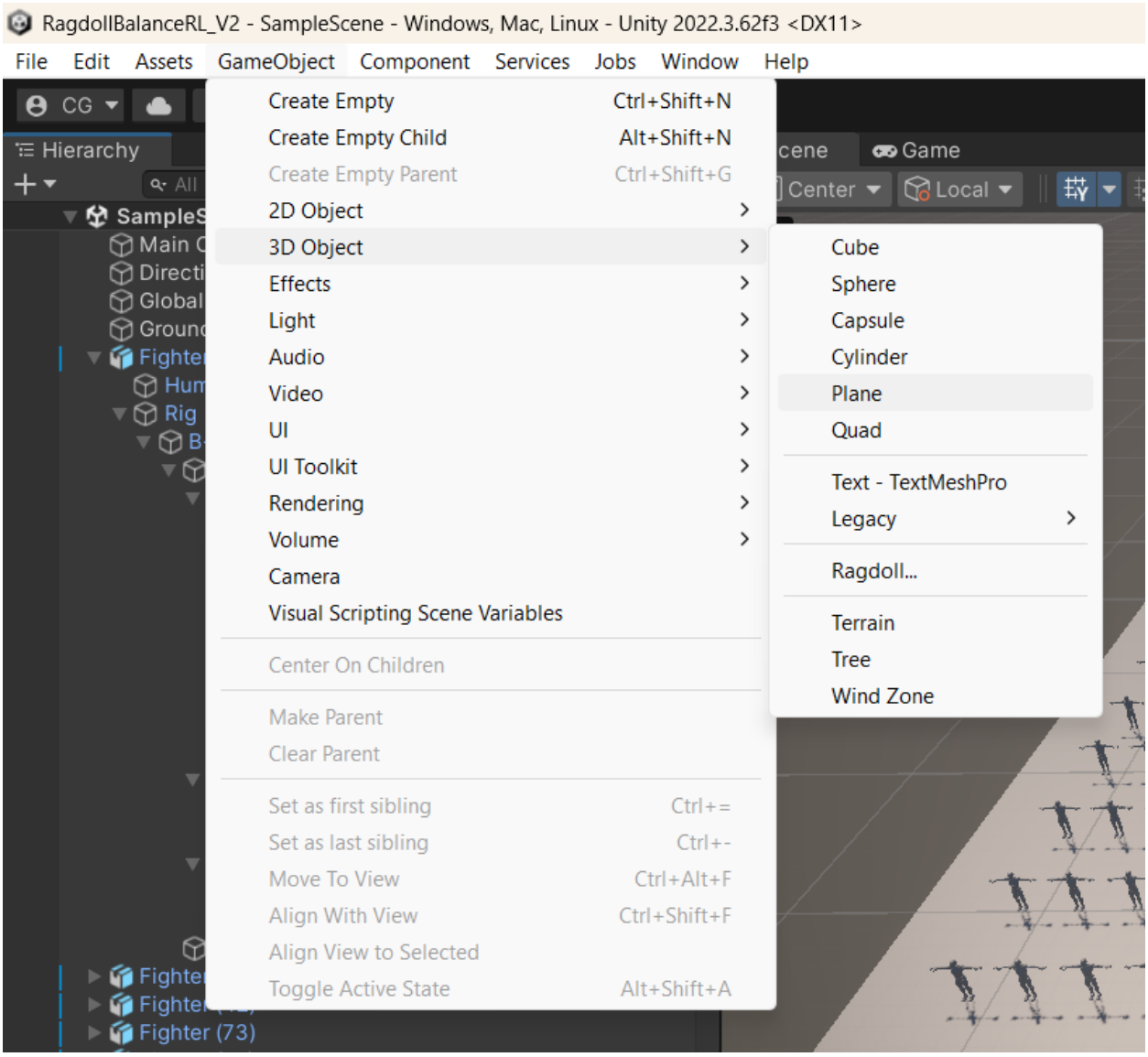
A) Create New Scene

1. File → New Scene
2. Save As: Assets/Scenes/Training_Balance.unity
3. 

B) Create Ground Plane

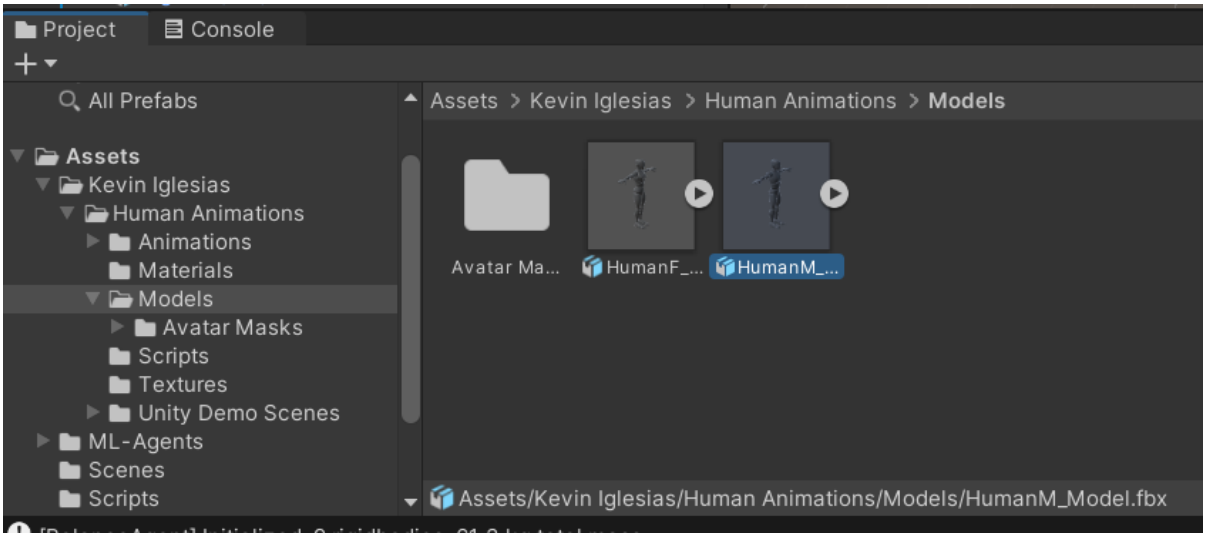
1. Hierarchy → Right-click → 3D Object → Plane
2. Rename: Ground
3. Transform (in Inspector): • Position: (0, 0, 0) • Rotation: (0, 0, 0) • Scale: (5, 1, 5) - creates 50m x 50m area
4. Create Layer: • Top of Inspector → Layer dropdown → Add Layer • In an empty slot, type: Ground • Select Ground plane again → Layer: Ground

