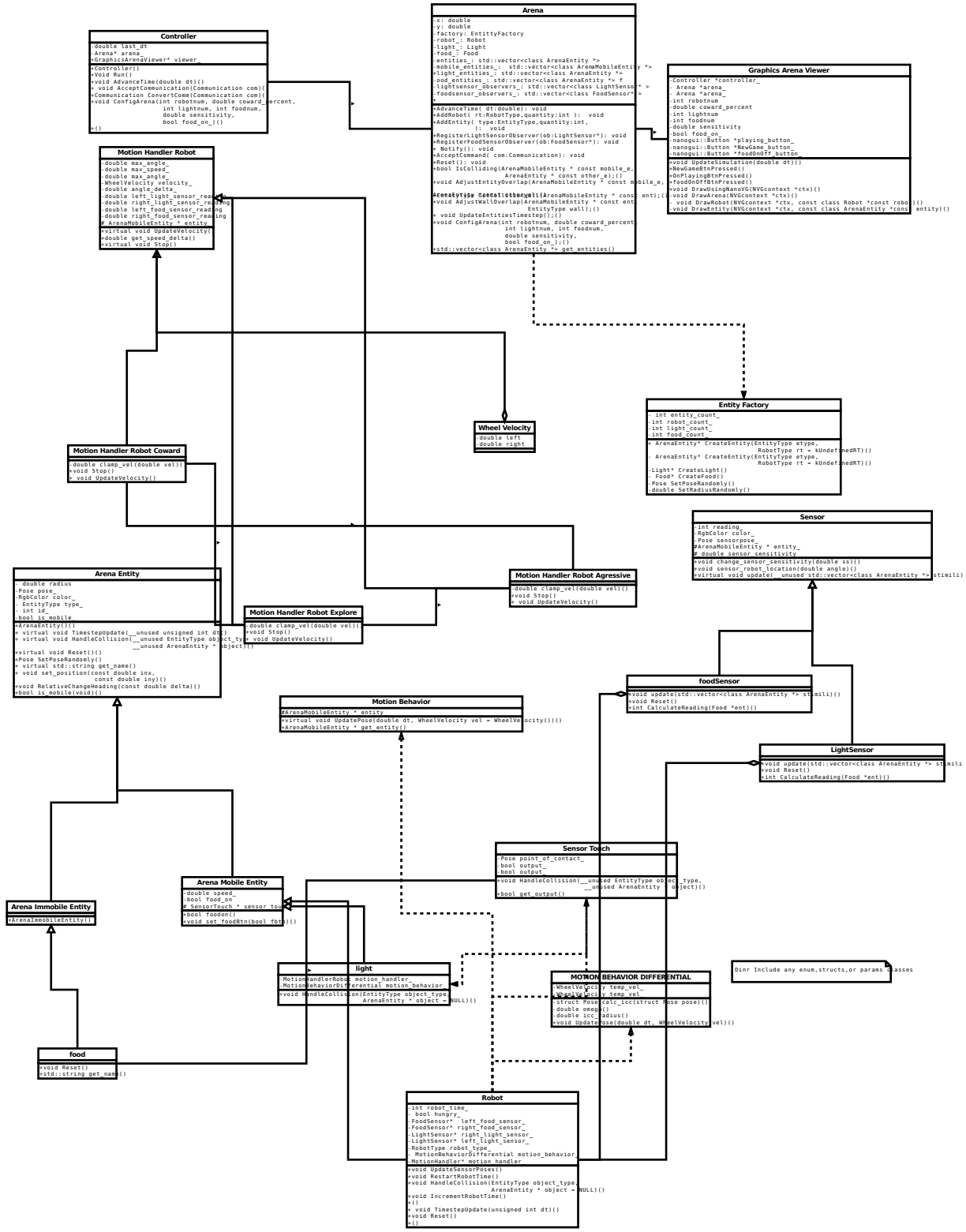
Controller
-double last_dt
-Arena* arena_
+GraphicsArenaViewer* viewer_
+Controller()
+Void Run()
+void AdvanceTime(double dt)()
+ void AcceptCommunication(Communication com)()
+Communication ConvertComm(Communication com)()
+void ConfigArena(int robotnum, double coward_percent,
int lightnum, int foodnum,
double sensitivity,
bool food_on_)()
+()
Arena
-x: double
-y: double
-factory: EntittyFactory
-robot_: Robot
-light_: Light
-food_: Food
-entities_: std::vector<class ArenaEntity *>
-mobile_entities_: std::vector<class ArenaMobileEntity *>
+light_entities_: std::vector<class ArenaEntity *>
-ood_entities_: std::vector<class ArenaEntity *> f
-lightsensor_observers_: std::vector<class LightSensor* >
-foodsensor_observers_: std::vector<class FoodSensor* >
+
+AdvanceTime(dt:double): void
+AddRobot(rt:RobotType,quantity:int): void
+AddEntity(type:EntityType,quantity:int,
): void
+RegisterLightSensorObserver(ob:LightSensor*): void
+RegisterFoodSensorObserver(ob:FoodSensor*): void
+ Notify(): void
+AcceptCommand(com:Communication): void
+Reset(): void
+bool IsColliding(ArenaMobileEntity * const mobile_e,
ArenaEntity * const other_e);()
+void AdjustEntityOverlap(ArenaMobileEntity * const mobile_e,
ArenaEntity * const other_e);()
+void AdjustWallOverlap(ArenaMobileEntity * const ent);()
EntityType wall);()
+ void UpdateEntitiesTimestep();()
+void ConfigArena(int robotnum, double coward_percent,
int lightnum, int foodnum,
double sensitivity,
bool food_on_);()
+std::vector<class ArenaEntity *> get_entities()
Graphics Arena Viewer
-Controller *controller_
- Arena *arena_
- Arena *arena_
-int robotnum
-double coward_percent
-int lightnum
-int foodnum
-double sensitivity
-bool food_on_
-nanogui::Button *playing_button_
-nanogui::Button *NewGame_button_
-nanogui::Button *foodOnOff_button_
+void UpdateSimulation(double dt)()
+NewGameBtnPressed()
+OnPlayingBtnPressed()
+foodOnOffBtnPressed()
+void DrawUsingNanoVG(NVGcontext *ctx)()
+void DrawArena(NVGcontext *ctx)()
- void DrawRobot(NVGcontext *ctx, const class Robot *const robot)()
+void DrawEntity(NVGcontext *ctx, const class ArenaEntity *const entity)()
Motion Handler Robot
-double max_angle_
-double max_speed_
-double max_angle_
-WheelVelocity velocity_
-double angle_delta_
-double left_light_sensor_reading
-double right_light_sensor_reading
-double left_food_sensor_reading
-double right_food_sensor_reading
ArenaMobileEntity * entity_
+virtual void UpdateVelocity()
+double get_speed_delta()
+virtual void Stop()
Wheel Velocity
-double left
-double right
Entity Factory
-int entity_count_
-int robot_count_
-int light_count_
-int food_count_
+ ArenaEntity* CreateEntity(EntityType etype,
RobotType rt = kUndefinedRT)()
- ArenaEntity* CreateEntity(EntityType etype,
RobotType rt = kUndefinedRT)()
-Light* CreateLight()
-Food* CreateFood()
-Pose SetPoseRandomly()
-double SetRadiusRandomly()
Motion Handler Robot Coward
-double clamp_vel(double vel)()
+void Stop()
+ void UpdateVelocity()
Sensor
-int reading_
-RgbColor color_
-Pose sensorpose_
#ArenaMobileEntity * entity_
double sensor_sensitivity_
+void change_sensor_sensitivity(double ss)()
+void sensor_robot_location(double angle)()
+virtual void update(__unused std::vector<class ArenaEntity *> stimili)
Arena Entity
-double radius
-Pose pose_
-RgbColor color_
-EntityType type_
- int id_
-bool is_mobile
+ArenaEntity()()
+ virtual void TimestepUpdate(__unused unsigned int dt)
+ virtual void HandleCollision(__unused EntityType object_type,
__unused ArenaEntity * object)()
+virtual void Reset()()
+Pose SetPoseRandomly()
+ virtual std::string get_name()
+ void set_position(const double inx,
const double iny)()
+void RelativeChangeHeading(const double delta)()
+bool is_mobile(void)()
Motion Handler Robot Explore
-double clamp_vel(double vel)()
+void Stop()
+ void UpdateVelocity()
Motion Handler Robot Agressive
-double clamp_vel(double vel)()
+void Stop()
+ void UpdateVelocity()
foodSensor
+void update(std::vector<class ArenaEntity *> stimili)()
+void Reset()
+int CalculateReading(Food *ent)()
Motion Behavior
#ArenaMobileEntity * entity
+virtual void UpdatePose(double dt, WheelVelocity vel = WheelVelocity())()
+ArenaMobileEntity * get_entity()
LightSensor
+void update(std::vector<class ArenaEntity *> stimili
+void Reset()
+int CalculateReading(Food *ent)()
Sensor Touch
-Pose point_of_contact_
-bool output_
-bool output_
+void HandleCollision(__unused EntityType object_type,
__unused ArenaEntity * object)()
+bool get_output()
Arena Mobile Entity
-double speed_
-bool food_on
SensorTouch * sensor_touch
+bool foodon()
+void set_foodBtn(bool fbtn)()
Arena Immobile Entity
+ArenaImmobileEntity()()
light
-MotionHandlerRobot motion_handler
-MotionBehaviorDifferential motion_behavior
+void HandleCollision(EntityType object_type,
ArenaEntity * object = NULL)()
MOTION BEHAVIOR DIFFERENTIAL
-WheelVelocity temp_vel_
-WheelVelocity temp_vel
+struct Pose calc_icc(struct Pose pose)()
-double omega()
-double icc_radius()
+void UpdatePose(double dt, WheelVelocity vel)()
Dinr Include any enum,structs,or params classes
food
+void Reset()
+std::string get_name()
Robot
-int robot_time_
- bool hungry_
-FoodSensor* left_food_sensor_
-FoodSensor* right_food_sensor_
-LightSensor* right_light_sensor_
-LightSensor* left_light_sensor_
-RobotType robot_type_
- MotionBehaviorDifferential motion_behavior_
-MotionHandler* motion_handler
+void UpdateSensorPoses()
+void RestartRobotTime()
+void HandleCollision(EntityType object_type,
ArenaEntity * object = NULL)()
+void IncrementRobotTime()
+()
+ void TimestepUpdate(unsigned int dt)()
+void Reset()
+()