ISAAC GYM

Viktor Makoviichuk, 03.19.19

SIMULATION IN ROBOTICS

- Limited access to hardware
- Well-controlled experiments
- Good progress recently in Sim2Real

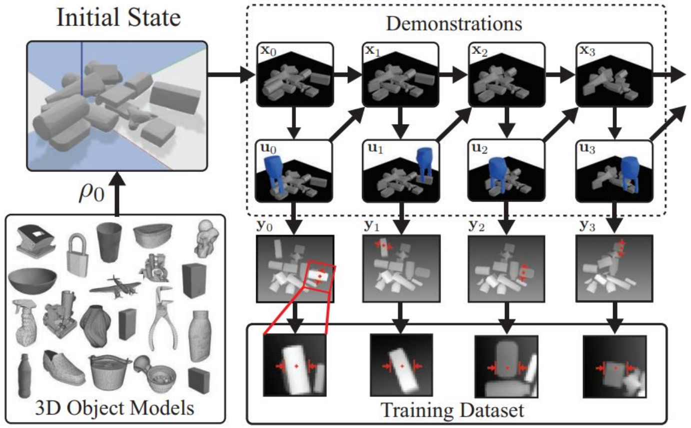
Quadruped Locomotion

Jemin et al, 2019
ETH

Dexterous Manipulation

OpenAI, 2018

Grasping in Clutter

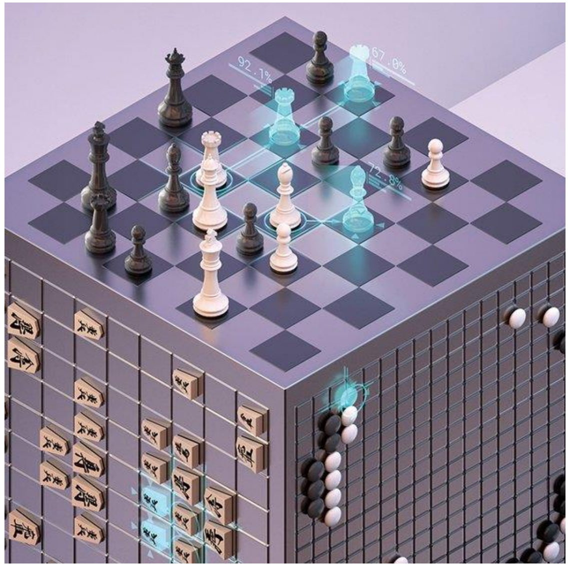


Mahler and Goldberg, 2017
UC Berkeley

MOTIVATION

Reinforcement Learning

AlphaZero



Deepmind, 2018

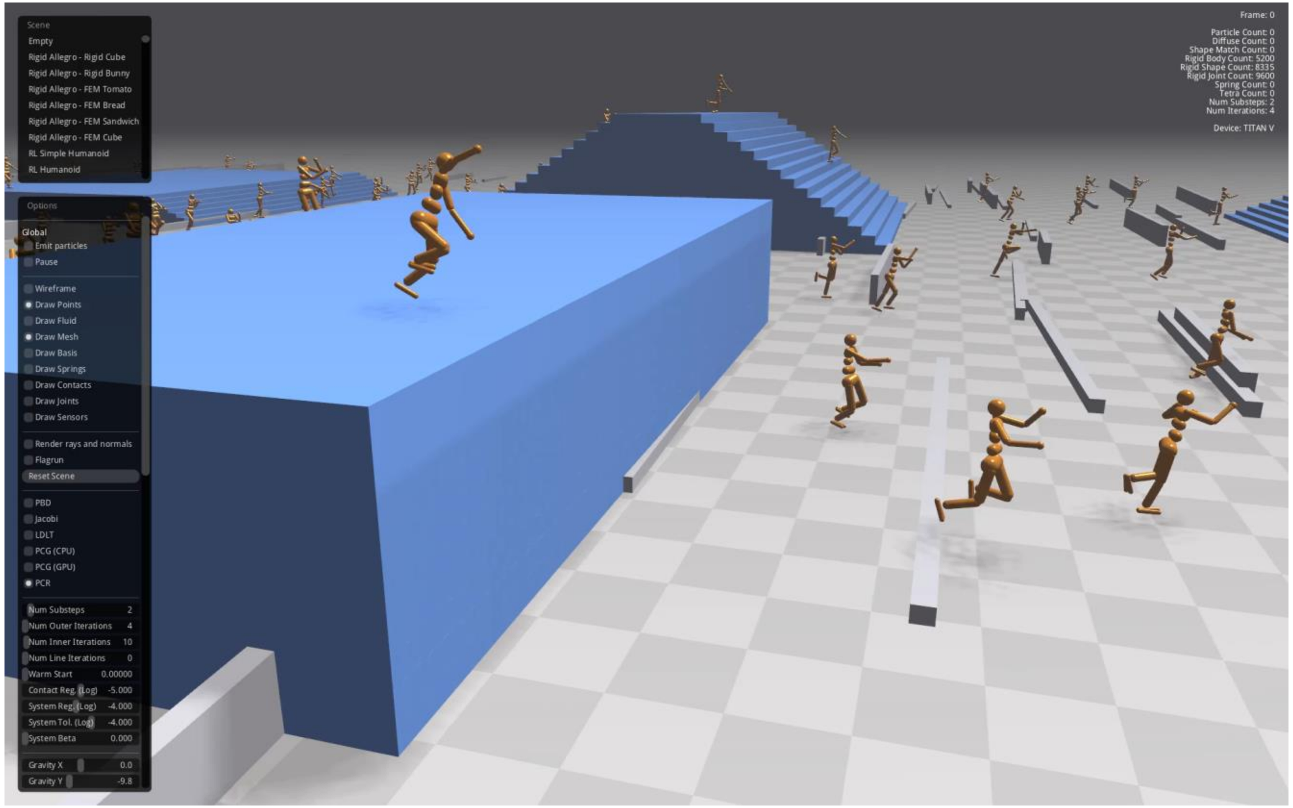
OpenAI Five

OpenAI, 2018

APPLICATIONS

Reinforcement Learning

Locomotion/Animation