5. Create Physics Material: • Project panel → Right-click → Create → Physics Material • Name: GroundMaterial • In Inspector:
    a. Dynamic Friction: 0.8
    b. Static Friction: 0.8
    c. Bounciness: 0 • Drag onto Ground's Mesh Collider → Material slot



C) Import Character Model

1. In Project panel, navigate to: Assets/Kevin Iglesias/Basic Motions/Models/
2. Find the character prefab ("HumanMale" or similar)
3. Drag into Hierarchy
4. Rename: Fighter

5. Set Position: (0, 0, 0)
6. • If floating, adjust Y until feet touch ground
7. • If sinking, raise Y slightly
8. 

D) CRITICAL: Remove Animator Component

1. Select Fighter in Hierarchy
2. In Inspector, find "Animator" component
3. Click ⚙ gear icon → Remove Component

This is essential! We need physics, not animations.

##

STEP 5: ADD RAGDOLL PHYSICS

##

Use Unity's Built-in Ragdoll Wizard:

1. Select Fighter in Hierarchy
2. Menu: GameObject → 3D Object → Ragdoll...
3. Ragdoll Wizard opens

Assign Bones (Assign manually):

Click each field and select from the dropdown:

1. Pelvis: B-hips
2. Left Hips: B-thigh.L
3. Left Knee: B-shin.L
4. Right Hips: B-thigh.R
5. Right Knee: B-shin.R
6. Left Arm: upperArm.L
7. Left Elbow: B-forearm.L
8. Right Arm: B-upperArm.R
9. Right Elbow: B-forearm.R
10. Middle Spine: B-spine
11. Head: B-head

Settings at bottom:

• Total Mass: 70 • Strength: 0 (we'll configure springs manually)

4. Click "Create"

What Just Happened:

• Unity added Rigidbody to each bone

• Unity added CharacterJoint to each bone (except root)

• Unity added Capsule Colliders to each bone

##

STEP 6: CONFIGURE CHARACTER JOINTS

##

CRITICAL FOR STABILITY!

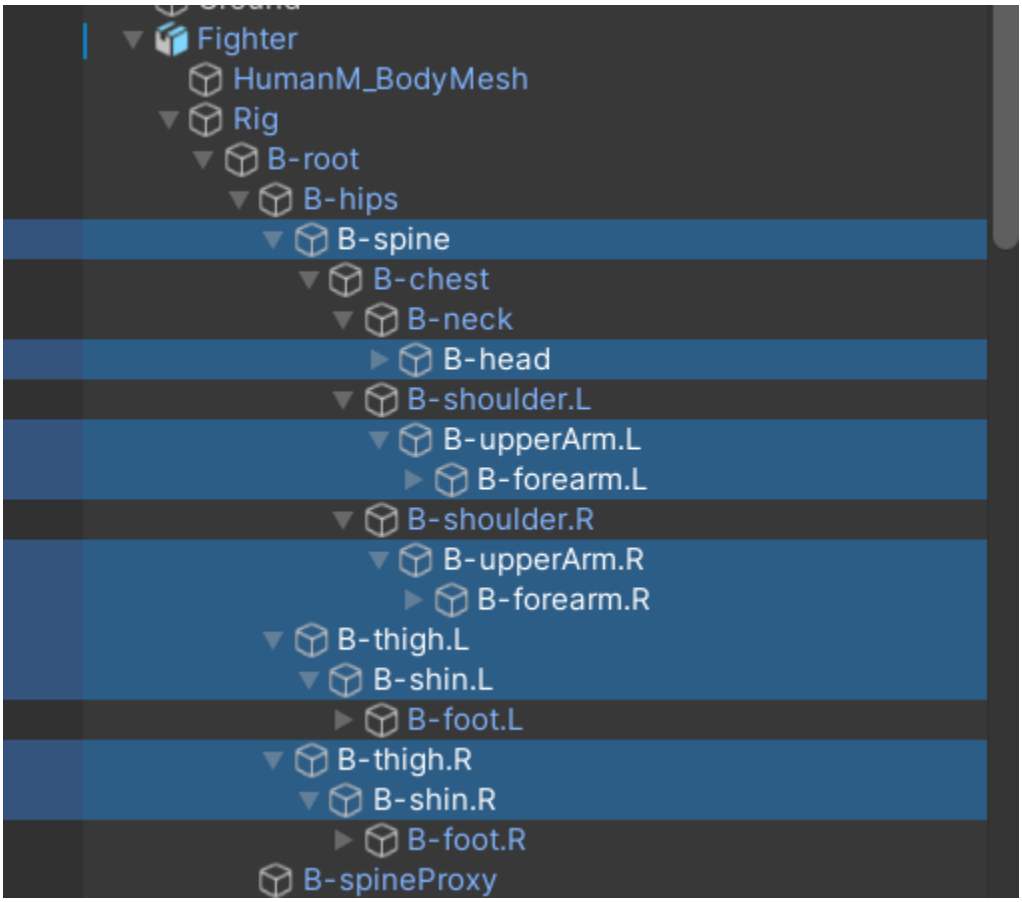
The Ragdoll Wizard creates joints but doesn't enable springs. We need them.

For EACH bone that has a CharacterJoint, do this:

1. Select the bone in Hierarchy
2. Find "Character Joint" in Inspector
3. ☐ Enable Collision ← UNCHECK THIS!
4. Break Force: Infinity Break Torque: Infinity

Bones to configure (expand Fighter hierarchy):

☐ Spine ☐ Head ☐ forearm.L ☐ upperArm.L ☐ forearm.R ☐ upperArm.R ☐ thigh.L ☐ shin.L ☐ thigh.R ☐ shin.R



Quick Method (Multi-select):

1. Hold Ctrl and click all bones with CharacterJoint
2. In Inspector, you'll see shared properties

Step 1: Configure CharacterJoint Springs

For **EACH bone with a CharacterJoint**, set these values:

Twist Limit Spring (expand this section)

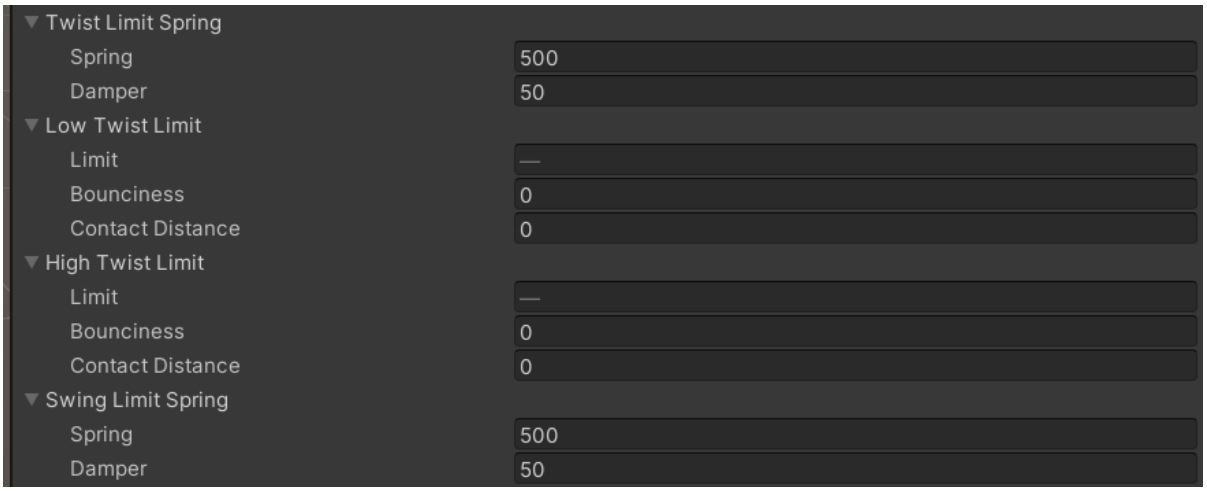
```
Spring:  500
Damper:  50
```

Swing Limit Spring (expand this section)

```
Spring:  500
Damper:  50
```

Do this for:

- B-spine
- B-head
- B-upperArm.L
- B-forearm.L
- B-upperArm.R
- B-forearm.R
- B-thigh.L
- B-shin.L
- B-thigh.R
- B-shin.R

3. 

ESSENTIAL!

##

STEP 7: CREATE RAGDOLL LAYER

##

1. Select Fighter (root object)
2. Top of Inspector → Layer dropdown → Add Layer
3. In User Layer 8 (or empty slot), type: Ragdoll
4. Select Fighter again
5. Layer → Ragdoll
6. Dialog: "Change children?" → YES, change children

Configure Physics Collision Matrix:

1. Edit → Project Settings → Physics
2. Scroll to "Layer Collision Matrix"
3. Find Ragdoll row/column
4. UNCHECK: Ragdoll ↔ Ragdoll (prevents self-collision)
5. KEEP CHECKED: Ragdoll ↔ Ground

##

STEP 8: TEST PHYSICS

##
########

Before adding AI, test that ragdoll works:

1. Save Scene: Ctrl+S
2. Save Project: File → Save Project
3. Click Play ▶

Expected Behavior:

• Character stands for ~0.5-1 second • Then falls/ragdolls realistically • Limbs bend naturally (no pancaking!) • Character doesn't explode or fly apart • Doesn't fall through ground

Troubleshooting:

Character EXPLODES: → Reduce Spring to 2000 → Increase Damper to 500

Character PANCAKES (flattens instantly): → Check ALL joints have Enable Spring ☑ → Increase Spring to 5000 → Edit → Project Settings → Physics: - Default Solver Iterations: 10 - Default Solver Velocity Iterations: 10

Character SINKS through ground: → Check Ground has Mesh Collider → Check physics material is applied

Limbs DETACH: → Set Break Force: Infinity on all joints → Set Break Torque: Infinity on all joints

##

STEP 9: CREATE SCRIPTS FOLDER

##

1. In Project panel, right-click Assets
2. Create → Folder → Name: Scripts
3. Inside Scripts, create: BalanceAgent.cs

Double-click to open in Visual Studio.

STEP 10: THE BALANCE AGENT SCRIPT

##

Delete everything in BalanceAgent.cs and paste the ENTIRE script from the separate file: BalanceAgent.cs

Key features of this script:

