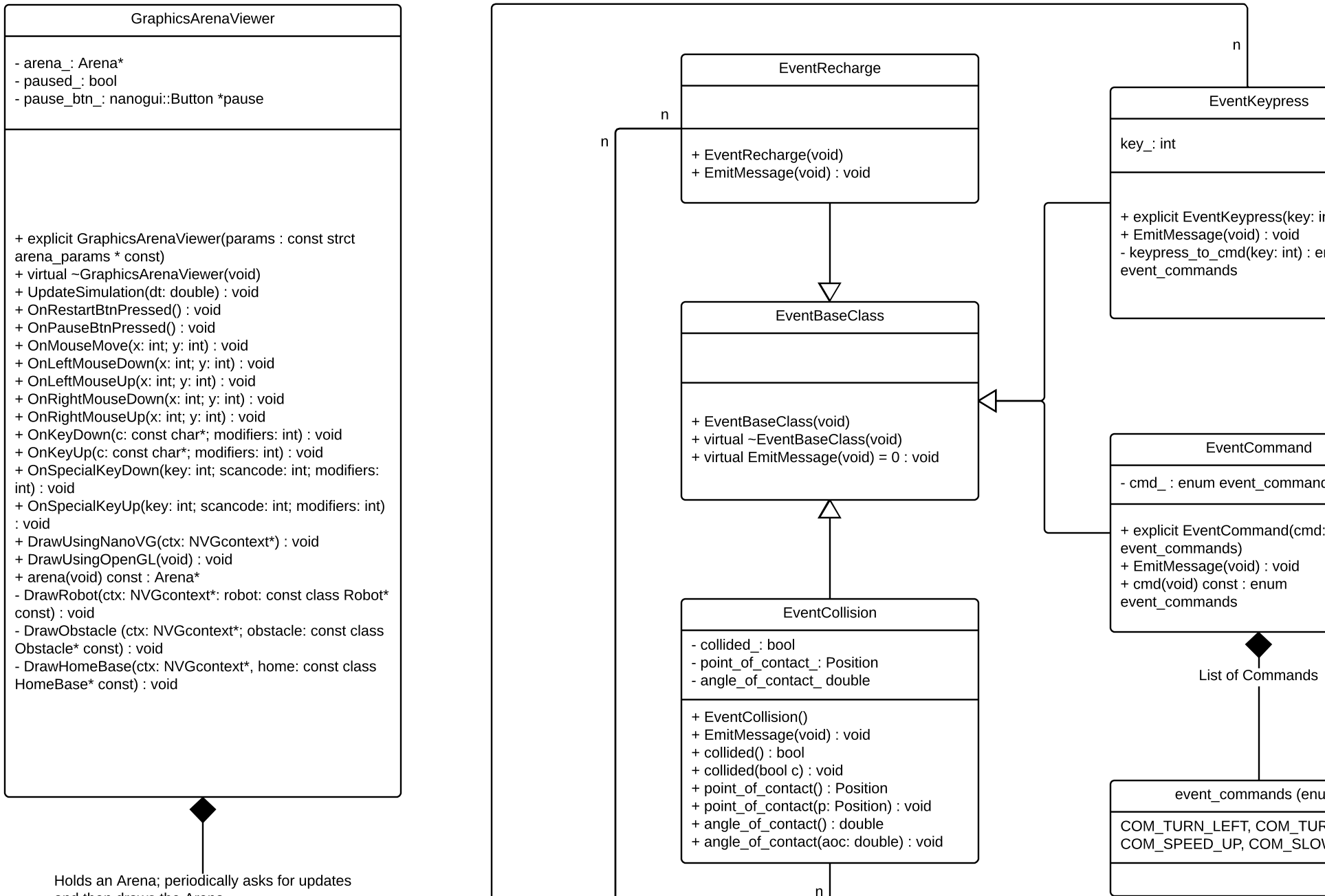
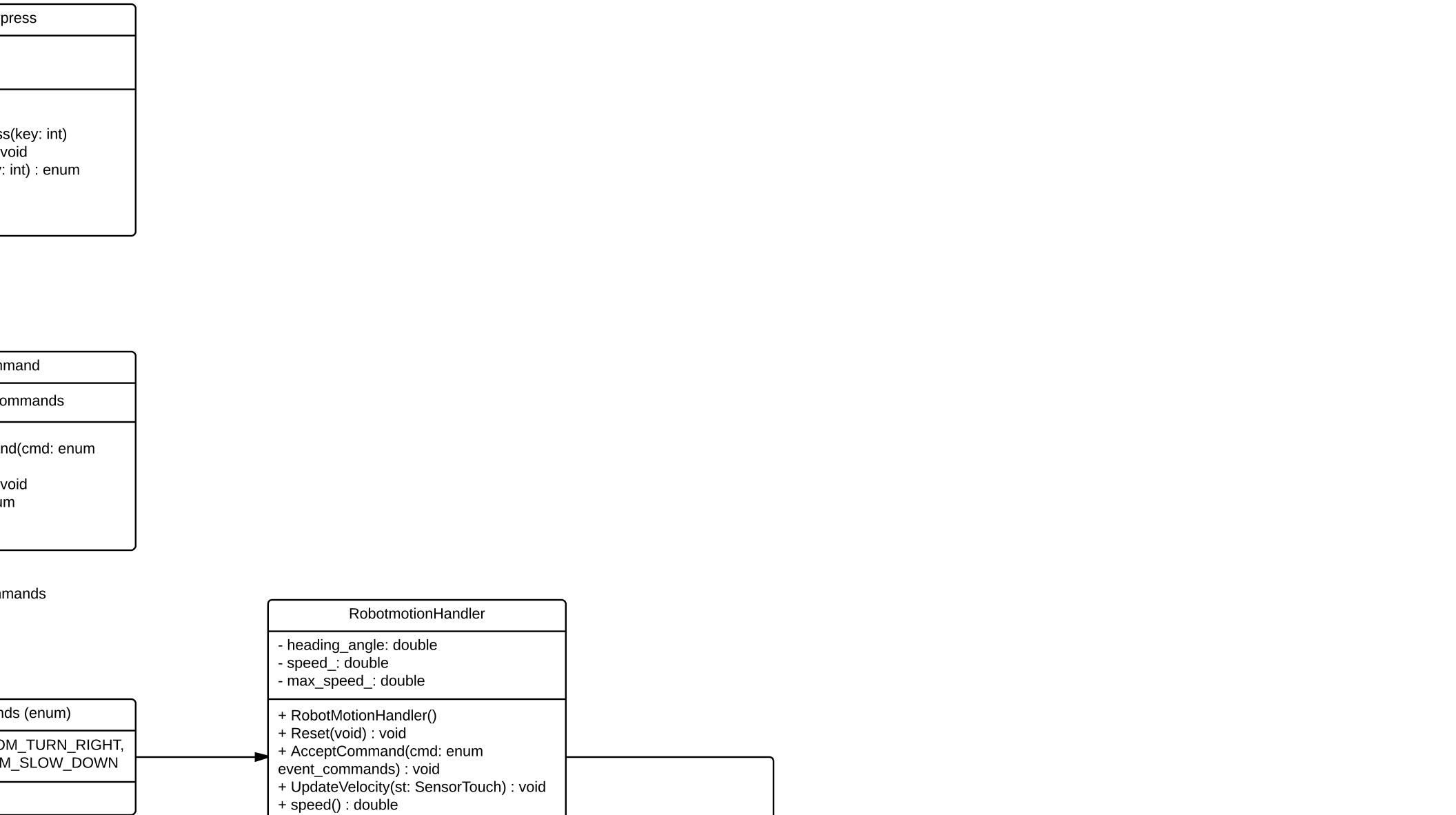
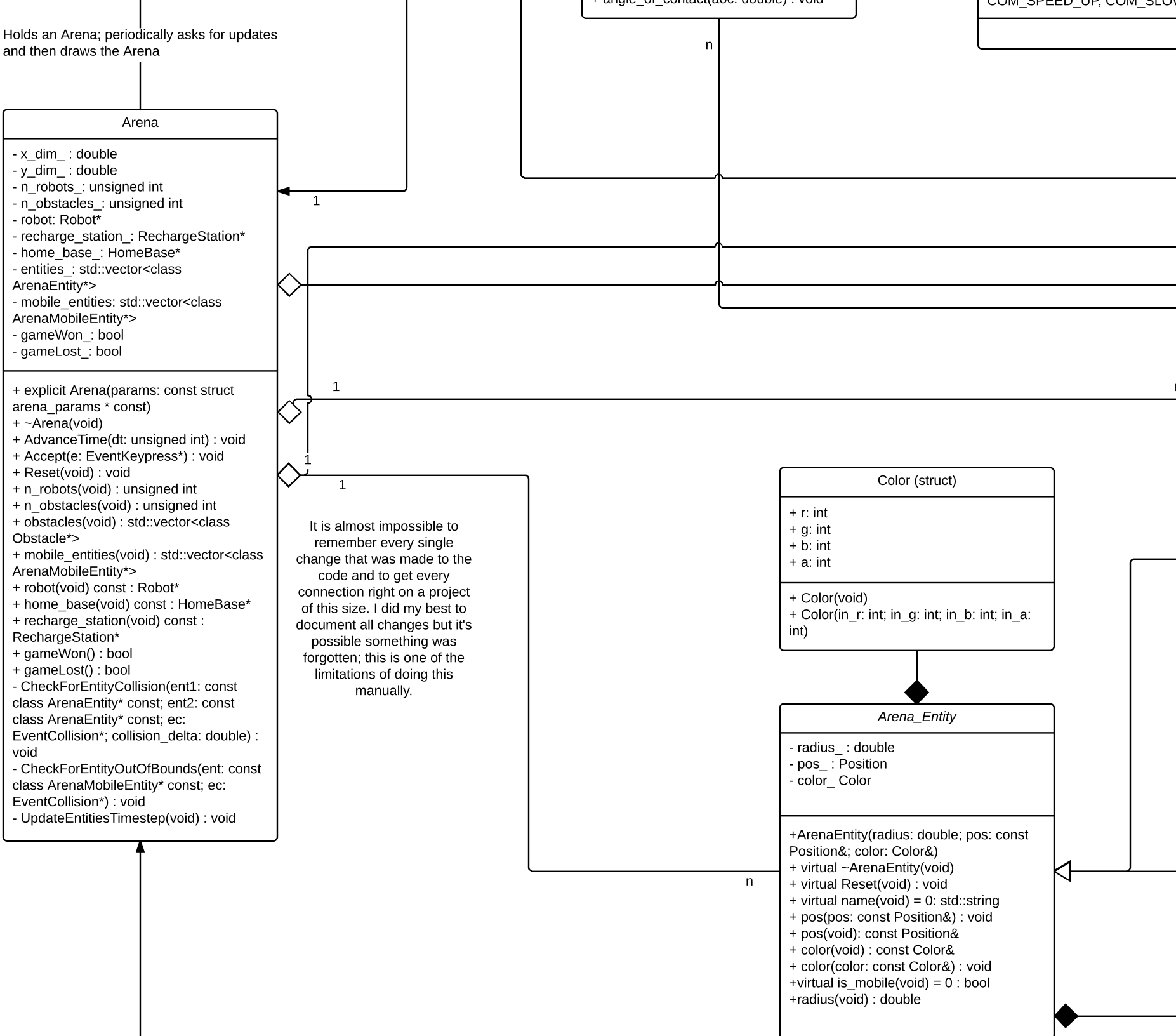
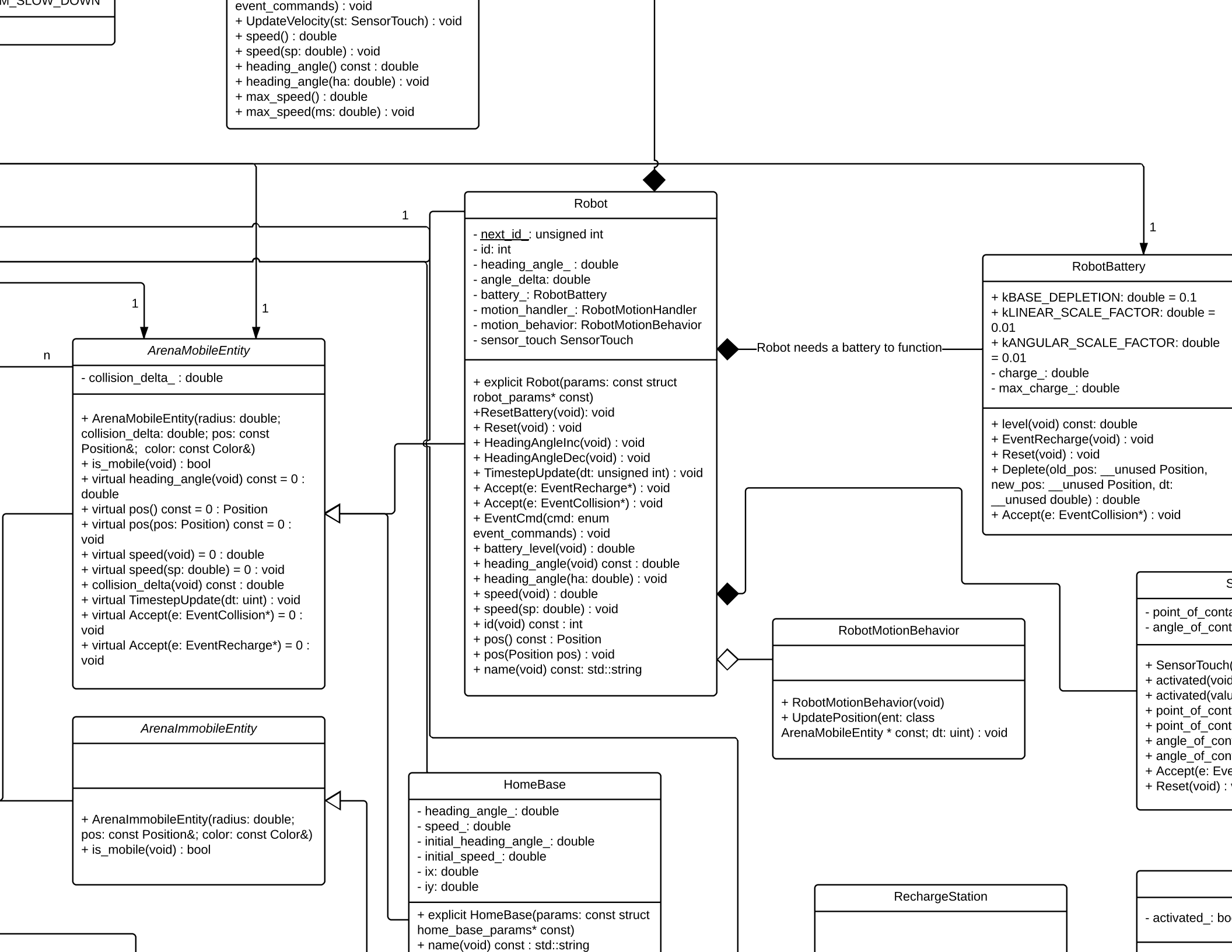
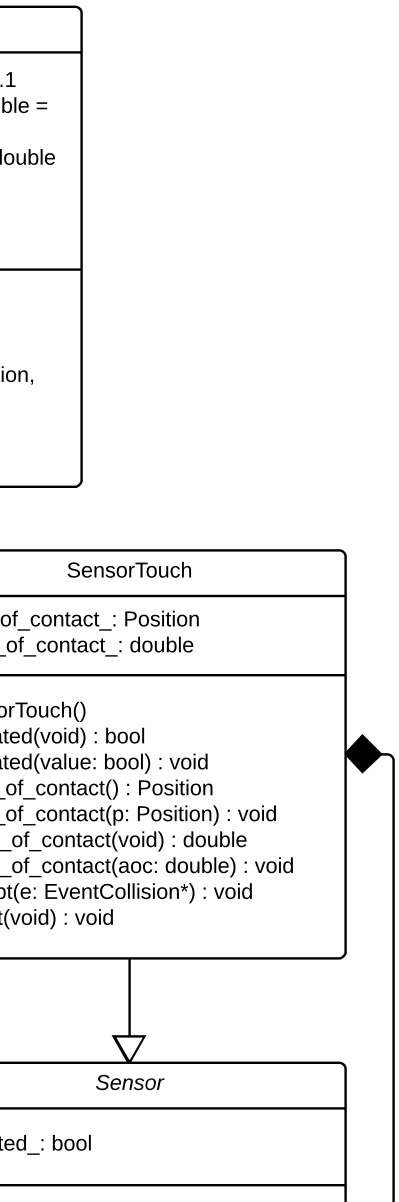
GraphicsArenaViewer inherits from GraphicsApp but that class isn't in the src folder

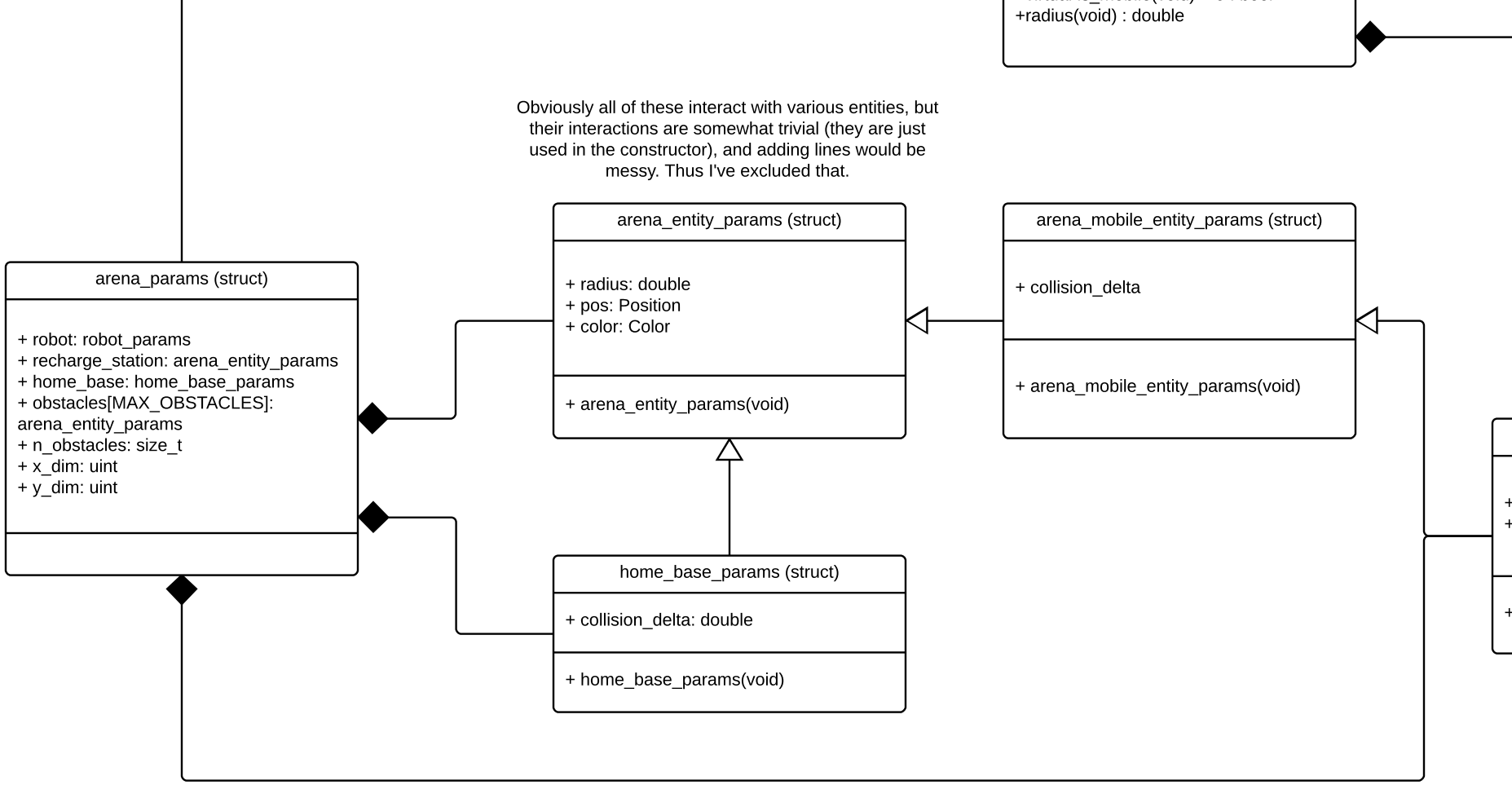


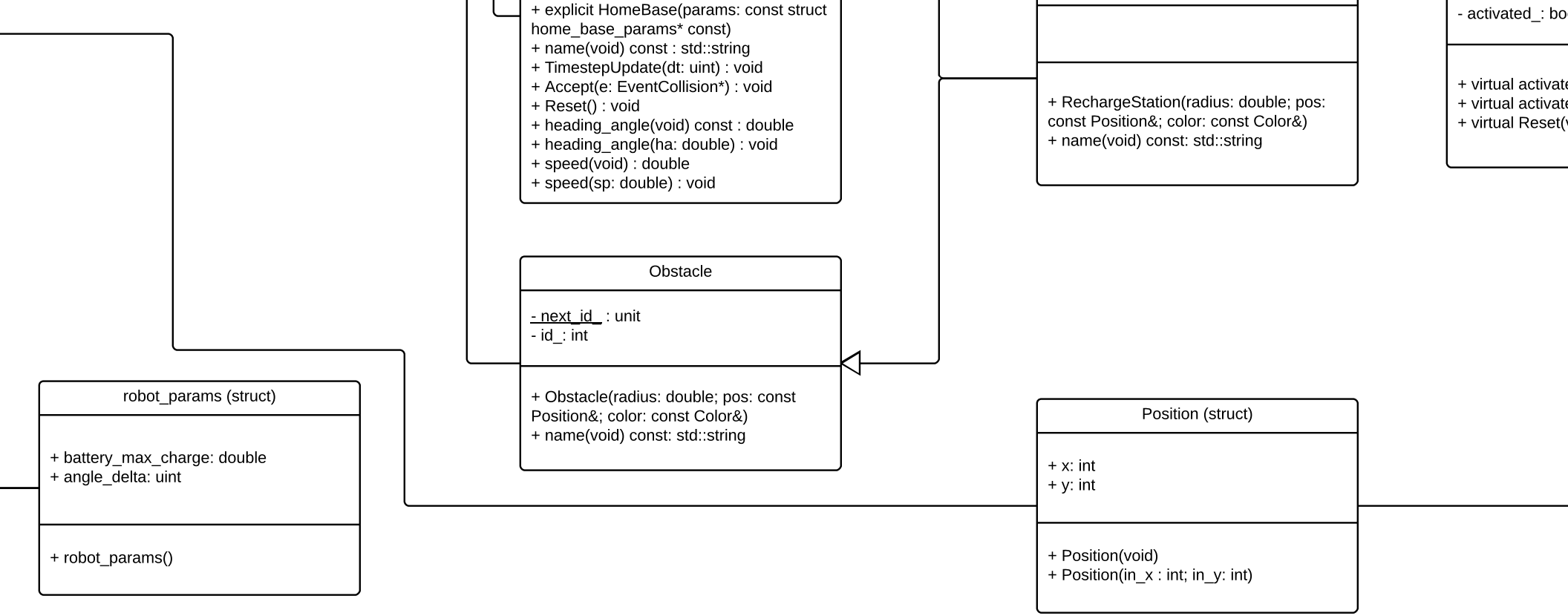

press
ss(key: int)
void
: int) : enum
mmand
ommands
nd(cmd: enum
void
um
mmands
nds (enum)
OM_TURN_RIGHT,
M_SLOW_DOWN
RobotmotionHandler
- heading_angle: double
- speed_: double
- max_speed_: double
+ RobotMotionHandler()
+ Reset(void) : void
+ AcceptCommand(cmd: enum
event_commands) : void
+ UpdateVelocity(st: SensorTouch) : void
+ speed() : double

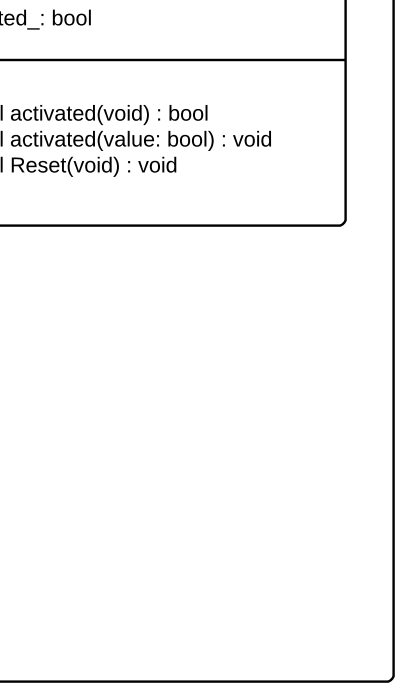

Holds an Arena; periodically asks for updates and then draws the Arena
Arena
- x_dim_ : double
- y_dim_ : double
- n_robots_: unsigned int
- n_obstacles_: unsigned int
- robot: Robot*
- recharge_station_: RechargeStation*
- home_base_: HomeBase*
- entities_: std::vector<class ArenaEntity*>
- mobile_entities: std::vector<class ArenaMobileEntity*>
- gameWon_: bool
- gameLost_: bool
+ explicit Arena(params: const struct arena_params * const)
+ ~Arena(void)
+ AdvanceTime(dt: unsigned int) : void
+ Accept(e: EventKeypress*) : void
+ Reset(void) : void
+ n_robots(void) : unsigned int
+ n_obstacles(void) : unsigned int
+ obstacles(void) : std::vector<class Obstacle*>
+ mobile_entities(void) : std::vector<class ArenaMobileEntity*>
+ robot(void) const : Robot*
+ home_base(void) const : HomeBase*
+ recharge_station(void) const : RechargeStation*
+ gameWon() : bool
+ gameLost() : bool
- CheckForEntityCollision(ent1: const class ArenaEntity* const; ent2: const class ArenaEntity* const; ec: EventCollision*; collision_delta: double) : void
- CheckForEntityOutOfBounds(ent: const class ArenaMobileEntity* const; ec: EventCollision*) : void
- UpdateEntitiesTimestep(void) : void
1
n
1
1
1
1
n
It is almost impossible to remember every single change that was made to the code and to get every connection right on a project of this size. I did my best to document all changes but it's possible something was forgotten; this is one of the limitations of doing this manually.
Color (struct)
+ r: int
+ g: int
+ b: int
+ a: int
+ Color(void)
+ Color(in_r: int; in_g: int; in_b: int; in_a: int)
Arena_Entity
- radius_ : double
- pos_ : Position
- color_ Color
+ ArenaEntity(radius: double; pos: const Position&; color: Color&)
+ virtual ~ArenaEntity(void)
+ virtual Reset(void) : void
+ virtual name(void) = 0: std::string
+ pos(pos: const Position&) : void
+ pos(void): const Position&
+ color(void) : const Color&
+ color(color: const Color&) : void
+virtual is_mobile(void) = 0 : bool
+radius(void) : double


M_SLOW_DOWN
event_commands) : void
+ UpdateVelocity(st: SensorTouch) : void
+ speed() : double
+ speed(sp: double) : void
+ heading_angle() const : double
+ heading_angle(ha: double) : void
+ max_speed() : double
+ max_speed(ms: double) : void
Robot
- next_id_: unsigned int
- id: int
- heading_angle_ : double
- angle_delta: double
- battery_: RobotBattery
- motion_handler_: RobotMotionHandler
- motion_behavior: RobotMotionBehavior
- sensor_touch SensorTouch
+ explicit Robot(params: const struct robot_params* const)
+ResetBattery(void): void
+ Reset(void) : void
+ HeadingAngleInc(void) : void
+ HeadingAngleDec(void) : void
+ TimestepUpdate(dt: unsigned int) : void
+ Accept(e: EventRecharge*) : void
+ Accept(e: EventCollision*) : void
+ EventCmd(cmd: enum event_commands) : void
+ battery_level(void) : double
+ heading_angle(void) const : double
+ heading_angle(ha: double) : void
+ speed(void) : double
+ speed(sp: double) : void
+ id(void) const : int
+ pos() const : Position
+ pos(Position pos) : void
+ name(void) const: std::string
Robot needs a battery to function
RobotBattery
+ kBASE_DEPLETION: double = 0.1
+ kLINEAR_SCALE_FACTOR: double = 0.01
+ kANGULAR_SCALE_FACTOR: double = 0.01
- charge_: double
- max_charge_: double
+ level(void) const: double
+ EventRecharge(void) : void
+ Reset(void) : void
+ Deplete(old_pos: __unused Position, new_pos: __unused Position, dt: __unused double) : double
+ Accept(e: EventCollision*) : void
ArenaMobileEntity
- collision_delta_ : double
+ ArenaMobileEntity(radius: double; collision_delta: double; pos: const Position&; color: const Color&)
+ is_mobile(void) : bool
+ virtual heading_angle(void) const = 0 : double
+ virtual pos() const = 0 : Position
+ virtual pos(pos: Position) const = 0 : void
+ virtual speed(void) = 0 : double
+ virtual speed(sp: double) = 0 : void
