Objetivo

- La documentación va orienta a guiar paso a paso, para personas que no hallan tenido contacto con entornos de simulación sin uso de hardware real.

¿Qué son?

> **¿Qué es Isaac Sim?**
>
> **NVIDIA Isaac Sim** es un **simulador de robótica 3D** desarrollado por **NVIDIA**, usado para **diseñar, probar y entrenar robots en un entorno virtual realista**.

> **¿Qué es Isaac Lab?**
>
> **Isaac Lab** (NVIDIA Isaac Lab) es un **framework de investigación y desarrollo en robótica**, creado por **NVIDIA**, que se usa para **entrenar y evaluar robots**, especialmente con **aprendizaje por refuerzo (Reinforcement Learning)**, **sobre Isaac Sim**.

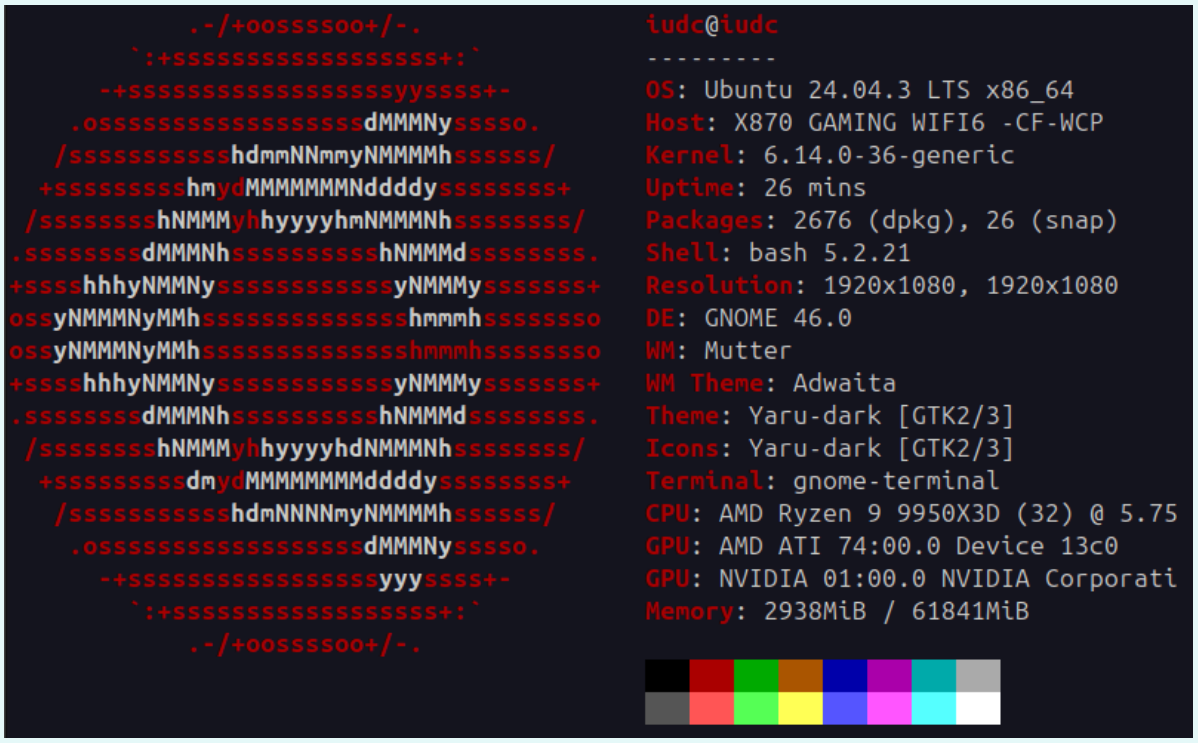
Entorno	¿Qué hace?
Isaac Sim	Mundo simulado
Isaac Lab	Laboratorio para el entrenamiento del robot dentro de **Isaac Lab**

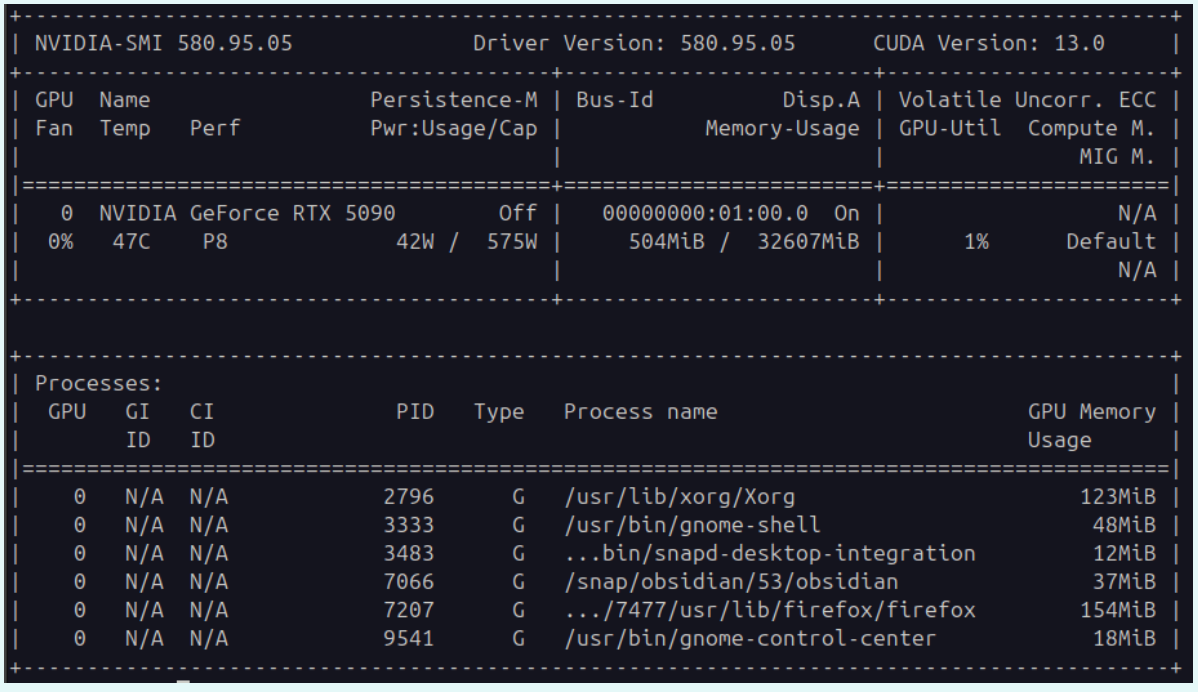
> **Requisitos minimos**
>
> - **OS:** Ubuntu 22.04 (Linux x64) o Windows 11 (x64)
> - **RAM:** 32 GB mínimo o más.
> - **GPU VRAM:** 16 GB o más.

