RUMBA 2.0

mercenaries engineering

Guerilla Render

RUMBA

RUMBA 1.2

Playback temps réel

Manipulation naturelle

Organisation de l’animation en **Layers**

Outils d’animations avancés (anim tools, onion skin, motion trails)

Compatible avec tous les rigs Maya

Déformations au plan (Sticky, Sculpt, …)

USD

RUMBA 2.0

Séquenceur

RUMBA 2.0

Dynamique pour les animateurs

Meilleur support USD

RUMBA 2.0

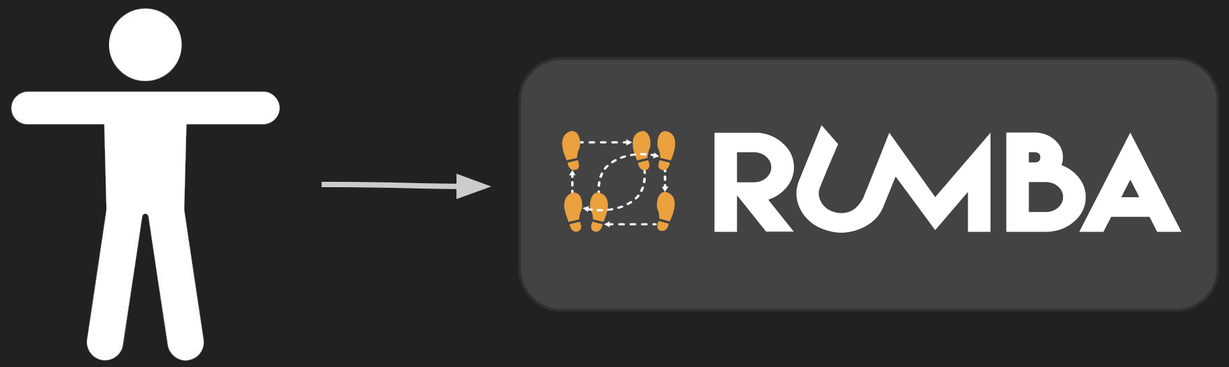
Meilleur support des rigs Maya

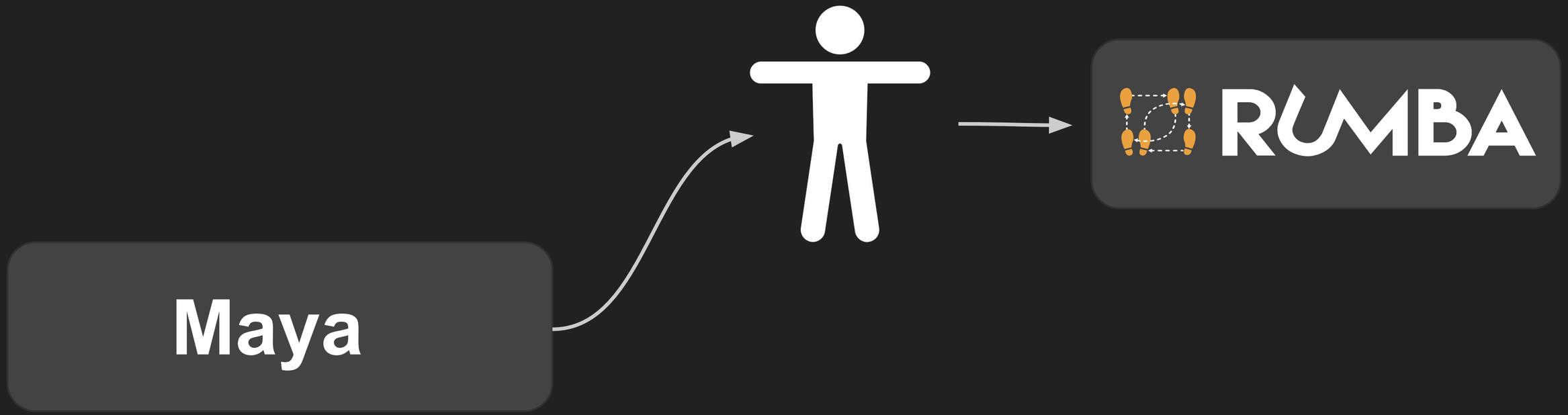
160+ noeuds Maya supportés

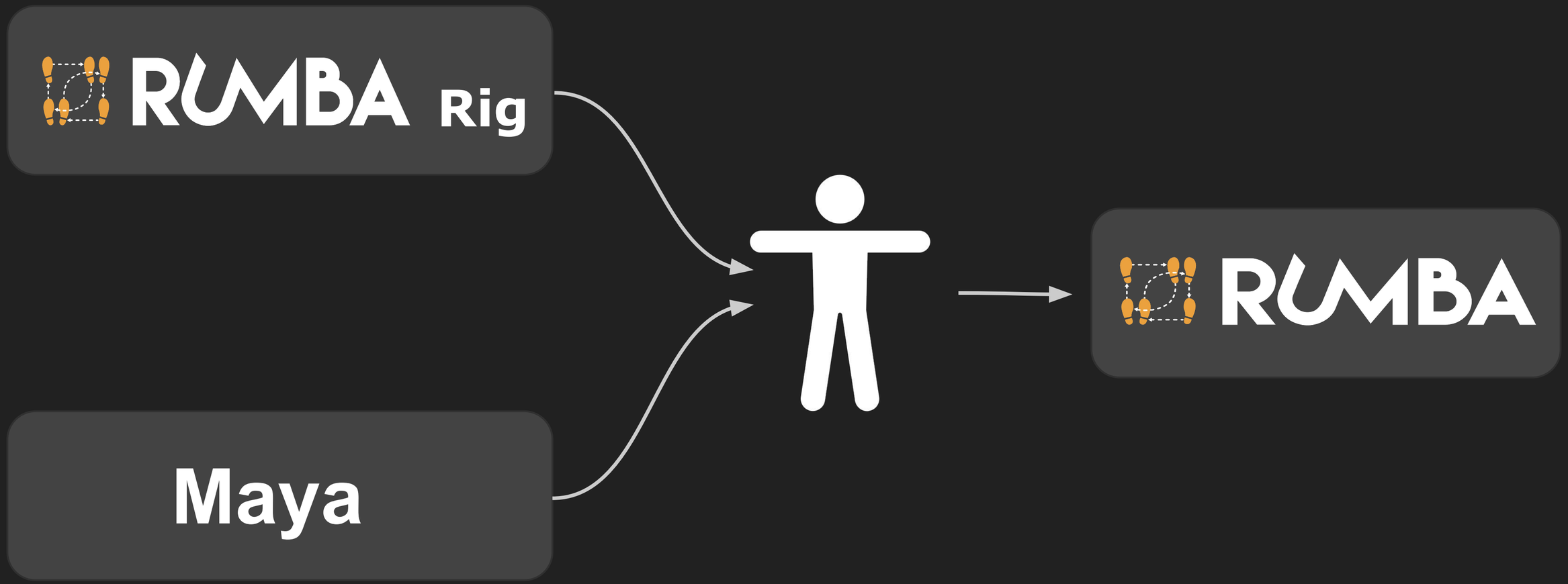
Noeuds complexes comme *expressions* (mel), *poseInterpolator, ...*

Noeuds de shading, *file, place2dTexture, layeredTextures, ramp, bump2d*, ...

RUMBA Rig


RUMBA


Maya
RUMBA


RUMBA Rig
Maya
RUMBA

Rumba Rig: A Procedural Rigging Framework with Direct Graph-Based Control

Florian Canezin
canezin@mercs-eng.com
Mercenaries Engineering
Rennes, France

Xing Ye
xingyep@gmail.com
On Animation
Montréal, Canada

Cyril Corvazier
corvazier@mercs-eng.com
Mercenaries Engineering
Rennes, France

Figure 1: The Rumba Rig framework can achieve professional-quality rigs. ©On Animation Studios.

Abstract

We present Rumba Rig, a new rigging framework that enables riggers to work directly on the final rig graph in a modular and non-linear way, without the need of auto-rig script or abstraction layer. By reducing complexity, the framework makes rigging more accessible and easier to maintain. We demonstrate through practical examples that Rumba Rig is able to express complex rig constructions and delivers professional-quality rigs.

1 Introduction

Designed from the ground-up for animation, the Rumba software [Corvazier and Robert 2021] makes the animation process fast and intuitive. It offers a real-time creative experience, allowing animators to focus on their artistic expression.

Until now, Rumba has relied on Autodesk® Maya® character rigs converted into its internal rig graph format. The Rumba Rig project aims to provide a stand alone alternative to rig characters directly in Rumba.

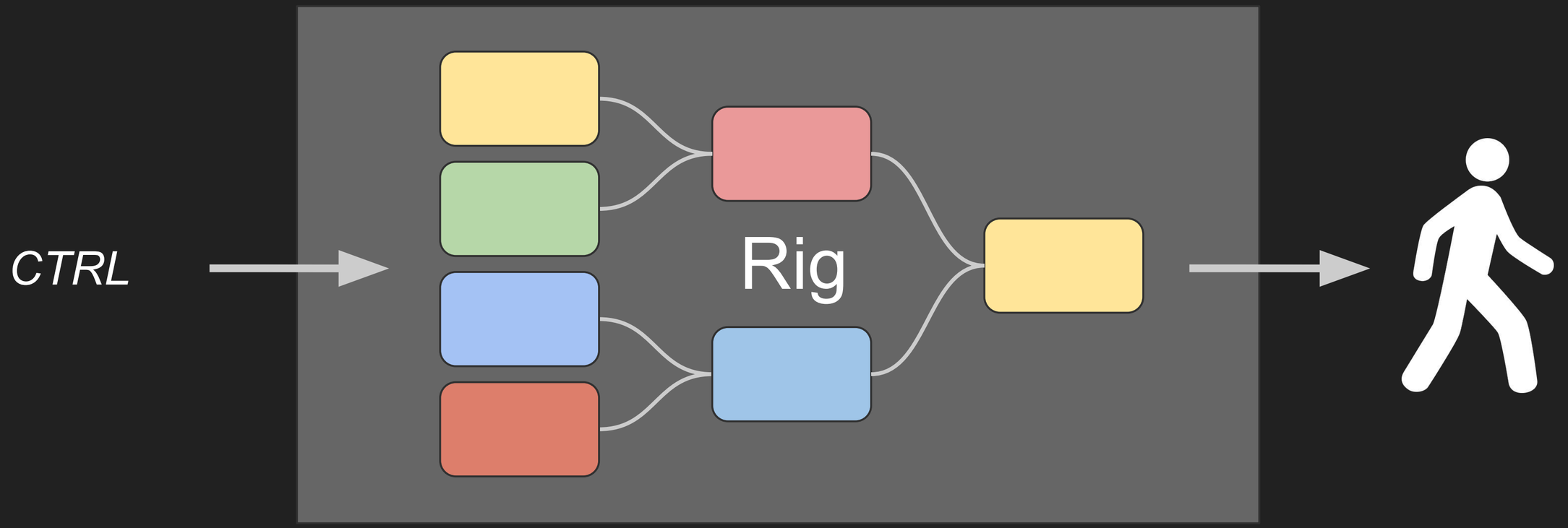
1.1 State-of-the-art

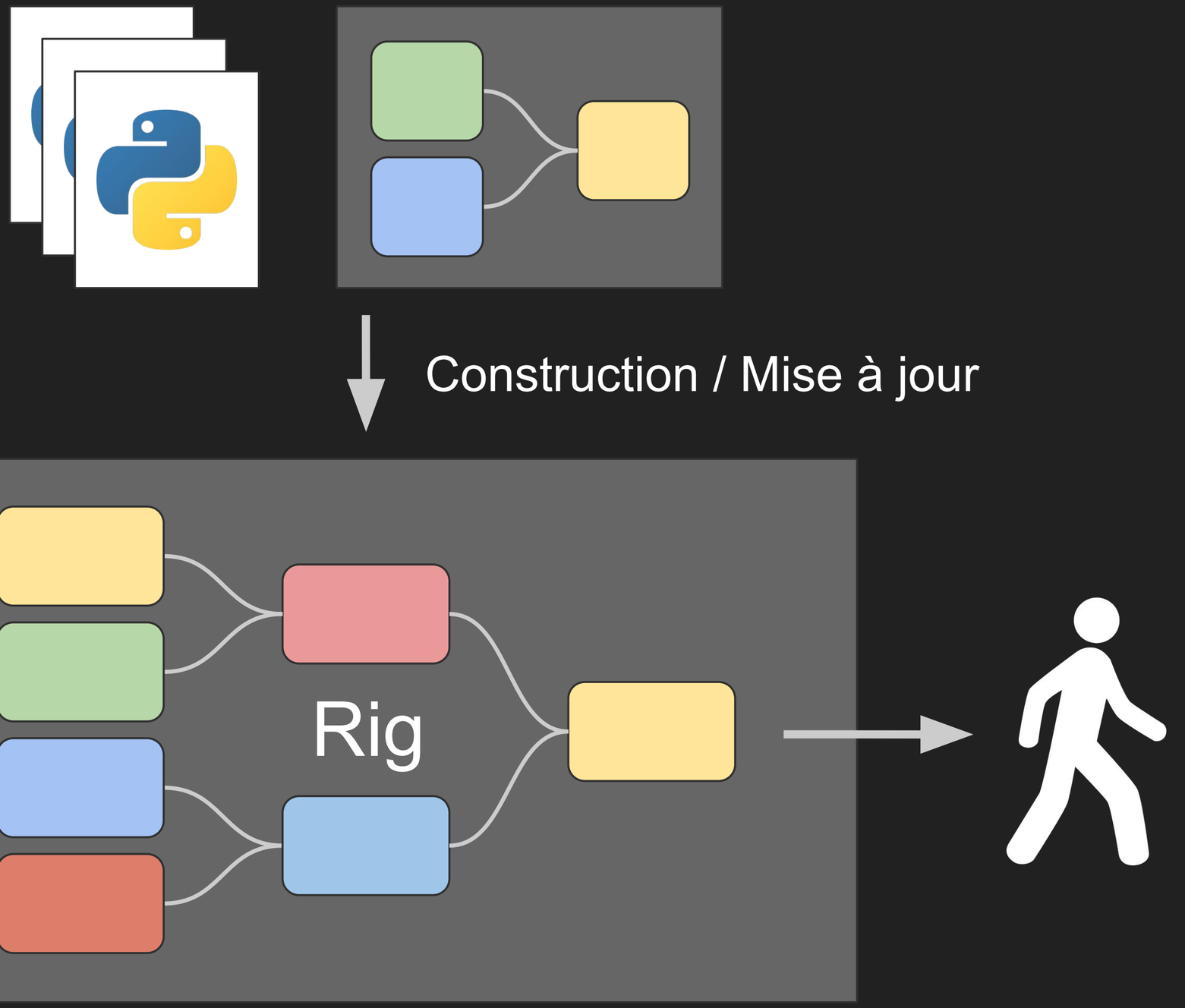
In this paper, we define a rig graph as a procedural geometry graph that deforms character geometries based on the values of animation controllers. The rig graph may also drive additional attributes, such as shader parameters. Rigging refers to the process of constructing and parameterizing this graph to achieve the desired behavior of the animation skeleton, as well as configuring the stack of geometry deformers applied to the character's geometries.

Even in top-tier industry softwares like Maya, directly managing the rig graph is impractical for several reasons. First, the rig graph operates at a low level, handling data such as floats, matrices, and shapes, which results in a graph composed of a large number of nodes and complex interconnections. Additionally, the graph must be customized for each character: each geometry must be individually connected and deformed, and every joint must be linked to its corresponding skinCluster node. Moreover, certain values, such as the bindPreMatrix in Maya's skinCluster, are computed during the binding pre-process and must be updated whenever the skeleton's rest pose or the character geometry changes. Finally, the nature of Maya's graph makes it difficult to organize rigs into modular, versioned, and referenced components.