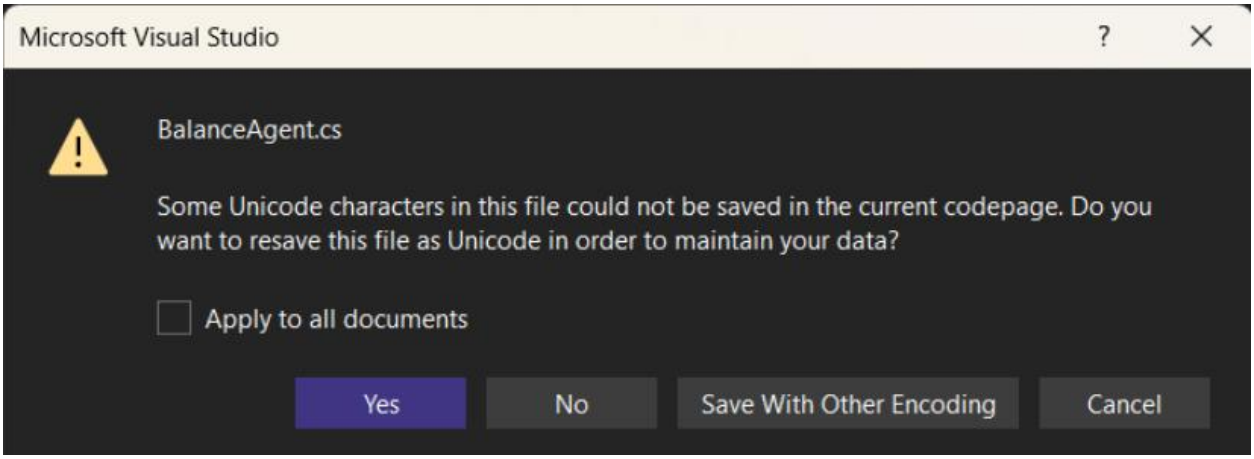
• Works with CharacterJoint (simpler than ConfigurableJoint)

• 45 observations (minimal, focused on balance)

• 0 actions for Gen 1 (observation-only, learns to use springs)

• Height-focused rewards to prevent lying down

• Proper termination timing



##

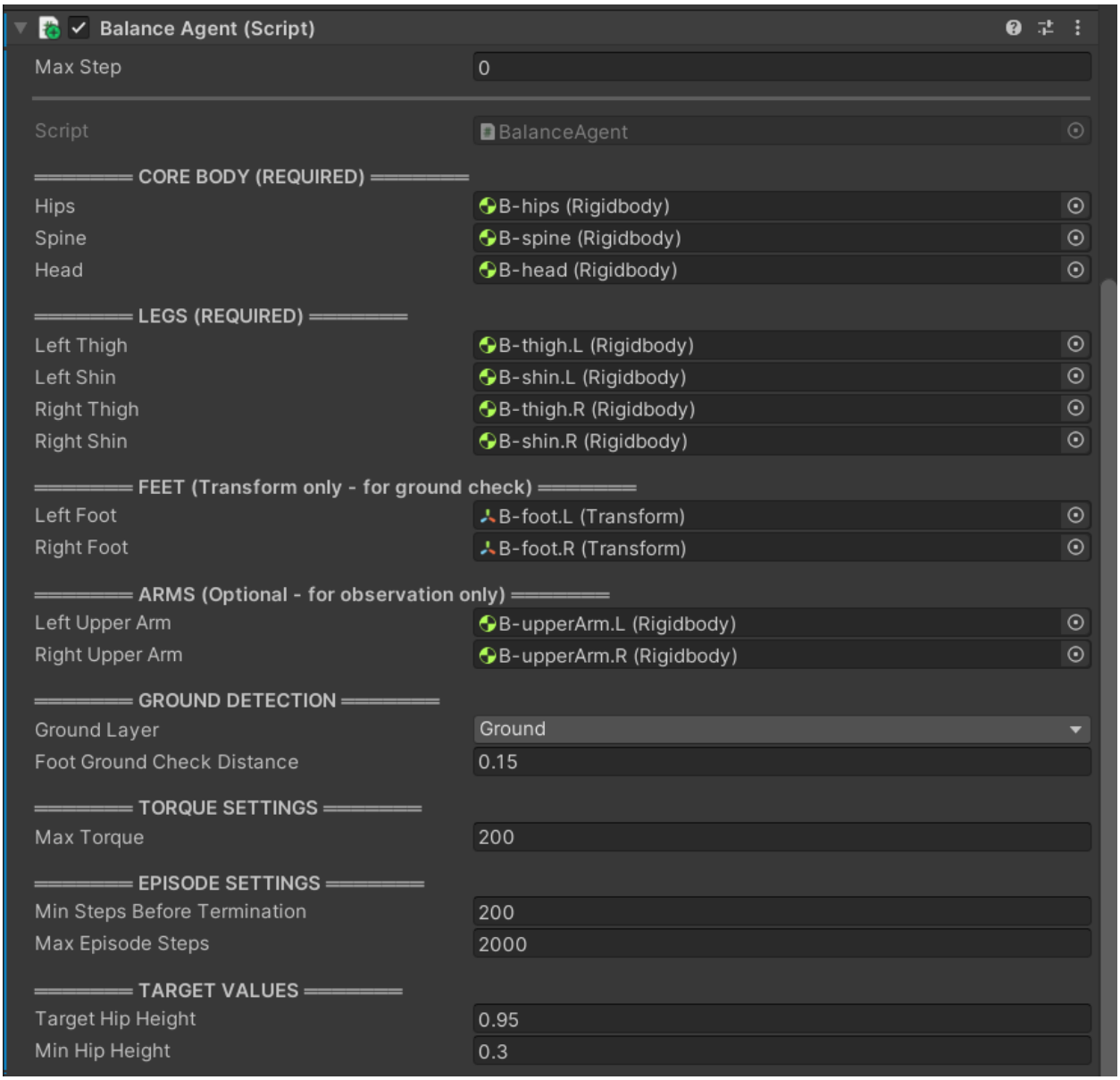
STEP 11: ATTACH SCRIPT TO FIGHTER

##
########

1. Select Fighter in Hierarchy
2. In Inspector, click "Add Component"
3. Type "BalanceAgent" and select it
4. Script appears in Inspector

Wire Up Body Parts:

Expand Fighter hierarchy and drag bones to the script fields:



BODY PARTS: • Hips: Hips bone • Spine: Spine bone • Head: Head bone • Left Foot: LeftFoot bone • Right Foot: RightFoot bone

OPTIONAL (for full observation): • Left Up Leg: LeftUpLeg bone • Left Leg: LeftLeg bone • Right Up Leg: RightUpLeg bone • Right Leg: RightLeg bone

Set Ground Layer:

• Ground Layer dropdown → Select "Ground"

