DEVELOPMENT REPORT FOR PACMAN DUNGEONEER

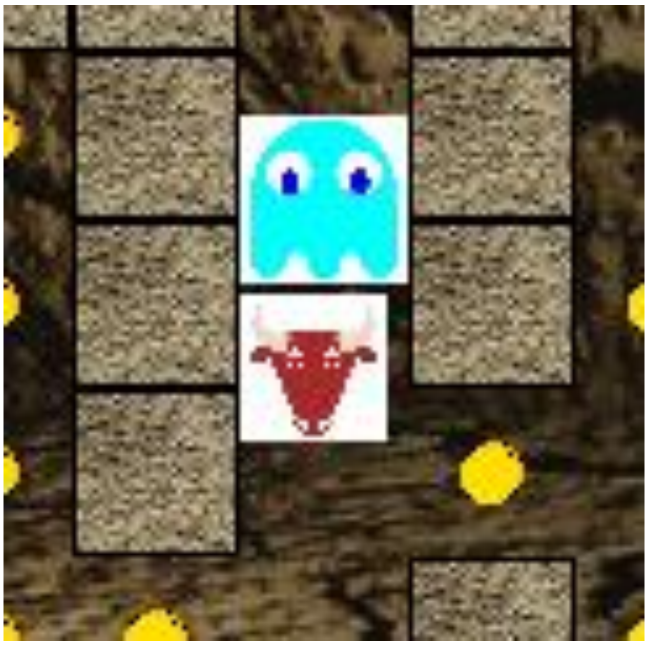
COMPSYS302 Java Project

APRIL 24, 2018

JR DEV
Group 3

Game Development Requirements

Some design features that improve the functionality of the system were: having multiple input being read and compared such that movement of the character is not hindered at the junctions. To help with the junctions, character boundary boxes and icon size where 28 by 28 while all other elements (asides from the small pellets) were 32 by 32. This is shown below.



Issues that came up during game development

One issue that cropped up during development was that we initially developed the game using swing as our display interface. This caused most of the work by Raymond to be scrapped as we started over in JavaFX. This occurred in the middle of the second week of the break. We found that transition from JavaFX from swing was quite difficult causing us to lose time due to inexperience. Also, another issue is that most online tutorials for game development were using swing as the basis of the game display. This made it so that there were very few resources we could use in understanding and developing the game using JavaFX. We also found that Oracle was not very helpful in understanding code and finding out what functions we could use to implement any of the features.

What we could have done to remedy this was to try to figure out in the beginning what were the libraries we should be using. Due to our own inexperience in using Java we neglected to think about the possibility that certain elements in Java were outdated.

In the end we managed to pull through using JavaFX with the help of online tutorials such as CarlFX[2] and Almas Baimagambetov[1] on YouTube. These tutorials formed the basic building blocks that allowed us to build our own game.

The other issue that arises from this project was that one of our team members become out of action due to medical issues.

We resolved this contacting the Teaching Assistant and course coordinator and a compromise was arranged.

The suitability of java in game development.

Java itself is a great tool to make a game in as it offers the ability to easily apply object oriented design. This is particularly useful in game design as it allows low coupling and high cohesion useful for game development. About our project requirements creating the game Pacman was relatively simple in Java compared to our experience in C++. Java offers many useful features including the JavaFX library for display. One such is the animation timer method which forms the basis of our game loop, and allows for multi-threading, which would be more useful in more complex and CPU-intensive programs.

It's a fine language for cross platform development suitable for both web-based applications and low-detail graphics games. I would suggest for someone interested in game development to use java as a starting point for making a 2D game. However, 3D games are still possible using Java, one such example is Minecraft developed by Notch. Java is a great way for someone who is interested in game development as it allows it to be easy to change the OS system you are working in and so it allows compatibility with many platforms without any extra effort to make them work.

How were coupling and cohesion dealt with

Coupling and Cohesion were very poor in our project. The lack of experience in working with object-oriented design using multiple classes and methods aided in this. Jonathan particularly had issues accessing private variables and passing them to other class classes. This lead him to develop the game mostly in one file which made it difficult for Raymond to read and understand the code. Attempts were made to try to bring the project more inline with low cohesion and coupling coding styles. That proved to be difficult due to the large amount of code we had to sort through. MVC style of coding would have helped in dealing with the cohesion and coupling. We did initially work with MVC style of coding but we had developed the project using swing. Moving from swing to JavaFX we neglected to make use of MVC style of coding.

Discussion of game development methodology

Iterative and incremental development was key to not being overwhelmed with developing the game. We started with a game loop using an animation timer as the base for updating our game view and game model. Next, we designed a way to get the map to be read and translated into our game window. Then we developed the player and player movement system. Next was collision detection for the wall and then the pellet creation and collision. Once that was developed, score then countdown timer was developed. Ai was done next then lastly sound [5] was added.

Discussion on game design experience.

The experience in working to make a game was beneficial in working with a complex and interconnected system. This game design helped us experience what it is like to make a fully functional and complex program. Most of our experience was with working with simple methods that we had to develop to the specifications. This project however let us experience what development of programs from the ground up. This bears more resemblance to real world development.

We had a lot of difficulty in making the game from scratch and finding help online, as most tutorial focus on the older swing. Most JavaFX tutorial are simple or are irrelevant to game design.

Suggested Improvements

Design the game in a more MVC style for easier reading and ease of changeability of code. This would be particularly useful if we decided to implement the extra features into the game, including map making to make it easier to create interesting maps, and special abilities of the AI.

Have more classes to deal with handling of different functions and task. We currently have most of our functionality within 1 file.

Time management

We started quite late with developing the game. We started during the second week of the break, leaving not much time to effectively follow MVC or object-oriented methods.

We should have taken the time before the break to develop our initial code base instead of leaving to start in the last 2 weeks. One lesson we can take from this development cycle is that project development is a much more time-consuming process than we think

Communication

Communication could have been better. A lot of the time we worked on the same bit of code at the same time. Game loop, movement, collision, display. This improved as we restarted our project when we shifted from java swing to JavaFX, but was still a long way from perfect.

More detail timeline with milestone. We ended up splitting the workload by working on different classes. What we should have done is added a to do list for a more incremental design process. This would have made it easier to work with someone else’s code or pick up from where the other person left off, and avoid working on the same features twice.

Using Git

Using git this year was new to us, meaning it was not without its issues. The concept and the reasoning behind why we use git was relatively straight forward. Issues did come up as we had trouble pulling and pushing on some occasions. This meant that we use Google drive more often then we should have. Leading to multiple files and obsolete code being present.

Commenting

We had very minimal commenting in our code which made working on code was difficult if you didn't write it yourself. This is especially hard for us as large sections of code were added periodically without any commenting. This took time to understand the code which could be used for development.

We should comment our code more often. In the future we should make a habit of commenting as we code or commenting before a successful commit, in more detail. Also, include block comments where needed at the beginning of all the classes.

Concluding Thoughts and Conclusion

Overall, we found this project engaging but difficult. We initially aimed high with the extra features that we wanted to develop but ended up not reaching our goal due to time and lack of experience. We now have much more appreciation towards those who develop games and other programs from scratch after finishing this project. In our eyes, regardless of the features we failed to implement, the game and project as a whole was a huge success, especially considering our previous coding experience.

References:
(references for swing have been
1. https://www.youtube.com/watch?v=aOcow70vqb4
2. https://github.com/AlmasB/FXTutorials/blob/master/src/com/almasb/tutorial14/Main.java
3. https://carlfx.wordpress.com/2012/04/09/javafx-2-gametutorial-part-2/
4. https://docs.oracle.com/
5. https://freesound.org/