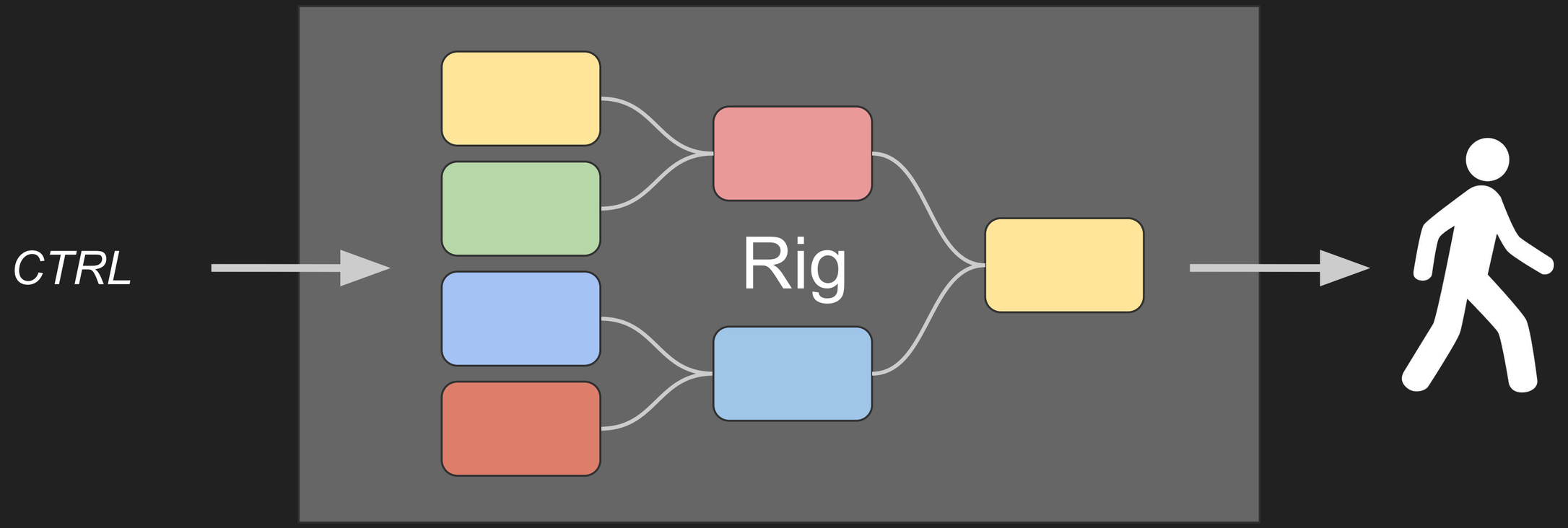
The commonly adopted solution in animation studios is to maintain an auto-rig system (like the one from Blue Sky Studios® [Malvarez et al. 2008]). This is typically a Python-based software project, managed by a small team of technical riggers or developers, whose goal is to provide the rigging team with tools to build and maintain the rig graph. The auto-rig system takes various inputs, such as joint positions and constraint weights, and generates the rig graph, including the skeleton, the controllers and the deformers. It also includes procedures for updating the rig graph when modifications are needed, or when additional character-specific rigging features

2025.

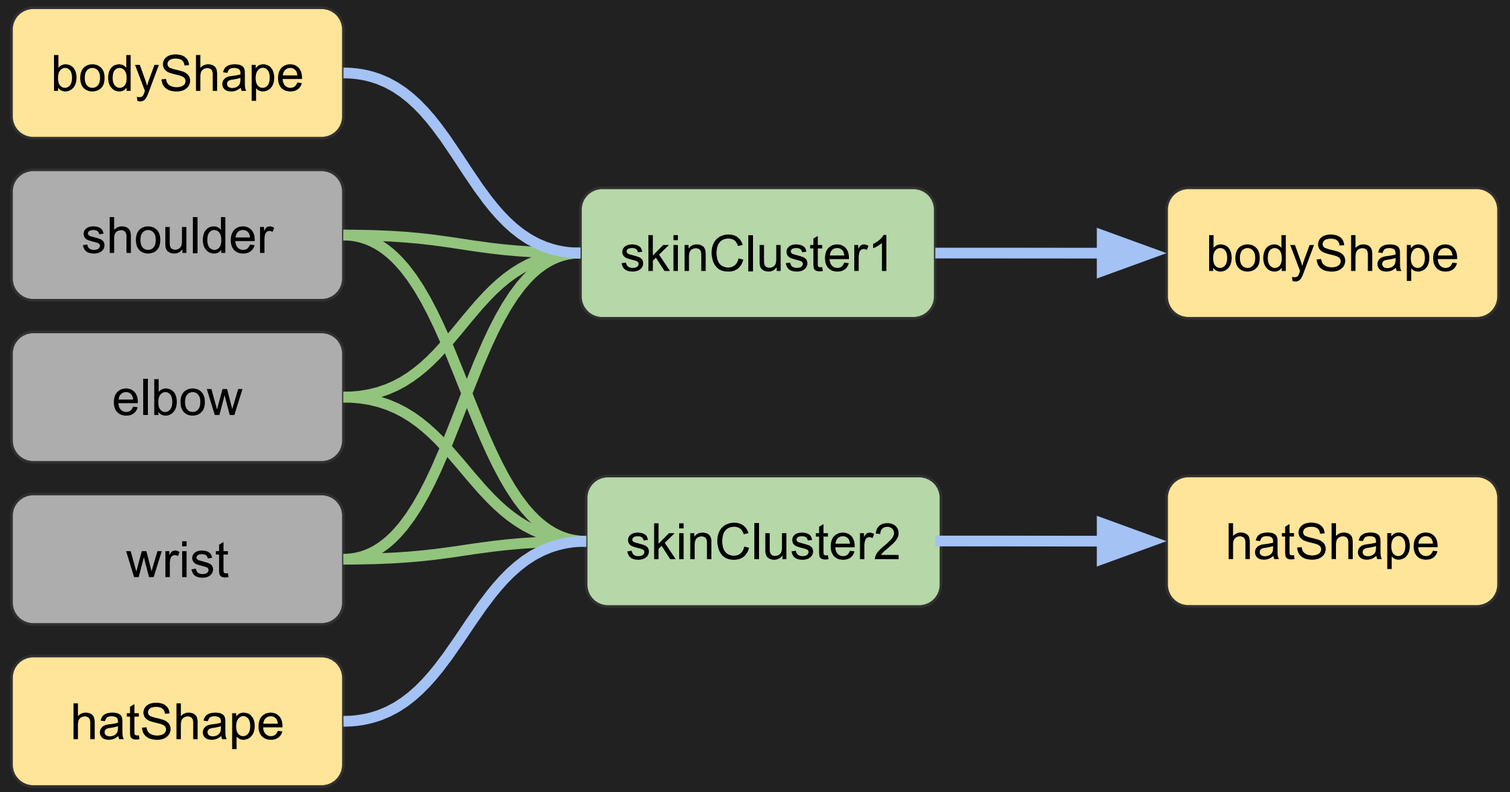

CTRL
Rig


Construction / Mise à jour
CTRL
Rig

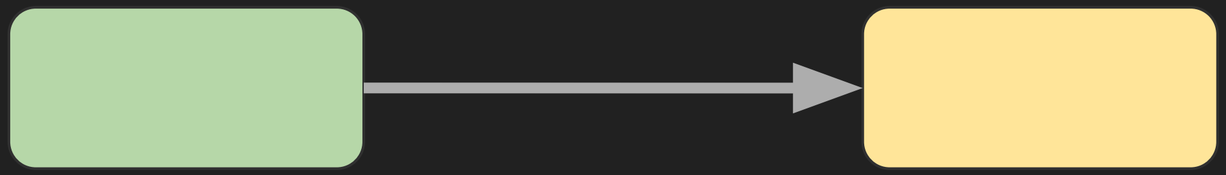
- Pas de script d’auto-rig ou de long processus de construction build
- Mise à jour du rig en direct
- Modulaire par design
- Des composants de rig plus simple à maintenir
- Aide à comprendre, inspecter et debugger un rig existant



