Implementation of A\* search and Finite State Machines (FSMs) algorithms

using python version of game “Bunner”.

The technique used in the assignment was Finite State Machine (FSMs).

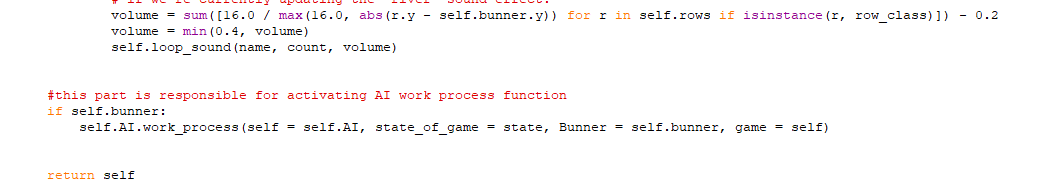
FSM is used to create a schematic of a closed/open system that will be functioning based on various states it is in. The way it is functioning is dependant on the state the machine/system is currently in. By changing the state we can change behaviour of system, allowing us to suit it to current needs.

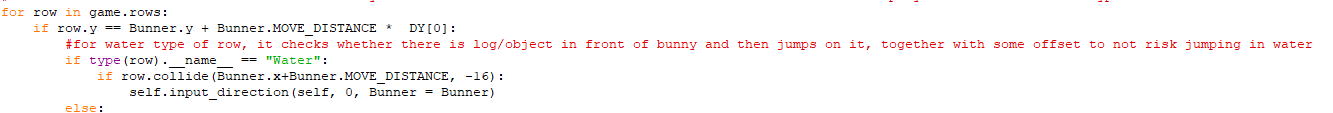
FSM allowed for easier creation of a system that will allow AI to answer to randomized environment that exist in “Bunner”. The game is infinite scroller in which you move your character against to the scrolling screen, trying to get as far as you can before you get to the bottom of the screen. Since the environment is randomized, demanding creation of an AI that will answer to the ever changing obstacles.

FSM allowed to easily design various states that were in this case mostly dependant on the are my character was, or was going to be soon. Depending on area, the way AI handled obstacles varied.

Looking at A\* search algorithm, It was possible to use it to create an additional system that could help me not only deal with obstacles as they appear like FSM already did, but also allowing to optimize route of AI to maximize points for distance. A\* search would allow for that since its an algorithm that based on a defined goal and initial state, uses created functions to achieve the goal in most cost optimized way, allowing in this case for safer and more efficient way of character movement. However since design of levels is random, algorithm would need to do more mathematical operations than FSM. A\* search would also be more complicated to create and design.

Included source code, that creates an AI.

Part in Game class that operates AI class:

Depending on what row was supposed to be next, AI uses implementation of nodes based on Python version of switch/case function that changes its behaviour depending on what part of level has to be dealt with:

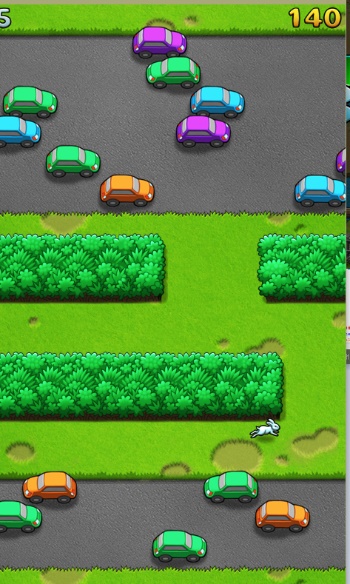
Similar nodes were used to deal with other types of terrain.

Full AI code in appendix

Game starting screen:



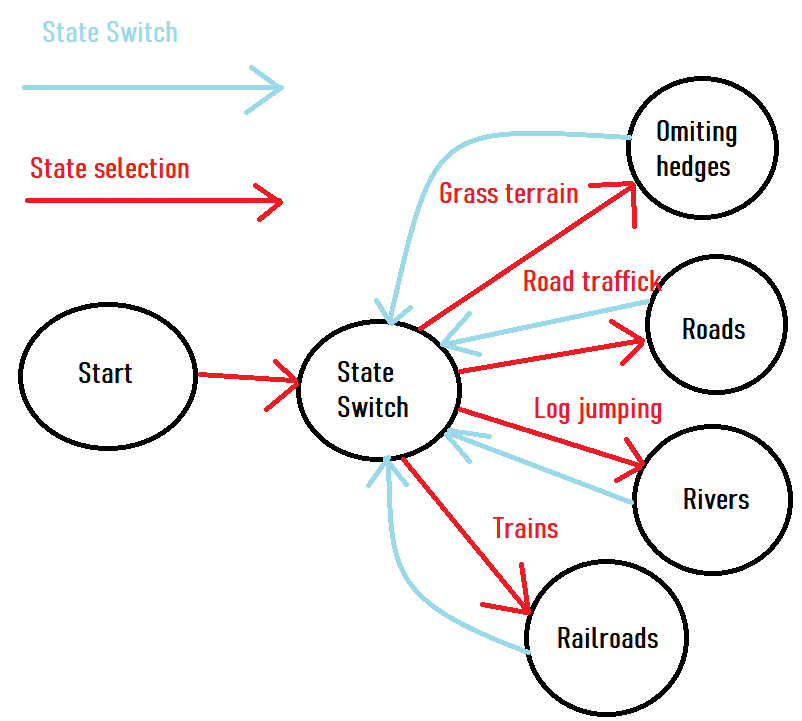
Implementation of node that deals with walking around hedges:



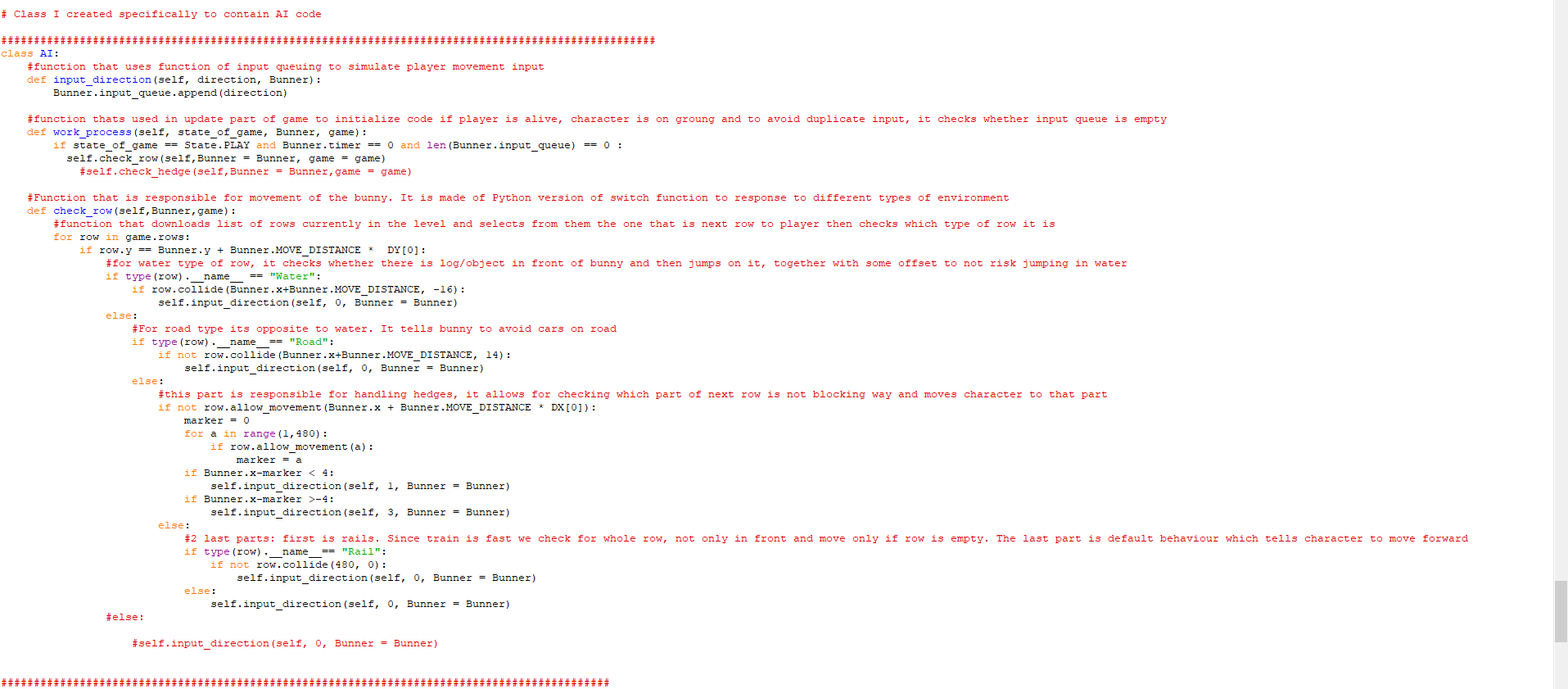
The project could have been made better if used also A\* search algorithm. Using it, AI could manage its vertical position so its better for traversing through roads and rivers. FSM is good algorithm that allows to create good quality first version of AI. If implemented together with A\* search in complex way, project would have to have added ability to control height in level so its optimal for decision making for avoiding obstacles and taking the shortest route through them.

Current level of implementation is working properly enough. AI happens to have hiccups sometimes with detecting collisions, walking by last pixels into cars. AI doesn’t know how to walk sideways on highways, making it harder to avoid cars.

Diagram showing simple Finite State Machine used for AI creation:



Appendix:

AI code: