# Content

Contents

[Content 1](#_Toc55587424)

[Assignment Brief 3](#_Toc55587425)

[Assessment Cover Sheet 2020-21 3](#_Toc55587426)

[Assessment Task 3](#_Toc55587427)

[Marking Scheme 6](#_Toc55587428)

[Description of project 7](#_Toc55587429)

[Research on Core AI algorithms 8](#_Toc55587430)

[A\* search 8](#_Toc55587431)

[A\* within Python and AI development 8](#_Toc55587432)

[Finite state Machines (FSMs) 9](#_Toc55587433)

[FSMs used within the Computing industry 9](#_Toc55587434)

[FSM used within my Bunner AI development project 9](#_Toc55587435)

[FSMs vs A\* search algorithm 10](#_Toc55587436)

[Testing of AI and implementation of techniques used in the code 11](#_Toc55587437)

[Movement 11](#_Toc55587438)

[Test description 11](#_Toc55587439)

[Test screenshots 11](#_Toc55587440)

[Result/improvements? 11](#_Toc55587441)

[Object detection 12](#_Toc55587442)

[Test description 12](#_Toc55587443)

[Test screenshots 12](#_Toc55587444)

[Result/improvements? 12](#_Toc55587445)

[Stopping before a Car collision 12](#_Toc55587446)

[Test description 12](#_Toc55587447)

[Test screenshots 12](#_Toc55587448)

[Result/improvements? 13](#_Toc55587449)

[Stopping before a Train collision 13](#_Toc55587450)

[Test description 13](#_Toc55587451)

[Test screenshots 13](#_Toc55587452)

[Result/improvements? 13](#_Toc55587453)

[Waiting for Log to appear in the water (Not immediately jumping into the water) 13](#_Toc55587454)

[Test description 13](#_Toc55587455)

[Test screenshots 14](#_Toc55587456)

[Result/improvements? 14](#_Toc55587457)

[Moving upon colliding with a bush 14](#_Toc55587458)

[Test description 14](#_Toc55587459)

[Test screenshots 14](#_Toc55587460)

[Result/improvements? 14](#_Toc55587461)

[Other Bugs or issues? 15](#_Toc55587462)

[Source Code (with annotation) 16](#_Toc55587463)

# Assignment Brief

 **Faculty Of Computing and Engineering Sciences**

# Assessment Cover Sheet 2020-21

|  |  |  |
| --- | --- | --- |
| **Module Code:** | **Module Title:** | **Module Team:** |
| CS3S667 | Artificial Intelligence for Game Developers | Mike Reddy |
| **Assessment Title:** | | **Assessment No.:** |
| (Re)Creating the Classics | | 1 |
| **Date Set:** | **Submission Date:** | **Return Date:** |
| 28-Sep-2020 23:59 | 06-Nov-2020 23:59 | 04-Dec-2020 23:59 |

**IT IS YOUR RESPONSIBILITY TO KEEP RECORDS OF ALL WORK SUBMITTED.**

|  |
| --- |
| **Marking and Assessment** |
| This assignment will be marked out of **100%**.  This assignment contributes to **50%** of the total module marks. |
| **Learning Outcomes to be assessed** |
| As specified in the validated module descriptor [https://icis.southwales.ac.uk](https://icis.southwales.ac.uk/)   1. Understand the theory that underpins, and the pragmatic difficulties associated with, thedevelopment of a working AI game system 2. Evaluate the relative effectiveness of different approaches to AI for a given problem |
| *Awarded mark is only provisional: subject to change and / or confirmation by the Assessment Board.* |

# Assessment Task

**Assessment Task:**

Over the progress of the autumn term you are required to implement a number of core AI algorithms as exercises; these are A\* search and Finite State Machines (FSMs). You are required to select an appropriate technique(s) to achieve one of the three [3] scenarios listed below. You should then complete a technical report which discusses your selection. This technical report will:

1. Provide a description and discussion of each of the techniques you have selected. Youshould ensure this discussion includes:
   1. Relevant evaluation of the technique in terms of tasks to which it is suited, or isperhaps definitely not suited, this should be placed in the context of the given scenarios.
   2. Factors you have identified which might influence its selection for a task (such ascomplexity/computational demands).
2. Include annotated source code (assuming the implementations are minimal; where a fullimplementation is included consider providing the code in an appendix with annotated snippets being used in the body of the report to illustrate operation). The intention is to demonstrate your successful implementation of the abstract concepts involved.
3. Supply evidence of the operation of the implementation, such as experimental results andscreen shots (you do not need to implement the scenario, merely demonstrate the technique).
4. Clearly indicates how the strengths of the technique in general terms (point 1 above) applyto the specific scenarios. This will require you to present your understanding of the requirements of the scenario.
5. Documents any implementation details specific to the scenario, such as datarepresentation.
6. Use appropriate diagrams and figures to support your explanation.

**Scenarios:** The FREE Python book "Code the Classics 1" (available from https://wireframe.raspberrypi.org/books/code-the-classics1 - look for the free option) and source code (available from https://github.com/Wireframe-Magazine/Code-the-Classics via GitHub) implements simple games inspired by classic arcade games - Pong, Frogger, BubbleBobble. Centipede, and Sensible Soccer - using Python and the PyGame library. All source code for these games is available. Pong would be too simple to do as more than a formative tutorial exercise, but the following scenarios are available for you to creata a player AI to create:

1. Bunny (aka Frogger, or more recently Crossy Road) - create an AI to go the furthest
2. Myriapod (aka Centipede) - create an AI to get the high score <- was meant to be Caverns

:-(

2b) Caverns (aka Bubble Bobble) - create an AI to get the high score <- Added in, but not removed Myriapod

1. Soccer (aka Sensible Soccer) - create an AI to beat (or not be too badly beaten by) thepre-existing opponent AI

You will need to install Python version 3.8.5 https://www.python.org/downloads/ and PyGame https://www.pygame.org/wiki/GettingStarted

**Group working:** While this is ultimately an individual assignment, by selecting specific scenarios from above, you will be self-selecting a group of like-minded (similarly cursed) colleagues. There scope for group working will consist of formative discussion as to how to approach considering an AI to solve the games from a player's point of view. Implementation details must be individually created, but general strategies, and some "gentle competition" between the self-selected group members will be encouraged.

**Notes and Hints:** In the scheduled workshop sessions there will be opportunities for students to discuss and obtain formative feedback on the achievement of the workshop tasks. Students are encouraged to consider how this feedback can be used to improve their submission for this assessment. It may appear that the report requires more than the recommended word count, but you can and should try to cut to the chase on each element; avoid unnecessary detail and divergences and remember the ABC of report writing: Accuracy, Brevity and Clarity. You must however demonstrate achievement of the learning outcomes Use the Grading criteria to guide your writing.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Fail (0/29)** | **Narrow Fail (30/39)** | **3rd Class / Pass (40/49)** | **Lower 2nd Class / Pass (50/59)** | **Upper 2nd Class / Merit (60/69)** | **1st Class / Distinction (70/100)** |
| Scenario  Analysis (20%) | ☐ No evidence  that an understanding of the characteristics of the scenario was reached ☐ The scenario was not adequately explained | ☐ Some indication that the characteristics of the scenario were understood, but failing to identify these correctly ☐ The scenario is poorly understood or explained | ☐ The significant aspects of the scenario are identified in the context of the assessment ☐ Investigation of the scenario may exhibit a few errors | ☐ The significant characteristics of the scenario have been identified ☐ A few of the subtleties have been identified and discussed | ☐ The characteristics of the scenario in the context of the assessment are identified ☐ Some of the subtleties are also identified and discussed | ☐ The characteristics of the scenario have been identified and generalised  ☐ The scenarios is compared to other cases and comparisons drawn in relation to the context of implementing AI solutions |
| Scenario  Solution (40%) | ☐ The selected technique is inappropriate ☐ The selected technique is described incorrectly ☐ Explanation is missing, or contains many significant errors | ☐ A technique which is not wholly inappropriate is selected  ☐ Explanation of the technique is lacking ☐ Explanation for the selection may have significant errors | ☐ A technique is selected which is adequate though may not be the most appropriate  ☐ An explanation  for the selection of the technique is given  ☐ The explanation of the technique is over simplified or has significant errors | ☐ An appropriate technique is selected that had been supported to a degree  ☐ The technique is clearly defined  ☐ The explanation has few if any significant errors | ☐ A technique is selected supported by a well-reasoned explanation which shows an understanding of the technique and its applicability  ☐ The technique is defined with understanding of implementation issues ☐ The explanation has only minor errors | ☐ A technique is selected supported by a wellreasoned explanation which shows an understanding of the technique, and places this selection into a wider context, perhaps by comparisons or consideration of practical issues  ☐ The technique is well defined  ☐ The explanation has no errors |
| Scenario  Implementation  (40%) | ☐ No evidence that a working implementation was completed  ☐ No evidence of testing has been provided | ☐ Some aspects of the core functionality are in place  ☐ Evidence that the technique has been implemented and tested to some degree | ☐ The major  aspects of the core functionality are in place  ☐ Evidence that the implementation has been tested | ☐ Code snippets and test evidence show a successful implementation of the core functionality of the technique  ☐ Some evidence of formal testing is present | ☐ Code snippets show a successful implementation which address more than just core functionality  ☐ Evidence is provided of a range of tests | ☐ The code snippets submitted show some sophistication  ☐ Clear evidence is provided that the implementation has been tested  ☐ Tests illuminate suitable applications for the technique |
| Global: | Scenario Analysis  Scenario Solution  Scenario Implementation  Summary and Future Work | | | | | |

# Marking Scheme

# Description of project

Within this documentation I will go through methods in which I have used in order to properly execute the task given to me, which was to develop an AI algorithm to play a given game (Bunner) as efficiently and as consistently as possible.

Below I will demonstrate how I have attempted to tackle this task given to me, as well as expanding on any knowledge that I have acquired through my own personal research for this project.

# Research on Core AI algorithms

## A\* search

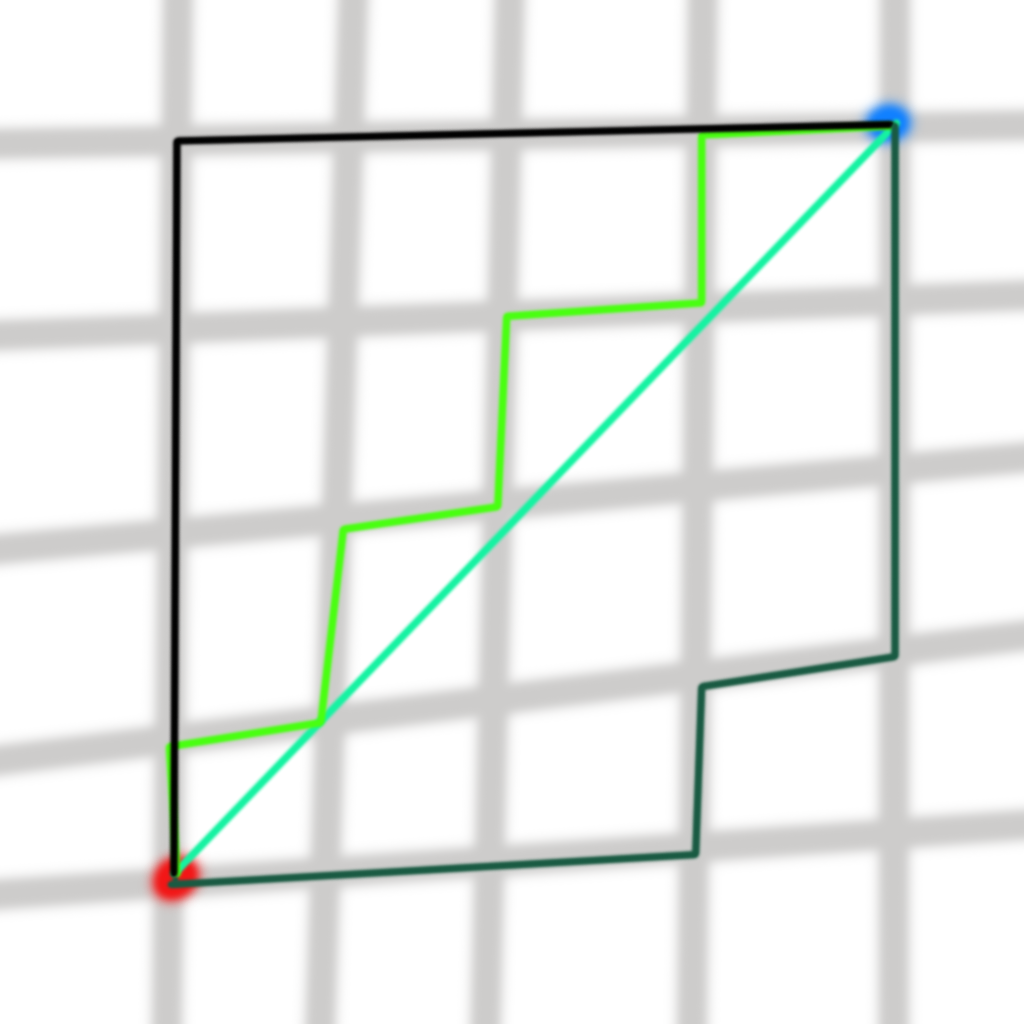
The A\* (A star) search algorithm is a method used within the development of creating AI and other similar functions within the computer science industry that, in general; attempts to create a function which allows the shortest path to be found within a graph and for that path to be traversed along by an artificial intelligence.

This algorithm is very similar to Dijkstra’s mathematical algorithm, in which a series of nodes are used in order to attempt to locate the shortest possible path to a set location, by analysing all of the nodes in order of priority and locating the next shortest node to go to I order to get to the specific location.

Overall, the A\* search method is extremely useful when it comes down to being used for events such as pathfinding which is used within this project. As the A\*search method is fantastic at being incorporated into pathfinding due to its use and reliance on finding nodes in the shortest possible time, it is due to this reasoning that I have decided to use it within my project while developing my AI.

### A\* within Python and AI development

Below I have created a diagram in order to better demonstrate the thinking paths and logic behind how the A\* method works, within the diagram below there are multiple paths within the map, the teal colour demonstrates the shortest possible route which the user is unable to take due to having to follow along the “streets” on the map, the green path is the most direct path and would be the path most likely taken if the A\* algorithm was used in order to create an artificial intelligence that let an AI direct itself from the red point to the blue point. The black pathing are alternate routes that the AI could take but would likely not as it be as directly linked towards the origin points.



#### A\* Nodes

Nodes are an important factor within the A\* search terms as a program created around use of the A\* search method will usually incorporate some sort of node system, in which it allows the program to have a starting node, an end node, and nodes connected to all other important locations between the start node and end node, this way under use of the A\* search algorithm the program would be able to calculate and follow the shortest possible path from the starting node to the end node.

## Finite state Machines (FSMs)

The Finite state machine is a type of automation that is used within computing. FSMs allow for a lot to be unlocked within the development of AI as they work around a node have multiple states which can be adjusted at any point in time to fit a specific given scenario.

FSMs are really good at simulating sequential logic as FSMs usually tend to use a logical circuit in order to convey actions.

In order to create a Finite state machine, you are required to have multiple states. Within my FSM I have 1 state named “idle” and then additional states surrounding it which emulate movement from the Bunny, this is due to FSMs being fantastic at adapting to new situations and changing state based off of a current task or situation.

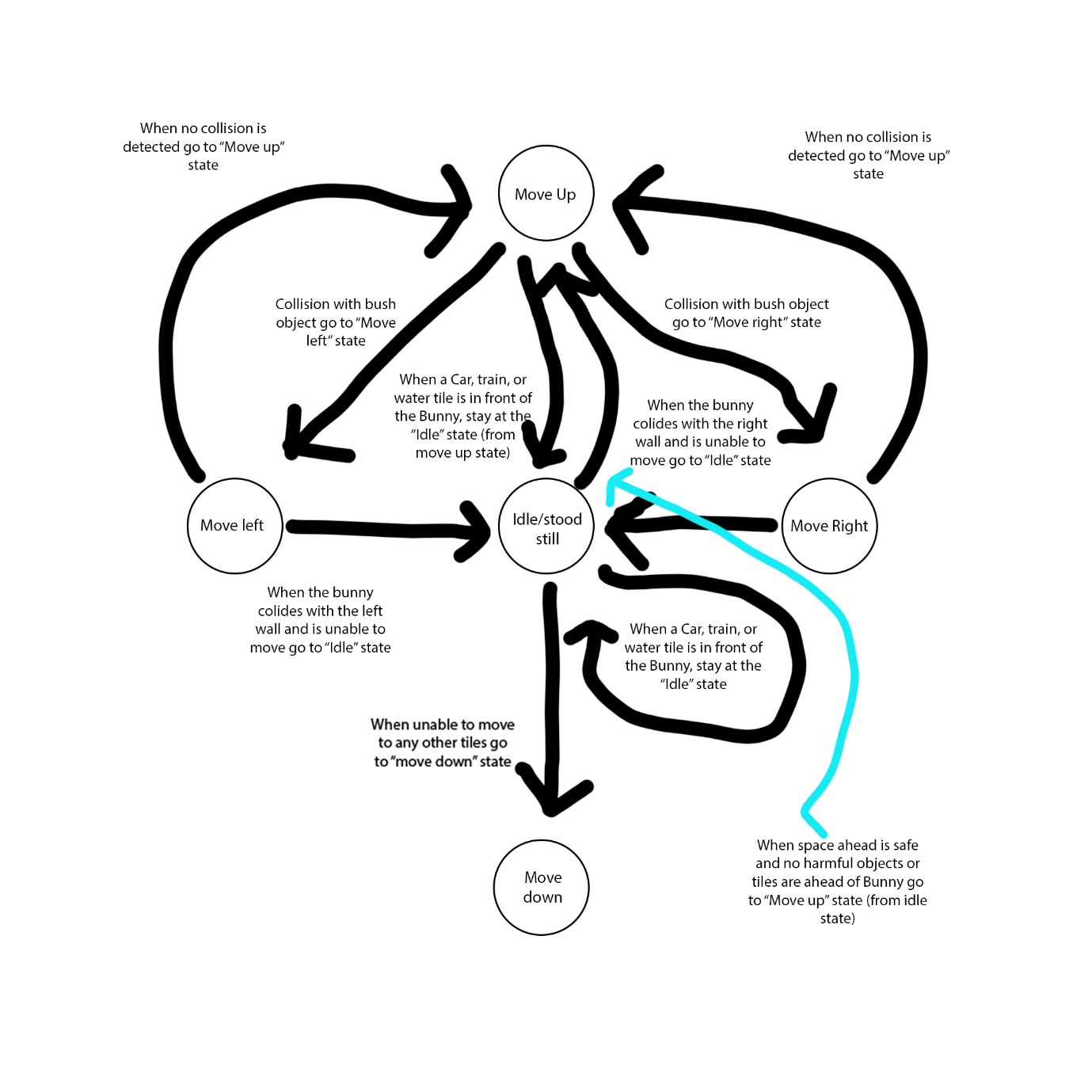
### FSMs used within the Computing industry

Within the project and similar fields of artificial intelligence building, the FSMs are used in order to aid and express the multiple paths and states a programmable object can have. FSMs greatly boost the efficiency and overall power of objects by boosting things such as decision making within AI that is built around that.

### FSM used within my Bunner AI development project

Below I have created my own FSM diagram to demonstrate FSMs being used within my own project for the game Bunner. As can be seen below I have annotated each of my states that the Bunny can be in within the Bunner game and I have demonstrated what causes each state to change.

(The blue arrow is not a part of the FSM it just shows where that specific text box is meant to be as I ran out of space on my canvas and could not make the whole diagram bigger).



## FSMs vs A\* search algorithm

Within my project I have decided to use FSM over the use of the A\* search algorithm as the FSM method greatly benefits the way my game and AI will function, the Bunny has to make decisions based off of given scenarios and due to this being in effect, it will have to change states constantly after being given a procedurally generated Map that will constantly change.

If I were to use the A\* search algorithm in order to attempt to create an AI for the game given to me it would be extremely difficult as the A\* search algorithm is significantly more useful when the AI and program knows exactly what it has been given, and when it has a set goal, such as finding the end of a maze etc; however as the main goal of the Bunner game is to survive through an endless flurry of obstacles and challenges that are thrown at random towards the AI, it is due to this fact; that the FSM method is the significant and superior method that should be used while tackling this challenge, hence why I have used it within my program and incorporated the decision making and state changing of FSMs into my AI.

# Testing of AI and implementation of techniques used in the code

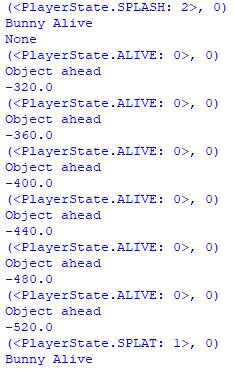
## Movement

### Test description

This test will determine weather the Bunny is able to move alone by itself using the AI.

Within this simple test I will output the Y Co-ordinands of the Bunny into the shell that way I am able to see if it is able to move freely without input from a controller.

### Test screenshots



### Result/improvements?

The results are successful and the bunny is able to move freely within the rules I have set for it within my code, however due to the limitations of my rules the bunny will never jump backwards.

The Y value displayed will only ever update when the Bunny interacts with an object due to the location in my code in which I have placed the function that displays the current location (on the Y axis) of the bunny.

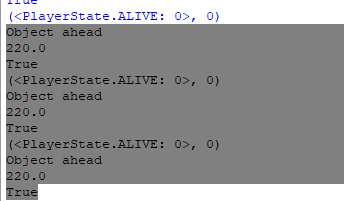
## Object detection

### Test description

Test in order to see if the bunny is able to see objects ahead of itself.

Within this test I will output whenever the Bunny is able to see an object that is 1 tile ahead of itself into the shell in order to see if the bunny AI is able to detect objects ahead of itself.

### Test screenshots



### Result/improvements?

The tests are a success the bunny AI is able to detect objects ahead of itself before it collides with them and this can be seen within the python shell in the screenshot above.

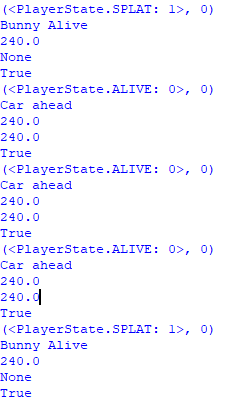
## Stopping before a Car collision

### Test description

Testing to see if the AI is able to stop before a Car and to see if it continues its path once the car has past the AI’s field of view ahead.

Within this test I will output whenever the Bunny is able to see a car into the shell, the bunny should stop its current path before the car hits it and should then continue its path upon the car passing it.

### Test screenshots



### Result/improvements?

The Bunny recognises the car ahead of itself during testing and stops during the amount of time it takes before it passes it.

Due to the way I have programmed the Bunny AI vision it is only able to see a car 1 tile ahead of itself and will not see any cars that are to the left or right of itself usually causing the Bunny AI to dive directly into an unseen car.

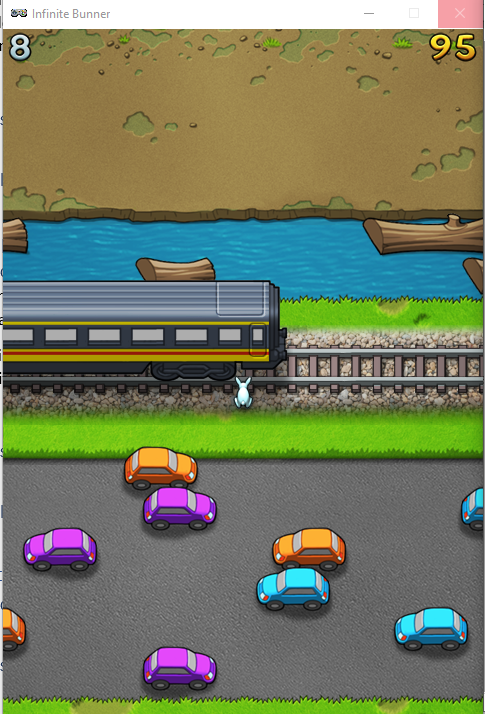
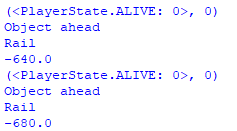
## Stopping before a Train collision

### Test description

Testing to see if the AI is able to stop before an oncoming train and to see if it continues its path once the train has past the Bunny AI view.

Within this test I will output whenever the Bunny AI is able to see a train into the shell, the bunny should stop its current path before the train hits it and should then continue its path upon the train passing it.

### Test screenshots

### Result/improvements?

As can be seen above the AI waits until the train has passed from being in view ahead of the bunny and the bunny safely continues its path, therefore making this test a success.

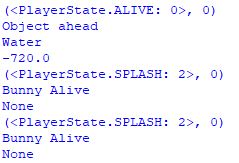
## Waiting for Log to appear in the water (Not immediately jumping into the water)

### Test description

Within this test the bunny should wait upon seeing the water ahead of itself and should jump onto the log when possible.

In order to test this I will output the current tile ahead of the bunny in order to see what it is able to see and I will wait until the bunny passes along onto the log in order to see if my test was successful or not.

### Test screenshots



### Result/improvements?

As can be seen within the screenshot the test was successful, the bunny stopped at the water area and waited until a log came into its jumping distance.

Unfortunately an issue I had with this test was that the bunny occasionally gets stuck with no other paths to hop to as no other logs would spawn ahead of the bunny so it would be forced into the water and unable to move backwards as I did not have it programmed to move backwards at all.

