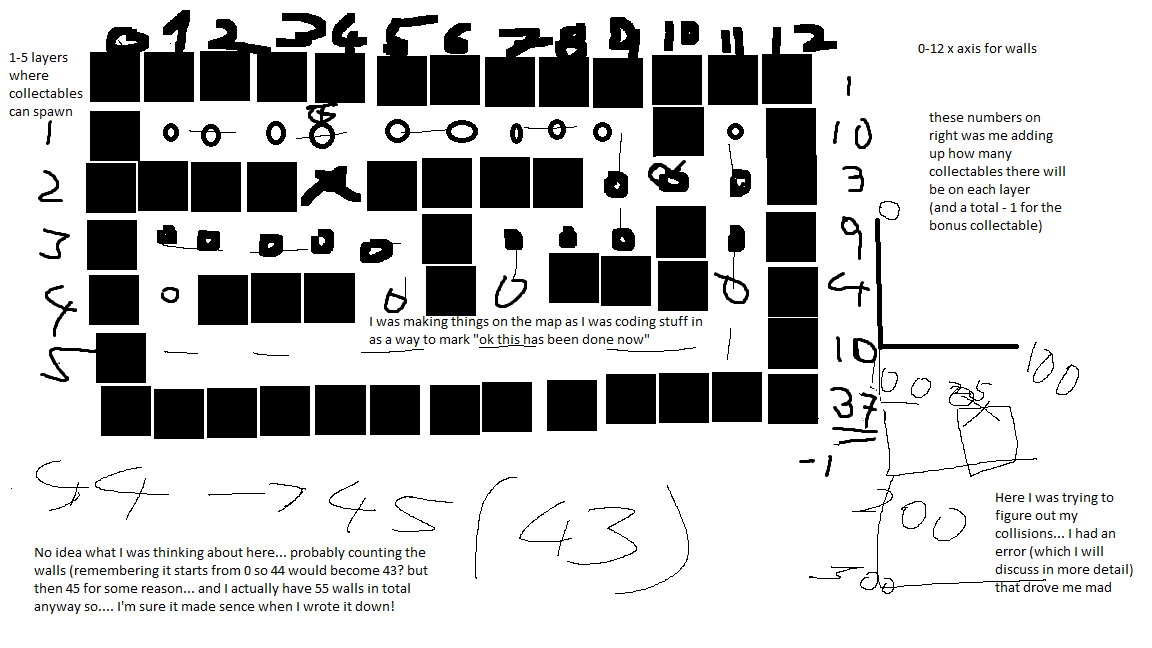
**Low Level Programming – Log Book**

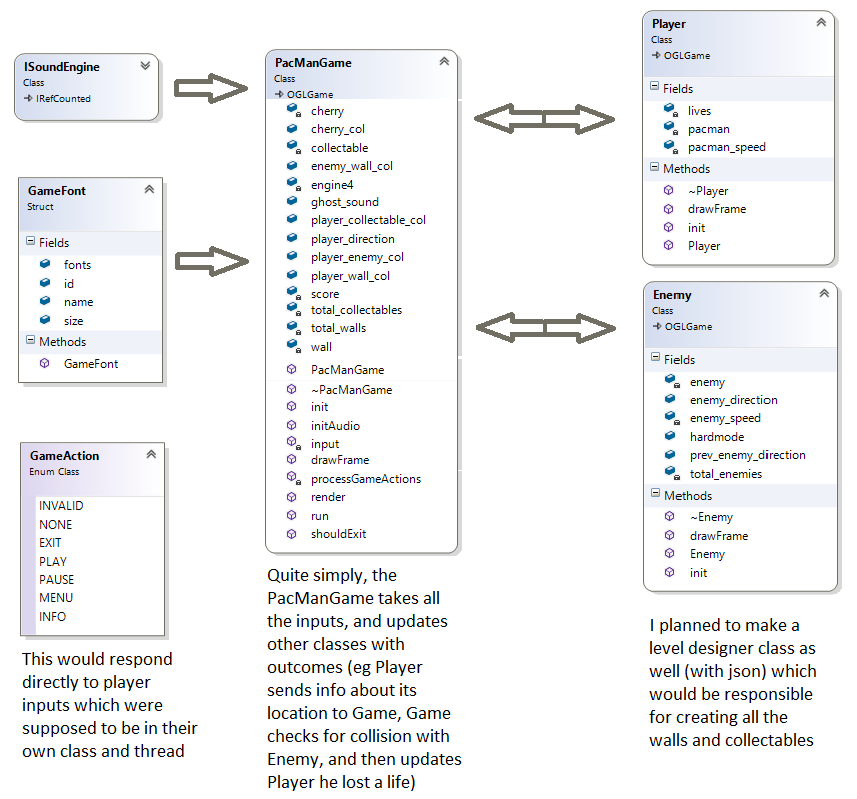
Design and planning

I have the unfortunate tendency of just "getting on with it" when it comes to programming. I rarely plan things thoroughly even though it often bites me later on... I like knowing what I'm doing but I only keep few mental notes in my head and I make the plan up as I go. This was also the case this time around hence I don't have much evidence of planning other than a simple UML diagram and some of my notes and drawings (if you can even call them that).

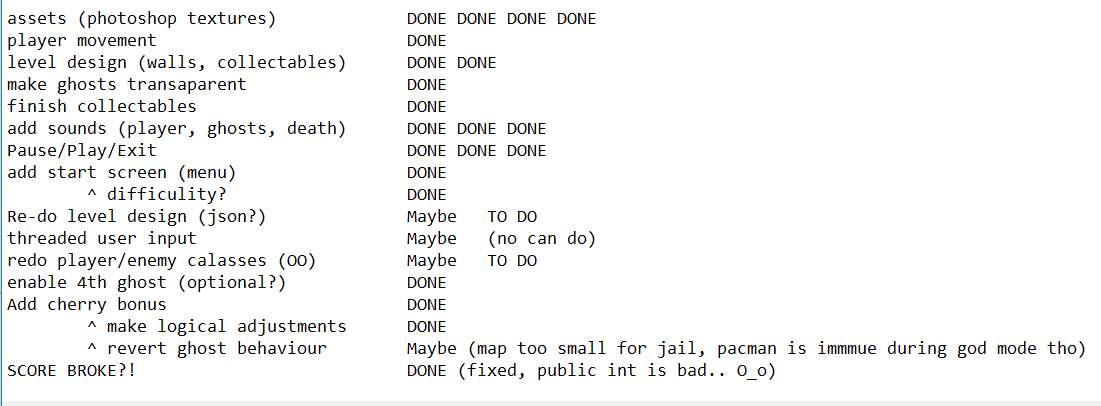
I started off with designing the level itself, and for the sake of simplicity I hardcoded the whole level (which was supposed to be only temporary at first) and below is a Paint file I used for some of my notes. (I added few text boxes here and there to try to explain what was going through my head while I was creating this masterpiece)



Next image shows the UML diagram roughly showing how the classes were planned out, how they would behave and what sort of parameters they will be passing through.



I also kept a notepad file open where I would write small comments of things to do, things that needed researching, bits I have completed and what else currently needs priority, I also made notes of important things like where the dll files were kept (explained in build instructions) and all references.



There were a lot of cases were I had spent hours and hours staring at my notes and at the code trying to figure out how to make something work, or why it works but only 90% of the time... When trying to figure out how collision would work I thought about how I will need to check for position of 2 objects as well as their size, because I needed the collision to trigger as soon as their borders touched, not just when they were exactly on top of each-other.

Using Size as well as X and Y co-ordinates of 2 objects I wrote this up:

if

(player-position.Y + player-size.Y) is bigger than object-position.Y

player position.Y is smaller than (object-position.Y + object-size.Y)

and

(player-position.X + player-size.X is bigger than object-position.X

Player-position.X is smaller than (object-position.X + object-size.X)

then collision is True

This worked well, when it worked, however I encountered a problem during implementation...

Implementation

Logical errors are the worst of all.

For example, assuming the following parameters:

pacman->position[P] = 600;

pacman->size[0] = 50;

enemy->position[E] = 300;

enemy->size[0] = 50;

Using the following if statement:

(I removed brackets so it is more readable and the asterix points to where I made the mistake):

if

pacman->position[P] + pacman->size[0] >= enemy->position[0]

& pacman->position[P] <= enemy->position[E] + enemy->position[E]\*

return true

Translating it into simple maths:

P=600=300 size=50

600+ 50 >= 300 --> True, 650 is bigger than 300

600<= 300 + 300 --> True, 600 is equal to 600

Collision == true

But the objects were not touching each-other, pacman was 200 pixels away from the ghost and with its size of 50 there was a 100 pixels gap between them.

That is when I noticed this typo (marked \*) the line adds enemy X position twice, instead of adding its position and size, correct code should look like this:

pacman->position[P] <= enemy->position[E] + enemy->>size[0]

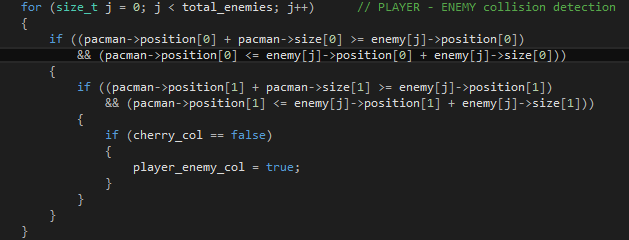
Therefore

600<= 300 + 50 --> False, 600 is bigger than 350

This would correctly return false

It can be annoying trying to pin point a logical error, compiler works fine, program behaves normally, it does exactly what it was designed to do... but when you make a mistake like this it simply does things you did not intend for it to do...

Overall, I am really happy with how the collision system turned out, after all it is an essential part of this game (or almost any game for that matter). Yes, I had a small hiccup with it, but I fixed it and it works just fine now. Following image represents the finished collision which later on I tweaked to also use for things like collectables. (Highlighted line is one that caused issues before I changed it)



There was a point where I was considering doing something else for collision detection but compared to other alternatives bounding boxes just seem so simple and exactly what I needed. I was going to expand on this though and make a more robust and a stand-alone collision manager that would handle all game collisions from within its own class but I did not get around to doing that.

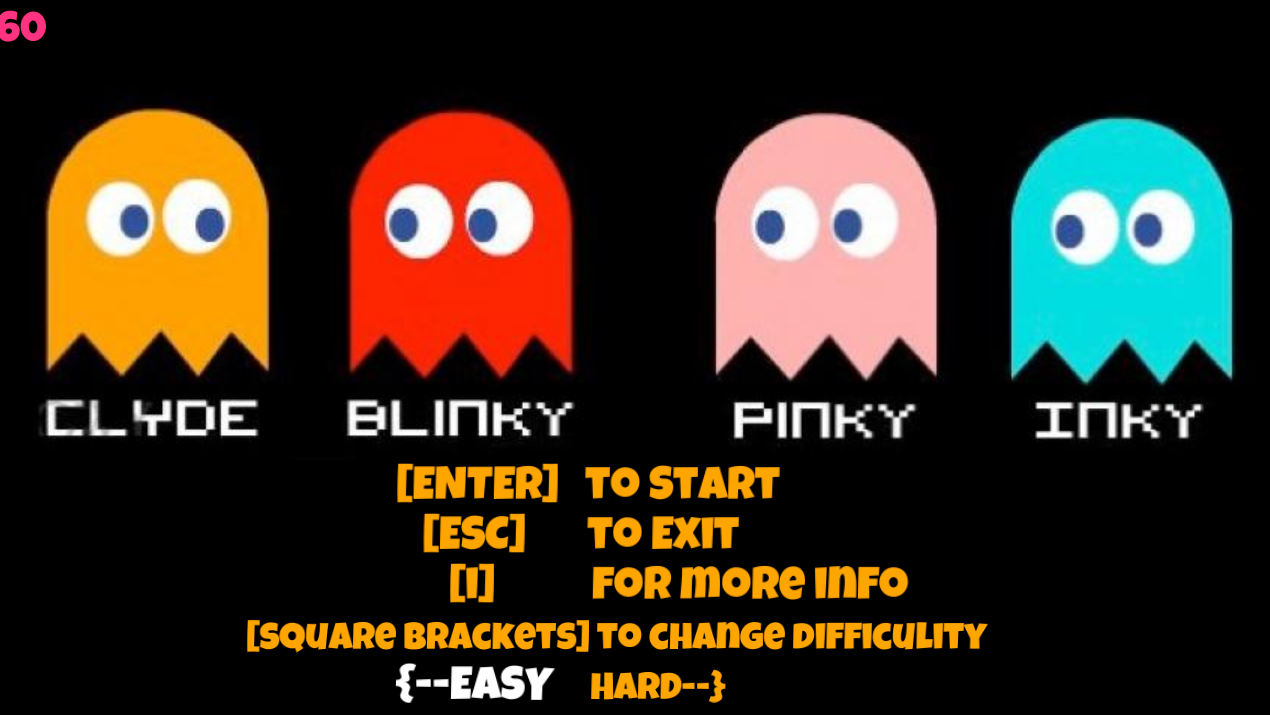
Similarly, the Enemy and Player class are not yet finished. The game is functional, it is in a finished state, there could (always) be more features etc., but, going back to what I mentioned previously regarding me not planning things properly, when I started coding in the level, player, enemies... I "temporarily" placed it all into game.cpp and by the end that file had over 1000 lines of code... debugging was getting more and more frustrating but I decided I want to finish the game as it is now before rewriting it all into its own classes because it would take me too much time and I prioritised finishing the game. As it stands now, there is a Player class and an Enemy class, but they are currently unfinished and disabled because I know for a fact that the game works as is and even though it is not very Object-Oriented-like I would rather hand in a working game than broken attempt at rewriting the code. The json file loader also did not make its way into the project even though I have previously done it for a different project, but I wasted too much time hardcoding the level into the project...

Reflection

Overall, I am pleased with how the game turned out. I received a positive feedback from friends that got to play the finished version. The visuals and sounds are a nice nostalgic trip. Ignoring, for a moment, how the code could be done better (with separate classes, and a threaded user input) I think this project is a fair representation of what a pacman game is. I think it's fun to play, and if it's too easy there is always a bit harder mode. The program does not crash, however there is a rare bug where player gets stuck in a wall (to solve it – player needs to press a key that corresponds to the opposite direction that he wishes to go, that will push him back out of the wall), the game pauses and un-pauses correctly, the UI and menu turned out well, the score (even though maybe a bit hacky) displays correctly and that cherry sprite looks amazing, if I may say so myself.

The things that are not so good – the previously mentioned fact that I need to split the code and move it into its own classes (already started doing that but it will not be finished on time, however I will still include it as a commented-out section). When pacman eats the cherry and gains super power, ghosts could perhaps start flashing as in the original game (could be done same way as the pacman sprite changes while he moves), and I would love to add an actual jail for ghosts to sit in when they get eaten by boosted pacman but the level is too small for that. That brings me onto the level itself – with json file loader (or even a simple text file) I could make much nicer, bigger and just more levels.

Knowing what I do now, I would definitely spend some quality time planning everything to avoid a huge game.cpp file I'm stuck with now, and the lack of expandability. Given more time I could make this code more OO-like, and I would definitely have another go at threading (which I will research more regardless, as I feel it's quite important to have a grasp on how to code it)



References:

All textures were made by me in Photoshop 2015

(excluding the menu image was provided together with the code-base)

All sounds were taken from freesound.org. Direct links:

<https://freesound.org/people/copyc4t/sounds/231769/>

<https://freesound.org/people/NoiseCollector/sounds/55654/>

<https://freesound.org/people/RutgerMuller/sounds/190501/> (edited in WavePad by me)

<https://freesound.org/people/plasterbrain/sounds/266163/>

Build / Play instruction:

There are 3 .dll files in PacMan\Projects\Debug folder, these need to be there or the sound will refuse to work, hence I am not uploading a "clean build"

In case those files get deleted, they are also in PacMan\Libs\irrKlang\bin

The play instructions will be displayed in-game (Press i while in menu)