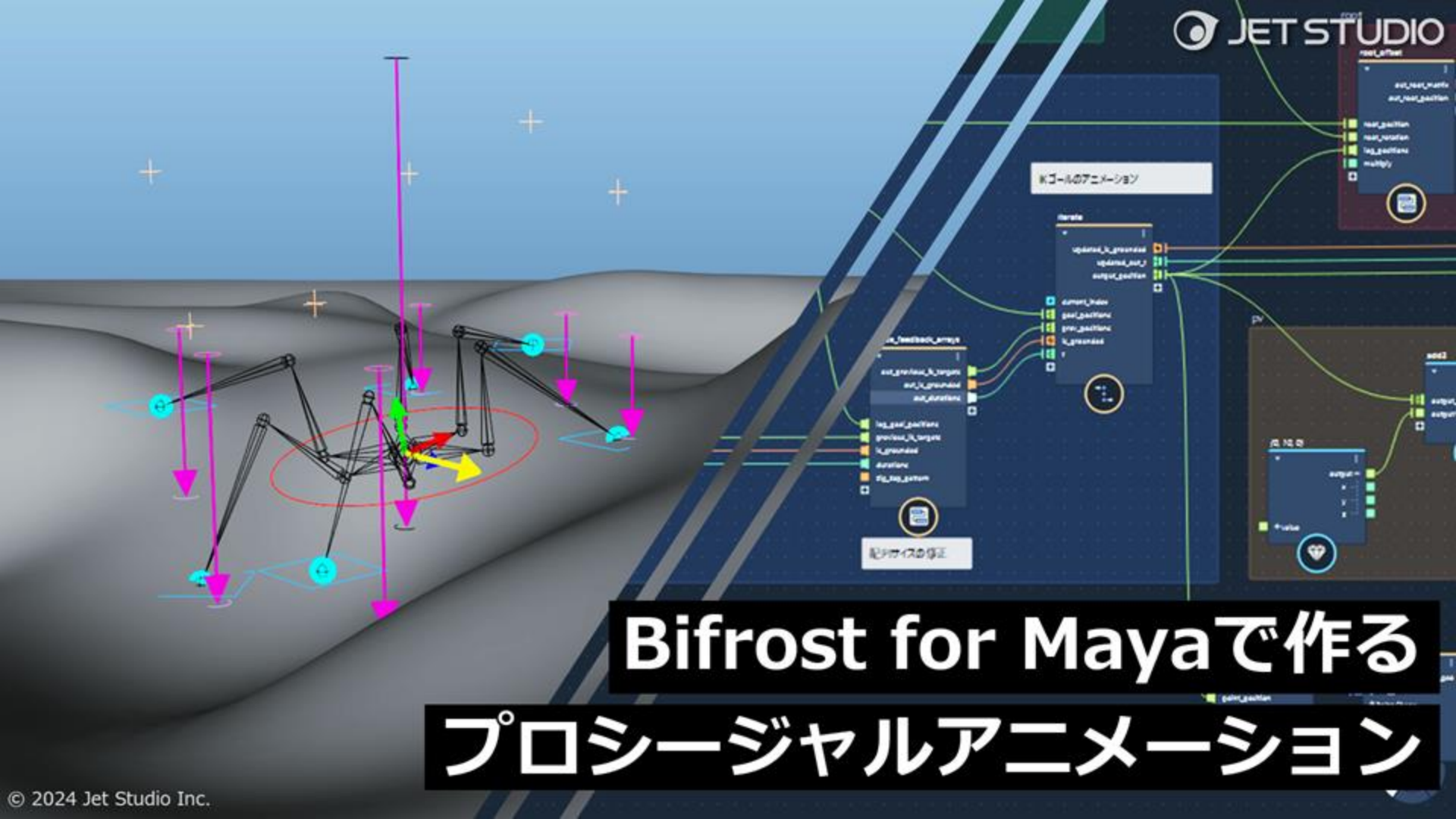


© 2024 Jet Studio Inc.

このセッションで得られるもの

- **Bifrost for Maya で何ができるのか**
  - リグ・アニメーション関連に絞って
  - Maya標準ノードではできなくてBifrostなら出来ること
- **プロシージャルアニメーションのグラフの組み方**
  - 6本足IKのプロシージャルアニメーション
  - 図解、サンプルデータ、実践

※本セッションの内容は具体的なタイトルにおける実績に基づくものではありません。
個人的な実験レベルの内容であることをご了承ください。

© 2024 Jet Studio Inc.

- スライドは後日公開します
- サンプルデータ（Mayaシーン）も公開します
- 動画アーカイブもあるみたいです

© 2024 Jet Studio Inc.

自己紹介

- 赤崎 弘幸 (あかさき ひろゆき)
- 株式会社ジェットスタジオ (2010 ~ 現在)
- CGI Div. チーフディレクター
- 普段の業務
  - アセット (主にキャラモデル+リグ) 系の案件ディレクション多め。
  - CGI Div.現場管理。技術サポート。社内ツール開発。R&D。
  - その他雑用。
- X : @akasaki1211

© 2024 Jet Studio Inc.

Bifrostとは？

© 2024 Jet Studio Inc.

Bifrost for Maya

Bifrost の新しい**ビジュアル プログラミング環境**を使用することにより、3D アーティストと TD は Maya で複雑なエフェクトを素早く簡単に作成できます。TD は Bifrost グラフ エディタを使ってカスタム グラフを作成してスタジオ内のアーティストへの配布用にパッケージ化し、さまざまなショー、シーン、ショットの中から使用してもらうことができます。アーティストにとってもさまざまなグラフをすぐに使えるというメリットが生まれ、火、煙、爆発、砂、雪などのエフェクトを追加設定なしで作成できます。

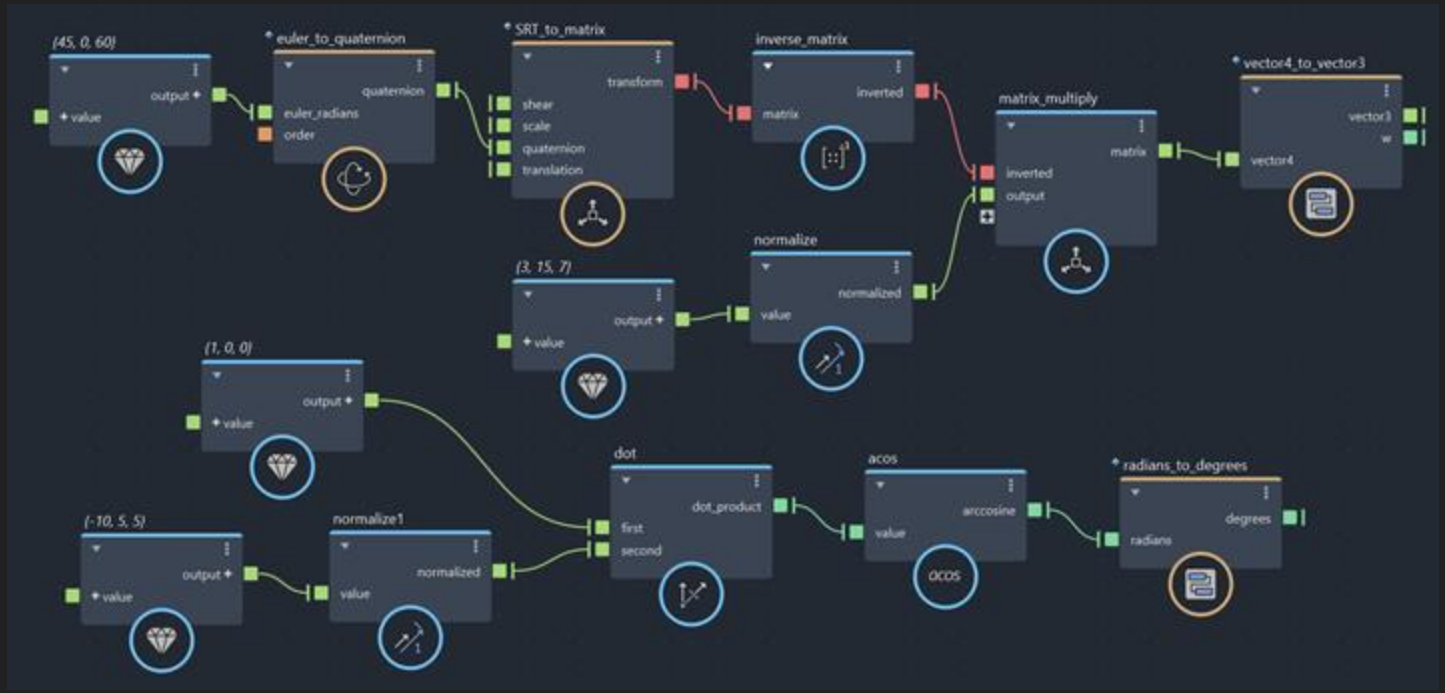
（※Bifrost公式ヘルプより引用）

というヘルプのコピペは置いておいて…

© 2024 Jet Studio Inc.

Bifrost = ビジュアルプログラミング環境

リグでお世話になりがちなベクトルや行列などの演算ノードが豊富！



© 2024 Jet Studio Inc.

Bifrost = ビジュアルプログラミング環境

ある区画のノードグラフを繰り返し処理する“for文“的な処理ができる！

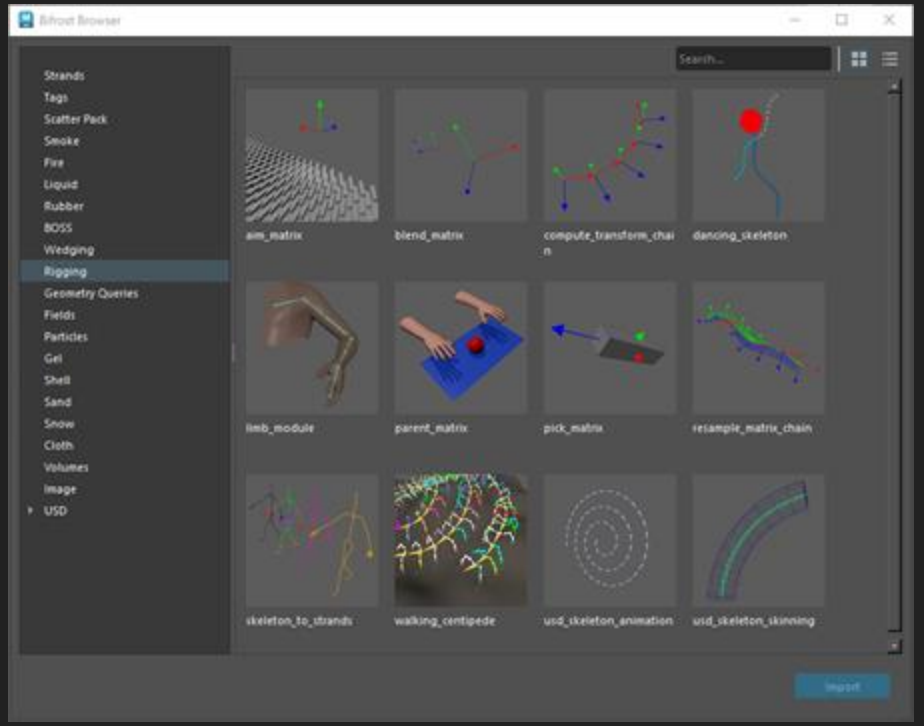
（※標準ノードだけでは「if」はできても「for」はできません）

© 2024 Jet Studio Inc.

Bifrost = ビジュアルプログラミング環境

2.10以降ではRigに特化した機能やサンプルが追加された！

（※画像は2.11のBifrost Browser）



© 2024 Jet Studio Inc.

Maya標準ノードで出来ないことに絞ると

- 反復計算が必要なもの
  → iterate
- 多数の頂点等をまとめて扱うもの
  → 自動ループ
- 前回の計算結果を再利用するもの
  → feedback
- レイキャストなど

※リグやアニメーションにフォーカスして挙げています。
※標準のノードだけではできませんがC++/Python APIを使えばできます。
※expressionノードでもある程度はできます。

© 2024 Jet Studio Inc.

iterate（反復計算）

①複数の要素（配列など）を順番に処理できる

：“反復”ポートアイコン。外部では配列として、内部では単一の値として扱われる。

②1つの値を反復で更新していくことができる

：“State”ポートのアイコン。出力値が次の反復の入力として使用される。

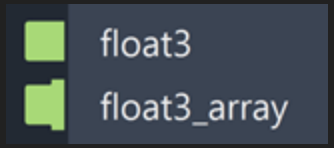
.¥samples

- **sample_01_iterate_01_iteration_target.ma**
- **sample_02_iterate_02_state.ma**

© 2024 Jet Studio Inc.

自動ループ（多数の値をまとめて処理）

入力ポートを単一の値にしておいて配列を接続すると、配列のすべての要素に自動で同じ処理を繰り返してくれる。



・□型は単一の値
・凸型は配列

：“自動ループ”発生時のアイコン

.¥samples
- sample_03_auto_loop_01.ma

© 2024 Jet Studio Inc.

Feedbackポート（値の再利用）

前回の評価結果をキャッシュし次回の評価の入力として使用できる。主にシミュレーションで使う。

 ：“Feedback”ポートのアイコン。結果をキャッシュして次回再利用。

.¥samples

- **sample_04_feedback_01.ma**

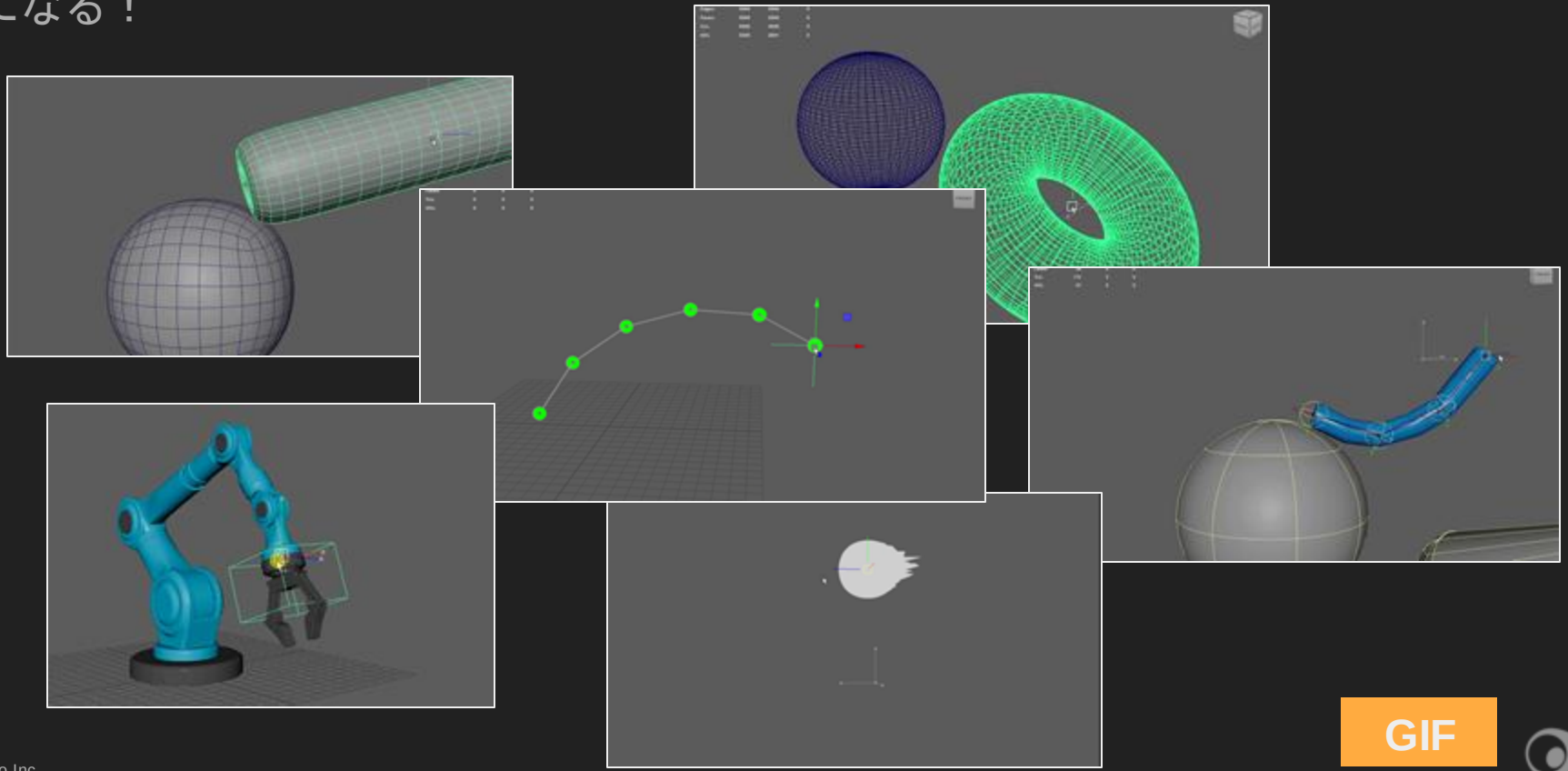
© 2024 Jet Studio Inc.

Bifrost ヘルプ 関連ページ

- ポート タイプとアイコン
- プログラミング ループを操作する
- iterate ループを作成する
- 配列を操作する
- カスタム シミュレーションを作成する

© 2024 Jet Studio Inc.

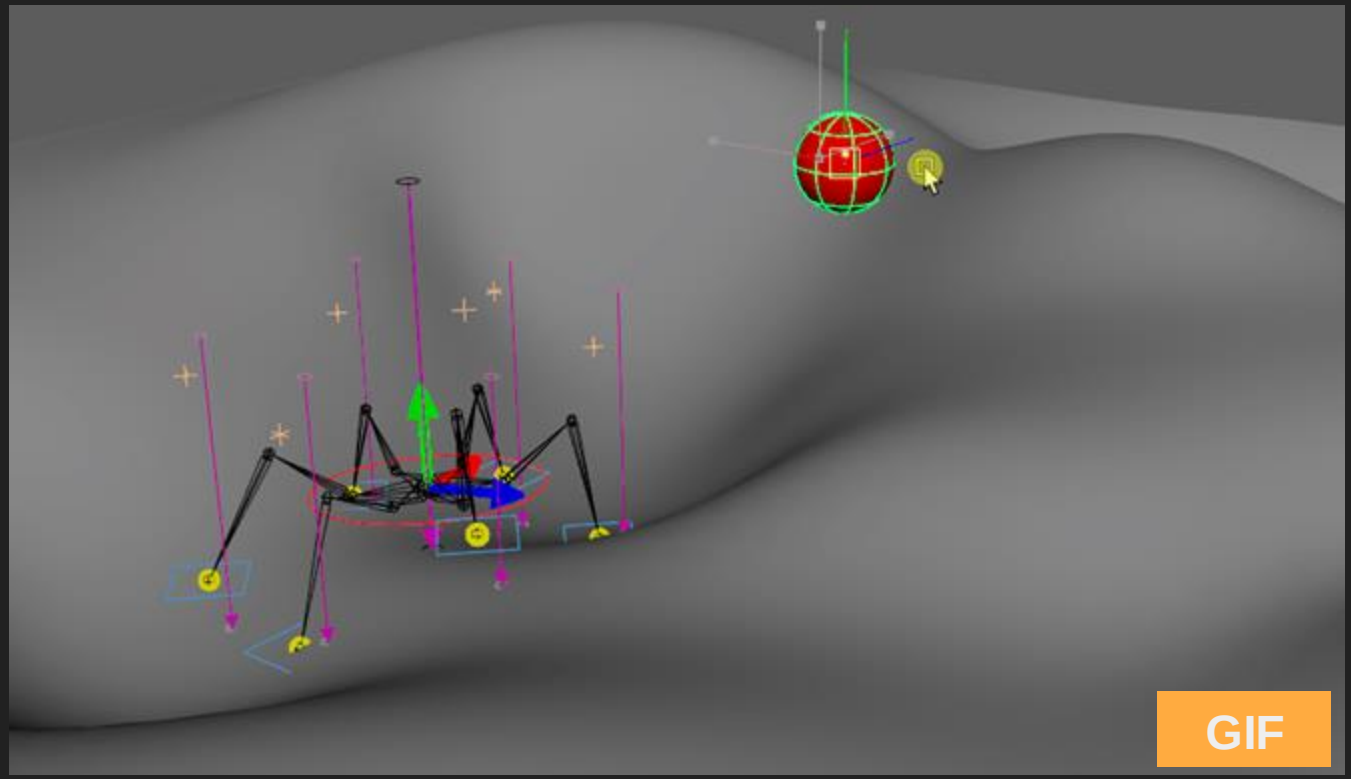
これらの機能を使うとMaya標準ノードでは出来なかったことがいろいろ出来るようになる！



GIF

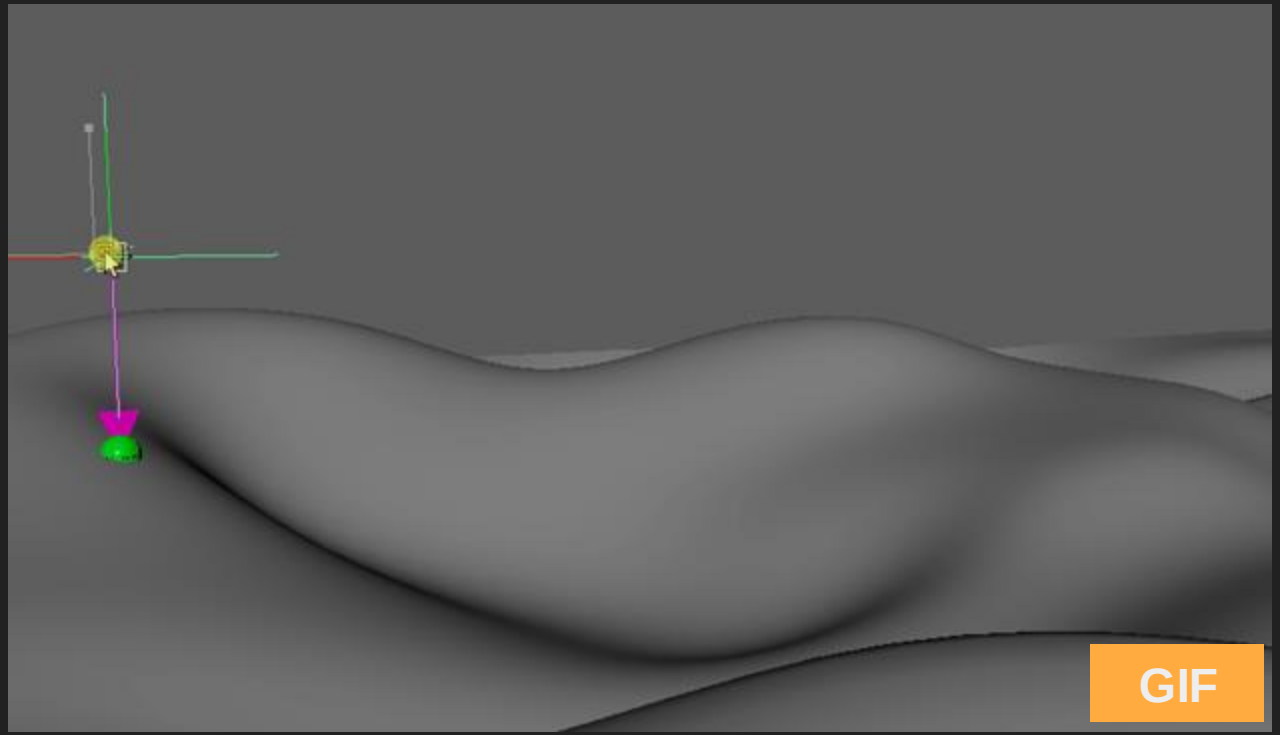
© 2024 Jet Studio Inc.

今日はこれを作ります



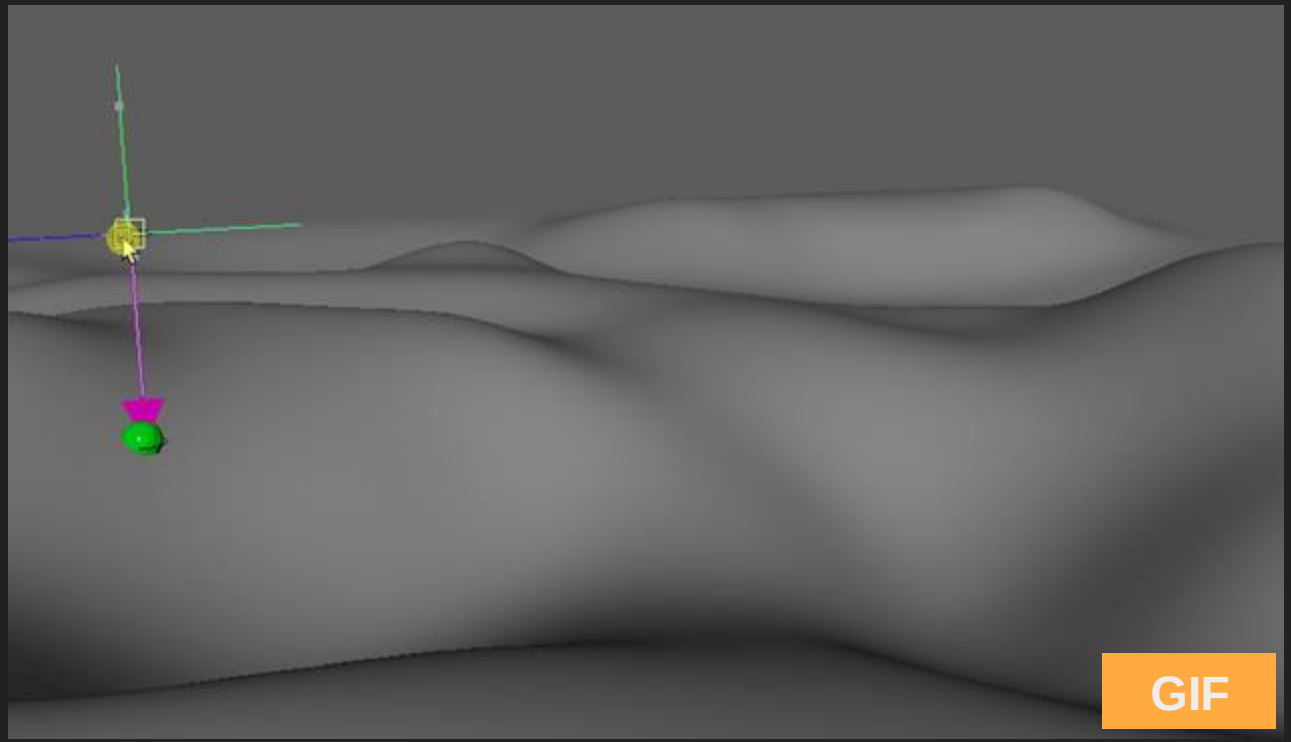
© 2024 Jet Studio Inc.

Step1：レイキャスト



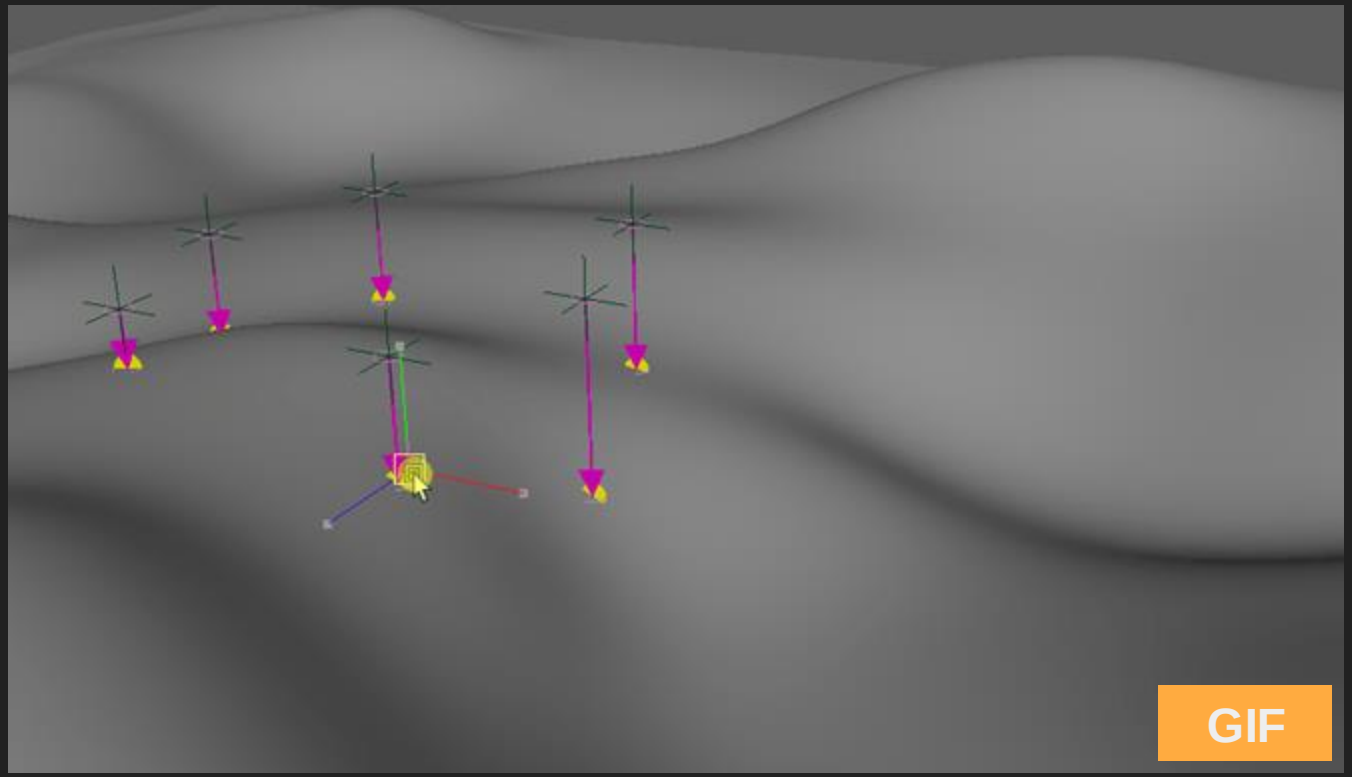
© 2024 Jet Studio Inc.

Step2 : ステップ



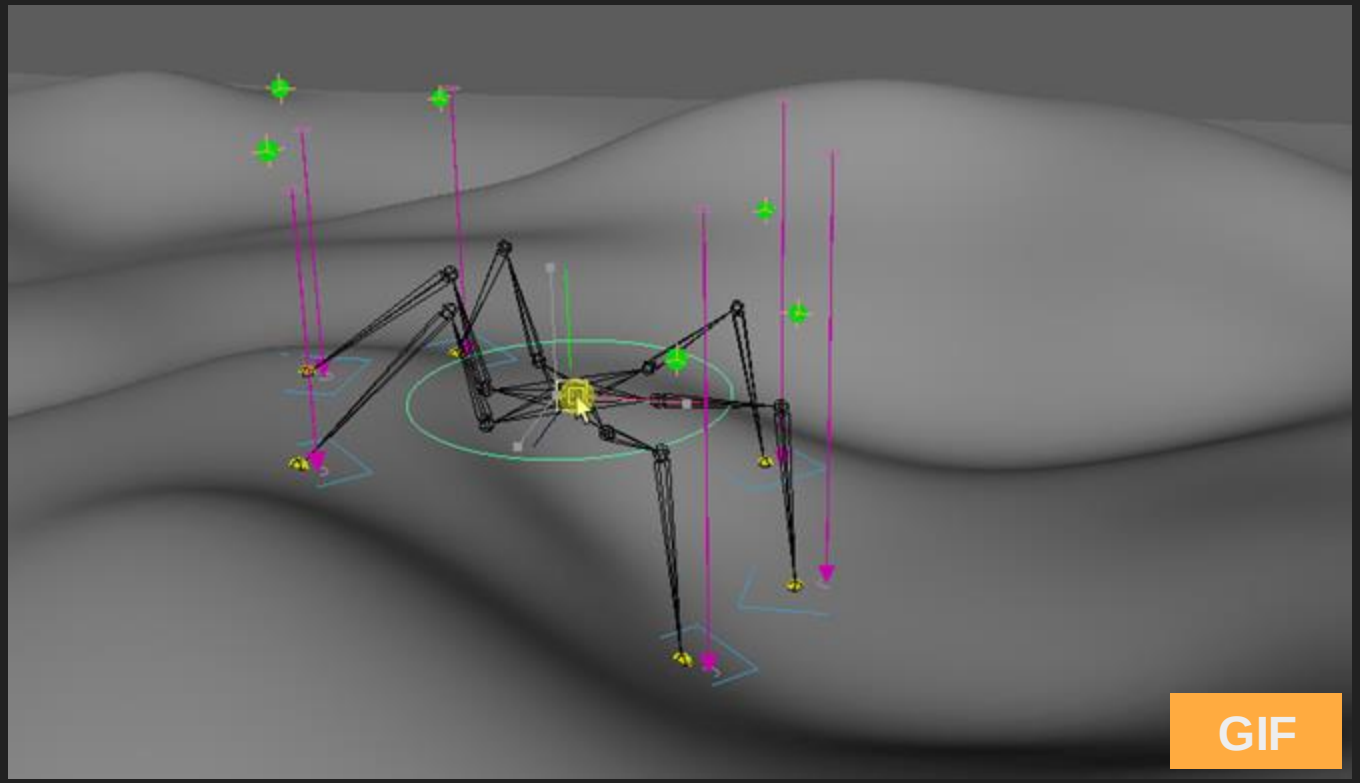
© 2024 Jet Studio Inc.

Step3 : 6つに増やす



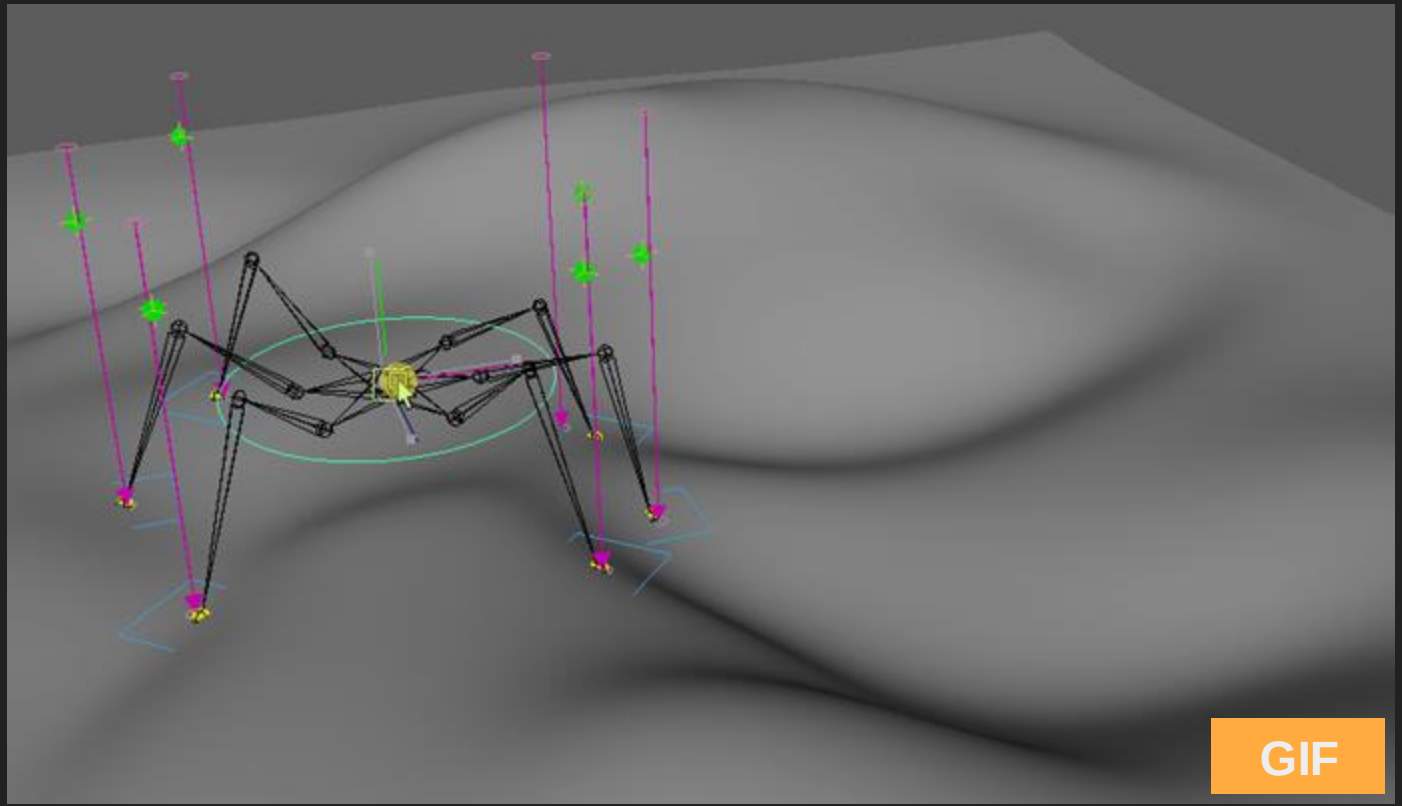
© 2024 Jet Studio Inc.

Step4 : リグを拘束



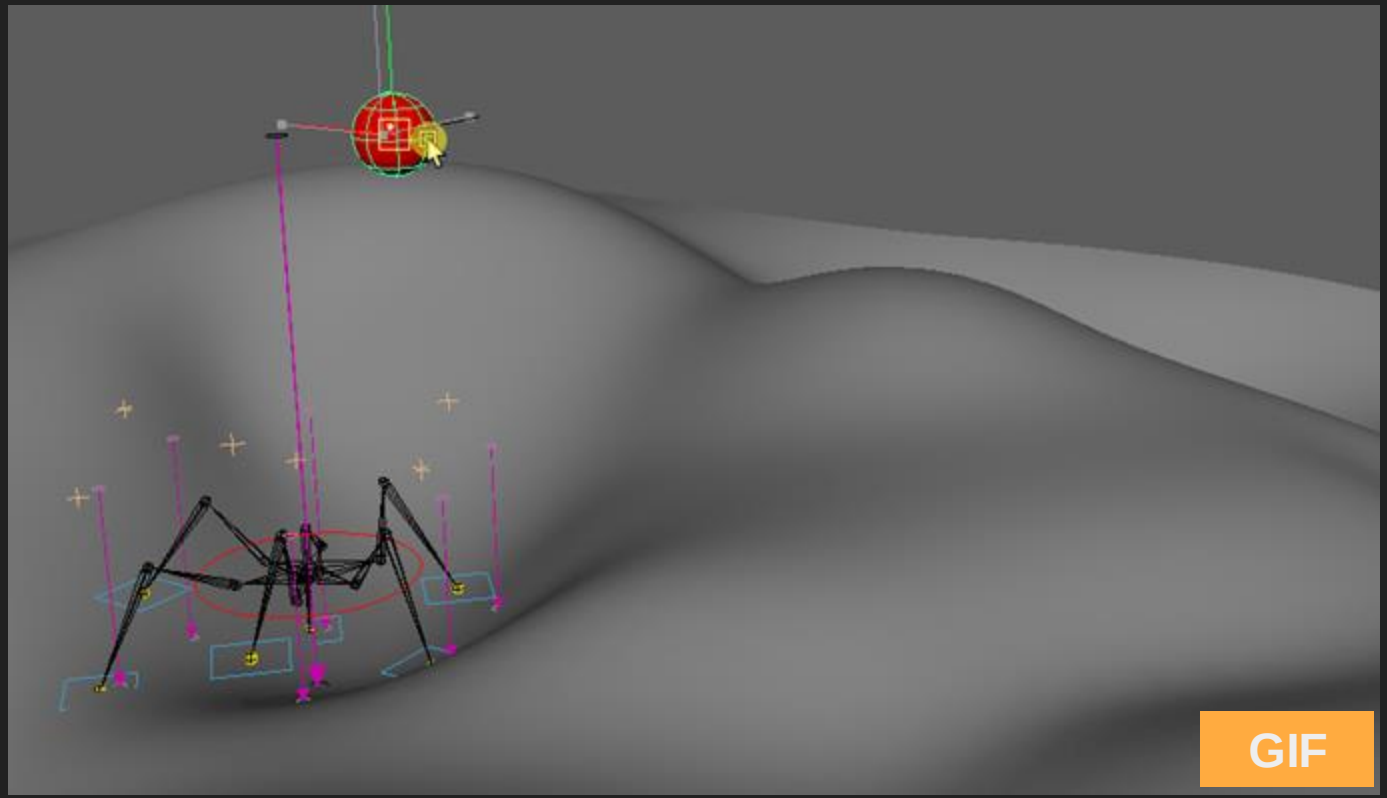
© 2024 Jet Studio Inc.

Step5 : 隣の足の接地確認



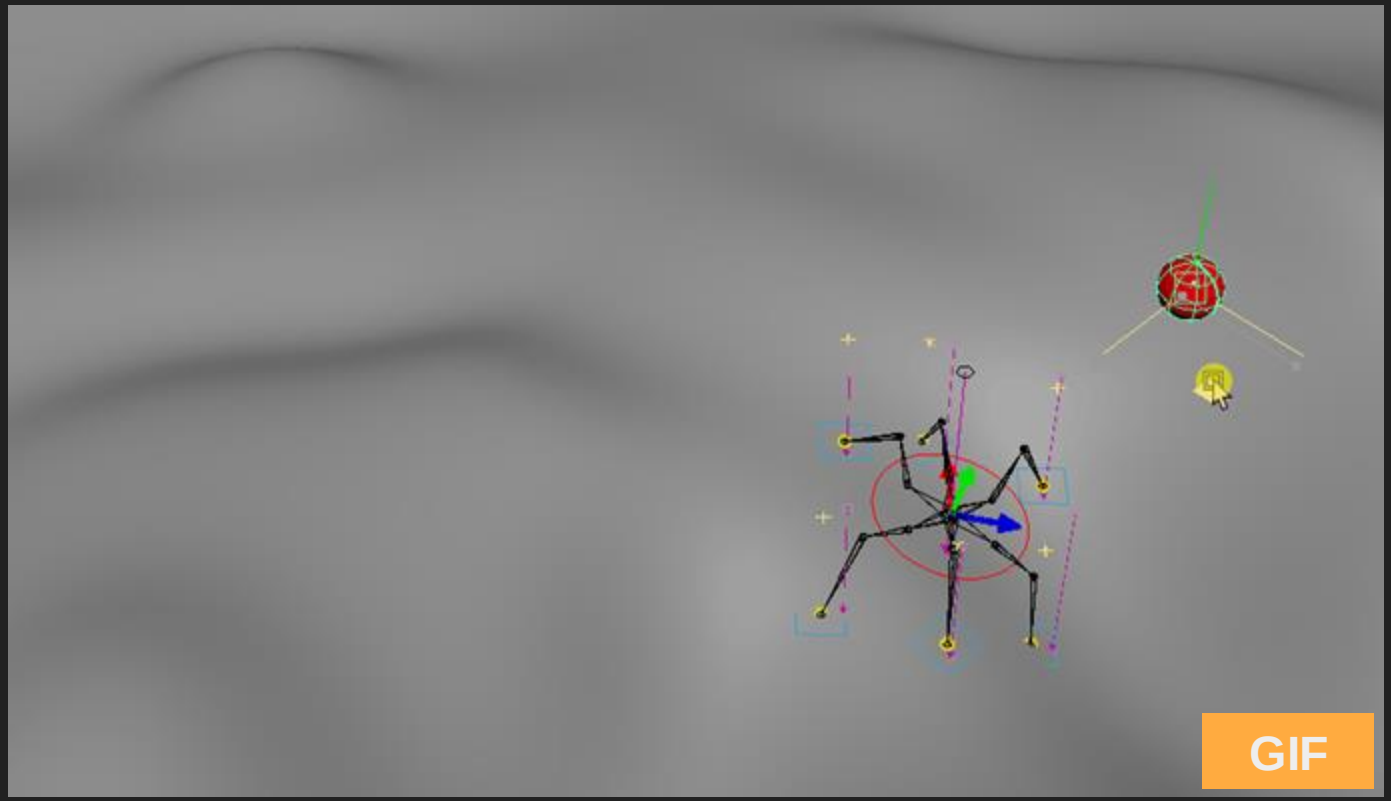
© 2024 Jet Studio Inc.

Step6：ルートの移動



© 2024 Jet Studio Inc.

Step7：ルートの回転



© 2024 Jet Studio Inc.

実践

© 2024 Jet Studio Inc.

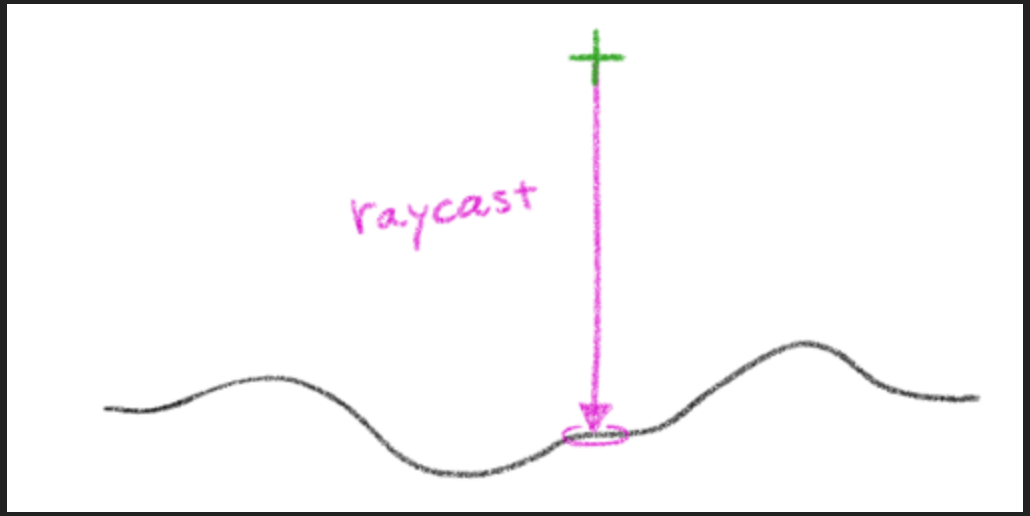
環境

- Windows 11
- Autodesk Maya 2025.2
- Bifrost 2.11.0.0

© 2024 Jet Studio Inc.

Step1

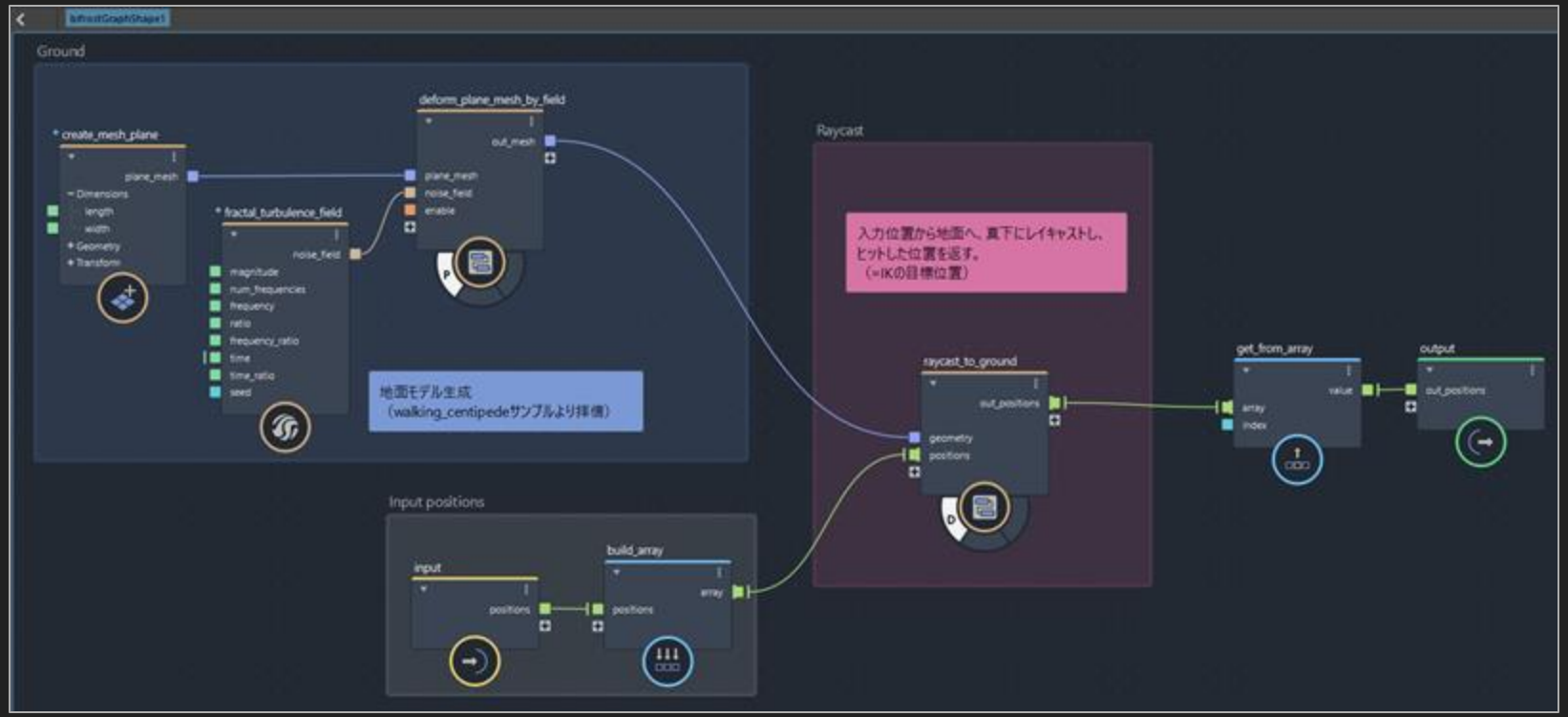
地面より少し上の位置から真下へレイキャストしてIKゴールの目標位置を決める



© 2024 Jet Studio Inc.

Step1

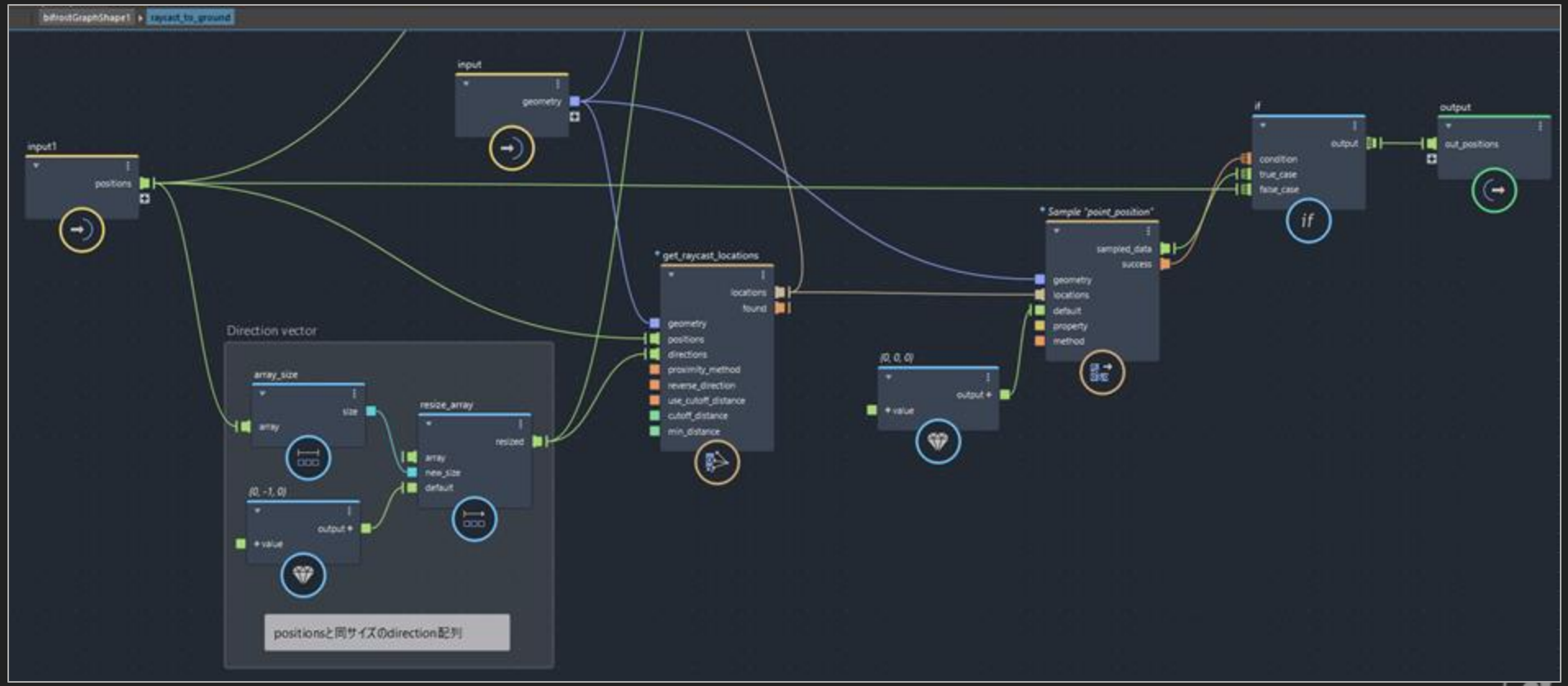
- **step01_raycast_to_ground.ma**



© 2024 Jet Studio Inc.

Step1

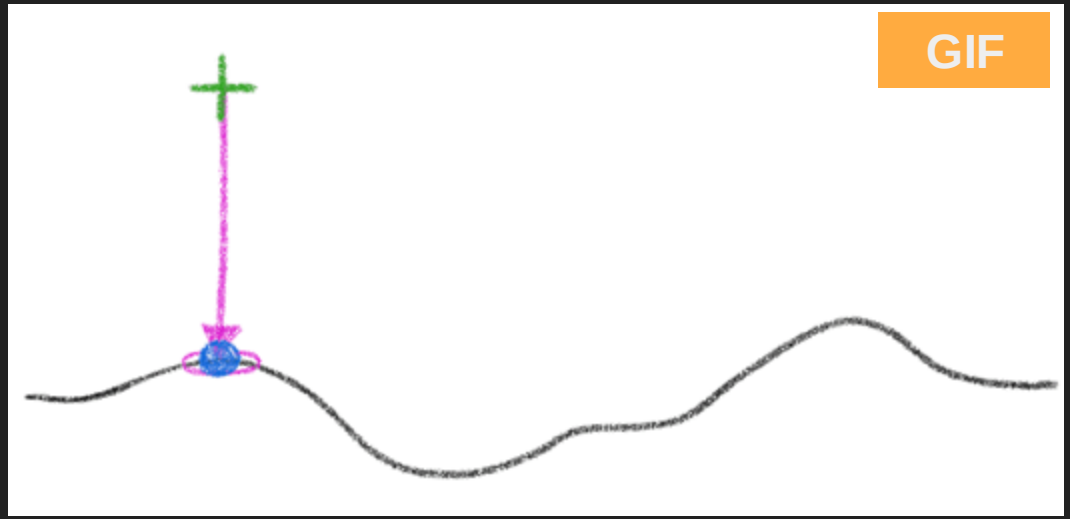
- **step01_raycast_to_ground.ma**



© 2024 Jet Studio Inc.

Step2a

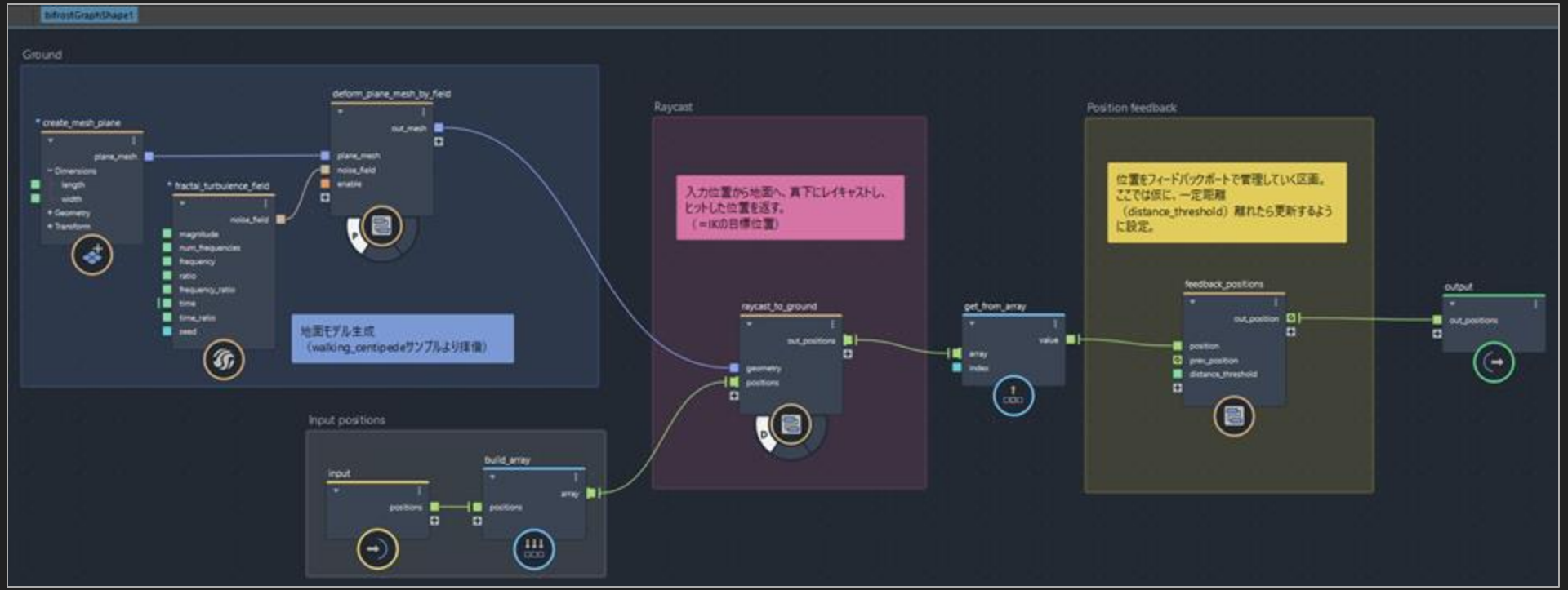
「現在位置」と「目標位置」の間の距離が一定距離を越えた場合のみ更新する



© 2024 Jet Studio Inc.

Step2a

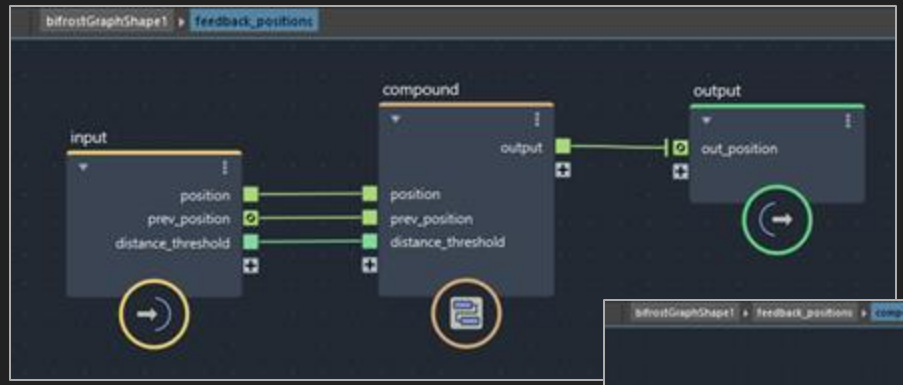
- **step02a_feedback.ma**

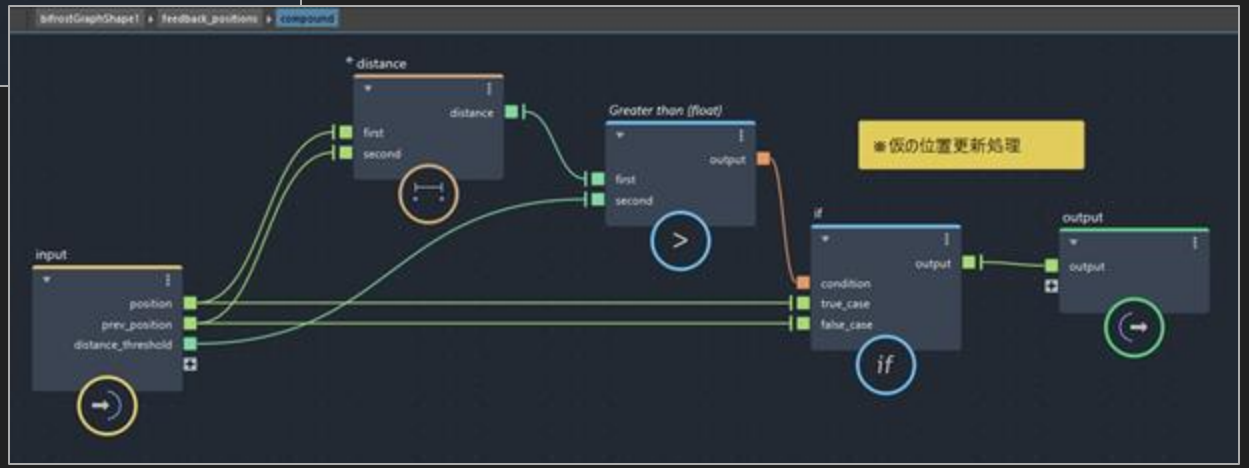


© 2024 Jet Studio Inc.

Step2a

- **step02a_feedback.ma**





© 2024 Jet Studio Inc.

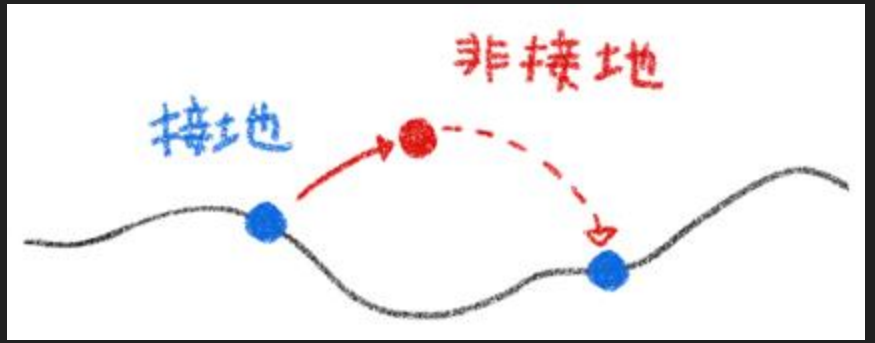
Step2b

が、瞬間移動してもらっては困るので...

時間をかけて遷移出来るよう2つの状態（接地／非接地）を管理する。

状態変化には以下の判定が必要：

- （接地中）非接地に移行しステップを開始するかどうか
- （非接地中）ステップを終了して接地状態に移行するか

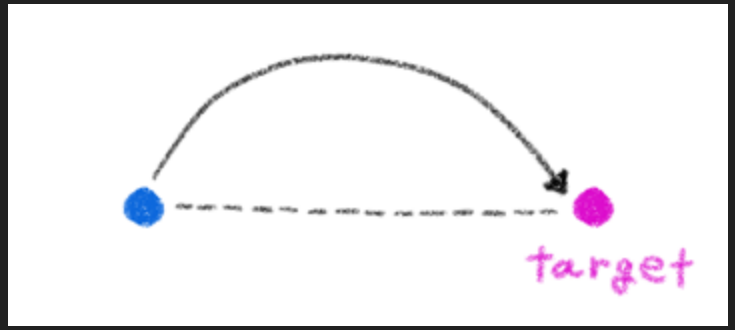


© 2024 Jet Studio Inc.

Step2b

その前に...

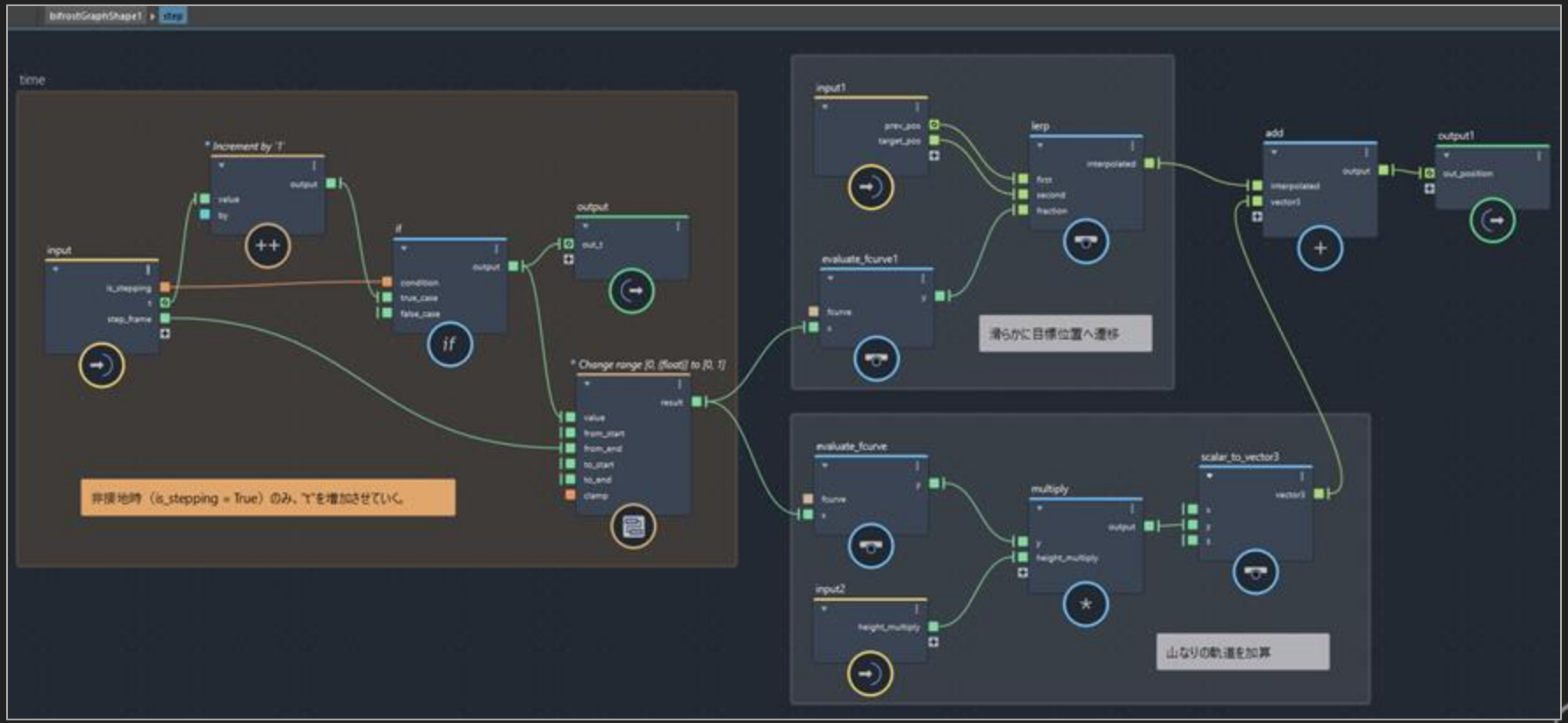
- 指定時間で「現在位置」から「目標位置」に滑らかに遷移させるグラフを作っておく
- 遷移中、山なりに動くようにY軸方向の位置補正も加える



© 2024 Jet Studio Inc.

Step2b

- **step02b_lerp_step.ma**



© 2024 Jet Studio Inc.

Step2c

話を戻して...

（接地中）非接地に移行しステップを開始するかどうか

→ 現在位置と目標位置の間が一定距離を超えたら、非接地状態に移行

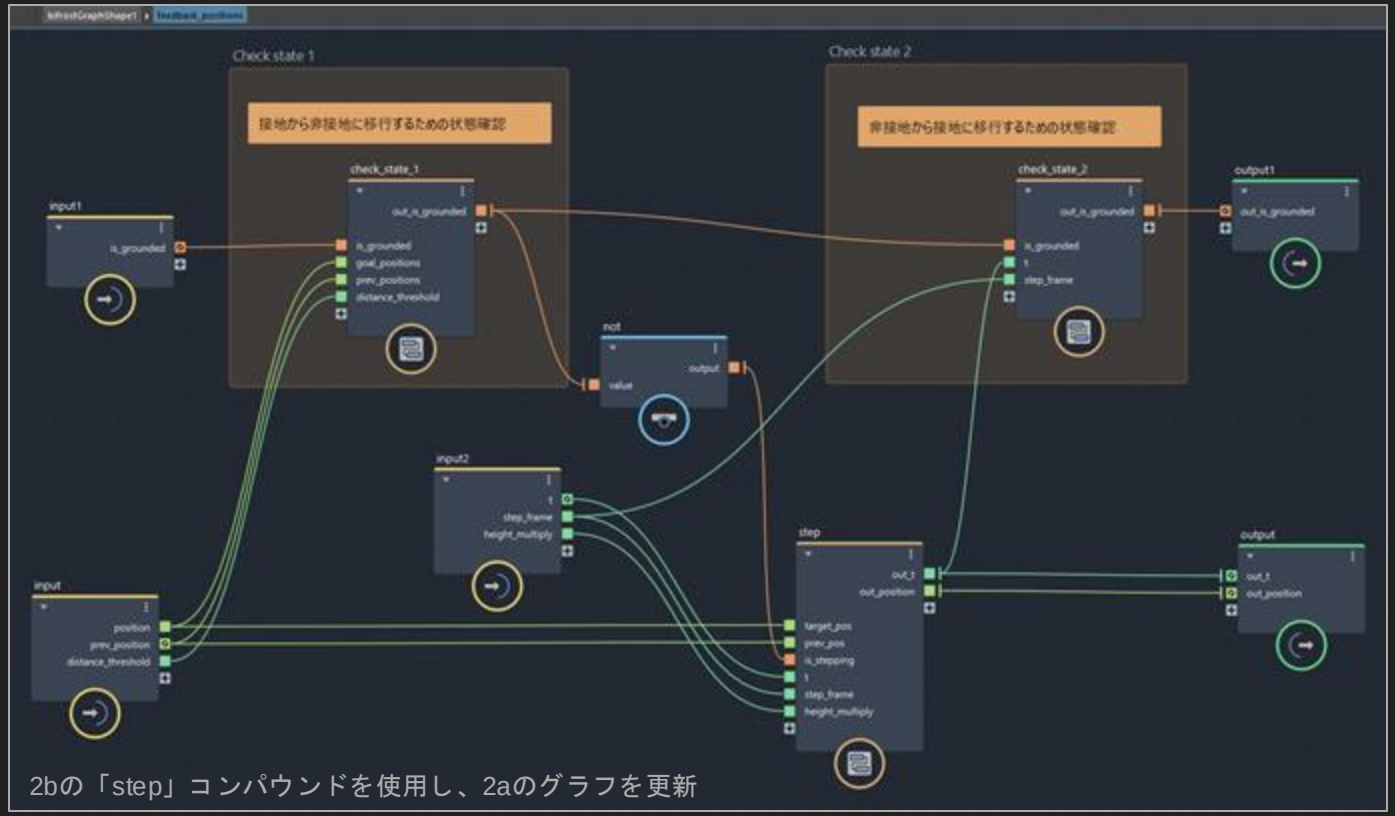
（非接地中）ステップを終了して接地状態に移行するか

→ 目標位置に到達し遷移を終えたら、接地状態に移行

© 2024 Jet Studio Inc.

Step2c

- **step02c_check_state.ma**

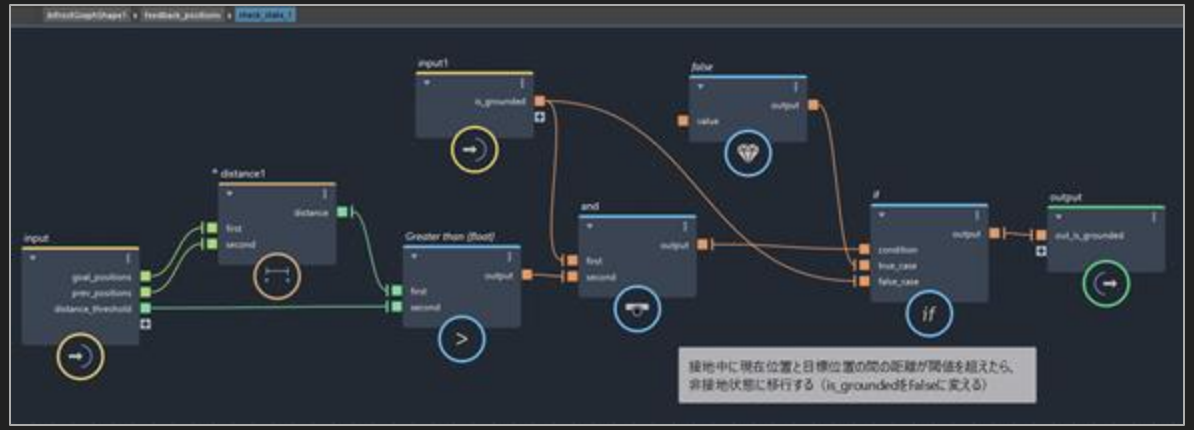


2bの「step」コンパウンドを使用し、2aのグラフを更新

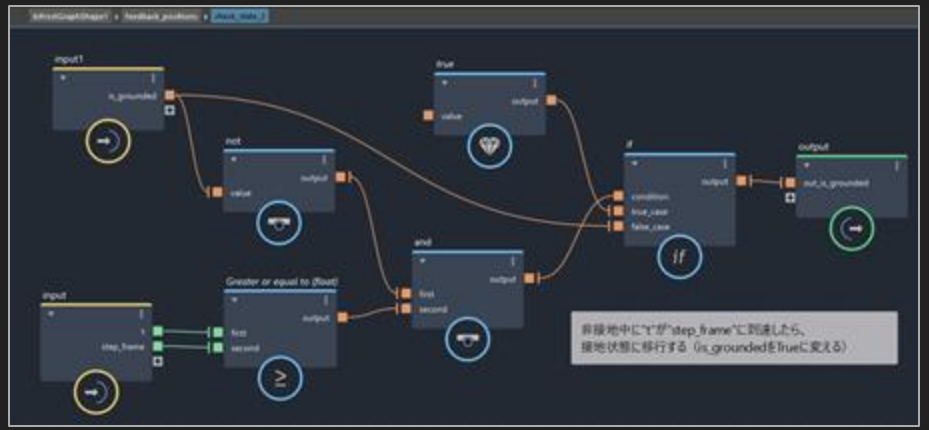
© 2024 Jet Studio Inc.

Step2c

- **step02c_check_state.ma**



状態チェック2種

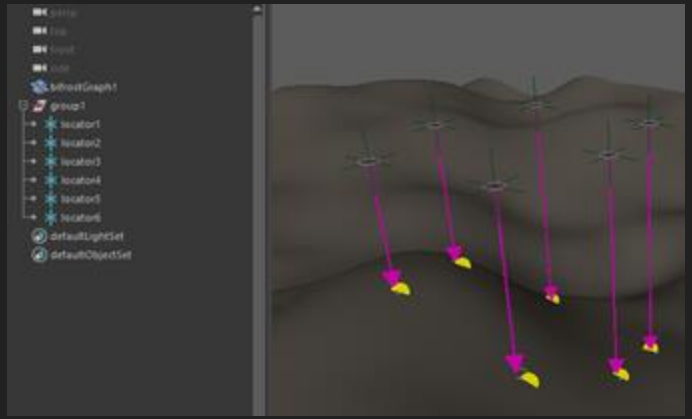


© 2024 Jet Studio Inc.

Step3

ルート＋6点に拡張

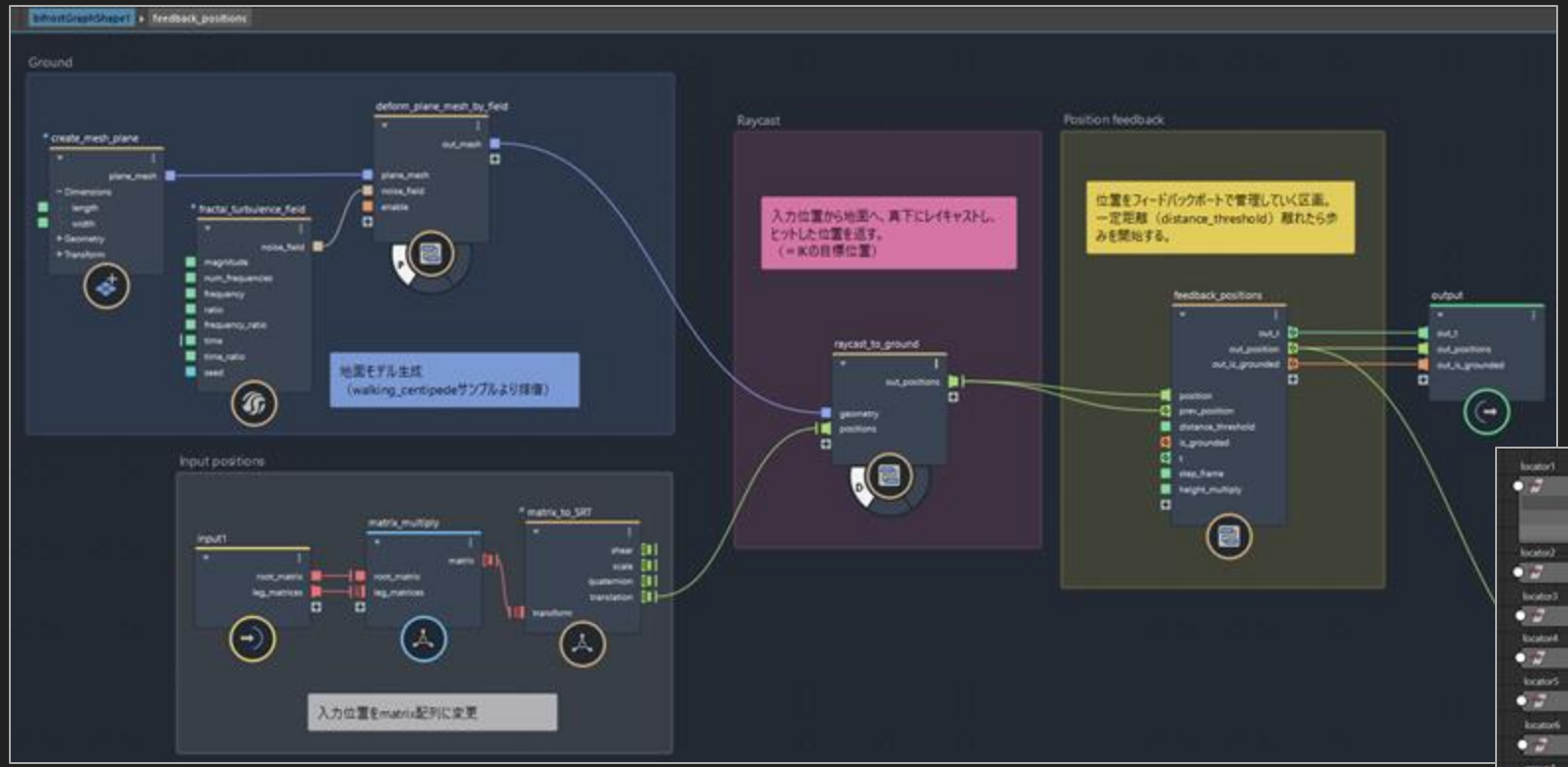
- IKゴール位置の入力を配列に変える
- Feedback部分の入出力を配列に変える
- 配列サイズのズレを修正するため、walking_centipedeサンプルより「**resize_feedback_arrays**」をコピーしてくる

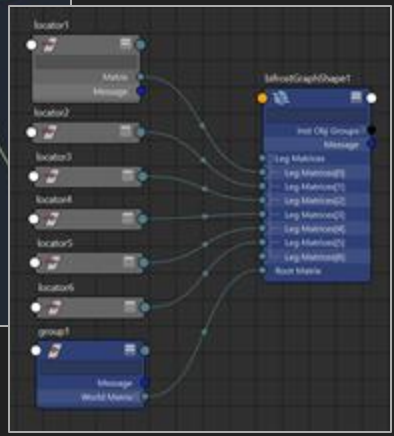


© 2024 Jet Studio Inc.

Step3

- **step03_6legs.ma**

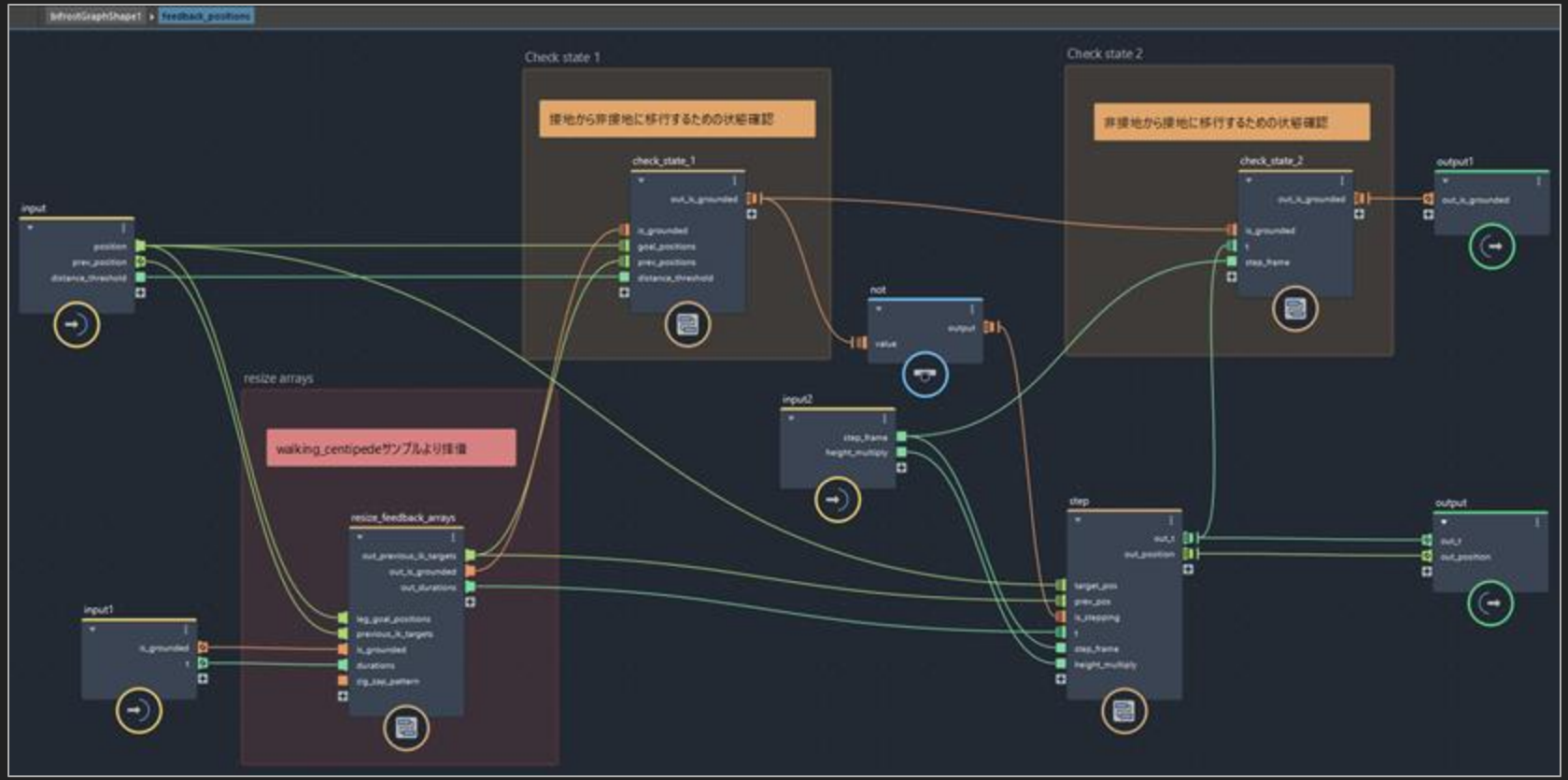




© 2024 Jet Studio Inc.

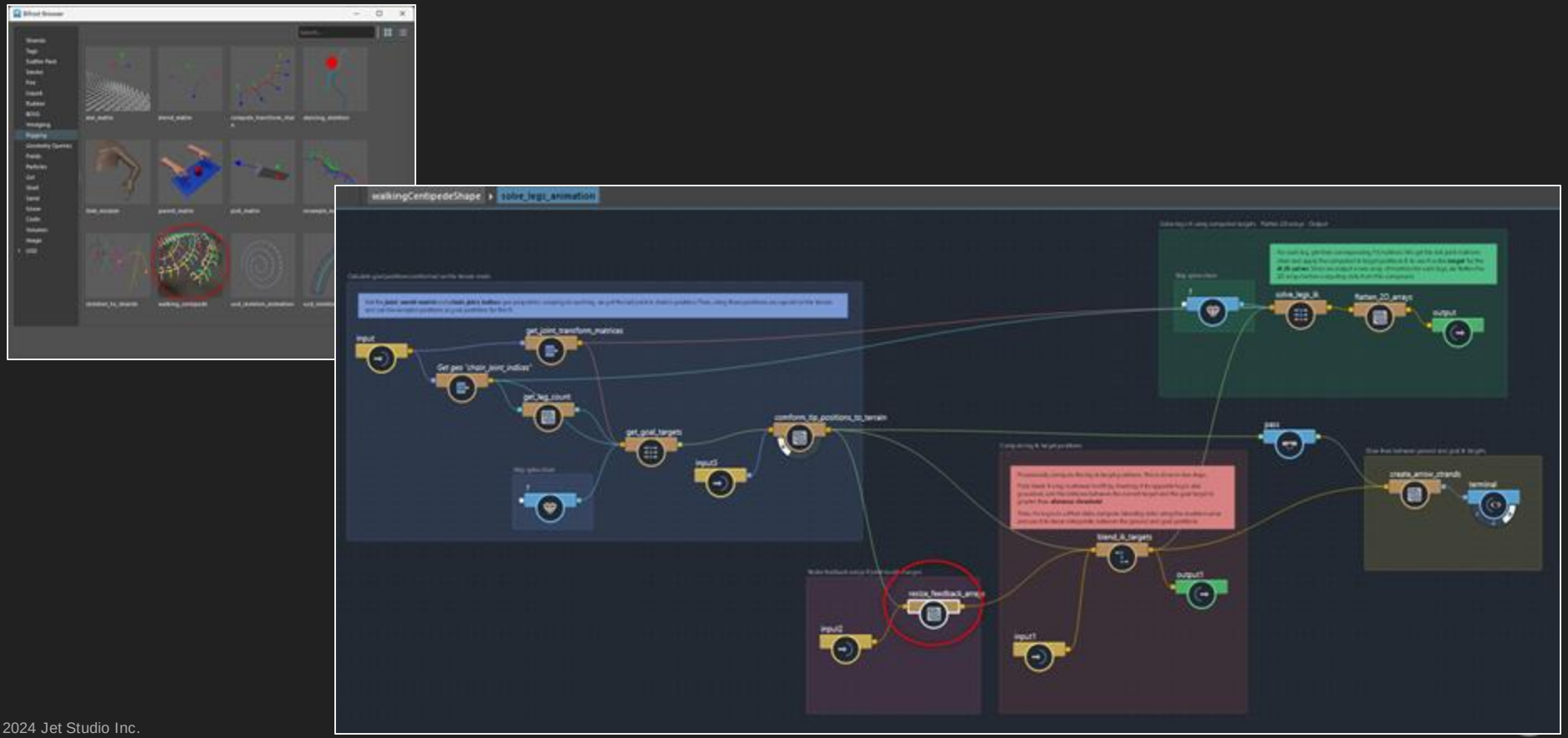
Step3

- **step03_6legs.ma**



© 2024 Jet Studio Inc.

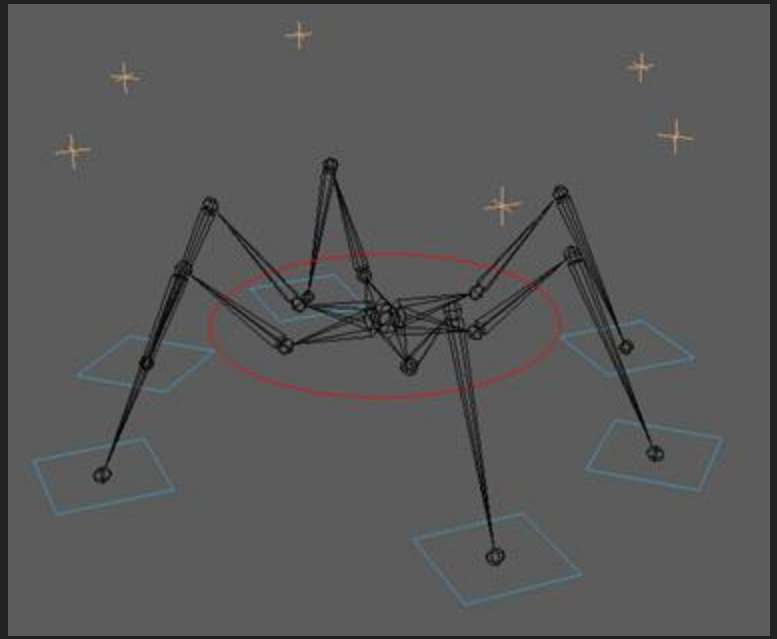
※walking_centipedeサンプルの「**resize_feedback_arrays**」



© 2024 Jet Studio Inc.

Step4

- リグに繋いでみる（ルート動かすと足が自動で動く状態にしてみる）
- pole vector用には、IKゴールから真上にオフセットした位置を追加出力

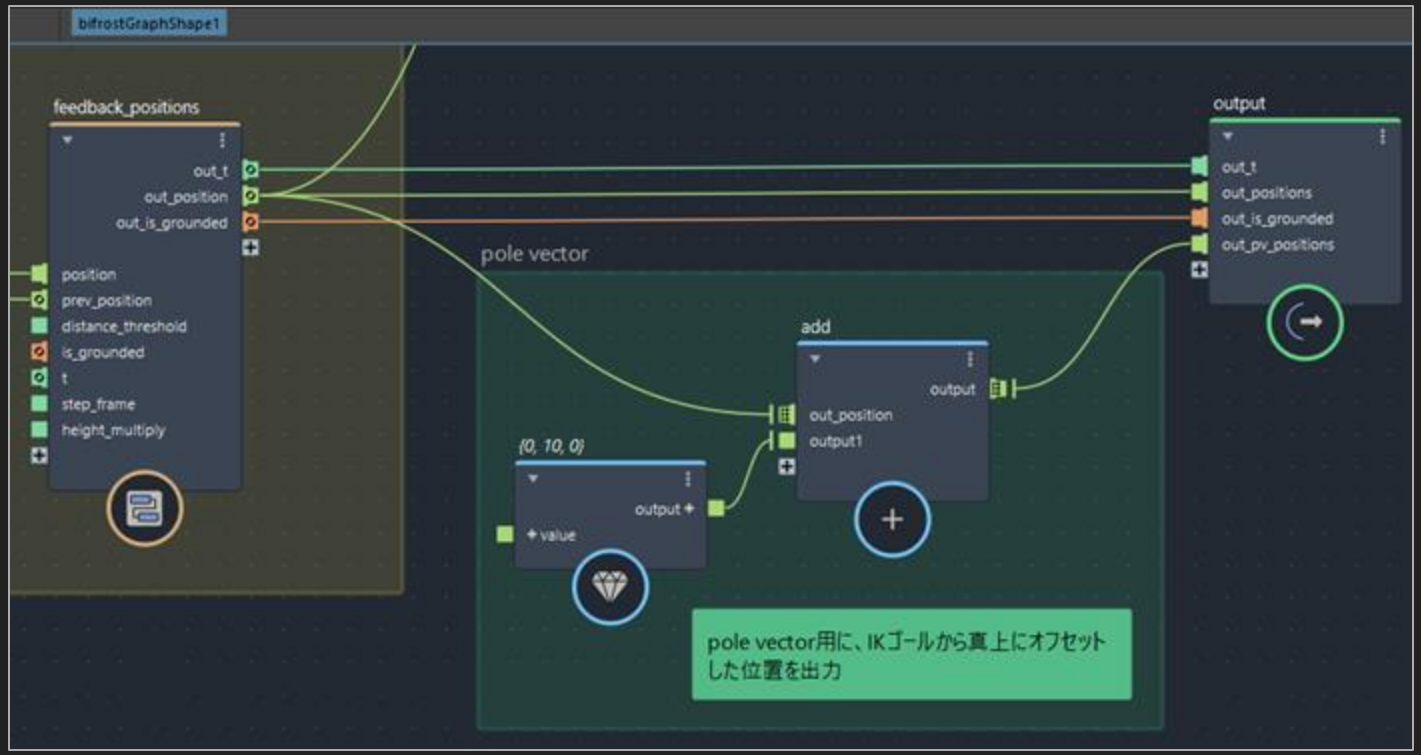


./assets
- **legs_rig.ma**

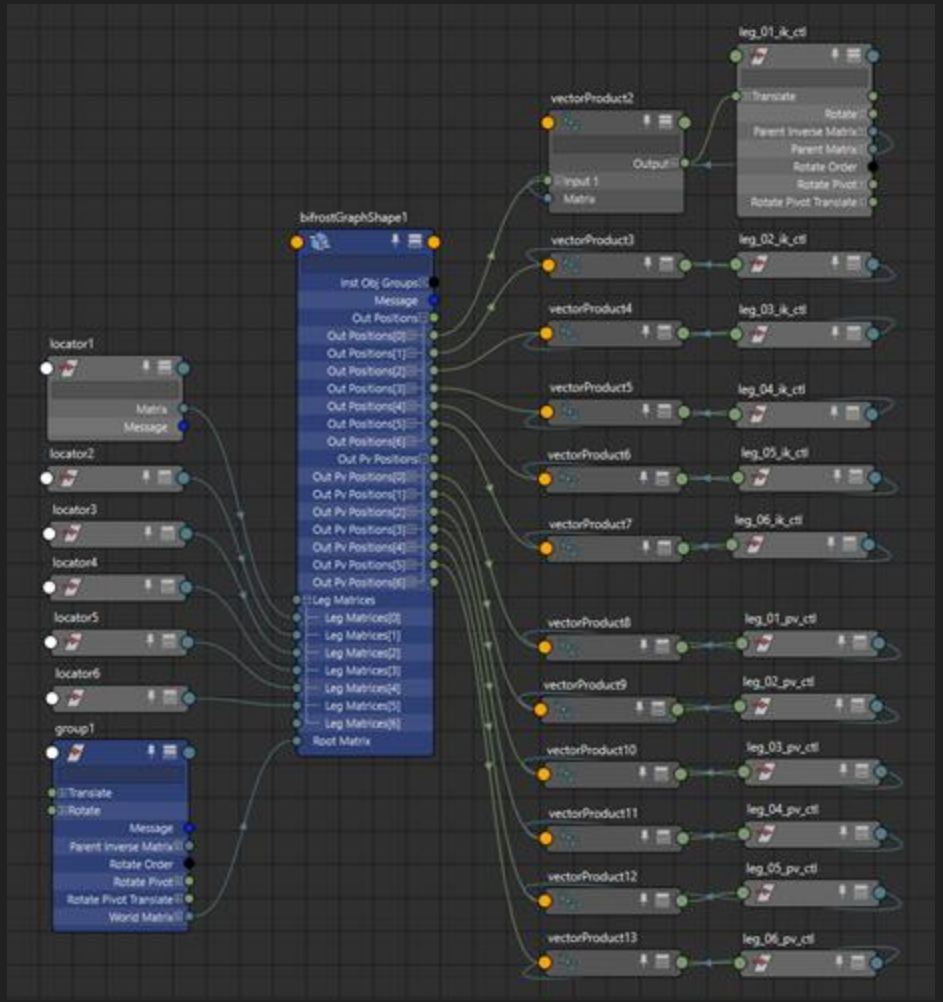
© 2024 Jet Studio Inc.

Step4

- **step04_to_rig.ma**



© 2024 Jet Studio Inc.



- **step04_to_rig.ma**

```python
# ================================
# 手でつなぐのが面倒な人向け
# ================================
from maya import cmds

bf = 'bifrostGraphShape1'

# out_position -> ik_ctl 接続
for i in range(6):
    ctl = 'leg_0{}_ik_ctl'.format(i+1)
    vp = cmds.createNode('vectorProduct')
    cmds.setAttr(vp + '.operation', 4)
    cmds.connectAttr(bf + '.out_positions[{}]'.format(i), vp + '.input1')
    cmds.connectAttr(ctl + '.parentInverseMatrix[0]', vp + '.matrix')
    cmds.connectAttr(vp + '.output', ctl + '.translate')

# out_pv_position -> pv_ctl 接続
for i in range(6):
    ctl = 'leg_0{}_pv_ctl'.format(i+1)
    vp = cmds.createNode('vectorProduct')
    cmds.setAttr(vp + '.operation', 4)
    cmds.connectAttr(bf + '.out_pv_positions[{}]'.format(i), vp + '.input1')
    cmds.connectAttr(ctl + '.parentInverseMatrix[0]', vp + '.matrix')
    cmds.connectAttr(vp + '.output', ctl + '.translate')
```

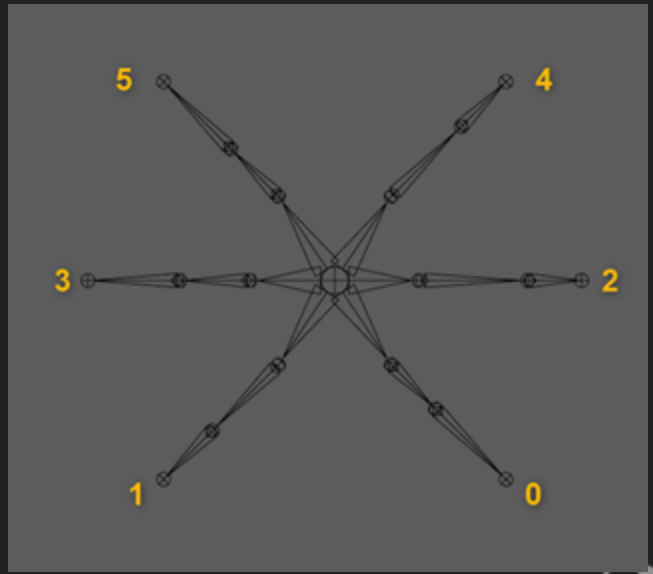
© 2024 Jet Studio Inc.

Step5

隣接する足が同時に歩みを始めると浮いてしまうので、状態チェックに「隣の足が接地しているか」を追加する

- 奇数：-1, +2, -2 の足がすべて接地しているか
- 偶数：+1, +2, -2 の足がすべて接地しているか

を確認する　※+2, -2が範囲外だった場合は接地とみなす

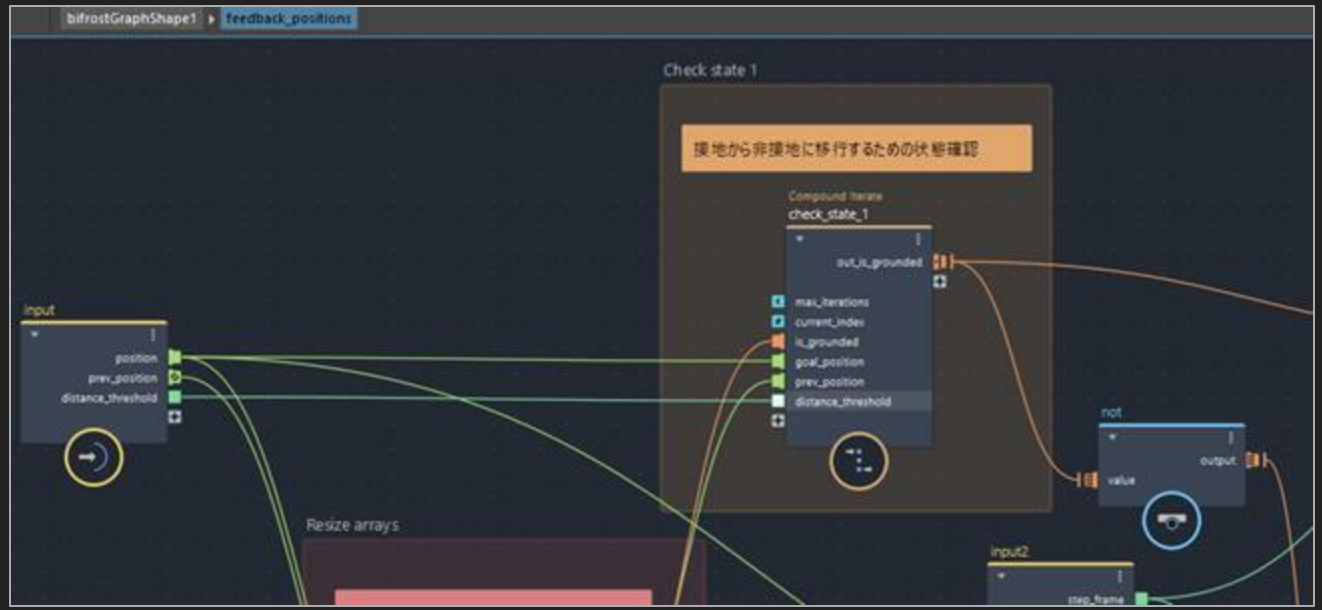


© 2024 Jet Studio Inc.

Step5

- **step05_check_neighbor_legs.ma**

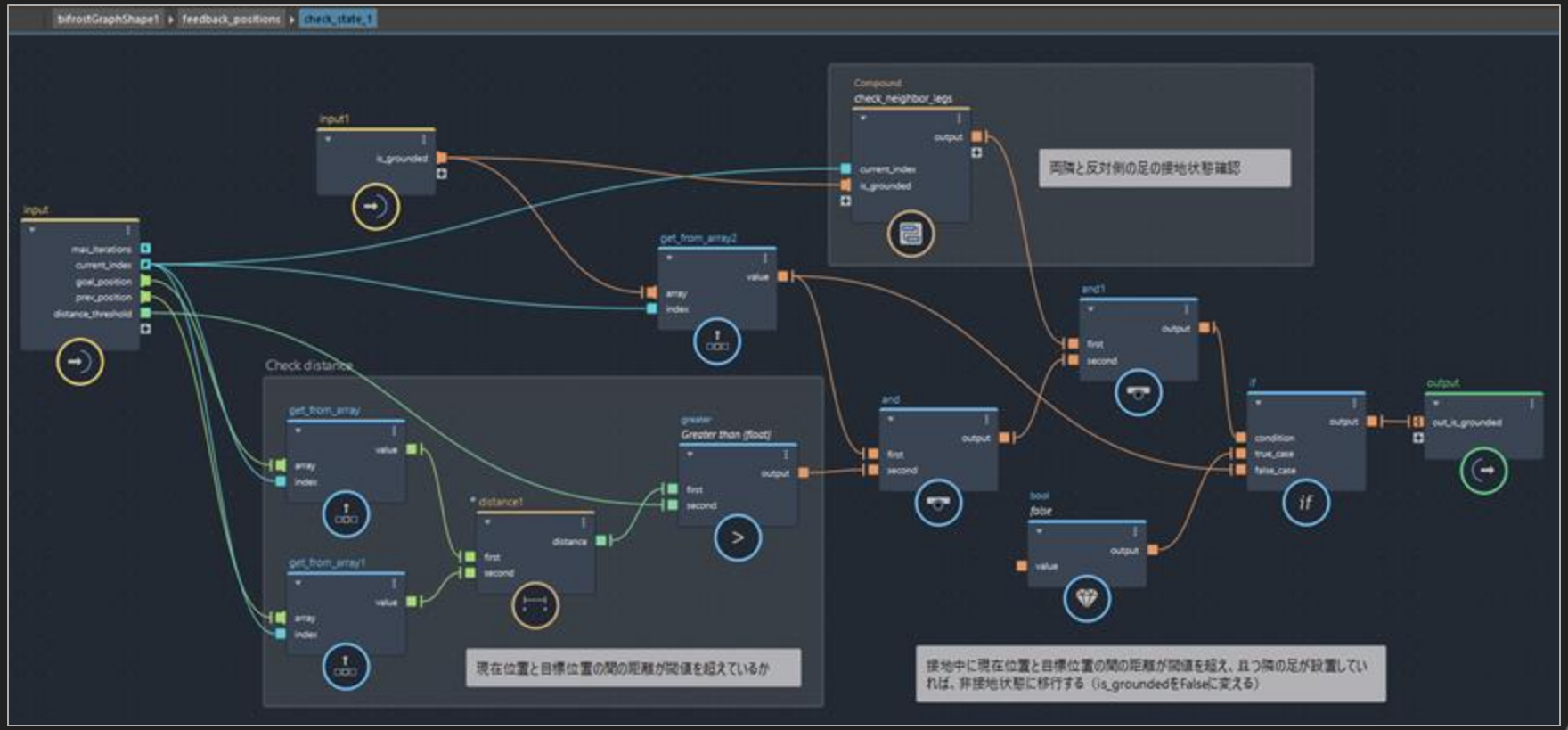
「**check_state_1**」を自動ループではなくiterateに変更し（current_indexが必要なため）、隣の足の接地判定を追加する



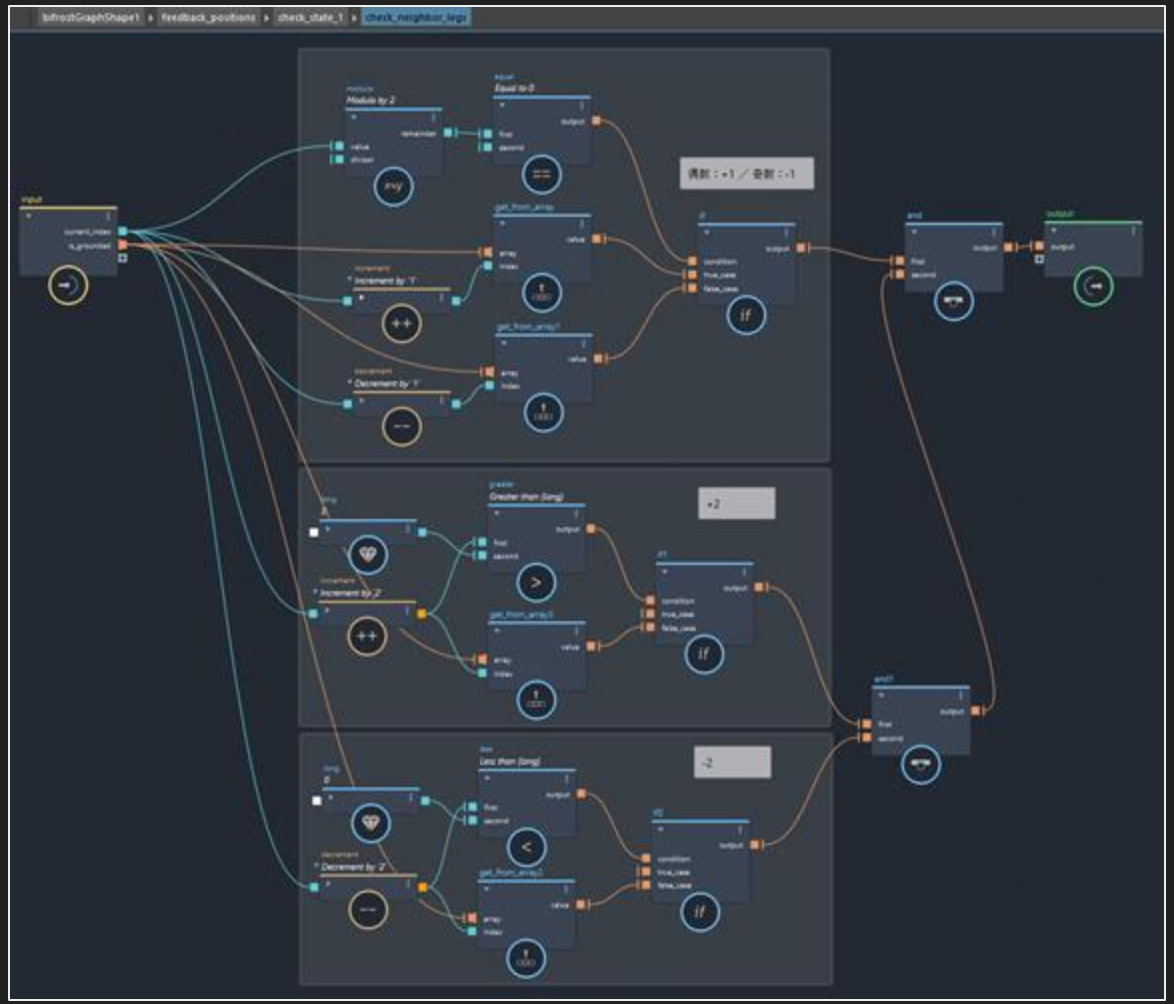
© 2024 Jet Studio Inc.

Step5

- **step05_check_neighbor_legs.ma**



© 2024 Jet Studio Inc.



- **step05_check_neighbor_legs.ma**

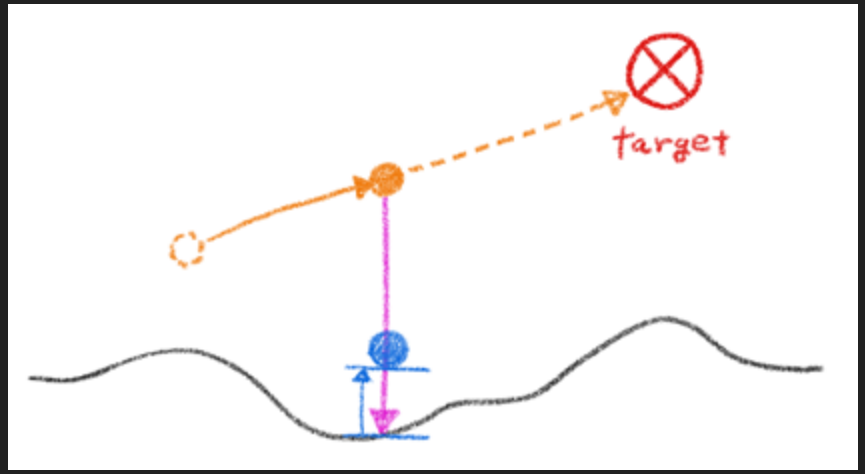
© 2024 Jet Studio Inc.

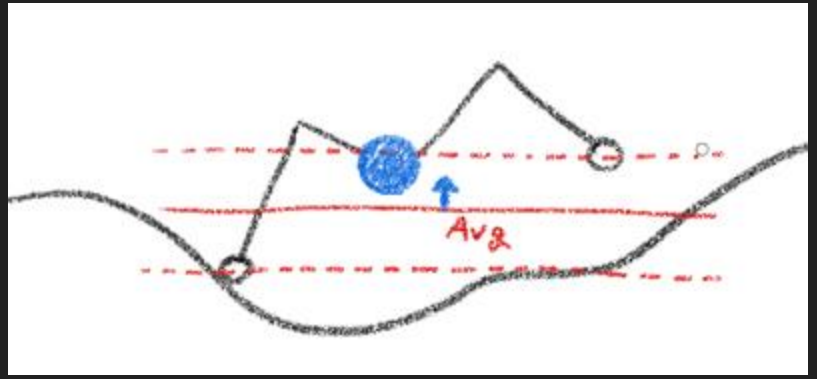
Step6

.¥samples
- sample_05_position_interp.ma

ルートの移動も自動にする

- ターゲット位置に向かって速度制限付きで補間 → 真下にレイキャスト
- Y座標を各足の高さの平均＋αで置き換える

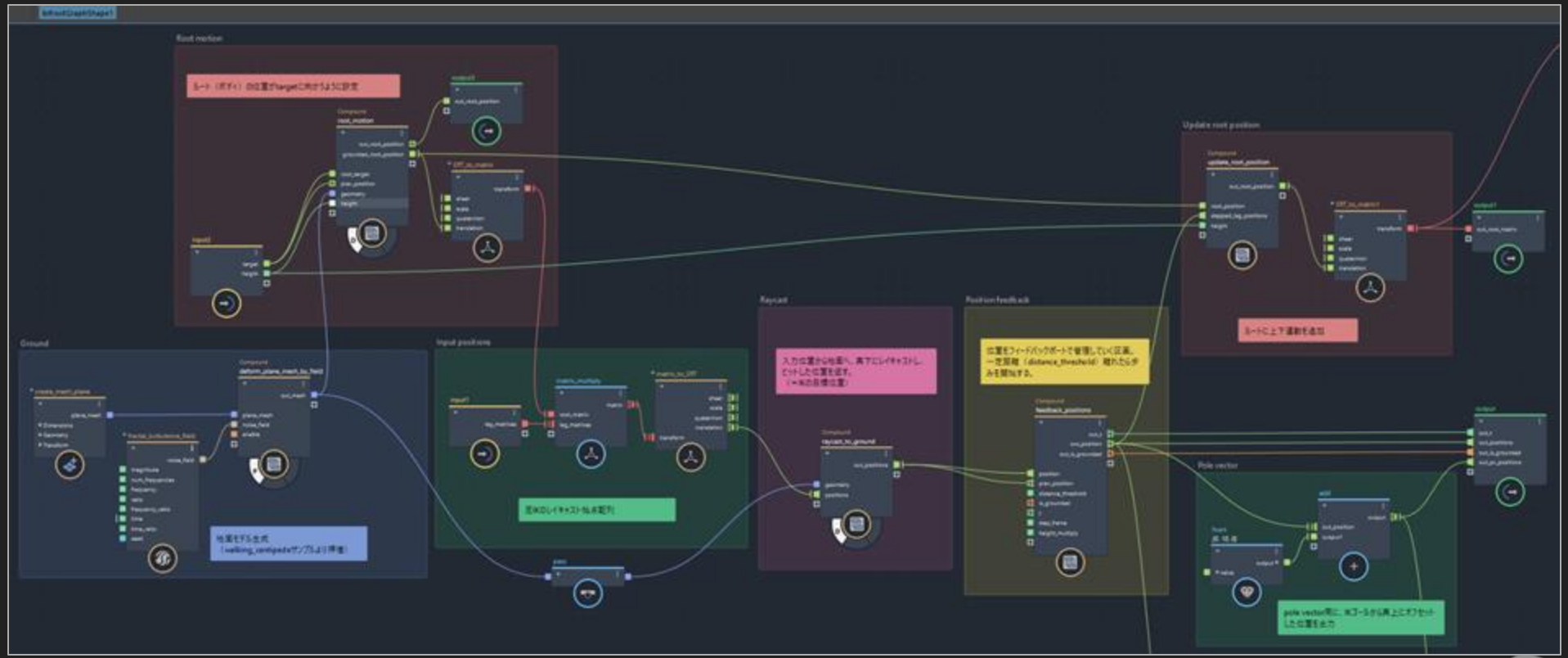




© 2024 Jet Studio Inc.

Step6

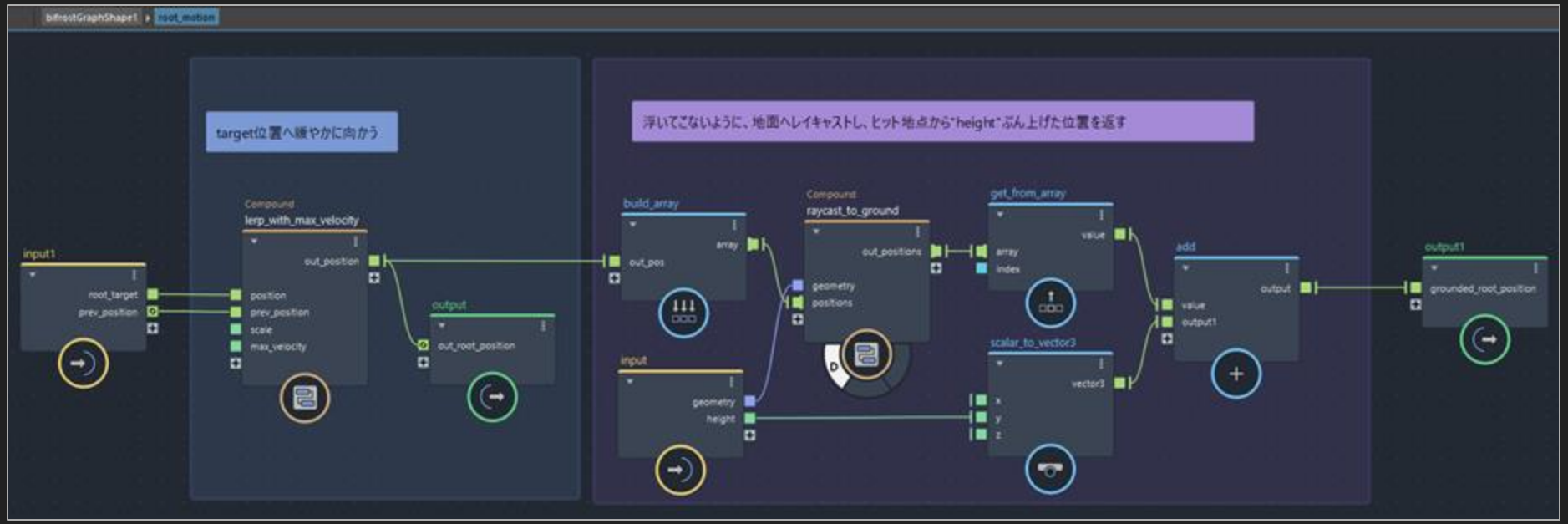
- **step06_root_position.ma**



© 2024 Jet Studio Inc.

Step6

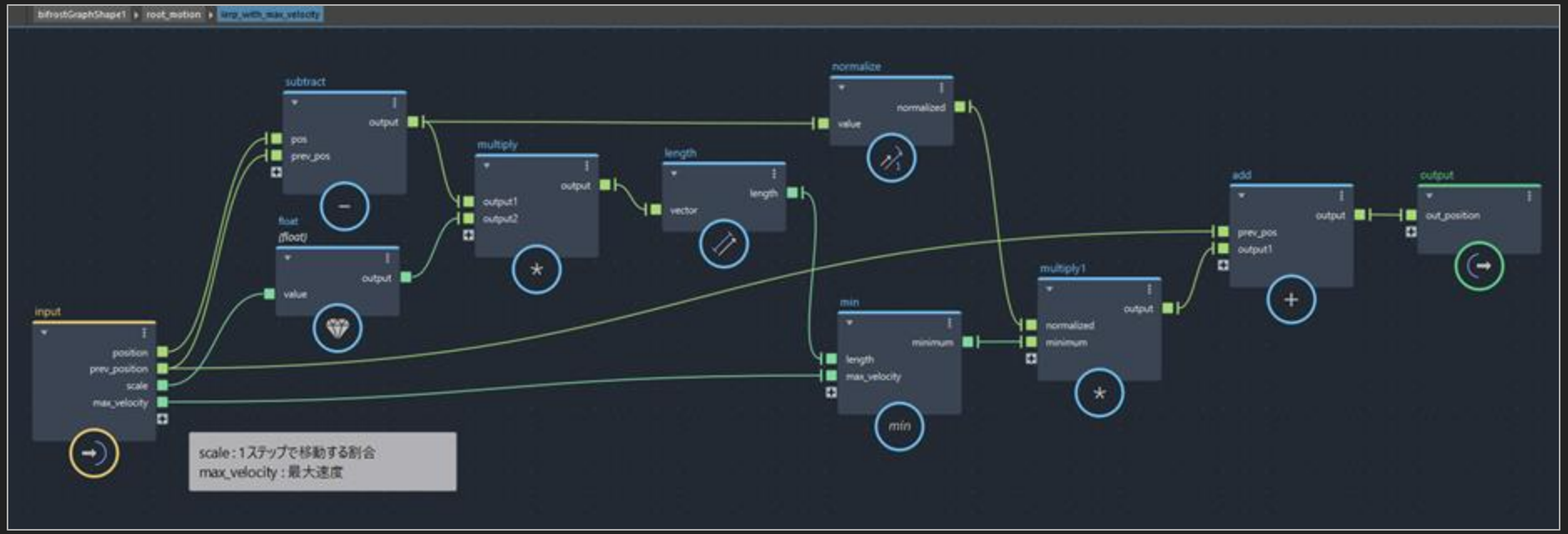
- **step06_root_position.ma**



© 2024 Jet Studio Inc.

Step6

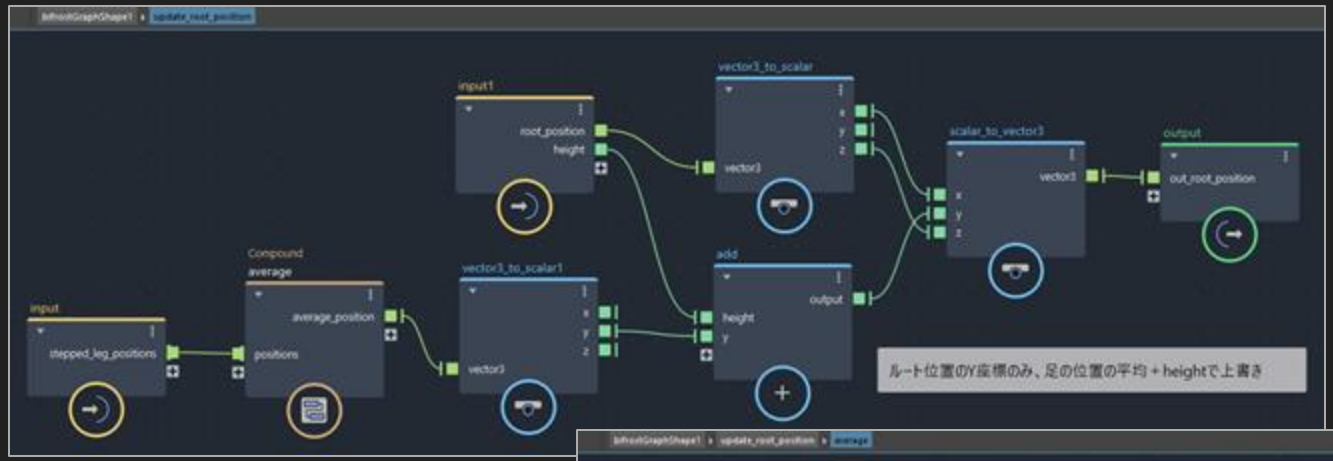
- **step06_root_position.ma**

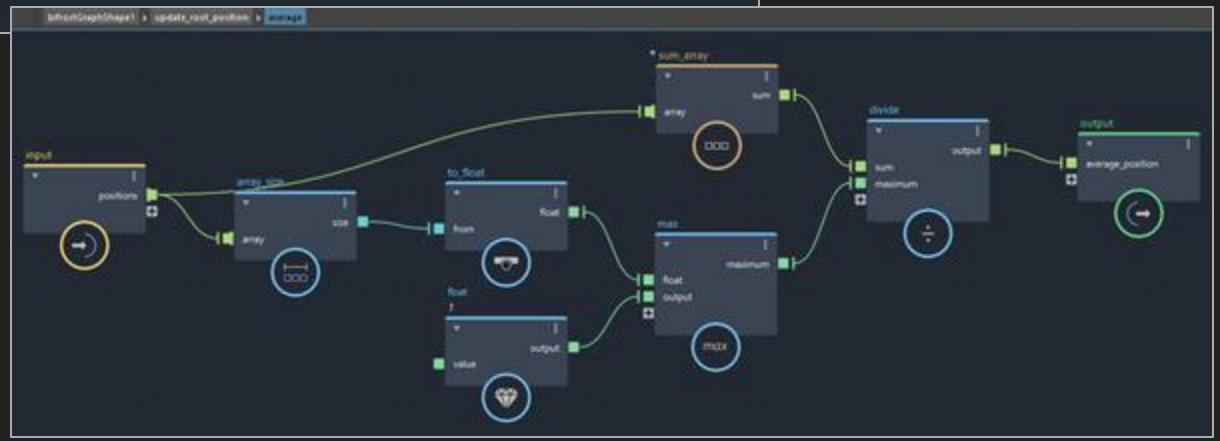


© 2024 Jet Studio Inc.

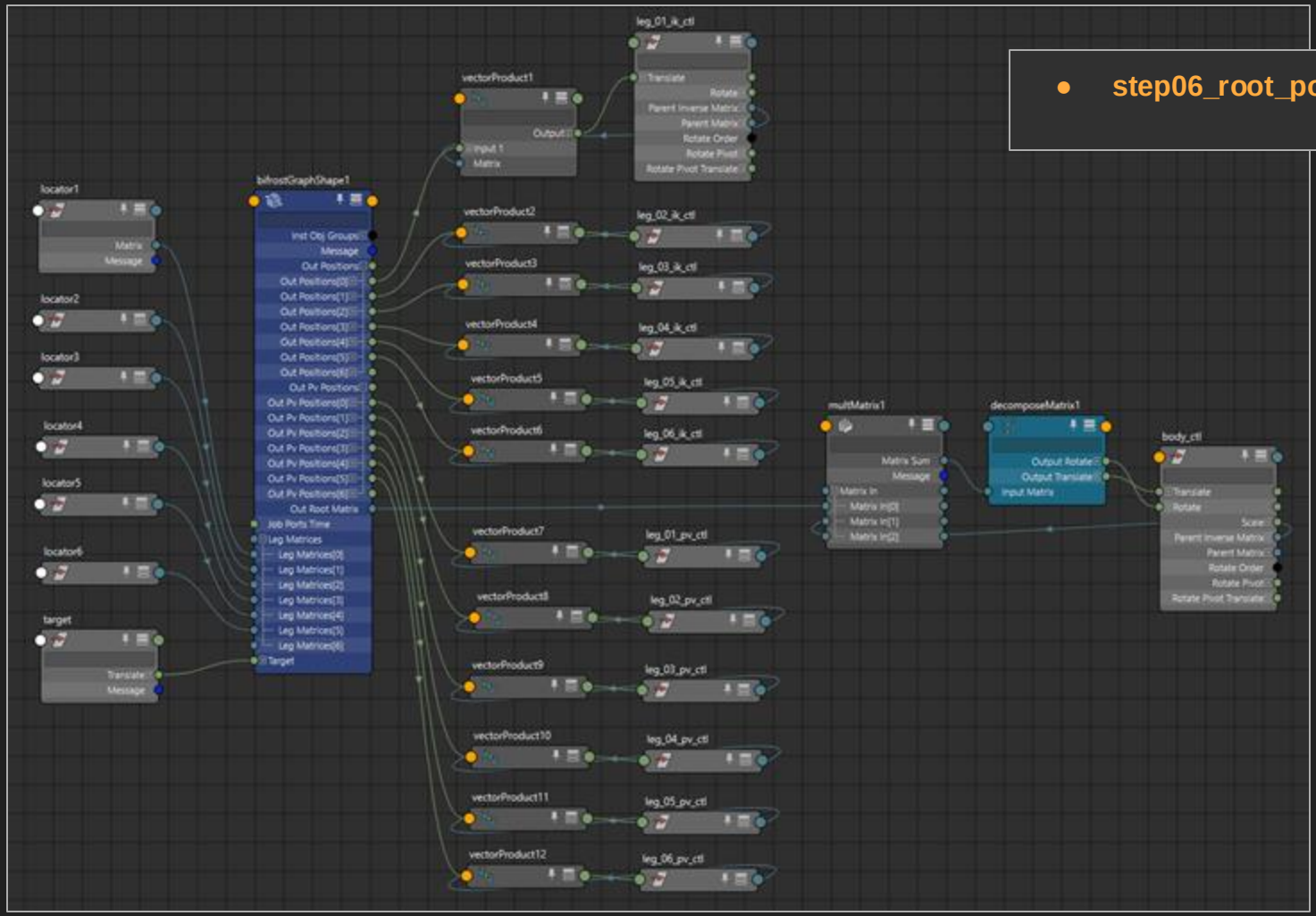
Step6

- **step06_root_position.ma**





© 2024 Jet Studio Inc.



- **step06_root_position.ma**

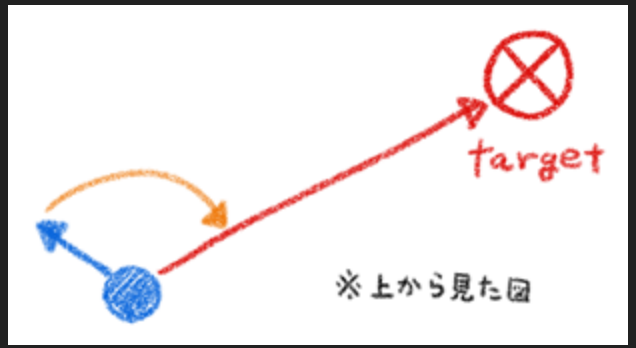
© 2024 Jet Studio Inc.

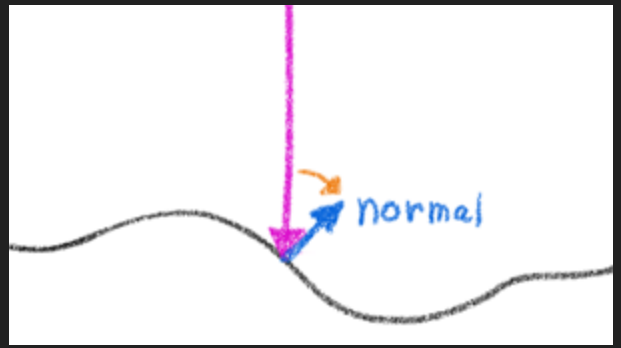
Step7

.¥samples
- sample_06_direction_interp.ma

ルートの方向転換も自動にする

- ターゲット位置に向かうベクトルと、現在のフロントベクトルを回転で補間する → ベクトルを回転に変換する
- レイキャストで地面の法線を取得し、傾きを回転に変換して上記に加える
- Step6の位置補間は方向ベクトルがある程度ターゲット方向に向いてから開始するように変更

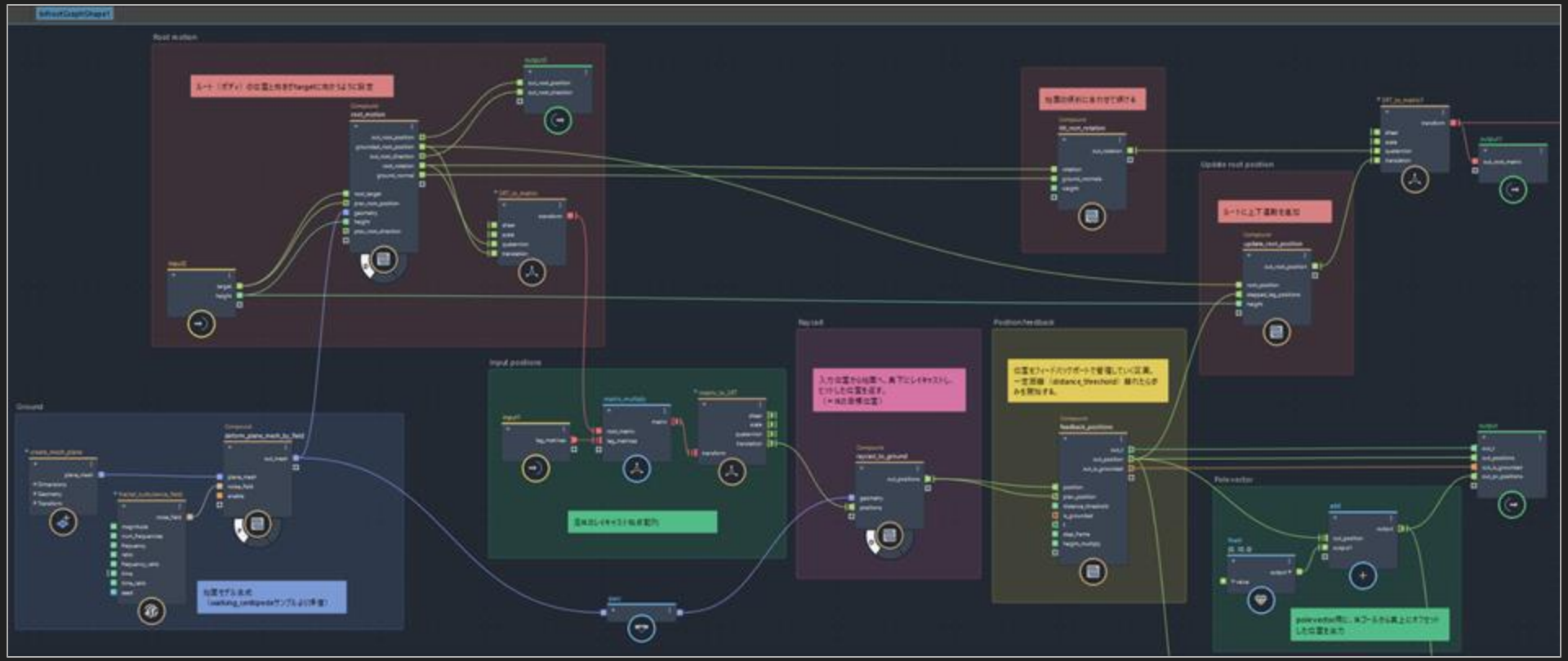




© 2024 Jet Studio Inc.

Step7

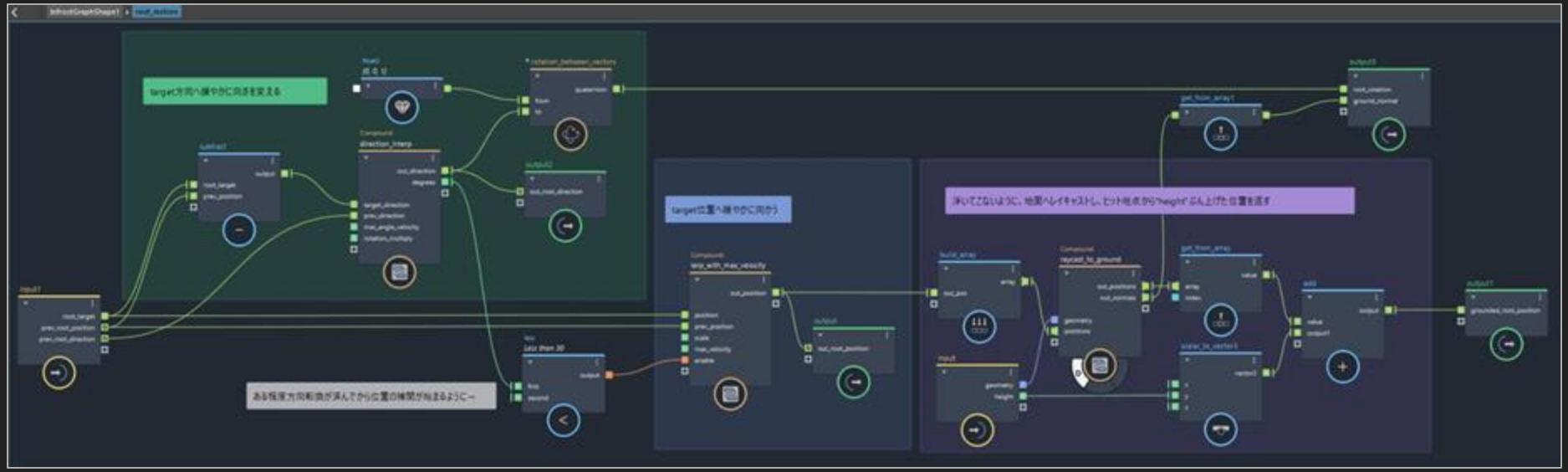
- **step07_root_rotation.ma**



© 2024 Jet Studio Inc.

Step7

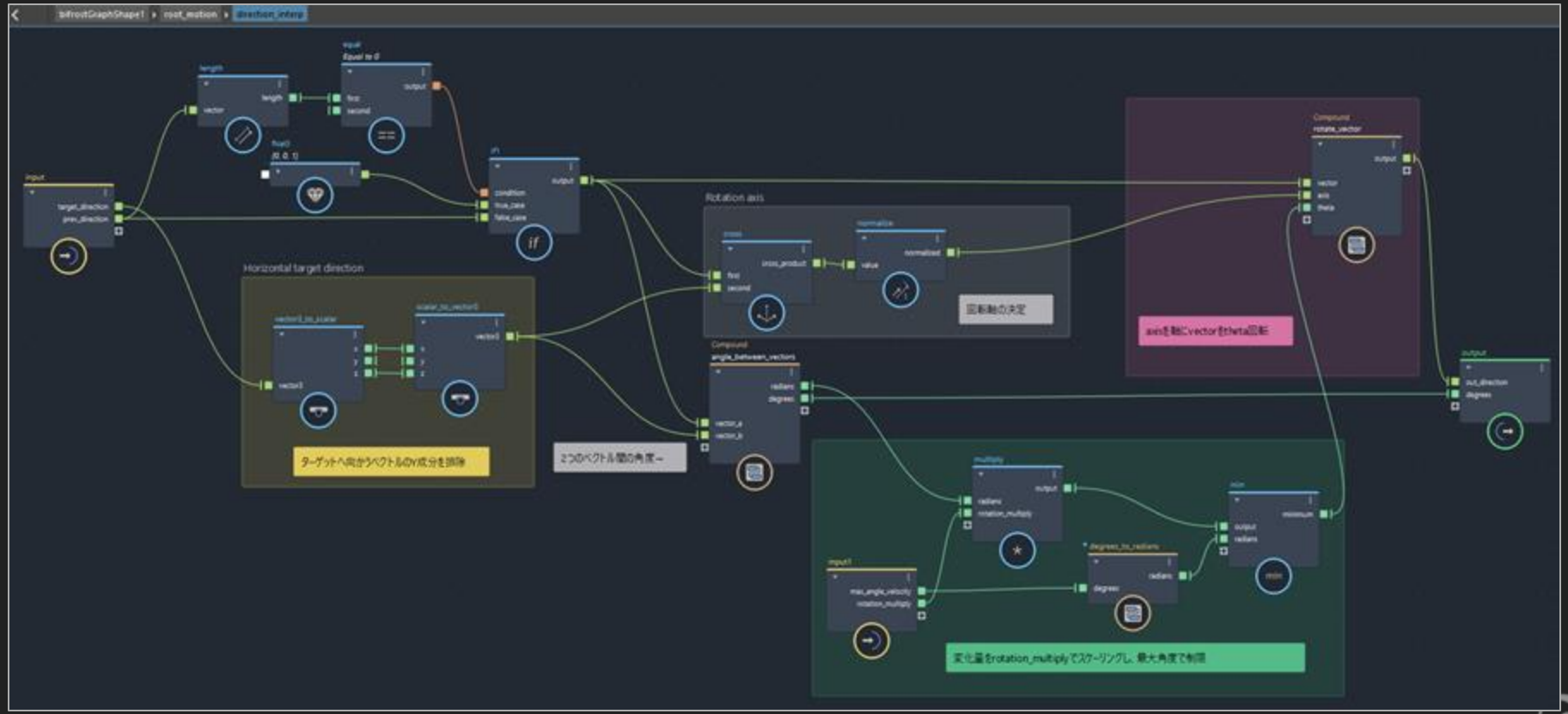
- **step07_root_rotation.ma**



© 2024 Jet Studio Inc.

Step7

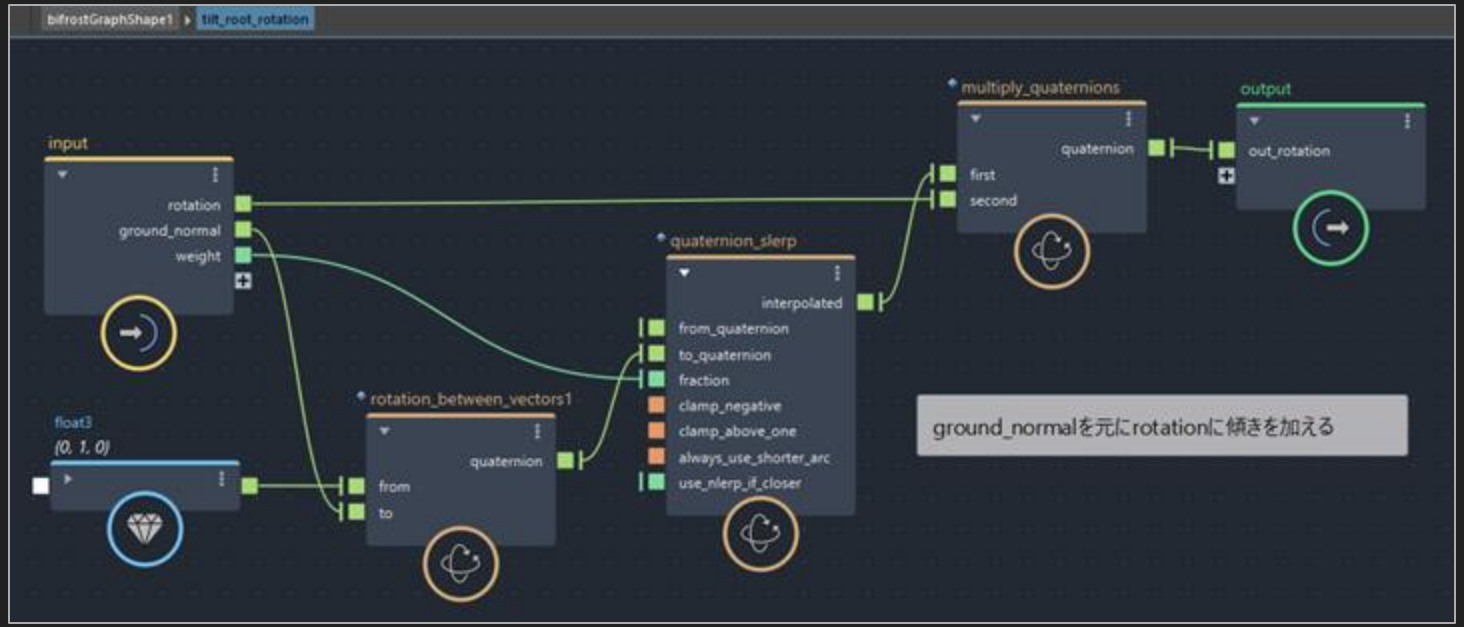
- **step07_root_rotation.ma**



© 2024 Jet Studio Inc.

Step7

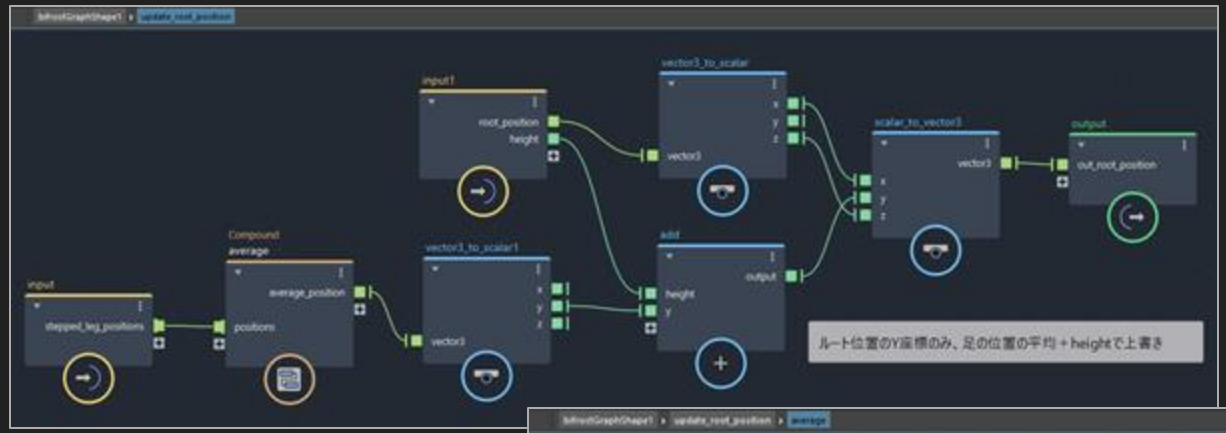
- **step07_root_rotation.ma**

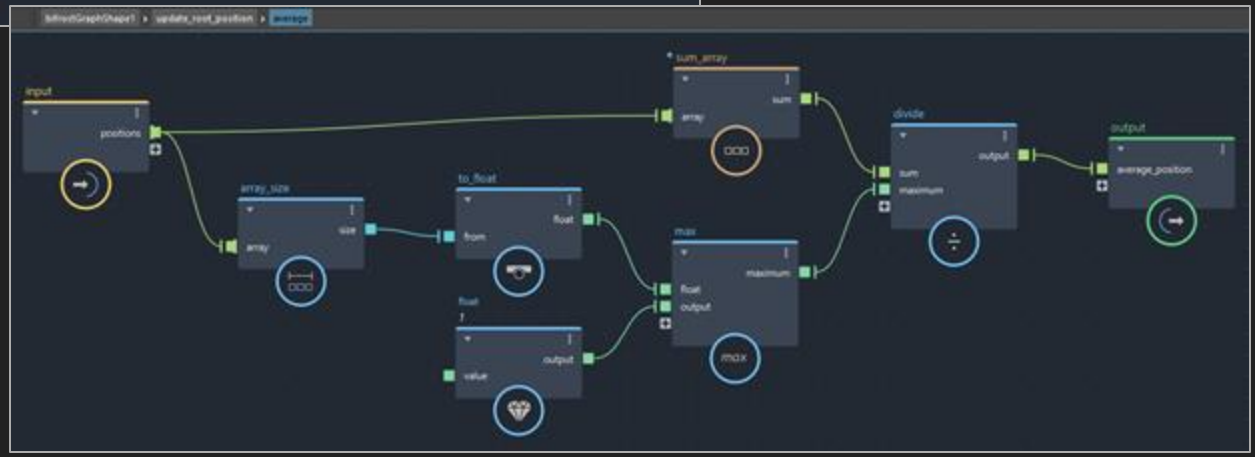


© 2024 Jet Studio Inc.

Step7

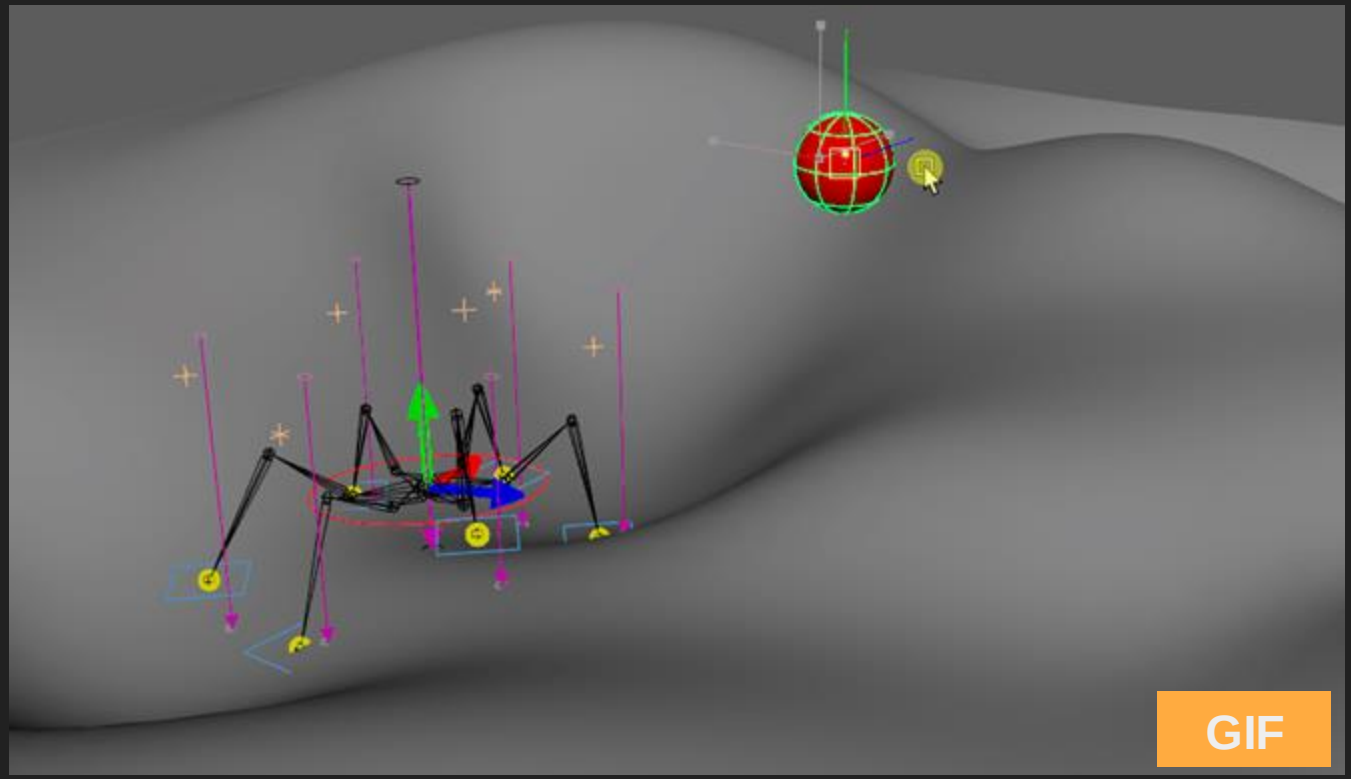
- **step07_root_rotation.ma**





© 2024 Jet Studio Inc.

完成！



© 2024 Jet Studio Inc.

まとめ（総評）

© 2024 Jet Studio Inc.

Bifrostはビジュアルプログラミング環境！

- ベクトルや行列などの演算ノードが豊富
- “繰り返し処理”が組める
  - iterate, 自動ループ
- 評価結果をキャッシュできる
  - feedback

Maya標準ノードでは出来なかったことが手軽にできる！

- 細かい機能レベルから自作できる
  - ソルバ, デフォーマ, シミュレーション, プロシージャルアニメーション, etc
- ロジックさえ組めれば、プログラミングやビルド環境構築の知識は不要!?

© 2024 Jet Studio Inc.

ご清聴ありがとうございました

© 2024 Jet Studio Inc.