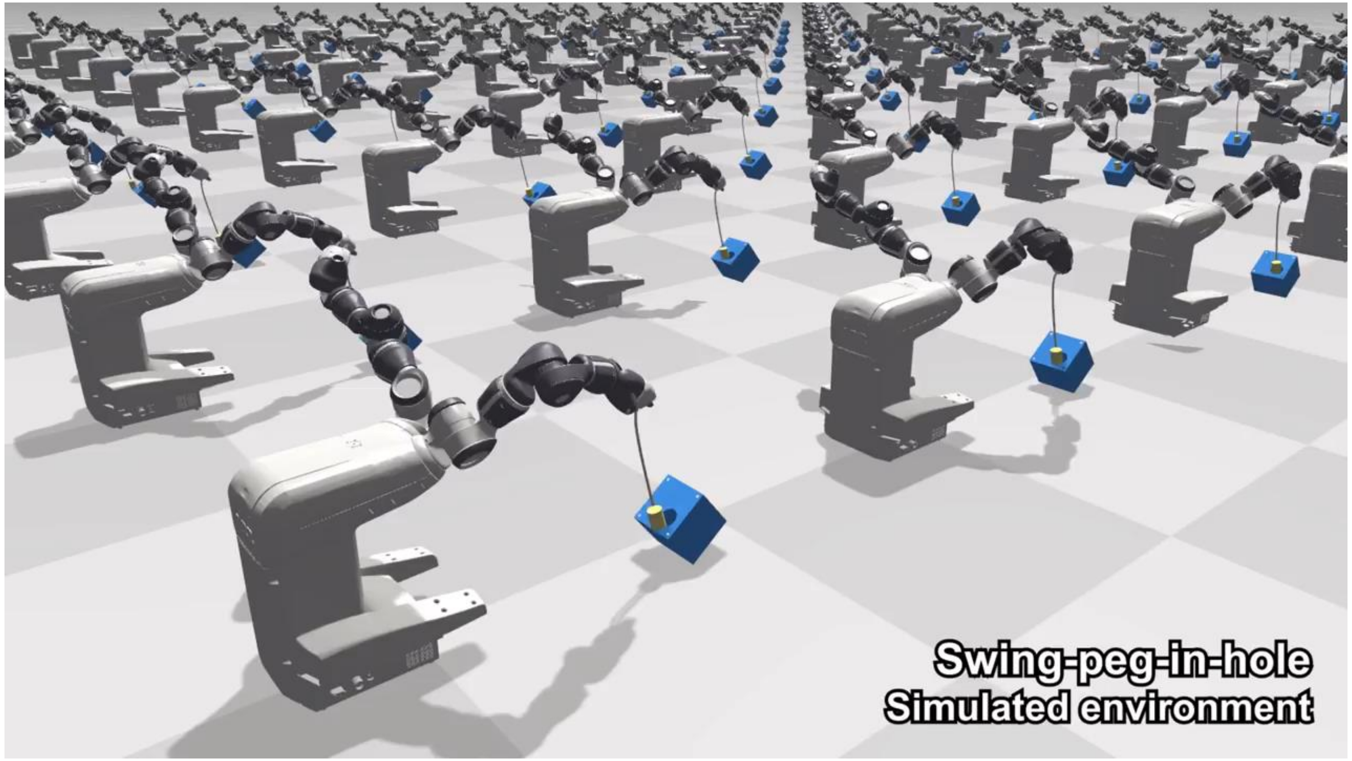


Liang, Makoviychuk, Handa et al, 2018
NVIDIA

APPLICATIONS

Robotics

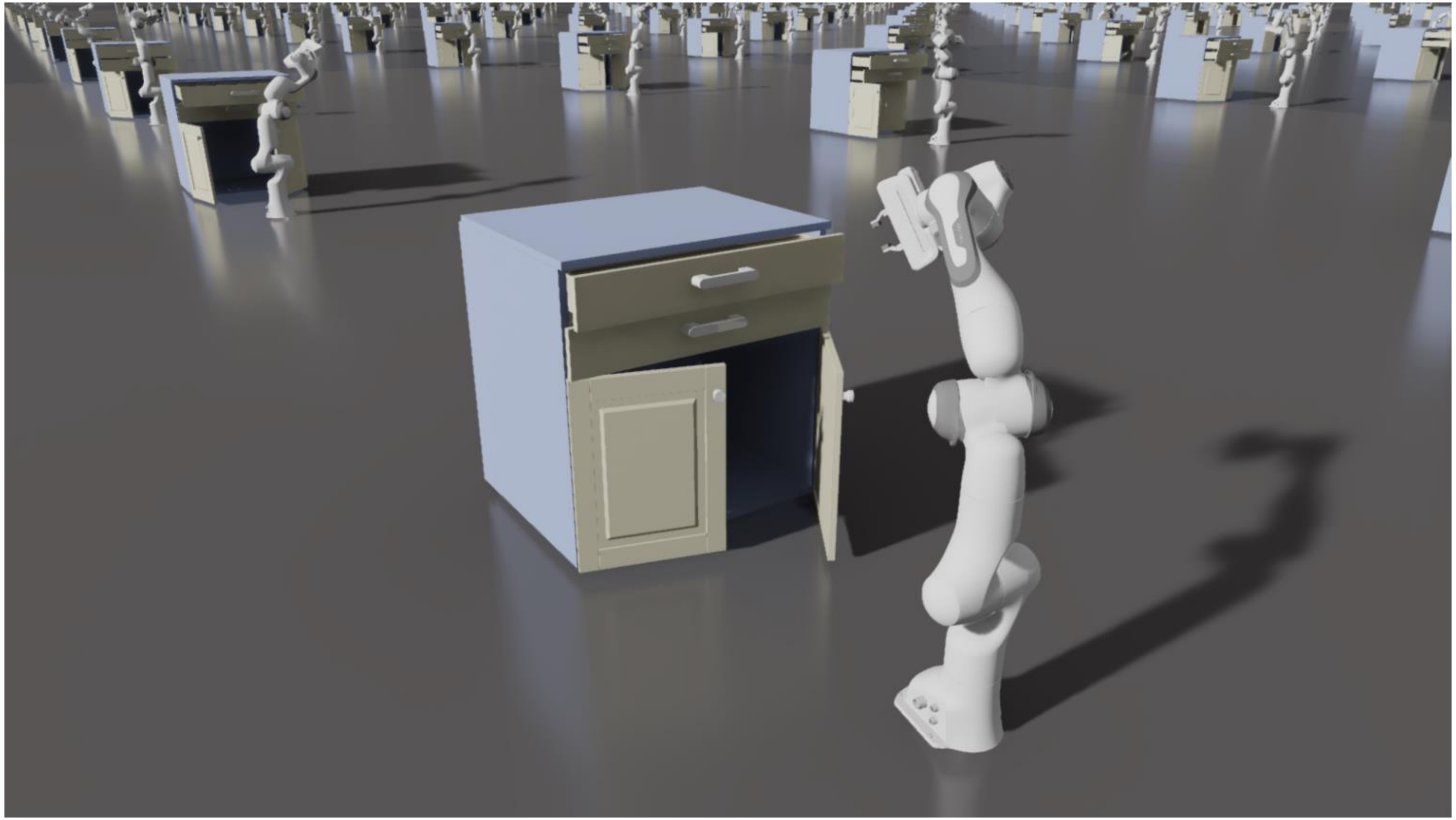
Sim2Real Robotics



Chebotar, Handa, Makoviychuk, et al, 2018
NVIDIA

ISAAC GYM

Platform for high-performace AI Learning Experiments



ISAAC GYM

Key Goals

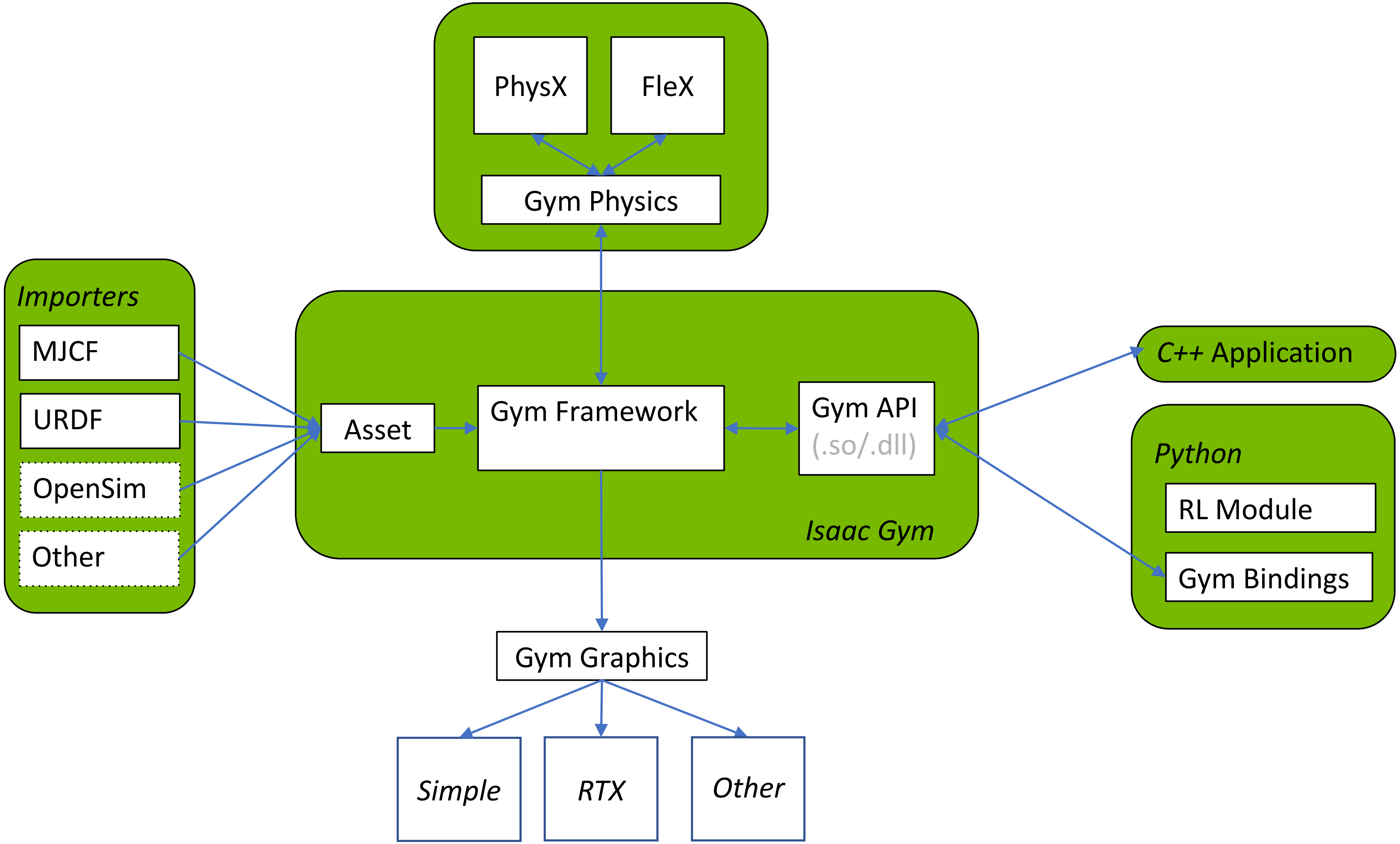
- Simple to get started
- Procedural API for scene and model definition
- Performance (from Python) and scalability
- Fast, high-fidelity physics/multi-physics
- Fast, high-quality image generation
  - Visualization and camera sensors, fast multi-camera rendering
- Decoupling of graphics/physics
- Learning algorithm/framework agnostic

ISAAC GYM

Key Features

- Multiple physics backends
- Multiple rendering backends
- Support for multiple robot definition formats
- Many environments simulated in parallel
- Scalable:
  - Many simple/single-agent environments
  - Complex/multi-agent environments

ISAAC GYM



PhysX
FleX
Gym Physics
Importers
MJCF
URDF
OpenSim
Other
Asset
Gym Framework
Gym API
(.so/.dll)
Isaac Gym
C++ Application
Python
RL Module
Gym Bindings
Gym Graphics
Simple
RTX
Other

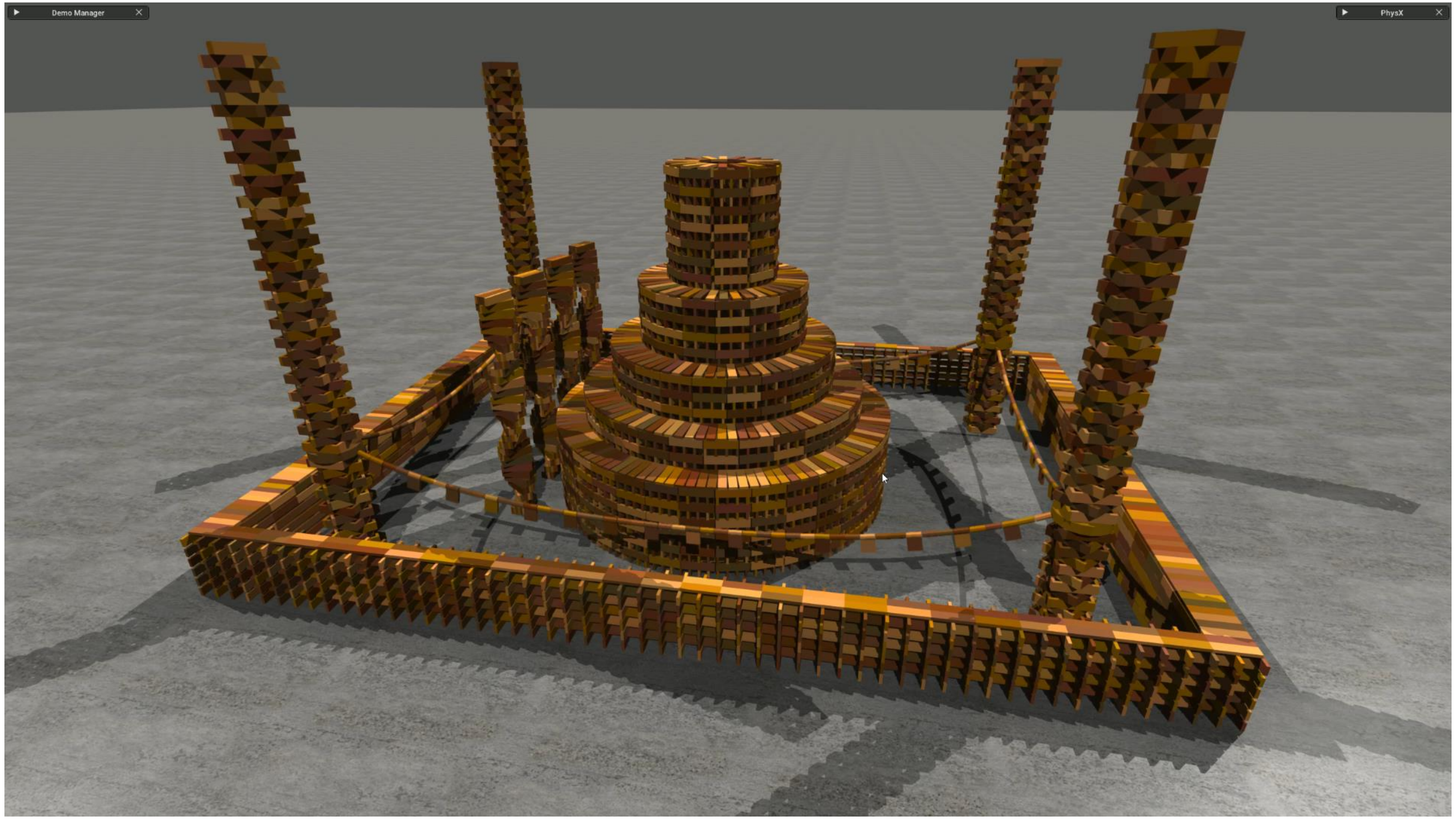
PHYSICS

PhysX

- New 4.x version for robotics, reinforcement learning and engineering applications
- Maximal coordinate representation and articulations
- Performance and scalability
    - From small training environments to large city-scale worlds
- CPU and GPU simulation

PHYSICS

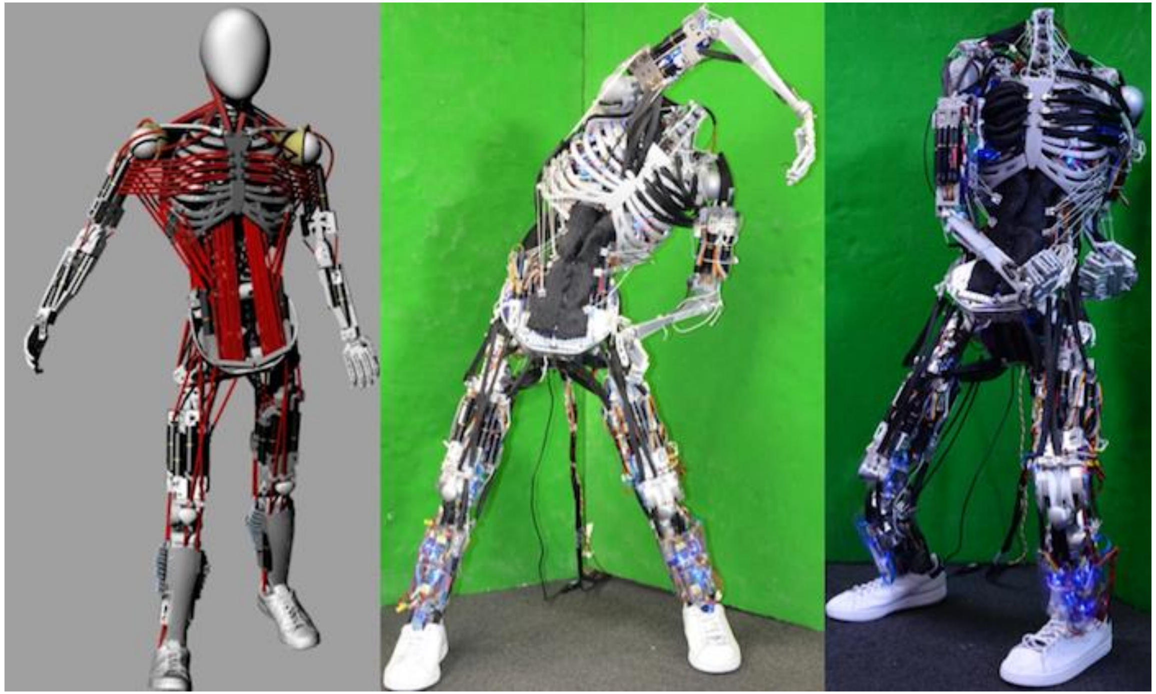
PhysX



PHYSICS

FleX

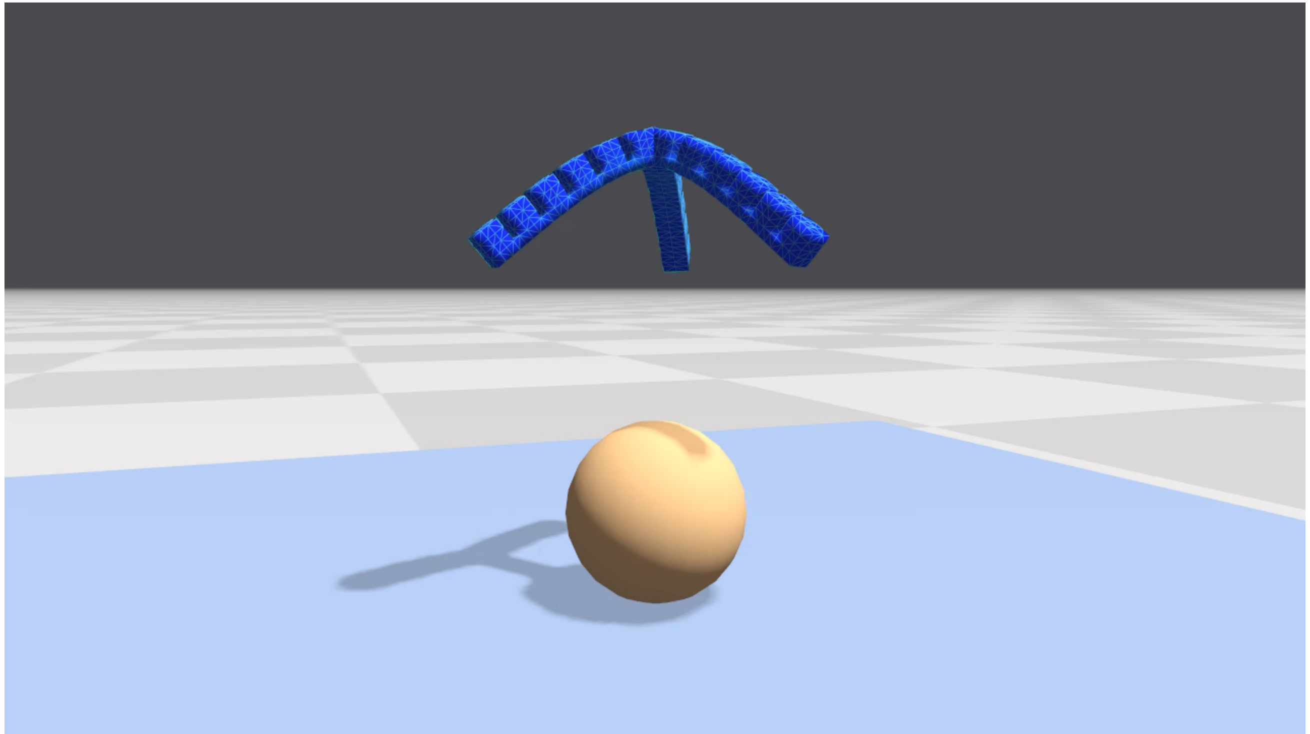
- Research features backends
- Only GPU simulation
- New Newton solver
- Multi-physics



PHYSICS

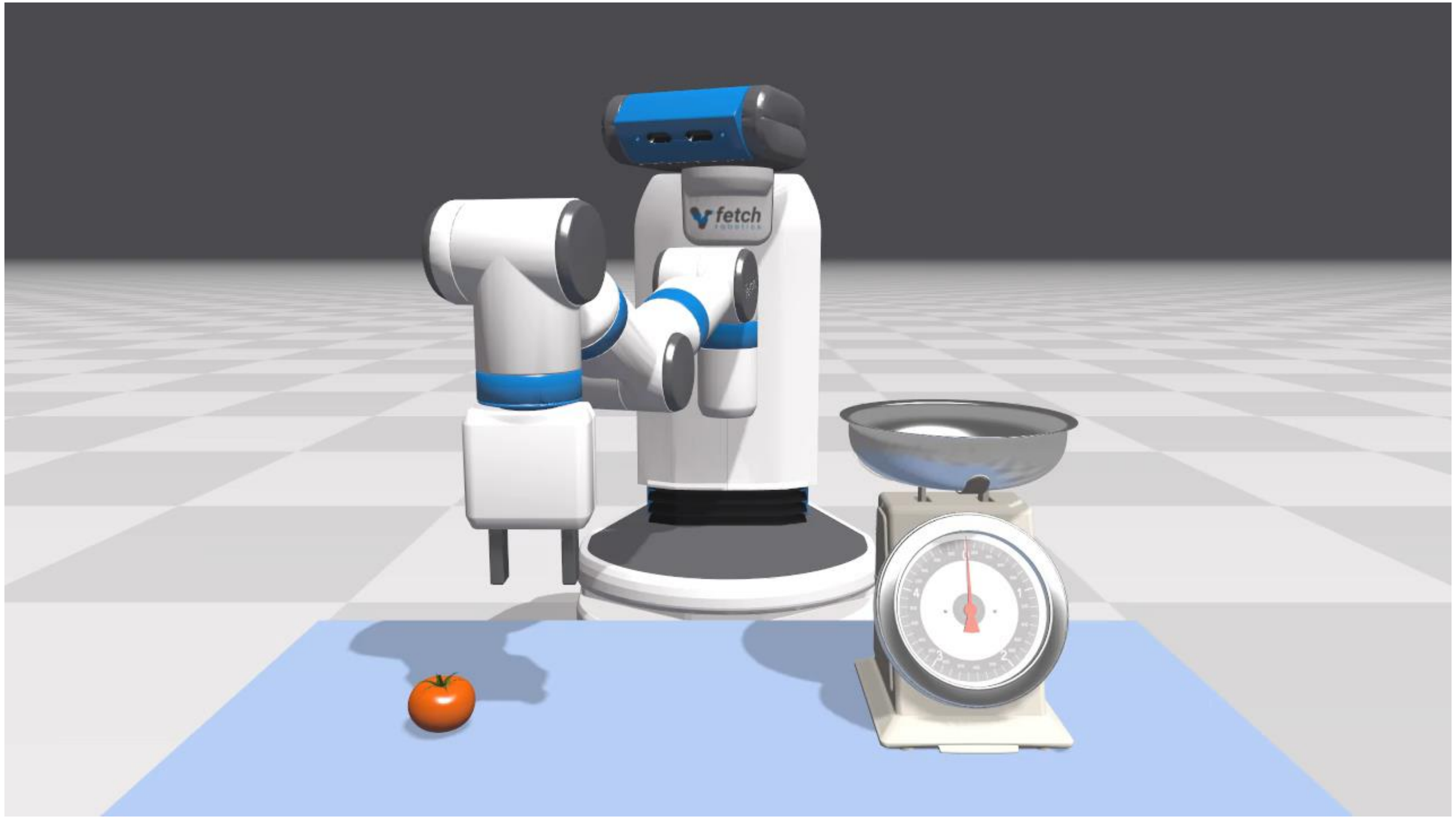
FleX

- Multi-physics
  - Rigid and FEM soft bodies
  - Cloth, ropes
  - Liquids
  - Two-way coupling and force propagation between different phases



