**Design Specification**

**Version 1.2**

**Pacman(Wackman)**

***The Dot Eater***

**Developers:**

Joshua Pritchett

Aaron Rice

Stuart Morgan

Elec McClellan

Contents

[Introduction 3](#_Toc417304398)

[Design Overview 3](#_Toc417304399)

[Assumptions and Dependencies 4](#_Toc417304400)

[Goals and Guidelines 4](#_Toc417304401)

[System Architecture 5](#_Toc417304402)

[Detailed System Design 7](#_Toc417304403)

[How to Run Program 12](#_Toc417304404)

[Abbreviations and Acronyms 14](#_Toc417304405)

[References 15](#_Toc417304406)

# Introduction

This document is designed to be a reference for any person wishing to implement the Pacman program created by: Joshua Pritchett, Stuart Morgan, Aaron Rice, and Elec McClellan. This document describes what is needed to run the program, how to run the program, what the program does, how the program works, why the program was built the way it was, and how to actually play the finished product.

This document should be understandable to any individual possessing even a basic technical background. However, if there is something that is not understood, refer to both the Glossary and Acronyms and Abbreviations pages at the end of this document.

This document includes but is not limited to the following information for Pacman; program overview, design considerations, assumptions and dependencies, program fallacies, goals and guidelines, architectural strategies, system architecture, and detailed system design.

# Design Overview

The Pacman game is composed of three primary components: Assets which is responsible for storing the necessary images and music that bring professionalism to the game, the library which consists of the game engine that provides structure and more common design patterns, and lastly the source code which is the root of the functionality of the game and is composed of three main classes. These classes are: components.js, game.js, and scenes.js and will be explain more thoroughly and detailed in the System-Architecture portion of this document.

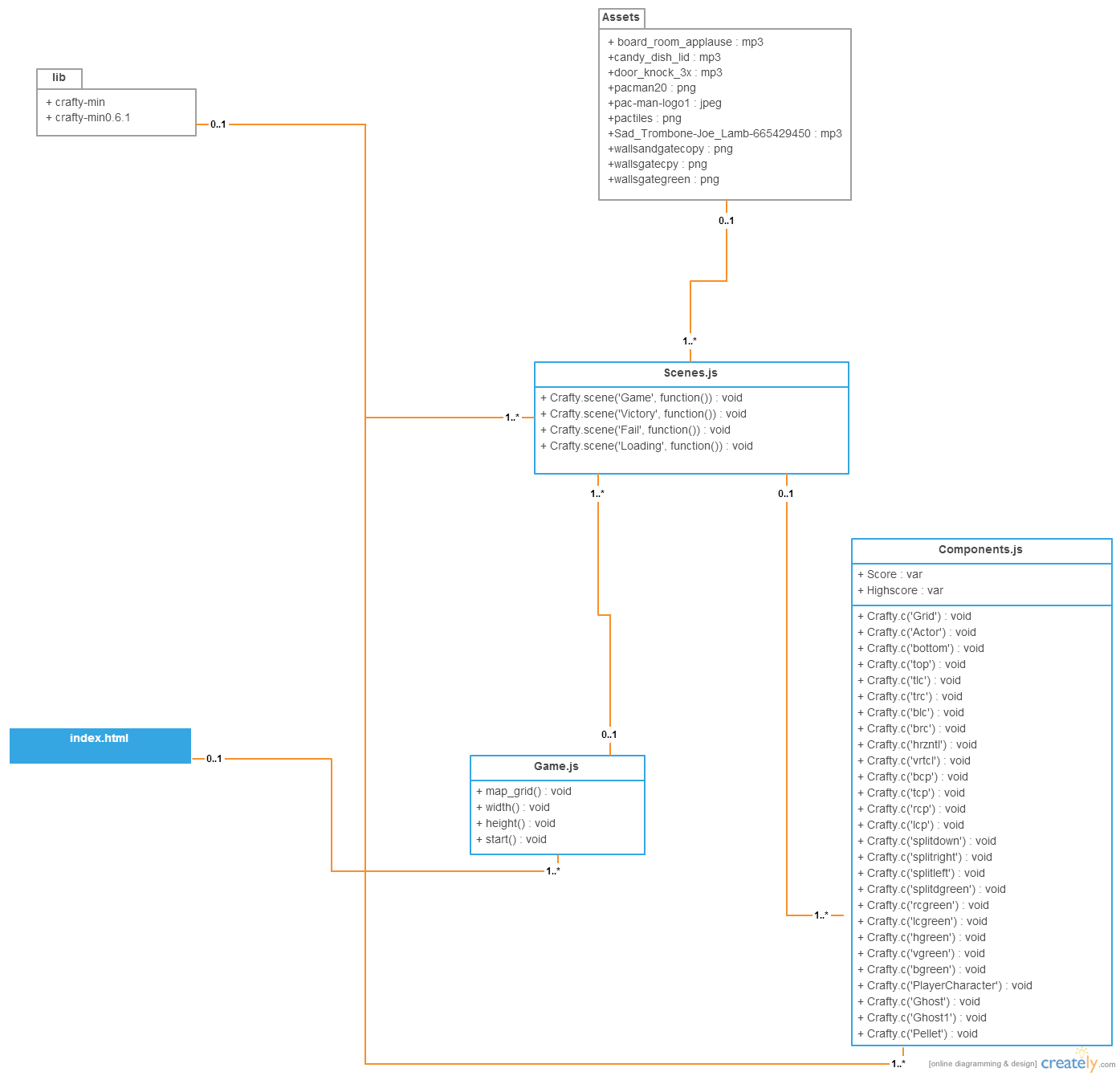
# Assumptions and Dependencies

1. Client has functioning Web Browser to run the Pacman application

# Goals and Guidelines

The major goal of this project is to create a fully-functioning Pacman that resembles as closely and accurately as possible to the original Pacman game created by Namco in 1980. Working in a team of four individuals to partition the workload and work in a cohesive, effective team to accomplish this goal was our number one priority. The goals for this project were, in no particular order: create a map of the same design pattern as the original Pacman, produce the character Pacman with the capabilities of moving in all four directions on an arrow key press, moving Pacman’s mouth open and closed when approaching a pellet, producing the Ghosts characters that seek out Pacman to collide with him in which he loses a life, produce similar background noises to that of the original Pacman, produce an interactive menu for the player and finally to keep a local score and a high-score value to be displayed for the current player.

# System Architecture



The Pacman game has been divided into three classes; game.js, components.js, and scenes.js.

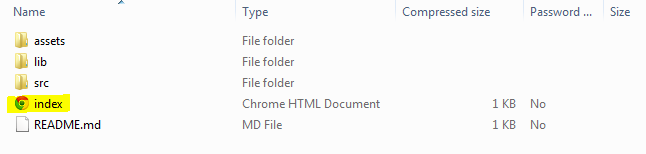
1. Game.js
   1. Defines the map size and initializes the start of the game
2. Components.js
   1. Defines the different components of the game such as:
      1. Locating an element on a grid of tiles
      2. Creating an instance of an “actor” or any “entity” that is drawn on the background via the logical grid coordinate
      3. Creates the instances of the solid walls for either blue or green
      4. Creates the Pacman character and the functionality and animation of such
      5. Creates the Ghost character’s and the functionality and animation of such
      6. Creates an instance of the pellets and functionality of Pacman visiting the pellet
3. Scenes.js
   1. Creates a gridded-map for the game in order to implement the walls and pellets and also to keep up with each component (Ghost, Pacman, Pellets, etc.) at all times at all locations.
   2. Implements the background sounds for the game
   3. Creates the victory scene assuming that the player wins the game which tells the player that they visited all of the pellets and allows for the player to restart the game if they so choose
   4. Creates a game over screen assuming that the player collides with a ghost character in which the player is informed that the game is over and allows the player to restart the game is they so choose
   5. Creates a loading screen in which the assets are loaded

# Detailed System Design

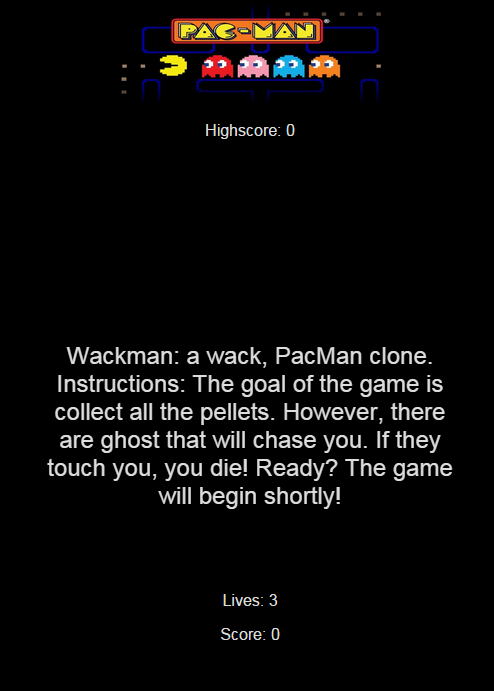
1. Game.js
   1. Map\_grid:
      1. Defines the width and height of the map size and tile size
   2. Width:
      1. The total width of the game screen
   3. Height:
      1. The total height of the game screen
   4. Start:
      1. Initializes and starts the game by calling the “loading” scene from scene.js
2. Components.js
   1. Crafty.c(‘Grid’)
      1. Init:
         1. Creates a variable w to represent the map grid tile width
         2. Creates a variable h to represent the map grid tile height
      2. At:
         1. Used to locate the entity or “actor” at the given position on the grid
   2. Crafty.c(‘Actor’)
      1. Init:
         1. Creates an “actor” object which is any entity drawn on the map
   3. Crafty.c(‘bottom’)
      1. Init:
         1. Creates a solid actor that is the bottom
   4. Crafty.c(‘top’)
      1. Init:
         1. Creates a solid actor that is the top
   5. Crafty.c(‘tlc’)
      1. Init:
         1. Creates a solid actor that is the top left corner
   6. Crafty.c(‘trc’)
      1. Init:
         1. Creates a solid actor that is the top right corner
   7. Crafty.c(‘blc’)
      1. Init:
         1. Creates a solid actor that is the bottom left corner
   8. Crafty.c(‘brc’)
      1. Init:
         1. Creates a solid actor that is the bottom right corner
   9. Crafty.c(‘hrzntl’)
      1. Init:
         1. Creates a solid actor that is the horizontal piece
   10. Crafty.c(‘vrtcl’)
       1. Init:
          1. Creates a solid actor that is the vertical piece
   11. Crafty.c(‘bcp’)
       1. Init:
          1. Creates a solid actor that is the bottom cap
   12. Crafty.c(‘tcp’)
       1. Init:
          1. Creates a solid actor that is the top cap
   13. Crafty.c(‘rcp’)
       1. Init:
          1. Creates a solid actor that is the right cap
   14. Crafty.c(‘lcp’)
       1. Init:
          1. Creates a solid actor that is the left cap
   15. Crafty.c(‘splitdown’)
       1. Init:
          1. Creates a solid actor that is the split down
   16. Crafty.c(‘splitright’)
       1. Init:
          1. Creates a solid actor that is the split right
   17. Crafty.c(‘splitleft’)
       1. Init:
          1. Creates a solid actor that is the split left
   18. Crafty.c(‘splitdgreen’)
       1. Init:
          1. Creates a solid actor that is the split down green color
   19. Crafty.c(‘rcgreen’)
       1. Init:
          1. Creates a solid actor that is the right cap green color
   20. Crafty.c(‘lcgreen’)
       1. Init:
          1. Creates a solid actor that is the left cap green color
   21. Crafty.c(‘hgreen’)
       1. Init:
          1. Creates a solid actor that is the horizontal green color
   22. Crafty.c(‘vgreen’)
       1. Init:
          1. Creates a solid actor that is the vertical green color
   23. Crafty.c(‘bgreen’)
       1. Init:
          1. Creates a solid actor that is the bottom green color
   24. Crafty.c(‘PlayerCharacter’)
       1. Starts Pacman going in the right direction
       2. Init:
          1. When Pacman collides with either a pellet or ghost, it calls either the visitPellet method or die method respectively
          2. Stores the direction (based on the directional arrow key pressed) in order to direct Pacman in that direction when Pacman moves by an opening in the direction of the arrow key that was pressed
          3. Allows for continuous movement of Pacman by running a loop with that direction of movement as a parameter until another directional arrow key is pressed
       3. visitPellet:
          1. responds to Pacman visiting a pellet
       4. update:
          1. Depending on the directional arrow pressed, the corresponding directional Pacman image will be displayed to match
       5. getX:
          1. returns the current horizontal positon on the map
       6. getY:
          1. returns the current vertical position on the map
       7. tryMove:
          1. Updates the current position of Pacman according to the key pressed
          2. If Pacman hits a wall, then Pacman stays in the current position until another arrow key is pressed
       8. Die:
          1. Calls the “fail” scene from scenes.js class
   25. Crafty.c(‘Ghost’)
       1. Init:
          1. Initializes the four different directions a ghost can go
          2. If a ghost hits a wall, this function generates a random number between 0-10 and splits this number range into four different parts to represent the four different directions (i.e 0-3 represents the ‘right’ direction). Whichever range of number is generated decides the direction the ghost will travel next.
       2. getX:
          1. returns the current horizontal positon on the map
       3. getY:
          1. returns the current vertical position on the map
   26. Crafty.c(‘Ghost1’)
       1. Init:
          1. Initializes the four different directions a ghost can go
          2. If a ghost hits a wall, this function generates a random number between 0-10 and splits this number range into four different parts to represent the four different directions (i.e 0-3 represents the ‘right’ direction). Whichever range of number is generated decides the direction the ghost will travel next.
       2. getX:
          1. returns the current horizontal positon on the map
       3. getY:
          1. returns the current vertical position on the map
   27. Crafty.c(‘Pellet’)
       1. Init:
          1. Gives the required inheritance for this method
       2. Visit:
          1. If the current position of Pacman contains a pellet, this method removes the pellet from the map, plays an audio sound, and trigger every callback attached to this visitation
3. Scenes.js
   1. Crafty.scene(‘Game’)
      1. Creates a 2-dimensional array to keep track of all tiles on the map
      2. Places Pacman at the coordinate where x is 10 and y is 15
      3. Keeps track that the tile located at this coordinate is occupied
      4. Places ghost one at coordinate (19,1)
      5. Places ghost two at coordinate (1, 1)
      6. Creates a character map to determine what is to be placed at which tile based on a letter representation that corresponds to the components in the components.js class
      7. Goes through each tile on the character map and depending on what letter is represented in that tile determines what is placed there (i.e. if there is a ‘p’ at coordinate (3,14) then a pellet will be placed at (3,14). (See character map chart in the Abbreviations and Acronyms section)
      8. Creates an initial sound depicting the start of the game
      9. Once all of the pellets have been visited, calls for the ‘Victory’ scene
   2. Crafty.scene(‘Victory’)
      1. Displays text stating that “All pellets visited!”
      2. Plays a sound that represents applause from a crowd
      3. After a short delay, allows the option for the player to restart the game in which the ‘Game’ scene is called again by the pressing of the down arrow key
   3. Crafty.scene(‘Fail’)
      1. Displays text stating that “Game Over. Press any key to play again.”
      2. Plays a sound that represents sad music for failing to win
      3. After a short delay, allows the option for the player to restart the game in which the ‘Game’ scene is called again by the pressing of the down arrow key
   4. Crafty.scene(‘Loading’)
      1. Displays text stating that the game is “Loading; please wait…”
      2. Loads all of assets necessary to be utilized in the game
      3. Onces all of the images are loaded, it utilizes the ‘sprite’ method to crop the images needed for the map walls, characters, pellets, etc.
      4. Defines the sounds that will be utilized in the other scenes
      5. Calls the ‘Game’ scene which starts the game

# How to Run Program

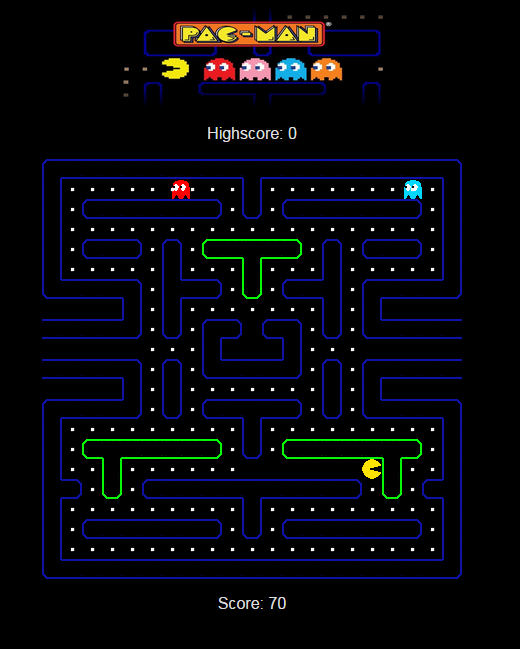
1. Included in the .zip file will be a HTML document named ‘index.” The first thing that must be done is this zipped file must be extracted. This is done by right clicking on the .zip file and clicking the option to extract all. Next, simply click on the option that says “extract”. In whichever directory you saved the .zip file, the extracted file will by default be saved to the same location. Simply open this now extracted folder and you will find a HTML document named “index.” Open this HTML file to run the program.



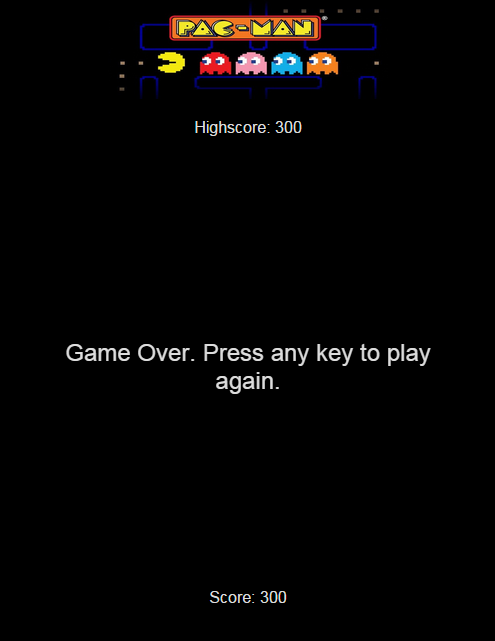
1. The first thing you will see is the menu screen which displays the title of the game and the instructions on how to play



1. The next thing you will see is the game screen in which you are now playing Pacman.



1. Simply utilize your direction arrow keys to move Pacman in whichever direction you so desire and collect all of the pellets but stay away from the ghosts!
2. In the case that you collide with a ghost before collecting all pellets, you will be directed to a screen that looks like below. In this event, press any button to play again.



1. In the event that you win, you will be directed to a screen that looks like below. In this event, press down to play again.



# Abbreviations and Acronyms

1. Character Map
   1. L Top Left Corner
   2. w Horizontal
   3. X Top Right Corner
   4. D Vertical
   5. p Pellet
   6. s Split Down
   7. b Bottom
   8. r Right Cap
   9. l Left Cap
   10. g Split Down Green
   11. m Horizontal Green
   12. k Left Cap Green
   13. e Right Cap Green
   14. v Vertical Green
   15. a Bottom Green
   16. Y Bottom Left Corner
   17. Q Bottom Right Corner
   18. t Top
   19. o Split Right
   20. u Split Left
   21. F Bottom Right Corner

# References

* "Build New Games." *An Introduction to the Crafty Game Engine -*. N.p., n.d. Web. 23 Mar. 2015. <http://buildnewgames.com/introduction-to-crafty/>
* "Crafty -." *Crafty -*. N.p., n.d. Web. 15 Mar. 2015. <http://craftyjs.com/api/>