

Snake game

Phase 1

Mohamed Nasser	20200480
Lilian Stephanos	20200404
Natalie Monged	20200586
Riyad Abdelmenam	20190212

➤Stage#1 (Planning & Analysis):

Problem definition and planning:

The Snake game is a classic arcade game where the player controls a snake that moves around a board collecting food while avoiding walls and its own body. The objective is to collect as much food as possible without crashing into anything. The planning process for creating a Snake game involves designing the game mechanics, graphics, user interface, and levels.

Environment type:

The environment in Snake game is a 2D grid that represents the playing board. It contains walls, food, and the snake's body. The environment is dynamic, as the snake moves and interacts with the food and the walls.

Agent type:

The agent in Snake game is the player who controls the snake. The agent's objective is to collect as much food as possible while avoiding walls and its own body.

PEAS:

Performance measure:

The performance measure for the Snake game could be based on the player's score, which is the number of food items collected. The higher the score, the better the performance.

Environment:

The environment in the Snake game is a 2D grid that contains walls, food, and the snake's body. The environment is dynamic, as the snake moves and interacts with the food and the walls.

Actuators:

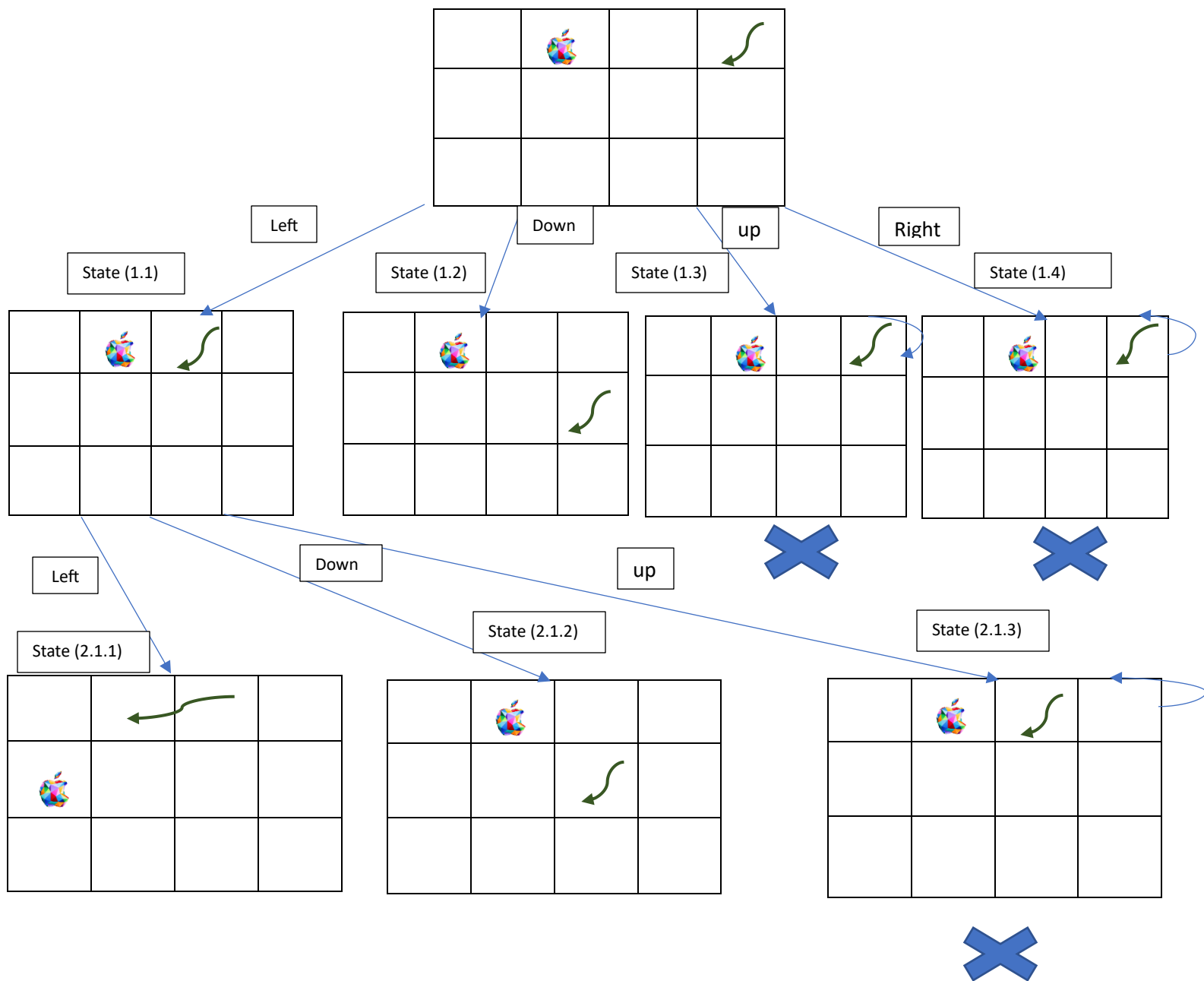
The actuators in the Snake game are the player's keyboard inputs, which control the direction in which the snake moves.

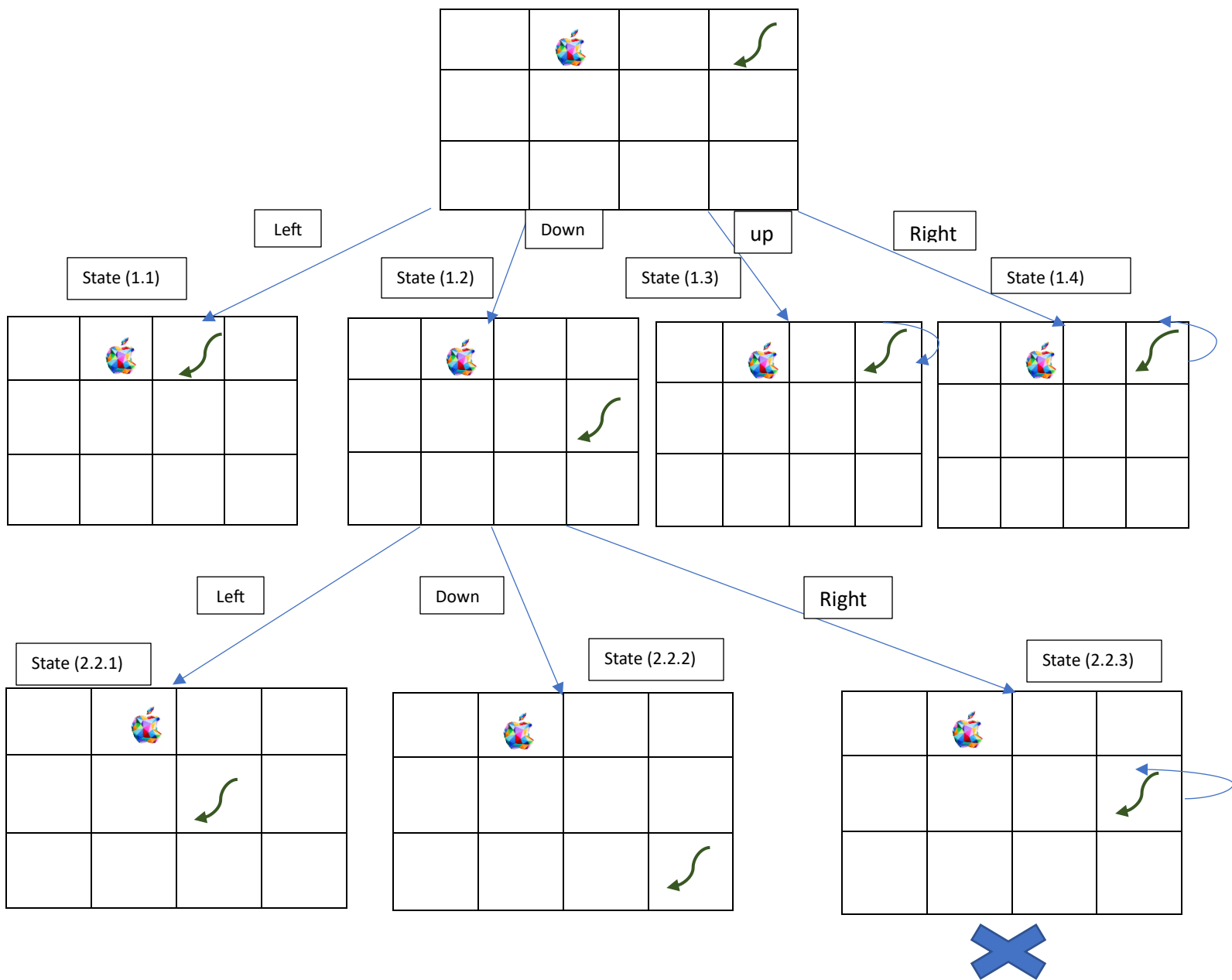
Sensors:

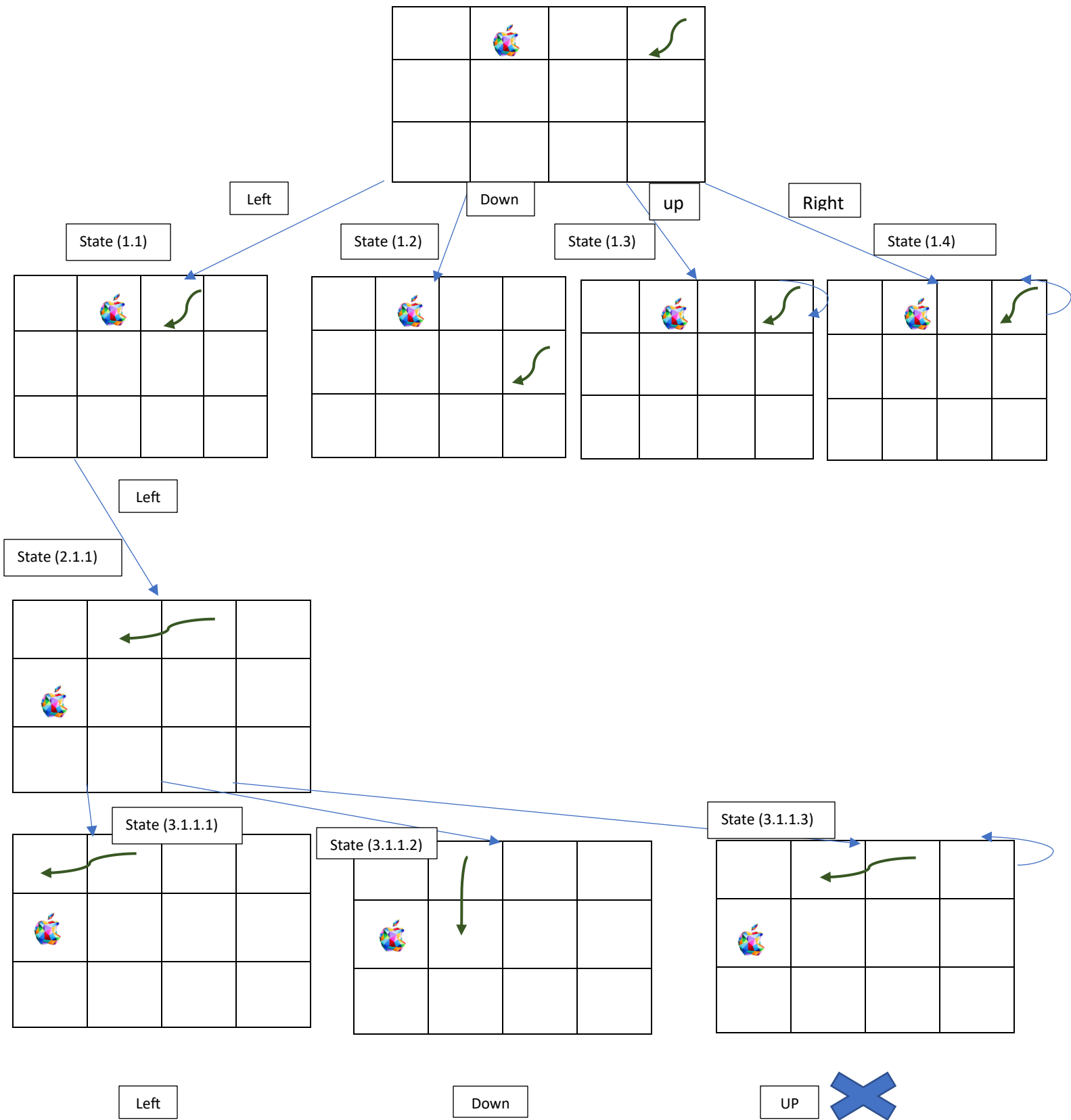
The sensors in the Snake game are the player's visual inputs, which allow them to see the board and the snake's movements, as well as the food items and walls. The player also needs to be aware of the snake's length and position, as well as the locations of the food items and walls, to make strategic decisions about the snake's movements.

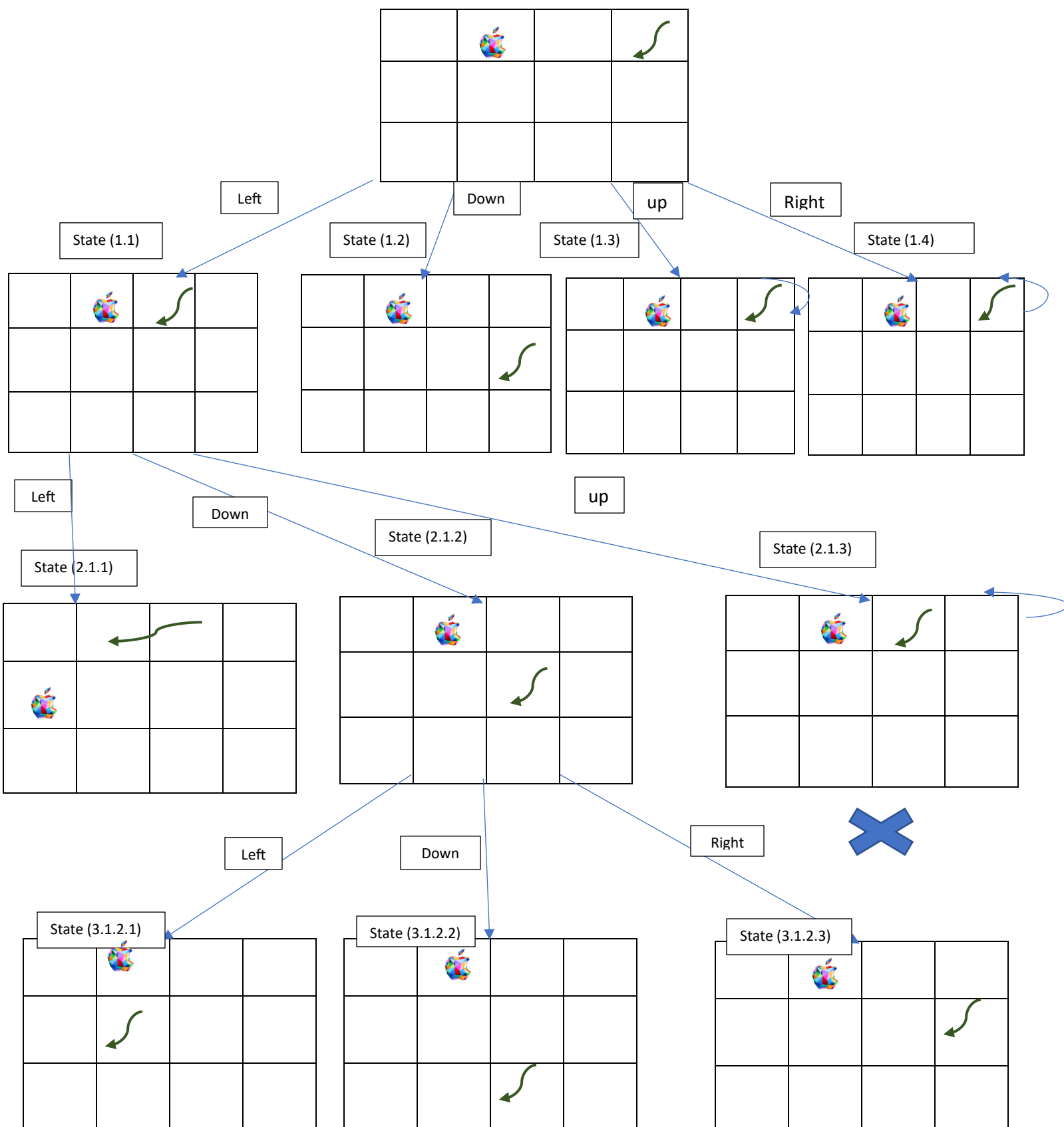
➤ Stage#2 (Design):

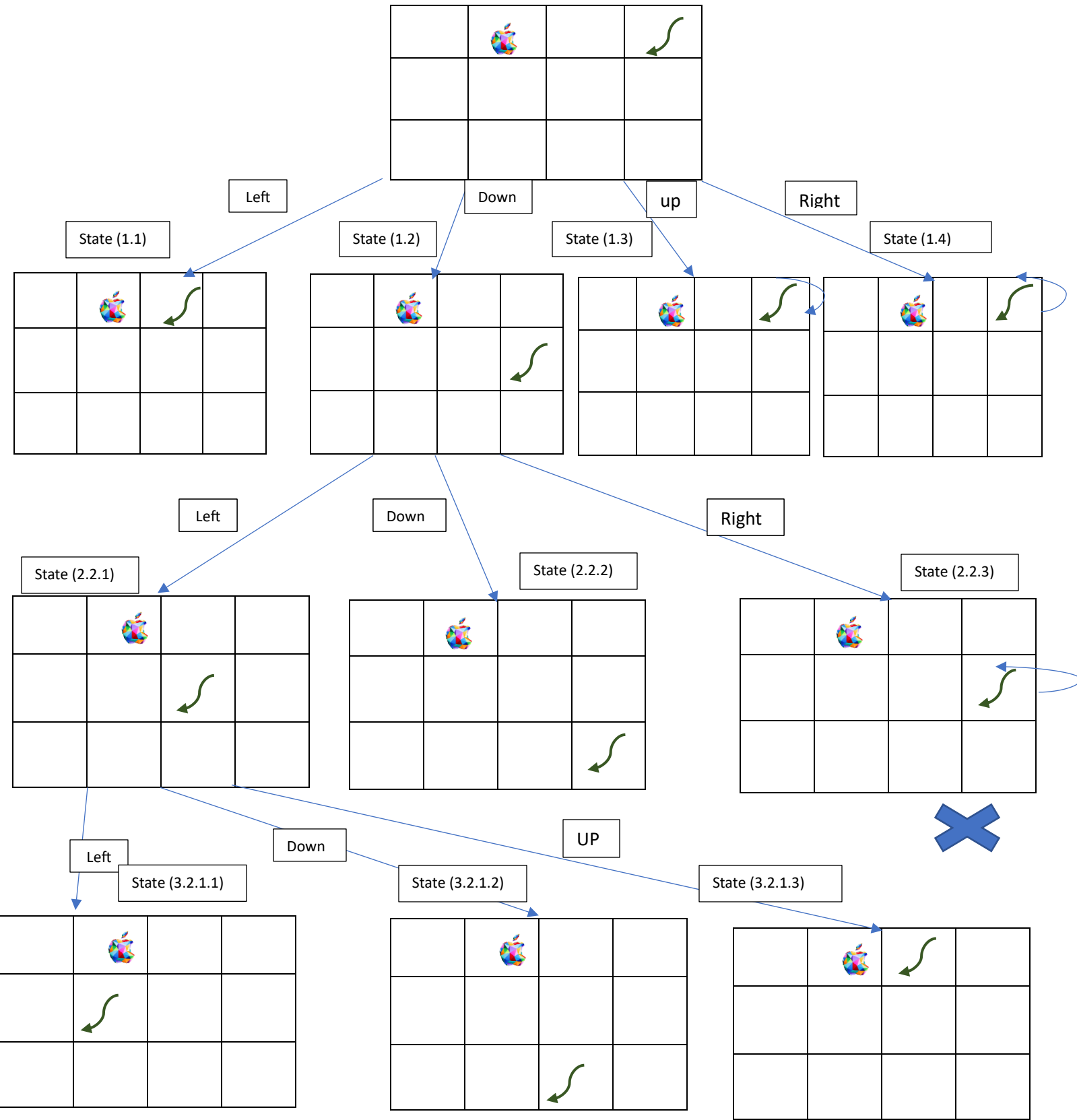
Problem state space:

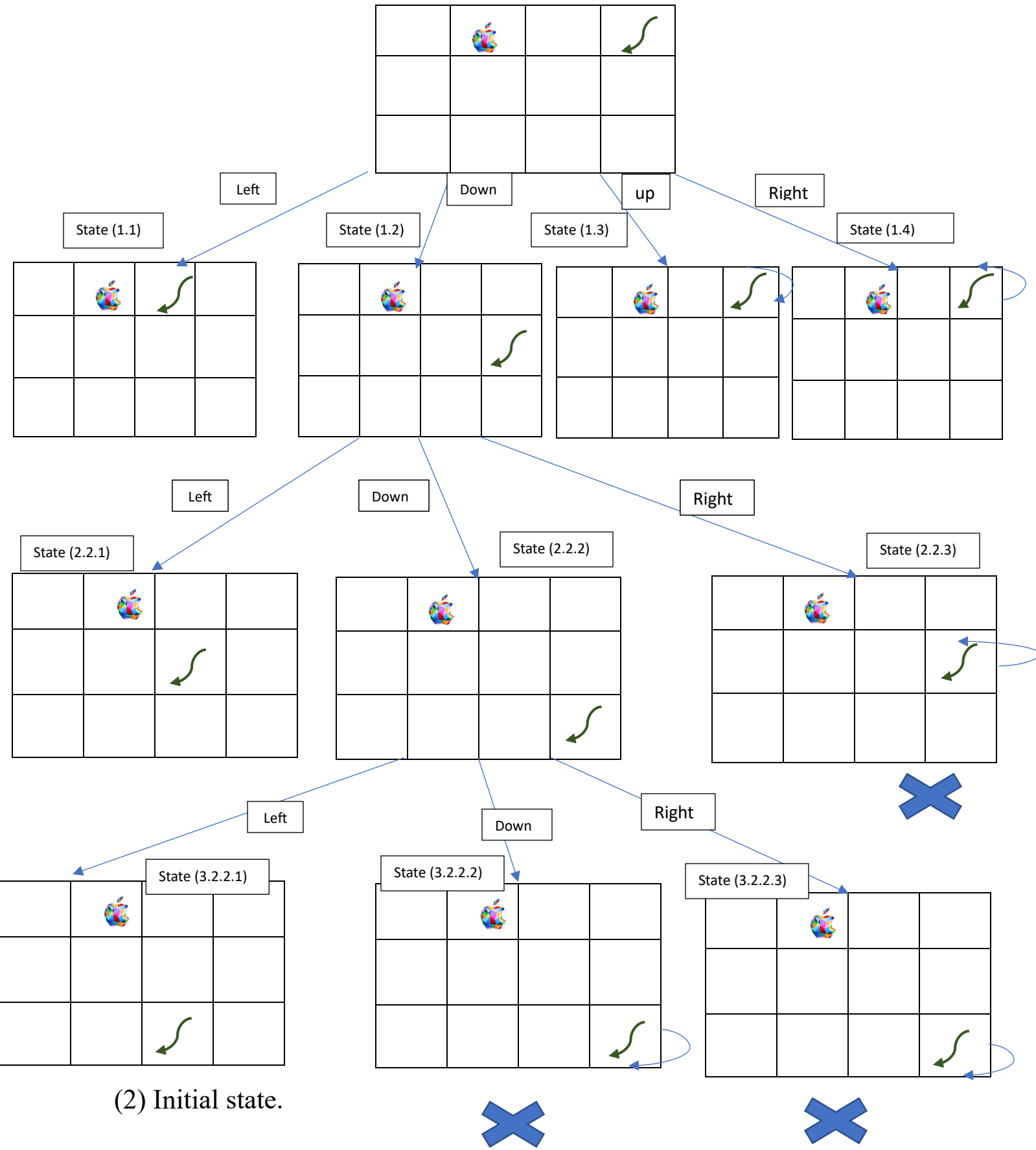


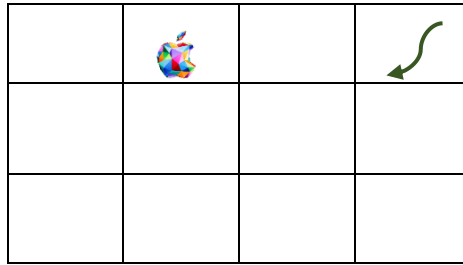












- A. Initial states, whether it is the place of the snake or the place of the apple, are random and not specific, but in our scenario, the snake starts from the top right corner, and the apple starts in the cell (0,2).
- B. In addition, as soon as the snake eats an apple, its size increases by one cell, and the location of the apple appears random in any cell (as shown in point No. (A solution/plan to show the transformation from initial state to goal state.))
- C. Our main goal is Maximize the score but considering that the length of the snake increases and that once its length reaches the size of all the cells, it will not be able to eat more than that, and therefore it will end up either biting itself or colliding with the border.

Goal test:

To verify if the game has reached a goal state, we need to check whether the snake's head has collided with any part of its own body or with the edge of the board. If the snake's head collides with any part of its own body or with the edge of the board, the game is over, and the goal test has failed. Otherwise, if the snake has collected all the food items on the board, the game is won, and the goal test has passed.



Successor function:

The successor function in the Snake game is defined as the set of possible actions that the player can take at any point in the game, and the resulting new state after acting. The actions are limited to moving the snake up, down, left, or right, and the resulting new state depends on whether the snake moves into an empty cell, a cell with food, a cell with its own body or the edge of the board.

For example, if the snake moves into an empty cell, the new state will have the snake's head in the new cell, and the last segment of the snake's body removed. If the snake moves into a cell with food, the new state will have the snake's head in the new cell, the snake's length increased by one, and a new food item placed on the board in a random empty cell. If the snake moves into a cell with its own body or the edge of the board, the game is over, and the state remains the same.

The cost associated with each action is defined as the distance travelled by the snake to reach the new state. For example, moving the snake up or down incurs a cost of one, while moving the snake left or right incurs a cost of one. If the snake collects a food item, the cost of that action is zero.

Transition Model:

(3)	(2)	(1)	(0)	
				(0)
				(1)
				(2)

Level 1:

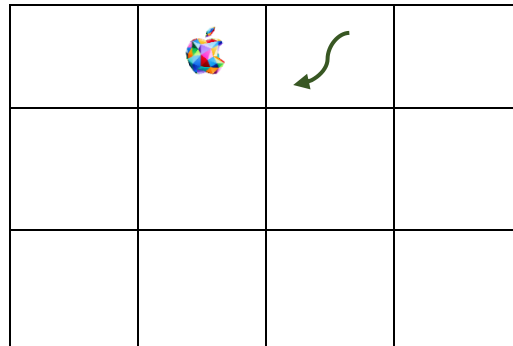
1.1 - From the initial state that the snake in the top right corner (0,0) and the apple is in position (0,2) **with the action [left]** the state will change to the state in which the snake will be in the position (0,1) and the apple is in the same position (0,2)

1.2 - From the initial state that the snake in the top right corner (0,0) and the apple is in position (0,2) **with the action [down]** the state will change to the state in which the snake will be in the position (1,0) and the apple is in the same position (0,2)

1.3 - From the initial state that the snake in the top right corner (0,0) and the apple is in position (0,2) **with the action [up]** the state will change to the state in which the snake will crash the wall “Game over state.”

1.4 From the initial state that the snake in the top right corner (0,0) and the apple is in position (0,2) **with the action [right]** the state will change to the state in which the snake will crash the wall “Game over state.”

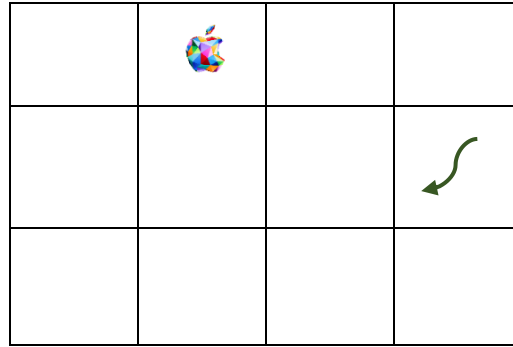
Level 2:



2.1.1 – from the state in which the snake is in the position (0,1) and the apple is in the position (0,2) **with the action [left]** the state will change to the state in which the snake will be in position (0,2) and will eat the apple and its length will increase to 2 and it will be in position (0,1) and (0,2) and the apple will appear in the other position randomly.

2.1.2 - from the state in which the snake is in the position (0,1) and the apple is in the position (0,2) **with the action [down]** the state will change to the state in which the snake will be in position (1,1) and the apple will be in the same position (0,2)

2.1.3 - from the state in which the snake is in the position (0,1) and the apple is in the position (0,2) **with the action [up]** the state will change to state in which the snake will crash the wall “Game over”

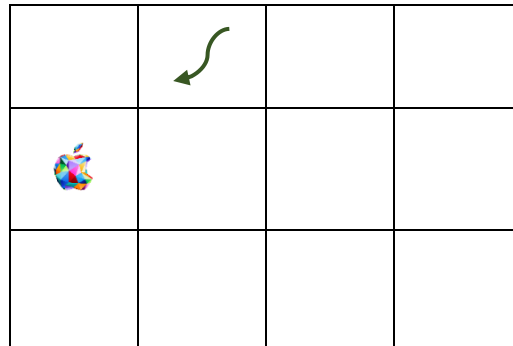


2.2.1 – From the state will change to the state in which the snake will be in the position (1,0) and the apple is in the same position (0,2) **with the action [left]** the state will change to the state in which the snake will be in position (1,1) and the apple will still in the same position

2.2.2 - From the state will change to the state in which the snake will be in the position (1,0) and the apple is in the same position (0,2) **with the action [down]** the state will change to the state in which the snake will be in position (2,0) and the apple will be in the same one

2.2.3- From the state will change to the state in which the snake will be in the position (1,0) and the apple is in the same position (0,2) **with the action [right]** the state will change to the state in which the snake will crash the wall “game over.”

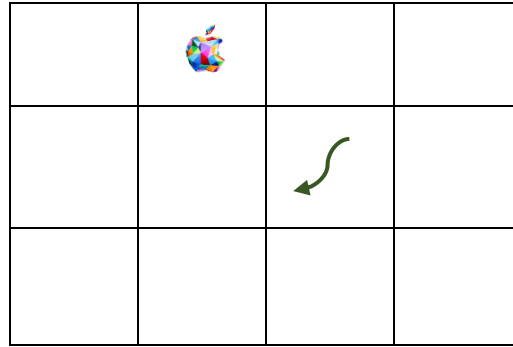
Level 3:



3.1.1.1 – From the state in which the snake is in position (0,2) and will eat the apple and his length will increase to 2 and it will be in position (0,1) and (0,2) and the apple will appear in the other position randomly assumption (1,3) **with the action [left]** the state will change to the state in which the snake will be in position (0,2) and (0,3) and the apple will stay in the same position

3.1.1.2 - From the state in which the snake is in position (0,2) and will eat the apple and his length will increase to 2 and it will be in position (0,1) and (0,2) and the apple will appear in the other position randomly assumption (1,3) **with the action [down]** the state will change to the state in which the snake will be in positions (0,2) and (1,2) and the apple will stay in the same position

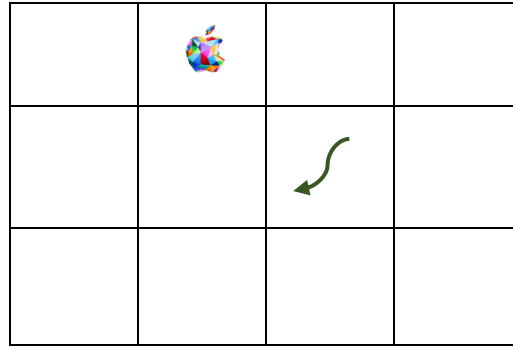
3.1.1.3 - - From the state in which the snake is in position (0,2) and will eat the apple and his length will increase to 2 and it will be in position (0,1) and (0,2) and the apple will appear in the other position randomly assumption (1,3) **with the action [up]** the state will change to the state in which the snake will crash the wall “game over.”



3.1.2.1 - from the state in which the snake will be in position (1,1) and the apple will be in the same position (0,2) **with action [left]** the state will change to the state in which the snake will be in position (1,2) and the apple will stay in the same position.

3.1.2.2 – from the state in which the snake will be in position (1,1) and the apple will be in the same position (0,2) **with action [down]** the state will change to the state in which the snake will be in position (2,1) and the apple will stay in the same position.

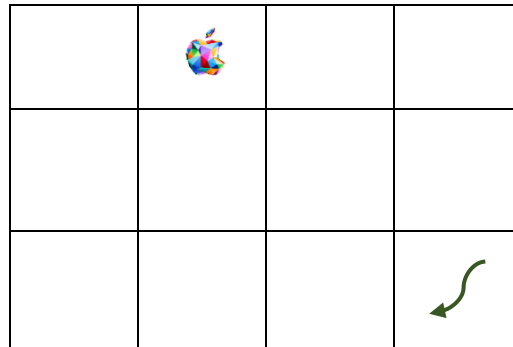
3.1.2.3 – from the state in which the snake will be in position (1,1) and the apple will be in the same position (0,2) **with action [right]** the state will change to the state in which the snake will be in position (1,0) and the apple will stay in the same position



3.2.1.1 – From the state in which the snake will be in position (1,1) and the apple in position (0,2) **with the action [left]** the state will change to be the state in which the snake will be in position (1,2) and the apple will stay in the same position.

3.2.1.2 - From the state in which the snake will be in position (1,1) and the apple in position (0,2) **with the action [down]** the state will change to be the state in which the snake will be in position (2,1) and the apple will stay in same position

3.2.1.3 - From the state in which the snake will be in position (1,1) and the apple in position (0,2)**with the action [up]** the state will change to be the state in which the snake will be in position (0,1) and the apple will stay in same position.

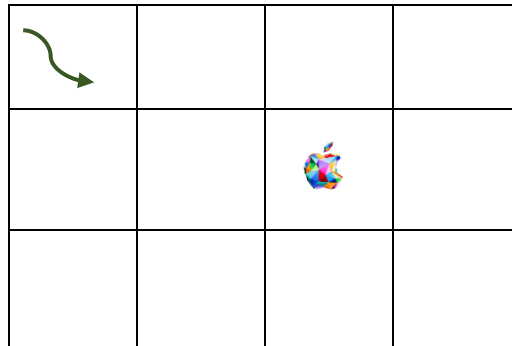


3.2.2 .1 –From the state in which the snake will be in position (2,0) and the apple will be in the position (0,2) **with action [left]** the state will change to the state in which the snake will be in position (2,1) and the apple position will stay the same

3.2.2 .2 - –From the state in which the snake will be in position (2,0) and the apple will be in the position (0,2) **with action [down]** the state will change to the state in which the snake will crash the wall “game over ”

3.2.2 .3 - –From the state in which the snake will be in position (2,0) and the apple will be in the position (0,2) **with action [right]** the state will change to the state in which the snake will crash the wall “game over.”

Plan to show the transformation from the initial state to the goal state:



-> Information:

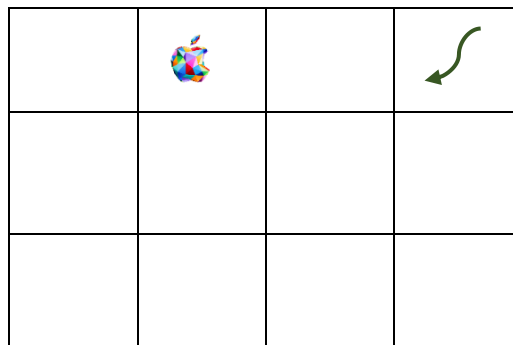
- Board size -> 3 * 4.

- There are walls around the board and the obstacles that the snake will face is the wall or its body.

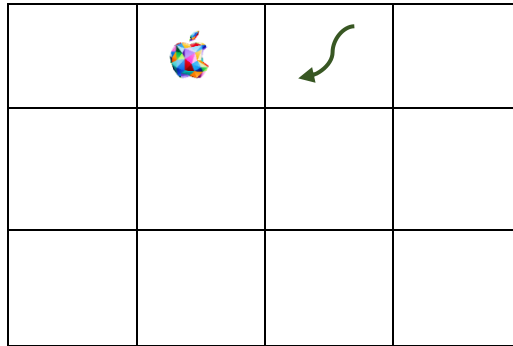
- Initial state: the snake is in the top right corner of the board with length 1 and the food is somewhere randomly.

- Goal state: the snake will fill all the board with its body as the length of its body will equal 12 and the score will equal 11 that is the best score.

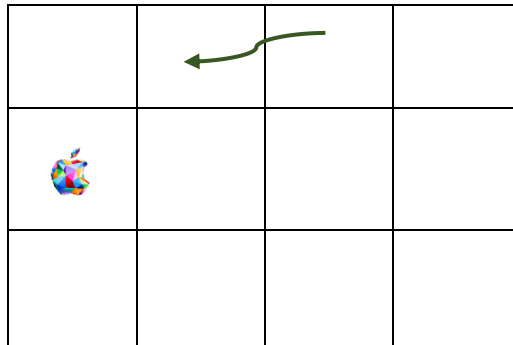
PLAN:



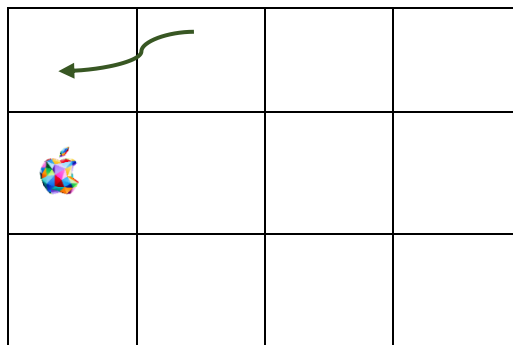
1. The snake starts in the top right corner of the board with length 1.



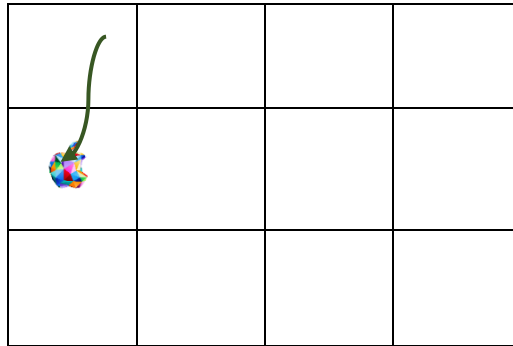
2. The snake moves to the left to reach the second column of the board.



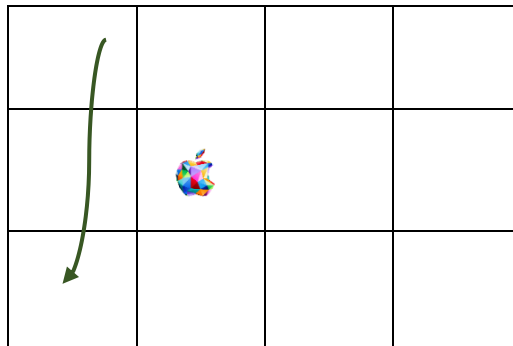
3. The snake moves another step to the left to reach the third column of the board and eats the food and grows by one unit.[2] The score increases by 1 [1].



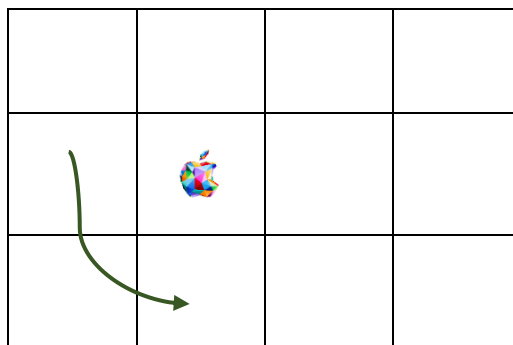
4. The snake moves another step to the left to reach the fourth column of the board.



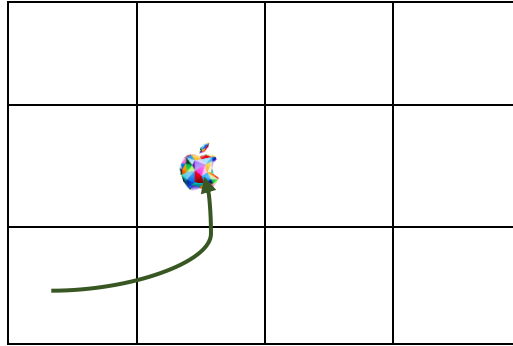
5. The snake moves down to the second row of the board and finds another one food and rats the food and grows by another unit [3] The score increases by 1 [2].



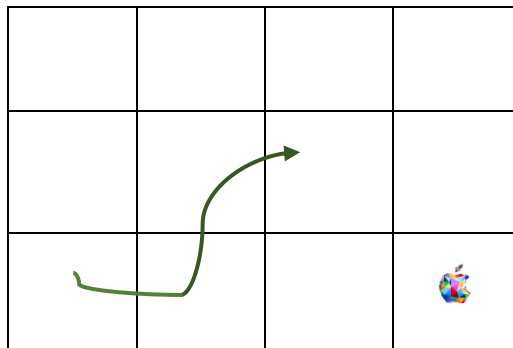
6. The snake moves down another step to the third row of the board.



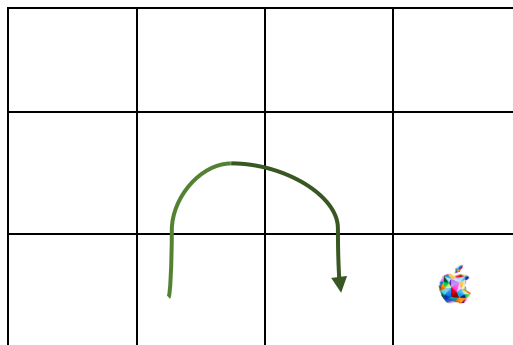
7. The snake move to the right to go back to the third column.



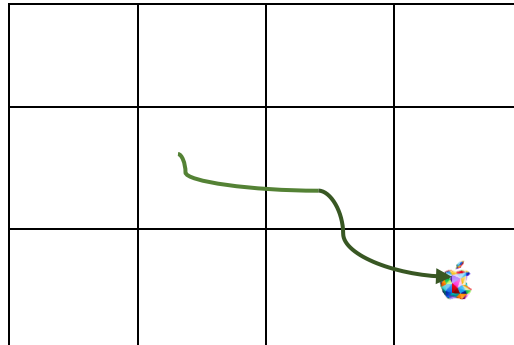
8. The snake moves up to the second row and finds food and eats it and grows another unit.[4] The score increases by 1 [3].



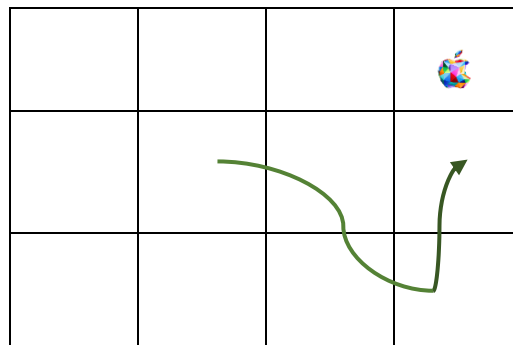
9. The snake moves to the right to reach the second column.



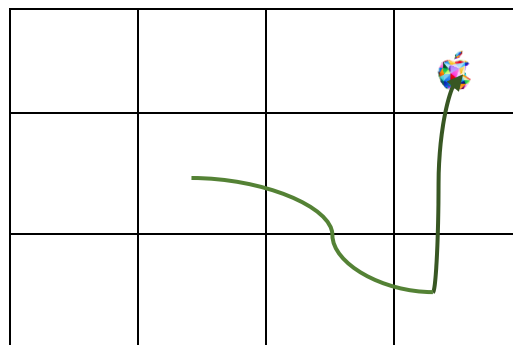
10. The snake moves down to reach the third row of the board. (2,3)



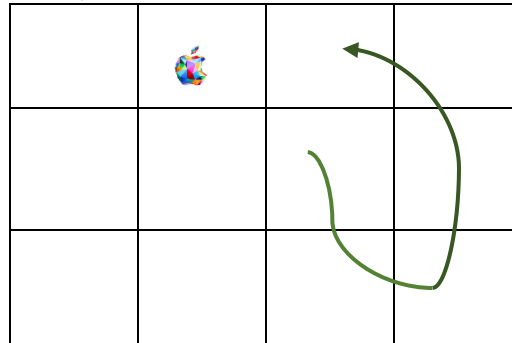
11. The snake moves to the right to go to the first column and finds food and eats it and grows one another unit [5] The score increases by 1 [4].



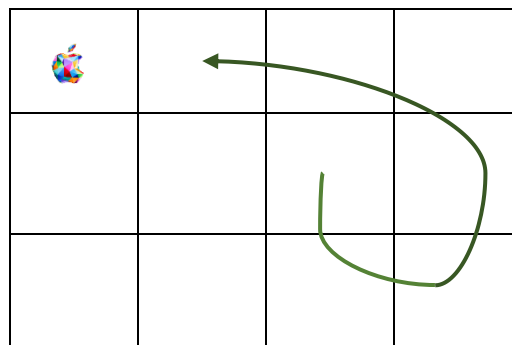
12. The snake moves up to reach the second row.



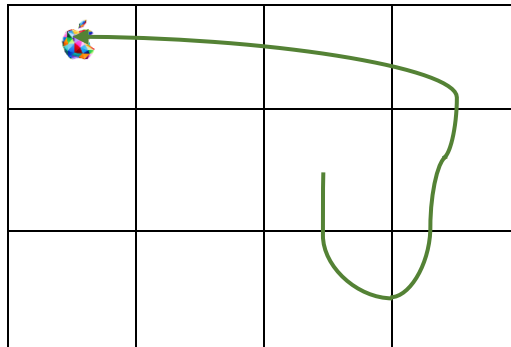
13. The snake moves another step up and finds food and feasts it and grows one unit [6] The score increases by 1 [5].



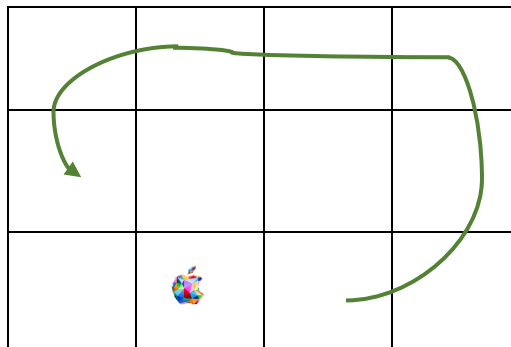
14. The snake moves to the left to reach the second column of the board.



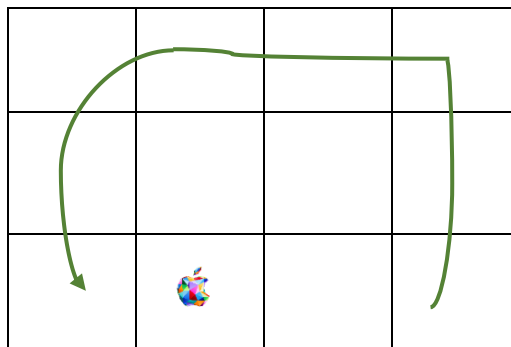
15. The snake moves another step left to reach the third column and finds a food and grows another unit [7] The score increases by 1 [6].



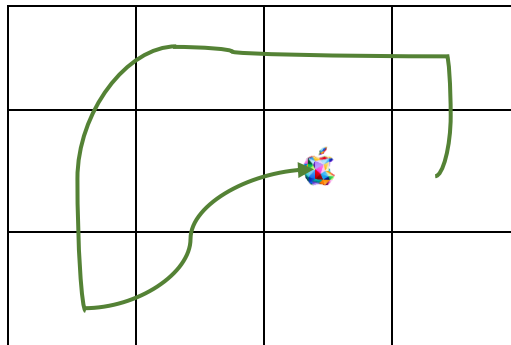
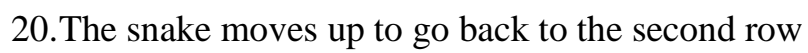
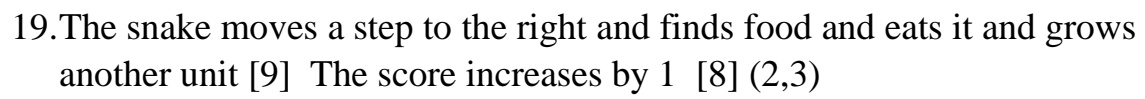
16. The snake moves another step to the left to reach the fourth column and finds food and eats it and grows another unit [8] The score increases by 1 [7].



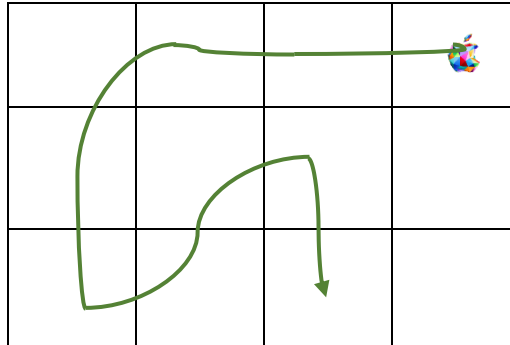
17. The snake moves down to the second row.



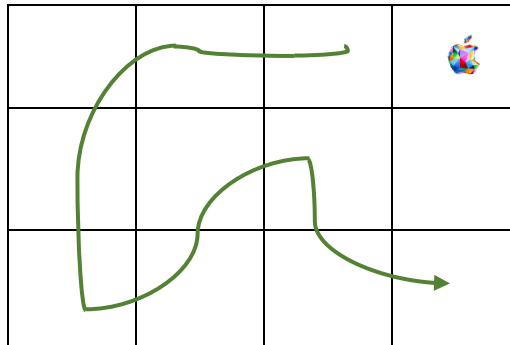
18. The snake takes another step down to reach the third row



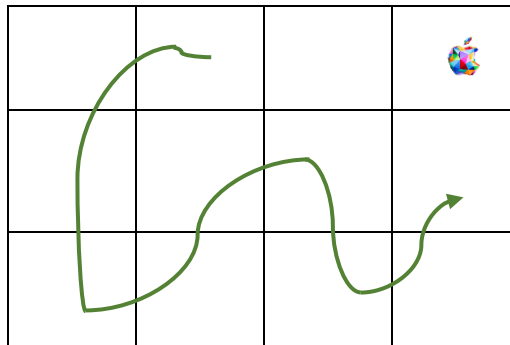
21. The snake moves right to reach the second column in the second row and finds food and grows another unit [10] (2,2)



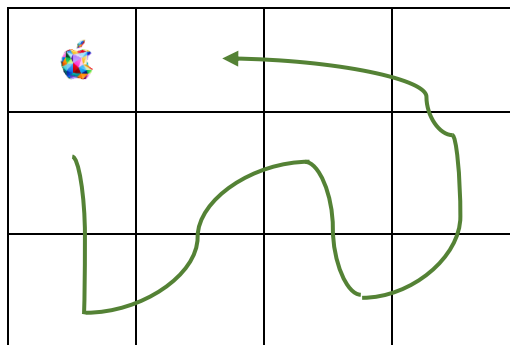
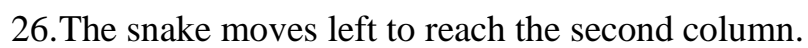
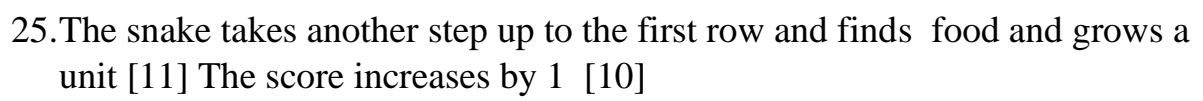
22. The snake moves down to reach the third row



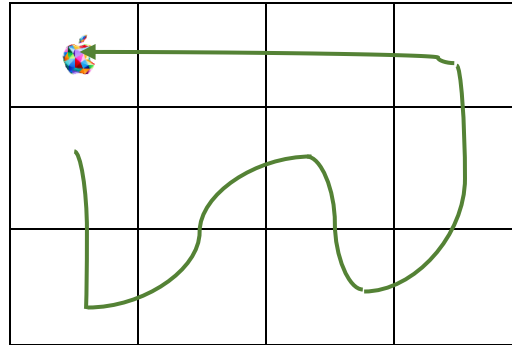
23. The snake takes a step to the right to reach the first column



24. The snake moves up to the second row



27. The snake moves to the left to reach the third column



28. The snake takes another step to the left to reach the fourth column and find a food and eats it and grows another unit [12] the score increases by 1 [11]

THE GOAL STATE

UML Diagram:

Snake
- body
- direction
- length
+ move()
+ grow()
+ change_dir()

Food
-position
-type

Board
- width
- height
- snake
- food
+ generate_food()
+ game_over()
+ play_game()

1 - The Snake class represents the player-controlled snake. It has private instance variables for its body, direction, and length. The class provides public methods for moving the snake, making it grow, and changing its direction.

2 - The Food class represents the food that appears randomly on the game board. It has private instance variables for its position and type (which determines its point value). There are no public methods in this class.

3 - The Board class represents the game board. It has private instance variables for its width, height, the current snake, and the current food. The class provides public methods for generating food, checking if the game is over, and playing the game.

4 - The Game class represents the overall game logic. It is not shown in the diagram but would use the Board class to display the game and handle user input. It would also use the Snake and Food classes to control their movements and collisions.