CSS2A spring 2015

Game Project Report

5 May 2015

**Contributors:** Salvador Ramirez, Karina Pizano, Noemi Cuin, Jose Sanchez

**UML DIAGRAMS:**

|  |
| --- |
| Board |
| **-** Layout[22][31]:char  **-** score:int  - life: int  +getPosition(x:int):string  +getLayout1(x:int, y:int):char  +getScore():int  +getLife():int  +setLayout1():void  +setPosition(x:int,y:int,symbol:char):void  +setLife(life:int):void  +setScore(score:int):void  +setLayout1():void  +placeGhost(x:int, y:int):void  +DisplayCounters():void |

|  |
| --- |
| Pac-Man |
| **- pacman:char**  **-rowX:int**  **-colY:int**  **-pellets:int**  **-hasLost:bool \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **+PacMan():constructor**  **+**  **+getPacMan():char**  **+getRowX():int**  **+getColY():int**  **+setPacman(pacman:char):void**  **+setRowX(rowX:int):void**  **+setColY(colY:int):void**  **+isAttacked:bool**  **+hasLostGame(Board& grid):bool**  **+hasWon(Board& grid):bool**  **+isDot(Board& board):void** |

|  |
| --- |
| Ghost |
| -x\_coordinate:int | +Ghost():constructor  -y\_coordinate:int | +Ghost(x:int, y:int):overloaded constructor  -previous\_x:int | +setPosition(x:int, y:int):void  -previous\_y:int | +differentRoutes(Board& grid):void  -directions[4]:bool | +setGhost(Board& grid):void  -previousDirecction[4]:bool | +CalculatePAth(PAcMan& pacman, Board& grid):void  -symbol:char | +getXcoordinate():int  -symbolOnGrid:char | +getYCoordinate():int |

**Brief Descriptions:**

Pac-Man:

We needed a class/header file of Pac man himself because we need to know his specific characteristics that make him Pac man such as lifes and points. We also need Pac man’s position in order to use it on other classes. Also since Pac man is controlled by the user we need to place him on the grid and help him move. Before he moves we need to make sure that there is no barriers blocking him and also if he intersects with the Ghost he dies until he runs out of lifes. He needs to win by eating all the dots and having at least one life left.

Ghost:

The ghost class is the class the will control how the ghost works. First of all we need to know where Pac man is in order to have the ghost follow him. Because the ghost will be placed in different places we are to make sure that before he moves the finds all the possible ways he can get to Pac man (Making sure there are no barriers) then the will calculate the fastest path to Pac man until he gets to Pac man. (So pretty much he is following Pac man throughout the whole game). Using the distance formula we are able to calculate the fastest way so the ghost can get on his way and trap pac man. We will be making 3 ghost that will be placed in different positions in order to attack pac man from different places, this was just different way and to make the game a little more challenging.

Board:

The Board class will make the Board and Display both the lifes and the Score. Takes in account where both the ghost and Pac man are in order to return what is at that particular place. This will help pacman and the ghost know if there is any barriers located so they can move or choose a different location. This class will also set the food which is also the called the dots.

Driver Class:

The driver class will make objects of all classes in order to get the game running. This will also call all functions like life and score and use the keyboard buttons(up, down, left and right) to help the pacman move. Make the game play. Pretty much runs the game. We did encounter a problem which was the glitching, were not yet sure what it was but it seems that it is too many frames per second since we have to keep calling the function from another class every time we make pac man move or of the ghost move.

**Summaries:**

Noemi Cuin:

During this project, my task was to complete the main implementation file for the entire Pac Man game. To complete this, I had to include all three classes that were: Board, Ghosts, and Pac Man. Although including three different classes wasn’t too difficult, the main struggle was creating a way that the user can control Pac Man so that he can move in whatever direction the user chooses. My initial thought process was to base the character’s movement on the selection of letters; however, I thought that it would be a better challenge to try to incorporate the use of the built in arrow keys on the keyboard. Doing some research into this technique, I came across a lot of examples that were based on the use of a Windows computer. Since I’m a Mac user, and the majority of my team was Windows based, I was forced to use my VMWare to set up the code in Windows. By including “Windows.h”, Code Blocks was able to allow me to use the arrow keys so that Pac Man could move based on the user’s input. This feature was the coolest thing I learned and I’m glad this project gave me the chance to work with it.

Karina Pizano:

I learned that you really don’t need to use classes you can literally do everything on the main, but using classes is harder to keep up with, but is also the most organized way I have seen so far. I did Pac man header file and the Pac man implementation file, which included Pac man himself, the position to keep track of him, and the functions to move Pac man. Also they include his life and points. It wasn’t much trouble until we couldn’t decide where exactly to make Pac man move in the main or in the Pac man class until we finally decided that Pac man is the one moving therefore it should be in the Pac man class. I think the hardest part was contributing Pac man’s position and understanding why I had to pass in objects into some of the functions. I also needed to think twice when moving Pac man (updating the previous position with a space). Starting the project I thought it would be easy fun game, but in reality it was a lot of thinking and problem solving, but I enjoyed talking about solutions and the outcomes. Also I got to listen to different perspectives of my teammates and understand their way of thinking.

Salvador Ramirez:

For the project, I was in charge of creating a class that contained the board for the game, functions that allowed the board to be accessed by other classes, and functions that allowed the score and lives to be displayed. Initially, I worked with different map layouts because we were going to create different levels, but the group decided to go with one map layout due to the time restrictions. I worked on the Board class for the pacman project, and continued adjusting the functions depending on what changes were required. I finished the Board class early so that my team members could use it because some functions of their classes relied on calling functions provided in the Board class. After finishing the board class I helped my team members with problems since they did the same for me. I learned that communication is very important in a big project because every team member approaches a problem differently, and this can lead to complications. In coding, I worked with arrays a lot, so I got a better understanding on how to access an array with different classes and how objects affect that interaction.

Jose Sanchez:

My part of the project was to build the Ghosts. In order to do this, I had to make a Ghost class and a Ghost driver so that they could be transported onto the Board. To create a good ghost, I used my past experience of playing PacMan and took notice that the purpose of the Ghosts was to chase PacMan and destroy him. If PacMan was attacked by a Ghost then he would lose a life and he had three lives in total. An important factor of one of the Ghosts was that they would move at random; so unlike PacMan, the user has no control on where or how the ghosts move. To be able to get the ghost to move, I had to make sure that the Ghosts were aware of their positions and make sure that they wouldn’t move to the same spot more than once. We had attempted to let it run randomly but it took a while for the ghosts to travel far. What I learned was that although the random feature is important, it’s also a good idea to incorporate certain conditions because otherwise random movements won’t progress the character far.

**Discussion:**

If we could restart the project we would do a little more organization before even starting to code, since in the beginning we had to change a lot of things from one file to another. We had the Driver file full of functions and user input that we really didn’t need most of the header files, so we ended up changing the whole code so that the driver file just made the objects and called the Board. We are using Aggregation instead of just one file so that it fits the requirements and makes the code a little shorter since it will be made into different parts. We decided on aggregation because it was necessary since Pac man has a board and Pac man has a ghost. We also believe that working on it sooner would have made our lives a little easier since we would have finished a long time ago. In fact if we had our own rules and no requirements we would use less classes and make it all into a file since it is easier but a bit slower. Maybe choose a different game or make Pac man a different character. We made Pac man ‘c’ character, but as we were looking for examples in the internet we could have made it into (<, >, ^, V) characters depending what way he is going. We believe most of the changes would be mostly of how to display it with color and maybe pictures or shades for a better program, but we needed time which we didn’t use as wisely as we could have.

**References:**

* YouTube (How implement the arrow keys in order to make Pac man move.)
* Google: Example of Pac man game already made. (Source Code)