Pacman: Space Edition

ABSTRACT

The aim of the project is to develop a game like the 1980's Pacman developed by Namco. For our project, we are developing a space themed game based on Pacman called Pacman: Space Edition. This technical report outlines our client's requirements, features, top-level view of our game, significant issues we encountered, suitability of tools, object-oriented design, software development methodology, and future improvements of the game. Attached in the appendices are diagrams and images that outline the software architecture, folder structure, and images of the game.

MEETING REQUIREMENTS

Determining a project's success is dependent on achieving client satisfaction. The client is key in defining and achieving project success. We must gather the client's requirements and meet their requirements to achieve client satisfaction. The client had 11 minimum requirements, these requirements didn't define the scope of the project. This allowed us creative freedom to create a unique game for our client.

Our game has met the minimum requirements, as we have a GUI menu that allows the player to select a game mode. We have implemented a countdown timer, where the characters only move once the countdown finishes. The window size for the menu is 1024x768, while the in-game window size is 1440x900. Our protagonist consumes pellets and cannot go through walls, the protagonist can also wrap around the map. The protagonist loses lives when it collides with an alien, which causes the characters to reset position. Once the protagonist collects all the pellets the game moves to the next level. The max time limit for our game is two minutes. To pause/resume you and exit you can press 'p' and 'esc' respectively. We have also implemented sound effects on actions and collisions. However, to make our game unique we have also implemented a few extra features.

FEATURES

Refer to **Appendix D** for screenshots of the game.

GUI and Control

Our game has been designed to be easy to control and understand for the user.
The visual representation of the components of the screen help emphasis which option is current available. In our main menu, we use a dark background that contrasts with our buttons to make them clear. The currently selected button will be highlighted with a yellow glow to clearly show where the user is doing. Sound is used as feedback to the user to they can aurally hear when they change options or press a button. The game modes such as multiplayer also show which controls are associated with each character whilst blurring the previous menu to minimise ambiguity for the players. In game, any not being focused on will be blurred as well. When pausing, exit, and in idle state before the game

starts, the level display will be blurred out whilst a yellow glow around the buttons of interest accompanied by sound will give feedback to the player controlling these interfaces.

AI

The AI used for our aliens to track spaceman are made using the A* algorithm. A* algorithm is similar to Dijkstra where the shortest path is determined through calculating path distances between nodes. We can use this algorithm to calculate the optimal path to wherever the spaceman is headed and move the aliens along that path. Each alien has its own personality. The red alien will target 4 tiles ahead of where Pacman is head whilst the pink ghost target 8 spaces. The blue ghost will target Pacman until it gets within 8 tiles of him. The yellow will just target Pac-man's current location. This keeps the game difficult enough

Powerups

Powers are a way to increase or decrease the difficulty of the game by creating interesting new play styles due to the range of options available to them. The current powerup line up in this patch include invincibility star, life-up, cherry, energy shield and mystical stopwatch.

The invincibility star and cherry make their re-appearance in this edition of the game. The invincibility star is the re-skinned version of the 'magic pellet' from the original Pac-man series where consuming it will allow the player to momentarily scare and overpower the aliens. This gives the player a chance to recover when cornered.

The cherry serves as a bonus for the players allowing them to gain a large sum of points by collecting them from dangerous locations. The cherry promotes a riskier play style in order to gain more points which will make it more fun and competitive for the players trying to reach the leader boards.

The life-up powerup gives the player another life as the name suggests. The purpose of this powerup is to also promote a riskier playstyle as players have more lives to spare, they can incorporate tactics such as dying at the other side of the map, so they can respawn on the unvisited side to save time.

The shield acts similar to the invincibility star where it will allow the player to overpower an enemy while it is active, but it only works once per pickup. The energy shield can also overcharge when you pick up another shield whilst active causing both pickups to disable. This powerup was intended to make the game less difficult since some of the new maps are unforgiving in certain area. However, the overcharge mechanic prevents players from stocking up shields and pushes them to break the shield in order to maximise the usefulness of other shields in the vicinity.

The mystical stopwatch is one of the more powerful powerups when used correctly. It stuns the aliens for a few seconds allowing the player to escape and reposition but any powerups effecting the ghosts (star and shield) will break them from this stun. There are a few interesting tactics we've seen from players testing our game. One example was a player herding the aliens together then freezing them, so the AI would not be spread out for a while. Unfortunately, our AI is smart enough to diverge when this tactic is used.

Story mode

Our story features interesting characters and interactions that you can immerse yourself with when playing this game. The completion of each level in story mode plays out a scenario for the player to put some perspective behind theme of the game. It gives backstory of why the player is collecting 'NRG' in

the game whilst avoiding aliens. It also includes several pop culture Easter eggs that appeal to a wide range of the target audiences. It helps show the player that we are also gamers ourselves and put a lot thought and effort into this game for them to enjoy. During the story stages, the player can also press backspace to re-read dialogue if they missed it.

Endless mode
Endless mode is a twist on the classic mode where instead of collecting all the pellets before you run out of time or lives, the pellets now respawn and there is no time limit. This game is aimed at higher score achiever as you could theoretically play as long as you can. The powerups were chosen to be respawn because we want the player to think about the optimal usage of the powerups to demonstrate their mastery of them.

Warp mode
Warp mode is our featured game modes for this edition of Pac-Man. The game mode is the same as classic Pac-Man but warping through the tunnels now changes the map you play on. All the maps are linked together so you retain your lives, score, time and consumables when switching maps. This gives the player more options to work with in each map. For example, when the current map has run out of safe pellets and powerups and is hard to transverse then the player can warp to another map connected to the current one. However, if they decide to warp back, the aliens will be waiting for them wherever they were last seen. This causes an interesting mechanic where you need to loop all the way around the other maps until they reach the original map in order to return safely to the other side where the aliens aren't waiting.

Classic mode
Classic mode is the classic 80's Pac-man game we tried to preserve mechanically. The graphics are updated to feel more modern but the gameplay stays the same. We choose to omit the most of the newer mechanics in this game mode so reminiscing players can indulge in this nostalgic trip. It was mentioned that the client used to play games in his childhood so we tried to cater to him as well as younger people.

Leader board
A leader board was added to keep the game social and competitive for the players. This game will likely be played locally so being able to keep track of high scores so you can beat friends and family create a social environment where you are not playing the game alone since you can share your achievements with others.

SIGNIFICANT ISSUES

Version Control
Before this project our team did not have any experience using any source control technology and were slightly tentative towards using git. When we started the project, we initially had our own local copy of the project and would send each other the update source code files. But this led to problems with having to fix the same problems again, losing some source code, and compilation errors due to merge files manually. However, git solved these problems for us, because git serves as a central repository, that

always has that latest version. Git merges files automatically, which prevented errors from manually merging files. Also, git allows for the use of branches for experimentation of new features, thus preventing breaking the solution for the master branch.

Limited Time

A major issue was the limited time we had to create a minimal viable product (MVP) for our client. A significant amount of time was spent debugging and fixing issues within our game. This is because a lot of bugs were caused by human error and not due to logic flaws. For example, this included assigning a 'y' variable to a 'x' variable due to copy and pasting the previous lines of code, while forgetting to edit the variable name in the new lines of code. These bugs were hard to notice within source code.

Due to a large amount of time being unexpectedly spent on debugging, we had a multitude of features that could not be delivered by the deadline such as random map generation, spaceman vs spaceman game mode, character skin and custom theme selection. Therefore, we had to rescope our project and decide to not include some of the features in our MVP. We also decided to pivot from our plan by replacing random mode with another mode that could delivered on time.