+ collision_delta(void) const : double
+ virtual TimestepUpdate(dt: uint) : void
+ virtual Accept(e: EventCollision*) = 0 : void
+ virtual Accept(e: EventRecharge*) = 0 : void
RobotMotionBehavior
+ RobotMotionBehavior(void)
+ UpdatePosition(ent: class ArenaMobileEntity * const; dt: uint) : void
ArenaImmobileEntity
+ ArenaImmobileEntity(radius: double; pos: const Position&; color: const Color&)
+ is_mobile(void) : bool
HomeBase
- heading_angle_: double
- speed_: double
- initial_heading_angle_: double
- initial_speed_: double
- ix: double
- iy: double
+ explicit HomeBase(params: const struct home_base_params* const)
+ name(void) const : std::string
RechargeStation
- activated_: bo
+ SensorTouch(
+ activated(void
+ activated(valu
+ point_of_cont
+ point_of_cont
+ angle_of_con
+ angle_of_con
+ Accept(e: Eve
+ Reset(void) :
- point_of_conta
- angle_of_cont
1
1
1
1
n


.1
ble =
louble
ion,
SensorTouch
of_contact_: Position
_of_contact_: double
orTouch()
ated(void) : bool
ated(value: bool) : void
_of_contact() : Position
_of_contact(p: Position) : void
_of_contact(void) : double
_of_contact(aoc: double) : void
pt(e: EventCollision*) : void
t(void) : void
Sensor
ted_: bool


+radius(void) : double
Obviously all of these interact with various entities, but their interactions are somewhat trivial (they are just used in the constructor), and adding lines would be messy. Thus I've excluded that.
arena_entity_params (struct)
+ radius: double
+ pos: Position
+ color: Color
+ arena_entity_params(void)
arena_mobile_entity_params (struct)
+ collision_delta
+ arena_mobile_entity_params(void)
arena_params (struct)
+ robot: robot_params
+ recharge_station: arena_entity_params
+ home_base: home_base_params
+ obstacles[MAX_OBSTACLES]: arena_entity_params
+ n_obstacles: size_t
+ x_dim: uint
+ y_dim: uint
home_base_params (struct)
+ collision_delta: double
+ home_base_params(void)


+ explicit HomeBase(params: const struct home_base_params* const)
+ name(void) const : std::string
+ TimestepUpdate(dt: uint) : void
+ Accept(e: EventCollision*) : void
+ Reset() : void
+ heading_angle(void) const : double
+ heading_angle(ha: double) : void
+ speed(void) : double
+ speed(sp: double) : void
+ RechargeStation(radius: double; pos: const Position&; color: const Color&)
+ name(void) const: std::string
- activated_: bo
+ virtual activate
+ virtual activate
+ virtual Reset(
Obstacle
- next_id_ : unit
- id_: int
+ Obstacle(radius: double; pos: const Position&; color: const Color&)
+ name(void) const: std::string
robot_params (struct)
+ battery_max_charge: double
+ angle_delta: uint
+ robot_params()
Position (struct)
+ x: int
+ y: int
+ Position(void)
+ Position(in_x : int; in_y: int)


ted_: bool
l activated(void) : bool
l activated(value: bool) : void
l Reset(void) : void