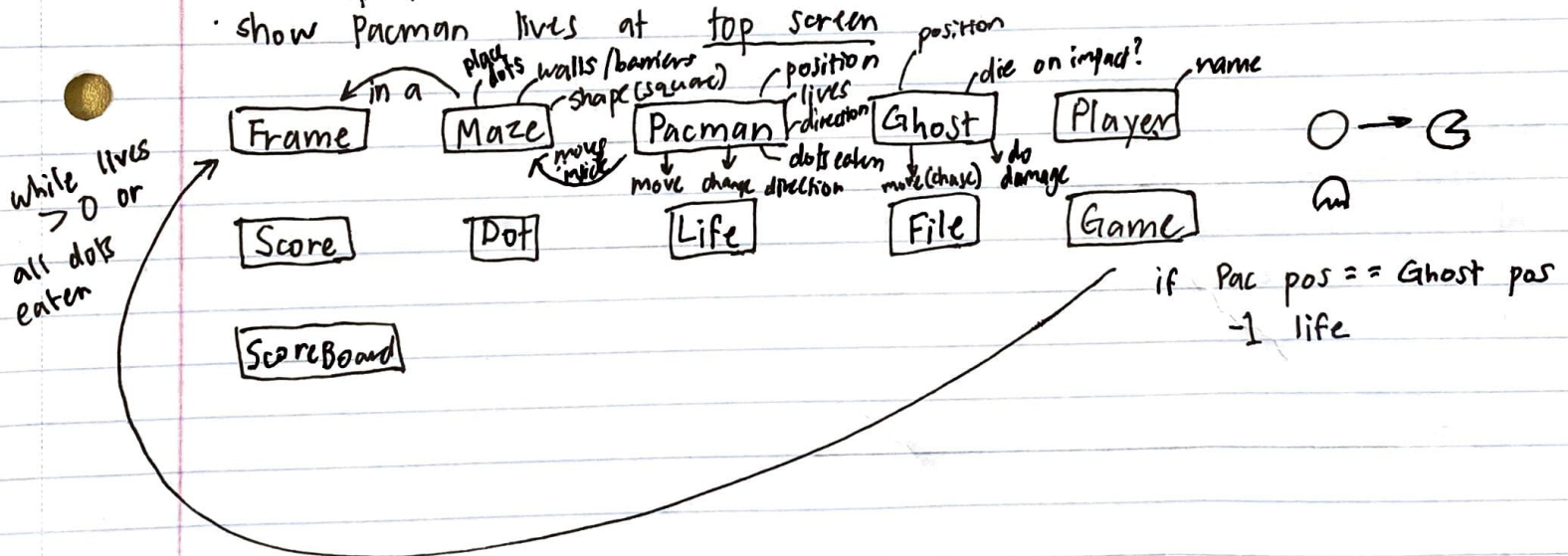


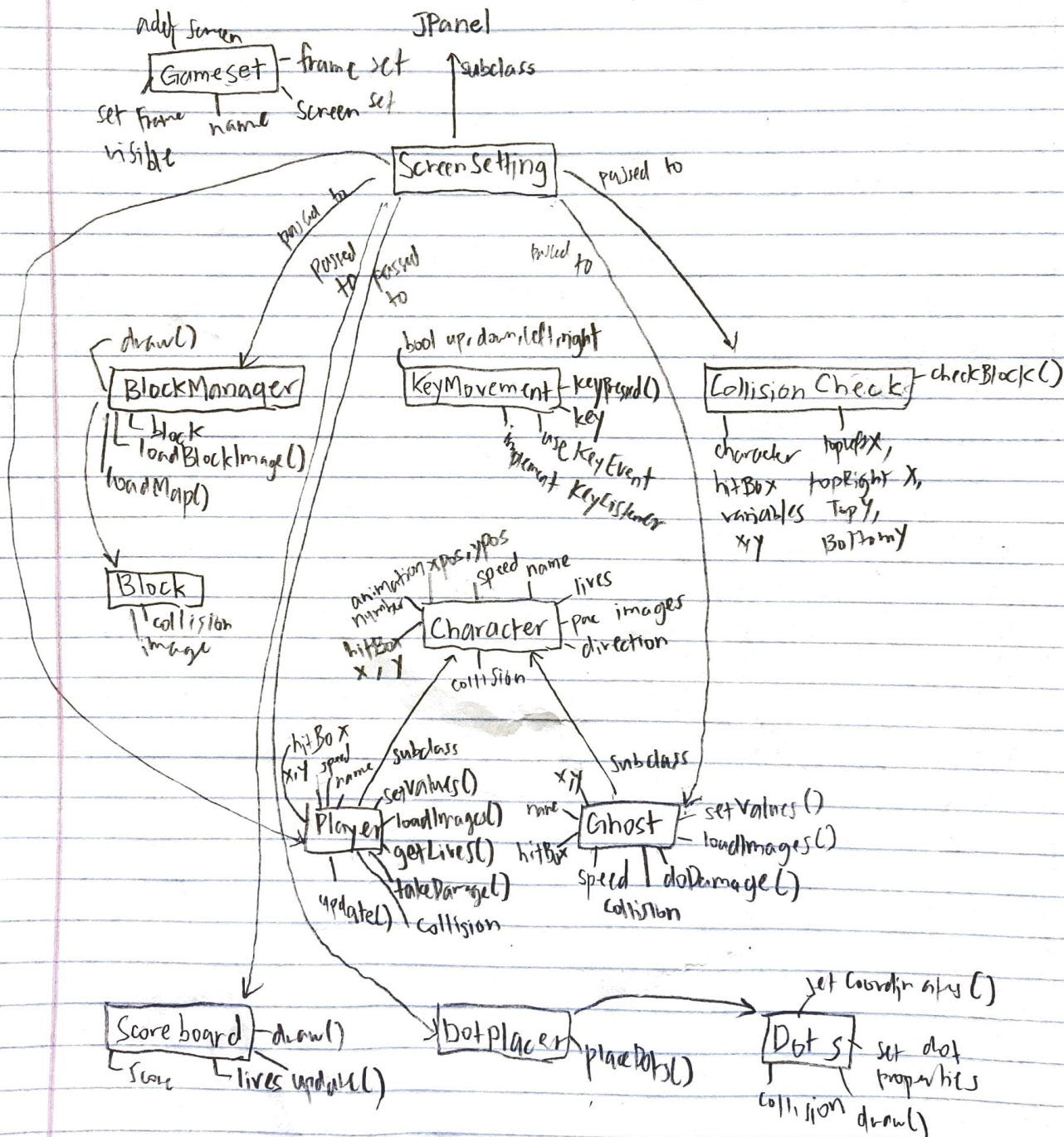
PAC-MAN PROJECT

- build a frame with a maze
- set Pac-man lives configurable (3)
- start Pacman at a specific positions
- start moving when key is pressed > event
- make Pacman eat dots as it moves
- stop Pacman at maze barriers or at end of frame
- allow keystrokes ($\leftarrow \uparrow \downarrow \rightarrow$) to change Pacman direction
- create ghosts that chase Pacman (only 1 needed)
- allow ghosts to take one life of Pacman on collision
- terminate game when Pacman loses lives or all dots are eaten
- tally all dots eaten and show at top of screen
- allow player to enter name & maintain score & all past scores > send to text file
- show Pacman lives at top screen



Pac-Man Design

[package com.usf.cs112] all names



MAZE BUILDING

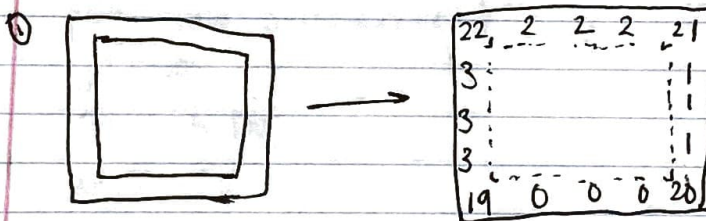
using block vals through numerical input from text file, create/draw objects with collision

ScreenSize: rows: 13 (additional row to give space at top to display lives (left))
columns: 12

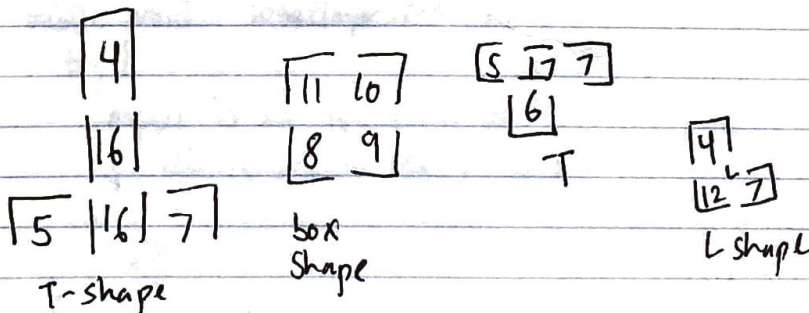
Block #	wall	topper	joint	jpoint	hall	square	corner	empty
0	—	4 □	8 L	12 L	16 11	18 □	19 L	23 (plain black square)
1	1	5 □	9 J	13 J	17 =		20 J	
2	—	6 □	10 T	14 T			21 T	
3	1	7 □	11 T	15 T			22 T	

image → - make all custom Sprites with pixel art

corner prevents awkward pixel gaps ex.)

basic screen filling idea (bordered screen)

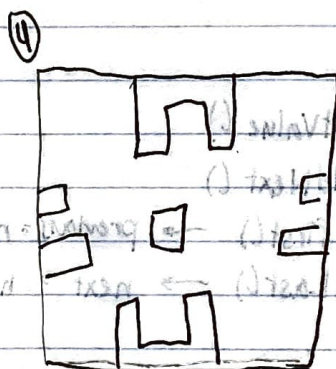
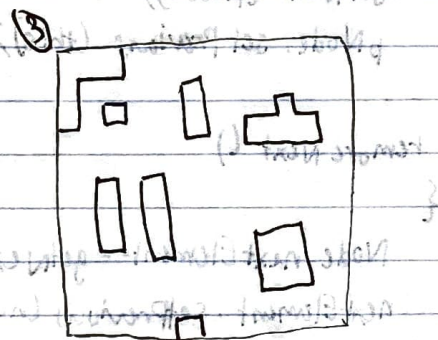
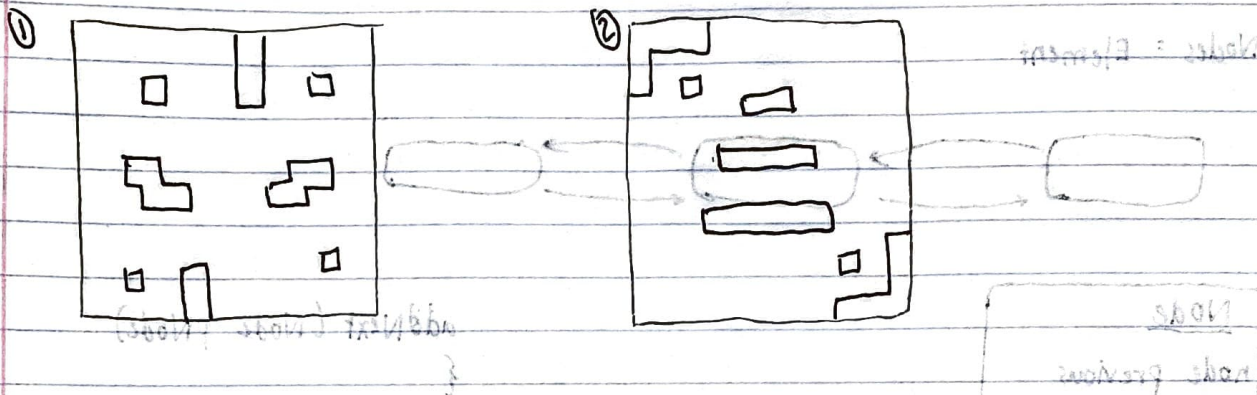
17
116 will be a gap here (add corner)

screen shape, making obstacles

any combination can be made, map completely generated from the values of the text file.

Maze Designs

20/10/20



Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

Maze 2 = 20/10/20

DIRECTION MECHANIC (PACMAN)

- X and Y positional variables for Pacman's position
- Separate Pac faces as GIF's so that he always 'eats' (chomps)

↳ pacUp, pacDown...

