ダイナミック版ユニティちゃん
（unitychan_dynamic）
マニュアル

Unity4.3.4以上対応

2014/07/23

ダイ ミ

ン版エ

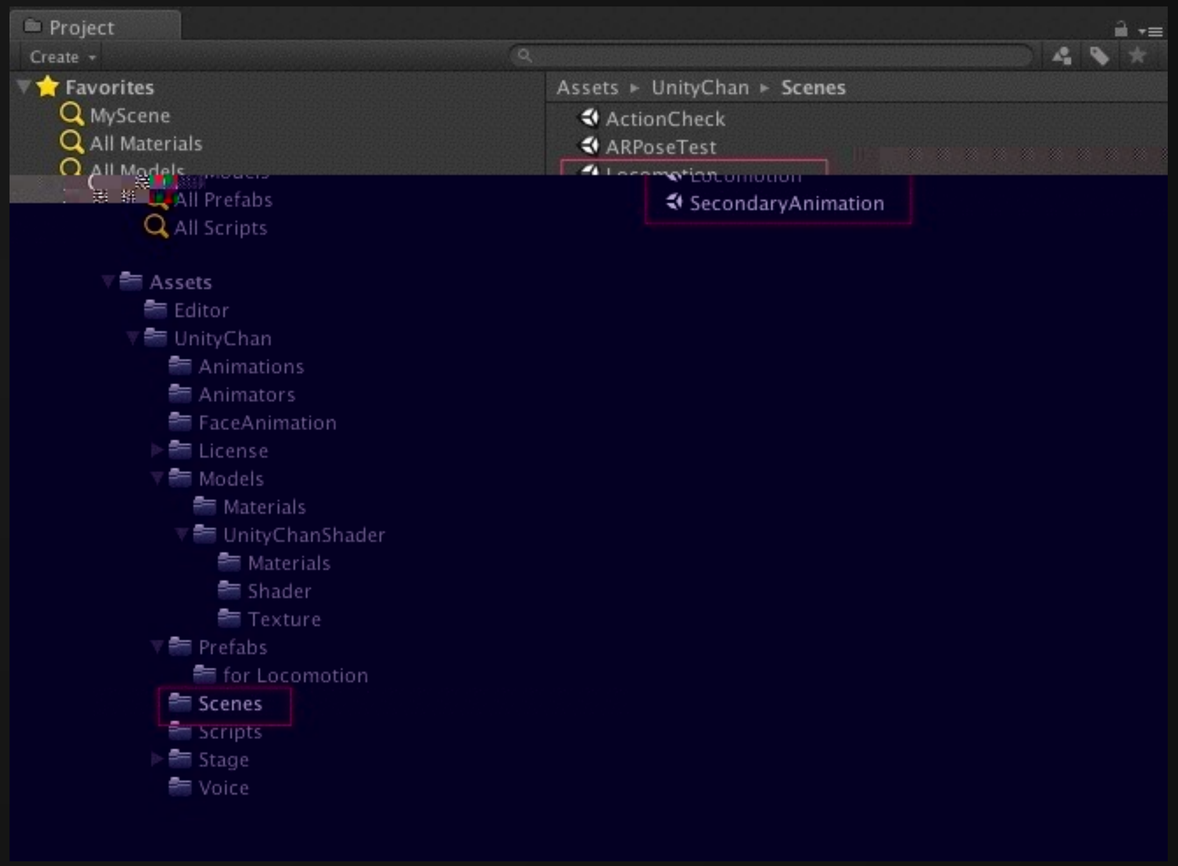
等の

通

ポ

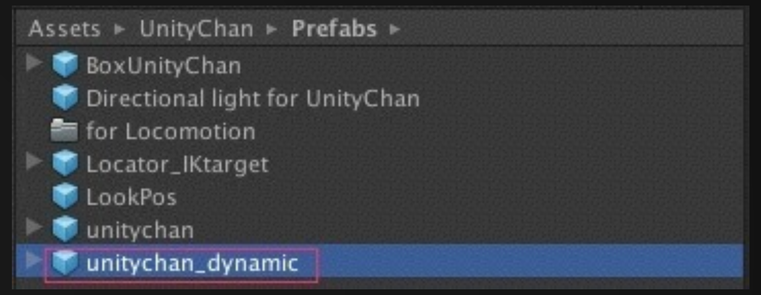
サンプルシーン
SecondaryAnimation
Locomotion

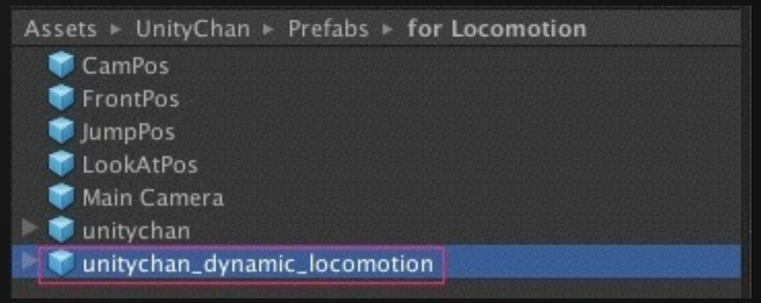
- Assets>UnityChan>Scenes中の「SecondaryAnimation」と「Locomotion」がサンプルシーンになります。
- **SecondaryAnimation**は、ポージングやダンスモーションなど、ユニティちゃんが原点の付近であまり動かない場合のセッティング例です。見栄え優先で揺れ物を揺らすセッティングになっています。
- **Locomotion**は、ユニティちゃんをPlayerControllerなどで動かす場合のセッティング例です。揺れ物のセッティングが少し固めになっています。
- 各シーンを開くことで、すぐにダイナミック設定のついたユニティちゃんで遊ぶことができます。



プレファブ
unitychan_dynamic
unitychan_dynamic_locomotion

- Prefabsフォルダ内には、サンプルシーンで使用されているダイナミック版ユニティちゃんのプレファブが入っています。
- Assets>UnityChan>Prefabs
  unitychan_dynamic
  ！SecondaryAnimationシーン用プレファブ
- Assets>UnityChan>Prefabs>for Locomotion
  unitychan_dynamic_locomotion
  ！Locomotionシーン用プレファブ
- これらのプレファブを自分の好きなシーンに配置することで、各ダイナミック用設定を済ませたユニティちゃんをすぐに使うことができるようになります。

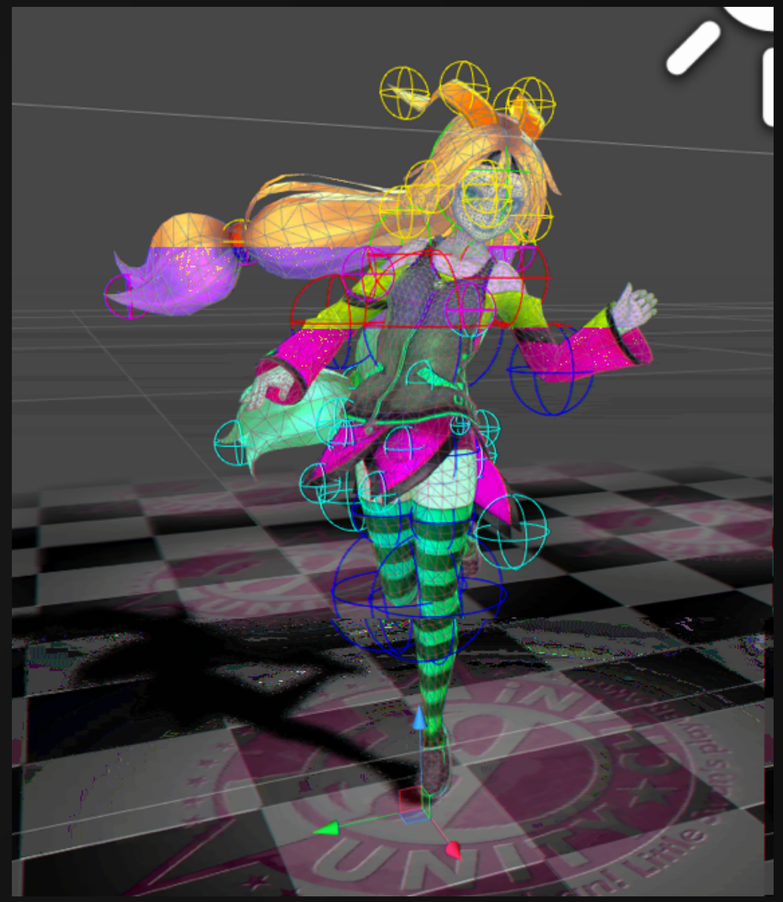




unity

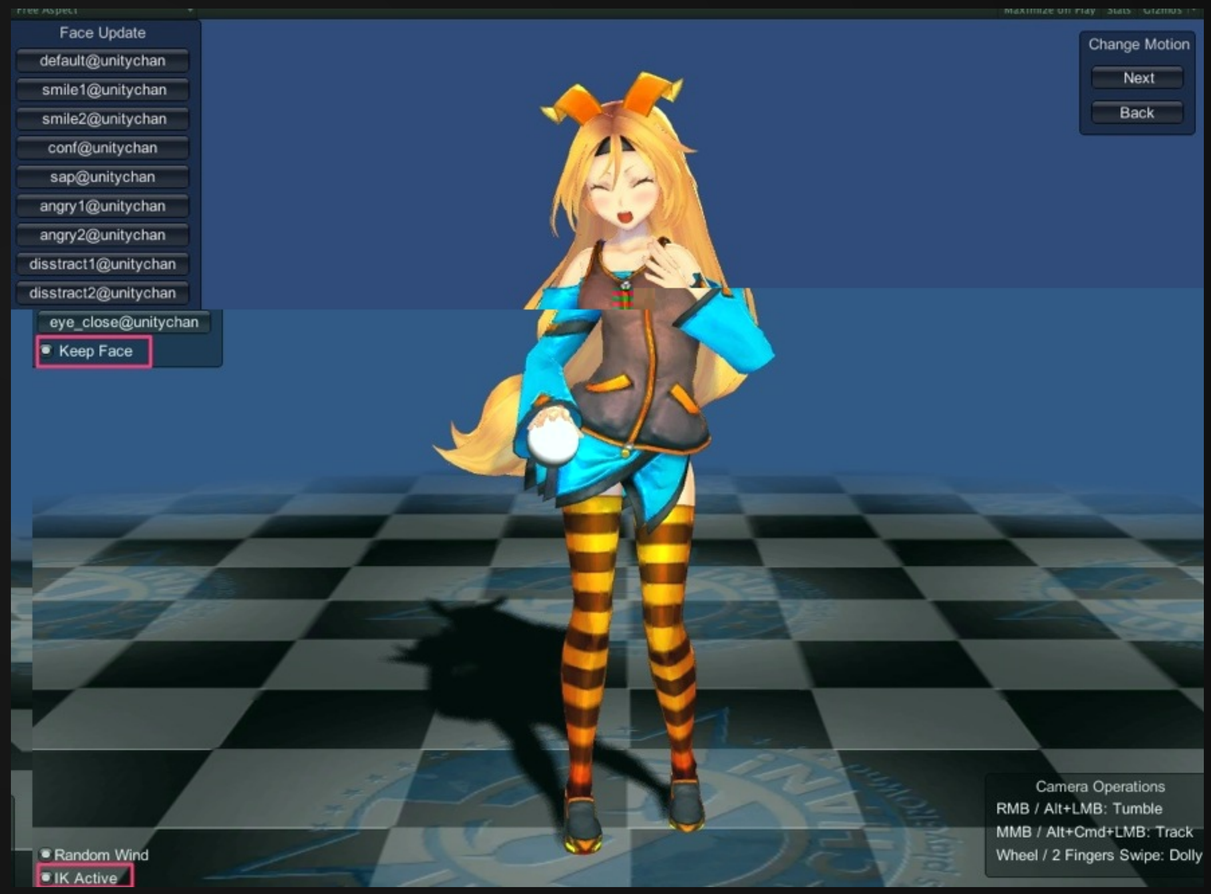
『ダイナミック版ユニティちゃん』の揺れ物制御用コンポーネントについて

- 『ダイナミック版ユニティちゃん』の揺れ物制御用のコンポーネントは以下のような構成になっています。
- **SpringManager.cs**：SpringBoneを制御するマネージャー。ユニティちゃん本体にアタッチされる。
- **SpringBone.cs**：各揺れ物用ボーンにアタッチされるコンポーネント。揺れ物に設定されている各ボーン（ジョイント）にアタッチされる
- **SpringCollider.cs**：SpringBone専用の当たり判定用コライダー。当たり判定を付けたいジョイントやロケータにアタッチされる。
- **RandomWind.cs**：ランダムに風にそよぐような効果を生み出すコンポーネント。ユニティちゃん本体にアタッチされる。ゲームビュー内の「Random Wind」で機能をON/OFFできる。
- **CreateLocatorHere.cs**：SpringCollider設定支援用エディタ拡張ツール。Hierarchy上であるジョイントを指定して実行することで、そのジョイントの子の位置にロケータを生成できる。ロケータは親のジョイントからローカル移動をすることで、位置をオフセットすることができる。生成したロケータにはSpringCollider.csをアタッチする。



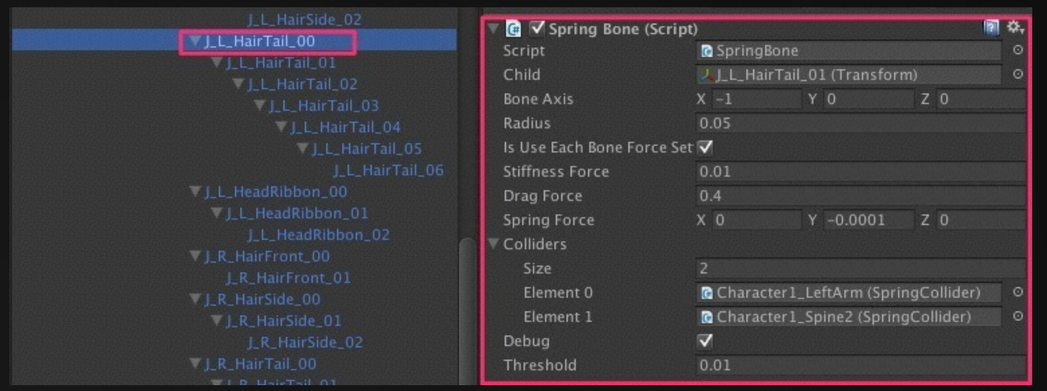
『ダイナミック版ユニティちゃん』に追加されたその他のコンポーネントについて

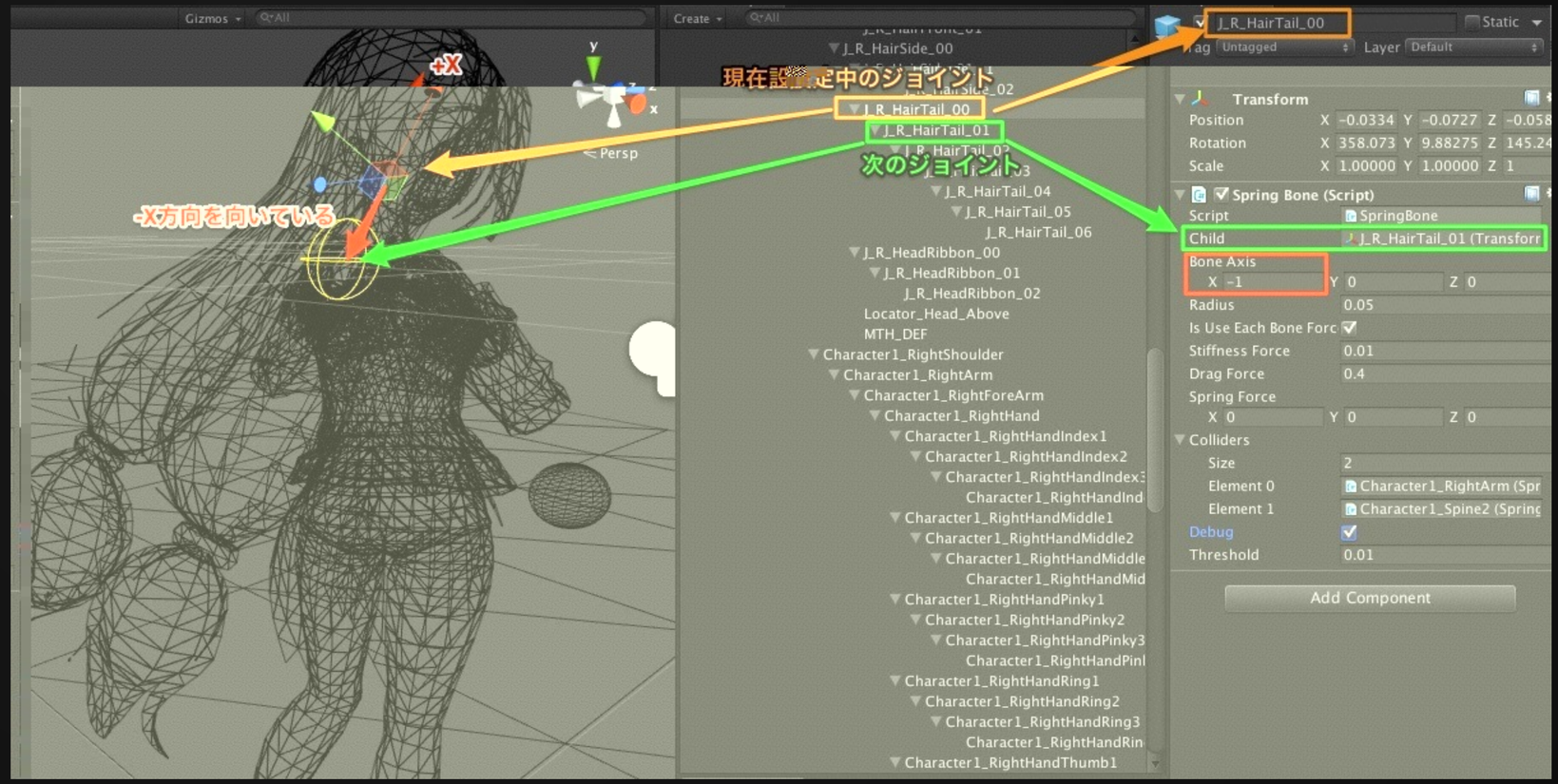
- **AutoBlink.cs**：自動で目パチアニメーションをさせるコンポーネント。
- **FaceUpdate.cs**：機能強化版。表情の固定ができるようになった（**KeepFace**をチェック後、表情スイッチを押す）他、Mecanimアニメーションイベントで、**OnCallFaceChange(“表情名”)**を呼び出すことで、各アニメーションデータ上で、タイムラインに沿って表情変化を設定できます。
- **IKCtrlRightHand.cs**：Pro Onlyの機能。右手のIKコントロールのサンプルです。コンポーネントを有効にした後、ゲームビュー内の「IK Active」でON/OFFできます。



SpringBone.csの設定①

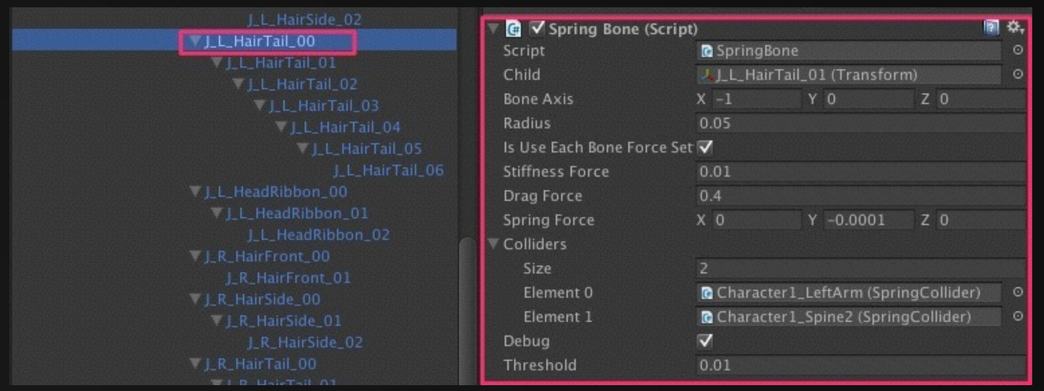
- **SpringBone**は、揺れ物用ダイナミクスの本体です。揺れ物用に入っているジョイント（ボーン）にアタッチすることで、連結バネ（Strand Spring）状に動かすことが可能です。
- 揺らしたいジョイントにSpringBone.csをアタッチします。**BoneAxis**にはジョイントの向いているローカル方向を入れ（図の例なら、-X方向なので、Xに-1を入れる）、**Child**には次のジョイント名を入れます。
- **Radius**は当たり判定を設定する時のSpringBone側の当たり判定用の球の半径です。
- Stiffness Force以降は、揺れ物ダイナミクスを制御するためのパラメーターです。おおよその初期値が入っています。そちらを参考に、望むような動きになるように、微調整をしていくとよいと思います。





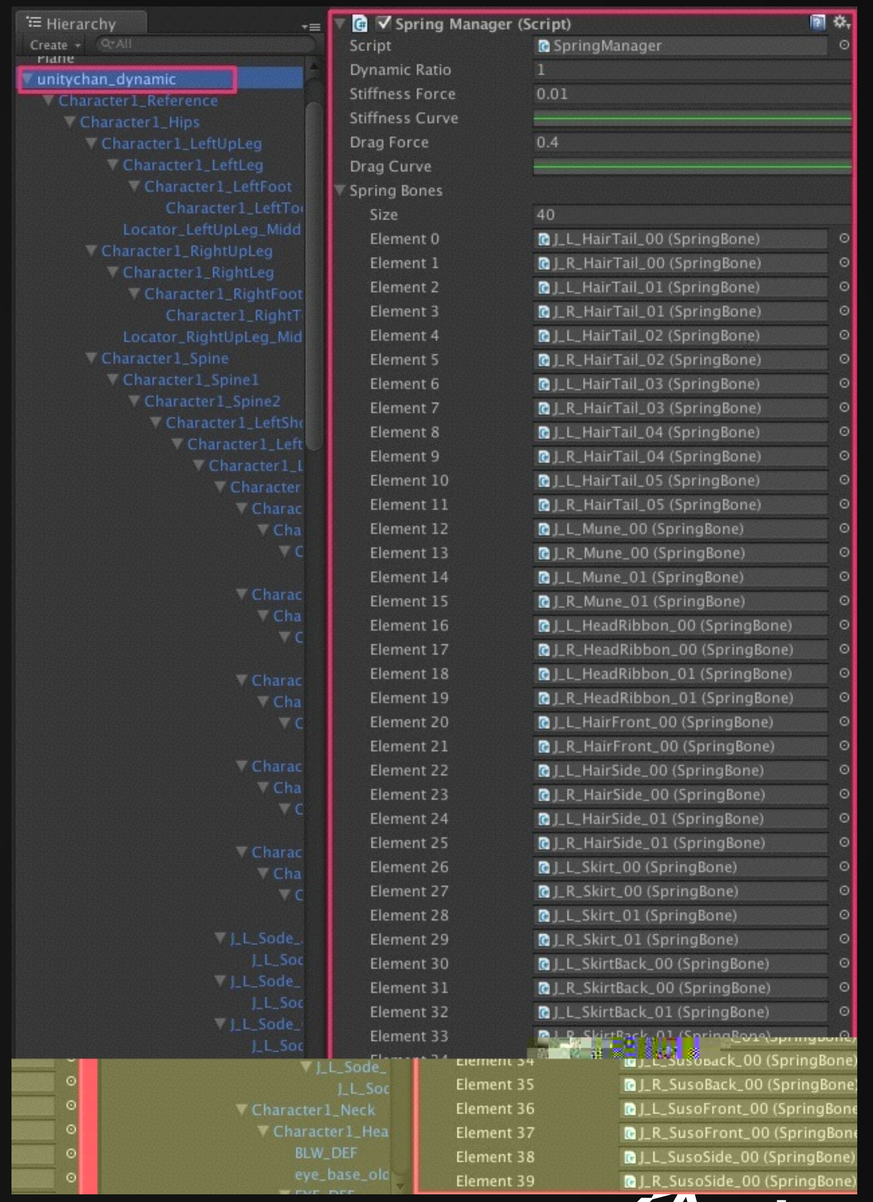
SpringBone.csの設定②

- **StiffnessForce**、**DragForce**はそれぞれ「**バネが戻るフォース**」「**フォースの減衰力**」です。後に説明しますが、**IsUseEachBoneForceSet**にチェックが入っている時には、SpringBone側に設定されているこれらのパラメターが使用されます。
- **Colliders**には、各SpringBoneコンポーネントと当たり判定を設定したいSpringColliderを設定します。
- 最後にThresholdは、このSpringBoneコンポーネントを制御しているSpringManager側のDynamicRatioが有効になる最低値を設定します。（この値を上げることで、一部のSpringBoneを個別に停止して、揺れ物ボーンに元々ついているアニメーションに切り替えることが可能になります。）
- SpringBoneの設定で重要なのは、①でのChild/BoneAxis/Radiusの設定と、②のCollidersの設定です。その他の値は、初期値を参考に微調整をするとよいでしょう。
- **SpringBoneの効き具合の調整**は、上の**StiffnessForce/DragForceの値で微調整をする**他に、後述する**SpringManager側のDynamicRatioで効果自体の効き具合を調整してしまう**という方法があります。大体の設定が終わった後は、DynamicRatio側で使う状況に合わせて調整するほうが便利なようです。



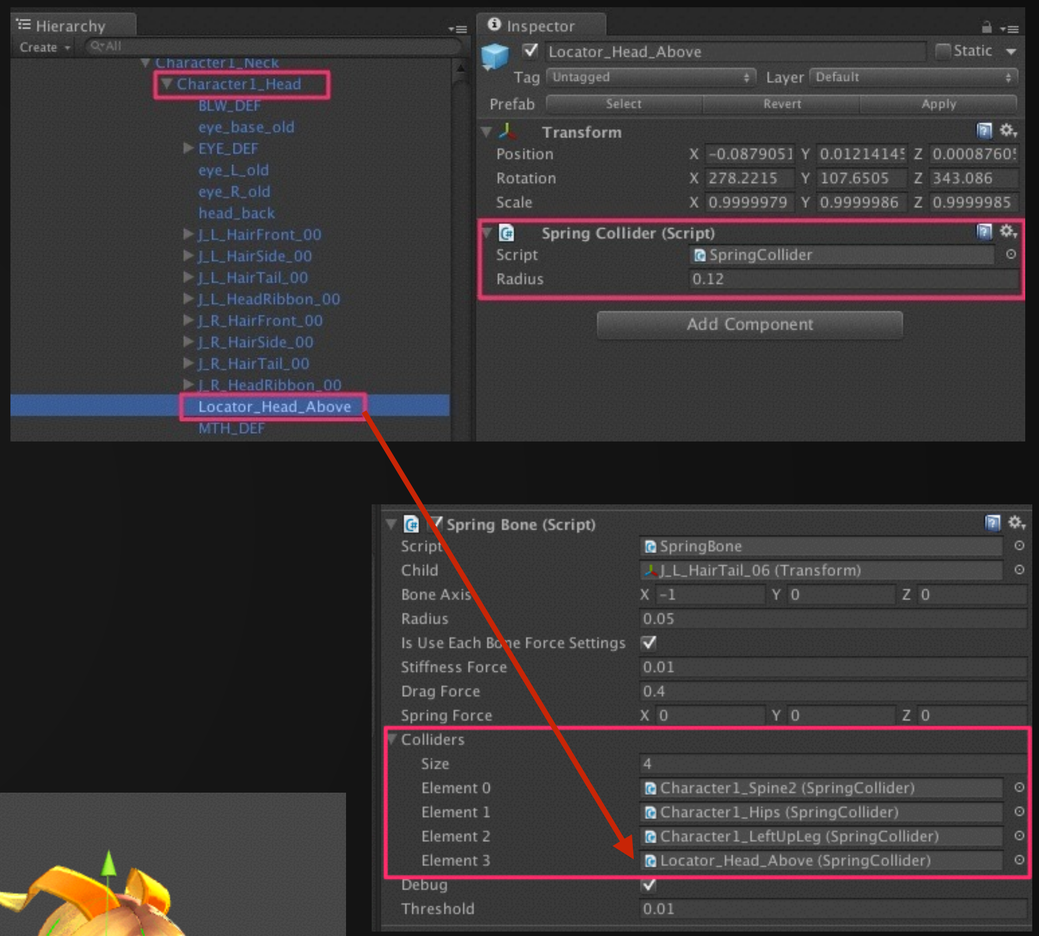
SpringManager.csの設定

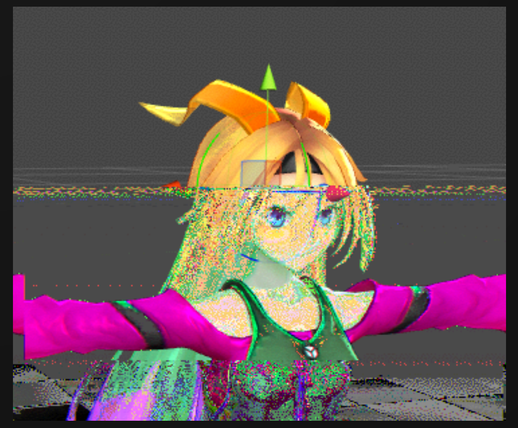
- **SpringManager**は、配下にあるSpringBoneを制御するコンポーネントです。
- **SpringBones**以下に管理させたいSpringBoneがアタッチされているジョイントを必ず**親から順に指定**します。
- **DynamicRatio**は、配下にあるSpringBoneの影響の強さを制御するパラメターです。**DynamicRatio=1の時、SpringBoneの影響が最大になり、DynamicRatio=0の時、SpringBoneの機能はキャンセルされます。**その場合、ジョイントが持っている元々のローカルローテーション値もしくは、揺れ物ボーンに付けられているアニメーションでのローテーション値が使用されます。DynamicRatioは、各SpringBoneに設定できるThresholdを参照できますので、各SpringBoneのThresholdを適宜設定し、DynamicRatioを調整することで、一部の揺れ物ボーンは、元々のアニメーションで、一部の揺れ物ボーンはSpringBoneで動かすというようなことも可能です。
- SpringManager側にあるStiffnessForceとDragForceは、配下にあるSpringBoneの同名の値の最適値を探るために使えます。有効にする場合には、各SpringBoneのIsUseEachBoneForceSetがオフになっている必要があります。カーブなどを利用して様々な設定のパターンを探ることができるようになっています。**SpringManager上でシミュレーションした値は、エディタ上でしか使えません**ので、よい値が見つかったらSpringBone側に値をコピーし、IsUseEachBoneForceSetをチェックすることをお薦めします。
- なおSpringManagerは、各揺れ物ボーンのチェインの一番上の階層に付けることで、各揺れ物ボーンのチェインを個別に管理することも可能です。



SpringCollider.csの設定

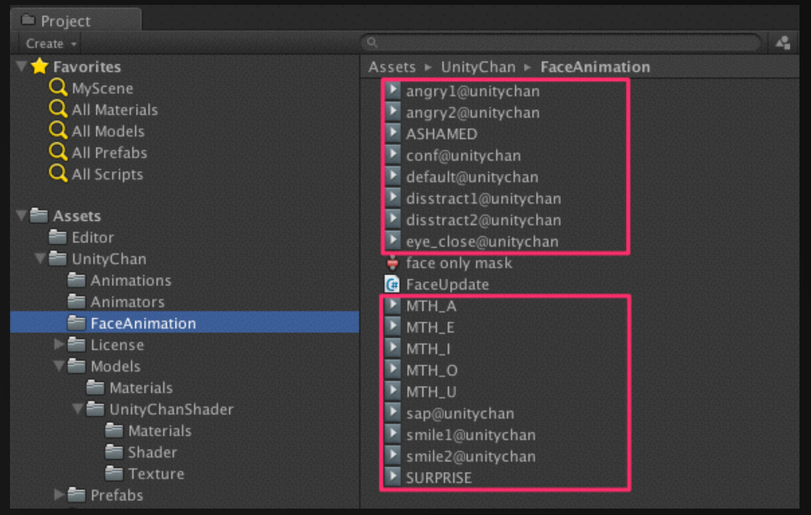
- **SpringCollider**は、SpringBoneコンポーネントと衝突判定を行うためのコンポーネントです。ジョイントやジョイントからオフセットした位置に設定したロケータにアタッチすることで、当たり判定用のスフィアコライダーとして利用することができます。**Radius**が当たり判定用のスフィアコライダーの半径となります。
- 当たり判定をとりたい**SpringBoneコンポーネント側の配列（Colliders）**に、判定したいSpringColliderをアタッチしたオブジェクト名を指定します。
- エディタ拡張コマンド**CreateLocatorHere.cs**を使うことで、Hierarchy上でジョイントの子の位置にロケータを生成できます。**ロケータは親のジョイントからローカル移動をすることで、位置をオフセットすることができます**ので、オフセットした位置でSpringColliderを複数設定することで、複雑な当たり判定領域を設定したり、ジョイントから離れている位置に当たり判定領域を設定することができます。CreateLocatorHereは、Hierarchy上でジョイントを選んだ後、GameObjectメニューからコマンド選択する他、Transformコンポーネント上で右クリックメニューから呼び出すことで利用できます。
- SpringColliderはUnity標準の物理エンジンによる当たり判定を使用していませんので、ご注意ください。



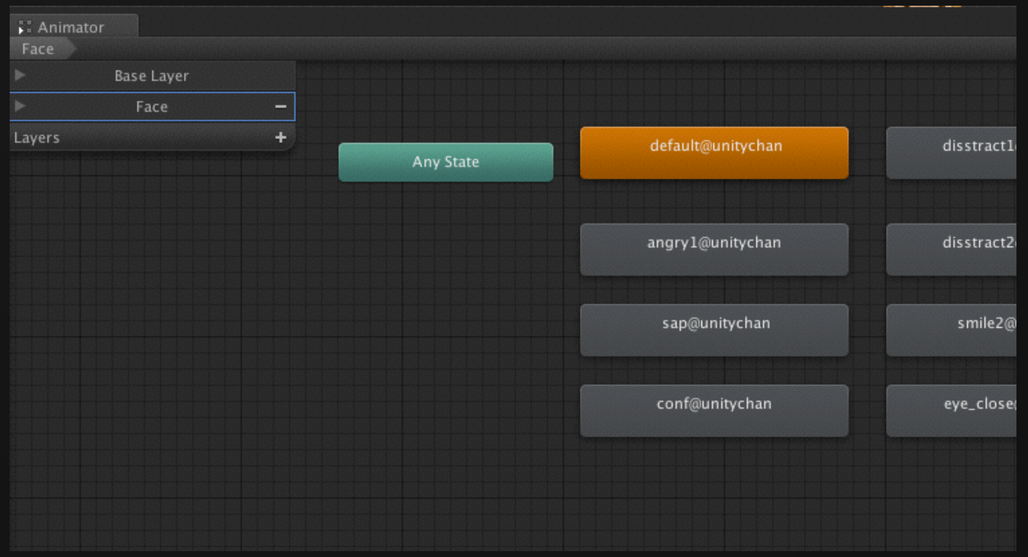


FaceUpdate.cs / 新規表情の追加

- FaceUpdateコンポーネントに新規に表情を追加するには、以下の手順で行います。
- 表情のブレンドシェイプパラメタをキー登録したanimationデータを新規作成する
- 新規に作成したanimationデータを、使用中のAnimatorの第２レイヤー「Face」に登録する
- FaceUpdateコンポーネントに、新規に登録したanimationデータを登録する。

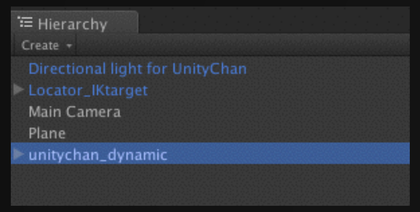


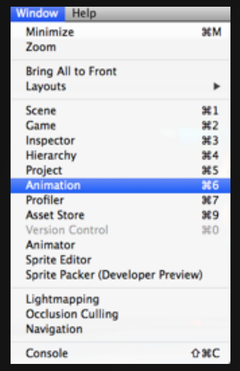
表情用animationデータ



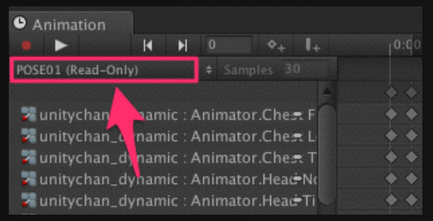
表情アニメーションデータの新規作成

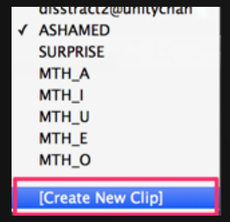
- シーンに置かれたユニティちゃんをHierarchyから選択し、Menu>Windows>Animationを選択してアニメーションウィンドウを開く。

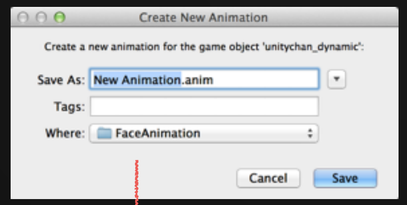




- アニメーションウィンドウの「POSE01」タブをクリックして、一番下にある[Create New Clip]を選択し、FaceAnimationフォルダ内に新規に表情用アニメーションファイルを作成する。

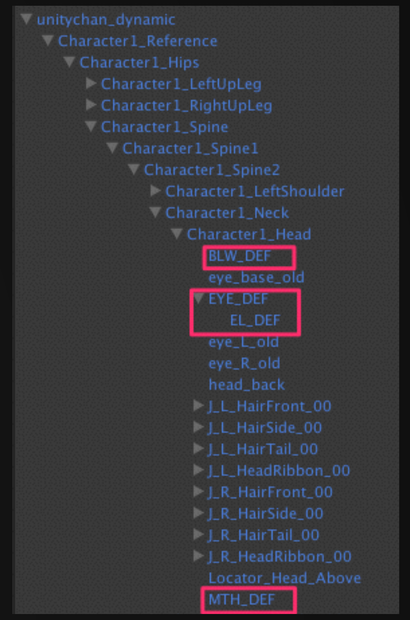




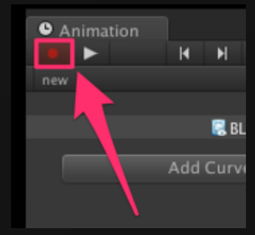


FaceAnimationフォルダ内に作成すること

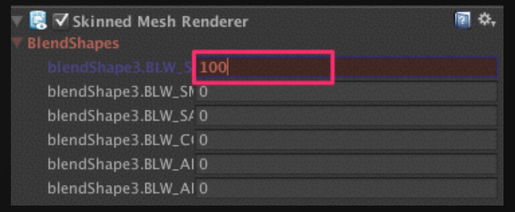
- ユニティちゃんの表情を作るブレンドシェイプは、Character1_Head 直下のBLW_DEF/EYE_DEF/EL_DEF/MTH_DEFのSkinned Mesh Rendererコンポーネント内にあります。こちらの各パラメータを0〜100の範囲で調整して様々な表情を設定します。



- アニメーションウィンドウよりアニメーションモードボタンをクリックし、キー記録を開始。各ブレンドシェイプに数値を入れます。Skinned Mesh Rendererの各ブレンドシェイプの数値がピンク色になればキーが記録されています。最後にアニメーションモードボタンを元に戻し、キー記録を終了します。

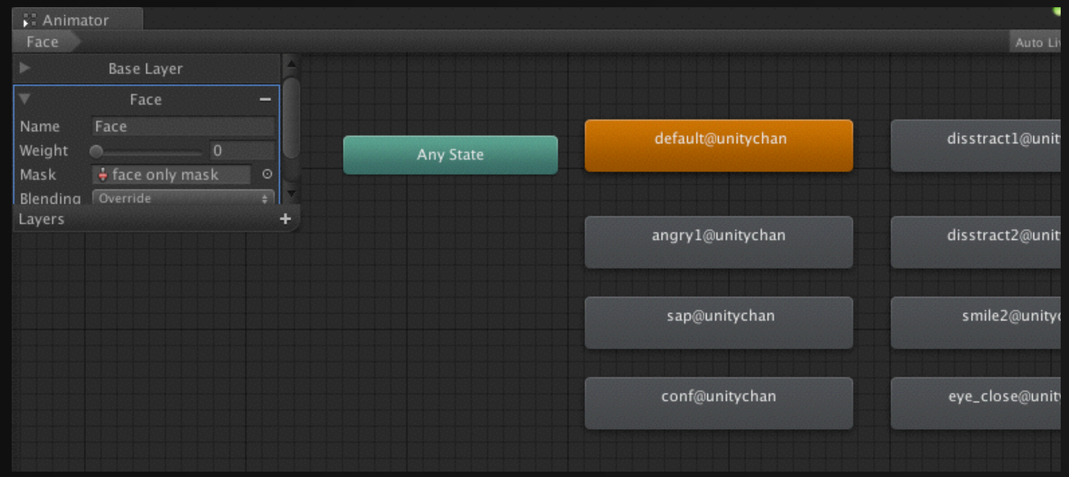


アニメーションモードボタン



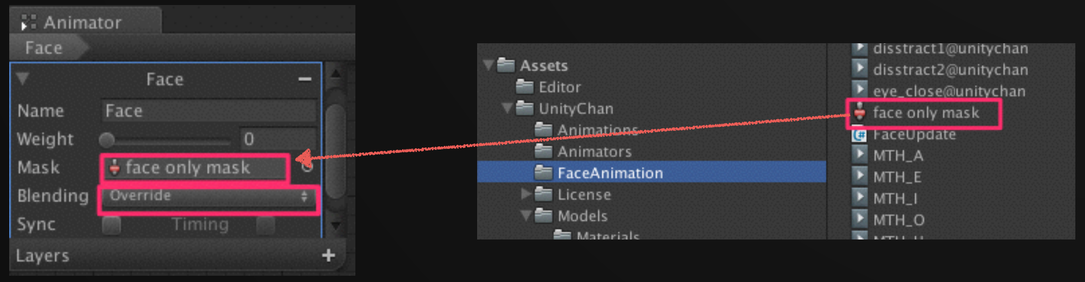
表情アニメーションデータの登録

- 使用しているAnimatorの第２レイヤーに「Face」がある場合には切り替え、ない場合には新規に作成します。先ほど新規に作成した表情アニメーションデータをFaceレイヤーにD&Dして登録します。



- 新規にレイヤーを追加した場合には、レイヤーの設定は以下のようにします。

Mask！「face only mask」、Blending！「Override」



- シーン中のユニティちゃんを選択し、FaceUpdateコンポーネントを開きます。Animations内にあるSizeの数値を新規に追加する表情数ぶん増加し、新規に作成されたElementスロットに追加する表情アニメーションデータを登録します。