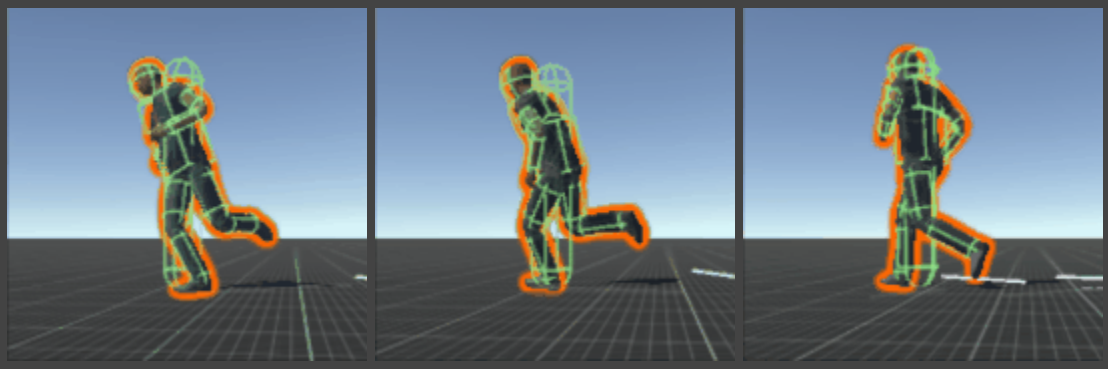
introduction

Renegade is a root-motion based third person controller for mecanim animator and humanoid rig.



---

overview

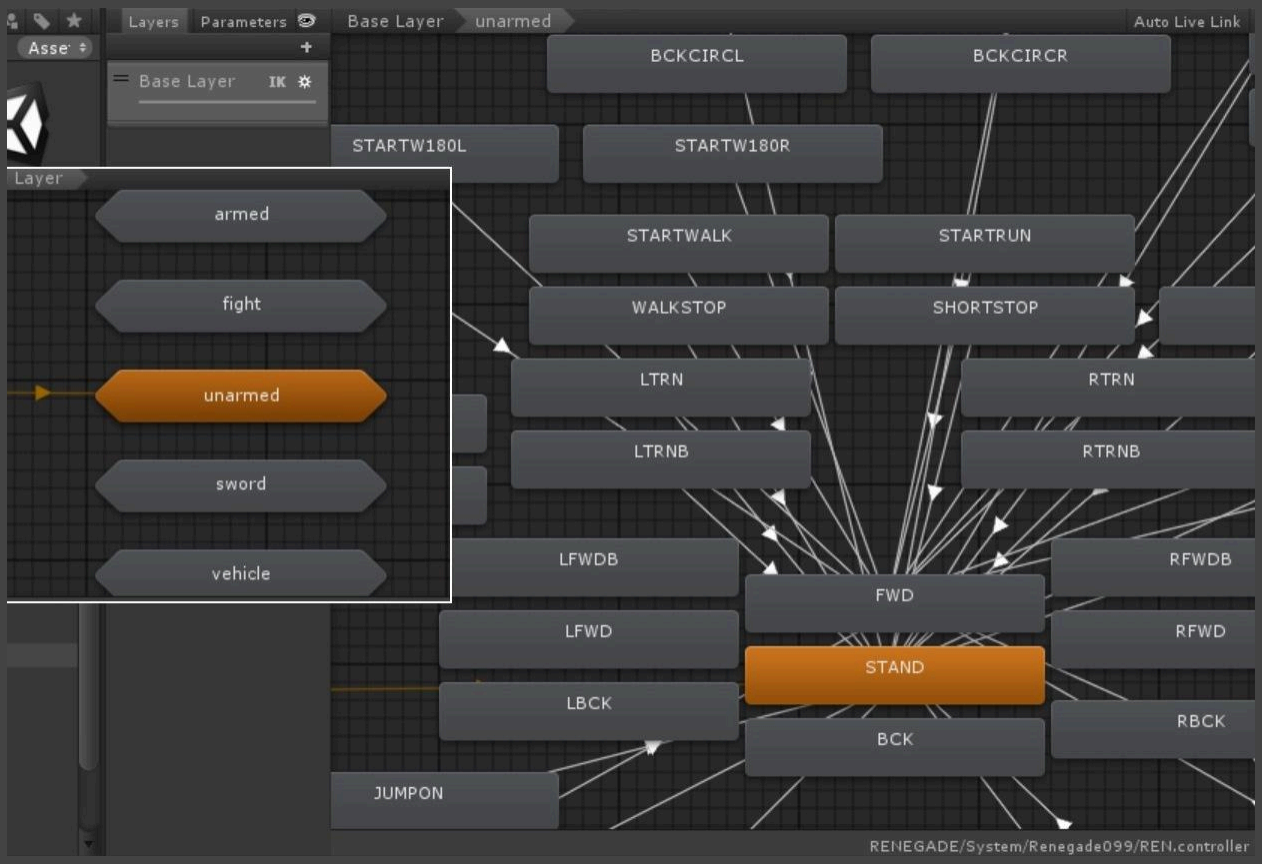
:: dynamic capsule collider
:: rigidbody
:: gamepad support