> **Hardware/Software actual**
>
> - Se darán especificaciones actuales:
> - **OS**: *Ubuntu 24.04.3 LTS x86_64*

- **GPU**: *Geforce RTC 5090 de 32 GBs de Ram*
- **RAM**: *62 GBs*

Adjunto imagen que demuestra las especificaciones aneterioes

- 

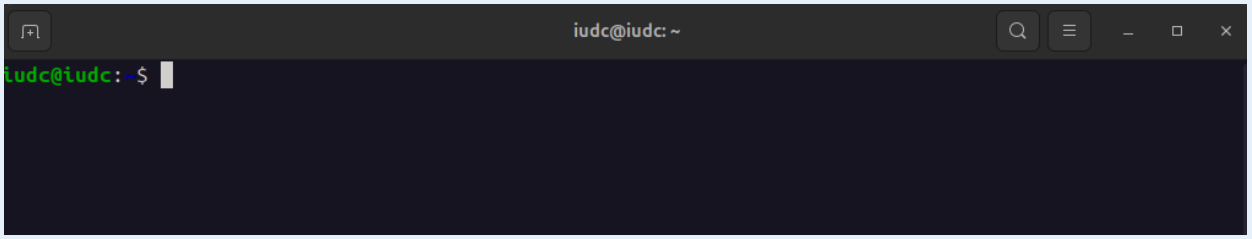
- 

Instalación

Para la instalación se guiara paso a paso, instalación de entorno comando de ejecución, etc...

Instalación de Conda

En una instancia de terminal, `CTRL + ALT + T`:

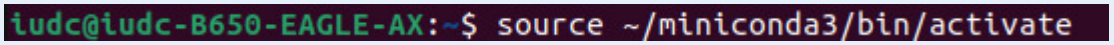


Colocaremos el siguiente comando:

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm ~/miniconda3/miniconda.sh
```

Activamos `Conda` en la terminal después de la instalación:

```
source ~/miniconda3/bin/activate
```



Ahora vamos a activar globalmente `Conda`

```
conda init --all
```

```
(base) iudc@iudc-B650-EAGLE-AX:~$ conda init --all
no change     /home/iudc/miniconda3/condabin/conda
no change     /home/iudc/miniconda3/bin/conda
no change     /home/iudc/miniconda3/bin/conda-env
no change     /home/iudc/miniconda3/bin/activate
no change     /home/iudc/miniconda3/bin/deactivate
no change     /home/iudc/miniconda3/etc/profile.d/conda.sh
no change     /home/iudc/miniconda3/etc/fish/conf.d/conda.fish
no change     /home/iudc/miniconda3/shell/condabin/Conda.psm1
no change     /home/iudc/miniconda3/shell/condabin/conda-hook.ps1
no change     /home/iudc/miniconda3/lib/python3.13/site-packages/xontrib/conda.xsh
no change     /home/iudc/miniconda3/etc/profile.d/conda.csh
modified      /home/iudc/.bashrc
modified      /home/iudc/.zshrc
modified      /home/iudc/.config/fish/config.fish
modified      /home/iudc/.xonshrc
modified      /home/iudc/.tcshrc

==> For changes to take effect, close and re-open your current shell. <==

(base) iudc@iudc-B650-EAGLE-AX:~$
```

Estos significa que cada vez que habramos una nueva terminal, nos aparecera lo siguiente:

```
(base) iudc@iudc:~$
```

Instalación de Isaac Lab

Ahora para la instalación de `Isaac Lab`, nos podemos basar a partir de la documentación Link_Isaac_Lab

---

Ahora una instalación guiada

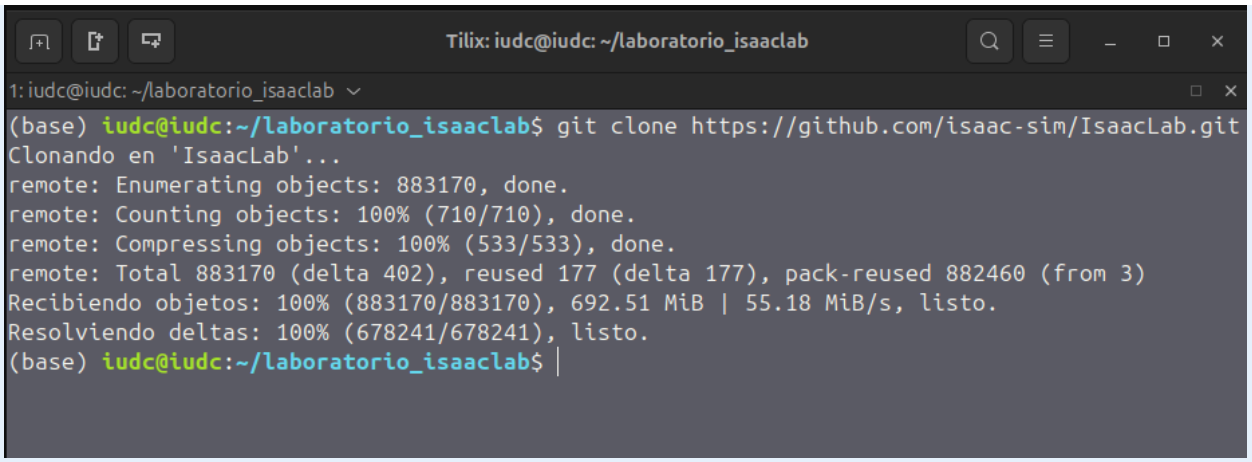
- Creamos una carpeta donde con clonaremos el

```
mkdir -p ~/laboratorio_isaaclab && cd ~/laboratorio_isaaclab
```

```
iudc@iudc: ~/laboratorio_isaaclab
iudc@iudc:~$ mkdir -p ~/laboratorio_isaaclab && cd ~/laboratorio_isaaclab
iudc@iudc:~/laboratorio_isaaclab$
```

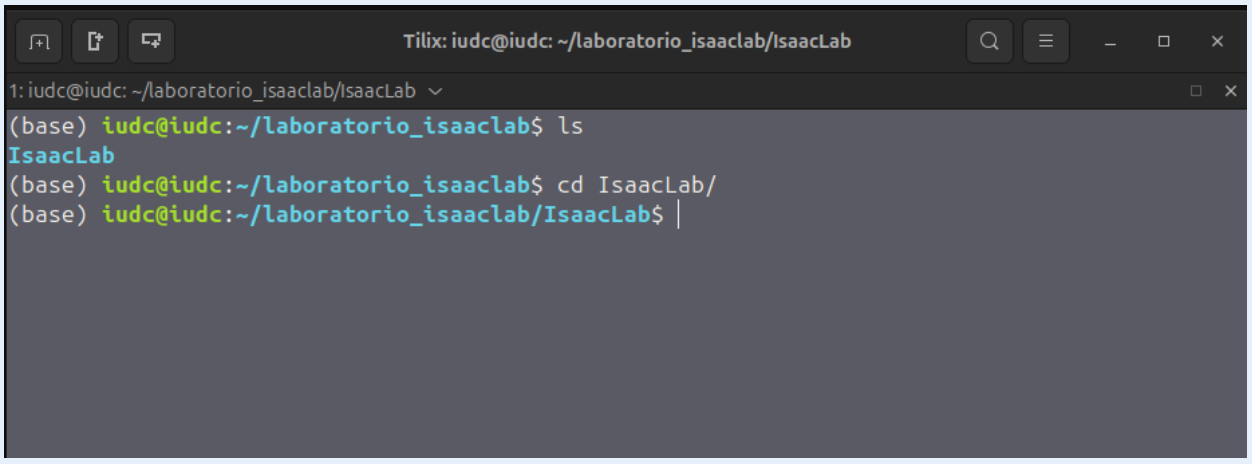
- Clonamos el entorno de `Isaac Lab`

```
git clone https://github.com/isaac-sim/IsaacLab.git
```



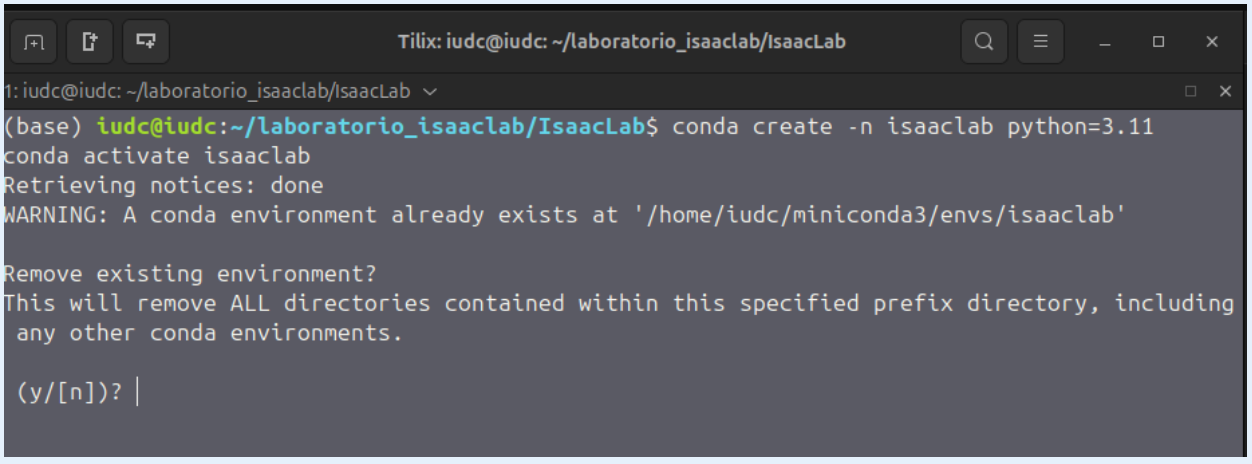
- Ingresamos al entorno clonado

```
cd IsaacLab/
```



- Creamos el entorno con `Conda`

```
conda create -n isaaclab python=3.11
conda activate isaaclab
```



- Confirmamos con `y` para continuar

```
  readline           pkgs/main/linux-64::readline-8.3-hc2a1206_0
  setuptools         pkgs/main/linux-64::setuptools-80.9.0-py311h06a4308_0
  sqlite             pkgs/main/linux-64::sqlite-3.51.0-h2a70700_0
  tk                 pkgs/main/linux-64::tk-8.6.15-h54e0aa7_0
  tzdata             pkgs/main/noarch::tzdata-2025b-h04d1e81_0
  wheel              pkgs/main/linux-64::wheel-0.45.1-py311h06a4308_0
  xorg-libx11        pkgs/main/linux-64::xorg-libx11-1.8.12-h9b100fa_1
  xorg-libxau        pkgs/main/linux-64::xorg-libxau-1.0.12-h9b100fa_0
  xorg-libxdmcp      pkgs/main/linux-64::xorg-libxdmcp-1.1.5-h9b100fa_0
  xorg-xorgproto     pkgs/main/linux-64::xorg-xorgproto-2024.1-h5eee18b_1
  xz                 pkgs/main/linux-64::xz-5.6.4-h5eee18b_1
  zlib               pkgs/main/linux-64::zlib-1.3.1-hb25bd0a_0