##

STEP 12: ADD BEHAVIOR PARAMETERS

##

Step 4: Add Behavior Parameters

1. Select Fighter
2. Add Component → **Behavior Parameters**:
3. Behavior Name: BalanceAgent
   Vector Observation:
      Space Size: 45
   Continuous Actions: 10
   Discrete Branches: 0
   Model: None
   Inference Device: GPU

Add Component → **Decision Requester**:

4. Decision Period: 5
   Take Actions Between: ☑ (checked)

##

STEP 13: CREATE PREFAB

##

1. In Project panel, create folder: Assets/Prefabs
2. Drag Fighter from Hierarchy into Prefabs folder
3. Rename prefab: Fighter_Balance

##

STEP 14: DUPLICATE FOR PARALLEL TRAINING

##

1. Select Fighter in Hierarchy
2. Ctrl+D to duplicate (repeat 15 times for 16 total or however much you want (i had 172 total)
3. Arrange Fighters in a grid

##

STEP 15: CREATE TRAINING CONFIG

##

1. In Windows File Explorer, navigate to: C:/Projects/RagdollBalanceRL_V2/
2. Create folder: config
3. Inside config, create file: gen1_balance.yaml
4. Open in Notepad, paste the YAML from the separate file.

##

PART 2: Standing Training

##

1. Open Command Prompt
2. Navigate to project: cd C:\Projects\RagdollBalanceRL_V2
3. Start training: mlagents-learn config/gen1_balance.yaml --run-id=balance_v2
   Or
   mlagents-learn config/gen1_balance.yaml --run-id=balance_v2 --force
4. When you see "Listening on port 5004", go to Unity and press PLAY ▶

Monitor with TensorBoard:

Open new Command Prompt:

cd C:\Projects\RagdollBalanceRL_V2 tensorboard --logdir results

Open browser: http://localhost:6006

##

GENERATION 1 EXPECTATIONS

##
########

Timeline (RTX 4090, 16 agents):

- Duration: 6-12 hours
- Steps: 50-100 million
- Success Criteria:
    - Cumulative Reward > 5.0
    - Episode Length > 500 steps
    - Agents stand without falling

What You'll See:

- Hours 0-2: Agents falling immediately
- Hours 2-4: Agents standing for a few seconds
- Hours 4-8: Agents maintaining balance for 10-20 seconds
- Hours 8-12: Agents standing for full episodes

##
########

SUCCESS CHECKLIST

##

Before moving to Gen 2 (Walking):

☐ Gen 1 trained for 50M+ steps ☐ Cumulative Reward > 5.0 in TensorBoard ☐ Trained model saved in results/balance_v2/BalanceAgent.onnx ☐ When loading trained model, agent stands upright consistently ☐ Ready to add movement actions in Gen 2

##

TROUBLESHOOTING

##

Training not starting:

• Check YAML syntax (no tabs, only spaces!) • Verify mlagents is installed: pip list | findstr mlagents

Reward not increasing:

• Check body parts are assigned in Inspector • Check Ground Layer is set • Verify all joints have springs enabled

Agents lying down instead of standing:

• The improved reward system penalizes low hip height • If still happening, increase lying penalty in script

Unity crashes:

• Reduce agents from 16 to 8 • Lower physics quality in Project Settings

GPU not being used:

• Check PyTorch CUDA: python -c "import torch; print(torch.cuda.is_available())" • Should print: True

Current Reward & Penalty System

☑ REWARDS (Positive)

Reward	Amount	Condition
Height	+0.0 to +0.5	Scaled by `hipHeight / targetHipHeight` (0.95m)
Tall Bonus	+0.3	If hip height > 70% of target (~0.67m)
Upright Torso	+0.0 to +0.15	Average of spine.up and hips.up dot with Vector3.up
Both Feet Grounded	+0.05	Both feet touching ground
One Foot Grounded	+0.02	Only one foot touching ground
Survival Bonus	+2.0	Reached max episode steps (2000)

Max per-step reward: ~1.0 (if standing perfectly tall with both feet down)

✖ PENALTIES (Negative)

Penalty	Amount	Condition
Very Low	-0.4	Hip height < 0.5m
Low	-0.2	Hip height < 0.7m (but ≥ 0.5m)
Tilted	-0.15	Spine up dot < 0.5 (leaning too much)
Moving Fast	-0.05	Hips velocity > 2 m/s
Death Penalty	-1.0	Episode ends from collapse

💀 EPISODE TERMINATION

Episode ends (after 200+ steps) if ANY of:

- Hip height < 0.3m (`minHipHeight`)
- Head height < 0.25m
- Spine up dot < 0.1 (nearly horizontal)

Summary

The system heavily rewards **being tall** and **penalizes being low**. This should prevent the "lying down" exploit from before. The agent gets roughly:

- **Standing tall:** +0.8 to +1.0 per step
- **Crouching:** +0.3 to +0.5 per step
- **Lying down:** -0.2 to -0.4 per step

After gen1 trains:

How to Test Your Trained Model

Step 1: Find the Trained Model

In Windows File Explorer, navigate to:

C:\Projects\RagdollBalanceRL_V2\results\balance_v2\BalanceAgent.onnx

(There might also be checkpoint files like `BalanceAgent-1999778.onnx`) (i personally used this one) the number before the .onnx is the step number, choose the largest number .onnx file.

Step 2: Import to Unity

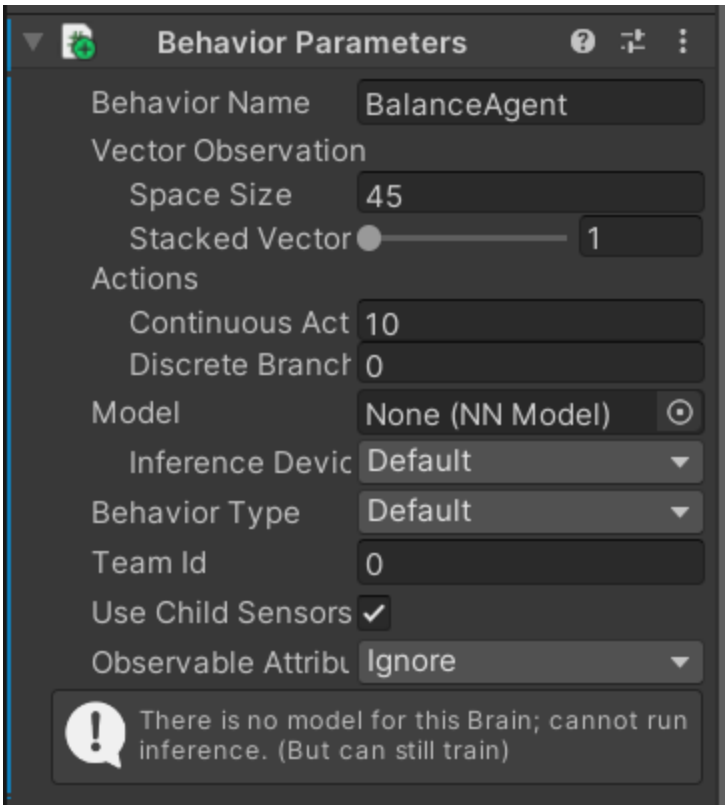
1. Drag the `.onnx` file into your Unity project
2. Put it in: `Assets/ML-Agents/Models/` (create this folder if needed)

Step 3: Assign the Model

1. Select **Fighter** in Hierarchy
2. Find **Behavior Parameters** component
3. Set:

Model: [Drag your BalanceAgent.onnx here]

Behavior Type: Inference Only ← Change from Default!



Step 4: Test It!

1. **Stop training** (Ctrl+C in command prompt if still running)
2. Make sure only **1 Fighter** is in the scene (delete duplicates for cleaner view)
3. Press **Play** ▶

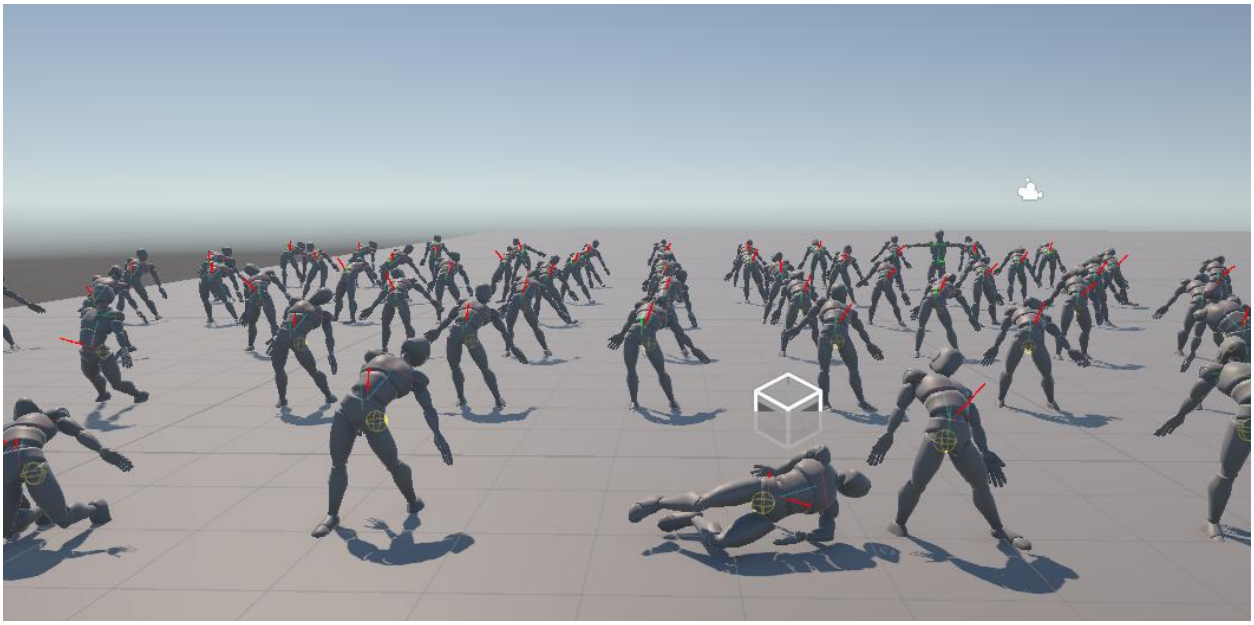
What You Should See

If training worked:

- Character stands upright
- May wobble slightly but stays balanced
- Doesn't fall over

THIS TOOK 2million steps for me, it will take a while to get a actually good stable agent. However my agents were funky

looking. So I made a new stable .cs script to train them to stand straight.