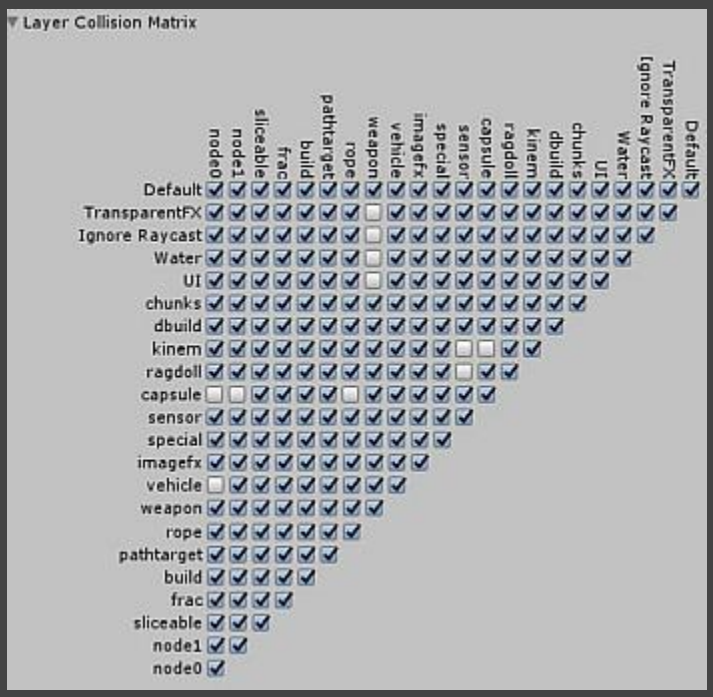
It is a modular design, most features and subsystems are selectable and can be switched on/off. No animation events (but possible to use) or curves. No dependencies.

Keeps track of states, state progression (normalized times), stances (right, left, neutral), velocities, foot positions, bodypart rotations, collisions (kinematic and physical) etc.

Debug options.

The system is built around the animator controller, made up of blend trees.


Layers
Parameters
Base Layer
unarmed
Auto Live Link
Base Layer
IK
Layer
armed
fight
unarmed
sword
vehicle
BCKCIRCL
BCKCIRCR
STARTW180L
STARTW180R
STARTWALK
STARTRUN
WALKSTOP
SHORTSTOP
LTRN
RTRN
LTRNB
RTRNB
LFWDB
RFWDB
FWD
LFWD
RFWD
STAND
LBCK
RBCK
BCK
JUMPON
RENEGADE/System/Renegade099/REN.controller

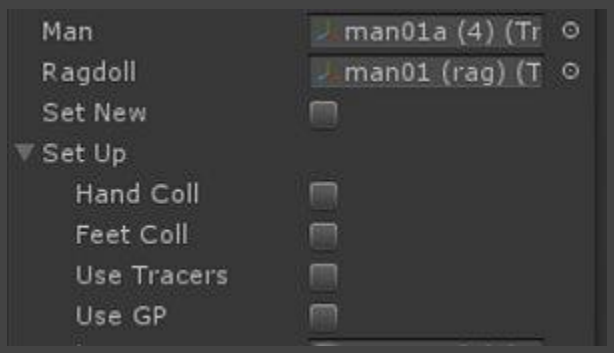

Layer Collision Matrix
Default
TransparentFX
Ignore Raycast
Water
UI
chunks
dbuild
kinem
ragdoll
capsule
sensor
special
imagefx
vehicle
weapon
rope
pathtarget
build
frac
sliceable
node1
node0

setting up

: create ragdoll out of your humanoid character
: duplicate
: add Renegade.cs [assign one character to man, assign other character to ragdoll slot] and tick SetNew
not needed in 0.99F+ :
: add rigidbody to it [constrains free rotation: x y z]
: add capsule collider to it

: start play and save your actor [copy]
: paste into scene

options:

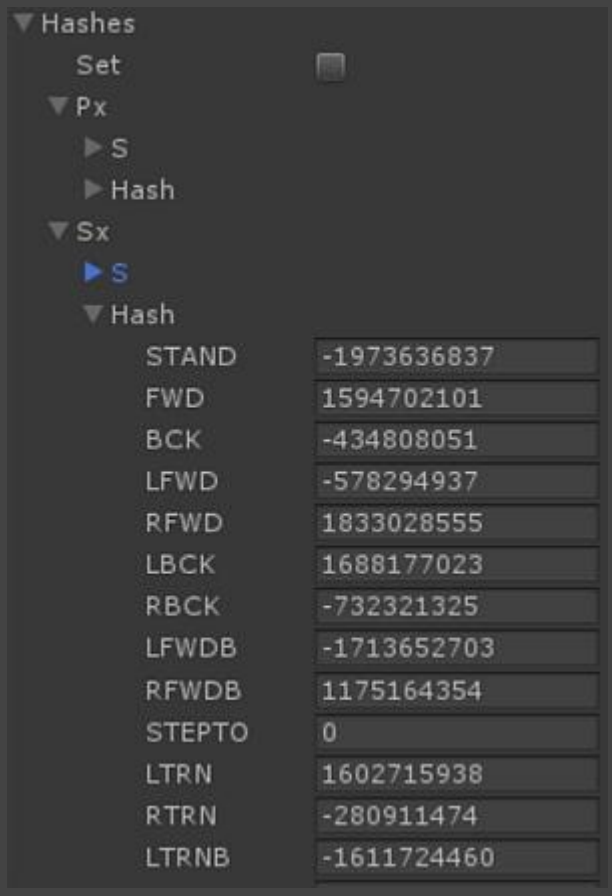


Renegade

the basics

Renegade.states show which state is on now.

