Ragdoll & Hit Reaction Manager

Introduction:

Lightweight accurate physics hit reaction and ragdoll system.
Works on humaniod and generic rigs.
Simulates hit reaction on hit body parts and blends with ragdoll and animator.
Get up animations included.
Also included is the third person shooter game that implements above systems and examples of usage of ragdoll manager on standard assets AIThirdPersonCharacter.

I recommend creating new project before importing package.

NOTES:
- after creating project click on menu item 'Ragdoll Manager Package/ Setup Tags Layers' that creates layers and tags and physics setup required for scenes in package.

Examine physics matrix setup.
Exclude collision between character capsule collider and ragdoll body part colliders by:
- excluding capsule collider layer and body parts collider layer in physics matrix
- disable capsule collider and set rigidbody to kinematic in RagdollManager::OnHit callback and revert it in RagdollManager::LastEvent or RagdollManager::OnStartTransition (in OnStartTransition method animatior gets enabled again)
- make capsule collider trigger and set rigidbody to kinematic in RagdollManager::OnHit callback and revert in RagdollManager::LastEvent.

In scenes I use disabling capsule collider and setting rigidbody to kinematic.
Also disable all kinds of movement when entering ragdoll state. In demos I use RagdollManager::OnHit

delegate to disable movement to get accurate results.
(I disable ThirdPersonCharacter movement, root motion, NavMeshAgent etc. They are enabled again in RagdollManager::LastEvent delegate on ragdoll exit).

Included are examples of usage of ragdoll like applying force to ragdoll, applying force to individual body part of ragdoll, creating and interaction with joints.
System is more oriented towards performance, so rigid bodies are not colliding with others when animated.
For more control of what is done on disable / enable ragdoll look at _enableRagdoll() and _disableRagdoll() methods in RagdollManager.cs.

To import only ragdoll system to your own project:
- copy RagdollSystem folder and paste to your project.
- copy Editor folder (not including standard assets CrossPlatforrmInput and ImageEffects) and paste to your project.

INSTRUCTIONS FOR SETUP - HUMANOID

- Drag humanoid model onto the scene.
- Drag RagdollManagerHum script to model.
- Create Ragdoll by clicking 'Ragdoll Wizard' button and assign bone transforms.
Its very similar to default unity way of creating ragdoll.
- Enter names of get up from back & front animation states if you want to use it.
Get up animations are included in package or use your own. You dont have to use transitions to get up states. System uses CrossFade() when necessary, but create transition from get up states so you can transition to other.
- in humanoid rig you can create ragdoll in single click by using humanoid animator bone transforms.
- assign physics material to apply to all body part colliders.

note: Remember to set OnGetUp event on import animation settings if using your own
get up animations.
Set OnGetUp at the end of each get up animation.
- Enter new or use default values of hit interval, hit resistance, hit tolerance and hitReactionTimeModifier fields.
- To start ragdoll, call RagdollManager::StartRagdoll or RagdollManager::StartHitReaction from code.
- Note that its smart to disable all movement when in ragdoll mode like root motion, ThirdPersonCharacter movement, NavMeshAgent and other.

INSTRUCTIONS FOR SETUP - GENERIC

- Drag generic model onto the scene.
- Drag RagdollManager script to model.
- In generic setup you must create manualy colliders, rigid bodies and joints for your model.
- After creating colliders drag and assign all transforms on which are colliders created to RagdollManager.RagdollBones field and click 'AddColliderScripts' button.
- assign physics material to 'Physics Material' firld to apply to all body part colliders.
- If you wish to change some transform, click 'AddColliderScripts' button again - it has to be clicked at the end last when all ragdoll colliders and other components on transforms are done.
- There is no premade get up animation on generic ragdoll, but you can see how to create them on
human and horse (or other four legged creature) in tutorial video and package demos.
- If you want to constrain creature legs, assign lower legs transform BodyColliderScript to 'LeftKnee' or 'RightKnee'. In humanoid setup legs gets constraied automaticly because we already know humanoid setup (how many legs there are and where they are).
- Choose if you want them constrained with joints or making them kinematic.

note: Remember to set OnGetUp event on import animation settings if using your own
get up animations.
Set OnGetUp at the end of each get up animation.

- Enter new or use default values of hit interval, hit resistance, hit

tolerance and hitReactionTimeModifier fields.
- To start ragdoll, call RagdollManager::StartRagdoll or RagdollManager::StartHitReaction from code.
- Note that its smart to disable all movement when in ragdoll mode like root motion,
ThirdPersonCharacter movement, NavMeshAgent and other. You can do that in RagdollManager::OnHit callback.
You can see example in PlayerControl script.
AIThirdPersonCharacter from sample assets is using ThirdPersonCharacter script for movement.
I modified it just a little by adding field 'simulateRootMotion' which enables / disables root motion mimicking in OnAnimatorMove() method.

AIThirdPersonCharacter from sample assets is using ThirdPersonCharacter script for movement.
I modified it just a little by adding field 'simulateRootMotion' which enables / disables root
motion simulation in OnAnimatorMove() method.
Setup instructions video1 : https://youtu.be/vaxXxYa9lZs
Setup instructions video2 : https://youtu.be/_CuqHpkc7oM

RagdollManager script has useful callbacks you can use to your advantage:
- **OnHit** - event that starts when ragdoll manager start hit reaction or ragdoll mode.
(StartRagdoll() or StartHitReaction() methods sets flags to start one of those modes, but
actual mode is started in next LateUpdate() method).
- **OnStartTransition** - event that start when ragdoll mode is finished and blending to
animated mode starts.
- **LastEvent** - event that will be last fired (when full ragdoll - on get up, when hit reaction -
on blend end).
- **OnTimeEnd** - event that will be fired when ragdoll counter reach

event time.It increments
in ragdoll state only.
- **OnBlendEnd** - event that will fire when blending to animated is finished.
- **OnGetUp** - event that will fire after get up animation end.

Scene Examples Requirement:

Layers:
1. PlayerLayer
2. NPCLayer
3. ColliderLayer
4. ColliderIgnoreLayer
5. FireBallLayer
6. TriggerLayer
7. DynamicObject

Tags:
1. NPC

Menu item 'Ragdoll Manager Package/Setup Tags Layers' creates these:

-Layers:
1. PlayerLayer
2. NPCLayer
3. ColliderLayer
4. ColliderIgnoreLayer
5. FireBallLayer

6. TriggerLayer
7. DynamicObject

-Tags:
1. NPC

-Axis:
1. Idle
2. Toggle Pause

-Physics matrix exclusions:
- Scripting define symbols: DEBUG_INFO - used for additional debug information.

About Components:

HorseController.cs

Controls horse ragdoll system, implemets IRagdollUser interface se projectiles can be interact
with. Moves horse with reference to ThirdPersonHorse.cs.
Implements example of getting up horse upon exiting ragdoll system.
fields:
(Transform)OrientTransform - helps orient horse when ragdolled.
IRagdollUser.cs:
Interface. Implements StartRagdoll, StartHitReaction and other methods/properties that
BallProjectile script interacts with.
NPCControl.cs:
Controls NPCs in scene (Swat) that chase and shoot after player.
fields:
(float)Additional Y Rotation - additional rotation that will be applied on spine transform on y
axis in LateUpdate. Purpose is to point upper parts of body in direction forward.

(float)Additional X Rotation - additional rotation that will be applied on spine transform on x
axis in LateUpdate. Purpose is to point upper parts of body in direction forward.
(Transform)ChestTransform - script rotates this transform towards aiming target. If left empty
script will try to assign from animator human bones.chest.
PlayerControl.cs
Script that controls player.
variables:

(OrbitCameraController.cs)Camera - camera that follows player
(float)Additional Y Rotation - additional rotation that will be applied on spine transform on y
axis in LateUpdate. Purpose is to point upper parts of body in direction forward.
(float)Additional X Rotation - additional rotation that will be applied on spine transform on x
axis in LateUpdate. Purpose is to point upper parts of body in direction forward.
RagdollUser.cs
Basic ragdoll implemetation for interaction with projectiles.
Implements IRagdollUser interface.
RagdollUserGenericHumanoid.cs
Implements IRagdollUser interface with generic rig human. Demonstrates code for getting up
after exiting ragdoll mode.
fields:
(Transform)OrientTransform - helps orient human when ragdolled.
RagdollUserUnityTPC.cs
Implements IRagdollUser interface on Unity Standard Assets ThirdPersonCharacter script.
ThirdPersonCharacter.cs
Based on Unity Standard Assets and suited for current needs.
Controls character movement.
fields:

(float)MovingTurnSpeed - speed of turning when moving.
(float)StationaryTurnSpeed - speed of turning when stationary.
(float)MoveSpeedMultiplier - multiplies with root motion speed.
(float)AnimatorSpeed - speed of animator animations.
(**PhysicsMaterial)MaxFrictionMaterial** - max friction material (when stationary).
(PhysicsMaterial)ZeroFrictionMaterial - zero friction material (when moving).

ThirdPersonHorse.cs

Attempt of ThirdPersonCharacter implementation for horse. Used for moving horse.
fields:
(**PhysicsMaterial)MaxFrictionMaterial** - max friction material (when stationary).
(PhysicsMaterial)ZeroFrictionMaterial - zero friction material (when moving).
(Collider[])Colliders - collider array to encapsulate horse to collide with enviroment.
(Transform)ForwardRay - transform set on front of horse (used for angle control).
(Transform)BackRay - transform set on back of horse. (used for angle control).
(float)MoveSpeedMultiplier - multiples animator root motion.
BallTrigger.cs
Collider that assigns projectile prefab on character when entered.
fields:

(BallProjectile.cs)Ball_prefab - prefab that will be assigned on character when entering collider.

GameControl.cs

Controls scene pause / unpause, toggle screen keys information and adjusts player hit
reaction mode.
fields:
(UI.Text)Info UI - Text UI to draw key information.
(UI.Text)PlayerInfoText - displays current player ammunition.
(bool)HideCursor - hide cursor upon start.

GrabTrigger.cs

Script that creates ConfigurableJoint on collided body and connect with this rigid body
Testing RagdollManager bodyparts interaction with joints.

OrbitCameraController.cs

Camera controller that look towards, rotates around and zoom in and out to player.
fields:

(Transform)Default Camera Target - default target transform that camera follows.
(float)Angular Speed - rotation speed of camera.
(float)Min X Angle - camera minimum rotation on x axis.
(float)Max X Angle - camera maximum rotation on x axis.
(float)Min Y Angle - camera minimum rotation on y axis
(float)Max Y Angle - camera maximum rotation on y axis
(float)Min Z - camera minimum zoom to target
(float)Max Z - camera maximum zoom to target
(float)Z Step - zoom increments

RayShootRM.cs

Shoots ray from the camera and interacts with any ragdoll manager on scene.
Left button starts ragdoll. Right button starts hit reaction.

SignCarrier.cs
Carries 3D text mesh to parent transform position.
fields:
(Tranfsorm)Parent - parent transform.

Trigger.cs

Scripts that triggers certain actions defined in TriggerTypes enumeration (currently only
Jump and Crouch).
Its used for testing AIThirdPersonCharacter and ThirdPersonCharacter , so it skips if
object dont have
AICharacterControlCustom component which uses sample assets ThirdPersonCharacter
script.
fields:
(TriggerTypes)Trigger Type - choose from trigger type enum.

Utilities.cs

Collection of enums, delegates and other used in other scripts.

BallProjectile.cs

Projectile script checking for collision between set collider layer by casting sphere from last postion to current.
fields:
(float)lifetime - lifetime of projectile. Starts incrementing after has been shot. When expires -
call OnLifetimeExpire delegate or destoys object if delegate is null.
(float)hit strength - force velocity of projectile that will apply on object that got hit.
(LayerMask)colliding layers - layer mask used for hit checking.
HarpoonBallProjectile.cs
Derived from BallProjectile script. Has additional force field that multiplies hit strength and
method for setup HarpoonBallProjectile.
fields:
- derived from BallProjectile script.

additional:
(float)force - additional force that multiplies with hit strength to increase impact on ragdoll.

HarpoonBallProjectile.cs

Derived from BallProjectile script. Has additional force field that multiplies hit strength and
method for setup HarpoonBallProjectile.
fields:
- derived from BallProjectile script.

additional:
(float)force - additional force that multiplies with hit strength to increase impact on ragdoll.
InflatableBall.cs
Derived from BallProjectile. Holds methods for inflate and setup for creating inflatable ball.
variables:
- derived from BallProjectile script.

RocketBallProjectile.cs

Derived from BallProjectile. When hits character, it activates afterburner particle system
and lift hit body part
and rest of character with it. Has method for setup RocketBallProjectile .
variables:
- derived from BallProjectile script

additional:
(float)up force - up force that lift hit body part
SoapBallProjectile.cs
Derived from BallProjectile script. When hits character, it applies equal up force on all
body parts lifting the character up. Has method for setup SoapBallProjectile .
variables:
- derived from BallProjectile script.

additional:
(float)up force - up force that lift all body parts.

ShootScript.cs

Abstract script that shoots projectiles and interacts with IRagdollUser interface.
fields:
(**BallProjectile.cs)ProjectilePrefab** - projectile that will be shot.
(Transform)FireTransform - fire position and direction from which the projectile will be
shot.
(GameObject)Owner - owner of shoot script (to ignore).

PlayerShootScript.cs

Derived from ShootScript. Enables Shooting of projectiles by input.

NPCShootScript.cs

Derived from Shoot script. Shoots projectiles on intervals.
Used on npcs.
fields:
- derived from ShootScript.cs
additional:
(float)ShootInterval - shoot interval.
ColliderScript.cs
Small class that holds reference to parent object.
fields:
(GameObject)ParentObject - reference to parent object.

BodyColliderScript.cs

Derives from ColliderScript and holds additional information of collider object: hit body
part and flag that notifies if this bodypart is critical (like head hit).
fields:
(bool)Critical - is collider hit critical.
(BodyParts)Body Part - body part enumeration on collider.
(RagdollManager.cs)ParentRagdollManager - reference to parent ragdoll manager.

RagdollManager.cs

Abstract class.
Workhorse of entire package alog with its derived classes. It simulates hit reaction and ragdoll
phyics.
fields:
(Transform[])Ragdoll Bones - transforms on which rigid bodies and colliders are applied.If left
empty,it will automaticly be taken from humanoid setup.
(bool)UseJoints - use joints for constraints, if disabled, constrained bodies will be kinematic.
(float)Blend Time - blend from ragdoll to animator time.
(float)Hit Interval - ragdoll manager will ignore hits outside given interval.
- Always = always take hits

- OnBlend = ignore hits before blending to animator
- OnGettingUp = ignore hits before on getting up animations
- OnAnimated = allow hits only when animated

(float)Hit Resistance - time modifier of time provided in ragdoll mode on hit. Higher the value,
higher the r
resistance (shorter the time).
(float)Hit Reaction Tolerance - If force velocity magnitude excedes this value, character falls
into ragdoll mode.
(float)Weight - weight of character. Influences hit reaction.

RagdollManagerHum.cs

RagdollManager intended to be used on humanoid rigs.
fields:
- derived from RagdollManager.cs

additional:
(bool)Enable Get Up Animation - enable get up animation after ragdoll time ended
(string)Name Of Get Up From Back State - name of get up from back animation state.
(string)Name Of Get Up From Front State - name of get up from front animation state.
Button for creating ragdoll system on humanoid rigs similar to Unity.

Button for deleting created ragdoll system.

RagdollManagerGen.cs

RagdollManager intended to be used on generic rigs.
fields:
- derived from RagdollManager.cs

additional:
Button for adding collider scripts on ragdoll transforms (make sure to add all transforms
to RagdollBones array before clicking).
Button for removing created ragdoll system.

About

Demo Videos:

Demo: https://youtu.be/L0KOcQawIrA

Basic Humanoid Setup Tutorial: https://youtu.be/vaxXxYa9lZs

Basic Generic Setup Tutorial: https://youtu.be/_CuqHpkc7oM

Demo Apps:

Demo 1: https://jamariothe.github.io/RagMan1/

Demo 2: https://jamariothe.github.io/RagMan2/

Website:

https://sites.google.com/site/gamedevstreet/

Support & Contact

mariolelascontact@gmail.com

Other Projects:

https://www.assetstore.unity3d.com/en/#!/publisher/14594/page=1

Double Sided Standard, Mobile Legacy Shaders

Same flexibility, all in single draw call and pass.
Plus included are two face shaders that draw two textures - one on each face.
Video: https://youtu.be/z-u3CEfgYhY
Asset Store: http://u3d.as/idQ

Light Beams Shader Pack

Simple, cheap light beam / shaft, laser beam shader pack.
Video: https://youtu.be/5uudyIdTzQE
Asset Store: http://u3d.as/m0W

Combat Framework

Third person character controller & npc creation tool.
Locomotion, trigger system, ragdoll & hit reaction, animated hit reactions, weapon&shield, two handed wepons, bhow, dual wield modes, ledge climb, steps & ladders, uneven terrain traversing.
And much more...
Demo: https://youtu.be/elBDWgxl1q4
Player setup tutorial: https://youtu.be/ZF5jUOtJGxo
Weapon Setup: https://youtu.be/jWm-zzAPJyo
Generic Npc Setup: https://youtu.be/y8vGG1l5NQE

Dynamic Obstacle Avoidance
Video: https://youtu.be/1DgPuwl-zUA
Asset Store Page:
https://www.assetstore.unity3d.com/#!/content/87355

Gray Cat Character Controller
Demo Video: https://youtu.be/WaWAAWZSon0
Asset Store: http://u3d.as/16oR

Author:
Mario Lelas & GrayCat Team

Table of Contents