↳ each direction called based on $\uparrow \downarrow \leftarrow \rightarrow$

→ actual position updates until wall or barrier reached
using KeyEvent to register inputted direction, assign a
corresponding X or Y input to each press

CHARACTER CLASS

serves as a superclass method for both player & ghost classes

initialize^{int} positional variables x, y;

initialize integer speed;

initialize BufferedImage (all Pac images);

initialize string direction;

initialize animationCounter = 0;

initialize animation Number = 1;

→ 0 1 2 1 2 1 2 1 2

↖ build custom sprites
pixel-art style, 1 open
mouth & 1 closed per
each direction ↑ ↓ ← →

↓
naming convention:

up: pacU1, U2

Down: pacD1, D2

Left: pacL1, L2

Right: pacR1, R2

position,
speed,
and
direction
on
start

get all
sprite
images

PLAYER CLASS extends CHARACTER

Player(screenset, keyMovement)

setDefaults();

loadPlayerImage();

public void setDefaults() {

set x = 100;

set y to 100;

set speed to 4;

set direction to "right";

}

public void loadPlayerImage() {

try {

pacU1 is set to pacU1.png

pacU2 is set to pacU2.png

...

cont.

}

catch (IOException) {

// error statement if png files are not found

}

}

// cont. on next page ⇒

→ PLAYER CLASS PT. 2

update() {

if (key entered is Up)

direction set to "up";

if (key entered is down)

direction set to "down";

if (key entered is left)

direction set to "left";

if (key entered is right)

direction set to "right";

IDEAS FOR HOW ANIMATION OF PACMAN'S FACE CAN WORK





1. ImageIcons for each animation frame ex. (pacU1, pacU2)

- cycle between them if possible?

2. ImageIcons, import each direction as GIFS (pacU.gif, pacD.gif, pacL.gif, pacR.gif)

- cons: Pac face always moving, never will stop changing even if standing still

• regardless of strategy, movement / animation should function the same

• ← ↑ ↓ → directional input should trigger the respective directional Pac Face    

• timer of some sort required to cycle through animations automatically, instantiate Java Timer or other timer method to trigger updates to the display?

package com.usf.cs112

George Spilars

import Scanner
import JFrame

```
public class GameSet {  
    public static void main {  
        display "Enter name, player:"  
        initialize Scanner  
        set and initialize String name to user-inputted string  
        set JFrame  
        frame.set frame to exit on close  
        frame.set resizable to false  
        frame.set title to "Professor Pac-Man"  
        ScreenSetting screen = new ScreenSetting()  
        frame.add screen to frame  
        frame.set Visibility to true  
        screen.setupDots();  
        screen.startThread();  
    }  
}
```

implements Runnable

```

public class ScreenSetting extends JPanel {
    set final int originalTileSize to 16
    set final int scale to 3
    set final int tileSize to originalTileSize * scale
    set final int maxScreenColumn to 12
    set final int maxScreenRow to 12 → 13 (gap at top for score & lives display)
    set final int screenWidth to tileSize * maxScreenColumn
    set final int screenHeight to tileSize * maxScreenRow
    set final int framesPerSecond to 60
    set int score to 0
    BlockManager blockM = new BlockManager()
    KeyMovement keyM = new KeyMovement()
    Thread thread
    CollisionCheck cc = new CollisionCheck()
    DotPlacer dP = new DotPlacer()

    Player player = new Player()
    Ghost ghost = new Ghost()
    Ghost ghost2 = new Ghost()
    Ghost ghost3 = new Ghost()
    Dots dot[] = new Dots[150]
    Scoreboard scoreboard = new Scoreboard()

```

```

public ScreenSetting() {
    set panel size to Dimension(screenWidth, screenHeight)
    set panel background color to black
    add key listener to panel (keyM)
    set panel focusable
}

```

```

public void setupDots() {
    dPlacer.placeDots()
}

```

(class cont. on next) →


```
public class ScreenSetting (continued)
    public void startThread() {
        thread = new Thread
        thread.start()
        playMusic() }
    public void run() {
        set double drawRate = 1000000000 / framesPerSecond
        set double nextDrawTime to System.nanoTime() + drawRate
        while (thread is not equal to null) {
            player.update()
            ghost.update()
            ghost2.update()
            ghost3.update()
            scoreboard.update()
            repaint()
            try {
                set double timeLeft to nextDrawTime - System.nanoTime()
                reassign timeLeft to timeLeft / 1000000
                Thread.sleep(timeLeft)
            } catch (InterruptedException) {
                exception StackTrace
            }
            reassign nextDrawTime to nextDrawTime plus drawRate }
        }
    }
```