Renegade.hashes is a list of animator parameters [px] and states [sx].



locomotion

grabbin

jumpMX

jump mechanics

the type of jumps

to initiate jump you must set true: which jump and which number
like this

Renegade.comp.jumpMX.jump.exeJump = true;
Renegade.comp.jumpMX.jump.x.exe[1] = true;
Renegade.comp.jumpMX.jumpOn.secKey = true;

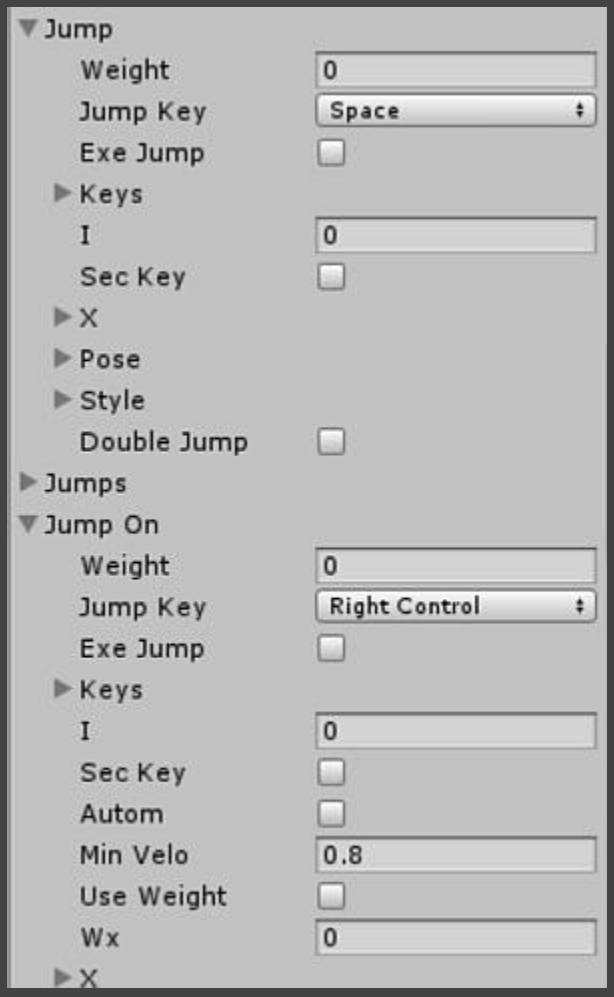
setting jumps

jumps are set up in .style

jump times in normalized time
jumpOT : jump-off time [when actor jumps up from the ground]
fallT : falling phase of the jump
canGoT : when new jump/anim can start
offsetT : start offset used [if want to start on diff foot / immediate jump etc.]
selected : which jump in the blend tree [parameter]

CONDITIONS: useState, usePose : must be this state/pose to start jump

weighting : used for ranking between jumps [which jump should go if more are ready/eligible]

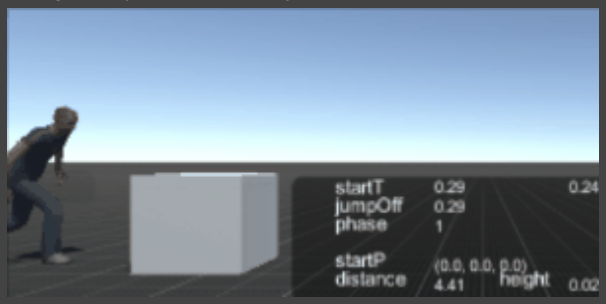


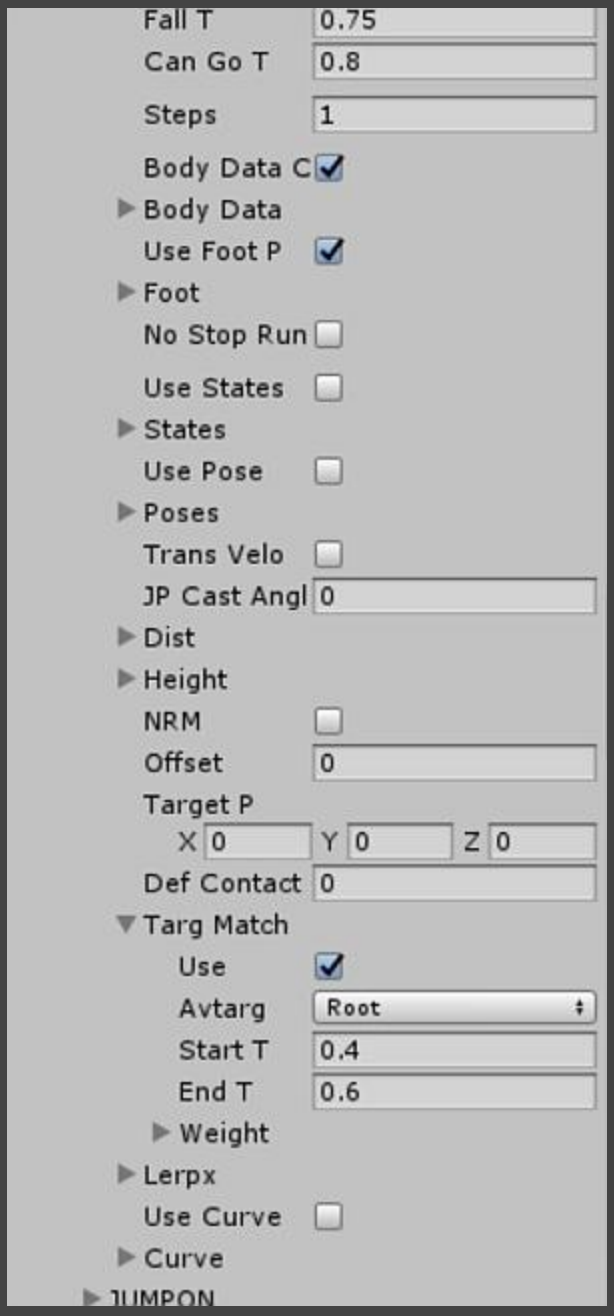
CONDITIONS: minVelo [minimum velocity to initiate jump]

: jumps can be automatic [if ready, based on raycast hits]
or initiated by input

: jumps are divided in phases [jumpMX.onJumpX.phase]

1. jump animation initiated
2. jump-off : style.jumpOT [jumpMX.jPoint()]
3. contact : style.contactT
4. falling : style.fallT [jumpMX.fallPoint()]





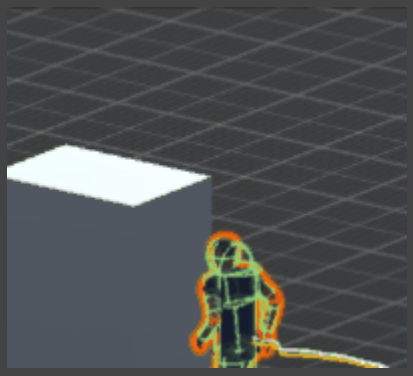
bodyData : for collision detection during jump

CONDITIONS: distance and height

targetMatch or Lerpx : to modify jump landing point

Curve : to modify jump height curve

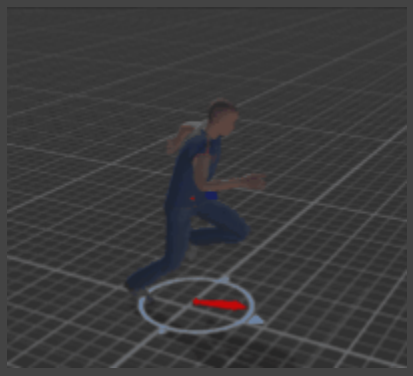
NOTES:
rayx.jOn.contactOffset must be set to match jump height



[jump].style.switchOffColl setting can only be be used with typ.rootMotion [not with typ.Physics]

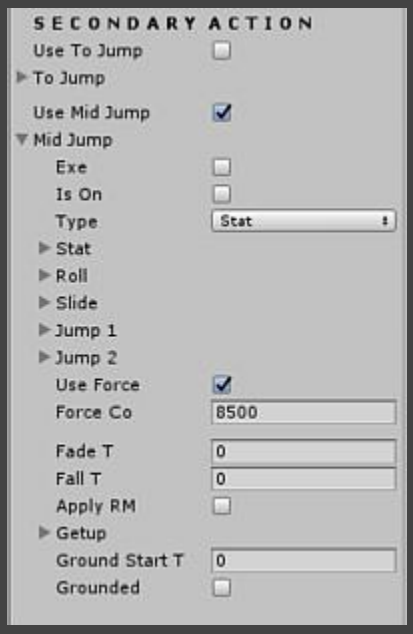
to-jump

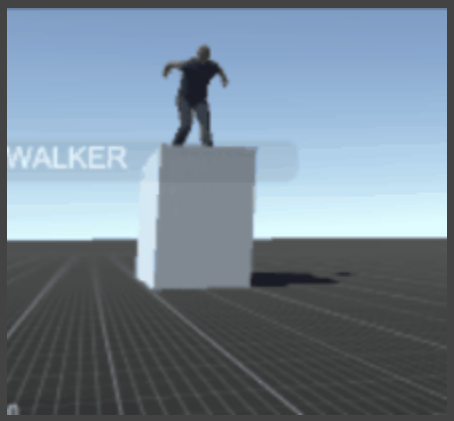
: jump / step before the actual jump



mid-jump

: jump can be modified into **second jump** (roll, slide etc) : midJump [inAir] secondaryAction [grounded]





bodyPosition blending

Renegade.root.blendBodyP

RENEGADE :: THIRD PERSON SYSTEM FOR UNITY3D

fightMX

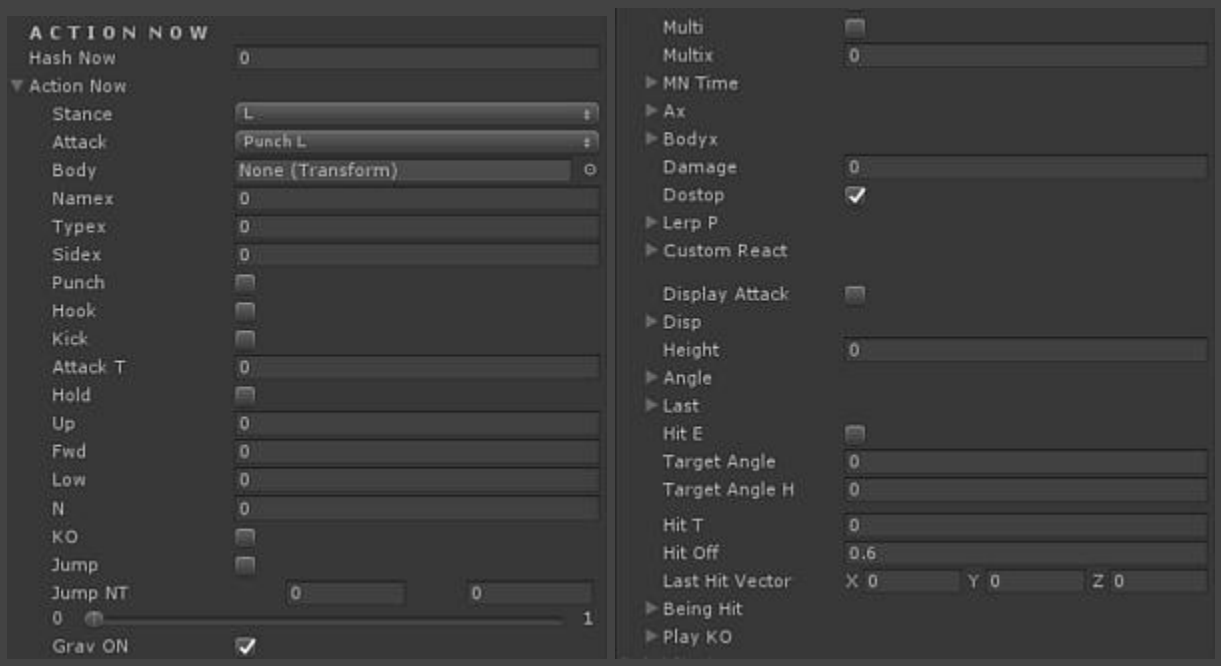
fight mechanics

Includes state tracking, enemy detection, lock on, attack audio

action types : punch, hook, kick, block, dodge, punch180, kick180

supports stances (right, left, neutral)

actionNow : shows action being executed and all parameters

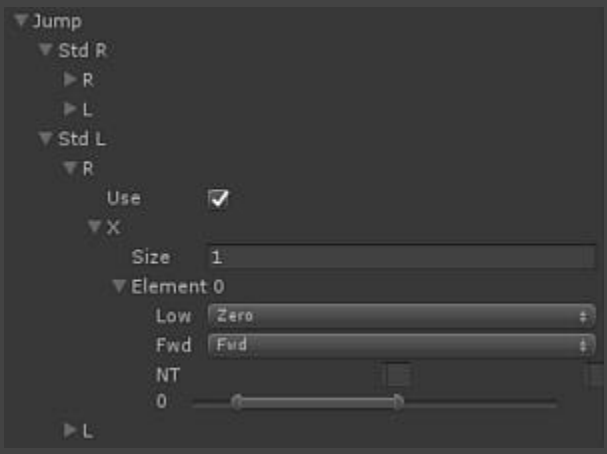


actionData

fightMX actions' attributes

jump

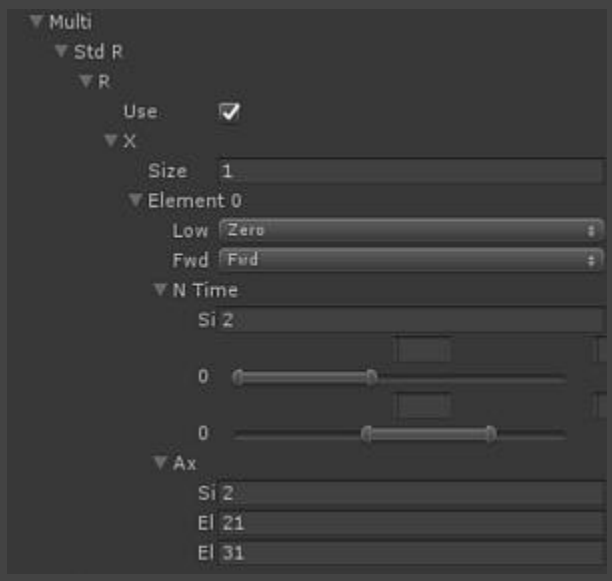
if punch or kick uses jump, here can set duration [normalized time range]
: set blend parameters fwd and low, and jump NT



multi-attack

can use animation that has more than one type of attack
on states:
RPUNCHR, RPUNCHL, LPUNCHR, LPUNCHL
RKICKR, RKICKL, LKICKR, LKICKL

define multi-attack in actionData.punch.multi and actionData.kick.multi
: set blend parameters fwd and low
: set NT (normalized time ranges for each attack)
: set attack types (.ax) [example 21: armR attack]
first digit> 1:armL 2:armR 3:legL 4:legR 5:head
secnd digit> 0:null 1:attack 2:defense



throwing

RENEGADE :: THIRD PERSON SYSTEM FOR UNITY3D

weaponMX

weapon mechanics

vehicles

---

createCar

---

enter - exit

for get-in

should set up:
in driveCX
.phys.ground
+ if using slip
.slip.prefab
.slip.fold

in dEnterExit
.ikPoint.seat
.setTrigIgnore (bool false)
.door.angle.limited

____
[gameplay] *after get-in should see:*

in Renegade
.states.driving (bool true)

.vehicle.dEnterExit
.vehicle.inUse (vehicle transform)
.vehicle.type.car (bool true)

in RenegadeGPX
.driveCX

in driveCX
useInput (bool false, if using gamepad)

---

chopperCX

helicopter physics

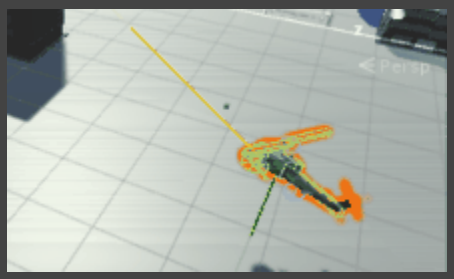
heightC : keeps helicopter stabilized at a given height
deadZone : zero uplift at steep angles

levelEq : noCorrect, Correct, CorrectZ, CorrectLMT
corrects helicopter angles, levels them out

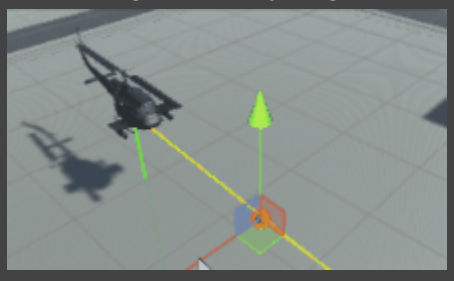
correctZ : only on Z axis
correctLMT : in given angle range [chopperCX.correctLMT.x and chopperCX.correctLMT.z]

