DELHI TECHNOLOGICAL UNIVERSITY

THEORY OF COMPUTATION (MC-304)

Midterm Innovative Project

SUBMITTED BY:

Aiman Siddiqua - 2K18/MC/008

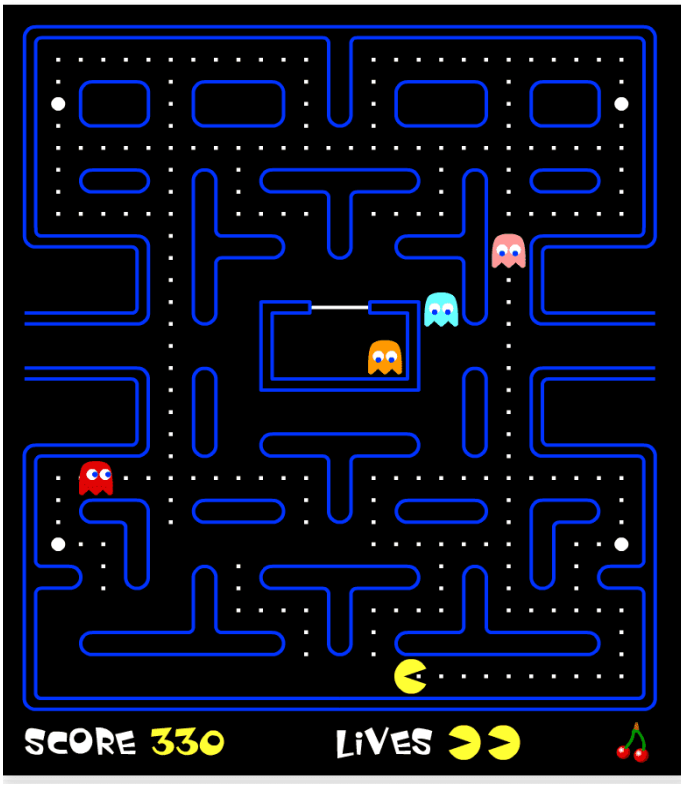
Apoorva - 2K18/MC/019

SUBMITTED TO:

Dr. Sangita Kansal

CONTROLLING THE DEAD

Modelling the behaviour of the Ghost NPCs in the Game ‘PAC-MAN’ Using Deterministic Finite State Machines



INDEX

INTRODUCTION

WHAT IS PACMAN?

The premise of Pac-Man is delightfully simple: the player guides Pac-Man-up, down, left, and right-through a maze filled with dots for him to gobble up. **Four ghost monsters** are also in the maze and chase after our hero, trying to capture and kill him.

The goal is to clear the maze of dots while avoiding the deadly ghosts. Each round starts with the ghosts in the "monster pen" at the center of the maze, emerging from it to join in the chase.

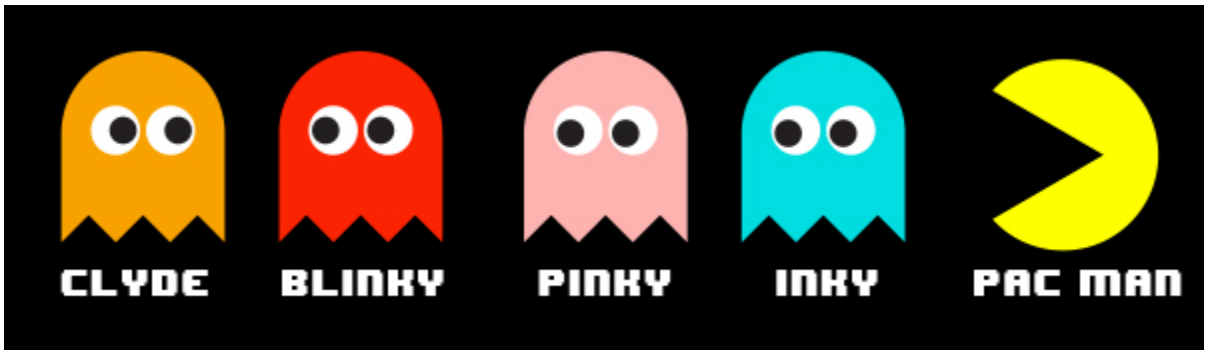
If Pac-Man is **captured** by a ghost, a life is lost, the ghosts are returned to their pen, and a new Pac-Man is placed at the starting position before play continues. When the maze is cleared of all dots, the board is **reset**, and a new round begins. If Pac-Man gets caught by a ghost when he has no extra lives (**three in total**), the **game is over**.

Placed at the four corners of the maze are large flashing "**energizers**", or "**power pellets**". Eating these will cause the ghosts to turn blue with a dizzied expression and reverse direction. Pac-Man **can eat** blue ghosts for bonus points; when eaten, their eyes make their way back to the center box in the maze, where the ghosts are "**regenerated**" and resume their normal activity. After a certain amount of time, blue-colored ghosts will flash white before turning back into their normal, lethal form.

TYPES OF GHOSTS

Each of the four ghosts have their own unique, distinct artificial intelligence (A.I.), or "personalities".

- Blinky gives direct chase to Pac-Man.
- Pinky and Inky try to position themselves in front of Pac-Man, usually by cornering him.
- Clyde will switch between chasing Pac-Man and fleeing from him.



Based on these behaviors and basic gameplay, the behaviour of the ghosts can be modelled using a Deterministic Finite State Machine.

In this project, we will attempt to identify the various states and design a Finite State Machine for the Ghosts and Pacman.

FINITE STATE AUTOMATA

The term "Automata" is derived from the Greek word "αὐτόματα" which means "self-acting". An automaton (*Automata* in plural) is an abstract self-propelled computing device which follows a predetermined sequence of operations automatically.

An automaton with a finite number of states is called a **Finite Automaton** (FA) or **Finite State Machine** (FSM).

It can be implemented with hardware or software and can be used to simulate sequential logic and some computer programs. Finite state automata generate regular languages. Finite state machines can be used to model problems in many fields including mathematics, artificial intelligence, games, and linguistics.

FORMAL DEFINITION OF A FINITE AUTOMATON

An automaton can be represented by a 5-tuple (Q, ∑, δ, q_o, F), where –

- Q is a finite set of states.
- ∑ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function.
- q_o is the initial state from where any input is processed (q_o ∈ Q).
- F is a set of final state/states of Q (F ⊆ Q).

Finite Automaton can be classified into two types :

- Deterministic Finite Automaton (DFA)
- Non-deterministic Finite Automaton (NDFA / NFA)

DETERMINISTIC FINITE AUTOMATON (DFA)

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automaton. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automaton.

GRAPHICAL REPRESENTATION OF A DFA

A DFA is represented by digraphs called state diagram.

- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Let $w = a_1a_2...a_n$ be a string over the alphabet Σ. The automaton M accepts the string w if a sequence of states, $r_0, r_1, ..., r_n$, exists in Q with the following conditions:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, ..., n - 1$
3. $r_n \in F$

In words, the first condition says that the machine starts in the start state q_0. The second condition says that given each character of string w, the machine will transition from state to state according to the transition function δ. The last condition says that the machine accepts w if the last input of w causes the machine to halt in one of the accepting states. Otherwise, it is said that the automaton *rejects* the string.

The set of strings that M accepts is the language *recognized* by M and this language is denoted by $L(M)$.

EXAMPLE

Let a deterministic finite automaton be

$Q = \{a, b, c\}$,

$\Sigma = \{0, 1\}$,

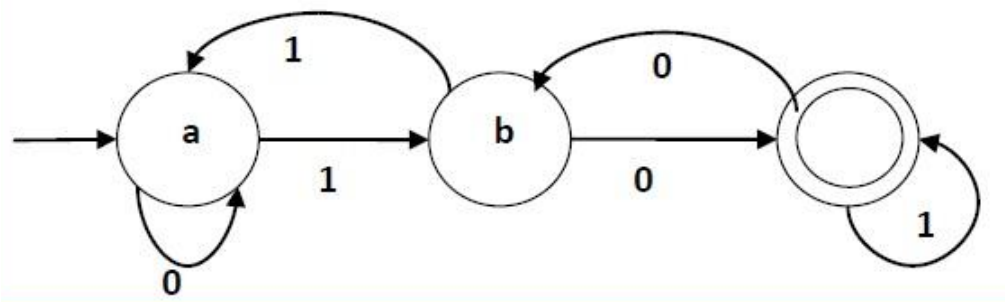
$q_0 = \{a\}$,

$F = \{c\}$, and

Transition function δ as shown by the following table −

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows:



APPLICATIONS OF DETERMINISTIC FINITE MACHINES

Deterministic Finite Automata, or DFAs, have a rich background in terms of the mathematical theory underlying their development and use.

DFA uses include protocol analysis, text parsing, video game character behavior, security analysis, CPU control units, natural language processing, and speech recognition. Additionally, many simple (and complex) mechanical devices are frequently designed and implemented using DFAs, such as elevators, vending machines, and traffic-sensitive traffic lights.

DFAs naturally lend themselves to concisely representing any system which must maintain an internal definition of state.

Pacman is one such application of finite state machines and that is what we have explored in this project.

ELEMENTS OF GHOST BEHAVIOUR IN PACMAN

THE GHOST HOUSE

When a player begins a game of Pac-Man, they are not immediately attacked by all the ghosts. One ghost begins in the actual maze, while the others are inside a small area in the middle of the maze, called the "ghost house". Other than at the beginning of a level, the ghosts will only return to this area if they are eaten by an energized Pac-Man, or as a result of their positions being reset when Pac-Man dies. The ghost house is otherwise inaccessible, and is not a valid area for Pac-Man or the ghosts to move into

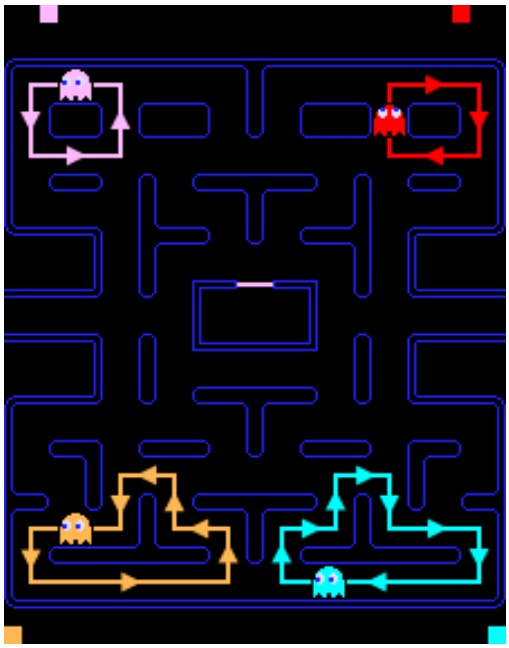
GHOST MOVEMENT MODES

Ghosts have three mutually-exclusive modes of behavior they can be in during play: chase, scatter, and frightened.

1. **CHASE**:
   A ghost's objective in *chase* mode is to find and capture Pac-Man by hunting him down through the maze.
2. **SCATTER:**
   In *scatter* mode, the ghosts give up the chase for a few seconds and head for their respective home corners. It is a welcome but brief rest-soon enough, they will revert to chase mode and be after Pac-Man again.
3. **FRIGHTENED**:
   Ghosts enter *frightened* mode whenever Pac-Man eats one of the four energizers located in the far corners of the maze. During the early levels, the ghosts will all turn dark blue (meaning they are vulnerable) and aimlessly wander the maze for a few seconds. They will flash moments before returning to their previous mode of behavior.

TARGET TILE

The key to understanding ghost behavior is the concept of a target tile. The large majority of the time, each ghost has a specific tile that it is trying to reach, and its behavior revolves around trying to get to that tile from its current one. All of the ghosts use identical methods to travel towards their targets, but the different ghost personalities come about due to the individual way each ghost has of selecting its target tile.



CHASE MODE: The target tile is decided on the basis of the position of Pac-Man. It is different for each ghost.

SCATTER MODE: Each ghost has a fixed target tile, each of which is located just outside a different corner of the maze. This causes the four ghosts to disperse to the corners whenever they are in this mode.

FRIGHTENED MODE: The ghosts do not have a specific target tile while in this mode. Instead, they pseudorandomly decide which turns to make at every intersection.

INDIVIDUAL GHOST BEHAVIOUR

"This is the heart of the game. I wanted each ghostly enemy to have a specific character and its own particular movements, so they weren't all just chasing after Pac Man in single file, which would have been tiresome and flat."

- Toru Iwatani, Pac-Man creator

BLINKY, THE RED GHOST

The red ghost starts outside of the ghost house, and is usually the first one to be seen as a threat, since he makes a beeline for Pac-Man almost immediately. He is referred to as "Blinky", and the game describes his personality as *shadow*.

Of all the ghosts' targeting schemes for chase mode, Blinky's is the most simple and direct, using Pac-Man's current tile as his target.

All ghosts move at the same rate of speed when a level begins, but Blinky will increase his rate of speed twice each round based on the number of dots remaining in the maze. While in this accelerated state, Blinky is commonly called "Cruise Elroy".

PINKY, THE PINK GHOST

The pink ghost starts inside the ghost house and exits immediately. His nickname is "Pinky", and his personality is described as *speedy*. Pinky and Blinky often seem to be working in concert to box Pac-Man in, leaving him with nowhere to run.

In chase mode, Pinky behaves as he does because he does not target Pac-Man's tile directly. Instead, he selects an offset four tiles away from Pac-Man in the direction Pac-Man is currently moving.

INKY, THE BLUE GHOST

The blue ghost is nicknamed Inky, and remains inside the ghost house for a short time on the first level, not joining the chase until Pac-Man has managed to consume at least 30 of the dots. His personality description is *bashful*.

Sometimes he chases Pac-Man aggressively like Blinky; other times he jumps ahead of Pac-Man as Pinky would. He might even wander off like Clyde on occasion!

Inky uses the most complex targeting scheme of the four ghosts in chase mode. He needs Pac-Man's current tile/orientation and Blinky's current tile to calculate his final target.

CLYDE, THE ORANGE GHOST

The orange ghost, "Clyde", is the last to leave the ghost house, and does not exit at all in the first level until over a third of the dots have been eaten. Clyde's personality description is *pokey*.

Clyde's target differs based on his proximity to Pac-Man. When more than eight tiles away, he uses Pac-Man's tile as his target (shown as the yellow target above). If Clyde is closer than eight tiles away, he switches to his scatter mode target instead, and starts heading for his corner until he is far enough away to target Pac-Man again.

FINITE STATE MACHINES

From all the game design discussed above, we can identify 2 separate Finite State Machines that we can design. One for the mental state for PACMAN, and the other for the behaviour of the ghosts.

GAME PLAY

The states include:

- **Main Menu :** Game is on Main Menu screen
- **Ready :** Game is in Ready Screen
- **Play :** Gameplay is active
- **Reset :** Game is Reset after Pacman is killed
- **Changing Levels :** Game Board changes when Pacman wins a Level
- **Game Complete :** All Levels completed

Transitions include:

- **Play :** Gameplay has begun
- **Timeout :** Game Timer to begin ends
- **Lives Left :** Player has more that 0 lives left
- **Level up :** Player completes the level
- **Lives Lost :** Player has no more lives left
- **Killed :** Pacman gets captured by the ghosts
- **Game Won :** Pacman wins the level

Then we can define a Finite State Machine for Pacman as (Q, ∑, δ, q_o, F) as:

Let Q = { Main Menu, Ready, Play, Reset, Changing Levels, Game Complete }

∑ = { Play, Timeout, Lives Left, Level Up, Lives Lost, Killed, Game Won }

q_o = { Main Menu }

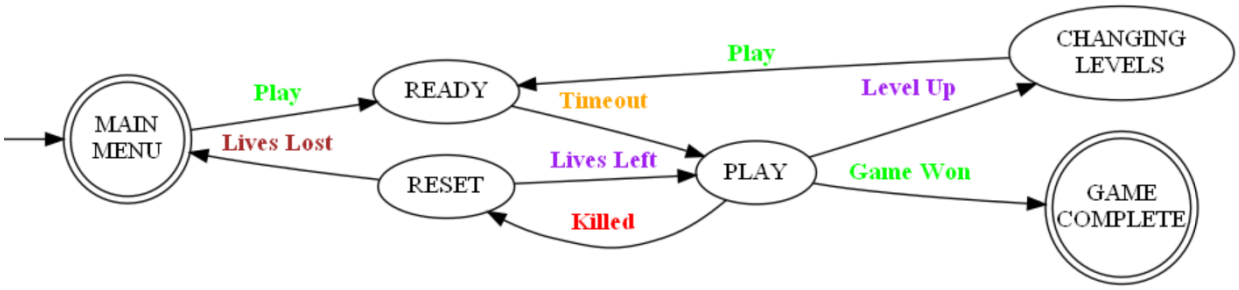
F = { Main Menu, Game Complete }

and δ =

States/ Transition	Play	Timeout	Level Up	Lives Left	Lives Lost	Killed	Game Won
MAIN MENU	Ready						
READY		Play					
PLAY			Changing Level			Reset	Game Complete
CHANGING LEVELS	Ready						
RESET				Play	Main Menu		
GAME COMPLETE							

Transition Table

Transition Diagram



PACMAN

Pacman's movements are controlled by the player, however, the pacman has different states that can be modelled by a Deterministic Finite State Automaton.

The states include:

- **Asleep** : Pacman is "Asleep" when the game is not started
- **Home** : Pacman is in Starting position
- **Active** : During gameplay i.e player is controlling Pacman
- **Powerful** : Pacman has eaten a power pellet and can eat ghosts

Transitions include:

- **Game Start** : Player starts a game
- **Game Restart** : Game is reset (after a loss of life)
- **Energized** : When Pacman becomes powerful after eating a power pellet
- **De-energized** : Pacman loses power status
- **Killed** : When a ghost captures Pacman
- **Game Won** : Level Completed
- **End Game** : All 3 lives lost

Then we can define a Finite State Machine for Pacman as (Q, ∑, δ, q_o, F) as:

Let Q = { Asleep, Home, Active, Powerful }

∑ = { Game Start, Game Restart, Energized, De-energized, Killed, Game Won, End Game }

q_o = { Asleep }

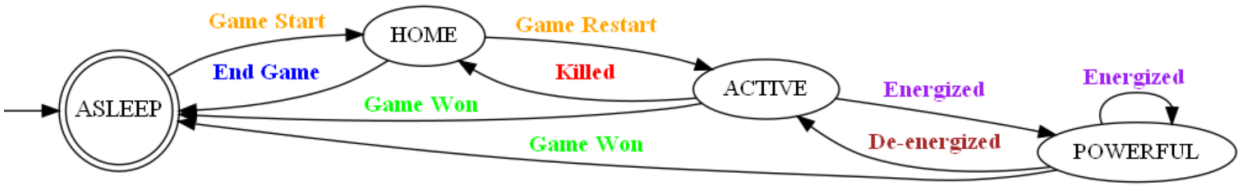
F = { Asleep }

and δ =

States/ Transition	Game Started	Game End	Game Restart	Killed	Energized	De-energized	Game Won
ASLEEP	Home						
HOME		Asleep	Active				
ACTIVE				Home	Powerful		Asleep
POWERFUL					Powerful	Active	Asleep

Transition Table

Transition Diagram



GHOSTS

Unlike Pacman, whose movements and states are controlled by the player, the Ghosts are controlled by the Game AI. The Ghosts' behaviour is modelled by a deterministic finite state automaton.

The states included in the model are:

- **Asleep** : Ghosts are not active when game has not been started
- **Home** : When Ghosts are in Ghost Home.
- **Scatter** : Scatter Mode
- **Chase** : Chase Mode
- **Frightened** : Frightened Mode

The transitions for ghosts are:

- **Game Start** : Player starts the game
- **Game End** : All lives of player lost
- **Leaving home** : Ghosts receive command to leave home and start moving through the maze
- **Enter Scatter** : Game sends a command to Ghosts to enter Scatter Mode
- **Enter Chase** : Game sends a command to Ghosts to enter Scatter Mode
- **Pacman Energized** : Pacman eats a power pellet
- **Pacman De-energized** : Pacman's timer for power runs out
- **Pacman Killed** : Pacman is captured by a ghost

Hence, we can define a deterministic finite state machine (Q, ∑, δ, q_o, F) as:

Let Q = { Asleep, Home, Scatter, Frightened, Chase }

∑ = { Game Start, Game End, Leaving home, Enter Scatter, Enter Chase, Pacman Energized, Pacman De-energized, Pacman Killed }

q_o = { Asleep }

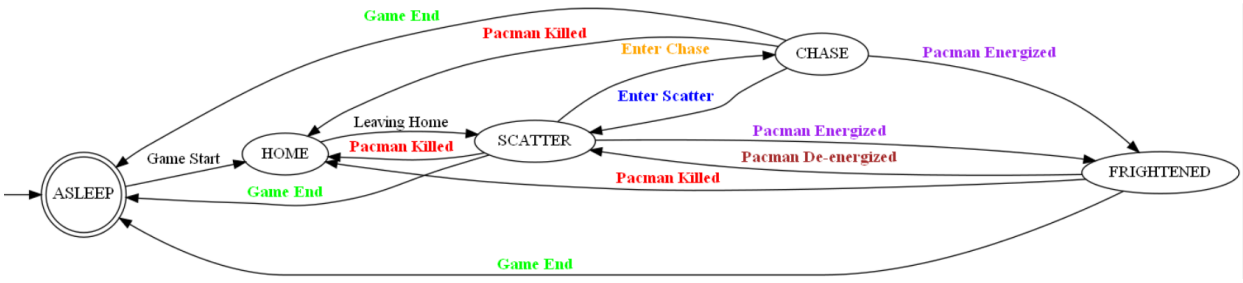
F = { Asleep }

and δ =

States/ Transition	Game Start	Game End	Leaving Home	Enter Scatter	Enter Chase	Pacman Energized	Pacman De-energized	Pacman Killed
ASLEEP	Home							
HOME			Scatter					
SCATTER		Asleep			Chase	Frightened		Home
FRIGHTENED		Asleep					Scatter	Home
CHASE		Asleep		Scatter		Frightened		Home

Transition Table

Transition Diagram



CLYDE

Clyde, the orange ghost, changes target based on his proximity to Pac-Man. When more than eight tiles away, he targets Pacman. If Clyde is closer than eight tiles away, he switches to his scatter mode target instead. Hence his transition from Chase to Scatter mode can be modelled by a deterministic finite state automaton.

The states include:

- **Ghost House :** Clyde is in the ghost house until he gets the command to leave by the game.
- **Attack Pacman :** Chase Mode (Target Pacman)
- **Scatter Mode :** Move away from Pacman towards his corner

Transitions include:

- **One Third Dots Eaten :** Clyde leaves the ghost house only when one third of the dots have been eaten by Pacman.
- **Pacman Close :** Pacman is closer than eight tiles.
- **Pacman Far :** Pacam is more than eight tiles away.
- **Game End :** Pacman is captured by ghosts.

Hence, we can define a deterministic finite state machine (Q, ∑, δ, q_o, F) as:

Let Q = { Ghost House, Attack Pacman, Scatter Mode }

∑ = { One Third Dots Eaten, Pacman Close, Pacman Far, Game End }

q_o = { Ghost House }

F = { Ghost House }

and δ =

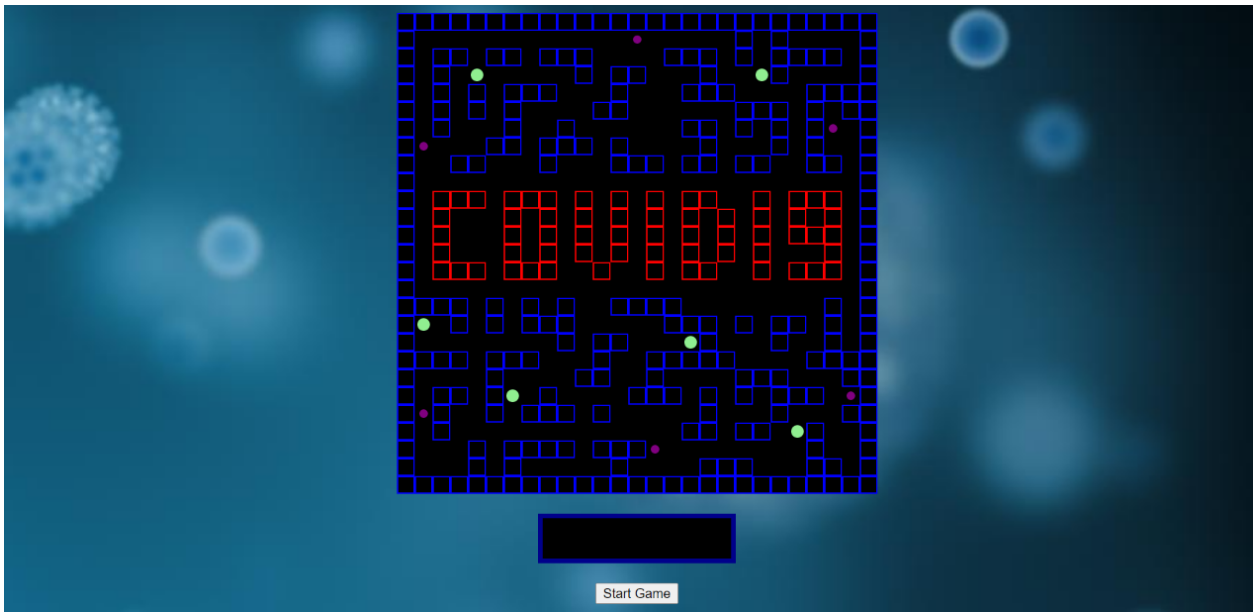
States/ Transition	One Third Dots Eaten	Pacman Close	Pacman Far	Game End
GHOST HOUSE	Attack Pacman			
ATTACK PACMAN		Scatter Mode		Ghost House
SCATTER MODE			Attack Pacman	Ghost House

Transition Table

Transition Diagram

IMPLEMENTATION

We implemented the concepts learnt and designed a prototype for the ‘Pacman’ game. We used **HTML**, **CSS** and **Javascript** for the same. We redesigned the maze completely to give the game the theme of ‘COVID-19’. We named it CoroNO-MAN.



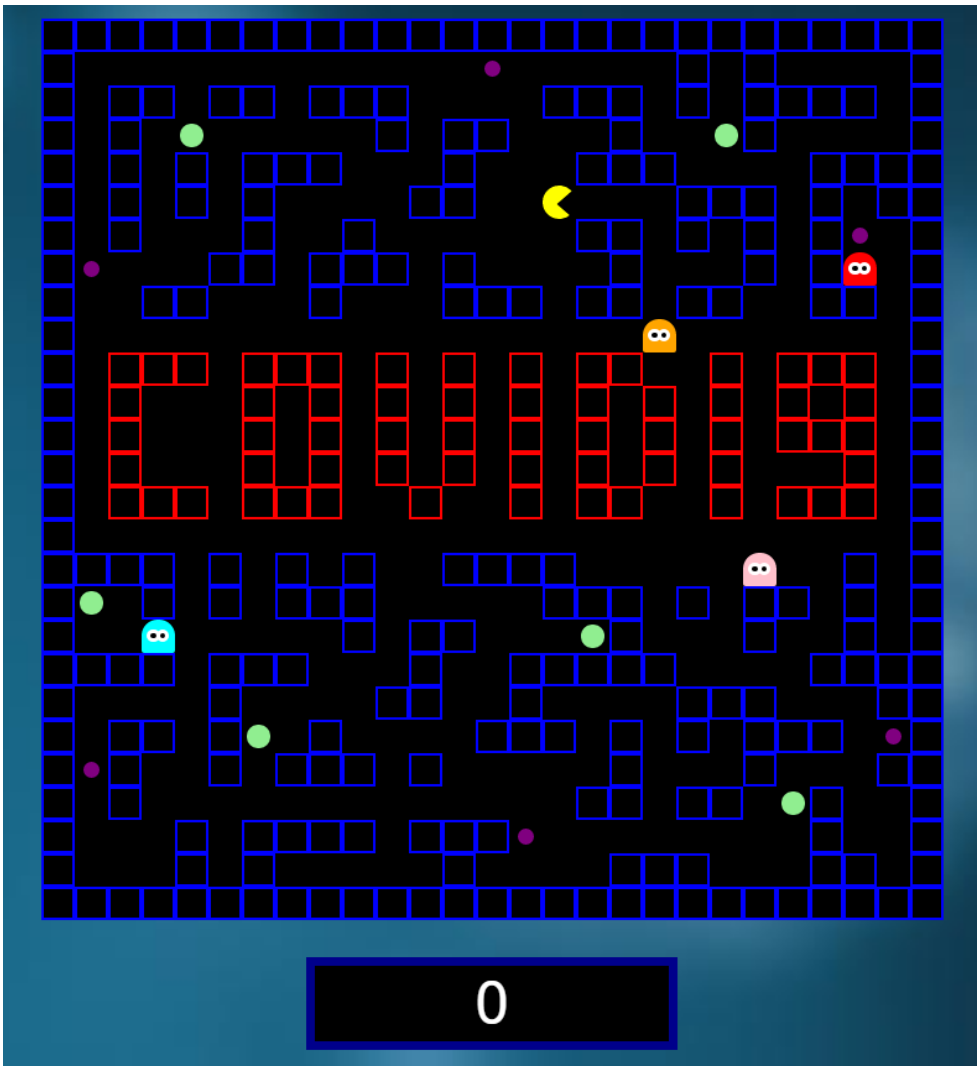
We made the following changes to the original Game:

CoroNO-Man : Our player is a hypochondriac who has to do his daily chores outside his home such as grocery shopping, post office, medicine shopping etc. which are represented by ‘green dots’ in the maze.

INVALIDS : They are people infected with Covid (without masks) that need to be avoided while doing the chores. Coming in contact with the invalids gives you Corona and the game ends.

VACCINES : The vaccines make the CoroNO-Man immune to the Invalids. It also gives CoroNO-Man the ability to send the Invalids back to quarantine i.e their starting positions.

The goal of the game is to complete the tasks without getting infected by the Invalids.



The source code for this can be found on our Github link here:
https://github.com/AIMAN-SIDDIQUA/CoroNO-Man

SIGNIFICANCE

Pac-Man was one of the earliest instances of computer game programming. Finite State Machines were used in its development. Since then, they have been employed to build and design various video games.The fact that they are relatively easy to understand, implement, and debug contributes to their frequent use in game development.

Nearly all of the games we used to play, and a lot of the modern games we're still playing use FSM-based AI. It is used not just to generate somewhat credible interactions between the player and the other characters, but to implement other critical elements of game development such as the graphical user interface (GUI), input handling, player controls and progression through the history.

State machines are certainly not the most sophisticated means of implementing artificially intelligent agents in games, but many games include characters with simple, state-based behaviors that are easily and effectively modeled using state machines.

REFERENCES

1. “Understanding Pac-Man Ghost Behavior”, Chad Birch, December 2, 2010. http://gameinternals.com/understanding-pac-man-ghost-behavior
2. “The Pac-Man Dossier”, Jamey Pittman, February 23,2009. https://www.gamasutra.com/view/feature/132330/the_pacman_dossier.php
3. Tory Iwatani, Creator of Pac-Man, Interview by Susan M. Lammers, Programmers at Work. https://programmersatwork.wordpress.com/toru-iwatani-1986-pacman-designer/
4. Finite State Machines. Brilliant.org. Retrieved 07:28, March 28, 2021, from https://brilliant.org/wiki/finite-state-machines/
5. “AI for Game Developers”,Glenn Seemann, David M Bourg. 2021, O’Reilly Media, Inc. https://www.oreilly.com/library/view/ai-for-game/0596005555/ch09.html
6. “Theory Of Computer Science: Theory, Automata, And Computation”, K L P Mishra and N Chandrasekaran, Prentice Hall India Learning Private Limited; 3rd edition (1 January 2006)

APPENDIX

The Transition Diagrams were created using a data visualization tool, Graphviz. Graphviz is an open source graph visualization software used for representing various structural information as diagrams of abstract graphs and networks.

The Graphviz layout programs take descriptions of graphs in a simple text language, known as DOT language, and make diagrams in useful formats, such as images and SVG for web pages.

The code in DOT language used to create both the diagrams is given below:

GAME PLAY

```
digraph{
    rankdir = "LR";

    //STATES
    {   node [margin = 0 fontsize = 14 width = 0.9 shape = doublecircle]
        A [fixedsize = true label = "MAIN\nMENU"]; }

    {   node [margin = 0 fontsize = 14 width = 1.1 shape = doublecircle]
        F [fixedsize = true label = "GAME\nCOMPLETE"]; }

    S;
    B [label = "READY"];
    C [label = "PLAY"];
    D [label = "RESET"];
    E [label = "CHANGING\nLEVELS"];

    {   rank = same;
        B, D; }

    //TRANSITIONS
    S -> A;
    A -> B [label = <<font color = 'green'><b>Play</b> </font>>];
    B -> C [label = <<font color = 'orange'><b>Timeout</b> </font>>];
    C -> D [label = <<font color = 'red'><b>Killed</b> </font>>];
    D -> C [label = <<font color = 'purple'><b>Lives Left</b> </font>>];
    C -> E [label = <<font color = 'purple'><b>Level Up</b> </font>>];
    E -> B [label = <<font color = 'green'><b>Play</b> </font>>];
    D -> A [label = <<font color = 'brown'><b>Lives Lost</b> </font>>];
    C -> F [label = <<font color = 'green'><b>Game Won</b> </font>>];

}
```

PACMAN MOVEMENTS

```
digraph {
    rankdir = "LR";

    //STATES
    {
        node [margin = 0 fontsize = 15 width = 0.9 shape = doublecircle]
        A [fixedsize = true label = "ASLEEP"];
    }

    S;
    B [label = "HOME"];
    C [label = "ACTIVE"];
    D [label = "POWERFUL"];

    //TRANSITIONS
    S -> A;
    A -> B [label = <<font color = 'orange'><b>Game Start</b> </font>>];
    B -> A [label = <<font color = 'blue'><b>End Game</b> </font>>];
    B -> C [label = <<font color = 'orange'><b>Game Restart</b> </font>>];

    C -> B [label = <<font color = 'red'><b>Killed</b> </font>>];
    C -> D [label = <<font color = 'purple'><b>Energized</b> </font>>];
    D -> C [label = <<font color = 'brown'><b>De-energized</b> </font>>];
    D -> D [label = <<font color = 'purple'><b>Energized</b> </font>>];

    D -> A [label = <<font color = 'green'><b>Game Won</b> </font>>];
    C -> A [label = <<font color = 'green'><b>Game Won</b> </font>>];
}
```

CLYDE MOVEMENTS

```
digraph{
    rankdir = "LR";

    //STATES
    {   node [margin = 0 fontsize = 14 width = 0.9 shape = doublecircle]
        A [fixedsize = true label = "GHOST\nHOUSE"]; }

    S;
    B [label = <<b>ATTACK PACMAN</b>>];
    C [label = <<b>SCATTER MODE</b>>];

    //TRANSITION
    S -> A;
    A -> B [label = <<font color = 'green'><b>One Third Dot Eaten</b> </font>>];
    B -> C [label = <<font color = 'purple'><b>Pacman Close</b> </font>>];
    C -> B [label = <<font color = 'brown'><b>Pacman Far</b> </font>>];

    B -> A [label = <<font color = 'red'><b>Game End</b> </font>>];
    C -> A [label = <<font color = 'red'><b>Game End</b> </font>>];
}
```

GHOST MOVEMENTS

```
digraph{
    rankdir = "LR";

    //STATES
    {   node [margin = 0 fontsize = 15 width = 0.9 shape = doublecircle]
        A [fixedsize = true label = "ASLEEP"]; }

    S;
    B [label = "HOME"];
    C [label = "CHASE"];
    D [label = "FRIGHTENED"];
    E [label = "SCATTER"];

    //TRANSITIONS
    S -> A;
    A -> B [label = "Game Start"];
    B -> E [label = "Leaving Home"];
    E -> C [label = <<font color = 'orange'><b>Enter Chase</b> </font>>];
    C -> E [label = <<font color = 'blue'><b>Enter Scatter</b> </font>>];

    C -> D [label = <<font color = 'purple'><b>Pacman Energized</b> </font>>];
    E -> D [label = <<font color = 'purple'><b>Pacman Energized</b> </font>>];
    D -> E [label = <<font color = 'brown'><b>Pacman De-energized</b> </font>>];

    D -> B [label = <<font color='red'><b>Pacman Killed</b></font>>];
    C -> B [label = <<font color='red'><b>Pacman Killed</b></font>>];
    E -> B [label = <<font color='red'><b>Pacman Killed</b></font>>];
    E -> A [label = <<font color='green'><b>Game End</b></font>>];
    C -> A [label = <<font color='green'><b>Game End</b></font>>];
    D -> A [label = <<font color='green'><b>Game End</b></font>>];

}
```