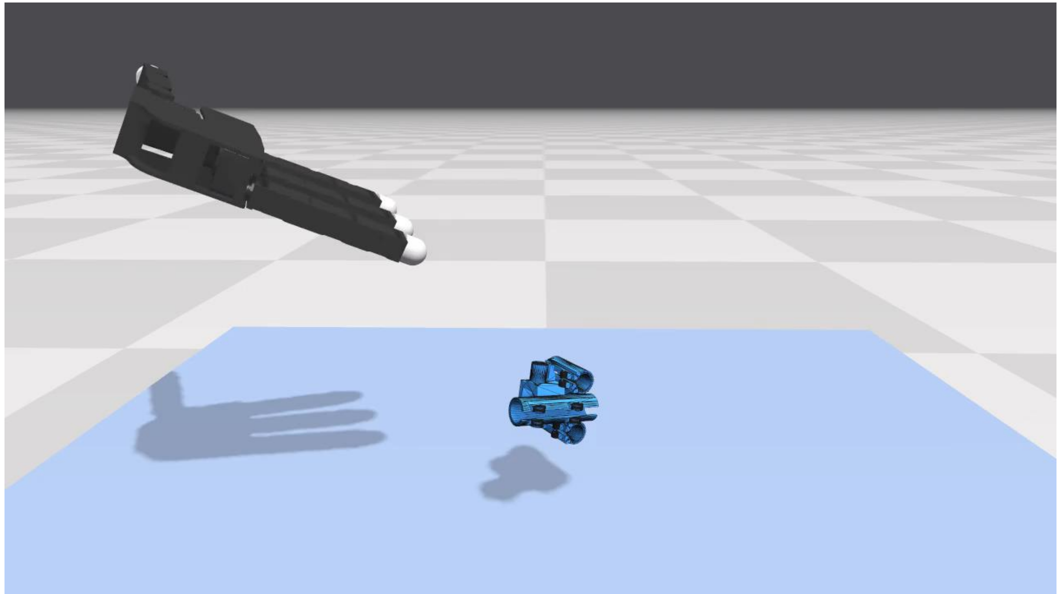
PHYSICS

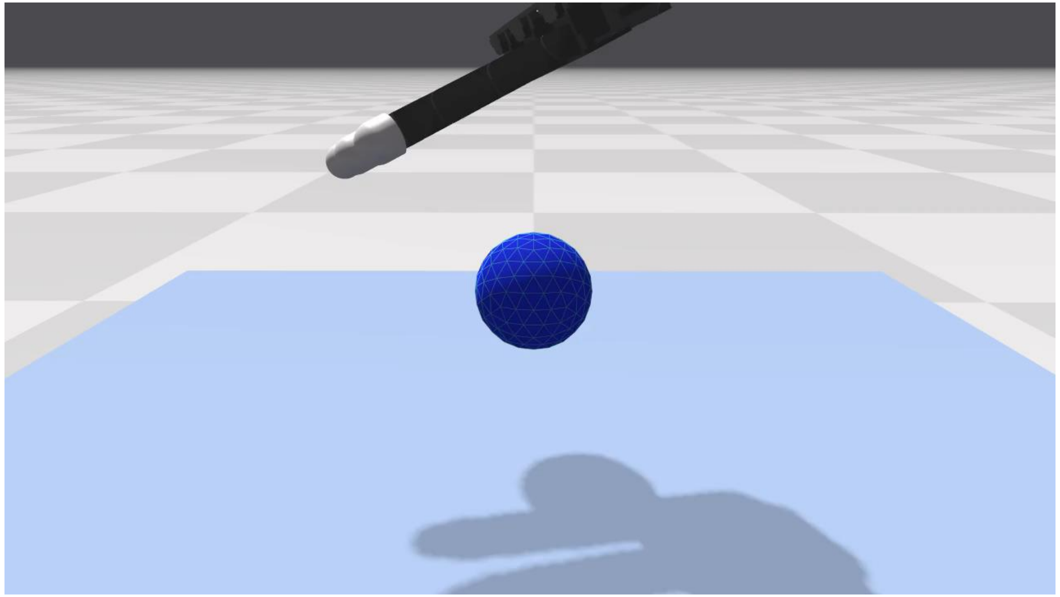
FleX



PHYSICS

FleX



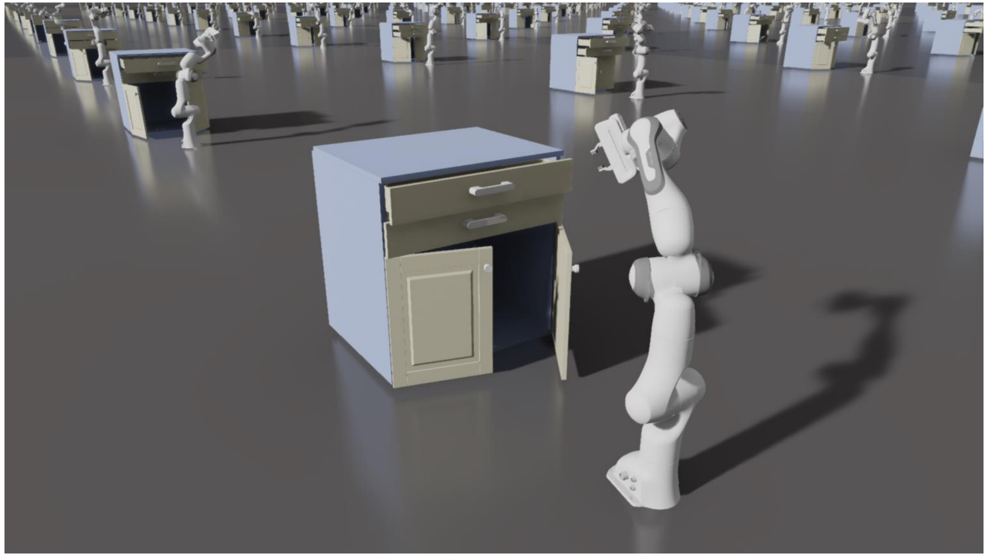


NVIDIA

RENDERING

Multiple Rendering Backends

- Vulkan-based Raster
  - Fast raster graphics
  - Simple materials and lights
- RTX-based Ray-Tracing
  - High-fidelity hardware-accelerated ray tracing
  - Support for MDL/complex materials
  - Area lights, ambient occlusion, reflections, refraction



RENDERING

Camera Sensors

- Camera sensors
  - Free control
  - Fixed
  - Attach to bodies
- Render to image buffers
  - Input to visual learning algorithms
  - Output
- High Performance – thousands of images per second

USING ISAAC GYM

- Use native C++ API or python bindings
- Scalable execution:
    - Single laptop/desktop
    - Cluster
- Remote viewer to visualize results of training on server/cluster
- Includes example environments / experiments

USING ISAAC GYM

```
from isaacgym import gymapi

# initialize gym
gym = gymapi.acquire_gym()

# create a viewer (optional)
viewer = gym.create_viewer(None, 1920, 1080);

# load asset
robot_asset = gym.load_asset("../assets", “franka.urdf")

# create a simulation
sim = gym.create_sim()
# get default sim params
params = gymapi.SimParams()
gym.get_sim_params(sim, params)
# set custom sim params
params.gravity = gymapi.Vec3(0.0, -9.8, 0.0)
params.solver_type = 5
params.num_outer_iterations = 4
params.num_inner_iterations = 10
params.relaxation = 0.75
params.warm_start = 0.5
gym.set_sim_params(sim, params)
```

