Waka Game Design Report

`INFO1113`

My design consisted of 3 components:

- The game (configuration, mainline, app class)
- The map (and included cells),
- The entities (ghosts and waka)

The Game

> **Relevant Classes:**
>
> - App
> - Config

This is the outermost layer of the game's mechanics.

The configuration file is parsed using the Config class which stores the information (map file path, lives, speed, mode lengths, frightened length) as instance attributes.

Using this configuration object, the App class is able to parse the map, loading all necessary images and instantiating the correct amount of cells, ghosts etc. and the waka according to the map file given.

The App class acts as an interface by triggering events from key presses. Finally, the app class calls the tick and draw methods for all objects, including entities and the cells of the map.

The Entities

> **Relevant Classes:**
>
> - Entity *(abstract)*
>   - Waka
>   - Ghost *(abstract)*
>   - Ambusher
>   - Chaser
>   - Ignorant
>   - Whim
> - Direction *(enum)*
> - ghostType *(enum)*

Entities are any objects on the screen that move. They have x and y coordinates (current and original), a direction, a speed, and a connected map . The Entity class provides methods to change direction (with checks of when to do so), move forward, and check for wall collisions.

The Entity parent class provides basic mechanics for the Waka and the Ghosts. The Waka adds functionality for collecting fruit, checking and dealing ghost collisions, and the tick method, combines these methods so that the Waka is able to move appropriately based on user input each frame.

The Ghost class provides a blueprint for the 4 different types of ghosts. It provides a functionality for the different ghosts to make a choice of movement based on a target coordinate. The ghost class also manages whether the ghosts are in frightened, chase or scatter mode.

The four ghost types all inherit from Ghost, and include their own functionality for calculating their target. Notably, the Whim class also stores an instance of Chaser it is linked to so it can calculate its vector and in turn target. If the Chaser dies, Whim simply stays in scatter mode until the chaser is resurrected or the game ends.

The Map

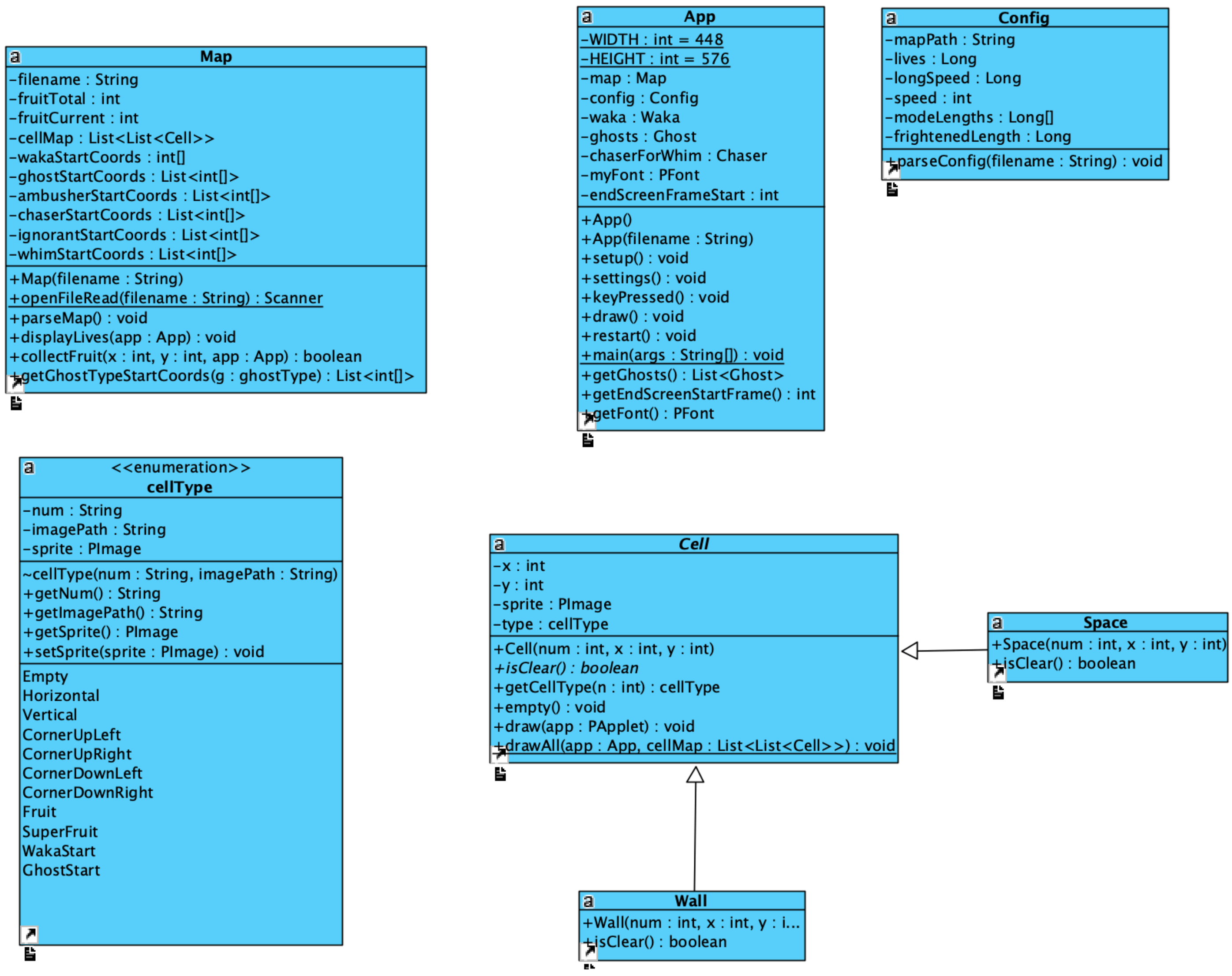
> **Relevant Classes:**
>
> - Cell *(abstract)*
>   - Space
>   - Wall
> - cellType *(enum)*
> - Map
>   The third component is the map. Using the filename given by the configuration file, the Map class parses the file into a two dimensional list of cells. The Map class stores information about the game state, specifically total and current count of fruit.

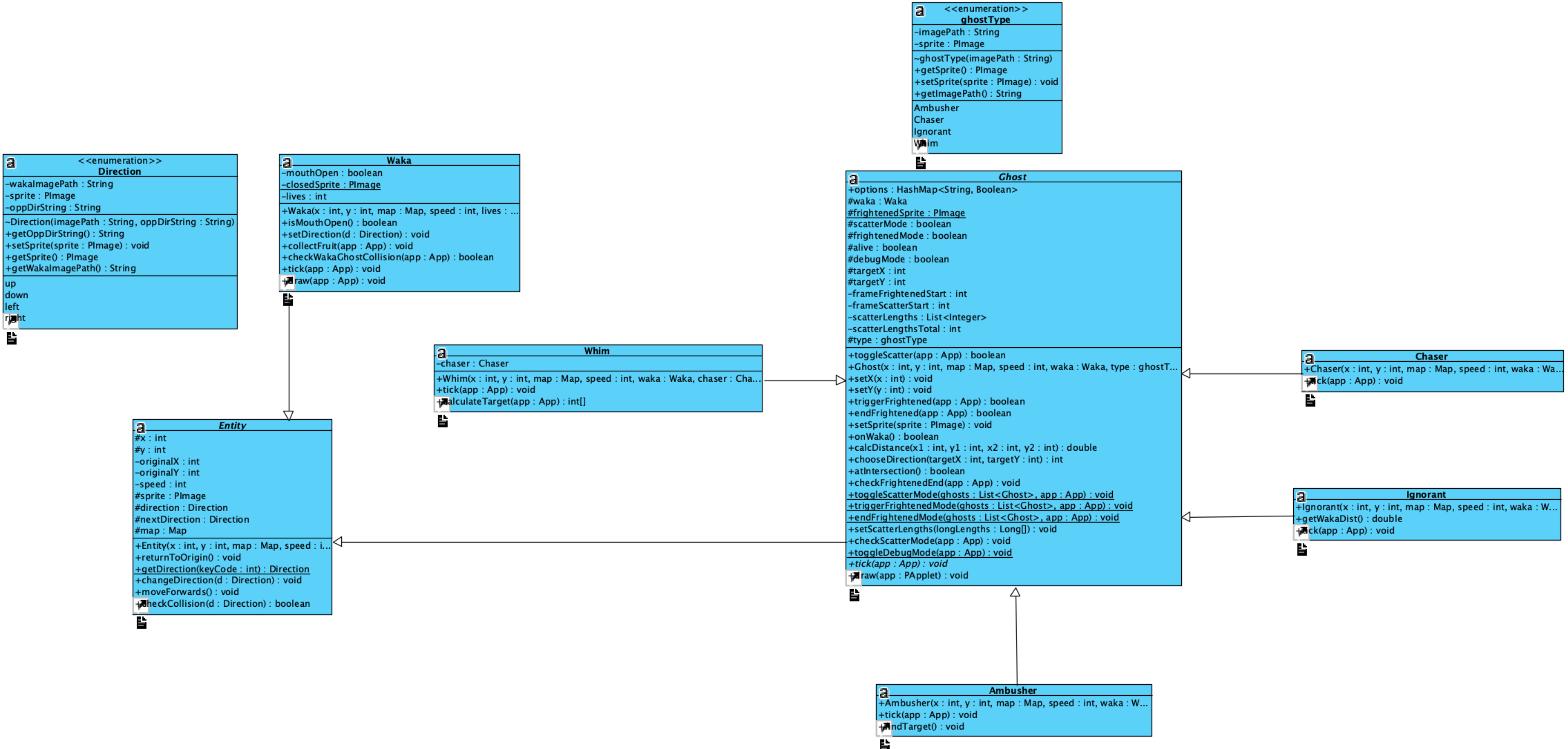
The Cell class is abstract, and holds the x and y coordinates, sprite and specific type for each cell. Cells can broadly be split into 2 categories, Walls and Spaces.

Spaces, (fruit and empty cells) can be moved through freely by entities and implement the isClear() method to return true.

All variations of walls cannot be passed through, and implement isClear() to return false accordingly.

The cellType enum holds the image file for each specific type of the cell, e.g. upRightCornerWall, so that the path can be drawn onto the screen.


Map
-filename : String
-fruitTotal : int
-fruitCurrent : int
-cellMap : List<List<Cell>>
-wakaStartCoords : int[]
-ghostStartCoords : List<int[]>
-ambusherStartCoords : List<int[]>
-chaserStartCoords : List<int[]>
-ignorantStartCoords : List<int[]>
-whimStartCoords : List<int[]>
+Map(filename : String)
+openFileRead(filename : String) : Scanner
+parseMap() : void
+displayLives(app : App) : void
+collectFruit(x : int, y : int, app : App) : boolean
+getGhostTypeStartCoords(g : ghostType) : List<int[]>
App
-WIDTH : int = 448
-HEIGHT : int = 576
-map : Map
-config : Config
-waka : Waka
-ghosts : Ghost
-chaserForWhim : Chaser
-myFont : PFont
-endScreenFrameStart : int
+App()
+App(filename : String)
+setup() : void
+settings() : void
+keyPressed() : void
+draw() : void
+restart() : void
+main(args : String[]) : void
+getGhosts() : List<Ghost>
+getEndScreenStartFrame() : int
+getFont() : PFont
Config
-mapPath : String
-lives : Long
-longSpeed : Long
-speed : int
-modeLengths : Long[]
-frightenedLength : Long
+parseConfig(filename : String) : void
<<enumeration>>
cellType
-num : String
-imagePath : String
-sprite : PImage
~cellType(num : String, imagePath : String)
+getNum() : String
+getImagePath() : String
+getSprite() : PImage
+setSprite(sprite : PImage) : void
Empty
Horizontal
Vertical
CornerUpLeft
CornerUpRight
CornerDownLeft
CornerDownRight
Fruit
SuperFruit
WakaStart
GhostStart
Cell
-x : int
-y : int
-sprite : PImage
-type : cellType
+Cell(num : int, x : int, y : int)
+isClear() : boolean
+getCellType(n : int) : cellType
+empty() : void
+draw(app : PApplet) : void
+drawAll(app : App, cellMap : List<List<Cell>>) : void
Space
+Space(num : int, x : int, y : int)
+isClear() : boolean
Wall
+Wall(num : int, x : int, y : i...
+isClear() : boolean


<<enumeration>>
ghostType
-imagePath : String
-sprite : PImage
~ghostType(imagePath : String)
+getSprite() : PImage
+setSprite(sprite : PImage) : void
+getImagePath() : String
Ambusher
Chaser
Ignorant
Whim
<<enumeration>>
Direction
-wakaImagePath : String
-sprite : PImage
-oppDirString : String
~Direction(imagePath : String, oppDirString : String)
+getOppDirString() : String
+setSprite(sprite : PImage) : void
+getSprite() : PImage
+getWakaImagePath() : String
up
down
left
right
Waka
-mouthOpen : boolean
-closedSprite : PImage
-lives : int
+Waka(x : int, y : int, map : Map, speed : int, lives : ...
+isMouthOpen() : boolean
+setDirection(d : Direction) : void
+collectFruit(app : App) : void
+checkWakaGhostCollision(app : App) : boolean
+tick(app : App) : void
+draw(app : App) : void
Ghost
+options : HashMap<String, Boolean>
#waka : Waka
#frightenedSprite : PImage
#scatterMode : boolean
#frightenedMode : boolean
#alive : boolean
#debugMode : boolean
#targetX : int
#targetY : int
-frameFrightenedStart : int
-frameScatterStart : int
-scatterLengths : List<Integer>
-scatterLengthsTotal : int
#type : ghostType
+toggleScatter(app : App) : boolean
+Ghost(x : int, y : int, map : Map, speed : int, waka : Waka, type : ghostT...
+setX(x : int) : void
+setY(y : int) : void
+triggerFrightened(app : App) : boolean
+endFrightened(app : App) : boolean
+setSprite(sprite : PImage) : void
+onWaka() : boolean
+calcDistance(x1 : int, y1 : int, x2 : int, y2 : int) : double
+chooseDirection(targetX : int, targetY : int) : int
+atIntersection() : boolean
+checkFrightenedEnd(app : App) : void
+toggleScatterMode(ghosts : List<Ghost>, app : App) : void
+triggerFrightenedMode(ghosts : List<Ghost>, app : App) : void
+endFrightenedMode(ghosts : List<Ghost>, app : App) : void
+setScatterLengths(longLengths : Long[]) : void
+checkScatterMode(app : App) : void
+toggleDebugMode(app : App) : void
+tick(app : App) : void
+draw(app : PApplet) : void
Whim
-chaser : Chaser
+Whim(x : int, y : int, map : Map, speed : int, waka : Waka, chaser : Cha...
+tick(app : App) : void
+calculateTarget(app : App) : int[]
Chaser
+Chaser(x : int, y : int, map : Map, speed : int, waka : Wa...
+tick(app : App) : void
Entity
#x : int
#y : int
-originalX : int
-originalY : int
-speed : int
#sprite : PImage
#direction : Direction
#nextDirection : Direction
#map : Map
+Entity(x : int, y : int, map : Map, speed : i...
+returnToOrigin() : void
+getDirection(keyCode : int) : Direction
+changeDirection(d : Direction) : void
+moveForwards() : void
+checkCollision(d : Direction) : boolean
Ignorant
+Ignorant(x : int, y : int, map : Map, speed : int, waka : W...
+getWakaDist() : double
+tick(app : App) : void
Ambusher
+Ambusher(x : int, y : int, map : Map, speed : int, waka : W...
+tick(app : App) : void
+findTarget() : void