|  |
| --- |
| Southampton Solent University |
| DAC619 AE1 |
| Artificial Intelligence for Games |

|  |
| --- |
| Q13375466\_RAY\_STEPHEN\_DAC619\_AE1\_Report |

Contents

[Design 2](#_Toc534326215)

[Identification of Required Behaviours 2](#_Toc534326216)

[Chosen Algorithm 2](#_Toc534326217)

[Other Algorithms Considered 2](#_Toc534326218)

[Implementation 3](#_Toc534326219)

[Changes to Base Project 3](#_Toc534326220)

[Design Choices 3](#_Toc534326221)

[Testing 4](#_Toc534326222)

[Test Plan 4](#_Toc534326223)

[Testing Results 4](#_Toc534326224)

[Conclusions and Evaluation 6](#_Toc534326225)

[Appendices 7](#_Toc534326226)

[Appendix A – State Diagram 7](#_Toc534326227)

[Appendix B – Testing Evidence 8](#_Toc534326228)

# Design

## Identification of Required Behaviours

These are the behaviours that each of the AI agents will need to perform to successfully play the game of capture the flag.

|  |  |
| --- | --- |
| * Go to Enemy Base * Grab Enemy Flag * Go to Friendly Base * Heal * Use Powerup | * Chase Enemy * Grab Friendly Flag * Go for Powerup * Defend Base * Attack Enemy |

## Chosen Algorithm

The algorithm that I decided to implement was a state machine. The primary reason behind this choice was the presence of distinct states in the behaviours required, which would heavily enable itself to that of a state machine. A state diagram of the state machine can be seen in Appendix A. Some of the advantages of using a state machine for the AI logic is that it is quite easy to trace a path of the AIs behaviour as it will be playing, making the logic that it will be applying easy to understand and to debug for any potential issues whilst it is going through the game.

A large disadvantage of the state machine is that even with such a small amount of states, connections between them and transitions are becoming complex, with many states all leading to many places.

The algorithm will work by first placing the AI in the state of going to the enemy base, as at the start of the game it is known that both flags are in their respective bases, and both teams will need to take the flag back to their base in order to start scoring points. Along the way to this goal, the AI can choose to do other actions based on what they can see, so they can be opportunistic and chose to go for Powerups or Med kits in order to have them for later use if needed, and if they encounter an enemy along their way, they will try to attack them in combat. AI agents will then try to claim flags and bring it back to their base, with an ideal situation being both having the enemy flag, and their own flag in their base to defend. This will continue forever, with flags going back and forth between the AI teams.

## Other Algorithms Considered

One algorithm that was considered for the game was goal orientated behaviour. Goal orientated behaviour would be a very good selection for this project, as it would be easy to identify needs / goals such as need to heal, claim flags, and to defend. This would be a very robust solution to use, as the variety of actions and goals allows for a wide range of behaviour, and the AIs would seem to have very different behaviours between them using some fuzzy logic and a few actions that meet multiple goals and needs. The main reason why this was not used, was the difficulty of designing a successful set of goals and actions, as to make the behaviour feel more realistic, it would need to be very vast in its capabilities, which in this project it would not be the easiest to do.

The other option that was considered was the use of a decision tree to determine the behaviour of the AI agent. However, this was not chosen as the final algorithm as it would be required for the AI to be able to change their behaviour quite fast based on conditions, and it was felt that it would not provide enough freedom for the AI to move between the actions or behaviours that it can be doing. However, there would be benefits to using a decision tree, for example, the decision tree would be easy to debug, as you can trace the behaviour along the tree, it would also be a very efficient solution if it is designed correctly, as careful planning of the tree would minimise the number of decisions that the will need to be made at any point, with the least common decisions such as using powerups right at the bottom of the tree, and more used ones, such as going to the enemy base and attacking enemies closer to the top of the tree.

# Implementation

## Changes to Base Project

During the implementation of the State Machine, I had to make a few changes to the base code of the project. These changes were done to facilitate the AI behaviour but not in a “cheating” way. The two functions that were added were a public Boolean function in the SetScore.cs file to let the base publicly let agents know if their friendly flag was in the base and if the enemy flag was in the base. This was done as it was an essential part of deciding the behaviour of the AI Agent at any one point, as if the enemy flag was in the base and their own one wasn’t, they should try to recover it to prevent the enemy from scoring any more points, likewise, if the enemy flag isn’t in their base, they should go and get it to score points.

The other change that was made was to the AI.cs file, where a set of public functions for retrieving the private agent members, so that they could be accessed within the State classes.

## Design Choices

One of the design choices made when it came to implement the project was adding a cooldown onto the attacks, as at the default rate, the health of the Ai agents didn’t last very long, so I decided to add in a cooldown for each attack that the AI Agents do.

Another choice that was made was some slight random chance when it came to decide if it was going to do certain behaviours, such as when it came to attack, the AI will sometimes choose not to engage with them and keep on going for their goal. This was done to give some variation to the Ai, so not everyone would be doing the same bits and to make them slightly more unpredictable.

# Testing

## Test Plan

In order to test the project, primarily black and white box testing will be used, with black box testing being used to trace the AIs route through the states using the state diagram to follow what the expected behaviour will be with what their situation is. White box testing will be used to look at the values using the debugger, to check variables that are maybe not the most distinct such as the state it is in, as that is not as easy to see from the game, as it could be travelling to a few locations.

Some of the behaviours that will be tested will be observing the actions that the AI should do in different situations.

## Testing Results

Testing evidence can be found in Appendix B, with headings for each test and its respective evidence. These pieces of evidence will show what is going on in the debugger and what is going on in the game, showing how the agent switches to different states when decisions are made or criteria have been met.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Situation | Expected AI Behaviour | Actual AI Behaviour | Action |
| 1 | Start of the Game | The Ai Agents move towards the enemy base. | The Ai Agents move towards the enemy base. | None needed. |
| 2 | Enemy is in line of sight of agent | They move towards the enemy. | They move towards the enemy. | None needed. |
| 3 | Enemy is in attacking range of agent. | Agent attacks Enemy. | Agent attacks Enemy. | None needed. |
| 4 | Agent sees health pickup. | Agent moves towards health pickup. | Agent moves towards health pickup. | None needed. |
| 5 | Agent is in grab range of health pickup. | Agent picks up health pickup. | Agent picks up health pickup. | None needed. |
| 6 | Agent sees powerup pickup. | Agent moves towards powerup pickup. | Agent moves towards powerup pickup. | None needed. |
| 7 | Agent is in grab range of powerup pickup. | Agent picks up powerup pickup. | Agent picks up powerup pickup. | None needed. |
| 8 | Agent sees enemy flag not in friendly base. | Agent picks up enemy flag. | Agent picks up enemy flag. | None needed. |
| 9 | Agent sees enemy flag in friendly base. | Agent ignores enemy flag. | Agent ignores enemy flag. | None needed. |
| 10 | Agent sees friendly flag not in friendly base. | Agent picks up the flag. | Agent picks up the flag. | None needed. |
| 11 | Agent sees friendly flag in friendly base. | Agent ignores the flag. | Agent ignores the flag. | None needed. |
| 12 | Agent is low on health and has health kit. | Uses health kit. | Uses health kit. | None needed. |
| 13 | Agent is low on health and doesn’t have health kit. | Runs away from enemies. | Runs away from enemies. | None needed. |
| 14 | Agent is in attacking range of enemy and has powerup in inventory. | Uses powerup and then attacks enemy. | Uses powerup and then attacks enemy. | None needed. |
| 15 | Agent is holding enemy flag. | Moves towards friendly base. | Moves towards friendly base. | None needed. |
| 16 | Agent is holding enemy flag and is over friendly base. | Agent drops flag in base. | Agent drops flag in base. | None needed. |
| 17 | Agent is holding enemy flag. | Moves towards friendly base. | Moves towards friendly base. | None needed. |
| 18 | Agent is holding enemy flag and is over friendly base. | Agent drops flag in base. | Agent drops flag in base. | None needed. |
| 19 | Friendly flag and enemy flag in friendly base. | Defends friendly base. | Defends friendly base. | None needed. |
| 20 | Enemy flag is not in friendly base. | Goes to enemy base. | Goes to enemy base. | None needed. |
| 21 | Friendly flag is not in friendly base. | Goes to enemy base. | Goes to enemy base. | None needed. |

# Conclusions and Evaluation

Overall, the use of a state machine for the behaviour of the Artificial Intelligence of this project was a success. The AI Agents play the game competitively, and have a balanced game after a while, with both teams going back and forth fighting for control.

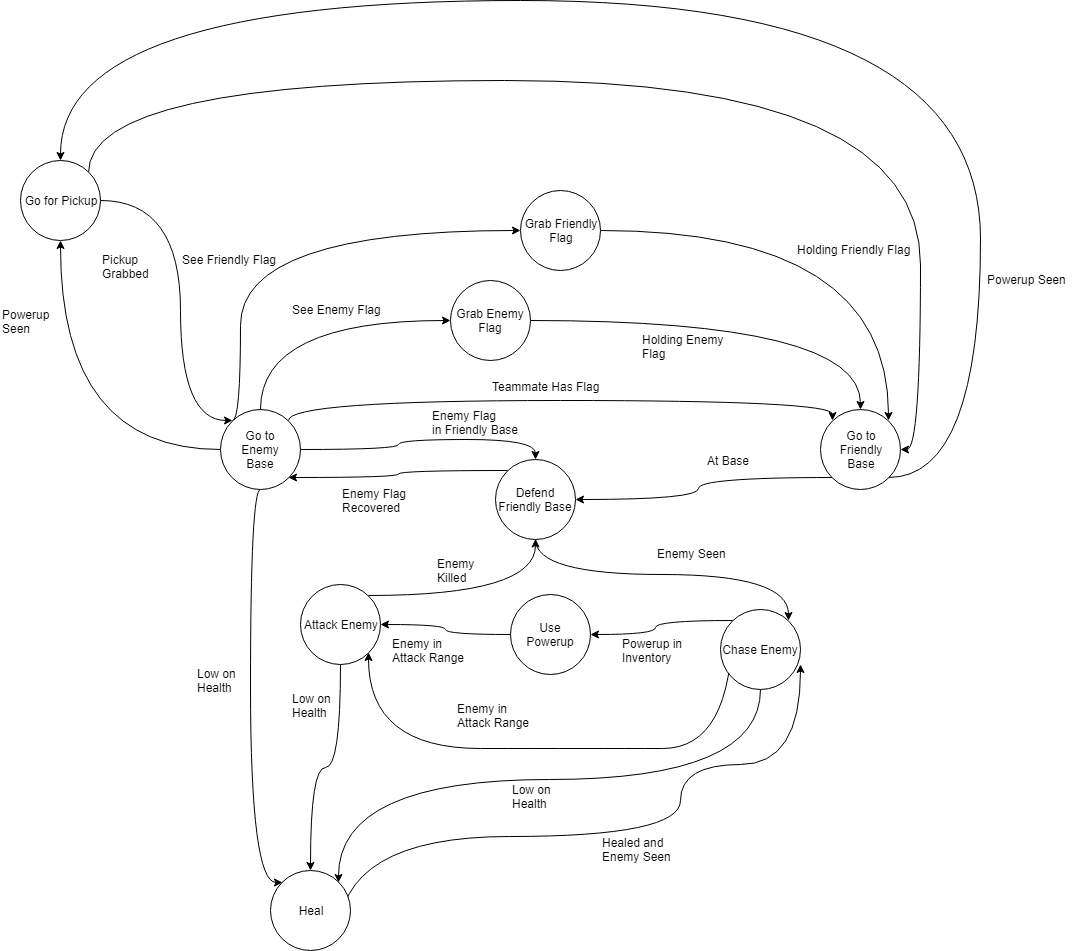
One feature that I would implement in another attempt at this would be a state where they are actively hunting for the flag, pickups or collectables, as there is the chance that these items can end up outside of the typical path that is traversed by the navmesh pathing, so if it was to be lost on the outside areas of the map, there is every chance that it may never be found, leading the game to either unplayable, or massively in one teams favour.

Another behaviour that would’ve been preferable to have is for teammates to group together and communicate more, with them moving to objectives together and distributing powerups and med kits between them to get much more effective usage out of them, by giving ones on low health the health kit rather than holding onto it.

I would have also wanted to have the AI use more fuzzy logic in their behaviour, with them taking more possible decisions in each state, which would have added onto the complexity of the state machine, and it may be better suited to an algorithm such as goal orientated behaviour, where there can be more possible actions to suit goals.