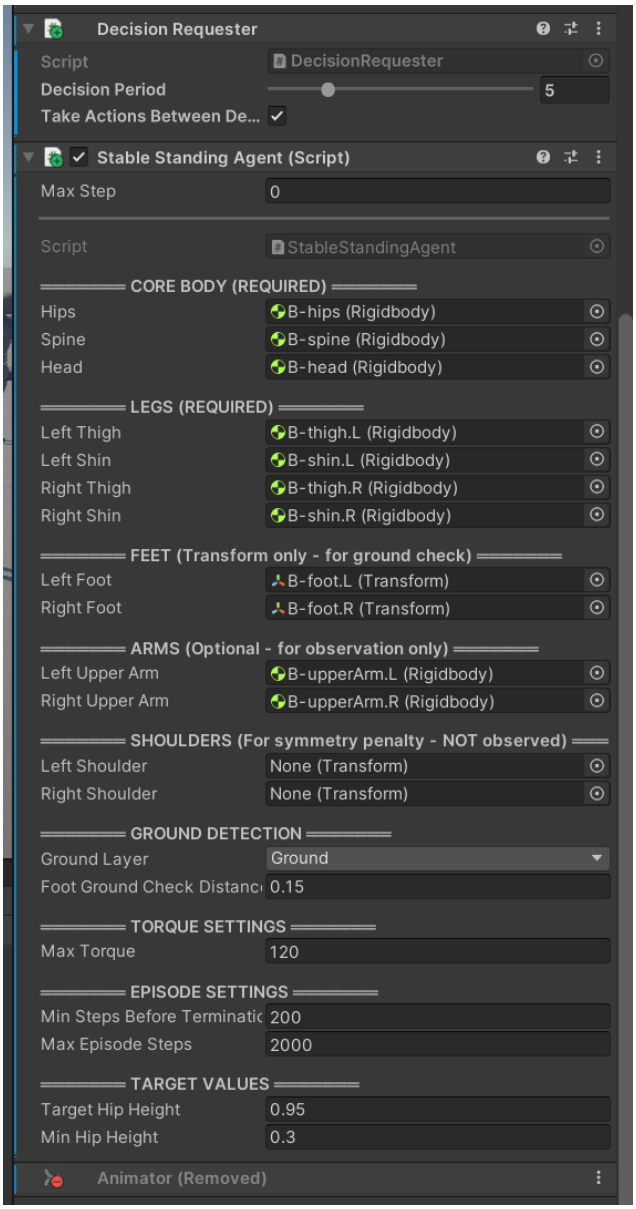
PART 3: Stability Training

Step 1 Create a new .cs script.

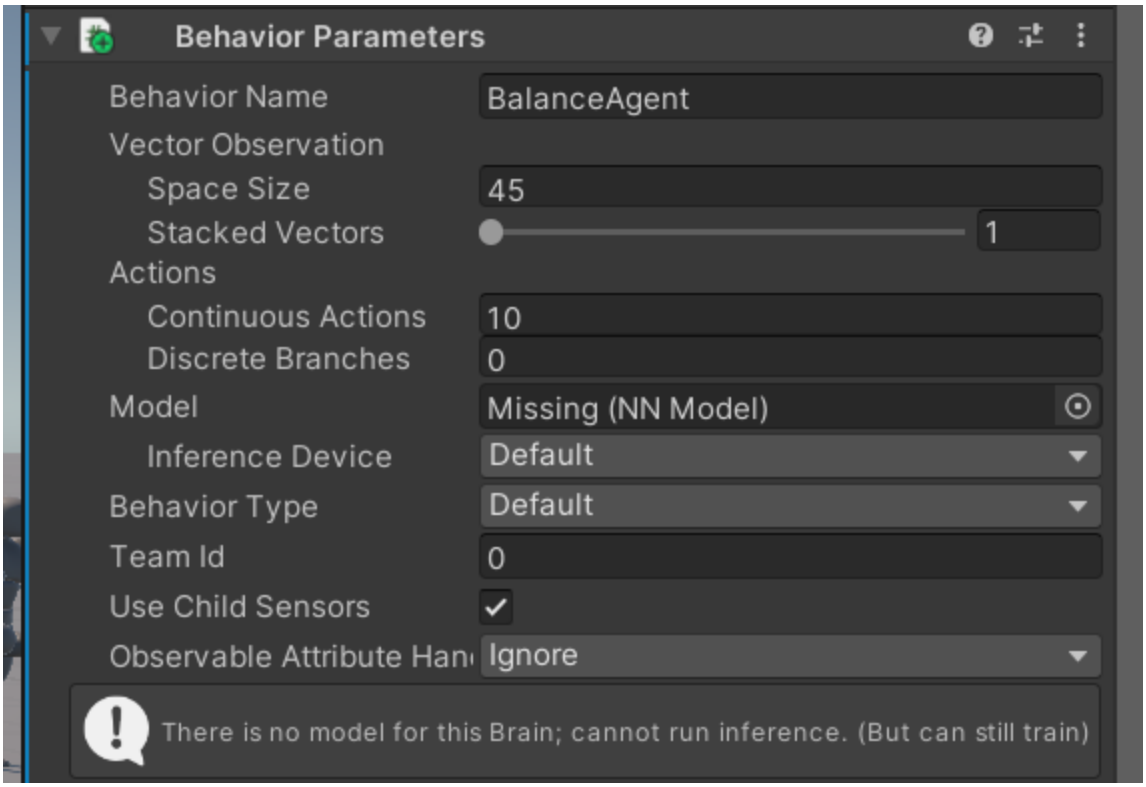
1. Go to Project in bottom left corner of your screen and click Assets -> Scripts
2. Delete your BalanceAgent.cs Script.
3. Right click and create new C# script named StableStandingAgent.cs
4. Double click and enter the code

Step 2, attach script to agents:

1. Select all Fighters
2. Delete Decision Requester
3. Right click on old script
4. Remove script
5. Add component = StableStandingAgent script (fill in the bones)
6. Add component = Decision Requestor

7. 

8. Make sure behavior parameters is this

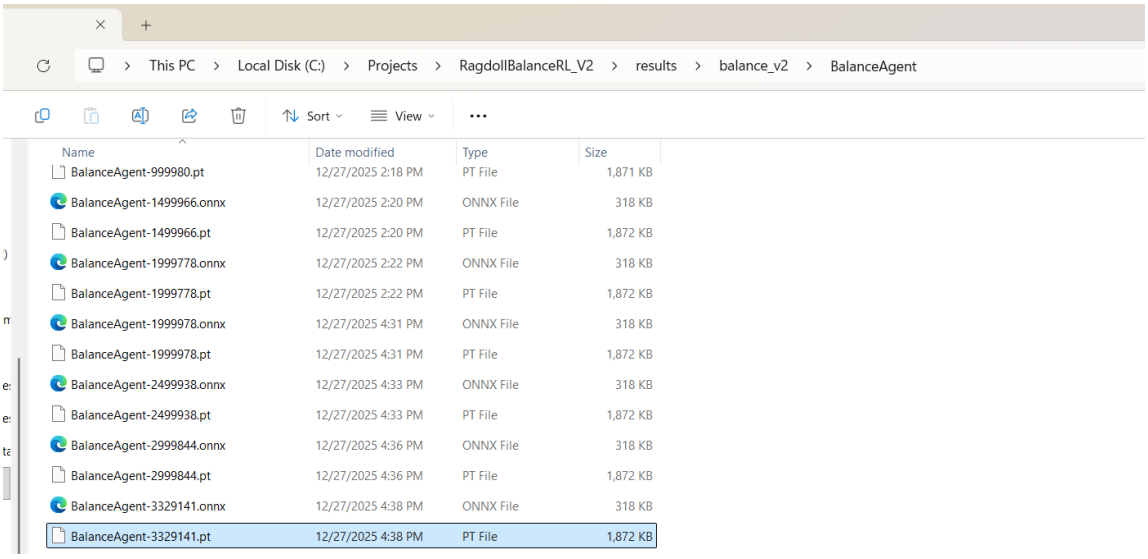
9. 

Step 3. Create new yaml file

1. Go to C:\Projects\RagdollBalanceRL_V2\Config
2. Create new file called StableStanding.yaml (be sure not to label StableStanding.yaml.txt)
3. Enter in new yaml code
4. **IMPORTANTTTTTTTTTTTTT, DO NOT CLOSE IT YET, GO TO FOLLOW NEXT STEPs, YOU NEED TO EDIT THE YAML TO WORK FOR YOUR FILES.**

So next, go to your file that holds your RESULTS of your agents,

1. the file name is results
2. then click your project
3. then click the first folder
4. Then find the .pt of your most recent agent brain
5. For me it was this: C:\Projects\RagdollBalanceRL_V2\results\balance_v2\BalanceAgent\
6. Then the most recent pt was this (YOURS WILL BE DIFFERENT): C:\Projects\RagdollBalanceRL_V2\results\balance_v2\BalanceAgent\BalanceAgent-3329141.pt



7.
8. This is your “brain”
9. Copy this file path AND REPLACE THIS HIGHLIGHTED LINE WITH THE PATH OF YOUR .pt FILE. (if you followed mine exactly, you probably only need to change the numbers in the end (ex. .....BalanceAgent-9310591.pt)

Stablestanding · YAML

Copy

```yaml
behaviors:
  BalanceAgent:
    trainer_type: ppo

    # NOW COMPATIBLE - 45 observations match pretrained model
    init_path: C:\Projects\RagdollBalanceRL_V2\results\balance_v2\BalanceAgent\BalanceAgent-3329141.pt

    hyperparameters:
```

10. Save and exit

Begin Training:

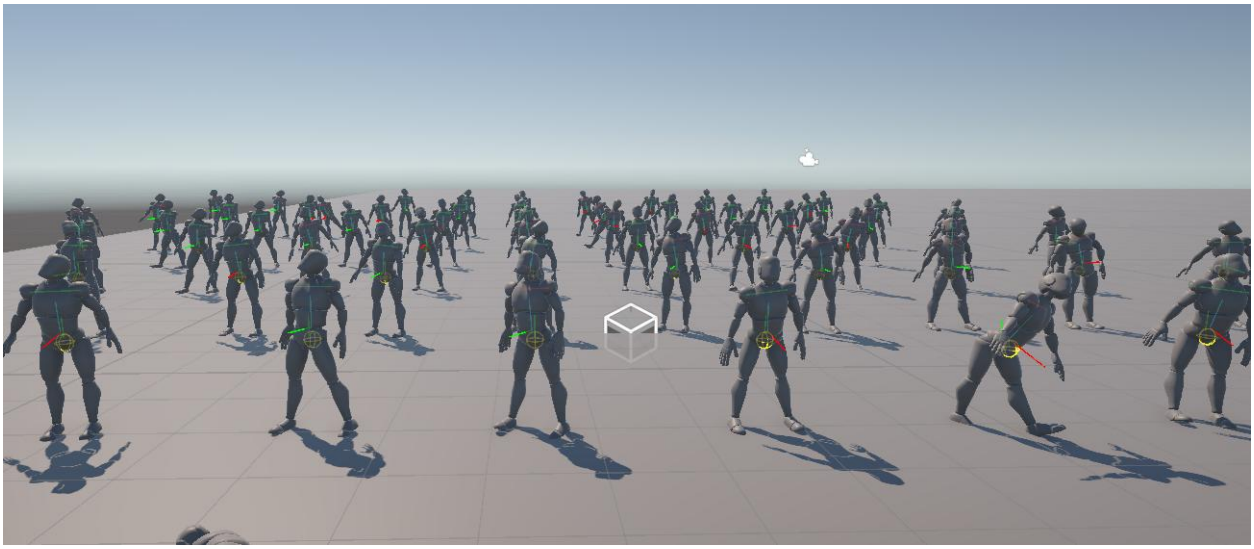
1. Open Cmd
2. Run (note the highlighted areas, change to match yours):
3. mlagents-learn C:\Projects\RagdollBalanceRL_V2\Config\StableStanding.yaml --run-id=stable_v3 --initialize-from=balance_v2 --results-dir=C:\Projects\RagdollBalanceRL_V2\results
4. C:\Projects\RagdollBalanceRL_V2\Config\StableStanding.yaml is the file where we are activating the new .yaml file
5. Stable_v3 is going to be the name or the new file where the new brain is being saved (change it to whatever you want, it doesn’t matter)
6. Balance_v2 is the file we are taking the results of the brain from and initializing from

7. C:\Projects\RagdollBalanceRL_V2\results is where we are saving the Stable_v3 folder to and saving the brain.

```
C:\Windows\System32>mlagents-learn C:\Projects\RagdollBalanceRL_V2\Config\StableStanding.yaml --run-id=stable_v3 --initialize-from=balance_v2 --results-dir=C:\Projects\RagdollBalanceRL_V2\results
```

You should see three phases:

1. They are standing on their feet shoulders not straight and back arched
2. They are falling and their heads are touching the ground, almost like they are crying on the floor.
3. They start standing back up with proper posture. (step 2 million)



This is the final product: