Much like before, GraphicsArenaView is responsible for actually drawing the Arena and everything in it. It asks for updates from Arena, and Arena returns information required to draw entities correctly. Since Arena contains all of the entities, it asks all of them to update themselves and then reports the results. Now Arena also keeps track of whether you've won (collided with home base) or lost (battery depleted fully). When that information is passed to GraphicsArenaViewer, it is displayed on top of the Robot.

The inheritance structure of the program is mostly unchanged. All entities inherit from ArenaEntity and then are subdivided into whether they're mobile or not. However, HomeBase needed to be mobile for Iteration1, so it now inherits from ArenaMobileEntity rather than ArenaImmobileEntity. To make HomeBase truly mobile, it needed to handle movement and collisions somehow; I opted to handle this directly in HomeBase rather than forcing a RobotMotionHandler and RobotMotionBehavior onto it. In the future, there should be a general MotionHandler and MotionBehavior (or just a MotionHandler, because splitting into two makes little sense) from which specific handlers and behaviors inherit. A collision delta had to be added to the params of home base as well (it is a required component of a mobile entity).

SensorTouch now inherits from Sensor. A generalized sensor only can be activated and reset, as well as being able to return its state. SensorTouch is largely the same, but now adds its specific behavior as part of an inheritance structure.

No changes were made to the event inheritance structure, which still has all Events inheriting from EventBaseClass. While collision events interact directly with mobile entities, keypress events interact with Arena and indirectly affect mobile entities. Recharge events interact directly with Robot, and then Robot passes them to RobotBattery which actually handles them.

EventCommands has been modified to include a slow down command; the forward command was modified to a speed-up command to give EventCommands a more "commandy" feel.

The color of entities used to be handled via passing nanogui::Color around, but now has been replaced with a custom Color struct. The Color struct contains integer values for red, green, blue, and alpha (brightness). A common convention is to represent colors with ints, since most of the time colors go from 0-255 which is a byte, but the Color struct is more convenient.

Note: Many changes were made to the code since the start of Iteration1; it is impossible to mention all of them. To my knowledge, all structural changes are mentioned here.