Josh Grazda

EGP-410

Timanus, Jeff

September 24, 2018

Assignment 1

Solution Architecture:

The overall architecture of my solution follows the architecture previously provided in the base code. I have followed many of the same practices demonstrated by the Seek Steering class to provide consistency throughout the solution. By providing pointers to various classes of Steering, I was able to create my blended steering types such as ArriveAndFace and WanderAndChase. The input system architecture is based off a previously created solution by Dean Lawson. Using the included GameMessagingManager and GameMessage classes I created messages of my own for user input, which is derived and executed from the InputSystem. I kept consistency in variables naming conventions throughout my code based off the base code provided.

Solution Challenges:

The solution came with many challenges and one I would later discovered. Initially I knew the main challenge I would face is the mathematical approach that all the steering methods take. While not my strongest characteristic, I was able to understand the principals behind what each steering type used to work correctly. This came about by deciphering the pseudo code provided in “Artificial Intelligence For Games: Second Edition” by Ian Millington and John Funge or by receiving help from my peers. Another challenge I faced was getting the steering method from the principal into code. While I had a basic understanding of how each one should work, the changing between vectors, radians, and degrees often confused me and created code that did not provide the needed movement. This remains a problem in some aspects of my code.

Solution Improvements:

My solution could greatly benefit from some improvements. My current steering classes have some minor issues, mainly due to FaceSteering, which need to be fixed for proper steering of the other types. In the same respect, another area that could use improvement is to streamline my use of steering types to create blended types of steering. Currently I have pointers to each steering type I need to blend and then simply get the data as I need it. By sub classing these steering types I could improve the architecture of my code for steering. I could also improve upon the use of my InputSystem. By creating a separate translator for buttons, I could consolidate the button functions and reduce the amount of messaging classes I have. This would allow me to easily add additional buttons and messaging classes. Finally, my code could use some basic cleaning and commenting to increase readability. This would also help with future debugging and making the improvements started above.