Le graphe du rig dépend de la structure du personnage

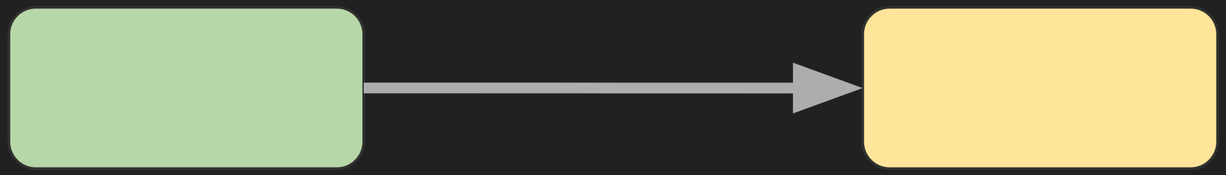


Valeurs de haut niveau



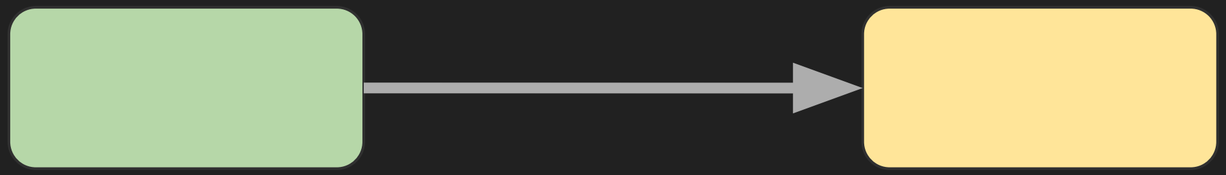
Numbers
Matrices
Shapes

Valeurs de haut niveau



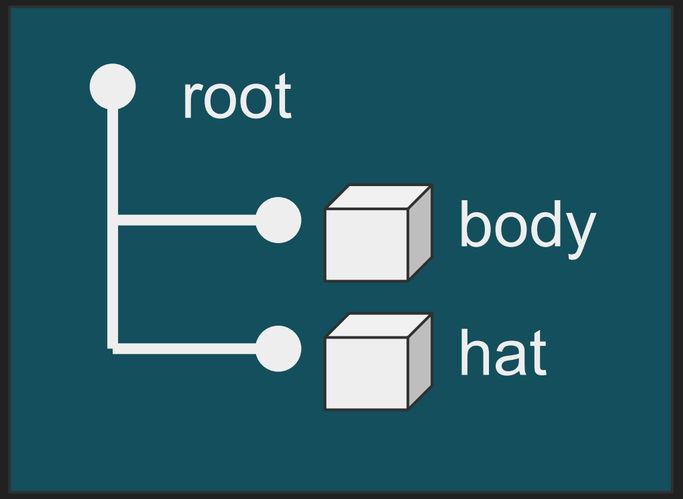
Numbers
Matrices
Shapes
Array
Dict
Expr
Scene