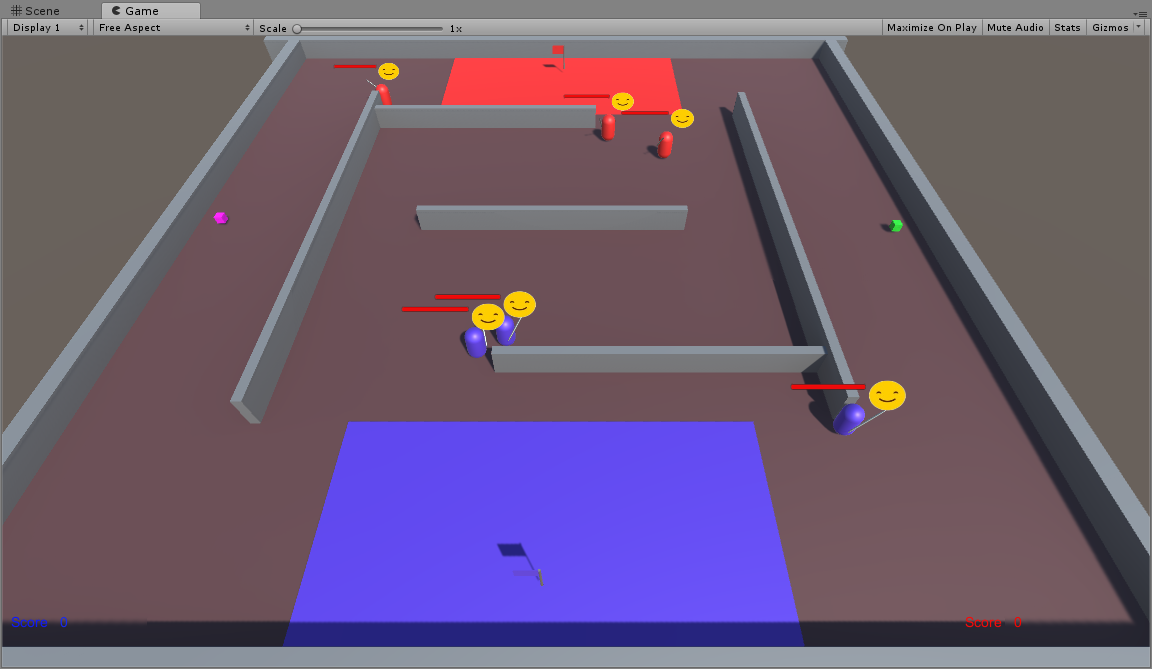
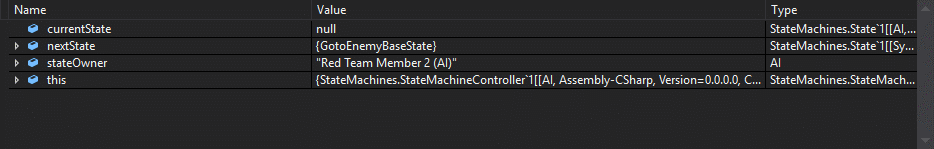
One thing in this project that I do like, is emergent behaviours that seem to appear when the AI agents are playing, such as them moving as a group, or going back to defend their base, which all appear due to how their state machines work and was not initially an intended behaviour of the system.

# Appendices

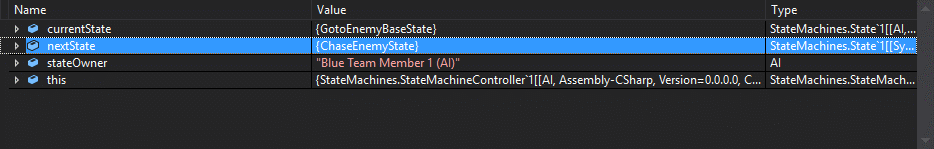
Appendix A – State Diagram

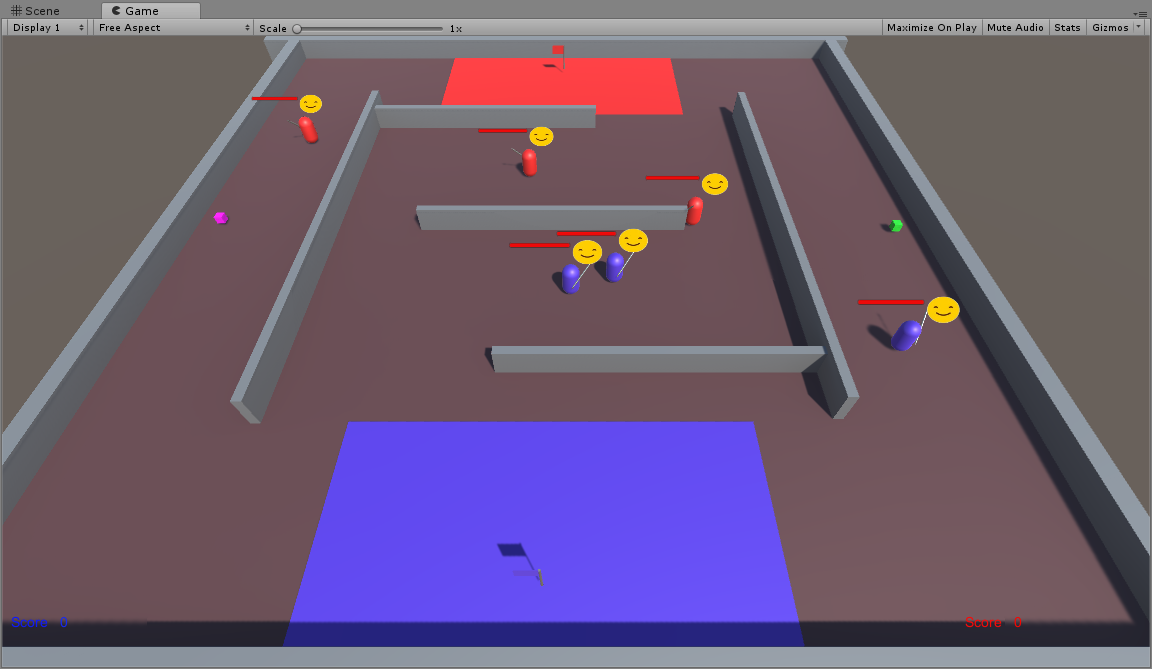
## Appendix B – Testing Evidence

**Test 1**

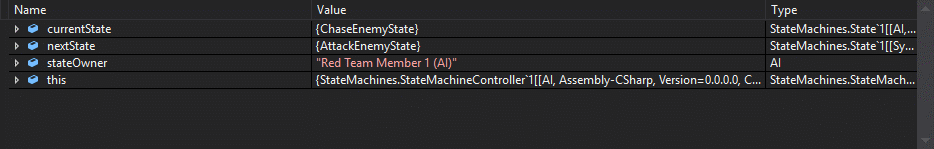
****

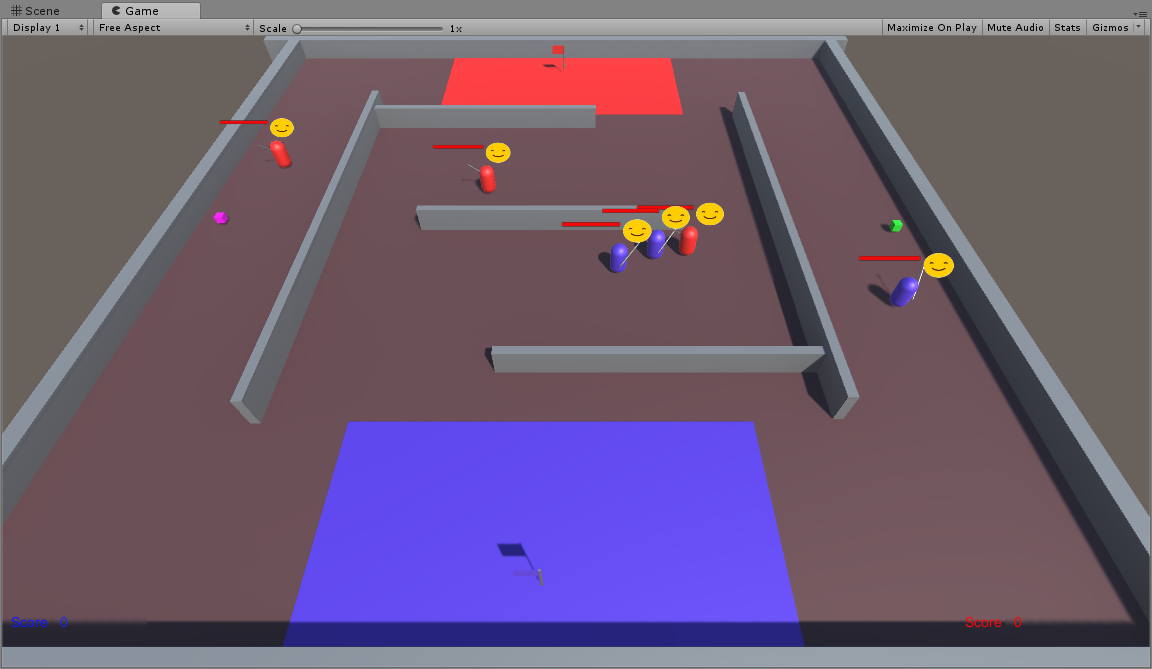
**Test 2**



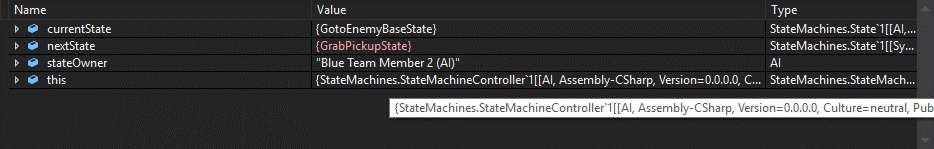


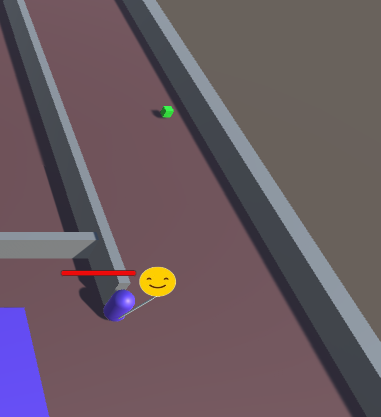
**Test 3**





**Test 4**

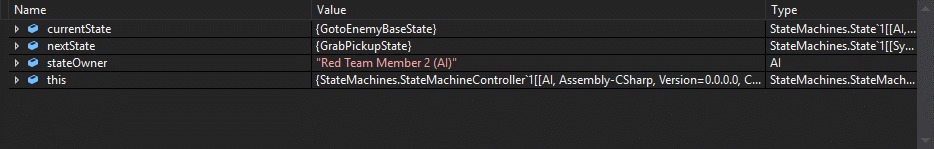


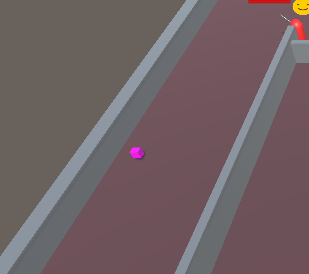


**Test 5**



**Test 6**

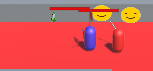




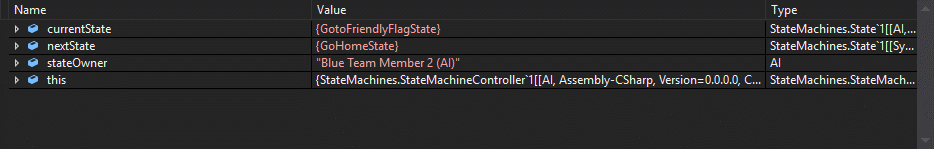
**Test 7**

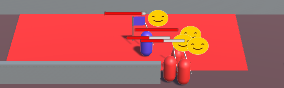


**Test 8**

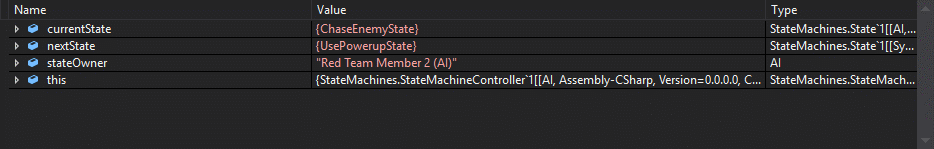


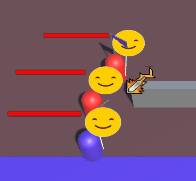
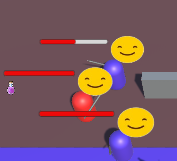
**Test 10**





**Test 14**





**Test 18**

