;; Auto-generated. Do not edit!

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

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

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

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

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

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

(in-package "ROS")

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

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

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

(defclass gazebo\_msgs\_new::ModelStates

:super ros::object

:slots (\_name \_pose \_twist ))

(defmethod gazebo\_msgs\_new::ModelStates

(:init

(&key

((:name \_\_name) (let (r) (dotimes (i 0) (push "" r)) r))

((:pose \_\_pose) (let (r) (dotimes (i 0) (push (instance geometry\_msgs::Pose :init) r)) r))

((:twist \_\_twist) (let (r) (dotimes (i 0) (push (instance geometry\_msgs::Twist :init) r)) r))

)

(send-super :init)

(setq \_name \_\_name)

(setq \_pose \_\_pose)

(setq \_twist \_\_twist)

self)

(:name

(&optional \_\_name)

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

(:pose

(&rest \_\_pose)

(if (keywordp (car \_\_pose))

(send\* \_pose \_\_pose)

(progn

(if \_\_pose (setq \_pose (car \_\_pose)))

\_pose)))

(:twist

(&rest \_\_twist)

(if (keywordp (car \_\_twist))

(send\* \_twist \_\_twist)

(progn

(if \_\_twist (setq \_twist (car \_\_twist)))

\_twist)))

(:serialization-length

()

(+

;; string[] \_name

(apply #'+ (mapcar #'(lambda (x) (+ 4 (length x))) \_name)) 4

;; geometry\_msgs/Pose[] \_pose

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

;; geometry\_msgs/Twist[] \_twist

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

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; string[] \_name

(write-long (length \_name) s)

(dolist (elem \_name)

(write-long (length elem) s) (princ elem s)

)

;; geometry\_msgs/Pose[] \_pose

(write-long (length \_pose) s)

(dolist (elem \_pose)

(send elem :serialize s)

)

;; geometry\_msgs/Twist[] \_twist

(write-long (length \_twist) s)

(dolist (elem \_twist)

(send elem :serialize s)

)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; string[] \_name

(let (n)

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

(setq \_name (make-list n))

(dotimes (i n)

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

))

;; geometry\_msgs/Pose[] \_pose

(let (n)

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

(setq \_pose (let (r) (dotimes (i n) (push (instance geometry\_msgs::Pose :init) r)) r))

(dolist (elem- \_pose)

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

))

;; geometry\_msgs/Twist[] \_twist

(let (n)

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

(setq \_twist (let (r) (dotimes (i n) (push (instance geometry\_msgs::Twist :init) r)) r))

(dolist (elem- \_twist)

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

))

;;

self)

)

(setf (get gazebo\_msgs\_new::ModelStates :md5sum-) "48c080191eb15c41858319b4d8a609c2")

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

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

"# broadcast all model states in world frame

string[] name # model names

geometry\_msgs/Pose[] pose # desired pose in world frame

geometry\_msgs/Twist[] twist # desired twist in world frame

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

MSG: geometry\_msgs/Pose

# A representation of pose in free space, composed of position and orientation.

Point position

Quaternion orientation

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

MSG: geometry\_msgs/Point

# This contains the position of a point in free space

float64 x

float64 y

float64 z

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

MSG: geometry\_msgs/Quaternion

# This represents an orientation in free space in quaternion form.

float64 x

float64 y

float64 z

float64 w

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

MSG: geometry\_msgs/Twist

# This expresses velocity in free space broken into its linear and angular parts.

Vector3 linear

Vector3 angular

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

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/ModelStates "48c080191eb15c41858319b4d8a609c2")