;; Auto-generated. Do not edit!

(when (boundp 'gazebo\_msgs\_new::ContactState)

(if (not (find-package "GAZEBO\_MSGS\_NEW"))

(make-package "GAZEBO\_MSGS\_NEW"))

(shadow 'ContactState (find-package "GAZEBO\_MSGS\_NEW")))

(unless (find-package "GAZEBO\_MSGS\_NEW::CONTACTSTATE")

(make-package "GAZEBO\_MSGS\_NEW::CONTACTSTATE"))

(in-package "ROS")

;;//! \htmlinclude ContactState.msg.html

(if (not (find-package "GEOMETRY\_MSGS"))

(ros::roseus-add-msgs "geometry\_msgs"))

(defclass gazebo\_msgs\_new::ContactState

:super ros::object

:slots (\_info \_collision1\_name \_collision2\_name \_wrenches \_total\_wrench \_contact\_positions \_contact\_normals \_depths ))

(defmethod gazebo\_msgs\_new::ContactState

(:init

(&key

((:info \_\_info) "")

((:collision1\_name \_\_collision1\_name) "")

((:collision2\_name \_\_collision2\_name) "")

((:wrenches \_\_wrenches) (let (r) (dotimes (i 0) (push (instance geometry\_msgs::Wrench :init) r)) r))

((:total\_wrench \_\_total\_wrench) (instance geometry\_msgs::Wrench :init))

((:contact\_positions \_\_contact\_positions) (let (r) (dotimes (i 0) (push (instance geometry\_msgs::Vector3 :init) r)) r))

((:contact\_normals \_\_contact\_normals) (let (r) (dotimes (i 0) (push (instance geometry\_msgs::Vector3 :init) r)) r))

((:depths \_\_depths) (make-array 0 :initial-element 0.0 :element-type :float))

)

(send-super :init)

(setq \_info (string \_\_info))

(setq \_collision1\_name (string \_\_collision1\_name))

(setq \_collision2\_name (string \_\_collision2\_name))

(setq \_wrenches \_\_wrenches)

(setq \_total\_wrench \_\_total\_wrench)

(setq \_contact\_positions \_\_contact\_positions)

(setq \_contact\_normals \_\_contact\_normals)

(setq \_depths \_\_depths)

self)

(:info

(&optional \_\_info)

(if \_\_info (setq \_info \_\_info)) \_info)

(:collision1\_name

(&optional \_\_collision1\_name)

(if \_\_collision1\_name (setq \_collision1\_name \_\_collision1\_name)) \_collision1\_name)

(:collision2\_name

(&optional \_\_collision2\_name)

(if \_\_collision2\_name (setq \_collision2\_name \_\_collision2\_name)) \_collision2\_name)

(:wrenches

(&rest \_\_wrenches)

(if (keywordp (car \_\_wrenches))

(send\* \_wrenches \_\_wrenches)

(progn

(if \_\_wrenches (setq \_wrenches (car \_\_wrenches)))

\_wrenches)))

(:total\_wrench

(&rest \_\_total\_wrench)

(if (keywordp (car \_\_total\_wrench))

(send\* \_total\_wrench \_\_total\_wrench)

(progn

(if \_\_total\_wrench (setq \_total\_wrench (car \_\_total\_wrench)))

\_total\_wrench)))

(:contact\_positions

(&rest \_\_contact\_positions)

(if (keywordp (car \_\_contact\_positions))

(send\* \_contact\_positions \_\_contact\_positions)

(progn

(if \_\_contact\_positions (setq \_contact\_positions (car \_\_contact\_positions)))

\_contact\_positions)))

(:contact\_normals

(&rest \_\_contact\_normals)

(if (keywordp (car \_\_contact\_normals))

(send\* \_contact\_normals \_\_contact\_normals)

(progn

(if \_\_contact\_normals (setq \_contact\_normals (car \_\_contact\_normals)))

\_contact\_normals)))

(:depths

(&optional \_\_depths)

(if \_\_depths (setq \_depths \_\_depths)) \_depths)

(:serialization-length

()

(+

;; string \_info

4 (length \_info)

;; string \_collision1\_name

4 (length \_collision1\_name)

;; string \_collision2\_name

4 (length \_collision2\_name)

;; geometry\_msgs/Wrench[] \_wrenches

(apply #'+ (send-all \_wrenches :serialization-length)) 4

;; geometry\_msgs/Wrench \_total\_wrench

(send \_total\_wrench :serialization-length)

;; geometry\_msgs/Vector3[] \_contact\_positions

(apply #'+ (send-all \_contact\_positions :serialization-length)) 4

;; geometry\_msgs/Vector3[] \_contact\_normals

(apply #'+ (send-all \_contact\_normals :serialization-length)) 4

;; float64[] \_depths

(\* 8 (length \_depths)) 4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

(make-string-output-stream (send self :serialization-length)))))

;; string \_info

(write-long (length \_info) s) (princ \_info s)

;; string \_collision1\_name

(write-long (length \_collision1\_name) s) (princ \_collision1\_name s)

;; string \_collision2\_name

(write-long (length \_collision2\_name) s) (princ \_collision2\_name s)

;; geometry\_msgs/Wrench[] \_wrenches

(write-long (length \_wrenches) s)

(dolist (elem \_wrenches)

(send elem :serialize s)

)

;; geometry\_msgs/Wrench \_total\_wrench

(send \_total\_wrench :serialize s)

;; geometry\_msgs/Vector3[] \_contact\_positions

(write-long (length \_contact\_positions) s)

(dolist (elem \_contact\_positions)

(send elem :serialize s)

)

;; geometry\_msgs/Vector3[] \_contact\_normals

(write-long (length \_contact\_normals) s)

(dolist (elem \_contact\_normals)

(send elem :serialize s)

)

;; float64[] \_depths

(write-long (length \_depths) s)

(dotimes (i (length \_depths))

(sys::poke (elt \_depths i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;;

(if (null strm) (get-output-stream-string s))))

(:deserialize

(buf &optional (ptr- 0))

;; string \_info

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_info (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;; string \_collision1\_name

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_collision1\_name (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;; string \_collision2\_name

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_collision2\_name (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;; geometry\_msgs/Wrench[] \_wrenches

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_wrenches (let (r) (dotimes (i n) (push (instance geometry\_msgs::Wrench :init) r)) r))

(dolist (elem- \_wrenches)

(send elem- :deserialize buf ptr-) (incf ptr- (send elem- :serialization-length))

))

;; geometry\_msgs/Wrench \_total\_wrench

(send \_total\_wrench :deserialize buf ptr-) (incf ptr- (send \_total\_wrench :serialization-length))

;; geometry\_msgs/Vector3[] \_contact\_positions

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_contact\_positions (let (r) (dotimes (i n) (push (instance geometry\_msgs::Vector3 :init) r)) r))

(dolist (elem- \_contact\_positions)

(send elem- :deserialize buf ptr-) (incf ptr- (send elem- :serialization-length))

))

;; geometry\_msgs/Vector3[] \_contact\_normals

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_contact\_normals (let (r) (dotimes (i n) (push (instance geometry\_msgs::Vector3 :init) r)) r))

(dolist (elem- \_contact\_normals)

(send elem- :deserialize buf ptr-) (incf ptr- (send elem- :serialization-length))

))

;; float64[] \_depths

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_depths (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_depths i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;;

self)

)

(setf (get gazebo\_msgs\_new::ContactState :md5sum-) "48c0ffb054b8c444f870cecea1ee50d9")

(setf (get gazebo\_msgs\_new::ContactState :datatype-) "gazebo\_msgs\_new/ContactState")

(setf (get gazebo\_msgs\_new::ContactState :definition-)

"string info # text info on this contact

string collision1\_name # name of contact collision1

string collision2\_name # name of contact collision2

geometry\_msgs/Wrench[] wrenches # list of forces/torques

geometry\_msgs/Wrench total\_wrench # sum of forces/torques in every DOF

geometry\_msgs/Vector3[] contact\_positions # list of contact position

geometry\_msgs/Vector3[] contact\_normals # list of contact normals

float64[] depths # list of penetration depths

================================================================================

MSG: geometry\_msgs/Wrench

# This represents force in free space, separated into

# its linear and angular parts.

Vector3 force

Vector3 torque

================================================================================

MSG: geometry\_msgs/Vector3

# This represents a vector in free space.

# It is only meant to represent a direction. Therefore, it does not

# make sense to apply a translation to it (e.g., when applying a

# generic rigid transformation to a Vector3, tf2 will only apply the

# rotation). If you want your data to be translatable too, use the

# geometry\_msgs/Point message instead.

float64 x

float64 y

float64 z

")

(provide :gazebo\_msgs\_new/ContactState "48c0ffb054b8c444f870cecea1ee50d9")