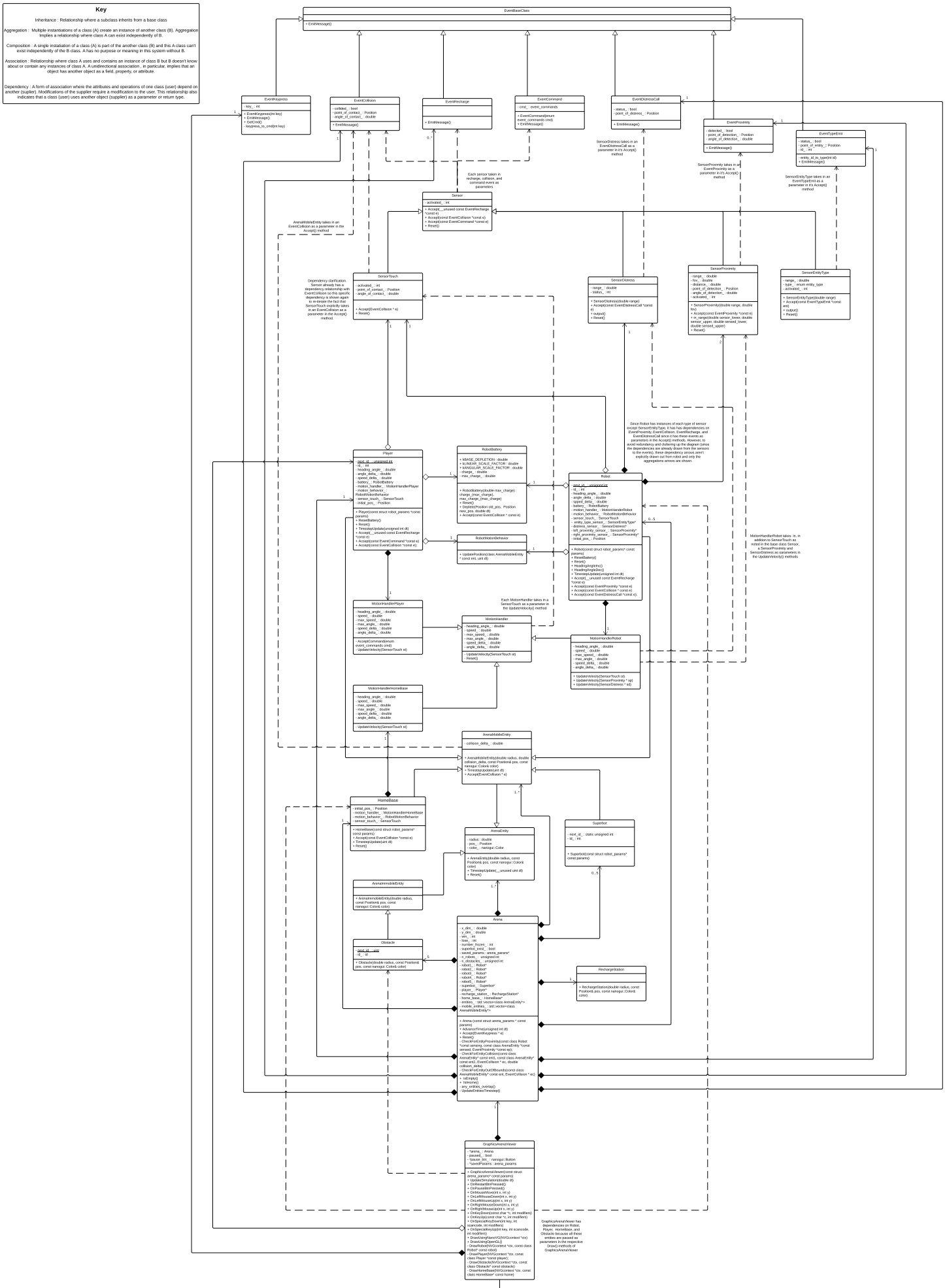
Key
Inheritance : Relationship where a subclass inherits from a base class
Aggregation : Multiple instantiations of a class (A) create an instance of another class (B). Aggregation Implies a relationship where class A can exist independently of B.
Composition : A single instatiation of a class (A) is part of the another class (B) and this A class can't exist independently of the B class. A has no purpose or meaning in this system without B.
Association : Relationship where class A uses and contains an instance of class B but B doesn't know about or contain any instances of class A. A unidirectional association , in particular, implies that an object has another object as a field, property, or attribute.
Dependency : A form of association where the attributes and operations of one class (user) depend on another (supler). Modifications of the supplier require a modification to the user. This relationship also indicates that a class (user) uses another object (supplier) as a parameter or return type.
EventBaseClass
+ EmitMessage()
EventExpress
- key_ : int
+ EventExpress(int key)
+ EmitMessage()
+ GetCmd()
- keypress_to_cmd(int key)
EventCollision
- collided_ : bool
- point_of_contact_ : Position
- angle_of_contact_ : double
+ EmitMessage()
EventRecharge
+ EmitMessage()
EventCommand
- cmd_ : event_commands
+ EventCommand(enum event_commands cmd)
+ EmitMessage()
EventDistressCall
- status_ : bool
- point_of_distress_ : Position
+ EmitMessage()
EventProximity
- detected_ : bool
- point_of_detection_ : Position
angle_of_detection_ : double
+ EmitMessage()
EventTypeEmit
- status_ : bool
- point_of_entity_ : Position
- id_ : int
- entity_id_to_type(int id)
+ EmitMessage()
SensorDistress takes in an EventDistressCall as a parameter in it's Accept() method
SensorProximity takes in an EventProximity as a parameter in ITS Accept() method
SensorEntityType takes in an EventTypeEmit as a parameter in it's Accept() method
Each sensor taken in recharge, collision, and command event as parameters
Sensor
- activated_ : int
+ Accept(__unused const EventRecharge *const e)
+ Accept(const EventCollision *const e)
+ Accept(const EventCommand *const e)
+ Reset()
ArenaMobileEntity takes in an EventCollision as a parameter in the Accept() method
SensorTouch
- activated_ : int
- point_of_contact_ : Position
- angle_of_contact_ : double
+ Accept(EventCollision * e)
+ Reset()
Dependency clarification. Sensor already has a dependency relationship with EventCollision so this specific dependency is shown again to re-iterate the fact that SensorTouch explicitly takes in an EventCollision as a parameter in the Accept() method.
SensorDistress
- range_ : double
- status_ : int
+ SensorDistress(double range)
+ Accept(const EventDistressCall *const e)
+ output()
+ Reset()
SensorProximity
- range_ : double
- fov_ : Double
- distance_ : double
- point_of_detection_ : Position
- angle_of_detection_ : double
- activated_ : int
+ SensorProximity(double range, double fov)
+ Accept(const EventProximity *const e)
+ in_range(double sensor_lower, double sensor_upper, double sensed_lower, double sensed_upper)
+ Reset()
SensorEntityType
- range_ : double
- type_ : enum entity_type
- activated_ : int
+ SensorEntityType(double range)
+ Accept(const EventTypeEmit *const e)
+ output()
+ Reset()
Since Robot has instances of each type of sensor except SensorEntityType, it has has dependences on EventProximity, EventCollision, EventRecharge, and EventDistressCall since it has these events as parameters in its Accept() methods. In order to avoid redundancy and cluttering up the diagram (since the dependencies are already drawn from the sensors to the events), these dependency arrows aren't explicitly drawn out from robot and only the aggregations arrows are shown.
Player
- next_id_ : unsigned int
- id_ : int
- heading_angle_ : double
- angle_delta_ : double
- speed_delta_ : double
- battery_ : RobotBattery
- motion_handler_ : MotionHandlerPlayer
- motion_behavior_ : RobotMotionBehavior
- sensor_touch_ : SensorTouch
- initial_pos_ : Position
+ Player(const struct robot_params *const params)
+ ResetBattery()
+ Reset()
+ TimestepUpdate(unsigned int dt)
+ Accept(__unused const EventRecharge *const e)
+ Accept(const EventCommand *const e)
+ Accept(const EventCollision *const e)
RobotBattery
+ MBASE_DEPLETION : double
+ KLINEAR_SCALE_FACTOR : double
+ MANDLE_SCALE_FACTOR : double
- charge_ : double
- max_charge_ : double
+ RobotBattery(double max_charge)
charge_(max_charge),
max_charge_(max_charge)
+ Reset()
+ Deplete(Position old_pos, Position new_pos, double dt)
+ Accept(const EventCollision * const e)
Robot
- next_id_ : unsigned int
- id_ : int
- heading_angle_ : double
- angle_delta_ : double
- speed_delta_ : double
- battery_ : RobotBattery
- motion_handler_ : MotionHandlerRobot
- motion_behavior_ : RobotMotionBehavior
- sensor_touch_ : SensorTouch
- sensor_distress_ : SensorDistress*
- distress_sensor_ : SensorDistress*
- left_proximity_sensor_ : SensorProximity*
- right_proximity_sensor_ : SensorProximity*
- initial_pos_ : Position
+ Robot(const struct robot_params* const params)
+ ResetBattery()
+ Reset()
+ HeadingAngle()
+ HeadingAngleDec()
+ TimestepUpdate(unsigned int dt)
+ Accept(__unused const EventRecharge *const e)
+ Accept(const EventProximity *const e)
+ Accept(const EventCollision * const e)
+ Accept(const EventDistressCall *const e).
