Michael Zhang
netID: mbz27

Part 1

Controls
WASD - Walk in any direction
'Left-Shift'+WASD - Run in any direction
Space+WASD or Space+'Left-Shift'+WASD or Space - Jump in direction (while running, walking, or idling)

I used two 2D Freeform Cartesian trees for the running and walking animations. The player can transition between any of the states. It can transition from: idle to walking, idle to running, idle jumping, walking to idle, walking to running, walking to jumping, running to idle, running to walking, running to jumping. The running and walking animation trees deal with all four directions that the player can move in (forward, right, left, and backward) and has animations for forward right, forward left, backward right, and backward left. I was not too sure what to use the 2D Blend Tree for since I already have running and walking animations so I used it to animate the walking backward animation in only part 1.

I add a player controller that will handle the collision/gravity for the animator such that it can jump on to and over obstacles like the block in the demo.

Part 2

Controls
Scroll up/down - Zoom in/out
Left-click on an agent - Deselect or select the agent
Right-click anywhere on the ground - Set the destination of the agent
Drag the slider to the desired speed

NOTE: I used an implementation that was on the unity documentation to implement the jump animation when navigating off mesh links

Part 3

Controls
The same controls in part 2
The red floor has a cost greater than the yellow and green floors
The yellow floor has a cost greater than the green floor but less than the red floor
The green floor has a cost less than the yellow floor, the green floor, and the regular floor

Michael Zhang
netID: mbz27

For this project, I did not use the braking mechanism I implemented in project B1 (due to time constraints) but opted to implement a simpler mechanism where the agents will stop if it detects the agent in front of it has stopped. To determine whether an agent has stopped each agent has a ‘hasStopped’ state where it will be set to true whenever it either is in a certain radius around the set destination or detect the agent in front of it has a ‘hasStopped’ state set to true. And when the ‘hasStopped’ state is set to true the agent with the state will stop.