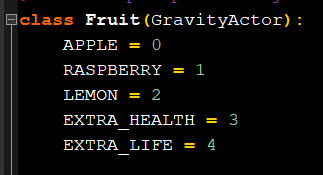
**(Re)Creating the Classics – 18101623**

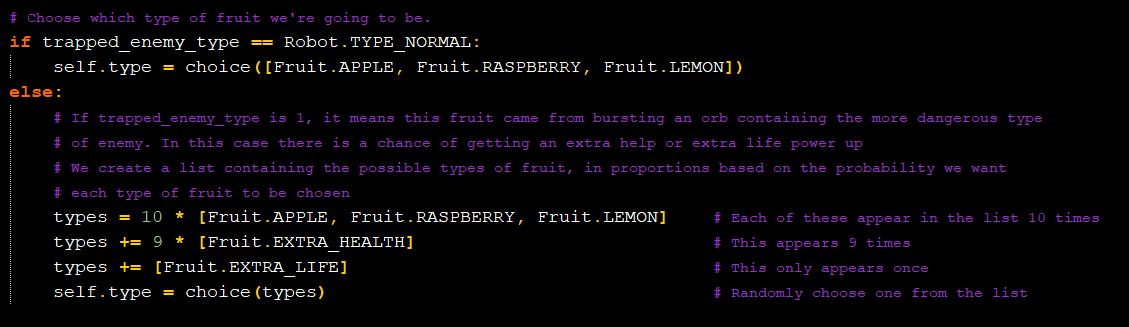
The next report will develop the use of different techniques of artificial intelligent applied in a game example. Among the different scenarios the chosen one is the Caverns game; Below I will explain the main characteristics and objectives of the game, which I use to choose and develop the correct AI techniques.

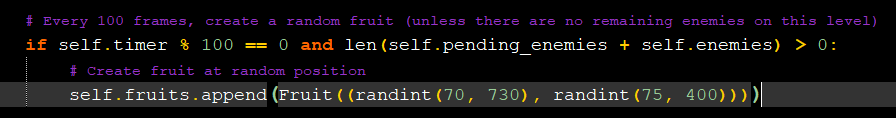
**SCENARIO ANALYSIS**

In Caverns, you have to obtain the maximum amount of points by collecting fruits with your player, and at the same time you have to try to avoid the enemy’s attacks. The game has three different levels which changes when a concrete number of enemies have been destroyed by trapping them in bubbles or by them reaching the end of the level. There are three core elements in the game, which are the fruits, the enemies and the player.

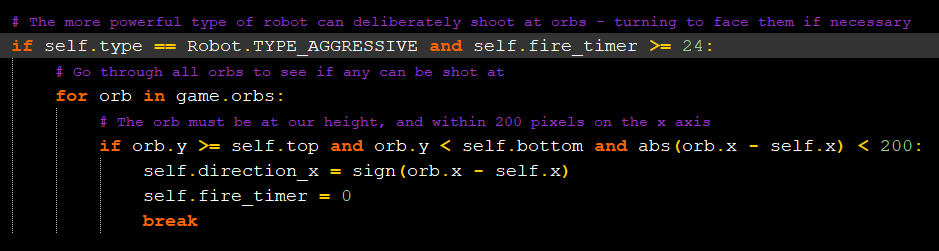
**Fruits:**

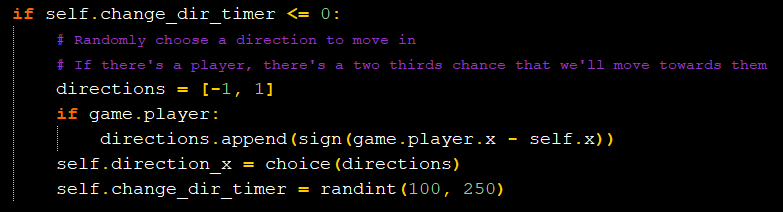
* There are 5 different types of fruits that you can collect in the game (apple, raspberry, lemon, extra health, and extra life).
* There are two different ways to spawn fruits, the first one is by trapping an enemy in a bubble and when it reaches the visual limit of the level a fruit is spawned, and the other one is to create a random basic fruit if there are still enemies in the level.

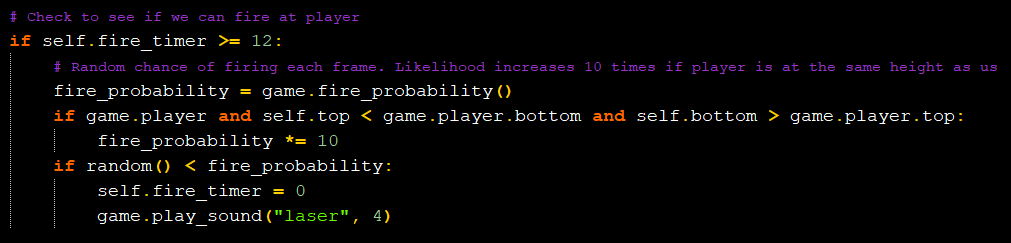


* As we can see in the previous picture if you trapped an aggressive enemy the possibilities of spawning a special fruit (extra health or extra life) are slightly increased. If the capture one is a normal enemy, or a usual fruit spawn, it will be a normal type (lemon, apple, raspberry).

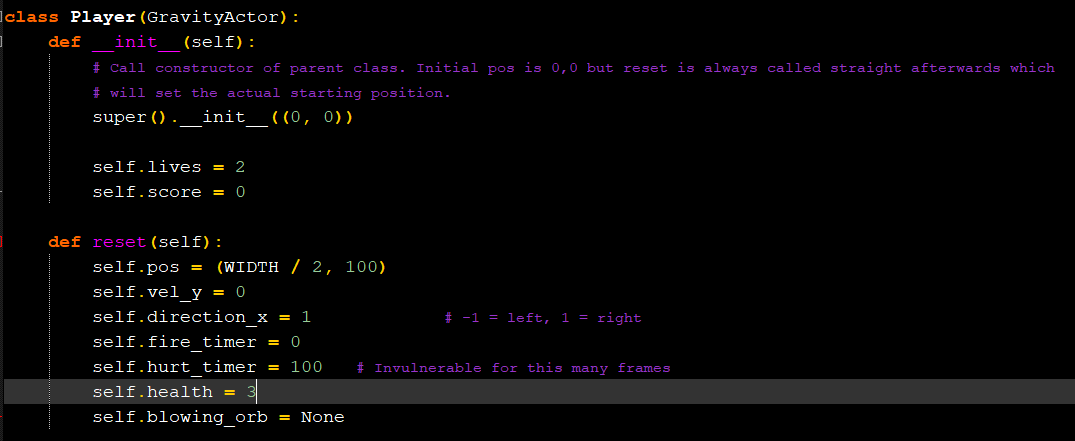
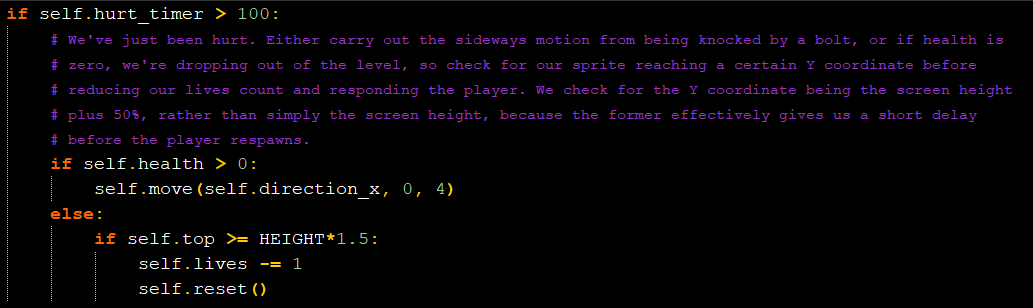
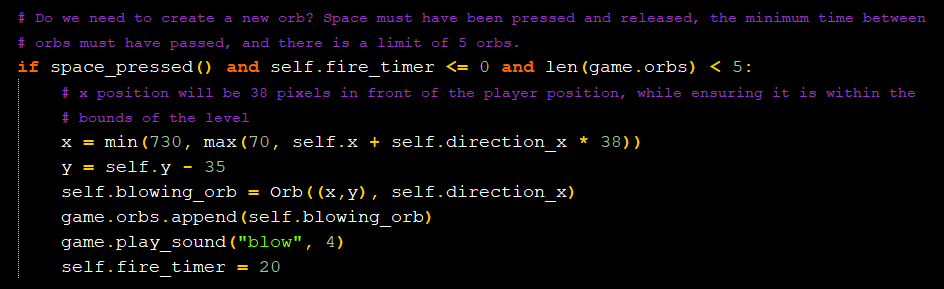
**Enemies:**

* There are 2 different types of enemies (normal and aggressive). The difference between both is that the aggressive one can shoot at your bubbles if they are close enough and destroy it.
* The enemies shooting and walking direction of the enemies is random, but they have a chance to move towards the player, or increments the shooting probability if they are in the same height.





**Player:**

* Among others as the player position, or the timer which controls the firing bubbles rate, our player has three important attributes, the number of lives, the total health of each live, and the score depending of the fruits that we collect, these variables control the main mechanics of the game which the AI should focus on solving.
* Each time that we get hit by an enemy shot we will lose health, if our health is equal to zero, we lose a live, and if our lives are equal to zero the game ends. We cannot be hit consecutively, there is a ‘safe time’ after being hit.
* Our player has two main mechanics running/jumping and create bubbles, the first one allows us to collect the fruits, change between the platforms, and avoid the enemy shots, and with the second we destroy enemy shots and capture enemies that are transformed in fruits later.

**SCENARIO SOLUTION**

After analysing the main objectives and mechanics of the game, now I will define the main tasks that the AI needs to complete, and the techniques that I will use to achieve it.

The objective of the AI is to get the maximum score, so, as a result of this there are two core tasks that we have to achieve, the first one is finding the fruits and collecting them, and the second one is to survive by avoiding or destroying the enemy shots.

My experience and the method that I have use to get the maximum score when I played the game various times is, that collecting the fruits is the highest priority task that we have to implement in our AI, capturing enemies with bubbles, and enemy shot protection is not so important because we have a very fast firing speed, unlimited ammo, and the bubbles go up, that causes that we can create a lot of bubbles in our way to the fruit, that protects us from getting shot and when they go up is very probable that they capture enemies.

So, we will create an AI which will prioritize finding and collecting fruits and will use the bubble mechanic as protection, the downside of this method is that we cannot walk as the same time as we shoot, which slow down the collecting fruits task but is not a big problem since we do not have a specific time to do it.

The first technique that we will use to implement our AI is a machine state which will define and control the different states that our player can have.

This FMS consist of the following states and transitions:

* **States:**

Search Fruit: This state will be in charge of searching the next fruit that the player will collect, to do this all the fruits spawned will be stored in an array, and the target fruit will be the closest to the player when the searchFruit() method is called.

Move: This state will be in charge of moving our character in the direction of the target fruit, ideally the target fruit will be the nearest to the player.

Create Bubble: This state will be in charge of creating bubbles every little time to protect the player from the enemy attacks.

Die: This state will be in charge of the player death.