0..1
RobotMotionBehavior
+ UpdatePosition(class ArenaMobileEntity * const ent, uint dt)
MotionHandlerPlayer takes in in addition to SensorTouch as noted in the base class Sensor, a SensorProximity and SensorDistress as parameters in the UpdateVelocity() methods
MotionHandlerPlayer
- heading_angle_ : double
- speed_ : double
- max_speed_ : double
- max_angle_ : double
- speed_delta_ : double
- angle_delta_ : double
- AcceptCommand(enum event_commands cmd)
- UpdateVelocity(SensorTouch st)
Each MotionHandler takes in a SensorTouch as a parameter in the UpdateVelocity() method
MotionHandler
+ heading_angle_ : double
+ speed_ : double
+ max_speed_ : double
+ max_angle_ : double
+ speed_delta_ : double
+ angle_delta_ : double
+ UpdateVelocity(SensorTouch st)
+ Reset()
MotionHandlerRobot
+ heading_angle_ : double
+ speed_ : double
+ max_speed_ : double
+ max_angle_ : double
+ speed_delta_ : double
+ angle_delta_ : double
+ UpdateVelocity(SensorTouch st)
+ UpdateVelocity(SensorProximity *sp)
+ UpdateVelocity(SensorDistress * sd)
MotionHandlerHomeBase
+ heading_angle_ : double
+ speed_ : double
+ max_speed_ : double
+ max_angle_ : double
+ speed_delta_ : double
+ angle_delta_ : double
+ UpdateVelocity(SensorTouch st)
ArenaMobileEntity
- collision_delta_ : double
+ ArenaMobileEntity(double radius, double collision_delta, const Position& pos, const nanogui::Color& color)
+ Timestep(update(uint dt)
+ Accept(EventCollision * e)
1..*
HomeBase
- initial_pos_ : Position
- motion_handler_ : MotionHandlerHomeBase
- motion_behavior_ : RobotMotionBehavior
- sensor_touch_ : SensorTouch
+ HomeBase(const struct robot_params* const params)
+ Accept(const EventCollision *const e)
+ Timestep(update(uint dt)
+ Reset()
AvenaEntity
- radius : double
- pos_ : Position
- color_ : nanogui::Color
+ AvenaEntity(double radius, const Position& pos, const nanogui::Color& color)
+ Timestep(update(__unused uint dt)
+ Reset()
1..*
Superbot
- next_id_ : static unsigned int
- id_ : int
+ Superbot(const struct robot_params* const params)
0..3
AvenaImmobileEntity
+ AvenaImmobileEntity(double radius, const Position& pos, const nanogui::Color& color)
Obstacle
- next_id_ : static
- id_ : int
+ Obstacle(double radius, const Position& pos, const nanogui::Color& color)
5
Arena
- x_dim_ : double
- y_dim_ : double
- res_ : int
- tsau_ : int
- number_frozen_ : int
- superbot_exist_ : bool
- arena_params : const arena_params*
- n_robots_ : unsigned int
- n_obstacles_ : unsigned int
- robot1_ : Robot*
- robot2_ : Robot*
- robot3_ : Robot*
- robot4_ : Robot*
- robot5_ : Robot*
- superbot_ : Superbot*
- player_ : Player*
- recharge_station_ : RechargeStation*
- home_base_ : HomeBase*
- entities_ : std::vector<class AvenaEntity*>
- mobile_entities_ : std::vector<class AvenaMobileEntity*>
+ Arena (const struct arena_params * const params)
+ AdvanceTime(unsigned int dt)
+ Accept(EventKeypress * e)
+ Reset()
- CheckForEntityProximity(const class Robot *const sensing, const class AvenaEntity *const sensed, EventProximity *const ep)
- CheckForEntityCollision(const class AvenaMobileEntity* const ent1, const class AvenaEntity* const ent2, EventCollision * ec, double collision_delta)
- CheckForEntityOutOfBounds(const class AvenaMobileEntity* const ent, EventCollision * ec)
+ isEmpty()
+ IsNoise()
- any_entities_overlap()
- UpdateEntitiesTimestep()
RechargeStation
+ RechargeStation(double radius, const Position& pos, const nanogui::Color& color)
GraphicsArenaViewer
- *arena_ : Arena
- paused_ : bool
- *pause_btn_ : nanogui::Button
- *savedParams : arena_params
+ GraphicsArenaViewer(const struct arena_params* const params)
+ UpdateSimulation(double dt)
+ OnResetBtnPressed()
+ OnPauseBtnPressed()
+ OnMouseMove(int x, int y)
+ OnLeftMouseDown(int x, int y)
+ OnLeftMouseUp(int x, int y)
+ OnRightMouseDown(int x, int y)
+ OnRightMouseUp(int x, int y)
+ OnKeyDown(const char *c, int modifiers)
+ OnKeyUp(const char *c, int modifiers)
+ OnSpecialKeyDown(int key, int scancode, int modifiers)
+ OnSpecialKeyUp(int key, int scancode, int modifiers)
+ DrawUsingNanoVG(NVGcontext *ctx)
+ DrawUsingOpenGL()
+ DrawPlayer(NVGcontext *ctx, const class Player *const player)
+ DrawObstacle(NVGcontext *ctx, const class Obstacle* const obstacle)
+ DrawHomeBase(NVGcontext *ctx, const class HomeBase* const home)
GraphicsArenaViewer has dependencies on Robot, Player, HomeBase, and Obstacle because all these entities are passed as parameters in the respective Draw() methods of GraphicsArenaViewer
GraphicsApp