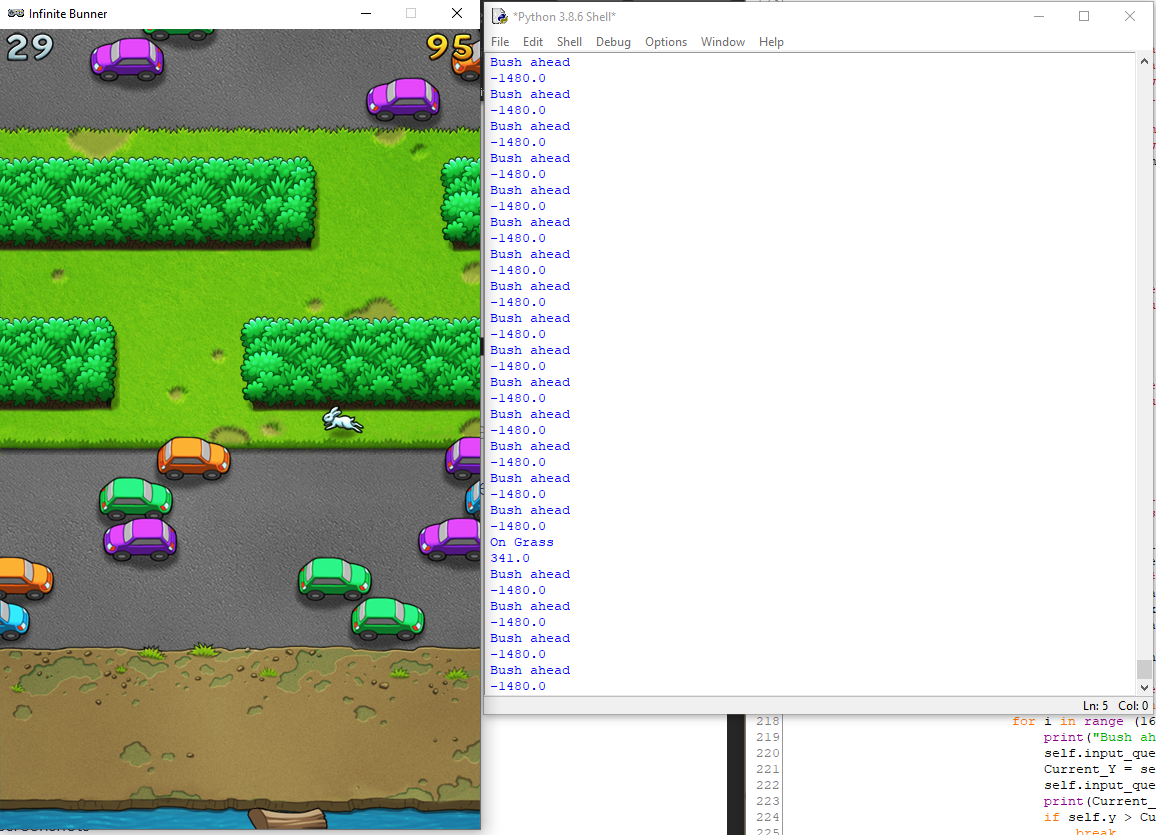
## Moving upon colliding with a bush

### Test description

Within this test the bunny should be able to navigate itself through the oncoming bush obstacle that has spawned.

In order to output test results, I will output the object detection of the Hedge into the Python shell and will observe the actions of the Bunny after.

### Test screenshots



### Result/improvements?

The bunny will travel to any gaps it is able to find successfully under the situation that the gap is located towards the left side of wherever the bunny is currently located, however an error and issue I have found within the AI is that it will unfortunately prioritise going left over going right, which results in the AI becoming permanently stuck in the left side of the screen. Overall, this test was a partial success however it could be greatly improved by fixing the issues stated within this report.

## Other Bugs or issues?

Within my Bunner program I had a few issues, one problem I came across was a bug that happened upon the Bunny passing through the Hedge section of the game, the AI would bug and would proceed to zig zag uncontrollably until the bunny died.

Another issue I had was with the Bunny being unable to pass through the hedge sections in a situation in which the exit to the hedge was to the right, the bunny will by default jump to the left of the screen and will proceed to get stuck once it reached the wall, I attempted to fix this issue by adding additional code to the AI that would force the Bunny to jump in the opposite direction in the event that it went too far to the left or right side of the screen and caused this issue, however I was unable to get that to correctly fix the issue and that in turn would leave the bunny in a loop in which it would jump back and forth between going left and right.

The Bunny was able to detect cars and trains that were already ahead of itself, and was able to act accordingly based off of that, however it was unable to detect cars and other obstacles that came from the left or right side of itself and act based off of that information, it is due to this factor that the bunny would jump directly into traffic and catch itself in a situation in which it would be unable to act accordingly towards the oncoming cars assuming that they were not directly on the tile ahead of the bunny.

# Source Code (with annotation)

## Code for Bunny Artificial Intelligence

## Coded and annotated By Tyraye Dennis-Mendez for The University of South Wales

##

def update(self):

# Check each control direction

## Variables I have made for the movement done by the AI

jumpDirection = 0

## Currently unused but was used earlier in development in order to help

## Locate and test which direction the Bunny would jump in.

Current\_Found = 0

Current\_Row = None

Next\_Row = None

## used in the Row checking section in order to check for tiles ahead of bunny and tiles it is currently on.

Current\_Y = None

Current\_X = None

Prev\_Y = None

i = 0

offset = 80

## Remaining variables that were either used in testing or other parts of the code to check locations of

## the Bunny or objects surrounding the Bunny.

## This section of my code finds the next row of the game and lets the bunny AI know what is on the

## the next tile ahead of itself.

for row in game.rows:

if Current\_Found:

Next\_Row = row

break

## Checks the Next row ahead of the bunny and stores it in a variable.

if row.y == self.y:

Current\_Row = row

Current\_Found = True

## Checks the Current row that the Bunny is stood on by using the bunnys Y co-ordinate.

## Finite State Machine Implementation.

if Next\_Row:

NextState = Next\_Row.check\_collision(self.x)

## Checks for collision between the bunny and anything ahead of itself

print(NextState)

## Used for testing, shows the output of the bunny and lets me know which direction the Bunny

## Next hops in.

PlayerNextState = str(NextState)

#if str(NextState) == "SPLAT":

#print(NextState)

if(PlayerNextState.find("ALIVE") == -1):

print("Bunny Alive")

## Checks to see if the bunny is alive and allows the rest of the code to continue to keep running until

## the Bunny dies within the game.

## Prints out the current state of the Bunny until it dies.

else:

#####Detection of what is ahead of Bunny and what bunny should do in each situation

## This is the Finite state machine implimentation part of my code.

## Below are multiple states which the AI will bounce between upon it coming upon

## an issue stated such as colliding with a bush.

## Removed old movement system and changed it to the code below as it better implemented

## the FSM into practise and worked a lot better and consistently than my previous code.

rowType = type(Next\_Row).\_\_name\_\_

self.input\_queue.append(0)

print ("Object ahead")

## Added this function for testing in order to output what the Bunny see's ahead of itself

## whenever there is an object such as a log, Car, Hedge, Train etc; the Bunny will stop moving

## and it will wait until the object has past before it continues its path forwards.

Prev\_Y = Current\_Y

Current\_Y = self.y

Current\_X = self.x

## Collision detection for when the Bunny goes too far to the left side of the screen

## the code below tells the Bunny to move in the other direction when it goes too far left.

if self.x <= 40:

self.input\_queue.append(1)

## Collision detection for when the Bunny goes too far to the right side of the screen

## the code below tells the Bunny to move in the other direction when it goes too far right.

if self.x > 415:

self.input\_queue.append(3)

## The section below is used in order to allow the Bunny AI to make decisions based off of

## the current situation it is in

elif rowType == "Grass":

for child in Next\_Row.children:

if Next\_Row.collide(self.x, 0):

##print("On Grass")

## Used to test to see if AI can see Grass tile ahead of itself

self.input\_queue.append(1)

Current\_X = self.x

self.input\_queue.append(0)

if self.x > Current\_X:

break

if self.y == self.y and type(child).\_\_name\_\_ == "Hedge":

for i in range (16):

##print("Bush ahead")

## Used to test to see if AI can see Bush ahead of itself

self.input\_queue.append(3)

Current\_Y = self.y

self.input\_queue.append(0)

##print(Current\_Y)

if self.y > Current\_Y:

break

## More print functions used within my testing

##print (rowType) ## Shows the tile the Bunny is currently on.

#print (self.x)

##print (Current\_Y)

##print (Current\_Found)