For my decision tree my boid engages in a few decisions, first it looks at if it is still completing a task(but only eating, or dancing) and if so returns to it. I decided to implement this as my root node so that I could prioritize finishing eating but not necessarily completing a path. The fruit on my map is generated at random time intervals at random locations so I wanted to avoid situations where the boid might walk past a fruit to eat a fruit that is further away. If an action is already completed then the decision tree checks if the boid has just finished eating. If so then it should trigger the boid to dance. If these both fail then the boid will check if there are any fruit and then if there are if he can see them. If this fails then the boid will check if it is in a corner location on the map.

Due to the design of my rooms I found that sometimes the characters will occasionally get stuck in corners and so I added this behavior to try and centralize the boid occasionally. The fall back behavior if not in the corner or if fruit is not visible is to wander using a modified wander steering behavior which essentially just arrive with randomly selected nearby goals. Due to the way my environment is encoded I cannot do any meaningful movement outside of one singular tile “freely” doing so will result in the characters ignoring my walls entirely. I must also be careful that when following a path the pointer to the next goal is moved up very carefully to avoid too much free movement. If the fruit does exist and if the boid is close enough to perceive it (based only on path length unfortunately not walls) then the boid should move to the fruit. Once reaching the fruit the fruit should be eaten.

I believe that the monster's behavior tree is probably the best part of my project. He has three main behavior routines. The first routine is a sequence which deals with killing the boid, and subsequently dancing. This routine starts by checking if the boid is within the mob's sights, if successful then the mob attempts to approach the player, should this succeed then the mob hits the boid, checks if they died, if not hits again, and if this is successful the mob dances and then the game is reset. If any of these steps fail (outside of a second hit to the boid if the first kills it) then the mob moves only the second branch of the root node which is a selector. The first child of this selector is a randomized selector dealing with fruit. Each time the tree is rebuilt this selector is randomized for is the mob will prioritize eating or smashin the fruit. I included the smashing as I was curious how a random selector would work, but it is not really optimized. Currently however it is still kind of fun to witness as the boid (and mob)don’t check if the fruit has been smashed until after they’ve approached the fruit and attempt to eat it.

If I had been able to get a node for wait working on the behavior tree one might imagine the boid waiting near a smashed fruit until they are able to ambush the boid. In its current state however it is still a little interesting to watch the boid approach a fruit and then simply wait near it. This leads to the mob usually killing the boid pretty quickly if the action to smash fruits is chosen. The last routine is to patrol. When the patrol action is selected the mob decides which corner it is closest to. This then becomes the last corner they have visited and then the mob moves to a different corner. Currently the only way to interrupt this routine is by seeing the boid or reaching a corner. As such the mob walks past fruit very consistently. I do find in general the mob tends to kind of get stuck patrolling and only break out of it to kill the boid. This is understandable given the current implementation as once the mob is patrolling many paths through the task end

up leading to returning a running status as opposed to a success status or failure. This is necessary currently however as otherwise the patrol action is extremely jerky as a new path is calculated every frame. I found it easy to visualize what I wanted the tree to do, but extremely difficult to implement the tree in c++, due to my own lack of experience with the language.

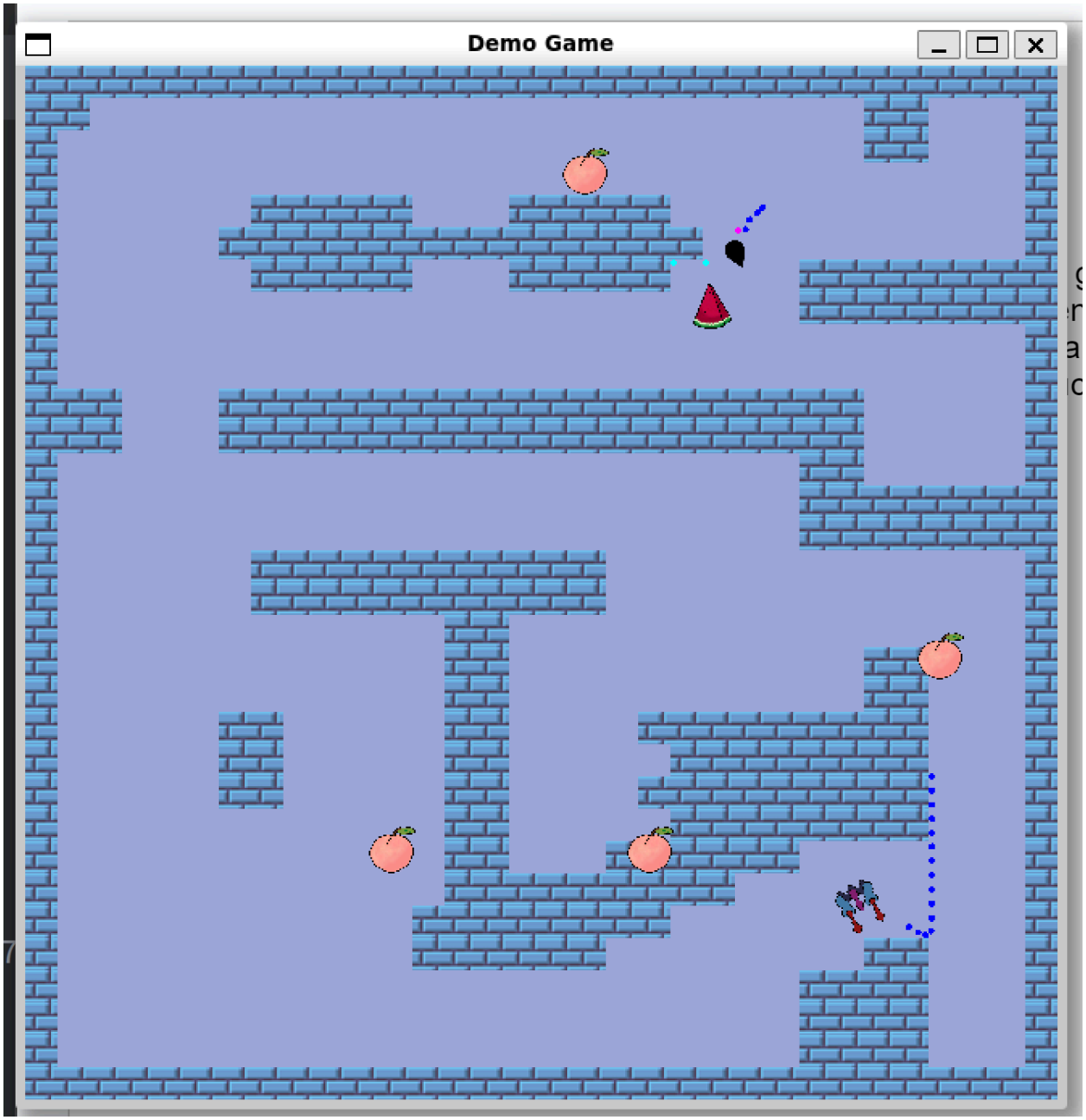
In my game environment I am able to expose a great many variables. Some would probably argue that I have exposed too many. On my first attempt at training the Mob I had 29 variables. Practically everything. Booleans for if either character was eating, dancing, or their movement along a path was satisfied(meaning they had hit a way point and are ready for the next one). As well as all the usual things, position, velocity, rotation, distance between the characters and their various goals. The list went on and I hoped that with enough information I could possibly recreate my overly complex behavior tree. Alas instead I only ended up with a mob that could barely move and took over 4 mins to win a game. There were many times where I simply quit as watching him fail to move after being shunted down to actions he couldn't possibly satisfy was horribly frustrating.

I realized after much frustration that my problem was two-part. In my behavior tree my actions were very strict on when they could activate, and what tolerances I had for how close the mob should be to the goals for various actions. The other of course was that I had over complicated my data and in essence my mob was unable to determine which pieces of information would be useful and which pieces were just essentially noise. In the end I kept player position data, including velocity, mob position data, including velocity, distance to player, and the path lengths from mob to player and fruit as well as player to fruit. I hope that with the more focused data the algorithm is better able to correlate the distance from player to mob with if they should move closer and a short path from distance to fruit with moving to eat the fruit instead. When nothing else is applicable I believe due to the frequency of patrol actions the mob should fall back to patrolling.

The current implementation is…somewhat better than the first as in some runs it can catch the boid in under a minute. Many runs however end in the boid simply sitting in place. I have attempted to edit some of my actions in order to be more lax with the mobs goals to reduce or eliminate the sitting in place but I believe part of the issue lies in an issue with both hitplayer and eat fruit where there aren't enough representations in the data to really learn preconditions for the actions. If I could find a way to more realistically model the data or even normalize the data I feed into the model I believe I would potentially find a fix to the issue. As it stands however the learned tree is much worse at killing the boid compared to the behavior tree.

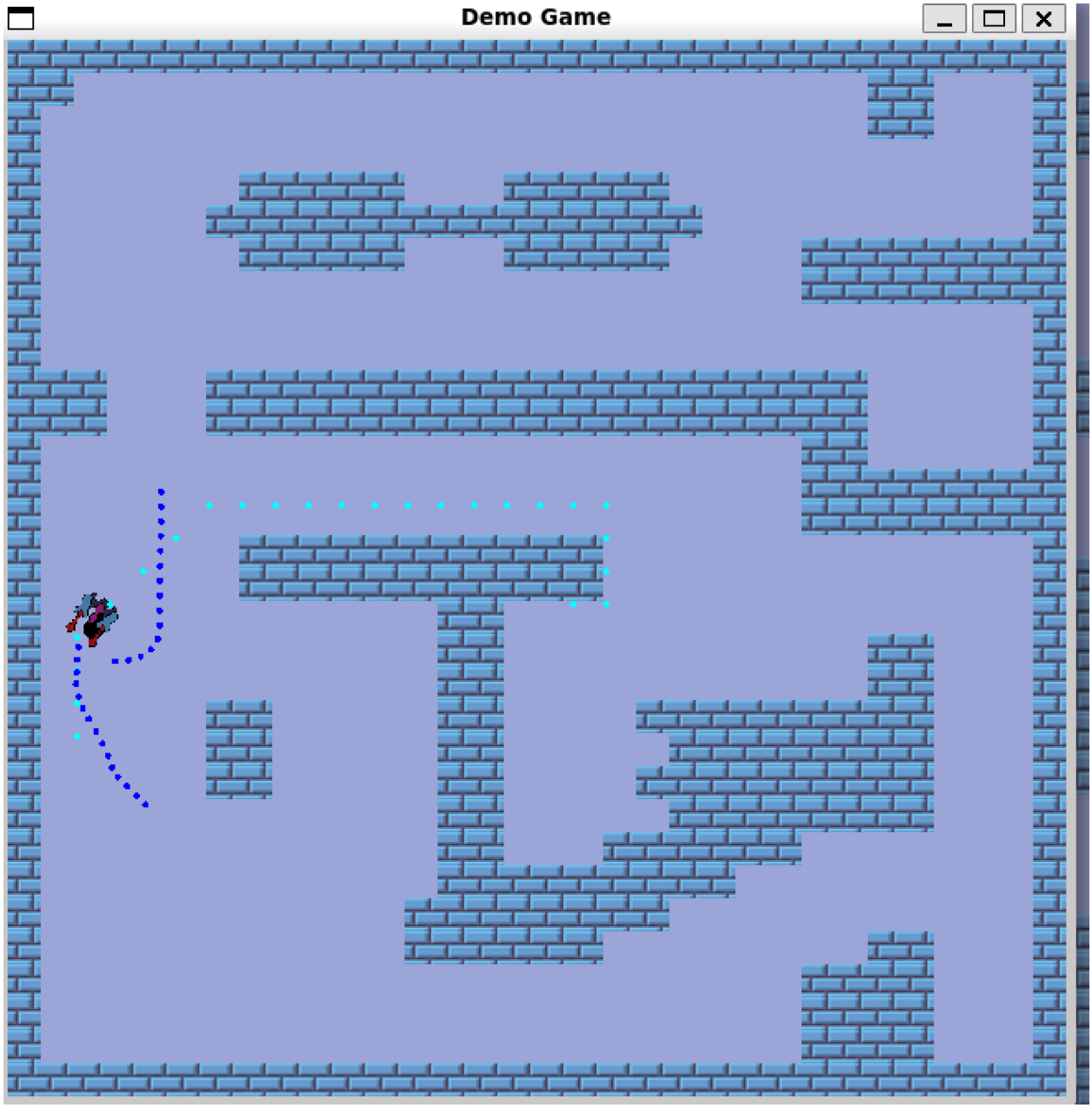
I believe something like reinforcement learning would have been much more effective of a model for my particular behavior tree due to the complexity in actions and their preconditions. Overall the learned decision tree does ok if the only thing on the board is the boid. The action to move to player seems to more or less work except for when the mob gets stuck attempting to move to fruit anyway. I found the learning model for this assignment to be extremely difficult to implement and as a result it took far longer than expected. Having worked with learning decision trees before outside of a games context I did not expect it to be as difficult as it was.

I did experiment with several different tweaks in the data gathering process as well as tweaks such as the mobs range of sight. I think the current implementation of the learning decision tree is likely as good as I am able to make it currently with the behavior tree I had already developed. Thank you so much for this class. I struggled through it but overall found it immensely rewarding.



Learned Tree demonstrates some mob patrol behavior

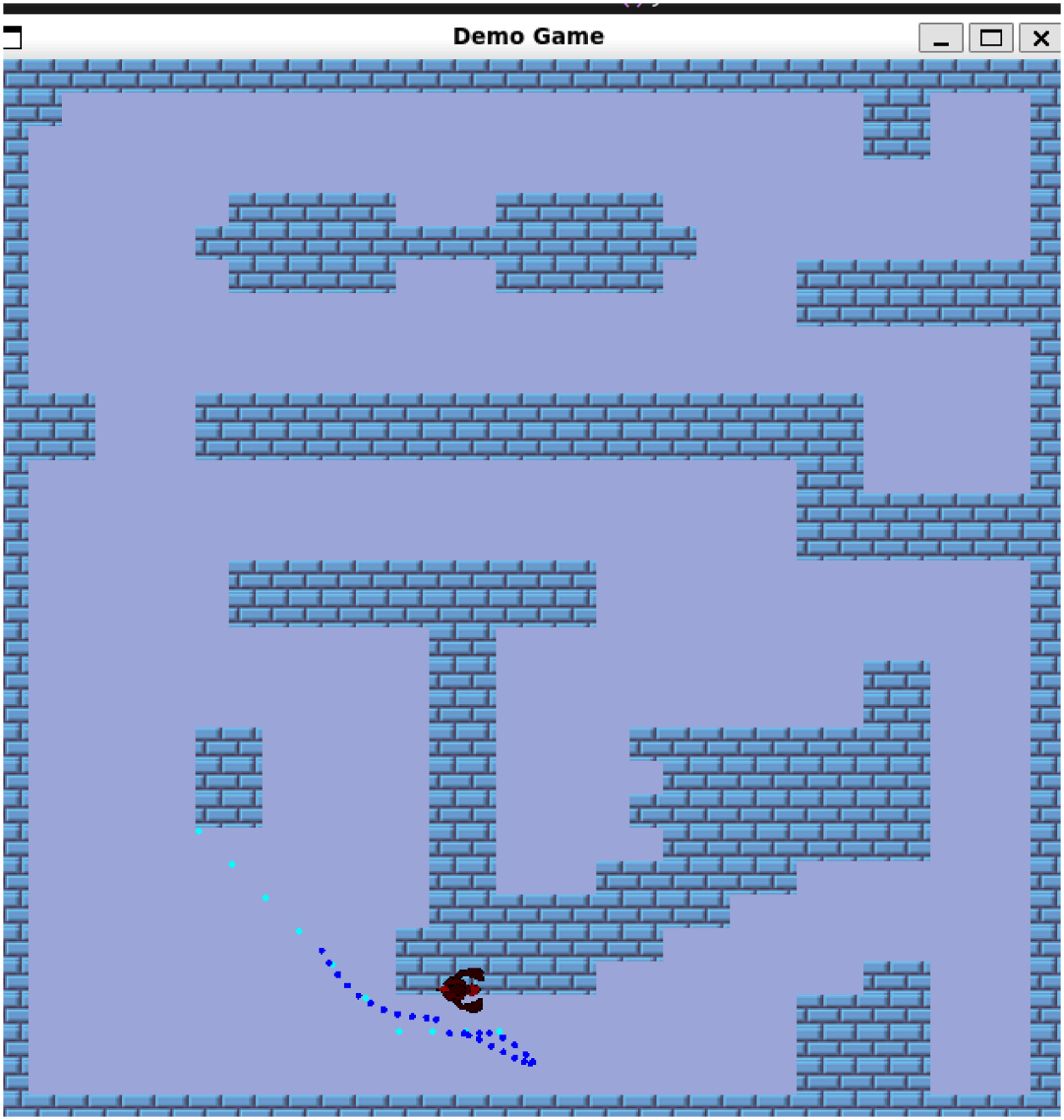
But sometimes halts near character without attacking due to low number of attack nodes(if any with some data)



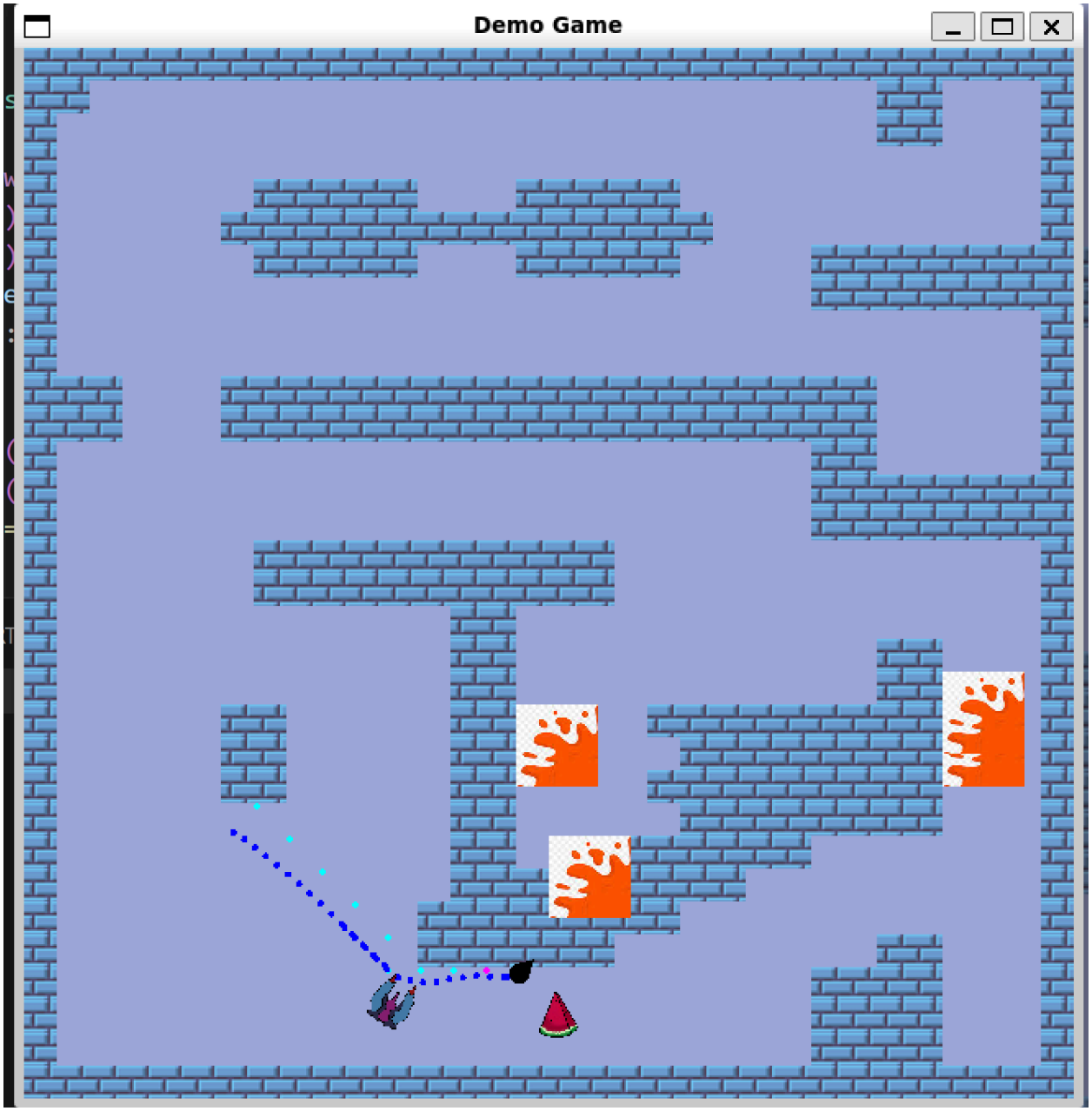
It can however and sometimes does kill the boid.

A	B	C	D	E
remake	3.20754	Remake	18.782	
remake	3.37231	Remake	13.6354	
remake	7.32571	Remake	62.5253	
remake	28.4432	Remake	79.9029	
remake	4.91969	Remake	13.4213	
remake	6.38581	Remake	13.0499	
remake	37.2501	Remake	5.05311	
remake	7.89749	Remake	1.08556	
remake	9.99336	Remake	11.1539	
remake	7.30636	Remake	74.2354	
remake	12.3961	Remake	17.5892	
remake	24.2917	Remake	8.44284	
remake	7.69594	Remake	3.70665	
remake	9.4765	Remake	15.0738	
remake	3.76462	Remake	31.0395	
remake	34.4481	Remake	15.2925	
	13.01091		23.99933	
	7.796715		14.3546	
	Behavior		Learned Decision	

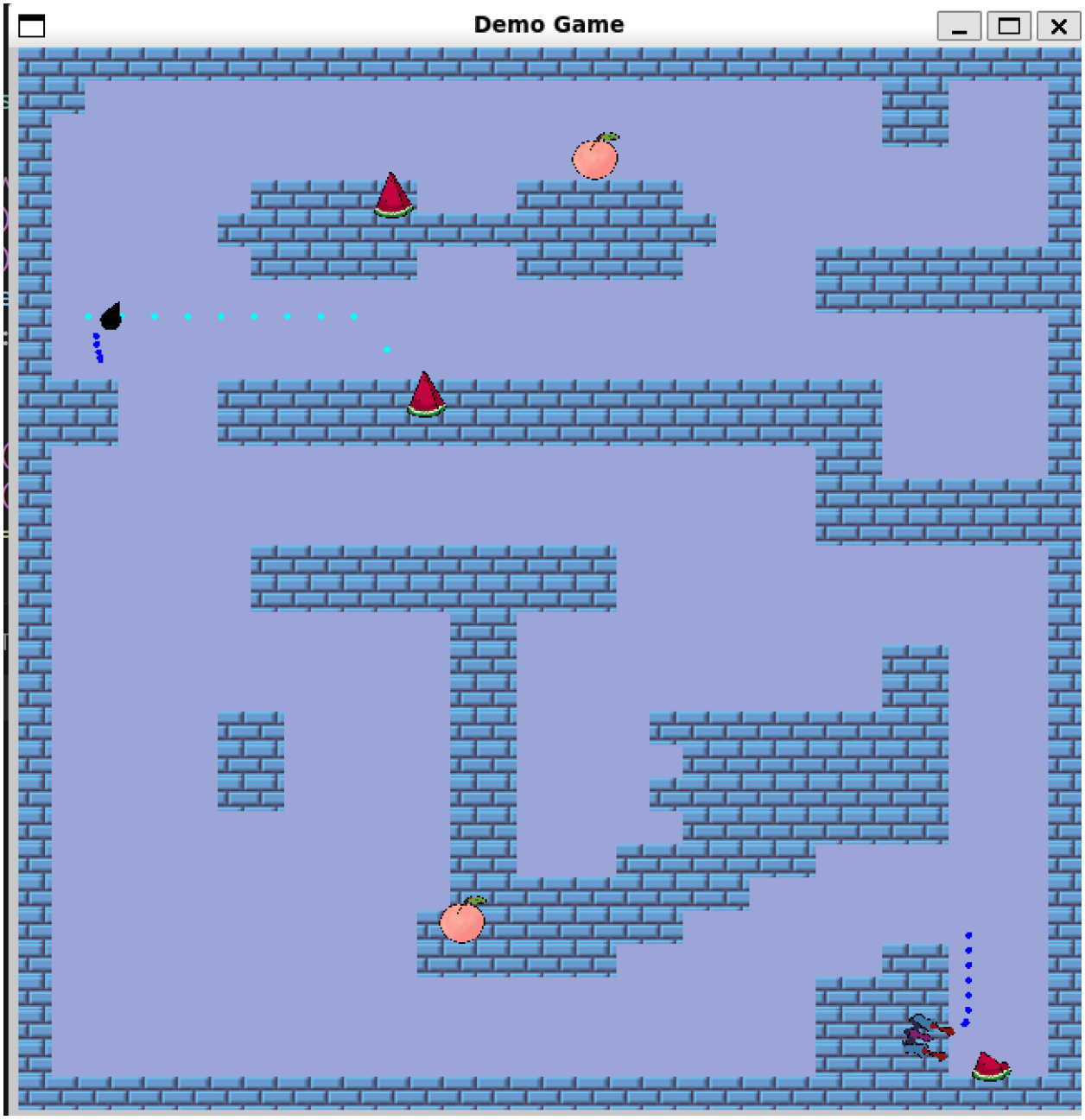
On avg the behavior tree is twice as fast as the learned decision tree however they both suffer from outliers.



Sweet dance moves from the mob

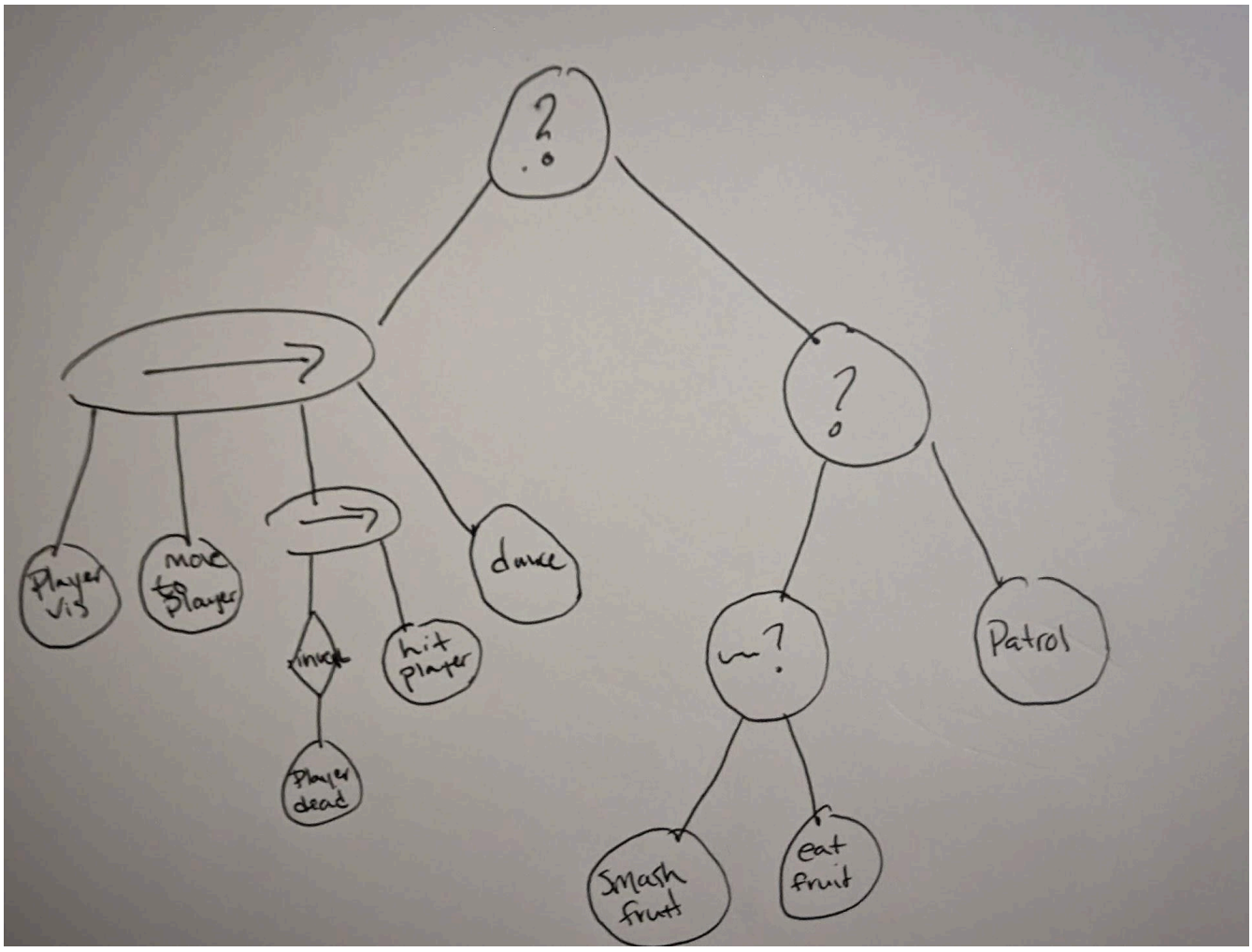


Proof that fruit can be smashed



Mob eating fruit

Hand drawn behavior tree with arrows representing sequences, diamonds inverters, the squiggly line near the question mark is a random selector, and the other question marks regular selectors



Decision tree for mob