Valeurs de haut niveau

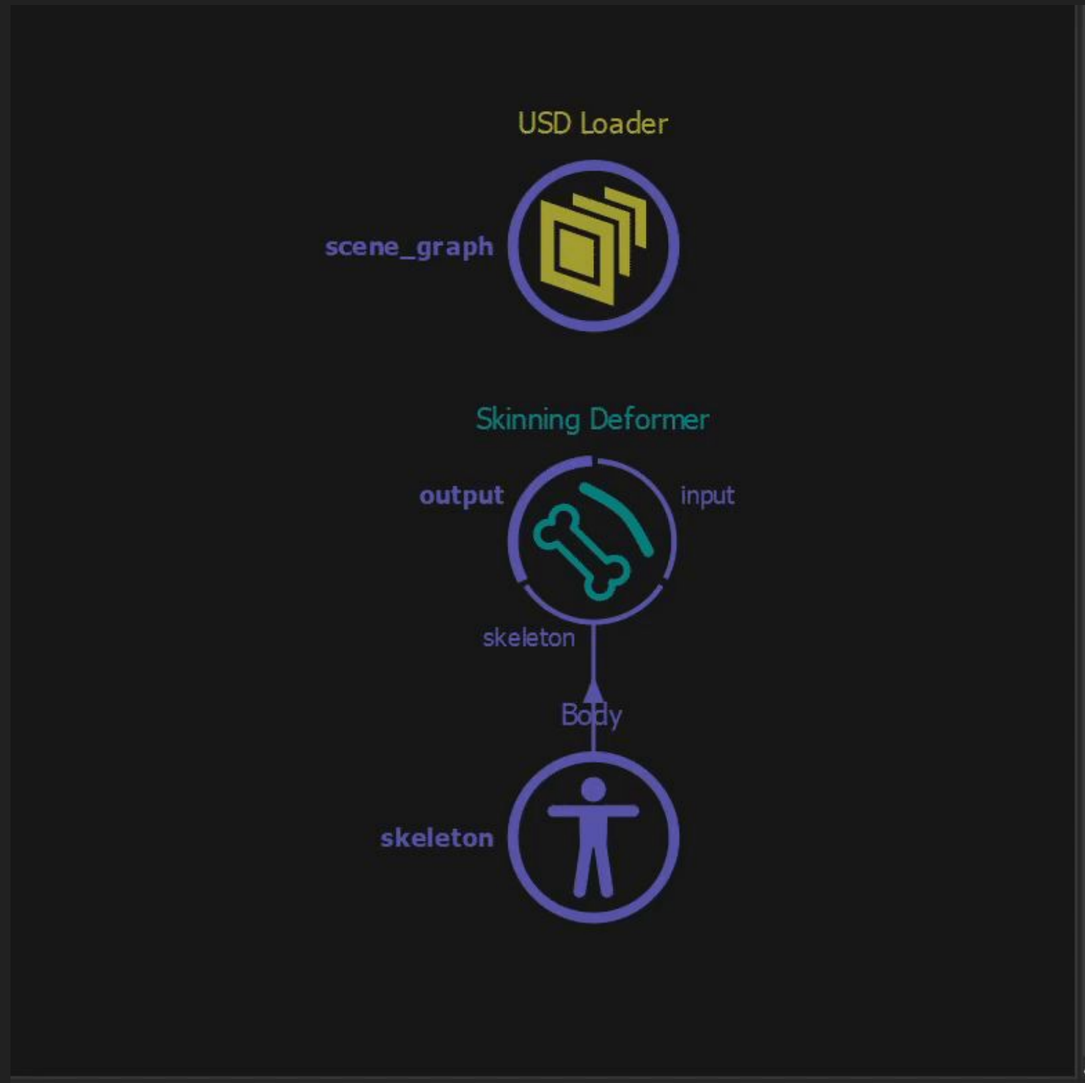


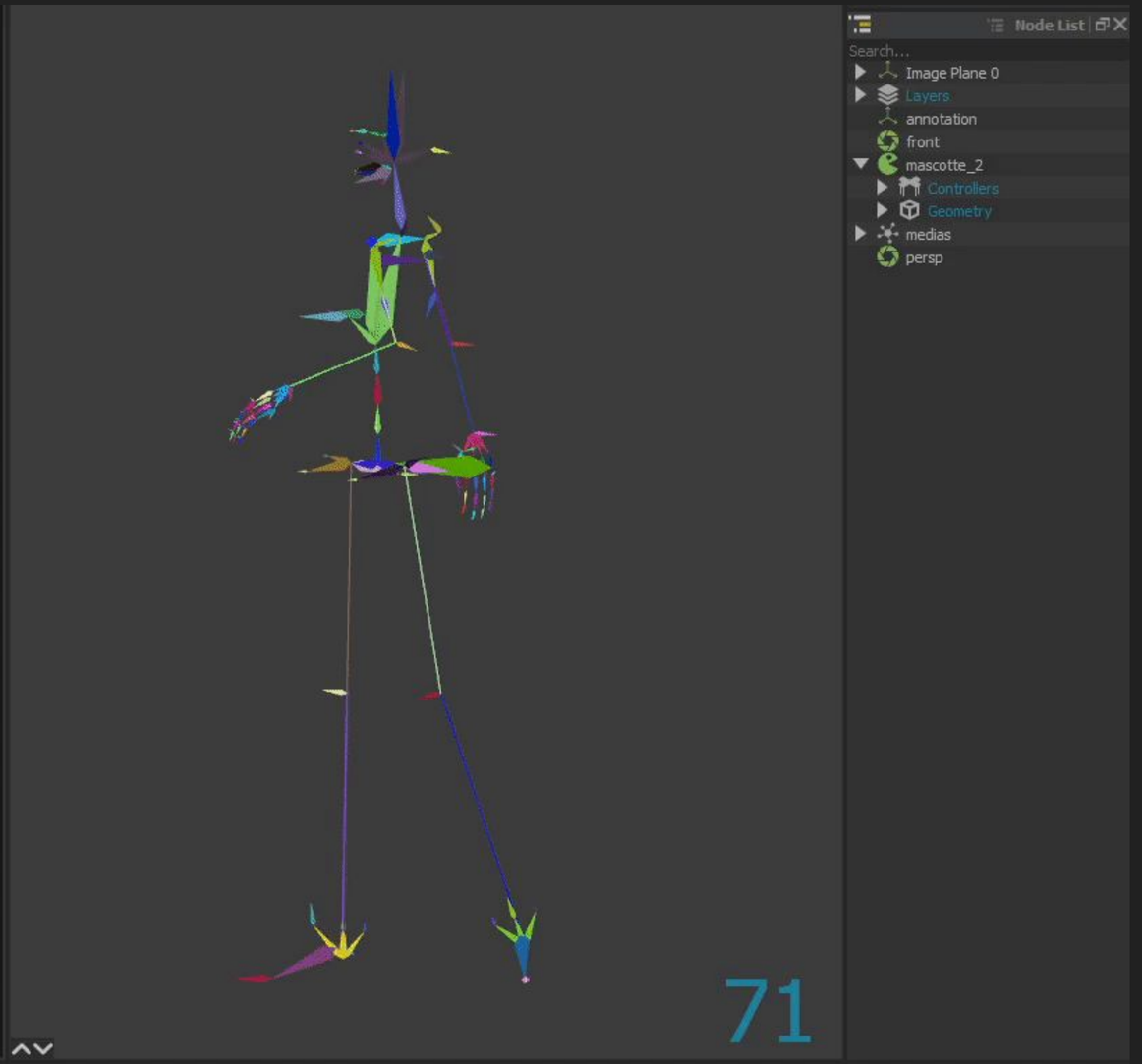
Numbers
Matrices
Shapes
Array
Dict
Expr
Scene

