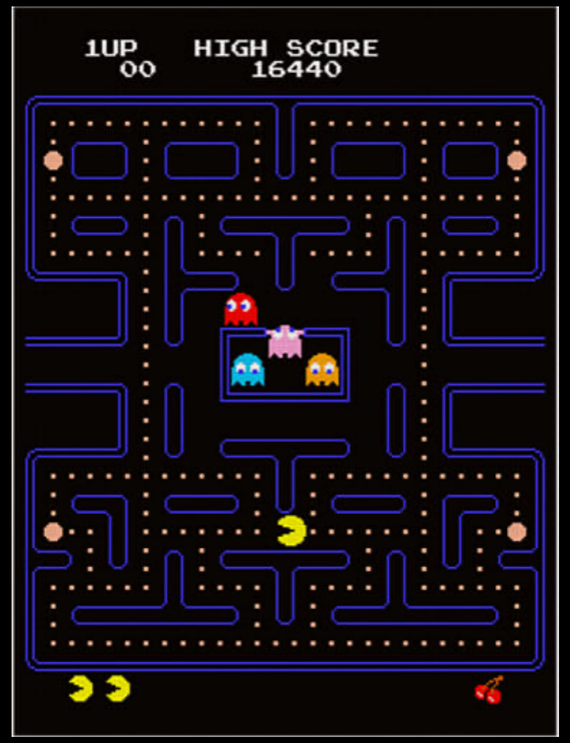
PAC-MAN™

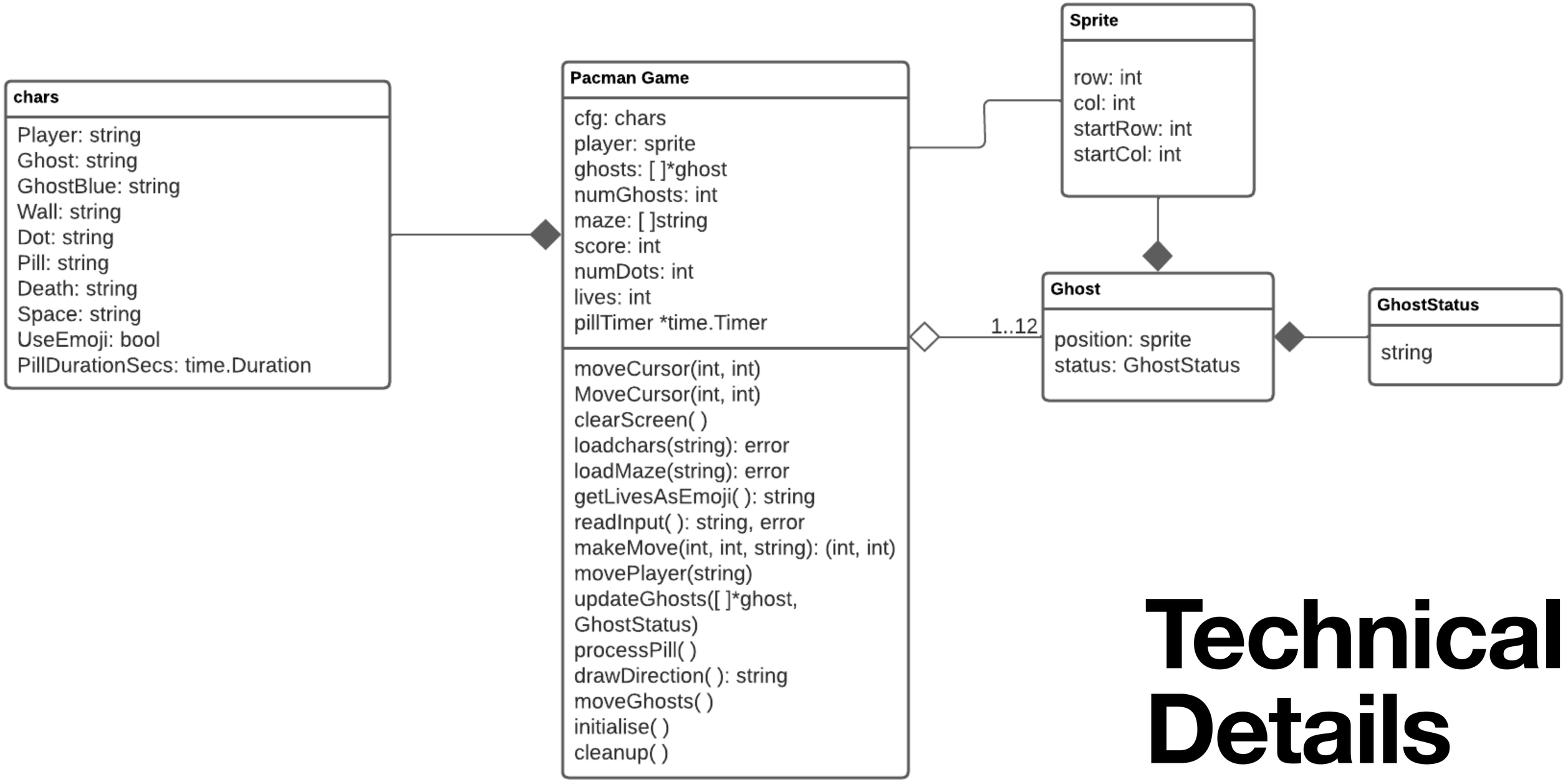
ONEMANTEAM's PACMAN IN GO

Jesús Riquelmer Gaxiola Higuera - A01740223

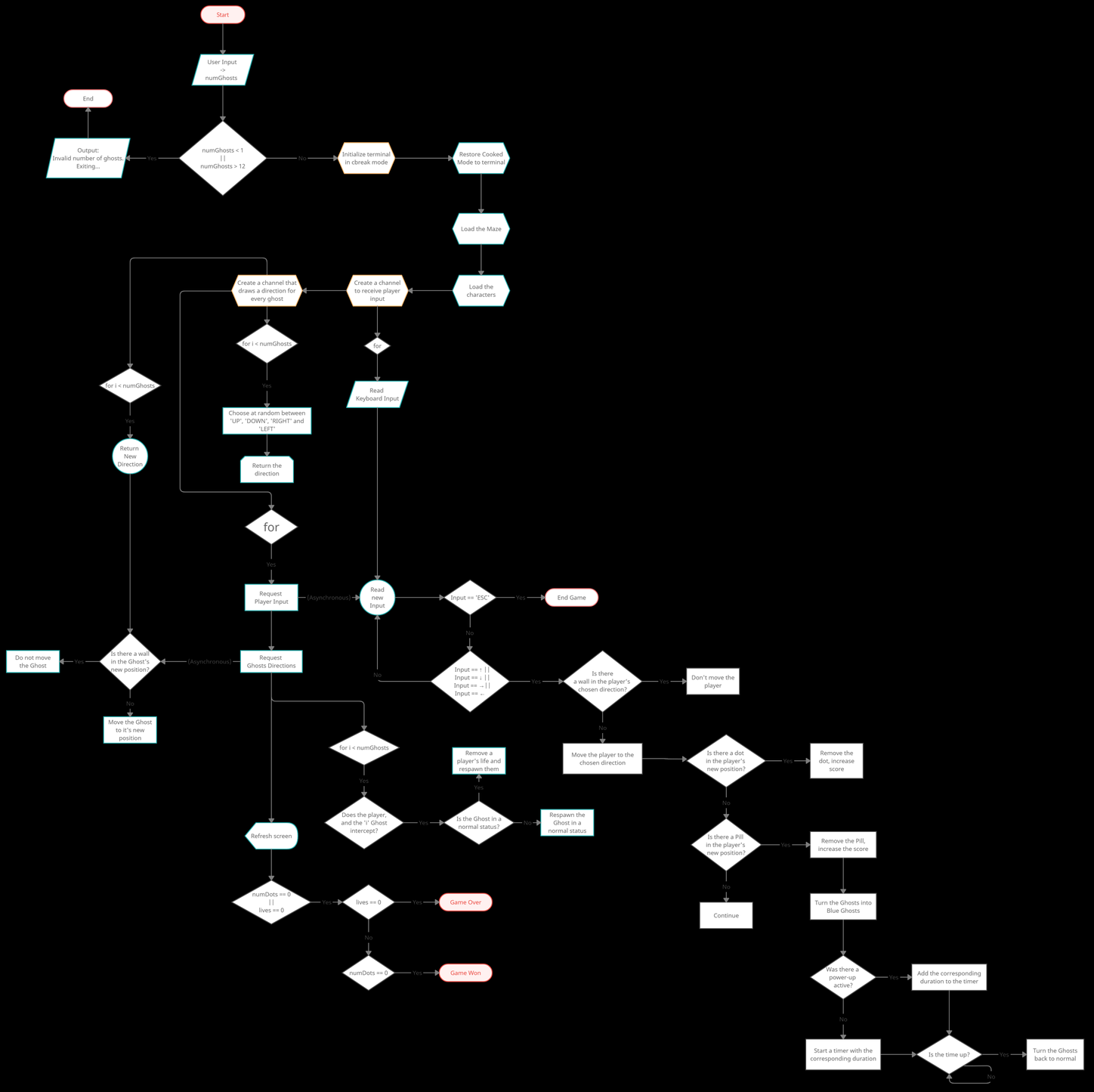
Why Pacman?

- As one of the best-selling games in history, loved by both the hardcore and the more casual audiences, Pacman is always a safe option to try new ideas!





Technical Details


Start
User Input -> numGhosts
End
Output: Invalid number of ghosts. Exiting...
numGhosts < 1 || numGhosts > 12
Yes
No
Initialize terminal in cbreak mode
Restore Cooked Mode to terminal
Load the Maze
Load the characters
Create a channel to receive player input
Create a channel that draws a direction for every ghost
for
Read Keyboard Input
for i < numGhosts
Yes
Choose at random between 'UP', 'DOWN', 'RIGHT' and 'LEFT'
Return the direction
for i < numGhosts
Yes
Return New Direction
for
Yes
Request Player Input
[Asynchronous]
Read new Input
Input == 'ESC'
Yes
End Game
No
Input == ↑ || Input == ↓ || Input == → || Input == ←
Yes
Is there a wall in the player's chosen direction?
Yes
Don't move the player
No
Move the player to the chosen direction
Is there a dot in the player's new position?
Yes
Remove the dot, increase score
No
Is there a Pill in the player's new position?
Yes
Remove the Pill, increase the score
No
Continue
Turn the Ghosts into Blue Ghosts
Was there a power-up active?
Yes
Add the corresponding duration to the timer
No
Start a timer with the corresponding duration
Is the time up?
Yes
Turn the Ghosts back to normal
No
Request Ghosts Directions
Is there a wall in the Ghost's new position?
Yes
Do not move the Ghost
No
Move the Ghost to it's new position
for i < numGhosts
Yes
Does the player, and the 'i' Ghost intercept?
Yes
Is the Ghost in a normal status?
No
Respawn the Ghost in a normal status
Yes
Remove a player's life and respawn them
Refresh screen
numDots == 0 || lives == 0
Yes
lives == 0
Yes
Game Over
No
numDots == 0
Yes
Game Won

Key Code Snippets

```go
func clearScreen() {
    fmt.Print("\x1b[2J")
    MoveCursor(0, 0)
}

func moveCursor(row, col int) {
    fmt.Printf("\x1b[%d;%df", row+1, col+1)
}
```

Borrowed straight from:
https://github.com/danicat/simpleansi

```go
func printScreen() {
    clearScreen()
    for _, line := range maze {
        for _, chr := range line {
            switch chr {
            case '#':
                fmt.Print(cfg.Wall)
            case '.':
                fmt.Print(cfg.Dot)
            case 'X':
                fmt.Print(cfg.Pill)
            default:
                fmt.Print(cfg.Space)
            }
        }
        fmt.Println()
    }

    MoveCursor(player.row, player.col)
    fmt.Print(cfg.Player)

    ghostsStatusMx.RLock()
    for _, g := range ghosts {
        MoveCursor(g.position.row, g.position.col)
        if g.status == GhostStatusNormal {
            fmt.Printf(cfg.Ghost)
        } else if g.status == GhostStatusBlue {
            fmt.Printf(cfg.GhostBlue)
        }
    }
    ghostsStatusMx.RUnlock()

    MoveCursor(len(maze)+1, 0)

    livesRemaining := strconv.Itoa(lives)
    if cfg.UseEmoji {
        livesRemaining = getLivesAsEmoji()
    }

    fmt.Println("Score:", score, "\tLives:", livesRemaining)
}
```

One of the most crucial functions to get right!

```go
func readInput() (string, error) {
    buffer := make([]byte, 100)

    cnt, err := os.Stdin.Read(buffer)
    if err != nil {
        return "", err
    }

    if cnt == 1 && buffer[0] == 0x1b {
        return "ESC", nil
    } else if cnt >= 3 {
        if buffer[0] == 0x1b && buffer[1] == '[' {
            switch buffer[2] {
            case 'A':
                return "UP", nil
            case 'B':
                return "DOWN", nil
            case 'C':
                return "RIGHT", nil
            case 'D':
                return "LEFT", nil
            }
        }
    }

    return "", nil
}
```

What’s a game without a player?

```go
func makeMove(oldRow, oldCol int, dir string) (newRow, newCol int) {
    newRow, newCol = oldRow, oldCol

    switch dir {
    case "UP":
        newRow = newRow - 1
        if newRow < 0 {
            newRow = len(maze) - 1
        }
    case "DOWN":
        newRow = newRow + 1
        if newRow == len(maze)-1 {
            newRow = 0
        }
    case "RIGHT":
        newCol = newCol + 1
        if newCol == len(maze[0]) {
            newCol = 0
        }
    case "LEFT":
        newCol = newCol - 1
        if newCol < 0 {
            newCol = len(maze[0]) - 1
        }
    }

    if maze[newRow][newCol] == '#' {
        newRow = oldRow
        newCol = oldCol
    }

    return
}
```

An easy function to mess up!

```go
func drawDirection() string {
    dir := rand.Intn(4)
    move := map[int]string{
        0: "UP",
        1: "DOWN",
        2: "RIGHT",
        3: "LEFT",
    }
    return move[dir]
}
```

The Latest in AI!

```
//Process input (async)
    input := make(chan string)
    go func(ch chan<- string) {
        for {
            input, err := readInput()
            if err != nil {
                log.Print("error reading input:", err)
                ch <- "ESC"
            }
            ch <- input
        }
    }(input)

    //make a channel for each ghost
    ghostChannels := make([]chan string, numGhosts)
    for i := 0; i < numGhosts; i++ {
        ghostChannels[i] = make(chan string)
        go func(ch chan<- string) {
            for {
                ch <- drawDirection()
            }
        }(ghostChannels[i])
    }
```

Fancy Words!

Demonstration Time!