Group: Chathuni and Urvashi

Pacman game design documentation.

**Step 1: Decomposition**

Purpose:

* To create a Pacman game which is smooth, fast and responsive where players can control Pacman to eat kibbles in a maze without running into ghouls.

Description:

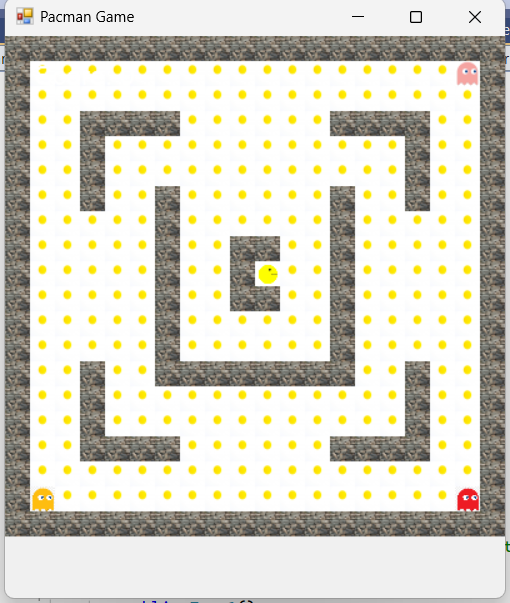
* We will start by making walls, Pacman, a maze, and ghouls to set up the game. Set Pacman movement, ghoul movement, and collision detection. Tracking the score and displaying it on the screen.

Extra Functionality:

* We added a method to make the ghouls chance of catching Pacman high.

**Step 2: Form design**

* At the beginning of the game, the design layout will display walls, Pacman, maze, and ghouls.
* It will display current score.



* At the bottom of the game we will add a score option when coding.

**Controls and components:**

* Pacman Movement (Player Character): Use arrow keys to move Pacman up, down, left, and right.
* Kibble: It is placed throughout the maze. When Pacman eats a kibble, it disappears, and the score increases.
* Ghouls: Ghouls move through the maze trying to catch Pacman. Each ghoul has a different movement pattern.
* Score display: Displays the current score, which increases when Pacman eats kibbles and ghouls.
* Timer: Manages game loops and timing for power-up durations and ghoul behaviour patterns.
* Game over message: Displays when Pacman loses all lives.
* Game won message: Displays when Pacman eats all the kibbles.

**Step 3: Abstraction**

Identify the classes, I am going to have five classes there are:

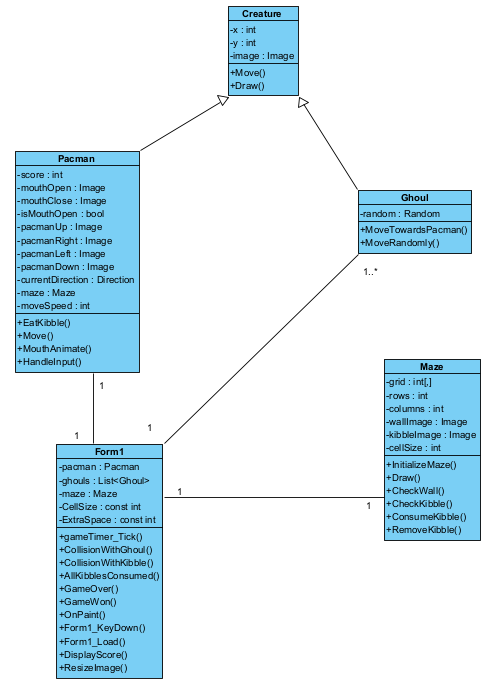
* Pacman (pacman.cs)
* Ghoul (ghoul.cs)
* Maze (maze.cs)
* Creature (creature.cs)
* Pacman game(form1.cs)

The purpose of each class:

* Pacman: To manage the Pacman movement, animation and scoring
* Ghoul: To manage the movement of the ghoul which represents an enemy character that is moving around the maze.
* Maze: To manage the layout and drawing of the maze.
* Creature: This is a base class for ghouls and Pacman.
* Pacman game: To manage overall game logic.

**Step 4: Encapsulation**

This is our UML class diagram



Methods:

Creature

* Move()
* Draw()

Pacman

* EatKibble()
* Move()
* HandleInput()
* MouthAnimate()

Ghoul

* MoveTowardsPacman()
* MoveRandomly()

Maze

* InitializeMaze()
* Draw()
* CheckWall()
* CheckKibble()
* ConsumeKibble()
* RemoveKibble()

Pacmangame(form1)

* gameTimer\_Tick()
* CollisionWithGhoul()
* CollisionWithKibble()
* AllKibblesConsumed()
* GameOver()
* GameWon()
* Form1\_KeyDown()
* OnPaint()
* Form1\_Load()
* DisplayScore()
* ResizeImage()

 Fields:

Creature

* x: int
* y: int
* image: Image

Pacman

* score: int
* mouthOpen: Image
* mouthClose: Image
* isMouthOpen: bool
* pacmanUp: Image
* pacmanRight: Image
* pacmanLeft: Image
* pacmanDown: Image
* currentDirection: Direction
* maze: Maze
* moveSpeed: int

Ghoul

* random: Random

Maze

* grid: int[,]
* rows: int
* columns: int
* wallImage: Image
* kibbleImage: Image
* cellSize: int

Form 1

* pacman: Pacman
* ghouls: List<Ghoul>
* maze: Maze
* CellSize: const int
* ExtraSpace: const int

**Step 5: Iterative Refinement**

Purpose of each method:

Move() – It will be used to move both Pacman and Ghoul with it’s specific movement for each creature.

Draw() – It will draw the creature at it’s relevant position.

EatKibble() – It will increase the Pacman’s score by 1 when it eats each kibble.

Move() – It will override the move method for the Pacman.

HandleInput() - It will update the movement based on the user input.

MouthAnimate() – It will set the Pacman’s mouth for animation.

MoveTowardsPacman() – It will adjust the direction of ghoul to move towards pacman.

MoveRandomly() - It will move the ghouls randomly in the maze.

InitializeMaze() – It will set up the maze with walls and kibbles.

Draw() – It will draw the maze on the form.

CheckWall() – It will check if the cell is a wall.

CheckKibble() – It will check if the cell is a kibble.

ConsumeKibble() – It will mark a kibble when it’s consumed by Pacman.

RemoveKibble() - It will remove a kibble from a relative position.

gameTimer\_Tick() – It will handle the timer tick event.

CollisionWithGhoul() – It will check the collision between Pacman and Ghoul.

CollisionWithKibble() – It will check the collision between Pacman and Kibble.

AllKibblesConsumed() – It will check if all the kibbles are consumed in the maze.

GameOver() – It will handle the game over scenario.

GameWon() – It will handle the game won scenario.

OnPaint() – It will draw the game components on the form.

Form1\_KeyDown() – It will move the Pacman using the arrow keys

Form1\_Load() - It will handle the form load event.

DisplayScore() – It will draw the label for score using graphics.

ResizeImage() - It will resize the images using graphics.