Proceed ([y]/n)?
```

- Volvemos a confirmar con `y` , de esa manera finalizara la creación del entorno y automaticamente nos movera al entorno recien montado (`isaaclab`)

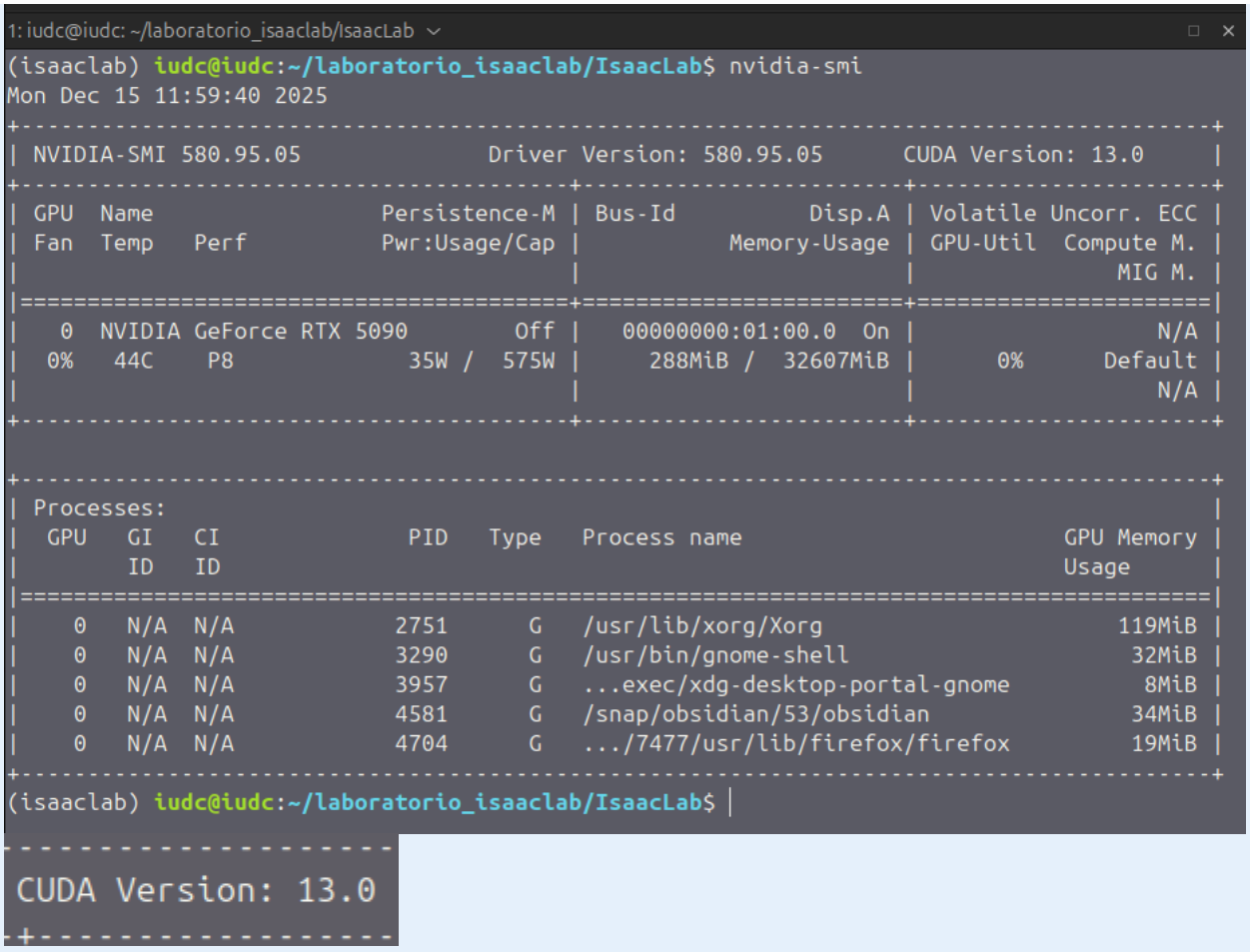
```
Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate isaaclab
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(isaaclab) iudc@iudc:~/laboratorio_isaaclab/IsaacLab$
```

- Ahora instalamos los paquetes de `Isaac Sim` y esperamos a que se instale la dependencia

```
pip install "isaacsim[all,extscache]==5.1.0" --extra-index-url
https://pypi.nvidia.com
```

- Ahora vamos a añadir las dependencias más importantes, esto es importante, ya que la siguiente dependencia va de la mano con la **GPU** , que es lo mas importante, la versión del `#CUDA` , como podemos conocer la versión del `#CUDA` , ingresando lo siguiente en la terminal

```
nvidia-smi
```



- Con base a la verificación del #CUDA modificaremos el siguiente comando de instalacion para la #Dependencia, modificando la parte `/cu128` a `/cu130`

```
pip install -U torch==2.9.0 torchvision==0.24.0 --index-url
https://download.pytorch.org/whl/cu130
```

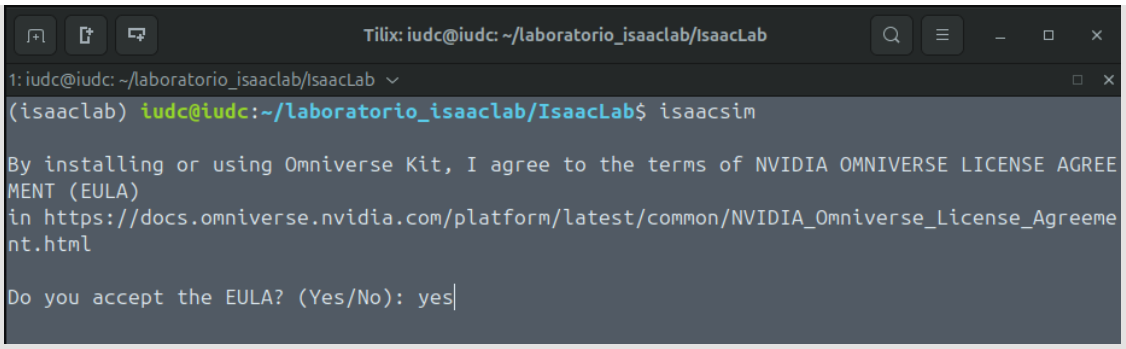
- Para versiones menores de #CUDA

```
pip install -U torch==2.7.0 torchvision==0.22.0 --index-url
https://download.pytorch.org/whl/cu128
```

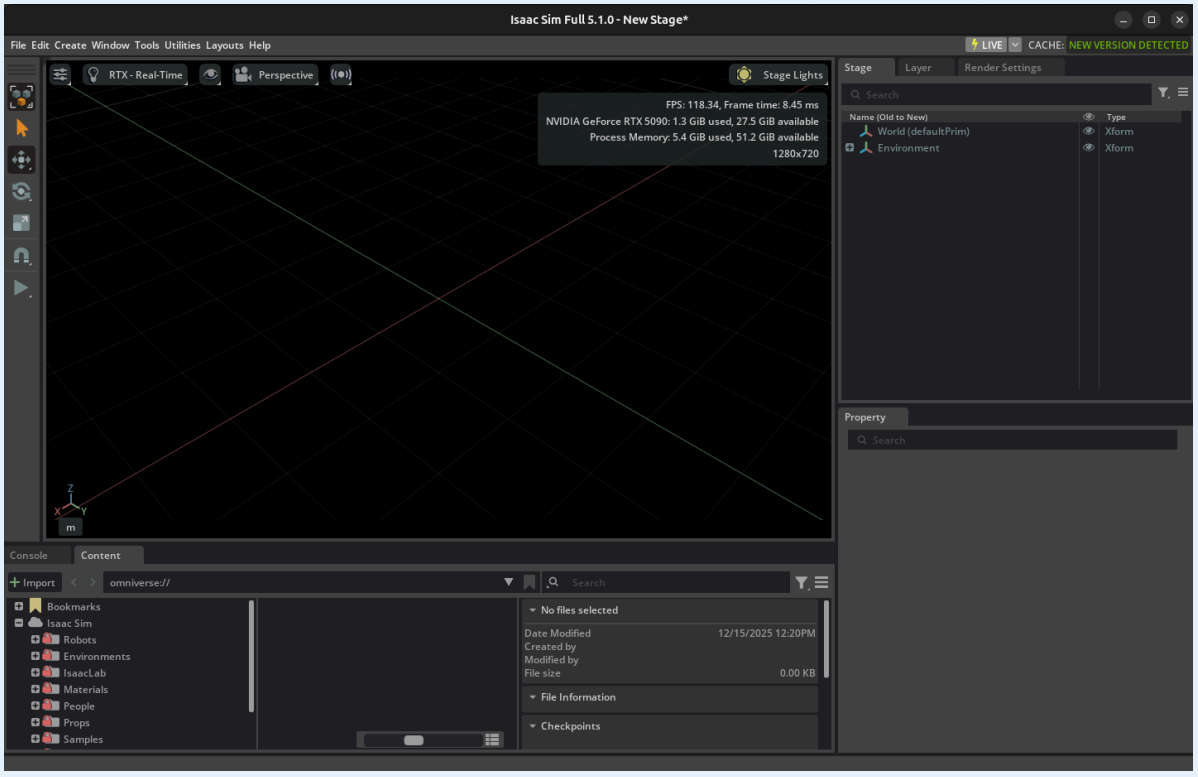
- Ahora vamos a verificar la instalación, ingresamos en la terminal el siguiente comando:

```
isaacsim isaacsim.exp.full.kit
```

> ⚠ **Atención**
> Nos pedira una confirmación de intalación de un kit, el cuál confirmaremos ingresando `yes`



- Esperamos, normalmente nos estara pidiendo que forcemos la salida ya que es muy pesado el software, aun asi teniendo lo minimo una **GPU 5090**, daremos continuamente a `Esperar/Wait`, hasta el punto de que no nos pida *forzar la salida*.

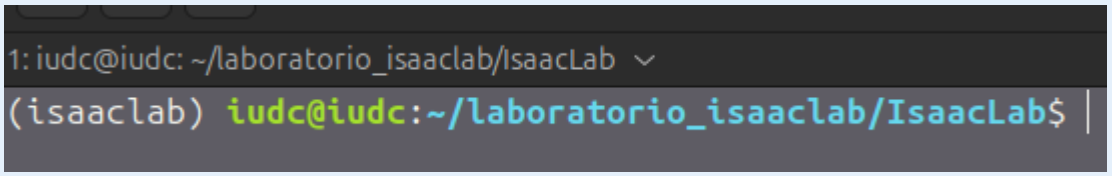


Instalación de las herramientas de #Isaac-Lab

- Instalación de dependencias usando `apt`

```
sudo apt install cmake build-essential
```

- Continuando en la terminal, ubicado en `~/laboratorio_isaaclab/Isaaclab` con el entorno de #Conda activado:



- Ahora vamos a correr la instalación con el siguiente comando y esperamos a que finalice, puede tomar:

```
./isaaclab.sh -i
```

Nos instalara las extensiones de #Isaac-Lab y para la estructura de entrenamiento por refuerzo.

Con esto finalizamos la instalación de #Isaac-Lab

Comando de ejecución

- En esta sección se realizara demostración de los comandos que se puede usar para, el entorno, tanto entrenamiento y visualización.

Entornos

- El siguiente comando nos permitirá visualizar los entornos activos para su ejecución:

```
./isaaclab.sh -p scripts/environments/list_envs.py
```

```
| 1  | Isaac-Repose-Cube-Allegro-Direct-v0
   | isaaclab_tasks.direct.allegro_hand.allegro_hand_env_cfg:AllegroHandEnvCfg
| 2  | Isaac-Ant-Direct-v0
   | isaaclab_tasks.direct.ant.ant_env:AntEnvCfg
| 3  | Isaac-Velocity-Flat-Anymal-C-Direct-v0
   | isaaclab_tasks.direct.anymal_c.anymal_c_env_cfg:AnymalCFlatEnvCfg
| 4  | Isaac-Velocity-Rough-Anymal-C-Direct-v0
   | isaaclab_tasks.direct.anymal_c.anymal_c_env_cfg:AnymalCRoughEnvCfg
| 5  | Isaac-AutoMate-Assembly-Direct-v0
   | isaaclab_tasks.direct.automate.assembly_env:AssemblyEnvCfg
| 6  | Isaac-AutoMate-Disassembly-Direct-v0
   | isaaclab_tasks.direct.automate.disassembly_env:DisassemblyEnvCfg
| 7  | Isaac-Cart-Double-Pendulum-Direct-v0
   | isaaclab_tasks.direct.cart_double_pendulum.cart_double_pendulum_env:CartDoublePe
| 8  | Isaac-Cartpole-Direct-v0
   | isaaclab_tasks.direct.cartpole.cartpole_env:CartpoleEnvCfg
| 9  | Isaac-Cartpole-RGB-Camera-Direct-v0
   | isaaclab_tasks.direct.cartpole.cartpole_camera_env:CartpoleRGBCameraEnvCfg
| 10 | Isaac-Cartpole-Depth-Camera-Direct-v0
   | isaaclab_tasks.direct.cartpole.cartpole_camera_env:CartpoleDepthCameraEnvCfg
| 11 | Isaac-Cartpole-Showcase-Box-Box-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:BoxBoxEnvCfg
| 12 | Isaac-Cartpole-Showcase-Box-Discrete-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:BoxDiscreteEnv
| 13 | Isaac-Cartpole-Showcase-Box-MultiDiscrete-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:BoxMultiDiscre
| 14 | Isaac-Cartpole-Showcase-Discrete-Box-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:DiscreteBoxEnv
| 15 | Isaac-Cartpole-Showcase-Discrete-Discrete-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:DiscreteDiscre
| 16 | Isaac-Cartpole-Showcase-Discrete-MultiDiscrete-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:DiscreteMultiD
| 17 | Isaac-Cartpole-Showcase-MultiDiscrete-Box-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:MultiDiscreteB
| 18 | Isaac-Cartpole-Showcase-MultiDiscrete-Discrete-Direct-v0
   | isaaclab_tasks.direct.cartpole_showcase.cartpole.cartpole_env_cfg:MultiDiscreteD
| 19 | Isaac-Cartpole-Showcase-MultiDiscrete-MultiDiscrete-Direct-v0
```

Son `152` entornos activos en total.

```
| 152  | Isaac-Navigation-Flat-Anymal-C-Play-v0                                  | isaac
   | isaaclab_tasks.manager_based.navigation.config.anymal_c.navigation_env_cfg:NavigationE
+------+------------------------------------------------------------------------+------
```

> **⚠ Importante**
>
> Al visualizar los entornos, podemos ver que para cada uno, hay dos entornos, uno para entrenar y otro para visualizar, para proximos ejemplos nos enfocaremos en el del `Unitree-G1`

```
| 86  | Isaac-Velocity-Rough-G1-v0                                              | isaac
   | isaaclab_tasks.manager_based.locomotion.velocity.config.g1.rough_env_cfg:G1RoughEnvCfg
| 87  | Isaac-Velocity-Rough-G1-Play-v0                                         | isaac
   | isaaclab_tasks.manager_based.locomotion.velocity.config.g1.rough_env_cfg:G1RoughEnvCfg
| 88  | Isaac-Velocity-Flat-G1-v0                                               | isaac
   | isaaclab_tasks.manager_based.locomotion.velocity.config.g1.flat_env_cfg:G1FlatEnvCfg
| 89  | Isaac-Velocity-Flat-G1-Play-v0                                          | isaac
```

Ejecución de entrenamiento

- Para la ejecución necesitamos seleccionar un entorno de la anterior imagen, `Flat` que es terreno plano, y `Rough` es terreno random.
- Comando de ejecución para el entrenamiento:

```
./isaaclab.sh -p scripts/reinforcement_learning/rsl_rl/train.py --task Isaac-Velocity-Flat-G1-v0 --headless
```

⚠ Importante

El `flag/bandera` que es `--headless`, nos permite entrenar sin visualizacion, solo ver en la terminal iteraciones

```
################################################################################
                      Learning iteration 1/1500

                       Computation: 144353 steps/s (collection: 0.644s, learning 0.037s)
               Mean action noise std: 1.00
          Mean value_function loss: 3.6480
               Mean surrogate loss: -0.0029
                 Mean entropy loss: 52.5727
                       Mean reward: -6.27
               Mean episode length: 47.02
Episode_Reward/track_lin_vel_xy_exp: 0.0063
Episode_Reward/track_ang_vel_z_exp: 0.0023
       Episode_Reward/lin_vel_z_l2: -0.0022
      Episode_Reward/ang_vel_xy_l2: -0.0543
     Episode_Reward/dof_torques_l2: -0.0022
         Episode_Reward/dof_acc_l2: -0.0049
     Episode_Reward/action_rate_l2: -0.0138
      Episode_Reward/feet_air_time: 0.0002
Episode_Reward/flat_orientation_l2: -0.0129
     Episode_Reward/dof_pos_limits: -0.0003
Episode_Reward/termination_penalty: -0.1956
         Episode_Reward/feet_slide: -0.0040
Episode_Reward/joint_deviation_hip: -0.0026
Episode_Reward/joint_deviation_arms: -0.0029
Episode_Reward/joint_deviation_fingers: -0.0018
Episode_Reward/joint_deviation_torso: -0.0008
Metrics/base_velocity/error_vel_xy: 0.0859
Metrics/base_velocity/error_vel_yaw: 0.4277
      Episode_Termination/time_out: 0.0270
  Episode_Termination/base_contact: 0.4604
--------------------------------------------------------------------------------
                   Total timesteps: 196608
                    Iteration time: 0.68s
                      Time elapsed: 00:00:02
                               ETA: 00:29:23
```

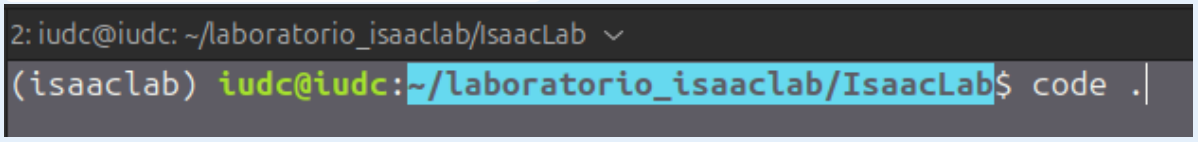
> ¿Porqué?
>
> - Al contar con el hardware minimo no podemos poder visualizar el entorno con la funcionalidad completa, entonces solo podemos esperar a que finalice el entrenamiento. Al usar el #Hardware minimo, el porcentaje de uso de la *GPU* es mayor al `75%`, se sobre calienta.