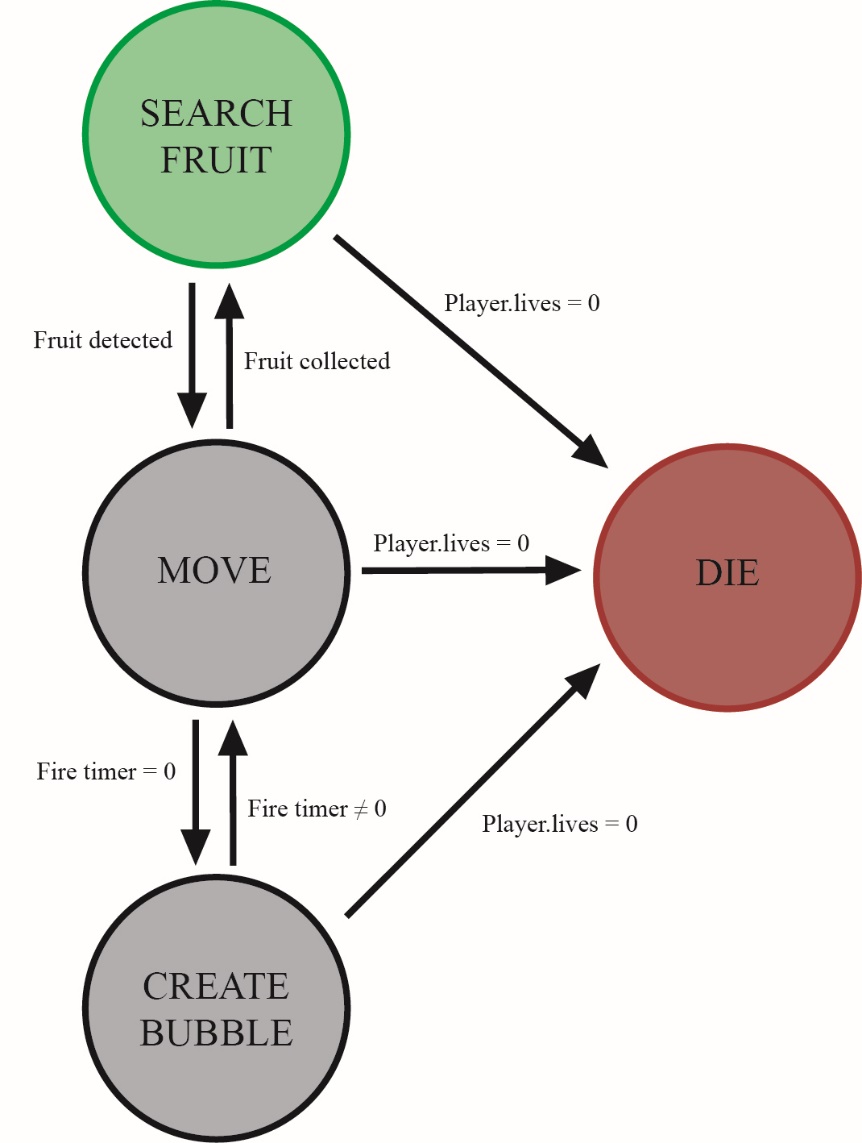
* **Transitions:**

Fruit Detected: This transition changes the state to ‘Move’ when a fruit is set as a target.

Fruit Collected: This transition changes the state to ‘Search Fruit’ when a fruit has been collected.

Fire Timer: This transition changes the state between ‘Create Bubble’ and ‘Move’ when player.FireRate is equals or different to zero.

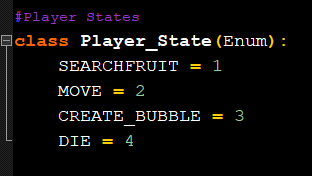
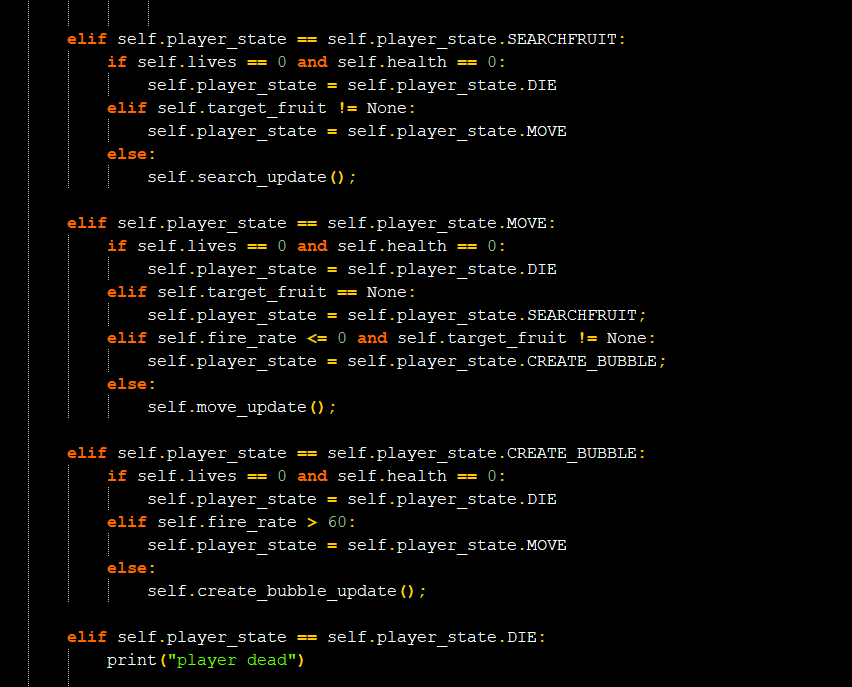
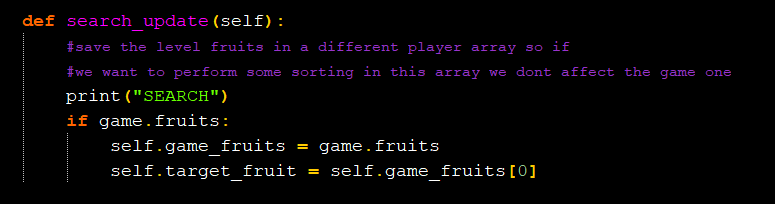
Player.Lives: This transition changes the state to ‘Die’ when player.Lives is equal to zero.

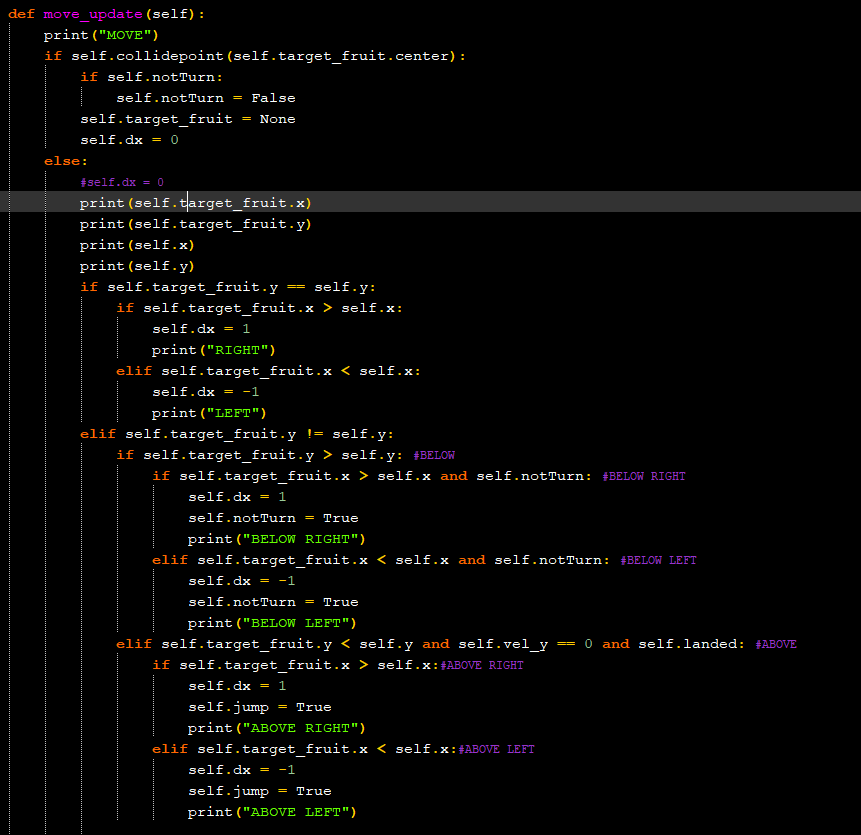


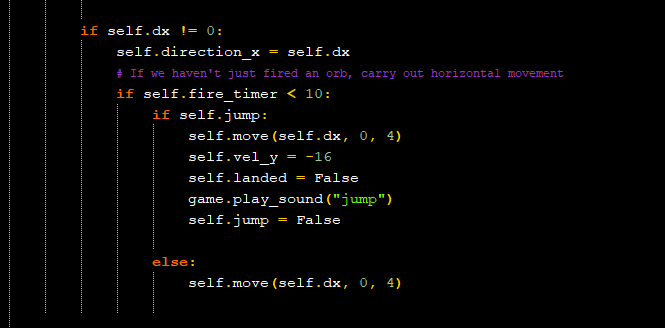
**AI CODE EXAMPLES**

This are parts of the code which implements the AI explained above, the final performance is not perfect.

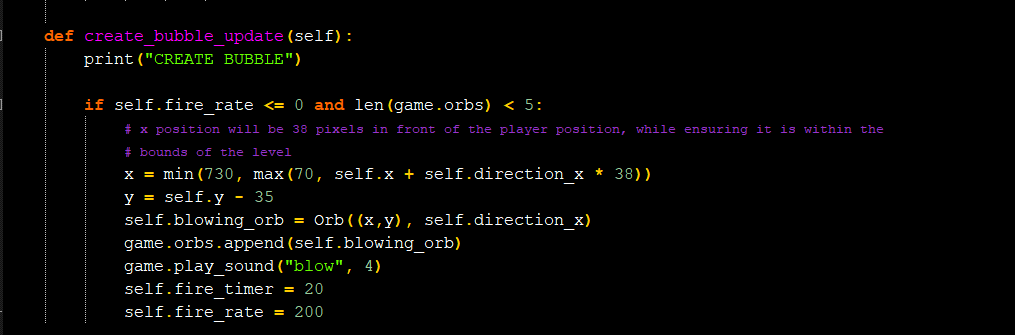
PLAYER FMS

* These are the states that the player can have.
* This is the implementation of the state machine in the player class, depending of the current player state it executes one update or another.
* This is the search update in which I find the next fruit, in this case it just stores in the variable target fruit the first fruit created, and when the player collects it, it refreshes the array and set the target fruit with the first position of the fruits array again.
* This is the move update that controls the player movement to the target fruit, and controls when the player collect the target fruit. Note: Sometimes the player gets stuck at start or half game, if you run it different times, some of it works collecting all the possible fruits until the player dies and other it bugs.





* This is the bubble update which creates random bubbles each few second.



* The dead state is not implemented only prints “Player Dead” and it was used as end state, the functionality related with the END GAME is done by the game state machine.