Karim Elzanaty

Jeff Timanus

AI Programming

September, 2018

1. Overall Architecture

I've created multiple Steering classes: Arrive, Wander, Face, Arrive and Face, and Wander and Chase. Each of these are their own class that derive from a Steering super class. Two of the classes, Arrive and Face, as well as Wander and Chase, use variables of other classes, such as Arrive, Face, and Seek. These variables are defined at the same time as the base class, and each of their getSteering functions are used to update the owner unit. The Arrive and Face steering uses the acceleration data as well as the velocity data from the Arrive class to decide on movement speed, but has no bearing on how the owner unit should rotate. It relies instead of the rotational acceleration and rotational velocity from the Face class to rotate accordingly. If the steering from Arrive or Face has reached its destination, it will return a value of '0'(zero) to the Arrive and Face class so that it stops rotating or moving. The Wander and Chase steering uses steering from Seek, Wander, and Face. The implementation of delegate variables is the same as in Arrive and Face. When the unit is not in range of its target, it will Wander. Wander already delegates to face to determine its steering, so the extra call is not needed here. When the unit *is* in range of its target, movement is delegated to Seek, which does not have a rotational component, so data from Face Steering is needed here. Each class's getSteering was moved to public access for ease of use. I used my old Event System and Input Manager from my Game Arch final

project from last year rather than the current event manager because I was more familiar with it. The Event class stores each Event type as an enum. The Event System class adds listener calls to a map, based on the listener type and who the listener is. It also removes listeners and fires events to be heard. The Event Listener class is a virtual class that can handle events from a listener. The Input System detects keyboard and mouse input using SDL and uses the Event System to fire events depending on the input. The Game is a listener that the Input System sends Events to so that Game's HandleEvent function can act. The Game uses Enter, Esc, left click, and the D-Key as input for different actions. There is also a function in the Input System to get the mouse position that the game uses which does not require user input, as it happens every frame regardless of any input.

2. Challenges faced in development

The biggest challenge was the Face steering. For the longest time I had no idea why a radius was required from rotation, as a radius implied a point from the center of a circle outward, and I didn't understand what that had to do with turning. This was until I realized that a turn radius is a thing, and even still, implementing what the book had suggested was difficult. In fact, attempting to decipher the book as a whole was challenging. The written theory was much more helpful than any pseudo code, since with the theory I could at least get an idea of what I was supposed to do. The Arrive and Wander steering were also a bit tough, as the math behind them just took me a while to get my head around.

3. Areas where further improvements could be made

I’m sure the way I implemented my Input System and Event Managers into the game could be cleaner. As it stands they are a bit glued on, and aren’t as clean as I would like them to be. The maximum turning speed on each unit seems a bit slow, to the point where during a chase they can move faster than they can turn. Which results in some lopsided movements.