```
|=========================================+========================+======================|
|   0  NVIDIA GeForce RTX 5090        Off |   00000000:01:00.0  On |                  N/A |
| 30%   58C    P1            220W /  575W |    5843MiB /  32607MiB |     79%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                      GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A            2751      G   /usr/lib/xorg/Xorg                      246MiB |
|    0   N/A  N/A            3290      G   /usr/bin/gnome-shell                     43MiB |
|    0   N/A  N/A            3957      G   ...exec/xdg-desktop-portal-gnome          8MiB |
|    0   N/A  N/A            4581      G   /snap/obsidian/53/obsidian               38MiB |
|    0   N/A  N/A            4704      G   .../7477/usr/lib/firefox/firefox        367MiB |
|    0   N/A  N/A            8275      G   /proc/self/exe                           55MiB |
|    0   N/A  N/A           16383      G   /usr/bin/gnome-text-editor               27MiB |
|    0   N/A  N/A           19621    C+G   ...nda3/envs/isaaclab/bin/python       4920MiB |
+-----------------------------------------------------------------------------------------+
```

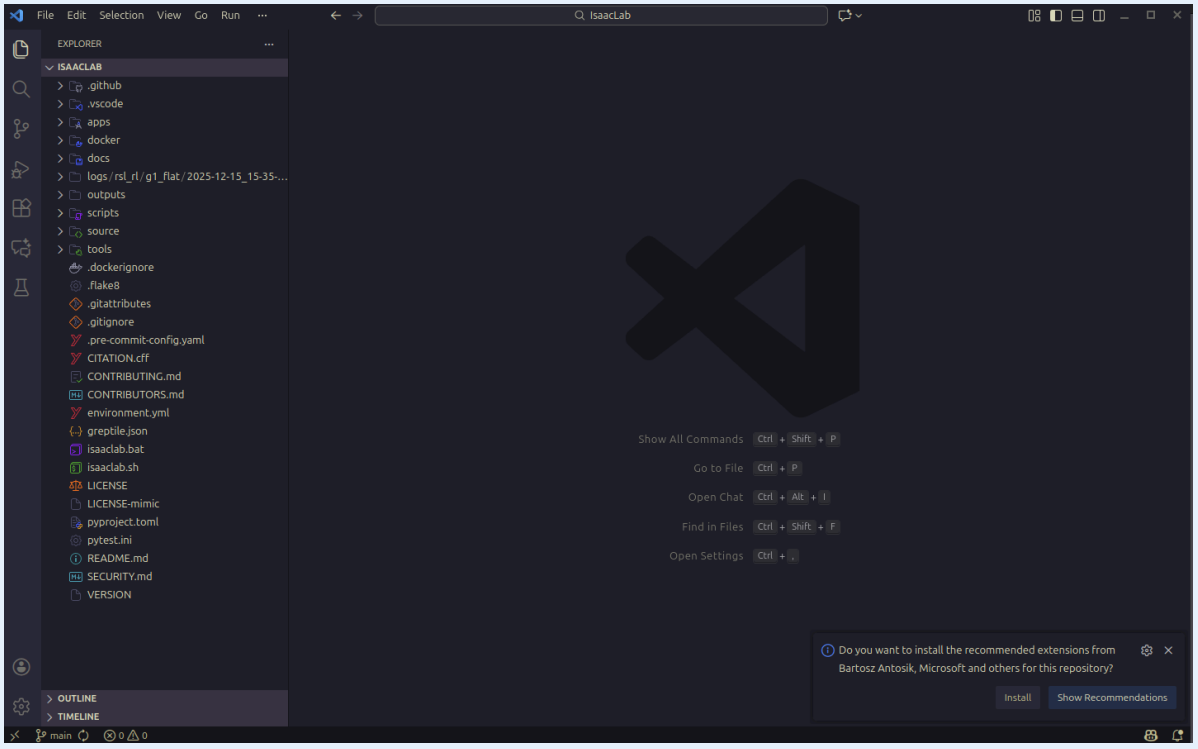
Visualización del entrenamiento

Al ya finalizar el entrenamiento se nos generaran archivos donde estaran puntos de guardado, parametros del entorno usado, para ello usar `VSCode`

- Si ya cuentas con `VSCode` , usa el siguiente comando en la siguiente ubicación `~/laboratorio_isaaclab/IsaacLab`

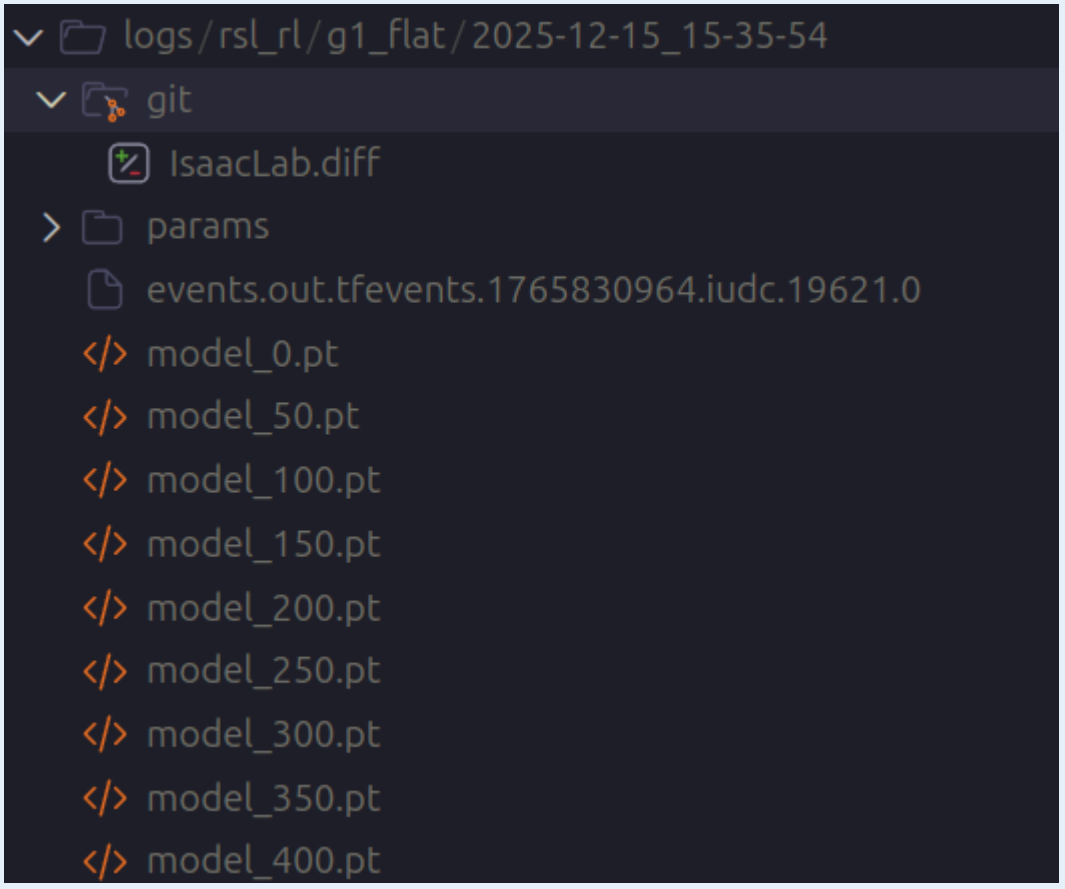


- Visualizaremos lo siguiente



- Al ejecutar el anterior entrenamiento, se nos generaron las siguientes carpetas

- Dentro de la carpeta `logs`, contamos con los siguientes directorios y archivos

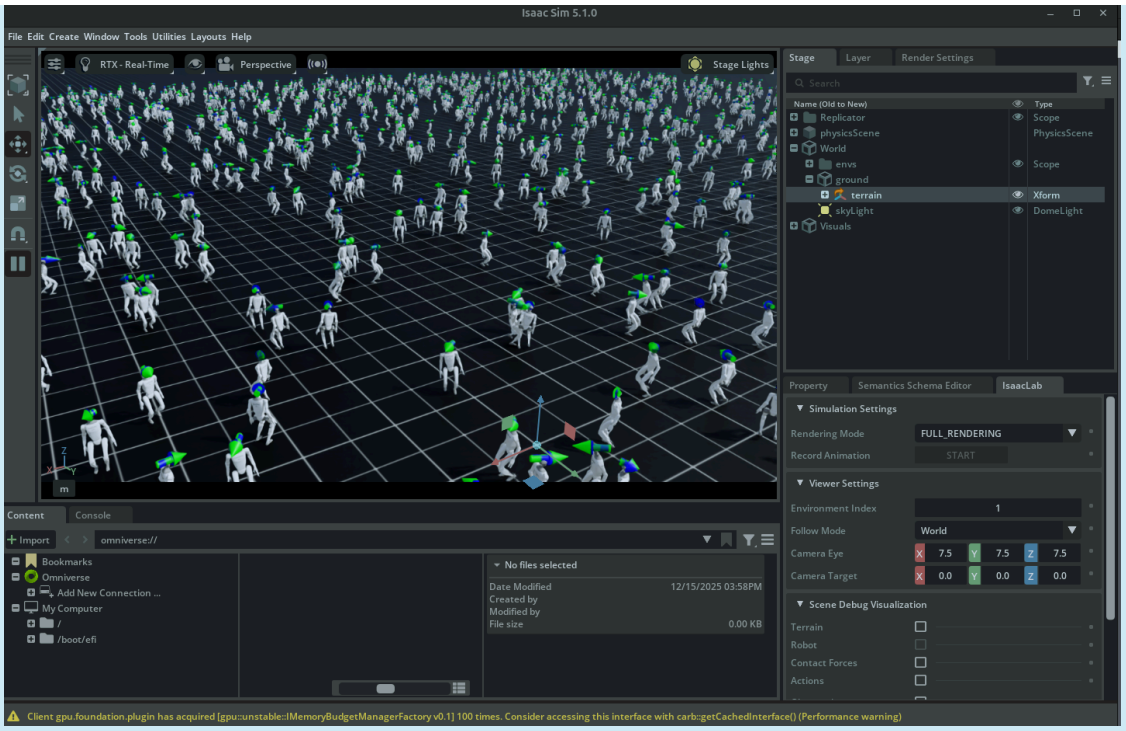


- Lo mas importante son los archivos con la extensión `.pt`, esto nos permitira visualizar el entrenamiento.

Ejecución

- Para la ejecución usaremos el siguiente comando:

```
./isaaclab.sh -p scripts/reinforcement_learning/rsl_rl/play.py
--task Isaac-Velocity-Flat-G1-v0
```



Visualización por checkpoint

```
./isaaclab.sh -p scripts/reinforcement_learning/rsl_rl/play.py
--task Isaac-Velocity-Flat-G1-v0  --checkpoint
logs/rsl_rl/g1_flat/2025-12-15_15-35-54/model_1499.pt
# Comando de ejemplo
```

Con la ruta absoluta

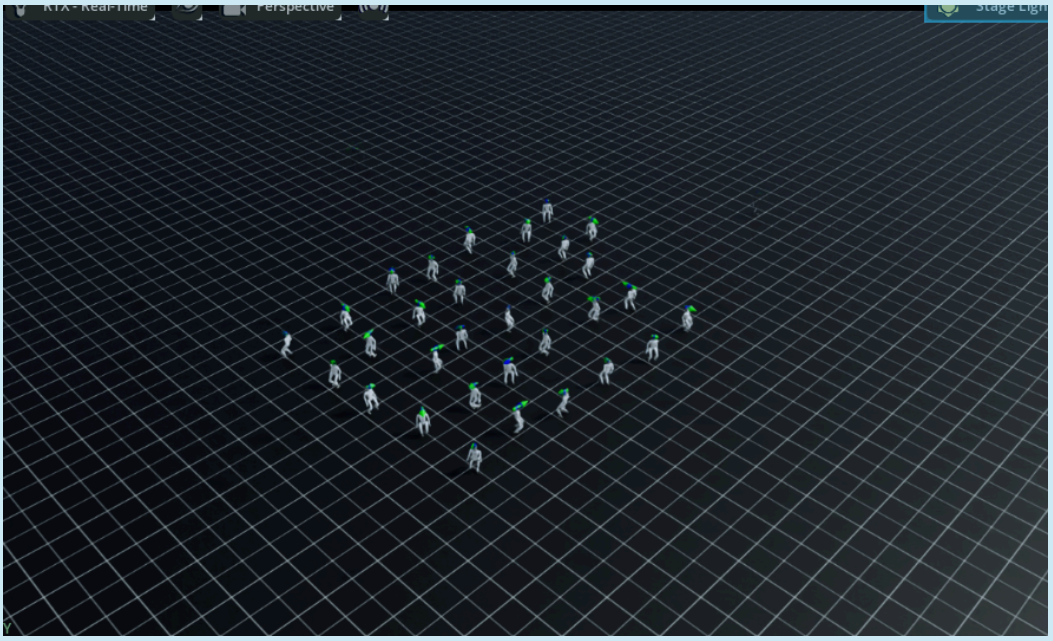
```
./isaaclab.sh -p scripts/reinforcement_learning/rsl_rl/play.py
--task Isaac-Velocity-Flat-G1-v0  --checkpoint
/home/iudc/laboratorio_isaaclab/IsaacLab/logs/rsl_rl/g1_flat/2
025-12-15_15-35-54/model_1499.pt
```

Visualizacion con un numero N de robots

- Usar el siguiente comando

```
./isaaclab.sh -p scripts/reinforcement_learning/rsl_rl/play.py
--task Isaac-Velocity-Flat-G1-v0 --num_evs 32 --checkpoint
logs/rsl_rl/g1_flat/2025-12-15_15-35-54/model_1499.pt
```

- Esto nos permite visualizar con mayor fluidez



Cosas importantes

1. Al realizar un entrenamiento nos genera archivos con la extension `.pt`, esos archivos los podemos usar para integrar al robot en la vida real y en simulación, pero lo complicados es la parte de la configuración. Ya que puede variar para el robot, pero el robot `Unitree-G1`, en la vida real ya tiene configrado estabilizador y modos de caminata, ya sea integración en simulación, necesta tener en cuenta, ya que la politica se configura para un numero especifico de grados de libertad (12,23,27,29...) y no todos funcionan tal cuál para cada robot.
2. Todo el entorno anterior funciona unicamente con hardware de #NVidia.
3. El hardware usado para el desarrollo de la documentación es lo minimo para poder ejecuar #Isaac-Lab con #Isaac-sim .