PycMan

Jan Dziedzic

October 7, 2017

Contents

Part 1

Project Definition and Analysis

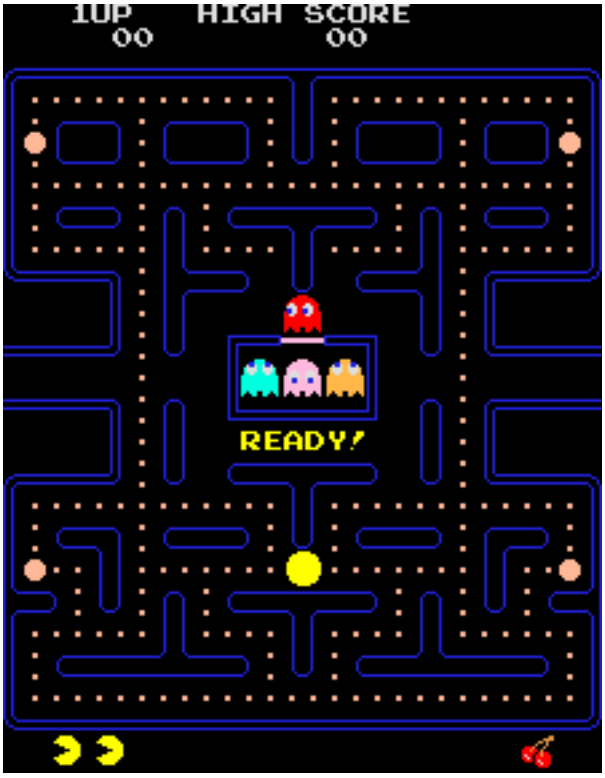

Figure 1.1: Original Pac-Man screen. Source is Reference 1.

1.1 Project Definition

As my project, I have decided to present a game written in Python, based around one of its modules - pygame. As I have not had as much time to prepare my project as my colleagues, I have decided that the concept of the game must be simple, yet allowing to demonstrate my programming and computational thinking skills. For this purpose I have decided to mimic the iconic PacMan game. For picture of the original game screen please refer to Figure 1.1

Called PycMan (utilizing the fact that many of Python modules have a prefix 'py' in their name), the game itself strongly resembles the original, yet with some meaningful changes. Like in the original - player can move in four directions (left, right, up and down), collecting coins and avoiding ghosts.

The game develops critical thinking skills and takes effect of human nature of taking risks and tackling unknown - with having more confidence when predicting ghost behavior, the player actually gets better when playing despite problems generally low complexity.

Actually the game itself may be referred as to playing tag in a transparent maze. Such a play would be very hard for a human to enjoy as analyzing transparent walls while running and predicting opponent's next move is a far too complex problem for human cognition. That's why computational approach on this game is required - where player can see ghosts from above the board whilst ghost may exhibit algorithmically advanced behavior.

1.2 Stakeholders

Proposed end user group is really wide. As the game itself can be launched on virtually any device due to Python versatility, it may be used on a spectrum of devices with different controllers, from mobile phones and smart watches through PCs reaching as far as game console emulators mimicking the most authentic PacMan game experience. Such portability should satisfy most users.

The game should grab attention of children - with small challenges that don't require complex thinking and rather promote manual skills and reflex; those who seek quick brainteaser whilst attempting to play with more focus on tactics of ghost avoidance and finally, old gamers who played original PacMan but now they might want to try more modern version with a different approach to critical game mechanisms.

1.3 Software Challenges

All software used to develop the game but is not necessarily required to run it:

- Python3
  - Pillow module
  - Pygame module
- Pycharm Professional
- GIMP - GNU Image Manipulation Program
- LaTeX
- TeXstudio

The greatest advantage that convinced me to actually use Python for my game was its portability. As an interpretable programming language, the code of the game remains universal throughout different operating systems and processor architectures. My decision was also influenced by my previous experience with this language and my desire to actually get better around it.

Also, whilst most of my experience was with Python2 I have decided that it's the right time to move on and actually start using Python3, which is reportedly faster and offers more functionality while providing greater stability.

Still, Python does not offer the speed one may expect from a programming language suitable for complex games. Advanced non-optimal solutions need to be thought of and substituted with faster algorithms in order for the game to run smooth on machines lacking processing power.

Pycharm by JetBrains was used as a Python IDE - it is a really convinient solution and comes with a lot of useful features. It was chosen due to my previous positive experience with it.

GIMP was used to produce graphics resources of the game as well as levels - described later.

LaTeXand Texstudio were used to develop this document.

All development was done on an Ubuntu-running computer, therefore I must proudly admit that during the entire design process no non-free software was used. All software was either opensource, freeware or free for education.

1.4 The Interview

To assess interest and gain insight of the potential market for the game, I have interviewed my fellow schoolmate - Wojciech Wojtkowski on his opinion of my approach to redesigning PacMan.

-Hello Wojciech, may I interest you with my IT project - the PycMan, next generation of the classic PacMan with smarter ghosts and different mechanics?

-Sure mate, go ahead!

-Ok, so these are some concept graphics [first two levels were exhibited]...

-...but it looks just like the original!

-Yes, there is a strong resemblance in terms of the graphics, that's what I am aiming for. The biggest difference is how the ghost work.

-Oh, tell me more.

-So, in the classical PacMan, the ghost have their designated area they launch from, A.K.A. "The Ghost House", I want to get rid of that, instead all ghosts will start from predefined locations different for each level.

-That means they will chase you from the exact beginning of the level, isn't that going to make the game harder?

-Yes it will, that is the goal. But that's not the main change. They will be smarter than the originals.

-You mean that the way they move is going to be less predictable?

-That's true, I want to utilize a rather complex algorithm to make them chase player more efficiently.

-Didn't the original have the most efficient solution?

-No, the target machines lacked processing power and memory in the past to actually use that with the game not slowing down. Now, when newer computers are available, I can actually use that.

-So if they were quite stupid then and the game was still hard, won't making them smart render the game impossible to win?

-That's what I am afraid of, I need to find a way to give the player some advantage. Do you, as an experienced gamer, have any idea how to do that?

-I think that making the ghost chase the player indirectly may be the way, how about them tailing the PycMan?

-Yeah, that migh be fine but this may eventually lead to them not catching it at all if it doesn't move.

-Oh, that might be the case.

-Actually I have one solution in mind - making them a little bit slower than the player.

-Seems okay, thought I think that one may run away from them, do a risky eating-maneuver then and regain the distance lost. Repeating that will make the game easy and very boring actually.

-Oh, true, I will have to think of these solutions. Probably final version will be designed during beta testing based on player satisfaction with each of these methods.

-Good idea to let the players decide. Actually - on the player-decision thing. One thing that I always wanted with PacMan is to design my own levels. Can you make it possible?

-Yeah, I already have a solution in mind that will make it very easy for anyone to design their own maze, if you say that's going to interest players, I will surely include that.

-Cool! Thanks for letting me know, can I play that game later?

-Of course, as soon as I release the beta.

-Perfect, thank you then, I look forward to playing it.

-Thank you for the talk and insight. Bye.

-Bye.

1.5 Requirement specification (success criteria)

For the project to succeed the following criteria must be met:

1. The game must resemble the original PacMan in terms of graphical design and some of the mechanics.
   (a) Game graphics should be of similar color and shape to the originals.
   (b) Player sprite is an iconic yellow ball with mouth.

(c) Ghosts chase the player who loses a life upon contact with a ghost.
(d) Upon losing a life ghosts and the player return to their initial locations.
(e) Player progresses through a level with eating 'coins' left throughout all accessible places on the map.
(f) Upon completing the level (eating all 'coins') new level is loaded.
(g) When player has less than one life the game finishes and the player loses.
(h) When player completes all the levels, the game finishes and the player wins.

2. The game must be different from the original in these ways:
   (a) It must be possible to win.
   (b) Majority of players must find the PycMan ghosts 'smarter' than the originals.
   (c) There is no 'ghost house' where ghosts start from. All sprites move from the beginning and ghost chase the player immediately.
   (d) There are no power-ups in the levels.
   (e) Ghosts cannot die.
   (f) Ghost don't change 'modes' and don't became frightened of the player.

3. Game must work smoothly (30 frames per second is considered to be the standard of human perception of fluency) on contemporary medium-class laptop PCs. The following hardware and software requirements are to be met:
   (a) Operating system supporting Python3 interpreter
   (b) Python3 interpreter
   (c) Pillow module (PIL)
   (d) Pygame module
   (e) Color display of resolution of at least 600x600px
   (f) Keyboard
   (g) 500 MB of storage space
   (h) 512 MB of RAM
   (i) 700MHz processor

   The game has been tested on described specification machine and has been found to meet fluency criteria. No testing on slower machines has been performed as these are not usually available on the market anymore.

Part 2

Design

As the game uses Pygame module it obviously derives some solutions natively implemented in it. All display solutions are actually handled using the module. Use of Tkinker was researched for pop-up messages but adding another module that doesn't bring any outstanding functionality above capabilities of Pygame has been ruled as an unnecessary waste of memory.

2.1 Main window layout

Main game window consists of a board where the actual game takes place. Below the board there is a set of informative counters kept in characteristic PacMan colours of gold-yellow on dark/royal-blue background.

2.1.1 Board

As each level has a different layout so look of the board may vary. Please refer to Figure 2.1 for an example of such board.

General idea is that all border tiles of each level (tile system explained later) have to be walls, which creates a nice, outer border of the board with rounded edges.

2.1.2 Counters

Directly below the board, counters are located, these provide information on:

1. Number of coins eaten
2. Number of coins that are required to be consumed before progressing to the next level
3. Number of player lives
4. Current level number

Every time the player eats a 'coin' the first counter increases. Throughout the level, this counter cannot decrease as even upon player's death, the already-eaten coins don't respawn.

Number of coins required to progress is constant through the level but may differ between levels. It is worth mentioning that as every coin must be eaten to progress, this acts as a total number of coins allocated in each level map. Coin placing algorithm described later in this document also proves that it equals to $32^2 - \#_{wall\_tiles} - \#_{empty\_tiles}$.

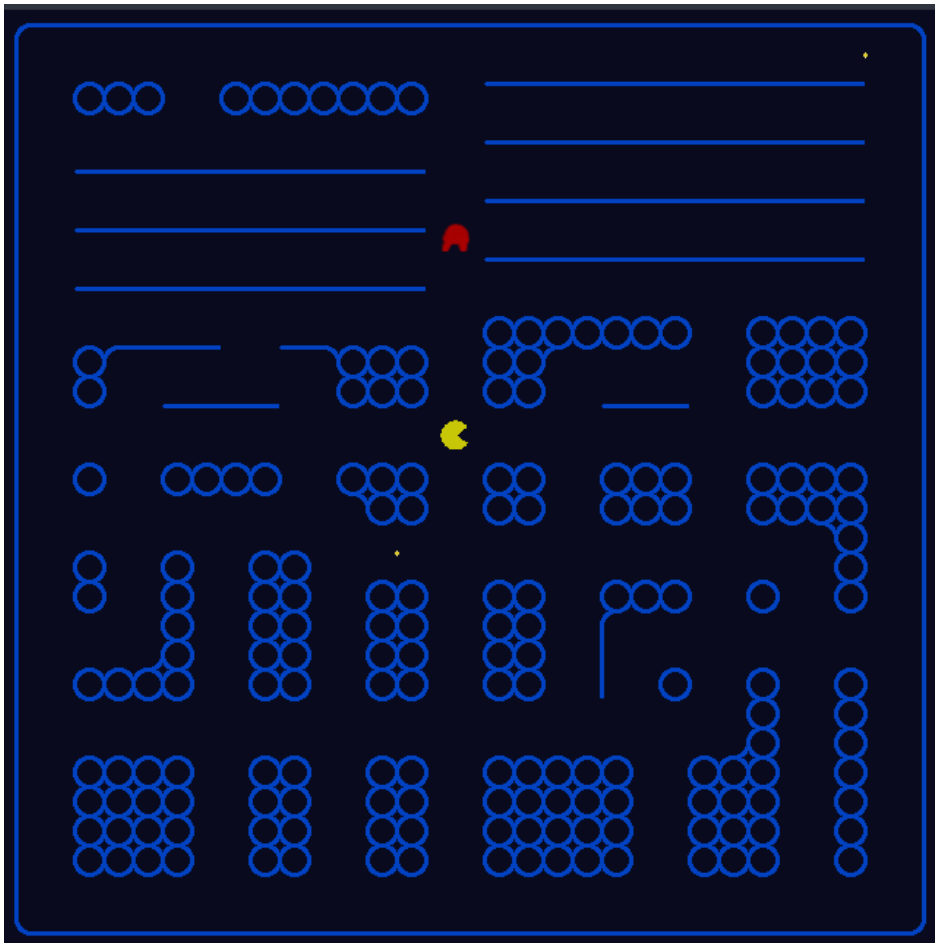

Figure 2.1: Example board. Walls, empty tiles, the player, coins and a red ghost are visible

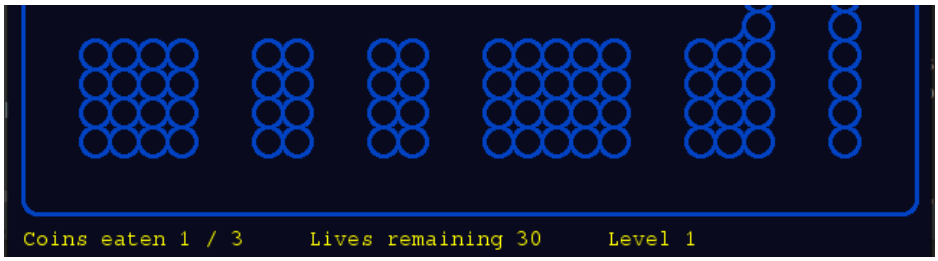

Figure 2.2: Example set of counters with part of the board included for position reference.

Each time a player comes in contact with a ghost(defined later in this document) a life is subtracted and every time the player eats a heart eatable, a life is added.

Level number increases from 1 (easiest level) up to ten (hardest level), as player advances through the levels.

2.2 Grid layout

Pygame provides a sprite attribute of its location but as I have decided to use a window of over 500x500px of size, its pixel-accurate positioning would be an overkill and could make programming harder as well as require more processing power to (later mentioned) pathfinding algorithms. Due to this I have turned to the original PacMan solution (as read in Reference 1), the grid system. The board was initially made as a 64x64 grid, where each piece, namely a tile, was either a wall or a space a sprite may move on. First levels were designed this way and I found the level design a really hard task. Only later I have noticed that the grid of the original PacMan was barely $\frac{1}{4}^{\text{th}}$ of the size I have used, my board was just to big for a pleasant gameplay.

As I wanted the board to be square I have decided to move to 32x32 grid. Some testing later I have decided that this size not only nicely divides by 2 making perfectly symmetrical or even fractal levels possible to make but also is actually quite the one most similar in terms of number of tiles to the one the original PacMan had (for square boards).

The following section describes types of different tiles used throughout the game.

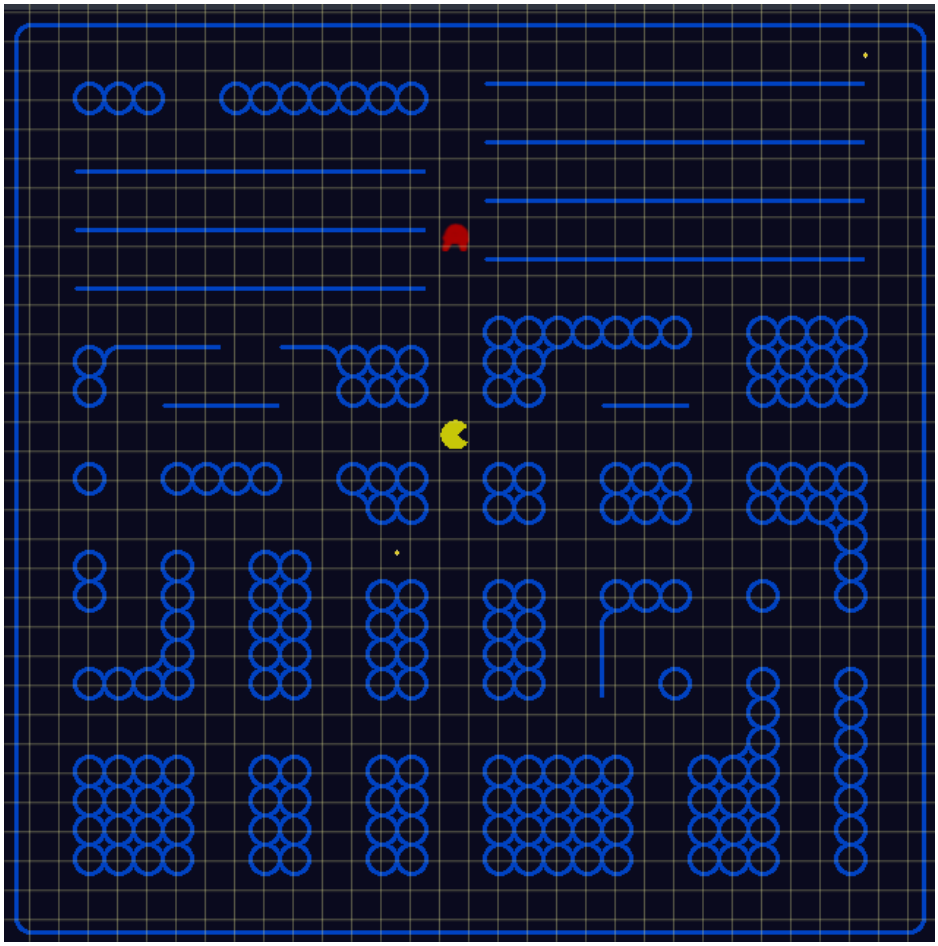

Figure 2.3: Example of a board with a grid applied (note that the grid was added just for demonstrative purposes and in not a part of the game.)

2.3 Types of tiles

2.3.1 Walls

2.3.2 Player

2.3.3 Ghosts

2.3.4 Eatables

Coins

Hearts

2.3.5 Empty tiles

2.4 Levels

2.4.1 Permanent storing

2.4.2 Interpreting

2.4.3 Storing while in game

2.4.4 Designing the levels

2.4.5 User-defined levels

2.4.6 Progressing to the next level

2.5 Gameplay scheme

2.5.1 Player movement

2.5.2 Ghost movement

2.5.3 Eating

2.5.4 Getting killed

2.5.5 Respawning after a loss of life

2.6 Types of message screens

2.6.1 Tutorial

2.6.2 Loss of life

2.6.3 Completing the level

2.6.4 Loosing the game

2.6.5 Completing the entire game

2.7 Testing

Part 3

References

1. http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior