I was asked to implement an AI that could control a tank and that the player could compete against. I initially used a State Machine design for the AI but after receiving feedback I opted to use a behavioural tree design. Below is the initial feedback I received and under it you can find the rationalisation for the changes I made.

Initial Feedback:

*Badly named timer method, does more than the method name implies*

*- Magic values in the hunting AI state*

*- When choosing a random waypoint, they didn't notice that Random Range with an int is exclusive for the max value, so it will never use the last waypoint. I would consider this a minor bug.*

*- Inconsistent casing on variable names*

*+ Attempted a reasonably architected solution to the AI problem*

*- Find used with no null checking*

*+ Demonstrates usage of cross product to get a correctly signed angle*

*+ Shows use of animation curves*

*+ clean (ish) code*

*- The ToXState methods aren't the best design, as a new method will need adding for every single state added, which means this will require a lot of boilerplate code to maintain in future if the AI is extended*

*My response with the updated projected:  
  
I have written this cover letter in the hopes of securing a graduate programmer position within Mediatonic. I had previously applied for this role in June but was unsuccessful after submitting my code exercise. The code exercise included making changes to the ‘Tanks’ game so that the world boundary wasn’t revealed when the two tanks moved away from each other and in addition I was also asked to implement one of the following features; Implement an AI with 3 modes of operation or convert the game to a checkpoint based race game. I chose to implement the AI.*

*For the first exercise, I deduced that the simplest and most efficient way to hide the world boundary was to increase the size of the world and set a barrier to stop the players from reaching the edge of the world. Most isometric games implement something similar to this especially when it is important for the player to be able to quickly see the entire map. Another possibility was to set the camera to follow the main player once the two tanks were at a certain distance away from each other; then using some sort of indicator to show where the other tank is positioned. However, since this is a game that can be played locally by two players, this would mean that one player would not be able to see their tank.*

*Furthermore, even if this was just a one player game, the player would need to be able to see obstacles which are in between his/her own tank and the enemy tank. For example, firing missiles at long distances would require the player to see the if the line of fire is clear or obstructed by any objects which are outside of the camera viewpoint; this could lead the player to be firing ‘blind’ where he/she is firing and hitting objects that it cannot see; again making this method non-effective.*

*Moreover, the code exercise asked me to implement an AI with 3 modes of operation; Roaming, Hunting and Evading; which was done. In addition to this I added a forth Alert Roaming operation which the AI switches to if the line of sight between the AI and the other tank is broken. In addition, the AI will roam around the last point the other tank was seen. I originally went with a Finite State Machine as the method to implement the AI as it was what I had previously used in university. A FSM allowed me to create a different state for the different operations mentioned above and also allowed me to state what conditions had to be met for the AI to transition to different states. I managed to implement all operations successfully and in addition even implemented features such as hiding behind obstacles and picking the correct launch force for the missile; enabling it to hit a moving tank.*

*However, my code exercise did not permit me to progress to a face to face interview. I was given positive and negative feedback which I have now used to improve my code. I will go over what the feedback was and what changes I have made.*

*Positive feedback:*

1. *Attempted a reasonably architected solution to the AI problem*
2. *Demonstrates usage of cross product to get a correctly signed angle*
3. *Shows use of animation curves*
4. *clean (ish) code*

*Negative feedback:*

1. *Badly named timer method, does more than the method name implies*
2. *Magic values in the hunting AI state*
3. *When choosing a random waypoint, they didn't notice that Random Range with an int is exclusive for the max value, so it will never use the last waypoint. I would consider this a minor bug.*
4. *Inconsistent casing on variable names*
5. *Find used with no null checking*
6. *The ToXState methods aren't the best design, as a new method will need adding for every single state added, which means this will require a lot of boilerplate code to maintain in future if the AI is extended*

*Changes*

1. ***“Badly named timer method, does more than the method name implies”***

*In my first submission the way I implemented a timer was to have a timer function in every state that needed one. The function would add to a float variable called timer in every frame and when ‘timer’ reached a certain amount, it would execute a function and would reset itself. Although this worked well enough, the method itself was doing a lot more than just keeping track of the time.*

*For this new submission I decided to have only one timer function instead of multiple ones for each state. The function itself would only add time.deltaTime to a float variable and then a different function would check if the timer was above a certain time depending on what state the tank was in; if it was, It would execute a certain block of code and then execute a timer reset function so that the timer was brought back down to 0.*

1. ***“Magic values in the hunting AI state”***

*In my first submission I manually inputted a parameter that a function needed so that it knew what it’s FOV angle would be. I understand it would be difficult for programmers and designers to make changes to the project without having to comb through my code. However, I have mainly been accustomed to programming on my own and have not needed to consider how easy someone else could make changes to my code. Being a graduate without much opportunity to have gained this consideration through working in a professional team unfortunately lead me to making this mistake.*

*For this submission I used a public variable. This allows the FOV amount to be changed from the inspector window (designer friendly), and it also allows it to be changed from other scripts.*

1. ***“When choosing a random waypoint, they didn't notice that Random Range with an int is exclusive for the max value, so it will never use the last waypoint. I would consider this a minor bug.”***

*In my last submission I used random.range to create an index to select a waypoint from a list of waypoints. It would pick a number between 0 and the List.Count. Random Range does exclude the max value which would be the amount of objects in the list but when working with lists or arrays, the first object starts at 0. This means that if there are 50 objects in a list, List.Count will return 50 but the 50th object is actually called with the index 49. So yes random.range with an int is exclusive for the max value but in this case it meant that that random.range wouldn’t return a value that couldn’t be used.*

1. ***“Inconsistent casing on variable names”***

*As already mentioned, being someone who usually writes code for himself, I had forgotten to follow some sort of casing convention. I do understand that a casing convention makes the code easier to read and understand for other programmers who will have to work with it and am fully aware of it being fundamental to good code. In my old submission I did follow the same casing convention as the project itself but after some last minute changes, I forgot to make sure that the new variables I added followed the same rules.*

*I have made sure to comb through my code in this submission to avoid this happening again and have learnt that this is essential to continue doing.*

1. ***“Find used with no null checking”***

*Similar to the reason above, I didn’t think to use null checking as I was 100% sure that the objects GameObject.Find would be looking for, would be in the scene.*

*However, I fully understand why it’s important to use null checking. If the project is worked on by someone else in the team and they remove objects from the scene, the scripts would not be able to find them. Null checking allows the scripts to find out if an object has been removed and then using a debug log, I can write a message which would be displayed in the console telling the person who is running the project, what is missing.*

1. ***“The ToXState methods aren't the best design, as a new method will need adding for every single state added, which means this will require a lot of boilerplate code to maintain in future if the AI is extended”***

*I chose to use the FSM method for the AI as it is what I have the most experience in when it comes to programming AI. It allowed me to create the several states needed and I could specify what condition needed to be met for it to transition from one state to another.*

*Although the FSM design worked, I understand the disadvantages of using a FSM. Removing or adding features could involve changing a large chunk of the code, making prototyping features difficult as it would require a lot of time and effort. For this submission I did some research on different AI designs and decided to attempt the behavioural tree method.*

*With behavioural trees I can create tasks (execution nodes) and they can be placed at the end of any sequence of any tree. This allows me to use the same tasks under different conditions in different operations without having to rewrite the task itself. It is also very designer friendly as behavioural trees can also be shown with a GUI allowing designers to easily understand what and why an AI is doing without having to look at the code. There are also behavioural tree assets available in the asset store that allow them to be built and modified using a GUI, making easy for prototyping different features or behaviours for both designers and programmers. An image of the behavioural tree which I created has been placed in the root of the project folder.*

*I’d like to thank you for the opportunity you have given me. As I am currently in a full time job, I have little time to work on projects or do research on different methods and programming techniques. Mediatonic looks to be an amazing company to work for, you have developed several different games which fall under different game genres. You also seem to have a great culture within your workplace that allows employees to express themselves. I’ve seen twitter posts from previous TonicJams and it appears to be a great way to experiment with new ideas whilst showing the rest of the company your talents. Although I am a graduate, my ambition is to eventually lead my own projects and even develop my own game studio (with the hopes of putting all others out of business, just a little humour) I am sure that over time I can bring a lot to the table as an employee for Mediatonic Games.*

*Assets used:*

*The only assets I used from the asset store is Panda BT Free by Eric Begue (*[*https://assetstore.unity.com/packages/tools/ai/panda-bt-free-33057)*](https://assetstore.unity.com/packages/tools/ai/panda-bt-free-33057))*. I used this as I didn’t think it was necessary to build an entire behavioural tree framework but I’m confident that with a lot more time I could build one.*