Scene Value

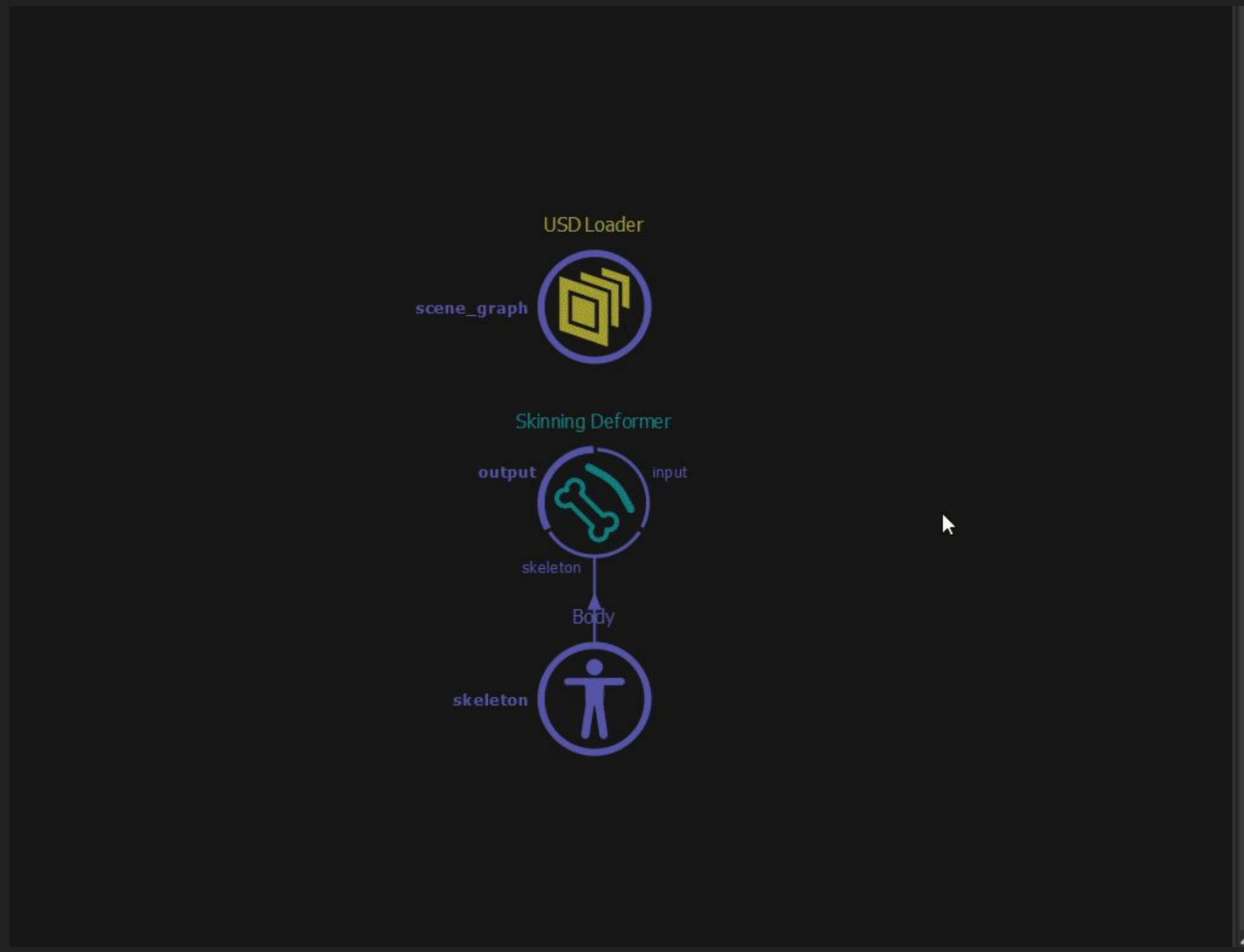


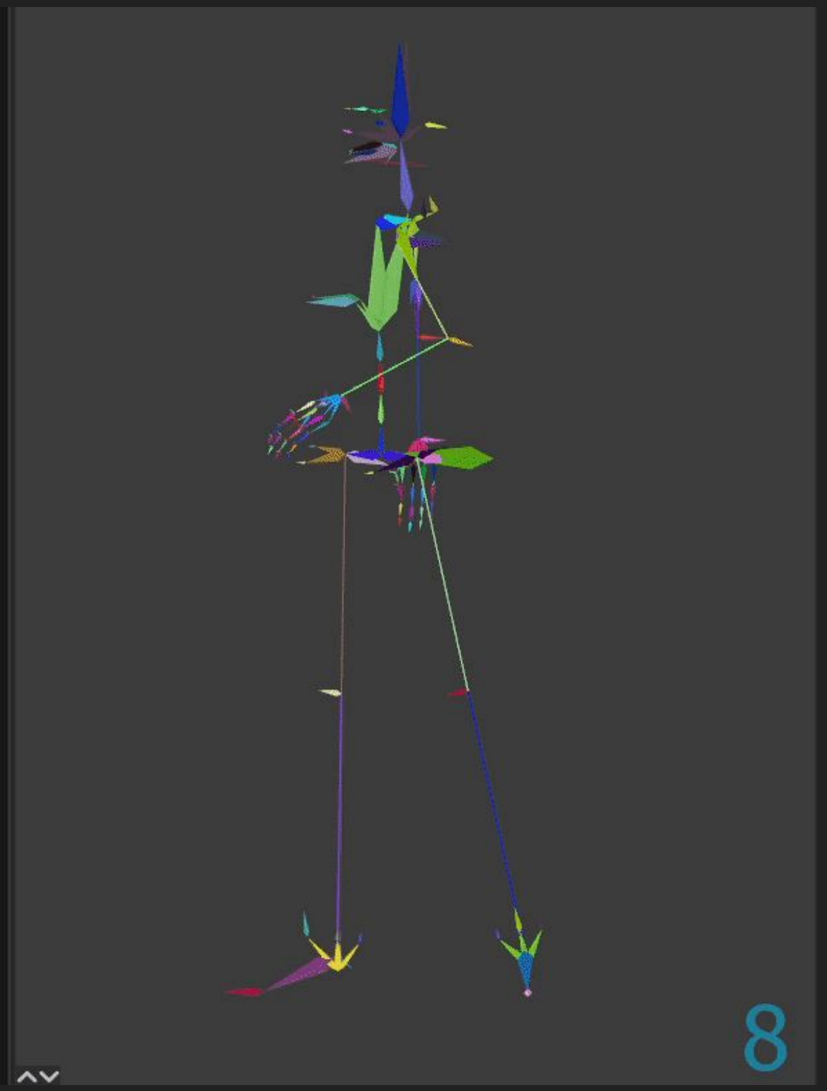
Valeur Scene: Géométrie du personnage



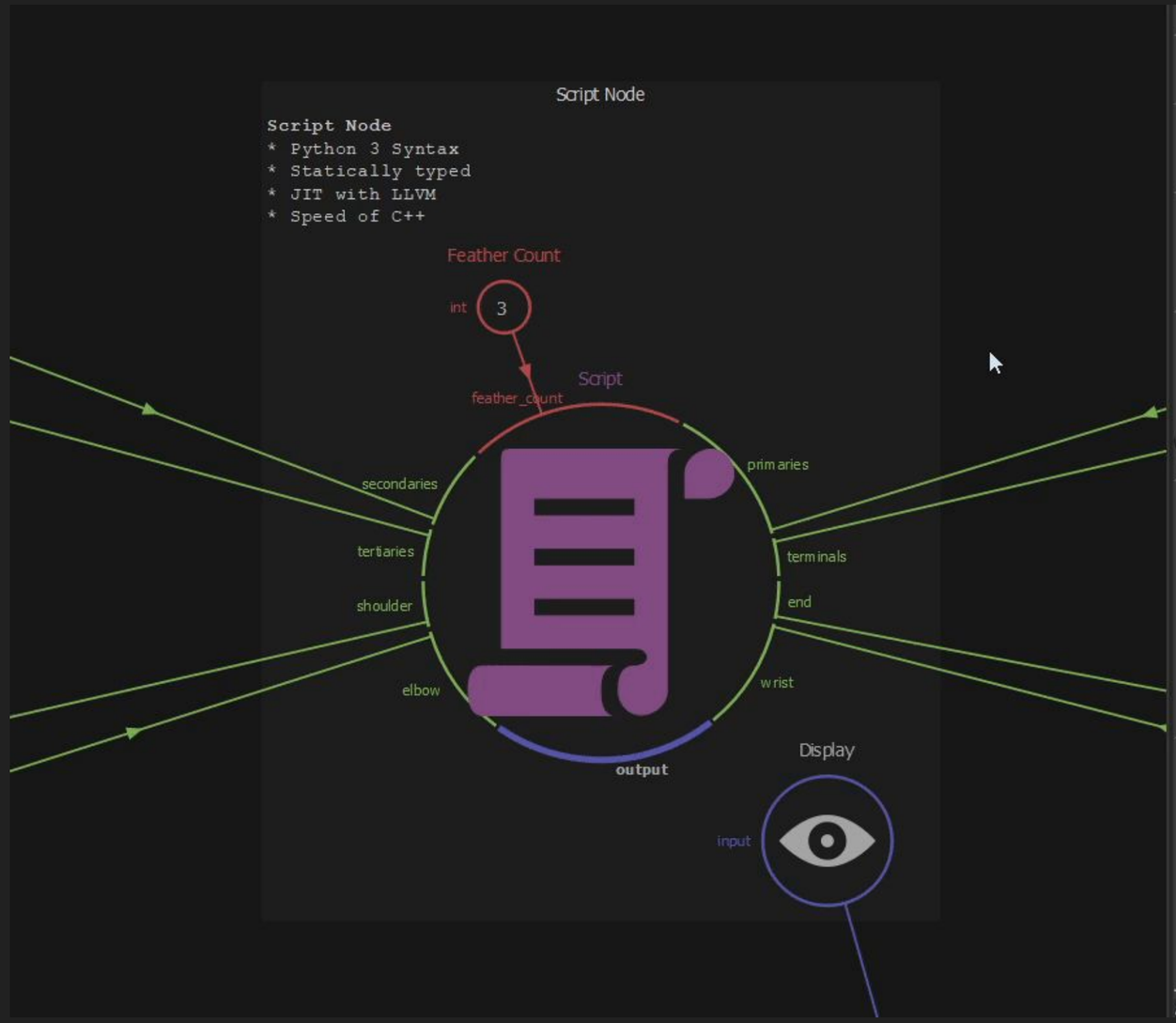


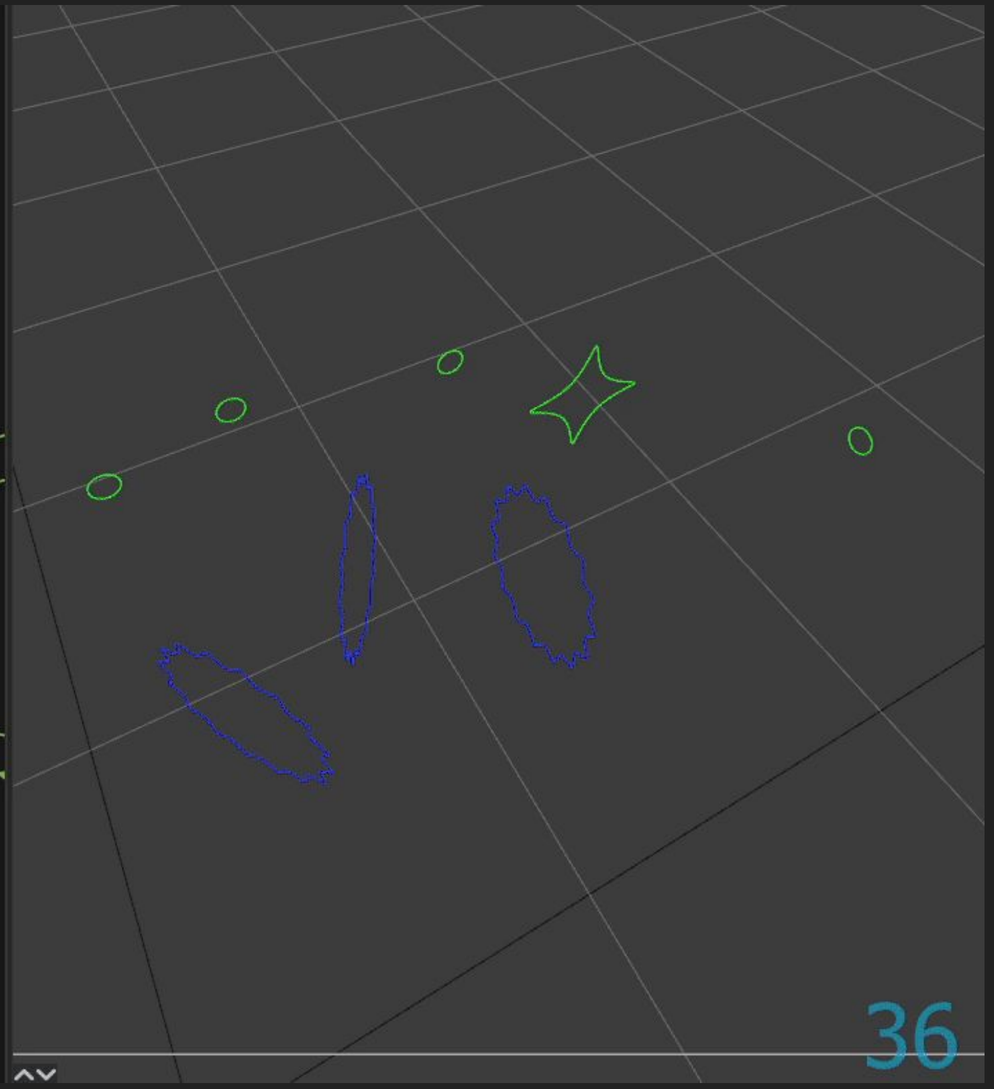
Valeur Scene: Portion de squelette





