Pygazebo

Class Statement

- **Global Function**
  - Initialize(const std::vector<std::string>& args, int port = 0)
    - initialize the env
    - default: std::vector<std::string>()
    - **py: initialize()**
  - StartSensors()
  - NewWorldFromFile(const std::string& world_file)
    - load world file and start sensors
    - return a pointer point to the world
    - return type: unique_ptr<World>
    - **py: new_world_from_file(world_file)**
- **Class CameraObservation**
  - **parameter**
    - width_ height_ depth_: image memory feature
    - data: image data
  - **relate function**
    - GetCameraObservation(const std::string& sensor_scope_name)
      - bind with Class Agent
      - it: map<sensor_scope_name, sensor>
      - ret: return of insert function
        - first: map<sensor_scope_name, sensor>
        - second: FLAG
      - sensor: CameraSensorPtr
      - camera: CameraPtr
      - return type: **Class CameraObservation**
      - **py: get_camera_observation(sensor_scope_name)**
    - buffer_info
      - {m.height(), m.width(), m.depth()}: shape of image
      - {sizeof(uint8_t) * m.width() * m.depth(), sizeof(uint8_t) * m.depth(), sizeof(uint8_t)}: strides of every level

- ## Class RayObservation
  - **parameter**
    - angle_min_ angle_max_ angle_resolution_: scan angle feature of lidar
    - range_min_ range_max_ range_resolution_: scan distance feature of lidar
    - num_: the number of scan points
    - data_: lidar's scan data-distance
  - **relate function**
    - GetRayObservation(const std::string& sensor_scope_name)
      - bind with Class Agent
      - it: map<sensor_scope_name, sensor>
      - ret: return of insert function
        - first: map<sensor_scope_name, sensor>
        - second: FLAG
      - result: get lidar scan data through function Ranges(result)
      - data_: copy data from result, a pointer
      - return type: **Class RayObservation**
      - **py: get_ray_observation(sensor_scope_name)**
    - buffer_info
      - {m.num()}: the number of scan points
      - double: data type
- ## Class ContactSensor
  - contact sensor detects collisions between two objects and reports the location of the contact associated forces
  - **relate function**
    - GetCollisionCount(): get the number of collisions that the sensor is observing
      - bind with Class Agent
      - **py: get_collision_count()**
- ## Class JointState
  - **parameter**
    - unsigned int dof_: degree of freedom
    - std::vector<double> positions_
    - std::vector<double> velocities_
  - **relate function**
    - GetDOF()
    - GetPositions()
      - return joint's positions
    - GetVelocities()
      - return joint's velocities
    - SetVelocities(const std::vector<double>& v)
    - SetPositions(const std::vector<double>& pos)

- ## Class Model
  - **parameter**
    - Pose
      - std::tuple<std::tuple<double, double, double>, std::tuple<double, double, double>>
    - Twist
      - std::tuple<std::tuple<double, double, double>, std::tuple<double, double, double>>
  - **relate function**
    - GetPose()
      - return robot's position
      - data: ((x,y,z), (roll, pitch, yaw))
      - **py: get_pose()**
    - GetTwist()
      - return robot's twist
      - data: ((linear.X, linear.Y, linear.Z),(angular.X, angular.Y, angular.Z))
      - **py: get_twist()**
    - SetPose(const Pose& pose)
      - set robot's position
      - param: ((x,y,z), (roll, pitch, yaw))
      - **py: set_pose()**
    - SetTwist(const Twist& twist)
      - set robot's twist
      - param: ((linear.X, linear.Y, linear.Z),(angular.X, angular.Y, angular.Z))
      - **py: set_twist()**
- ## Class Agent
  - **parameter**
    - cameras_ rays_: std::map<std::string, gazebo::sensors::XXXSensorPtr> link sensor name with sensor object pointer
    - contacts_: std::map<std::string, gazebo::sensors::ContactSensorPtr> link contact sensor name with contact sensor object pointer
  - **relate function**
    - GetJointNames()
      - return agent's joints names and print agent's joints name
      - return type: std::vector<std::string>
      - **py:get_joint_names()**
    - GetJointState(const std::string& joint_name)
      - return joint's velocities and positions state value
      - return type: **Class JointState**
      - **py: get_joint_state()**
    - GetLinkPose(const std::string& link_name)
      - return link which name is link_name state
    - SetJointState(const std::string& joint_name, const JointState& joint_state)
      - set joint which name is joint_name state
      - **py: set_joint_state**
    - GetCollisionCount(const std::string& contact_sensor_name)
      - statement in **Class ContactSensor**
    - GetCameraObservation(const std::string& sensor_scope_name)
      - return data get from camera

        - statement in **Class CameraObservation**
    - GetRayObservation(const std::string& sensor_scope_name)
        - return data get from lidar
        - statement in **Class RayObservation**
    - TakeAction(const std::map<std::string, double>& forces)
        - set robot's controller based on force
        - if failure return false
        - **py: take_action()**
    - Reset()
        - reset the robot state
        - **py: reset()**
- Class World
    - parameter
        - gazebo::physics::WorldPtr world_: a pointer, point to a gazebo world
    - relate function
        - GetAgents(const std::string& name)
            - **input param can be empty**
            - return the models whose joint count > 1
            - return type: vector<std::unique_ptr<Agent>>
            - **py: get_agents()**
        - GetModel(const std::string& name)
            - return model
            - return type: unique_ptr<Model>
            - **py: get_model()**
        - Step(int num_steps)
            - default num_steps: 1
            - robot run steps in the gazebo world
            - **py: step()**
        - Reset()
            - reset the world
            - **py: reset()**
        - GetSimTime()
            - **py: get_sim_time()**
        - GetWallTime()
            - **py:get_wall_time()**