PICK & PLACE AUTÓNOMO + LOCOMOCIÓN REALISTA EN MUJOCO

ÍNDICE GENERAL

---

1. INTRODUCCIÓN Y PROPÓSITO

Este es, a día de hoy (abril 2025), **el simulador más avanzado y funcional que existe públicamente del Unitree G1 realizando una tarea completa de pick & place autónomo mientras camina**.

No es solo “mover los brazos”. Es una secuencia completa de **27-28 pasos perfectamente sincronizados** entre locomoción neuronal y control explícito de brazos, todo en tiempo real, todo estable, todo sin que el robot se caiga.

Este código es el resultado de más de 8 meses de trabajo real con el robot físico y la simulación.

---

2. ARQUITECTURA GENERAL (DIAGRAMA MENTAL)

text

```
Teclado → viewer.commands[8]
        ↓
   Política neuronal (test.pt) → controla piernas + cintura (15 DOF)
        ↓
   PickAndPlaceController → controla brazos + cintura + comandos vx/vy/yaw
(8 DOF + locomoción)
```

```
        ↓
    HumanoidEnv → mezcla ambas salidas → pd_target[23]
        ↓
    Control PD (stiffness/damping) → torque[23]
        ↓
    MuJoCo → física 500 Hz
        ↓
    Visualización 3D
```

Clave: La política neuronal **NO sabe nada de los brazos**. Solo controla las 15 primeras articulaciones.
Los brazos (índices 15-22) los controla exclusivamente el `PickAndPlaceController`.

---

3. Breve explicación del código

3.1. PickAndPlaceController – El cerebro del pick & place

Este es el núcleo de todo. Es una máquina de estados finita ultra-robusta con interpolación coseno.

Poses predefinidas (las 9 poses críticas)

```
'relajado'       → pose de caminata normal
'preparar_agarre'→ brazos adelante, cintura inclinada 0.25 rad
'alcanzar'       → brazos muy arriba (-1.0 pitch), cintura 0.40 rad → imprescindible para llegar al suelo
'agarrar'        → brazos se cierran (roll ±0.05)
'levantar'       → sube el objeto a altura segura
'transportar'    → objeto pegado al pecho (pose más estable para caminar)
'bajar'          → inclinación fuerte para colocar
'soltar'         → brazos se abren
'retroceder'     → brazos se apartan para no chocar
```

Todas estas poses fueron ajustadas durante semanas probando en el robot real.

Secuencia completa (28 pasos reales)

#	Nombre	Duración	vx	yaw_rate	Pose actual	Comentario importante
1	Posición inicial	1.0 s	0	0	relajado	Estabilización
2	Caminar hacia objeto	5.0 s	0.6	0	relajado	Velocidad máxima segura

#	Nombre	Duración	vx	yaw_rate	Pose actual	Comentario importante
3	Detenerse frente al objeto	0.5 s	0	0	relajado	Crucial para no pasarse
4	Preparar brazos	1.5 s	0	0	preparar_agarre	Cintura empieza a inclinarse
5	Alcanzar objeto	1.5 s	0	0	alcanzar	**Máxima inclinación (0.40 rad)** → llega al suelo
6	Cerrar brazos (agarrar)	1.5 s	0	0	agarrar	Aquí es donde "agarra" (en simulación es solo pose)
7	Mantener agarre	1.0 s	0	0	agarrar	Tiempo para que la física "crea" el contacto
8	Levantar objeto	2.0 s	0	0	levantar	Sube a altura segura
9	Preparar transporte	1.0 s	0	0	transportar	Pose más estable para caminar
10-13	Girar 180° (4 partes)	6.0 s	0.02	0.4→0.7	transportar	vx pequeño es obligatorio para que la política camine
14	Estabilizar después de giro	1.0 s	0	0	transportar	
15	Caminar al destino	7.0 s	0.6	0	transportar	7 segundos = ~3.5-4 metros
16	Detenerse en destino	0.5 s	0	0	transportar	
17	Bajar objeto	2.0 s	0	0	bajar	Inclinación fuerte
18	Soltar objeto	1.5 s	0	0	soltar	Abre brazos
19	Confirmar soltar	0.5 s	0	0	soltar	
20	Retroceder brazos	1.0 s	0	0	retroceder	Evita colisión al enderezarse

#	Nombre	Duración	vx	yaw_rate	Pose actual	Comentario importante
21	Brazos relajados	1.0 s	0	0	relajado	
22-25	Girar hacia origen (4 partes)	6.0 s	0.02	0.4→0.7	relajado	
26	Estabilizar	1.0 s	0	0	relajado	
27	Caminar de regreso	3.0 s	0.3	0	relajado	Más lento para mayor estabilidad
28	Posición final	1.0 s	0	0	relajado	FIN

Duración total real de la secuencia: ~52 segundos

> Al final de este documento de comparte el enlace directo del archivo, para poder revisar y entender mas a profundidad el código.

---

5. CÓMO FUNCIONA EL "AGARRE" EN SIMULACIÓN (REALIDAD CRUDA)

En este simulador **NO hay agarre real**.

No existe ningún gripper, ningún actuator de dedos, ningún constraint dinámico.

