AnimFollow

What is it?

Play the web player and judge for yourself before you buy. The web player is exactly the example scene.

Also consider the Physically Based Character controller, which is a more advanced asset, currently available in Alpha (Not released on the asset store, contact me for purchase. Updates are off course included). The PBC is where the development is focused now. AnimFollow licence holders buy the PBC at half price. Se a demo here:

https://www.youtube.com/watch?v=0EHnvzcffbQ&list=PLEkyNrA4JVcsCyTXrY6PFFaK_gKUe9dAr

Read the PBC manual here: http://www.kavorka-racing.com/uploads/1/4/6/9/1469198/pbc_manual.pdf

AnimFollow is the name of one script that uses physics to make a ragdoll mimic an animated character.

The AnimFollow script offers very flexible ways of tweaking the manner in which it tries to follow the animation. You may, per limb, in real-time, set the strength with which the ragdoll tries to follow:

The world rotations of each animated limb.

The local rotations of each animated limb.

The world position of each animated limb.

The Foot IK

The AnimFollow asset now also includes a foot IK that alters the animations so that the feet movements is performed in relation to the ground even if the ground is not flat.

The RagdollControl and the example scene

The RagdollControl script is an example of how to utilize the flexibility of the AnimFollow script. RagdollControl scripts can be made infinitely complex and do wonderful things. The included RagdollControl was initially intended as a starting point and was a such not super polished. Due to the popularity of the included RagdollControl (few feel the urge to make their own) it has received a slightly better polish for version 4. This is still not a complete project, AI or character controller.

Play the web player and judge for yourself before you buy. The web player is exactly the example scene.

Feel free to sell your custom RagdollControl utilizing AnimFollow. But Kavorka Games owns the rights to the AnimFollow script. You may not sell the AnimFollow script or any modified version of it.

Setup

See also appendix and tutorials on kavorka-racing.com.

You should have a moving character (the master) before you start setting up the AnimFollow. The animator must include the getup animations that transitions from any state to idle. The RagdollControl script talks to the PlayerMovement script. If you make your own PlayerMovement script make sure it works together with the ragdollControl.

- Make a ragdoll from a copy of your model. The transform hierarchy must exactly match your model. Put the ragdoll on an ignored layer (e.g. the Water layer).
- This version of AnimFollow is set up for a ragdoll with the Ethan hierarchy and made with the Unity whizzard. But with an extra collider in the spine and the excludeTransforms in the AnimFollow script as in the example scene.
- Drag and drop the ReplaceJoints script on the ragdoll. The script will replace all character joints with configurable joints. If you have a ragdoll with more than twelve rigidbodies, drag and drop the ResizeProfiles[1] script onto the ragdoll.
- Place the ragdoll in the same position and rotation as the master.
- Add the RagdollControl and the AnimFollow script to the ragdoll. Assign the root transform of your master to the AnimFollow script in the inspector.
- Add the SimpleFootIK script to the master and assign the ragdoll.
- Add the RagdollHitByBullet script to the ragdoll.root.
- Add the ShootStuff script to a gun or camera.

In unity 5 you may have to lower PTorque, PForce, Dtorque, Dforce a little or fiddle with torqueProfile and forceProfile if the character is shaking.
In Unity 5 You should also open the HashIDs script and uncomment the commented lines.

Study how the example scene_AF_SFIK is set up. Try to not make anything very different than the example scene in the beginning.

RagdollControl

AnimFollow must be strong to closely match fast paced animations. AnimFollow is to strong to be realistically affected by physics. Therefore the strength must be adjusted to the context. RagdollControl is a script that adjusts the strength during: **colliding, falling, matching the masters pose and getting back up**. You may use any mecanim state machine, but it must include the getup animations that transitions from any state to idle. The getup animations must be named, and set up, like in the example project.

AnimFollow

AnimFollow comes set up for the EthanRagdoll_12AF in the example scene. The EthanRagdoll_12AF ragdoll is a ragdoll created with the Unity ragdoll wizard but there is an extra joint/rigidbody/collider added to one of the spine transforms.

The main parameters of AnimFollow is **PTorque**, **PForce** and **Dtorque**, **DForce**. P-parameters are the proportional term of the PD-controller. Consider them the strength of the ragdoll´s muscles. If set to high the system becomes unstable. The D-parameters are the derivative term of the PD-controller. Expect that a fair amount of tuning may be required. Apart from the above parameters that are for all limbs, the **torqueProfile** and **forceProfile** sets the strength per limb. Use stronger forces near the centre of mass.