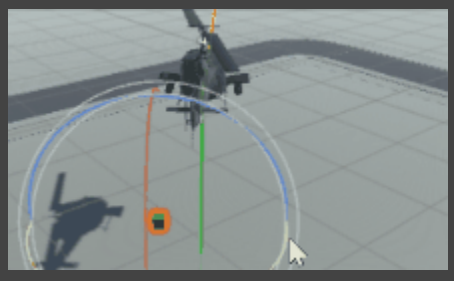
forceEQ : stabilizes helicopter turning

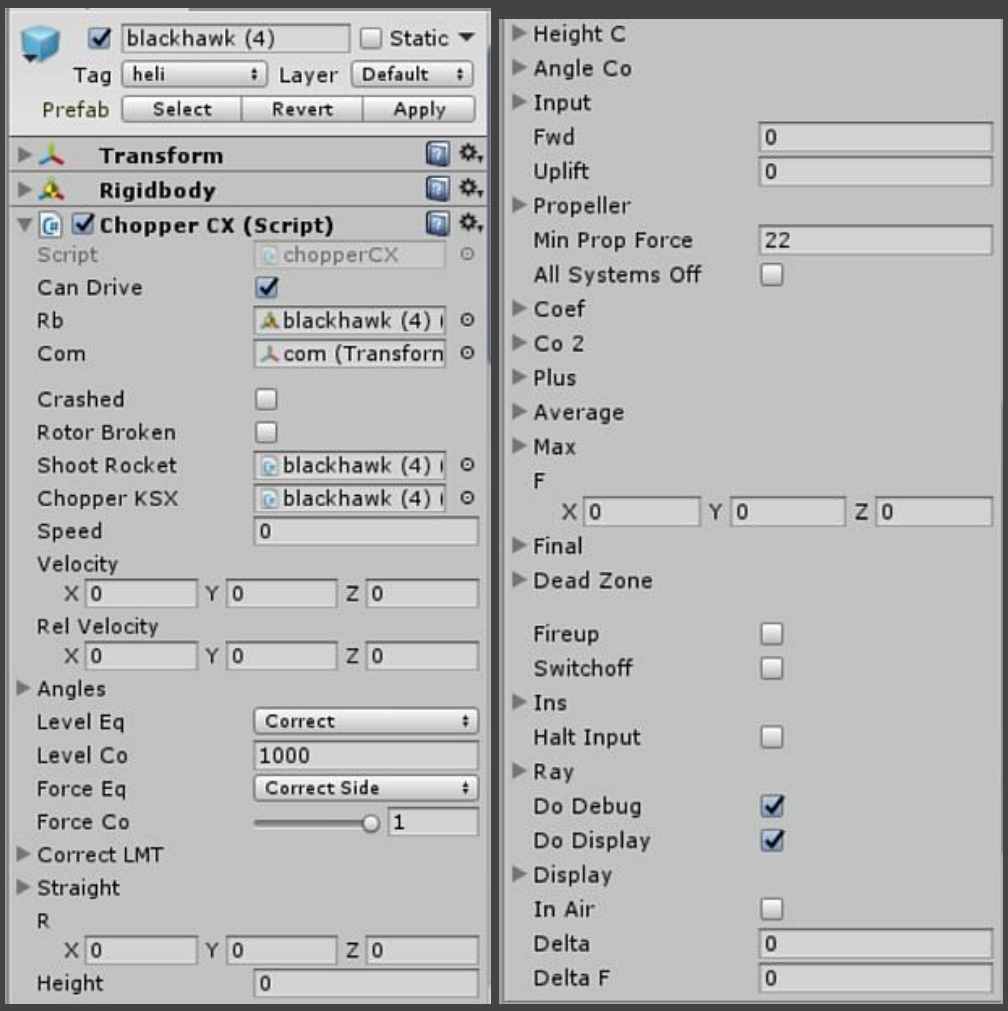
aiChopper



turn to target / turn by target







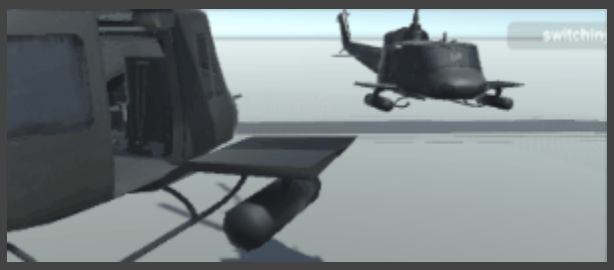
coef.upRate : upforces: 0.9 >> 0.9xVector3.up + 0.1xhelicopter.up
plus.fwd : additional helicopter forward force on input

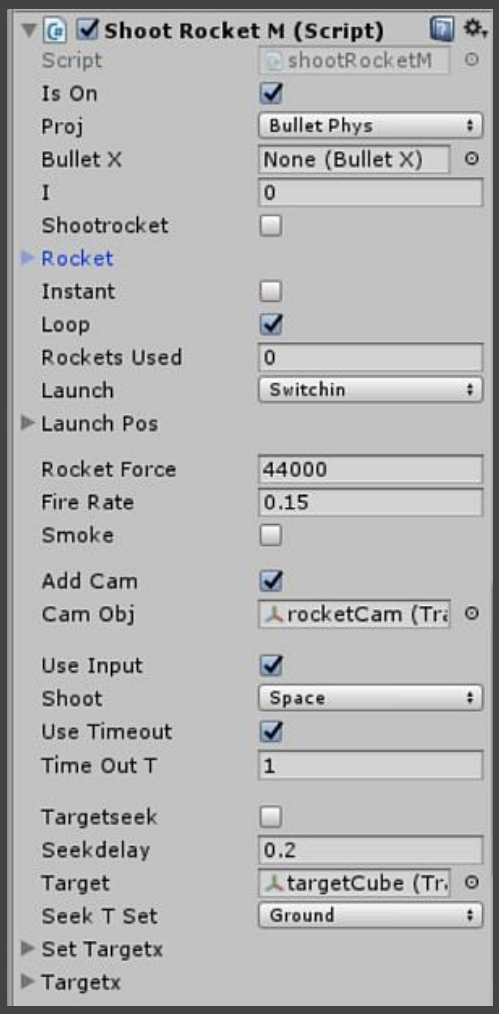
Ray.dwn : height measured by raycast
inAir : helicopter in air [height > 2]coef.side : the force added when helicopter is rotated on z axis

……
coef.side: the force multiplier [side] when helicopter is rotated on z axis
coef.fwd: the force multiplier [forward] when helicopter is rotated on x axis
coef.uplift: force multiplier [up] (dependant on co2. up/down-under/below average)

coef.yaw: torque increase multiplier on z axis
coef.h: torque multiplier on y axis
coef.v: torque increase multiplier on x axis
……
or correct.co : [Vector3] .x .y .z : torque multipliers *default value* [40, 1000, 2000]

shootRockets



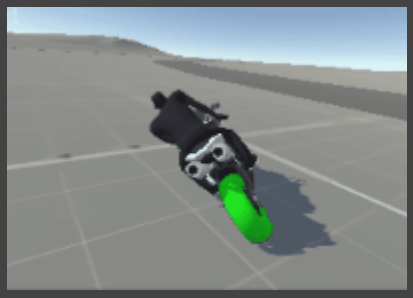


target seeking

INTEGRATIONS

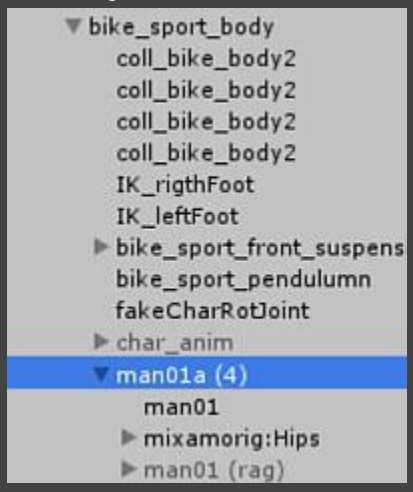
sportBike Pro

https://assetstore.unity.com/packages/templates/packs/sportbike-pro-kit-45928



- simple (just driving)

add Renegade man



change animator *REN > RiderAnimController*

copy-paste component *bikerAnim*

bikerAnim.cs
+

```
public Transform IK_rightFootTarget;
public Transform IK_leftFootTarget;
```

: set them up like hands in *OnAnimatorIK()*
: these IK target transforms should be under *bike_sport_body*
: also add *if (bikeController.crashed) return* in *OnAnimatorIK()*
+

```
public Renegade Renegade;
```

: change createRagdoll() >> createRagdollREN()

RENEGADE :: THIRD PERSON SYSTEM FOR UNITY3D

: add

```
public void createRagDollREN()
{
	Renegade.inc.rb.velocity = bikeController.m_body.velocity;
	transform.parent = null;
	Renegade.setRagdollOff ();
	Renegade.bones.ragd.rb.Hips.velocity = bikeController.m_body.velocity;

	ragdollLaunched = true;

	if (!bikeController.crashed)
	{
		bikeController.crashed = true;
		bikeController.m_body.centerOfMass = new Vector3(0, -0.2f, 0);
	}

}
```

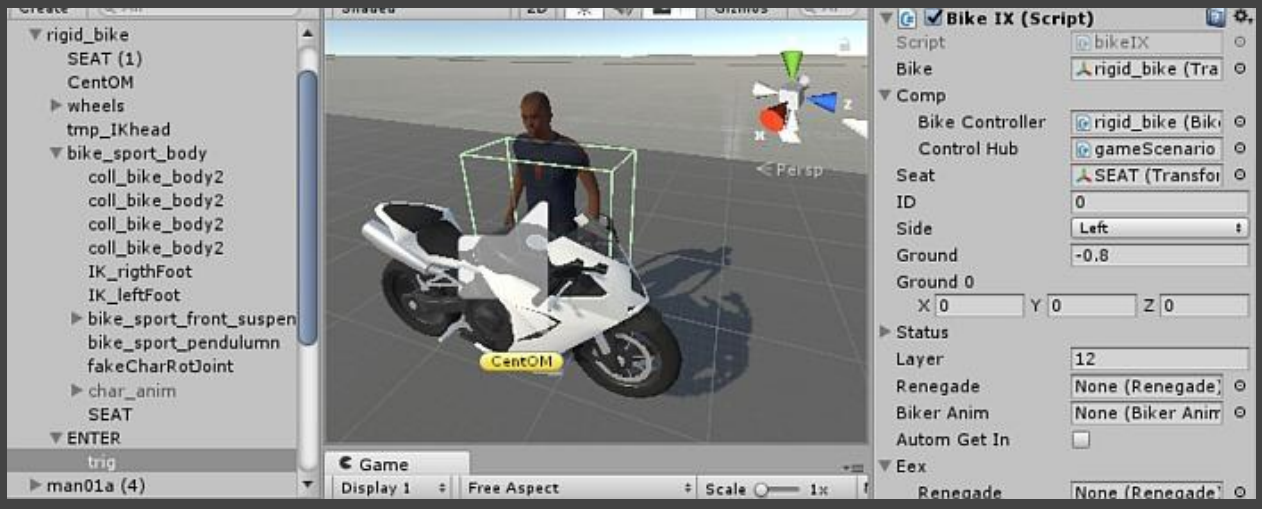
: in *Update()* change

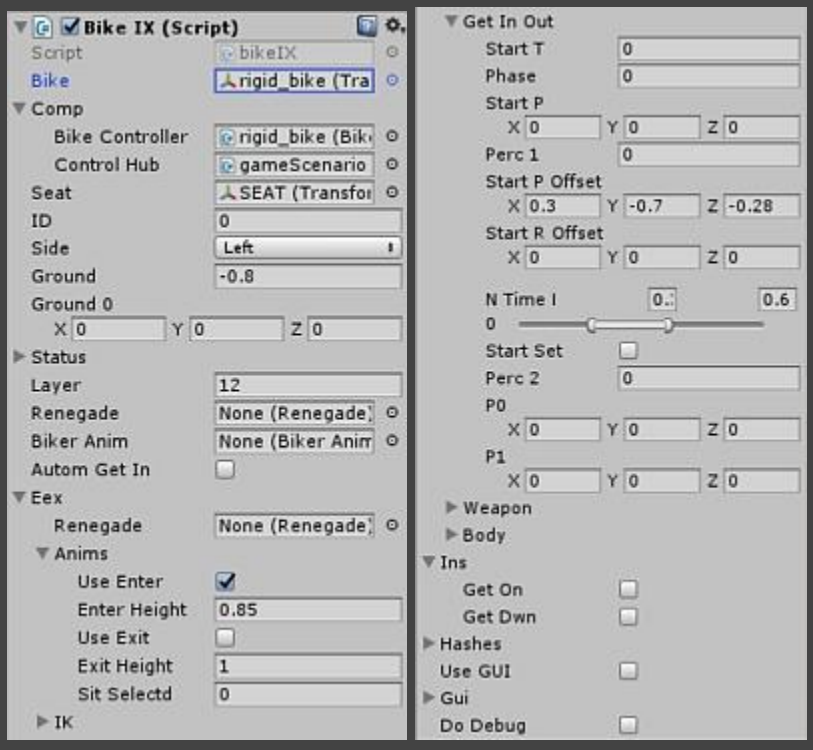
1. *if (fakeCharPhysJoint != null)* to *if (fakeCharPhysJoint != null && !bikeController.crashed)*
2. *else return* to *//else return*

+
in Renegade.cs
set bool *inGame* > false
set *root.useDragOnLandin* > false
set root.blendBodyP.adjustBodyP > false

- advanced (Renegade walk n bike driving)

uncomment bikeIX.cs and Renegade.cs / RenegadeGPX.cs lines w BikeController






```
public class bikeIX : MonoBehaviour {

	public Transform bike;

	[System.Serializable]
	public class CMP {

		public BikeController BikeController;
	}
	public CMP comp = new CMP();

	public Transform seat;

	public int ID;//eexused;

	public enum ix {left, right};
	public ix side;

	[System.Serializable]
	public class ST {
		public bool waiting;
		public bool gettinIn;
		public bool sittin;

		public bool wait4nextaf;//wait for next anim frame

		public bool gettinOut;
	}
	public ST status = new ST();

	public int layer = 12;//capsule
	public Renegade Renegade;//detected
	public BikerAnim BikerAnim;
	public bool automGetIn;//automatik get-in
```

```
[System.Serializable]
public class EEX {

	public Renegade Renegade;//gettin in

	[System.Serializable]
	public class Anims {

		public bool useEnter = true;
		public bool useExit;

		public int sitSelectd;
	}
	public Anims anims = new Anims();

	[System.Serializable]
	public class ik {

		public bool useHands = true;
		public bool useFeet = true;
		public bool useLookAt = true;
	}
	public ik IK = new ik();


	[System.Serializable]
	public class Weapon {
		public bool store = true;//should be stored on enter
		public Transform stored;
		public bool isStored;
	}
	public Weapon weapon = new Weapon();


	[System.Serializable]
	public class BD {

		public bool bodyDataOff = true;//switch trigger collision detection while riding
		public bool dirCtrlOff = true;//switch linecast hit detection off while riding
	}
	public BD body = new BD();


}
public EEX eex = new EEX();

[System.Serializable]
public class H {

	public bool Set;

	[System.Serializable]
	public class PX {
		//animator parameters
		[System.Serializable]
		public class S {

			public string lean = "lean";
			public string moveAlong = "moveAlong";
		}
		public S s = new S();

		[System.Serializable]
		public class H {

			public int lean;
			public int moveAlong;
		}
		public H hash = new H();
	}
```

```
		public PX px = new PX();

		[System.Serializable]
		public class SX {
			//animator states
			[System.Serializable]
			public class S {

				public string ridin = "ridin";
			}
			public S s = new S();

			[System.Serializable]
			public class H {

				public int ridin;
				public int getup;
			}
			public H hash = new H();
		}
		public SX sx = new SX();

	}
	public H hashes = new H();

	public bool useGUI;
	[System.Serializable]
	public class Gui {

		public GameObject Enter;
	}
	public Gui gui = new Gui();


	public bool doDebug;

	void Start () {

		if (!hashes.Set) {
			hashes.sx.hash.ridin = Animator.StringToHash ("Base Layer.bike." + hashes.sx.s.ridin);

		}
	}

	public void getInNoAnim1(){

		status.waiting = false;
		status.gettinIn = false;
		status.sittin = true;
		eex.Renegade.vehicle.type.bike = true;
		//eex.Renegade.vehicle.dEnterExit = this;

		eex.Renegade.man.position = seat.position;
		eex.Renegade.man.rotation = seat.rotation;
		eex.Renegade.man.parent = seat;

		eex.Renegade.inc.rb.isKinematic = true;
		eex.Renegade.inc.capsule.isTrigger = true;

		eex.Renegade.states.driving = true;
		eex.Renegade.animator.Play (hashes.sx.hash.ridin, 0);
	}


	public void OnTriggerEnter(Collider trigx){


		if (trigx.gameObject.layer == layer && !status.waiting && !status.sittin) {

			//if (doDebug) Debug.Log (driveCX.transform.name + " : man at vehicle");
```

```
                        eex.Renegade = trigx.gameObject.GetComponent<Renegade> ();
                        eex.Renegade.vehicle.inSight = bike;
                        eex.Renegade.vehicle.type.bike = true;


                        if (eex.Renegade.isPlayer) {

                                if (!automGetIn) {
                                        status.waiting = true;

                                        if (useGUI) {
                                                clixGetIn clixGetIn = gui.Enter.GetComponent<clixGetIn> ();
                                                //clixGetIn.driveCX = driveCX;
                                                //clixGetIn.ID = ID;
                                                gui.Enter.SetActive (true);
                                        }
                                }

                                if (automGetIn) {

                                        getInNoAnim1 ();
                                        Renegade = eex.Renegade;
                                        BikerAnim = Renegade.transform.GetComponent<BikerAnim> ();
                                        BikerAnim.enabled = true;
                                        BikerAnim.Renegade = Renegade;
                                        BikerAnim.useIK = true;
                                        BikerAnim.ragdollLaunched = false;
                                }
                        }


                }
        }

}
```