SUITABILITY OF TOOLS

Java

When developing a game, an object-oriented programming language such as java aligns with our goals. An object-oriented language key features are classes, instances, data abstraction, data encapsulation, inheritance and polymorphism. Classes provide a structural interface to define objects so that multiples of the objects with the same attributes and methods can be created from the same class definition. The definitions of classes also provide encapsulation by keeping the implementation and data in the classes so that the outside world only knows and uses what it needs to. Inheritance allows us to break down a class so that it can be re-used elsewhere. These OOP concepts are useful in game design. An example of OOP usefulness in our game is the power ups and pellets. Powers ups and pellets can be defined by a class where they can be used to generate multiple instances of it. These objects hide their implementation as we would only need to know what they look like in game and where they are in our map.

BitBucket

In a project of this scale, the use of git in the form of bitbucket allows us to maintain a robust framework in order to work efficiently. It is easy to put into practice even with little experience in version control systems. Working on a distributed version control system is beneficial since we can work locally and quickly when not pulling or pushing changes to the remote repository.

One of the primary reasons to use git is because it is an effective version control system. This means that we can track all of our file and its changes over the course of this project easily. Projects of this scale are typically iterative thus the ability to track these changes and refer back to them is valuable. This is especially important in the case that we want to rescope as git allows us to find the version that we can want rebase to.

In team-based project such as this one, the use of a version control system allows us to work on the project separately whilst still maintaining structural compatibility with each other. Bitbucket is a distributed version control system which means that the main repository can be cloned to create a local copy so we can work remotely on the copy. Developers can work on different features at the same time as their colleagues by working on their own local copy. The promotes streamlined collaboration where features can be built on top of the repository without having to worry about conflicts. The ability to branch out from this clone so that we can test experimental changes without having to affect the working copy. In programming, it is common for bugs to occur from the smallest of changes which can affect the functionality of the whole project. Branching out and version control helps mitigate and control this issue by isolating the untested changes from the working copy.

OBJECT-ORIENTED DESIGN AND ADDRESSING COHESION AND COUPLING

Object-oriented design (OOD) allows us to spilt modules into self-contained objects with their own logic and data. Object-oriented programming (OOP) lets us create classes which represents a collection of objects having the same properties and behaviour.

But to ensure the use of OOP results in reusable code, we try to minimise coupling and maximise cohesion. To minimise coupling we try to reduce the inter-dependability among classes so that classes don't depend on each other so much, that you cannot change one class without modifying the other classes. To maximise cohesion we try increase the intra-dependability of a class by grouping variables and functions into a self-contained class.

To try and maximise cohesion and minimise coupling, we utilise the Model View Controller design pattern (MVC) refer to **Appendix C**. The idea behind MVC is make a clear separation between domain objects that model the data of the environment, and the objects that present the GUI elements visible on screen. We communicate between the model and view objects using controller objects to keep the model and view separate. Therefore, using MVC design pattern helped us to maximise cohesion and minimise coupling.

ITERATIVE AND INCREMENTAL DEVELOPMENT

Iterative software development is a method of breaking down the development of a large project into smaller segments of development, where software is designed, developed, and tested in frequent cycles. Each iterative cycle of development, will allow developers to design, develop, and test additional features until the product is fully functional and ready to be deployed.

Normally iterative software development methodology is used with incremental development, where a more long-term software development project is split into smaller chunks that build upon each other.

In contrast to iterative and incremental software development, a traditional waterfall methodology is a more linear approach to software development. In the waterfall methodology there are a sequence of stages that are followed, each stage normally finishes before we start on the next stage. Therefore, in waterfall methodology the development is not broken down into smaller segments, and therefore we can only start the testing stage once all the coding development has finished. This is a major drawback

to the waterfall methodology, because once development is finished and the product does not satisfy clients, then it will be costly and difficult to make changes.

Therefore, for our project we decided to develop our game using an iterative and incremental approach to software development. This allowed us to develop small parts and actively test our game iteratively, which made it easier to make modifications to our product.

TOP-LEVEL VIEW OF SYSTEM

Refer to **Appendix A** for the class diagram

Our game utilises the MVC design pattern, therefore our classes are separated into model, view, and controller (refer to **Appendix C**). We have two controller classes called InterfaceController and LevelController which handle the menu and the actual game respectively. We have four Model classes called Person for our leader board, Map which is our map data model, Level which handles the map data and the game variables. We also have a multitude of view classes that handle the GUI, characters, consumables, and our story slides. Our system starts (refer to **Appendix B**) at the MainApp class, where the main method is located. Which then initialises the InterfaceController, then the InterfaceController creates the GUI and initialises the LevelController. The LevelController initialises LevelVisuals, Level, Map and generates the actual level on the screen by calling generateMap(), then generateMap() function refers to the Level, Map, and character classes. Once the level ends, the LevelController calls the resetStartState() function which then calls the showHome() function in InterfaceController to return back to the main menu.

FUTURE IMPROVEMENTS

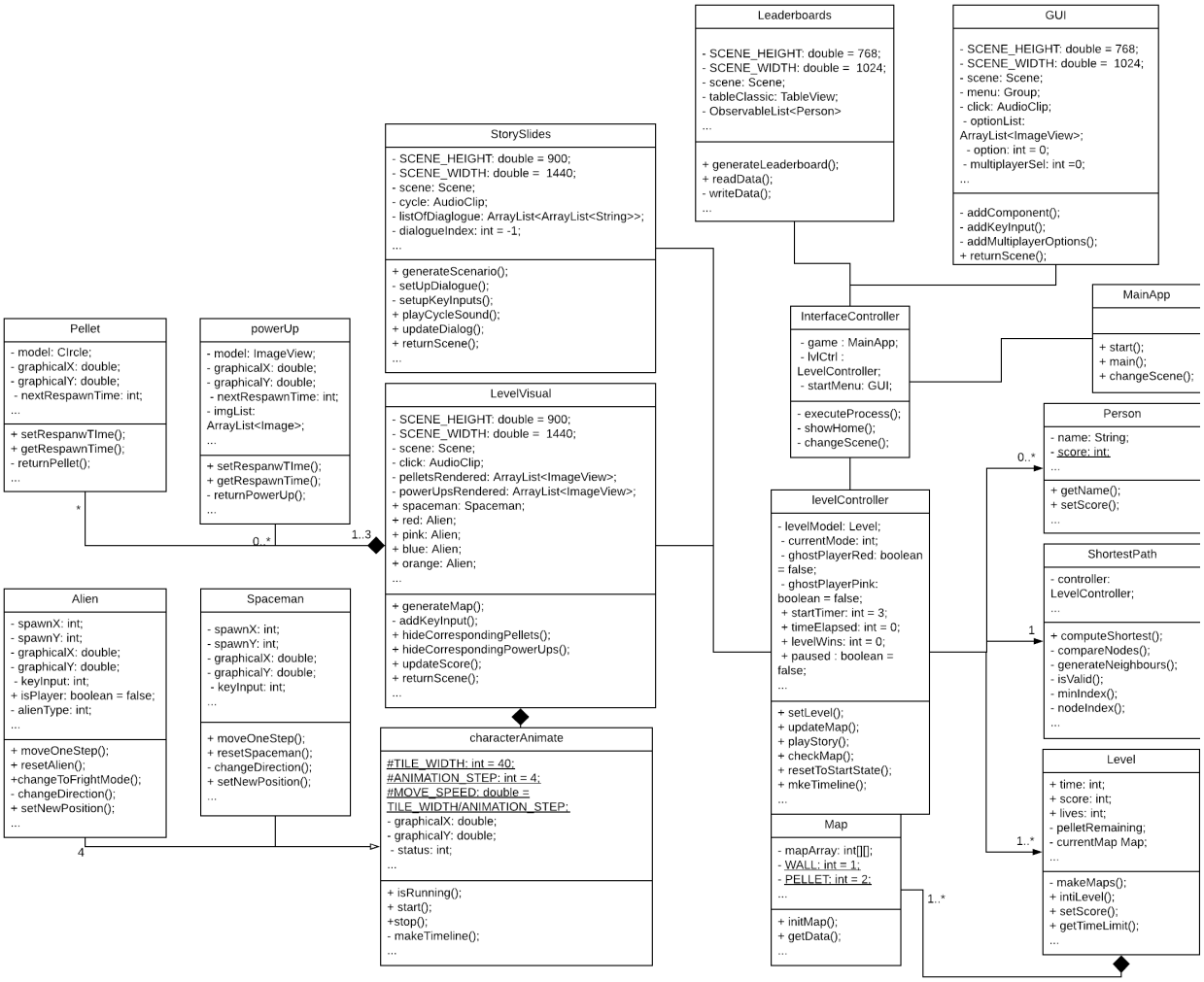
We were not able to complete all the features we planned by the deadline. Therefore, additional features and improvements can be implemented in the future to further improve our product.

The first step would be to improve the alien AI by including a scatter mode, so the user can have a short break from being chased. Secondly, we could add the ability to select the spaceman skin so the user chooses how the protagonist should appear. We could also add the ability to select from a few themes so the user can modify how the game's appearance. The game can be further expanded by creating new game modes such as Spaceman vs Spaceman mode where there are two user controlled protagonists and the one with the highest score wins, and Random Map mode where the game would generate maps randomly using pre-set Tetris style blocks and utilising our Shortest Path class to check if the map is valid. Further improvements can be made to the gameplay, where we could add more unique powerups such as a laser powerup that consumes any aliens that is in the same axis as the direction of the protagonist, we also make the wall objects with images instead of just using the Rectangle class.

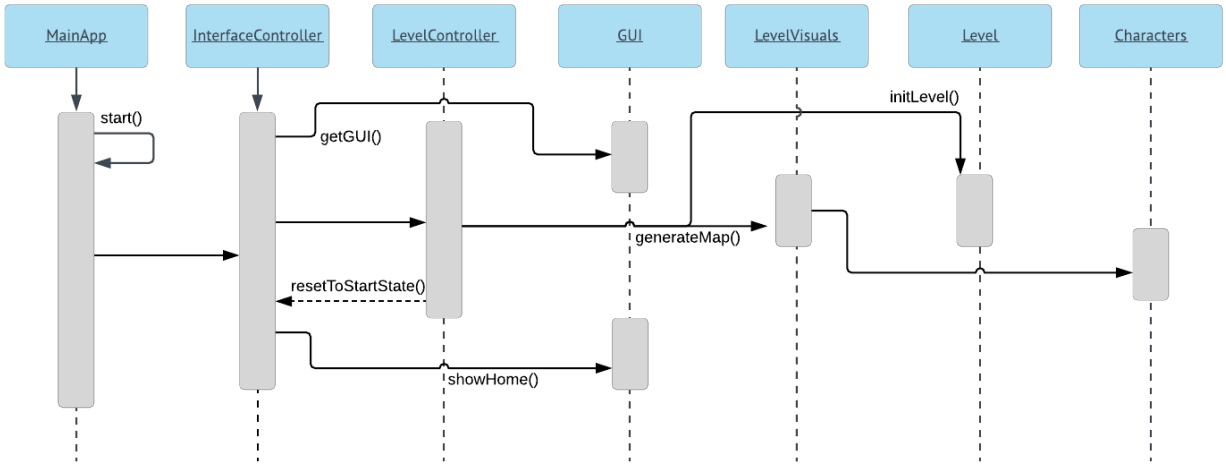
These additional features would greatly improve the gameplay experience and give the user more variety in game mode selection. Thus, the user is less likely to get bored and would enjoy the game experience much more.

