Assignment 1

Brian Baron

The architecture for Assignment 1 is built on the base steering project. The project was already linked, and just needed to be recompiled (including external DeanLib).

Unit Management

I made a UnitManager class to handle all of the “enemy” units in the game. Units can be added and deleted dynamically, and they are referenced by pointers stored in a std::map, mUnits. This allows for easy organization as well as random access. The UnitManager works with no memory leaks, and integrates with the input/messaging system in order to create units.

Input Manager/Messaging

The InputManager handles all mouse and keyboard input. When a key press is detected, the manager sends a message of corresponding events (ADD_UNIT_EVENT, DELETE_UNIT_EVENT, etc.) to the message manager. This way the UnitManager knows when to add and delete units without having to be directly coupled to the input system. The init function for this system returns a bool—if true, everything installed properly. If false, the InputManager failed to install a component (mouse or keyboard). This was put in to help with debugging, as it is directly using Allegro syntax.

Steering Behaviors

The steering used in this solution are Dynamic Seek and Dynamic Arrive. These were already integrated in the project, and just had to be applied to units based on input. When creating units, the constructor asks for a bool to tell is whether to seek or arrive, as for this project we only used the two patterns. In future implementations where more steering is used, I would replace this with and enumerated data type for ease of use.