The Arena Viewer program begins with the main function that creates a new controller and begins to set up the arena by using the controller’s Run() function. The controller is the bridge between the arena and the graphics viewer. The graphics viewer takes information from the virtual arena and displays it based on what the controller tells it about object positions and states of the arena. When the arena is initiated it creates a virtual space for objects to interact with each other including 4 border walls. It first uses the Object Factory to create the controllable robot, then bases and finally the obstacles. All objects are included in the intial ent\_ vector which is a vector of pointers to arena entities. The mobile\_entities\_ vector is a separate vector that all mobile entities get added in to along with ent\_. The robot gets it’s own separate vector and the obstacles get their own pointer vector as well. This is so that you can apply functions to and from just the robot, obstacles, or both mobile entities. Once the entity factory works with the arena to set up and initialize all objects the UpdateEntitiesTimestep function is called continuously to simulate time passing for each object. This function can be broken down into a few phases. It first updates all entities timesteps by 1. Then it checks to see if the game is won or lost. It finally detects collisions, first with walls and then other objects and lets each mobile object know which wall or entity it has bumped in to. Robots are made up of a motion behavior, motion handler and sensor. The motion behavior uses the robot’s previous position and current speed to determine its new pose. The motion handler describes how each of the increase velocity, decrease velocity, turn left and right. The robot also has a date member to keep track of current lives. If the robot runs into a wall or obstacle it loses a life. If the robot runs into an obstacle then it gains mercy invincibility for a brief moment, but lives can still be decreased during that period if the robot collides with a wall. Each time update checks the current life count of the robot and if it is 0 or less it pauses the game and displays the lose message to the player. The obstacles are mobile and initially all go right as the default heading but deviate their path once they hit a wall or other object. The RecoilInFear function is called and turns the object around and has it arc out of the way for a brief period. The time can be set to different rates for more dramatic recoiling. As of now, the time for it to recoil is short and is more akin to a bounce. The bases are immobile and contain just themselves. When the robot collides with them, their color changes and their captured\_ flag is set. When the UpdateTimeStep is called from arena and checks the status of each base. If all bases are captured the game is paused and an indication that the player won is displayed.