APPENDIX

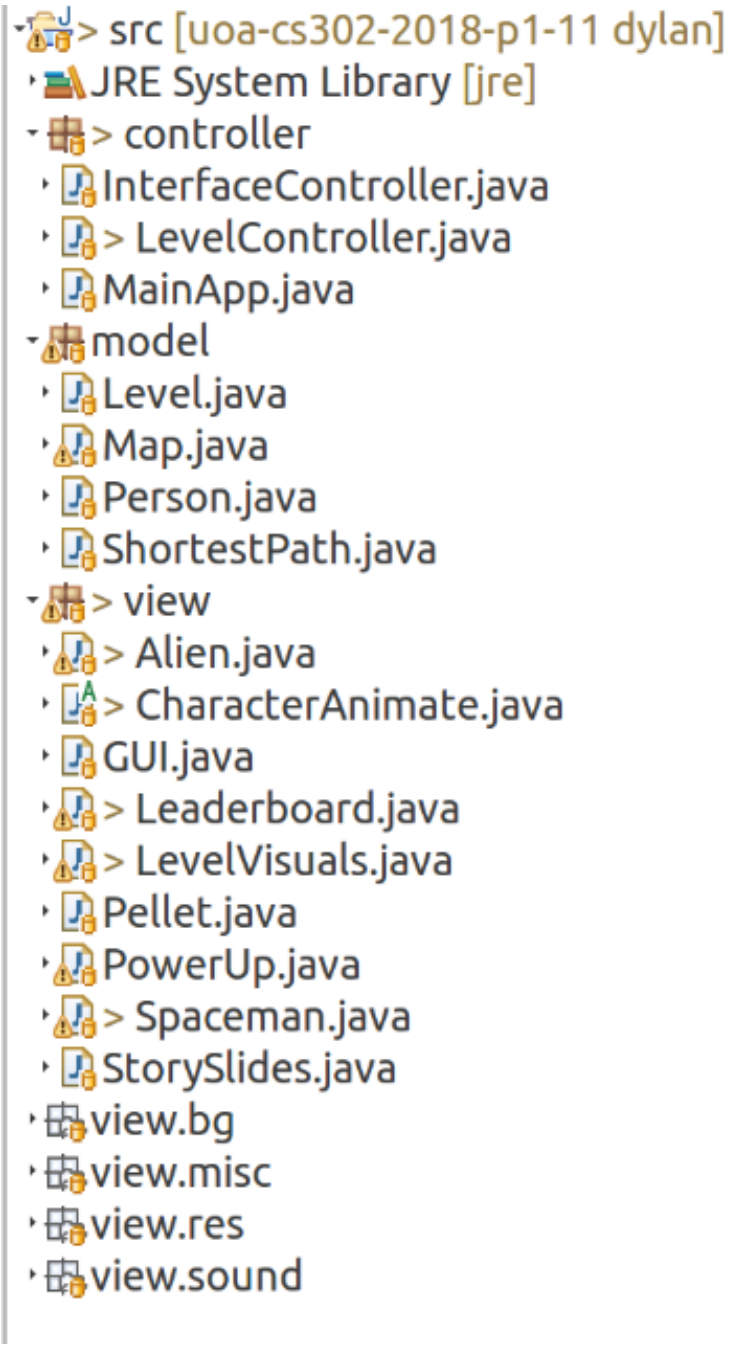
APPENDIX A: CLASS DIAGRAM



APPENDIX B: SEQUENCE DIAGRAM



APPENDIX C: MVC FOLDER STRUCTURE



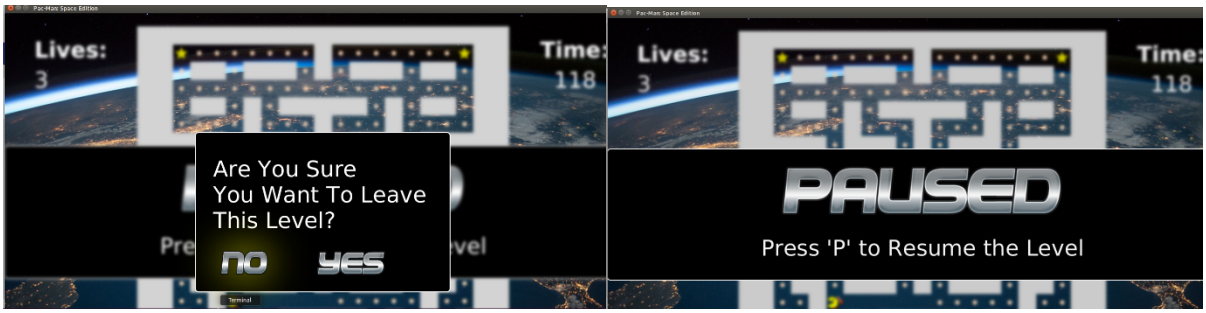
APPENDIX D: SCREENSHOTS OF GAME

D1: GUI; Demonstrating controlling the window

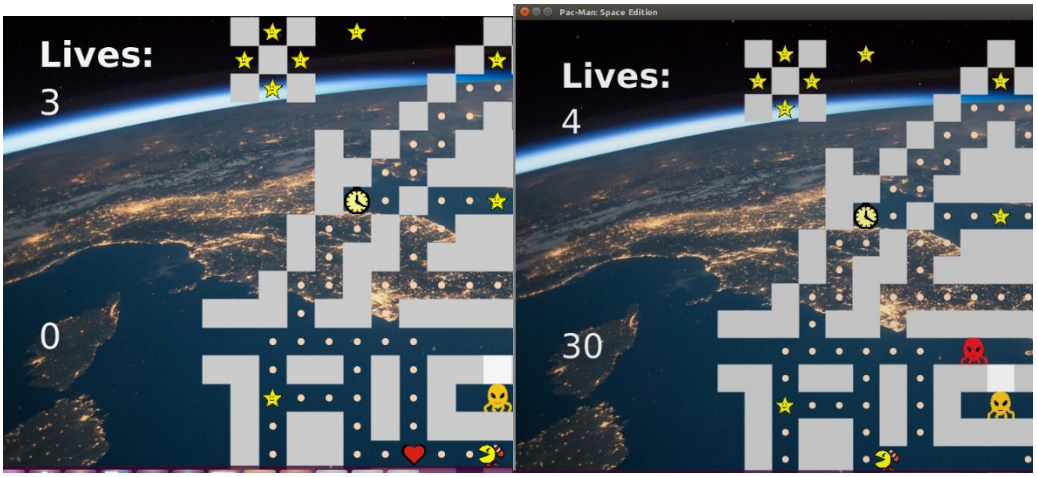
D2: Multiplayer Menu Interface

D3: Idle State

D4: Exit and Pause Menu

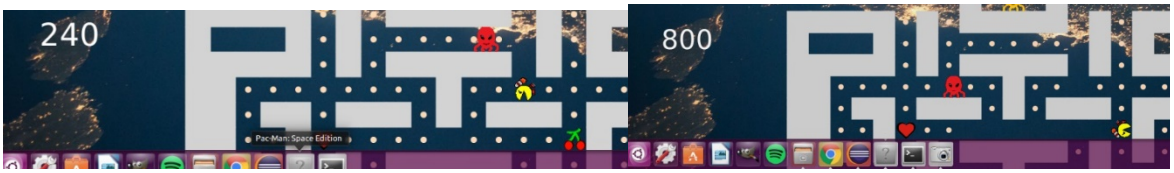


D6: Life-Up Powerup



D7: Shield Powerup

D8: Cherry Powerup



D9: Warp Mode; Switch maps through left tunnel