Valeur Scene: Noeud de script

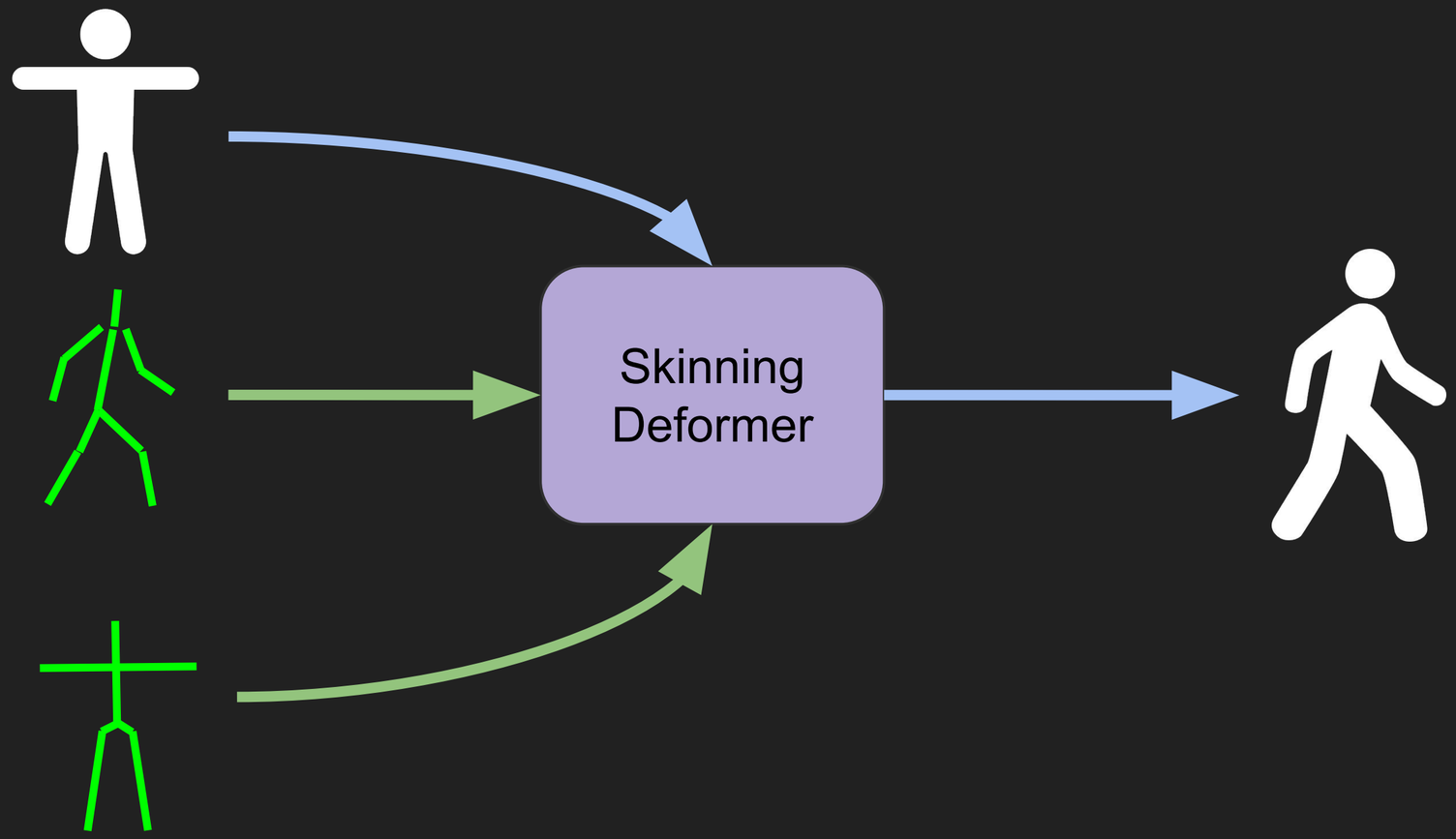




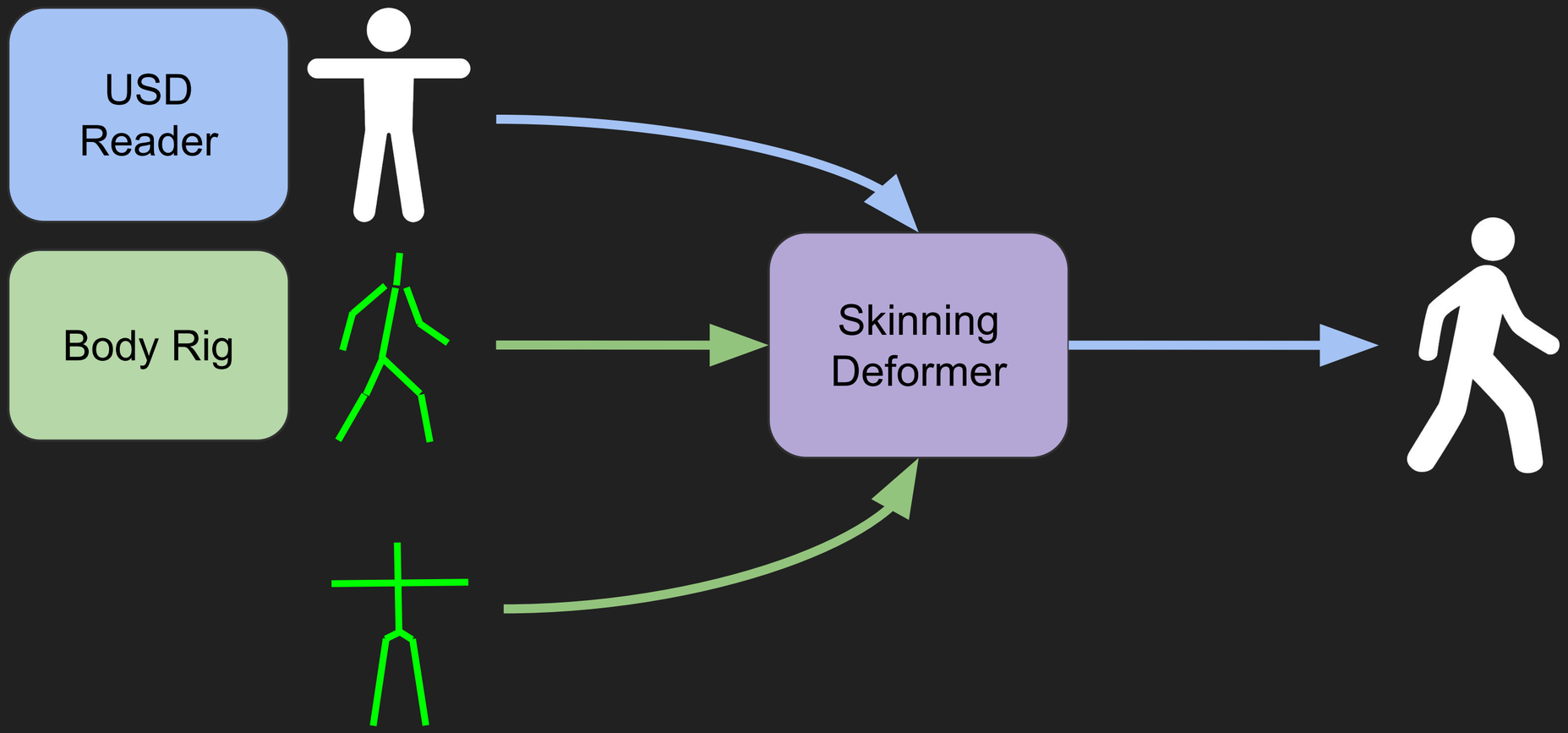
Valeurs initiales

- Matrices initiale de skinning
- Offset de contraintes
- Attachement des déformations Wrap
- …

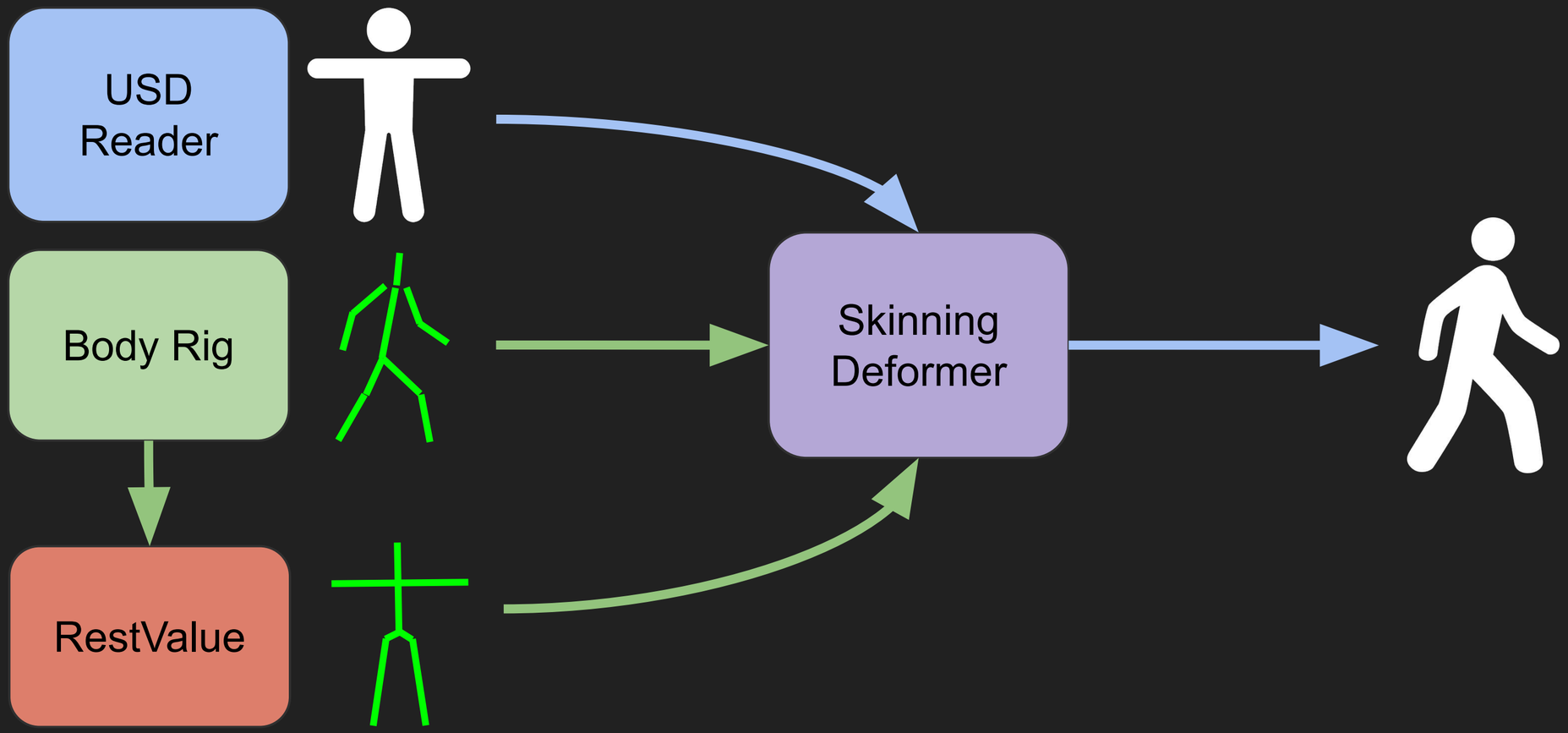
Valeurs initiales: exemple du Skinning

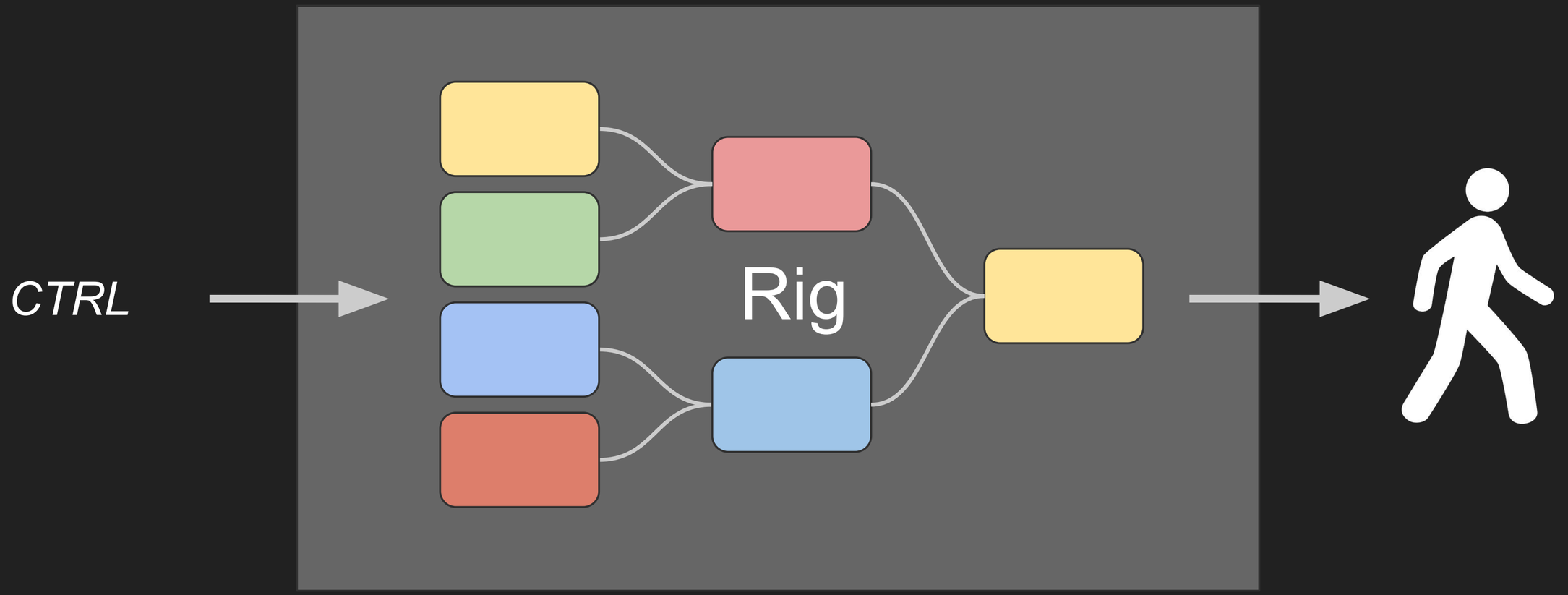


Valeurs initiales: exemple du Skinning



Initial Values: skinning example




CTRL
Rig

RUMBA *for Unreal*

centre national
du cinéma et de
l'image animée

RUMBA *for* ***Unreal***

- Animer avec Rumba dans le context d’Unreal lorsque nécessaire
- Utiliser des rigs de qualités (Maya ou Rumba) dans Unreal
- Garder le temps réel dans Unreal avec de nombreux rigs
- Plus besoin d’exporter d’énormes fichiers Alembic
- Subdivision *Catmull Clark* en direct dans Unreal

centre national
du cinéma et de
l’image animée


Merci !
RUMBA
2.0