Project 2: Building Multiple Pacman Game

Emanuel Calderon

John Hopkins University

Table of Contents

[Building Multiple Pacman Game 3](#_Toc69067796)

[Design and Analysis: 3](#_Toc69067797)

[Prior Build to Current Build 3](#_Toc69067798)

[General Programming Design 3](#_Toc69067799)

[Alternatives 4](#_Toc69067800)

[Learning Outcomes 5](#_Toc69067801)

[Wrap Up: 5](#_Toc69067802)

[References 6](#_Toc69067803)

[Appendix A 7](#_Toc69067804)

# Building Multiple Pacman Game

This writing contains information for solution of Pacman game built using Java as directed by Project 2 assignment. The objective of this assignment was to utilize the skills and tools gained from the beginning of the course to the now module 11 learnings. The task is to create a game and have a bit of fun doing it.

## Design and Analysis:

The program created to meet minimum requirements of this project. The program had its challenging moments compared to the initial challenge of the project. Much of the decisions made in the prior building either assisted or made programming difficult for current parameters.

### Prior Build to Current Build

Some of the items which helped the programmer convert over to the multi-Pacman play was cleaning of the prior build via methods and excessive comments to both better debug and transition to multicollection of cookies. Due to placed methods prior, when converting over to a “.java” class (see Figure 2 example), it was a simple copy and paste with minor changes. Rather than having to understand and do things by scratch. The excessive comments helped with major debugging and pointed the programmer to get rid of the unnecessary loops which caused most of the crashes. The comments on the cookie randomization made it easier to reverse engineer and add in the numbered values.

### General Programming Design

The program design is similar to project 1 with exception of the changes performed. One thing to note is the failure to write out the “.txt” file while continuing the game, the knowledge was not obtained in time.

Program organization follows as (1) Variable call outs (2) Logic to load text or input data (3a) User input txt file is loaded with all the saved data within it which runs through read command to jump you into game play (3b) User inputs the data similar to project 1 with the exception of the different input requests made, depending on how many Pacman user inputs means more typing (4) Actual game play for user to input five commands for the Pacman switch play (5) End statistics for separate Pacman (6) Carried over methods from project 1 (8) PacMans.java Class.

Current program layout includes main, methods and a class. Class is used to assist with the benefits of the use of methods. Class does a good job to keep track of movements and cookies. Structure of entire code is a mixture of arrays, for’s, if’s, switches, objects write outs and read ins. Within callouts, small arithmetic was used to complete desired action placed by player/user. Methods and class call outs were used wherever possible as to reduce the redundancies of the algorithm, shorten huge lines of code in main body and make the code look neater. It should be said that the code could do some work to reduce the number of lines inputted in it.

The Pacmens state is maintained (using PacMans.Java) via switch/if statements. The main method calls upon the class to retrieve the location and logic for position, see Figure 1-2. Essentially, program is being told to simply find the position of the Pacman in question and then rotate it per the user’s command. Compared to project 1 the class simplified and eliminated for loops.

### Alternatives

The code looks messy and very long to sift through. Going forward there is much redundancies that could be shrunk by setting up the right classes and taking out the extensive logic (like the initial direction code that can be reduced see Figure 3). There were some algorithms that made it impossible to replace with methods given the number of variables needed to be returned. Specifically, the moving of the Pacmen. It may be possible, though it would have had major overhaul of the code. Alternatives probably include using collections for the cookie logic. Only one class was set up to believing having the need of a “Move”, “Rotate”, etc… class would be more redundant. Though looking back, attempting to have “cookie” as a collection and the “Pacman on top of Pacman” logic might free up the “move” logic to become a class. For now I was well enough with the one class. Using the Extend feature may have helped also in some cases to open more return values for the move of the Pacmen. To reiterate there is much room for optimization.

### Learning Outcomes

There were much learning outcomes in this project than the other. (1) The benefit of utilizing other classes for proper collection of moves/cookies for the Pacmen (2) It strengthened the confidence in understanding the use of the Class and write/read statements (3) Improved understanding on how to use probability with loop functions (4) Learned new cookie randomization feature for future projects.

The code was completed with decent time left which gave the programmer time to clean up other than write/read feature. Movement of Pacmen was completed about a week ago.

## Wrap Up:

This project challenged students in many ways. Not only were they building a new program (if they opted to do so) they were also challenged to incorporate new tools gained while working through the modules. The learning outcomes will be very useful future assignments and projects performed in the remainder of the course.

# References

Deal, J. (2021, 04 11). *Mini Project 2.* Retrieved from JH: https://blackboard.jhu.edu/bbcswebdav/pid-9226731-dt-content-rid-98449282\_2/courses/EN.605.201.81.SP21/Mini%20Project2ClassesInstructionsSpr2020%281%29.pdf

Deal, J. (2021, 1 1). *Mini Project1 Instructions Spr2021*. Retrieved from 605.201 Mini-Project 1: https://blackboard.jhu.edu/bbcswebdav/pid-9226692-dt-content-rid-99036029\_2/courses/EN.605.201.81.SP21/Mini%20Project1InstructionsSpr2021.pdf

Purdue University. (2019, 10 1). *Purdue Online Writing Lab*. Retrieved from APA Style Introduction: https://owl.purdue.edu/owl/research\_and\_citation/apa\_style/apa\_style\_introduction.html

Schildt, H. (2019). *Java The Complet Reference Eleventh Edition.* Chicago: McGraw-Hill Education.

Streefkerk, R. (2017). *APA format for academic papers (6th edition)*. Retrieved from Scribr: https://www.scribbr.com/apa-style/6th-edition/archived-format/

# Appendix A

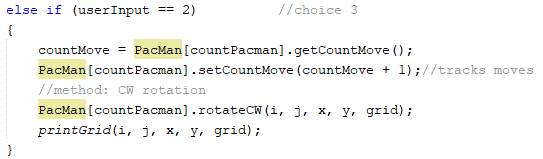


Figure . User input choice 1 Pacman rotation

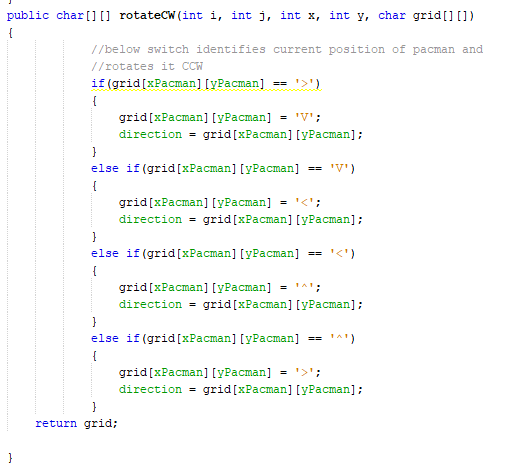


Figure 2. Class call out to rotate, direction inherits for write out

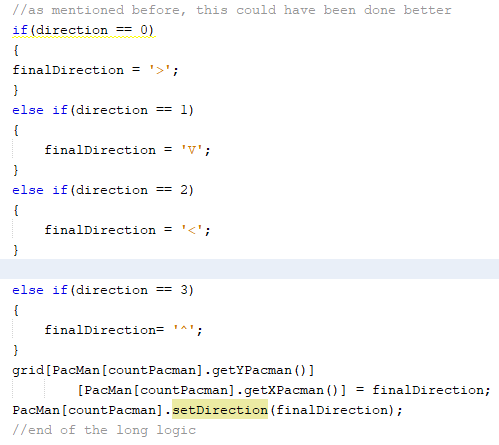


Figure 3. Clean up to reduce lines