USING ISAAC GYM

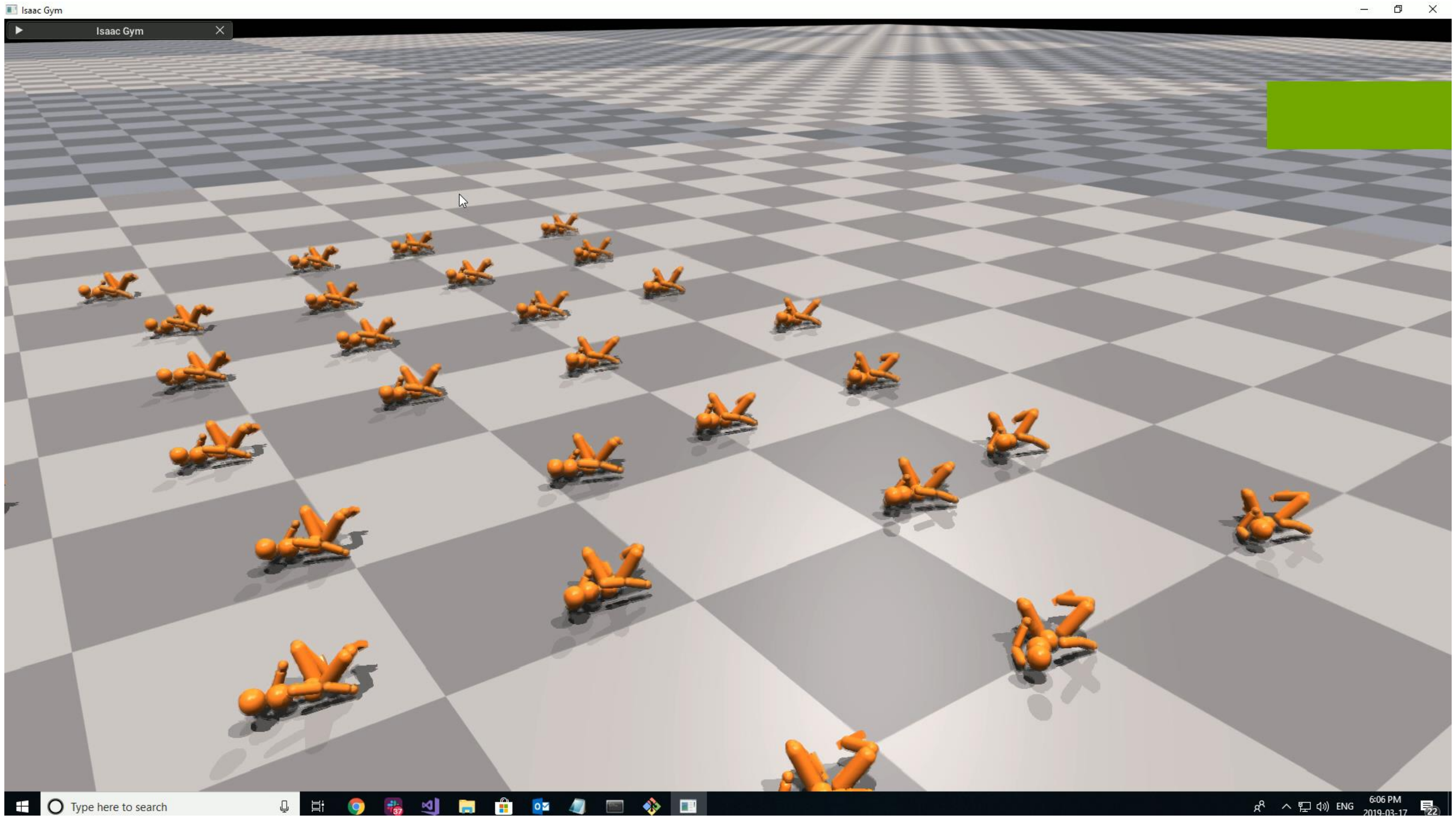
```
# specify number of envs in the simulation
# - multiple envs can be stepped in parallel
num_envs = 1024

# specify environment spacing and bounds
spacing = 2.0
lower = gymapi.Vec3(-spacing, 0.0, -spacing)
upper = gymapi.Vec3(spacing, spacing, spacing)

# initialize an array of environments using a procedural API
# - easy to randomize properties
for i in range(num_envs):
   # create env
   env = gym.create_env(sim, lower, upper)

   # add actor
   pose = gymapi.Transform(gymapi.Vec3(0.0, 2.0, 0.0), gymapi.Quat(-0.707107, 0.0, 0.0, 0.707107))
   gym.create_actor(env, robot_asset, pose, “franka")

# set some simulation parameters
dt = 1.0 / 60.0
num_substeps = 2
```

USING ISAAC GYM

```
# main loop
while not gym.query_viewer_has_closed(viewer):
   for i in range(num_envs):

       torque = 20.0
       # get some useful handles (this can be done before the main loop)
       env = gym.get_env(sim, i)
       joint3_handle = gym.get_joint_handle(env, “franka", “panda_joint3")

       # apply efforts to individual joints
       gym.apply_joint_effort(env, “panda_joint3, torque)

   # step the simulation
   gym.simulate(sim, dt, num_substeps)
   gym.fetch_results(sim, True)

  # update the viewer
  gym.step_graphics(sim)
  gym.draw_viewer(viewer, sim, True)

  # Wait for dt to elapse in real time.
  # This synchronizes the physics simulation with the rendering rate.
  gym.sync_frame_time(sim)
```

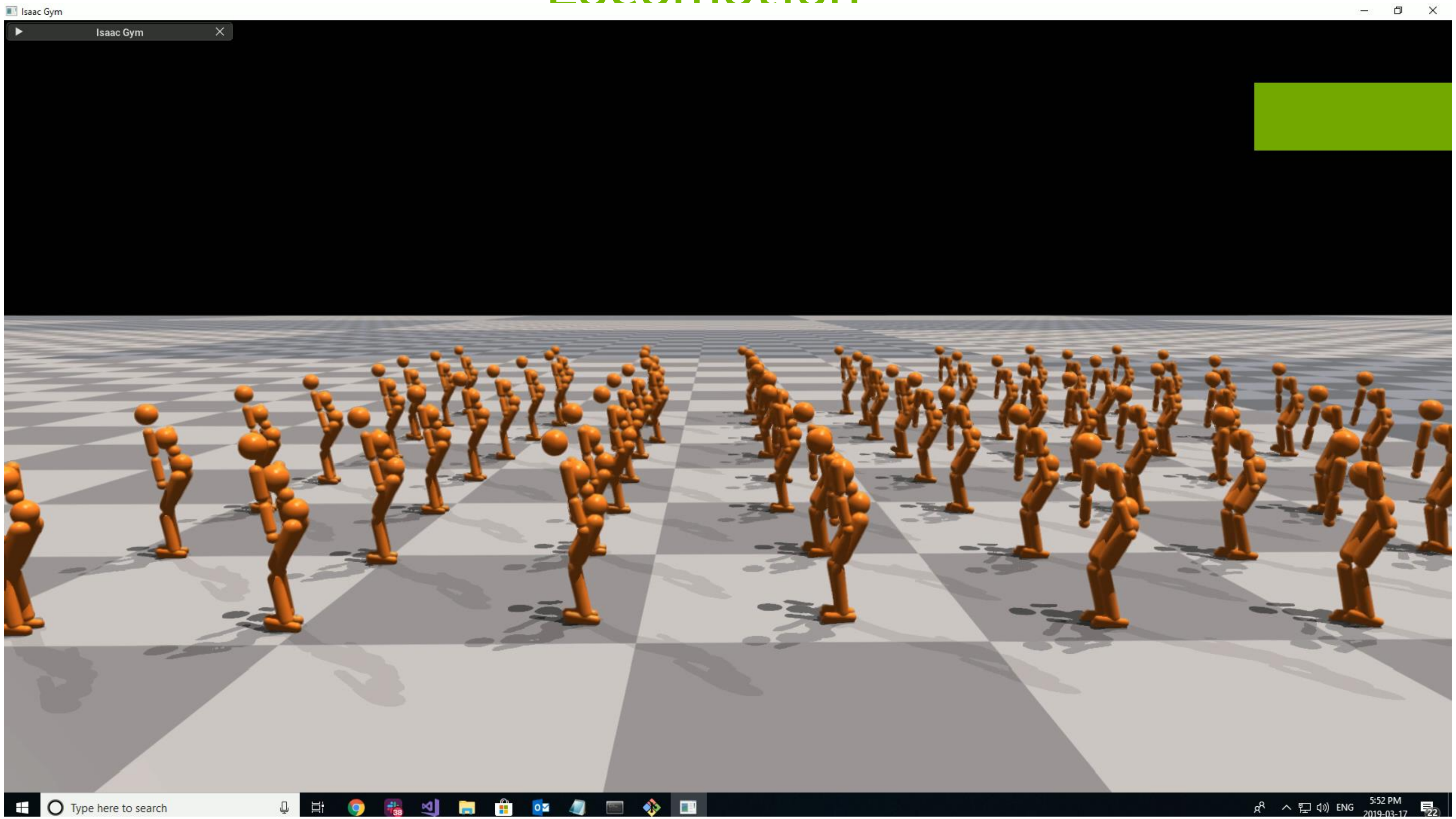
EXAMPLES

Locomotion



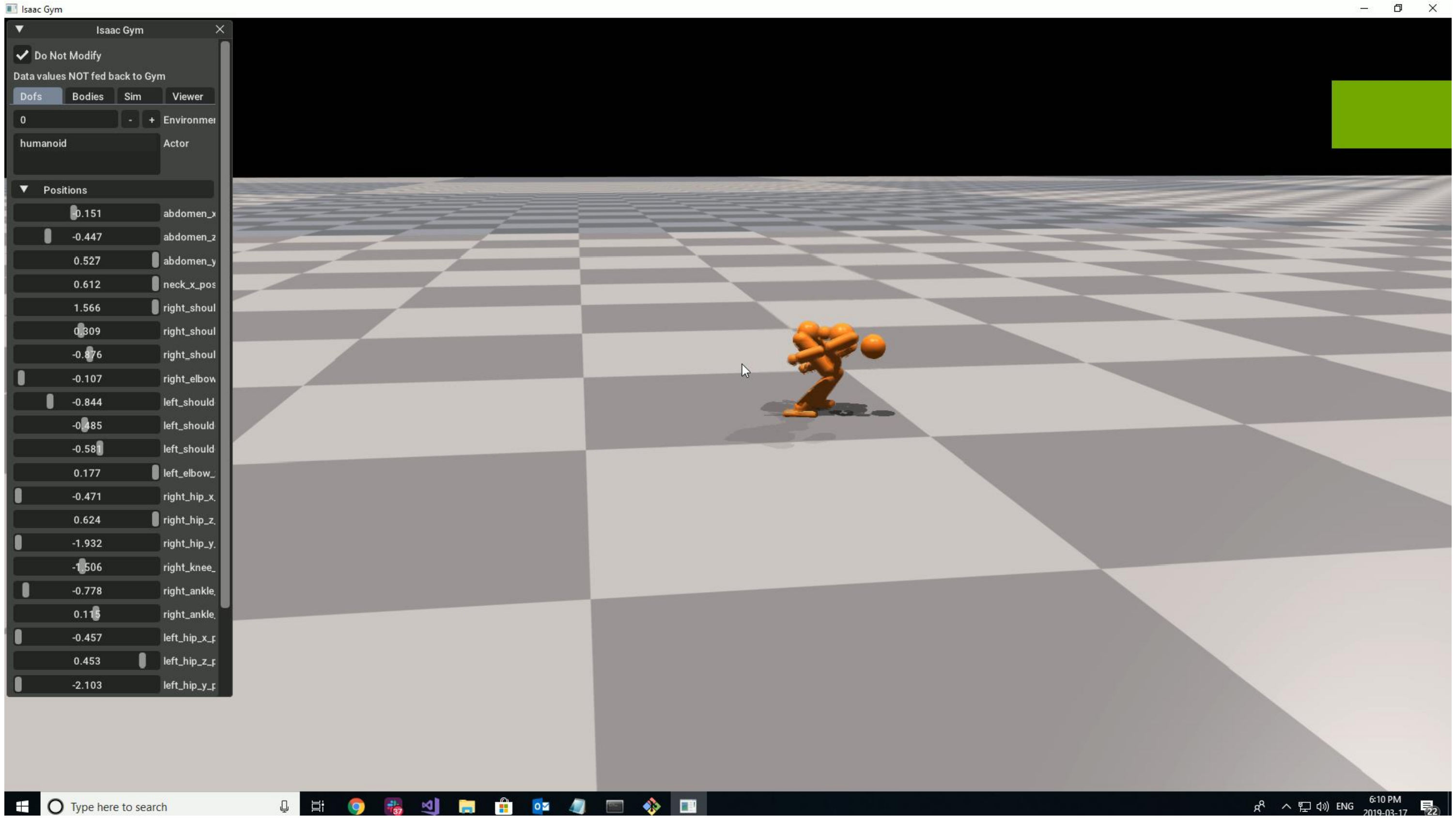
EXAMPLES

Locomotion



EXAMPLES

Locomotion



EXAMPLES

Robotics

- Trained using RL in Isaac Gym
- RTX renderer, raytraced reflections and refractions

WHAT'S NEXT?

- Further performance optimization:
  - GPU observations and control
  - No-copy communication of camera image to learning framework
  - More training environments and examples: robotics, locomotion, multi-agent
- Physics:
  - Support of deformable objects – soft bodies, cloth, etc
  - Soft actuators
- Early access soon (Contact if interested!)
- General release in 2019

Thank You!

vmakoviychuk@nvidia.com

NVIDIA®