You may also use maxTorque, maxForce and maxJointTorque to clamp the strength. RagdollContol clamps the strength and leaves the P-parameters unaltered.

If your ragdoll has more than twelve rigid bodies the size of all per-limb-arrays (ending with profile in the editor) must match the number of rigid bodies. A working torqueProfile for an Ethan

1 You must change a value in one of the profiles to make the size change permanent. (I do not set dirty automatically)

hierarchy are:
Hips = 20, left thigh = 30, left calf = 10, right thigh = 30, right calf = 10, spine1 = 30, spine2 = 30, Head = 30, left biceps = 30, left fore arm = 10, right biceps = 30 and right forearm = 10.
Those are good starting points. To see the order of your transforms, press play and look at the slaveRigidTransforms (in AnimFollow) in the inspector.

Support

Feel free to ask question in the forum or email: kavorka.games@gmail.com.

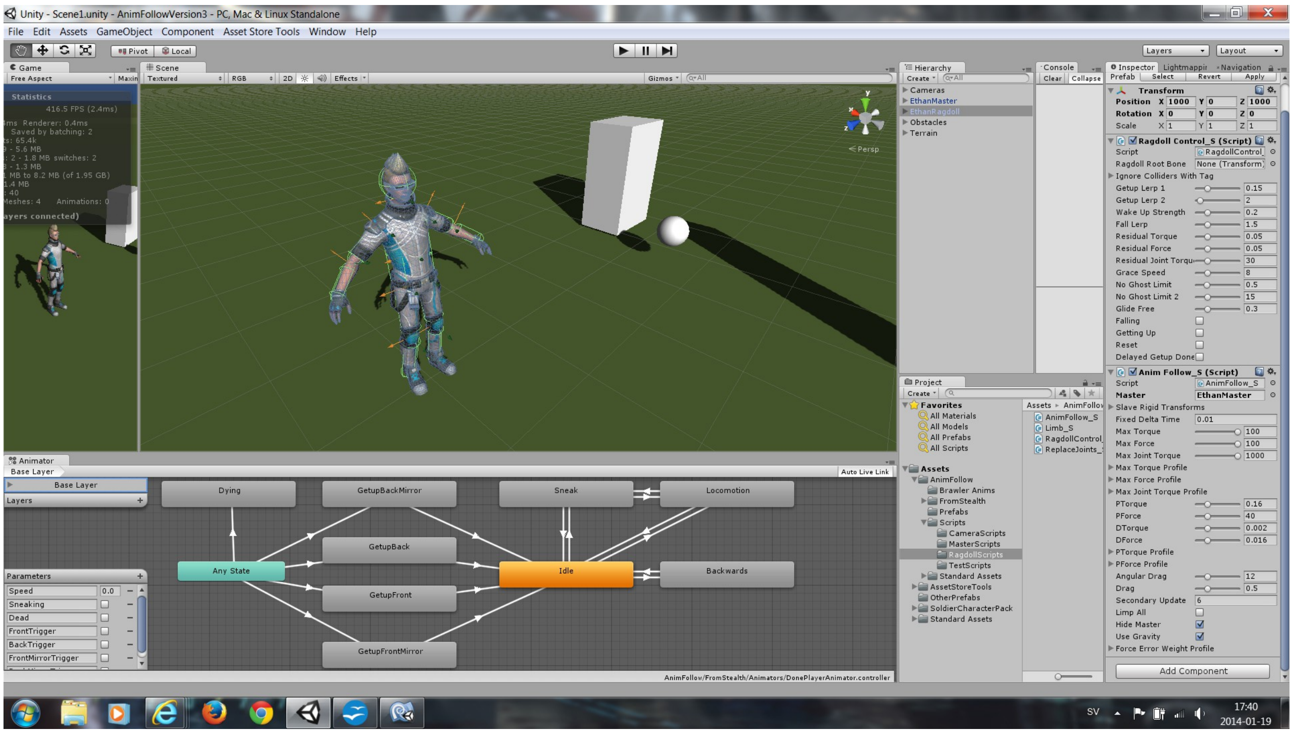
Appendix

To set the rotational limits of the ragdoll I recommend the following.

Make all the orange axes point to the characters right or upwards as in illustration 1. All the green axes should point forwards. Then set the limits to:

Limb	Low	High	Swing 1	Swing 2
Left thigh	-20	90	30	40
Left calf	-120	0	5	5
Right thigh	-20	90	30	40
Right calf	-120	0	5	5
Spine 1	-30	30	30	30
Spine 2	-30	30	30	30
Head	-50	50	50	60
Left biceps	-100	60	60	40
Left forearm	-150	5	5	5
Right biceps	-60	100	60	40
Right forearm	-5	150	5	5

These are good starting points for the rotational limits but you may want to do adjustments to fit your character. When making the ragdoll in the Unity wizard, make the T-pose have the arms slightly below horizontal.



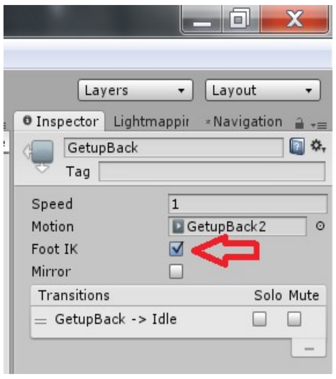
Transform
Position X 1000 Y 0 Z 1000
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Ragdoll Control_S (Script)
Script RagdollControl_
Ragdoll Root Bone None (Transform)
Ignore Colliders With Tag
Getup Lerp 1 0.15
Getup Lerp 2 2
Wake Up Strength 0.2
Fall Lerp 1.5
Residual Torque 0.05
Residual Force 0.05
Residual Joint Torque 30
Grace Speed 8
No Ghost Limit 0.5
No Ghost Limit 2 15
Glide Free 0.3
Falling ☐
Getting Up ☐
Reset ☐
Delayed Getup Done ☐

Anim Follow_S (Script)
Script AnimFollow_S
Master EthanMaster
Slave Rigid Transforms
Fixed Delta Time 0.01
Max Torque 100
Max Force 100
Max Joint Torque 1000
Max Torque Profile
Max Force Profile
Max Joint Torque Profile
PTorque 0.16
PForce 40
DTorque 0.002
DForce 0.016
PTorque Profile
PForce Profile
Angular Drag 12
Drag 0.5
Secondary Update 6
Limp All ☐
Hide Master ✓
Use Gravity ✓
Force Error Weight Profile

Illustration 1: Screen-shot of the interface.

The animator



GetupBack
Speed 1
Motion GetupBack2
Foot IK ✓
Mirror ☐
Transitions
GetupBack -> Idle

Illustration 2: Check IK

You may use any animator, but for the RagdollControl script to work, the four get up animations and the transitions to and from them should exist as in the example project. Make the transitions from the get up animations non atomic. Get up and idle animations must have the same names (including capitalization) as in example project.For the included Get up animations I have found it beneficial to check the Foot IK, See illustration 2. That might not be the case if you provide your own get up animations.

The parameters for the RagdollControl script

IceOnGetup
These transform will have no friction during get-up to avoid for example legs to snag.

ignoreCollidersWithTag
Collisions with colliders with these tags will be ignored by the limb script and will not be reported to the ragdollControl script. Game objects named “Terrain” are also ignored. This is useful for things that the ragdoll is supposed to walk on.

getupAngularDrag
The angular drag of all limbs during the get-up.

getupDrag
The drag of all limbs during the get-up.

fallLerp
After a collision the characters maxStrength lerps from the residual strengths to zero with this speed. Because the ragdoll has some residual strength it can be made to look more life like than a regular ragdoll. E.g. if falling when running it will (if tuned properly) try to take a few more steps

and not just go limp in the blink of an eye.

ResidualStrengths
The strengths immediately after the collision. Lerps to zero with fallLerp speed. Makes the ragdoll try to follow the animation even after a fall is triggered.

residualIdleFactor
A factor that multiplies the residualStrengths if the character was in the Idle state when hit.

graceSpeed
Collisions with relative speed higher than this will trigger a fall. Adjust this to set how easy a fall is triggered.

noGhostLimit1
Above this total character distortion a fall is triggered if not in the getting up or the falling state. Adjust this to set how easy a fall is triggered.

noGhostLimit2
Above this total character distortion a fall is triggered if not in the falling state.

glideFree
Makes the character glide free from an object if collision is not severe.

falling
Indicates when character is in falling state.

gettingUp
Indicates when character is in getting up state.

jointLimits
Indicates when joint limits are active.

There are additional, less critical, parameters that can be made public and adjusted by uncommenting a directive in the beginning of the script. These are:

bool **fellOnSpeed** = false; // For tuning. Tells the reason the fall was triggered

float **limbErrorMagnitude**; // This may be interesting to see if you are tuning the noGhostLimits. Read from AnimFollow. Contains the magnitude of the total position error of the limbs. When this is above the noGhostLimit falling is triggered

float **settledSpeed** = .1f; // When ragdollRootBoone goes below this speed the falling state is through and the get up starts

float **masterFallAnimatorSpeedFactor** = .4f; // Animator speed during transition to get up animations

float **getup1AnimatorSpeedFactor** = .25f; // Animation speed during the initial part of the get up state is getup1AnimatorSpeedFactor * animatorSpeed

float **getup2AnimatorSpeedFactor** = .85f; // Animation speed during the later part of the get up state is getup1AnimatorSpeedFactor * animatorSpeed

float **contactTorque** = 1f; // The torque when in contact with other colliders

float **contactForce** = 1f;

float **contactJointTorque** = 3f;

float **getupLerp1** = .15f; // Determines the initial regaining of strength after the character fallen to ease the ragdoll to the masters pose

float **getupLerp2** = 2f; // Determines the regaining of strength during the later part of the get up state

float **wakeUpStrength** = .2f; // A number that defines the degree of strength the ragdoll must reach before it is assumed to match the master pose and start the later part of the get up state

float **toContactLerp** = 70f; // Determines how fast the character loses strength when in contact

float **fromContactLerp** = 1f; // Determines how fast the character gains strength after freed from contact

float **maxTorque** = 100f; // The torque when not in contact with other colliders

float **maxForce** = 100f;

float **maxJointTorque** = 10000f;

float **maxErrorWhenMatching** = .1f; // The limit of error acceptable to consider the ragdoll to be matching the master. Is condition for going to normal operation after getting up

The parameters for the AnimFollow script

master (Transform)
Assign your master character.

slaveRigidTransforms
An array containing all transforms that have a rigid body component.

slaveExcludeTransforms
An array containing transforms that are not mimicked. These transforms will always hold the local rotation they had in the initial pose. Use this to exclude transforms that can not be mimicked because it is constrained by a collider on the ragdoll.

fixedDeltaTime
The update time for the physics loop.

maxStengths
Clamps the strengths.

jointDamping
The damping of the limbs movement relative their parents.

maxStrengthProfiles
Clamps the strengths on a per-limb basis.

jointDampingProfile
The damping of the limbs movement relative their parents on a per-limb basis.

P-Strengths
The proportionality of the muscle regulator. Use this to tune the ragdoll to follow the master properly.

D-parameters
The D-parameters are the derivative term of the PD-controller.

P-StrengthProfiles
The proportionality of the muscle regulator on a per-limb basis. Use this to tune the ragdoll to follow the master properly.

angularDrag
The angular drag of the ragdoll´s rigid bodies.

drag
The drag of the ragdoll´s rigid bodies.

hideMaster
If true the master renderer is disabled and only the ragdoll is visible. For debugging it can be beneficial to see both the master and the slave.

useGravity
If true gravity is applied to the ragdoll´s rigid bodies.

forceErrorWeightProfile
The limb weight for calculating the ragdoll´s total distortion.

AnimFollow version history

Version2

- Fixed bug in Limb script (bugged at zero or >2 ignoreMe tags).
- Added noGhostLimit2 to RagdollControl.
- Added orientateY to RagdollControl.

- Added mirror getup animations to RagdollControl.
- Added glideFree to RagdollControl.

Version3

- Replaced localTorque with jointTorque.
- Added secondary update parameter for improved performance.

Version4

- Included foot IK.
- Added shooting to the example scene.
- Added joint limit control that only use joint limits during fall.
- Added slaveExcludeTransforms.
- Changed the ragdoll in the example to a twelve rigidbodies ragdoll.
- Changed the AnimFollow script to be set up for a twelve rigidbodies ragdoll.
- Namespaced the scripts.
- Added possibility to have custom rigidbody drag during get-up animations.
- Eliminated foot wiggle during fall.

Version5

- Fixed a bug in the foot IK that made the knees bend strange on some models.
- Added a stay dead on head-shot feature. Character is destroyed next time it goes off camera.