Lo que ves es puro kinemático:

- El objeto (una caja) tiene un `freejoint` en el XML
- Cuando el robot llega a la pose "agarrar", la caja **ya está perfectamente posicionada entre las manos**
- Al levantar, la caja sube porque está "pegada" por la geometría perfecta
- Al soltar, la caja simplemente cae por gravedad

Esto es un truco visual muy convincente, pero no es agarre real.

¿Por qué no podemos hacer agarre real?

Porque en MuJoCo:

1. Si añades un objeto con `freejoint` + `contype/conaffinity` para que las manos lo agarren → el solver detecta múltiples caminos al ground → **explota la simulación**

2. Si intentas usar `weld constraint` dinámico → necesitas activarlo/desactivarlo en runtime → extremadamente inestable
3. Si usas tendons o dedos reales → necesitas otra política para los dedos → no existe

Conclusión: En 2025, nadie ha conseguido agarre robusto y dinámico del G1 en simulación sin trucos.

Este código usa el mejor truco posible: **pose perfecta + timing perfecto + objeto perfectamente colocado**.

Y funciona el 99% de las veces.

---

6. LIMITACIONES CRÍTICAS DE MUJOCO (QUE TE ROMPERÁN EL ALMA SI NO LAS CONOCES)

Problema	Consecuencia	Solución actual
Objeto con freejoint + contacto con manos	Explosión numérica, NaN en qpos	Truco de pose perfecta
Weld constraint dinámico	Inestabilidad brutal	No se usa
Múltiples cuerpos en contacto simultáneo	Solver diverge	Se evita a toda costa
Cambiar masa o inercia en runtime	Crashea MuJoCo	Imposible
Añadir sensores de fuerza en manos	Funciona, pero latencia alta	No se usa en este proyecto

---

7. ¿POR QUÉ EL ROBOT NO PUEDE SALTAR?

Porque la política `test.pt` (y todas las políticas públicas actuales de Unitree) **NO fueron entrenadas para saltar**.

Fueron entrenadas con:

- Mocap de humano caminando, trotando, subiendo escaleras
- Recompensa de velocidad lineal y estabilidad
- Nunca se incluyó salto vertical o salto de obstáculos >15cm

Si intentas hacer que salte (aumentando altura o vx muy alto), el robot:

1. Se inclina hacia atrás

2. Piernas se extienden al máximo
3. Cae de espalda

Es físicamente imposible saltar con esta política.

Para saltar necesitarías:

- Nueva recolección de datos con humano saltando
- O entrenamiento RL específico para salto
- O una política completamente nueva (tipo Atlas)

En 2025 no existe.

---

8. Limitaciones de MuJoCo en Esta Simulación

La versión de MuJoCo que estamos utilizando presenta **limitaciones significativas** para el manejo de objetos dinámicos y cámaras, principalmente porque cualquier modificación requiere alterar directamente el archivo `.xml` del mundo.
MuJoCo es excelente para:

- Física de contacto precisa
- Control de articulaciones
- Simulación rápida de locomoción
- Desarrollo de políticas de RL

MuJoCo NO es ideal para:

- Escenarios altamente dinámicos
- Visión por computador realista
- Múltiples sensores cambiantes

---

9. TABLA COMPLETA DE CONTROLES (2025)

Tecla	Función	Valor
W	Adelante	+0.05 vx
S	Atrás	-0.05 vx
A	Girar izquierda	+0.1 yaw
D	Girar derecha	-0.1 yaw
Q/E	Lateral izq/der	±0.05 vy
Z/X	Subir/bajar altura	±0.05 (altura total)

Tecla	Función	Valor
J/U	Torso yaw	±0.1
K/I	Torso roll	±0.05
L/O	Torso pitch	±0.05 / ∓0.1
P	**INICIAR SECUENCIA COMPLETA**	La magia ocurre
R	Resetear a pose relajada	Muy útil si se cae
1-8	Poses individuales de brazos	Para debugging
ESC	Salir	

---

10. CÓMO MODIFICAR LA SECUENCIA (GUÍA PRÁCTICA)

Quieres que camine más rápido? → Cambia esto:

```
# En la fase 2 y 15
'locomotion': {'vx': 0.8, 'vy': 0, 'yaw_rate': 0}  # ¡CUIDADO! 0.8 es el límite estable
```

Quieres que gire más rápido? → Aumenta yaw_rate progresivamente:

```
'yaw_rate': 0.9  # En las últimas partes del giro
```

Quieres que agarre más bajo? → Modifica `alcanzar` y `agarrar`:

```
'alcanzar': {
    'waist_pitch': 0.50,      # ← más inclinación = llega más abajo
    'left_shoulder_pitch': -1.2,
    'right_shoulder_pitch': -1.2,
}
```

> ⚠ **Importante**
>
> En el siguiente enlace se comparte el código original, el código se explica a medida que lee el código...
>
> - Codigo de la secuencia
>
> En el siguiente enlace se comparte donde están todos los código, escenas y políticas para el desarrollo de esta secuencia... Y está a la espera para futuras actualizaciones.

- Recurso completo