

GUÍA DE INTEGRACIÓN DEL MODULADOR 3D

Estructura de Archivos Final

```
trajectory_hub/
├── core/
│   ├── __init__.py
│   ├── enhanced_trajectory_engine.py      # [MODIFICAR]
│   ├── motion_components.py              # [REEMPLAZAR]
│   ├── spat_osc_bridge.py                # [MANTENER]
│   ├── trajectory_deformers.py           # [MANTENER]
│   ├── macro_behaviors.py               # [MANTENER]
│   └── distance_controller.py            # [MANTENER]
├── interface/
│   ├── __init__.py
│   ├── interactive_controller.py         # [MODIFICAR]
│   └── interface_utils.py                # [MANTENER]
├── presets/
│   ├── __init__.py
│   └── artistic_presets.py               # [MANTENER]
├── demos/
│   ├── __init__.py
│   ├── demo_enhanced_system.py           # [MANTENER]
│   ├── comprehensive_test.py            # [MANTENER]
│   └── test_modulator_integration.py      # [NUEVO]
├── visualizers/                          # [NUEVA CARPETA]
│   ├── __init__.py
│   └── modulator_visualizer.py           # [NUEVO - OPCIONAL]
├── docs/                                # [NUEVA CARPETA]
│   ├── MODULATOR_GUIDE.md              # [NUEVO]
│   └── Simulador_Modulacion.html         # [COPIAR EXISTENTE]
├── tools/
│   ├── reorganize_project.py             # [MANTENER]
│   └── update_imports.py                 # [MANTENER]
└── main.py                              # [MANTENER]
```

PASOS DE INTEGRACIÓN

PASO 1: Crear Estructura de Carpetas

bash

```
# En la raíz del proyecto trajectory_hub  
mkdir -p trajectory_hub/visualizers  
mkdir -p trajectory_hub/docs  
touch trajectory_hub/visualizers/__init__.py
```






PASO 2: Actualizar motion_components.py

Archivo: `trajectory_hub/core/motion_components.py`

Acción: REEMPLAZAR TODO EL ARCHIVO con el contenido del artifact

`motion_components_updated`

Este archivo contiene:

-  Clase `AdvancedOrientationModulation` completa
-  8+ presets alineados con el simulador
-  Sistema de interpolación
-  Control de intensidad (C2)
-  Soporte para serialización

PASO 3: Modificar enhanced_trajectory_engine.py

Archivo: `trajectory_hub/core/enhanced_trajectory_engine.py`

Acción: AÑADIR las siguientes secciones (NO reemplazar todo el archivo)

3.1 - Añadir imports al principio:

python

```
from trajectory_hub.core.motion_components import AdvancedOrientationModulation
```

3.2 - Modificar `__init__`:

python

```
def __init__(self, n_sources: int = 64, update_rate: int = 60,
              distance_mode: str = 'perceptual', enable_modulator: bool = True):
    # ... código existente ...

    # Sistema de modulación de orientación
    self.enable_modulator = enable_modulator
    self.orientation_modulators = {} # Dict[int, AdvancedOrientationModulation]

    # Configuración global del modulador
    self.global_modulator_intensity = 1.0
    self.global_modulator_preset = None

    # Cache de últimas orientaciones enviadas
    self._last_orientations = {}
    self._orientation_update_threshold = 0.01 # radianes
```

3.3 - Añadir TODOS los métodos nuevos del artifact `engine_modulator_integration`:

- `create_orientation_modulator()`
- `apply_orientation_preset()`
- `set_orientation_lfo()`
- `set_orientation_intensity()`
- `interpolate_orientation_presets()`
- `set_orientation_shape()`
- `toggle_orientation_modulation()`
- `get_orientation_presets()`
- `get_modulator_state()`
- `save_modulator_state()`
- `load_modulator_state()`

3.4 - Modificar el método `update()`:

Buscar el método `update()` existente y añadir la sección de moduladores DESPUÉS de actualizar posiciones:

python

```
def update(self, dt: float):
    # ... código existente de actualización de posiciones ...

    # Actualizar moduladores de orientación si están habilitados
    if self.enable_modulator:
        for source_id, state in self.motion_states.items():
            if source_id in self.orientation_modulators:
                modulator = self.orientation_modulators[source_id]
                if modulator.enabled:
                    # Actualizar estado con modulación
                    state = modulator.update(current_time, dt, state)
                    self.motion_states[source_id] = state

    # ... resto del código ...
```

3.5 - Modificar `_send_osc_update()`:

Reemplazar el método completo con la versión del artifact que incluye orientaciones.

3.6 - Modificar `create_macro()`:

Añadir al final del método existente:

python

```
# Crear moduladores de orientación si está habilitado
if self.enable_modulator:
    for i, sid in enumerate(source_ids):
        modulator = self.create_orientation_modulator(sid)
        # Desfase temporal para efecto orgánico
        modulator.time_offset = i * 0.05

# Si hay un preset global configurado, aplicarlo
if self.global_modulator_preset:
    self.apply_orientation_preset(name, self.global_modulator_preset,
                                  self.global_modulator_intensity)
```

PASO 4: Actualizar `interactive_controller.py`

Archivo: `trajectory_hub/interface/interactive_controller.py`

Acción: MODIFICAR (no reemplazar todo)

4.1 - Añadir en `print_menu()` después de las opciones existentes:

python

```
# Añadir sección del modulador
print("\n🌀 MODULADOR DE ORIENTACIÓN:")
print("| 20. Aplicar Preset de Modulación      | Presets P1/P2/P3 predefinidos      |")
print("| 21. Ajustar Velocidad (LF0)           | Control de frecuencia P2           |")
print("| 22. Ajustar Intensidad                 | Control global C2 (0-100%)         |")
print("| 23. Configurar Forma                   | Tipo y parámetros P1               |")
print("| 24. Interpolación Presets              | Morphing entre dos presets         |")
print("| 25. Toggle Modulación                  | Activar/Desactivar por macro       |")
```

4.2 - Añadir la función `handle_modulation_menu()` completa del artifact

4.3 - En el método `run()`, añadir los casos para las opciones 20-25:

python

```
elif choice == "20":
    # Código del artifact para aplicar preset rápido

elif choice == "21":
    # Código del artifact para ajustar LF0

elif choice == "22":
    # Código del artifact para ajustar intensidad

elif choice in ["23", "24", "25"]:
    # Abrir submenú de modulación
    self.handle_modulation_menu(self)
```

PASO 5: Crear Archivos Nuevos

5.1 - Crear `test_modulator_integration.py`:

bash

```
# Guardar el contenido del artifact en:
trajectory_hub/demos/test_modulator_integration.py
```

5.2 - Crear `modulator_visualizer.py` (OPCIONAL):

bash

```
# Guardar el contenido del artifact en:
trajectory_hub/visualizers/modulator_visualizer.py
```

5.3 - Crear documentación:

```
bash
```

```
# Guardar el contenido del artifact en:  
trajectory_hub/docs/MODULATOR_GUIDE.md
```

```
# Copiar tu archivo HTML existente a:  
trajectory_hub/docs/Simulador_Modulacion.html
```

PASO 6: Verificar la Instalación

```
bash
```

```
# 1. Verificar sintaxis  
python -m py_compile trajectory_hub/core/motion_components.py  
python -m py_compile trajectory_hub/core/enhanced_trajectory_engine.py  
  
# 2. Ejecutar test de integración  
python trajectory_hub/demos/test_modulator_integration.py  
  
# 3. Si todo está OK, ejecutar con demo  
python trajectory_hub/demos/test_modulator_integration.py --demo
```

PASO 7: Probar con el Sistema Interactivo

```
bash
```

```
# Ejecutar el sistema principal  
python main.py --interactive  
  
# Probar:  
# 1. Crear un macro (opción 1)  
# 2. Aplicar preset de modulación (opción 20)  
# 3. Verificar en Spat que las fuentes rotan
```

⚠ PUNTOS IMPORTANTES

NO BORRAR:

- Ningún archivo existente
- Ninguna funcionalidad existente

SOLO AÑADIR:

- Nuevos métodos a `enhanced_trajectory_engine.py`
- Nuevas opciones a `interactive_controller.py`

- Nuevos archivos en las carpetas indicadas

BACKUP RECOMENDADO:

bash

Antes de empezar, hacer backup

```
cp -r trajectory_hub trajectory_hub_backup_$(date +%Y%m%d)
```

DEPENDENCIAS:

Si el visualizador da problemas:

bash

```
pip install matplotlib
```

VERIFICACIÓN FINAL

Checklist de Integración:

- ☐ motion_components.py actualizado con AdvancedOrientationModulation
- ☐ enhanced_trajectory_engine.py con todos los métodos nuevos
- ☐ interactive_controller.py con opciones 20-25
- ☐ test_modulator_integration.py creado y funcionando
- ☐ Documentación copiada a la carpeta docs/
- ☐ Test básico ejecutado sin errores
- ☐ Verificado en Spat que llegan orientaciones

Test Rápido:

python

En Python interactivo

```
from trajectory_hub import EnhancedTrajectoryEngine
engine = EnhancedTrajectoryEngine(enable_modulator=True)
engine.create_macro("test", [0,1,2])
engine.apply_orientation_preset("test", "lissajous")
print("✅ Modulador integrado correctamente")
```

SOPORTE

Si algo no funciona:

1. Verificar que no hay errores de sintaxis

2. Revisar los logs en la consola
3. Verificar que Spat está recibiendo en puerto 9000
4. Ejecutar el test de integración en modo debug

¿Necesitas ayuda con algún paso específico?