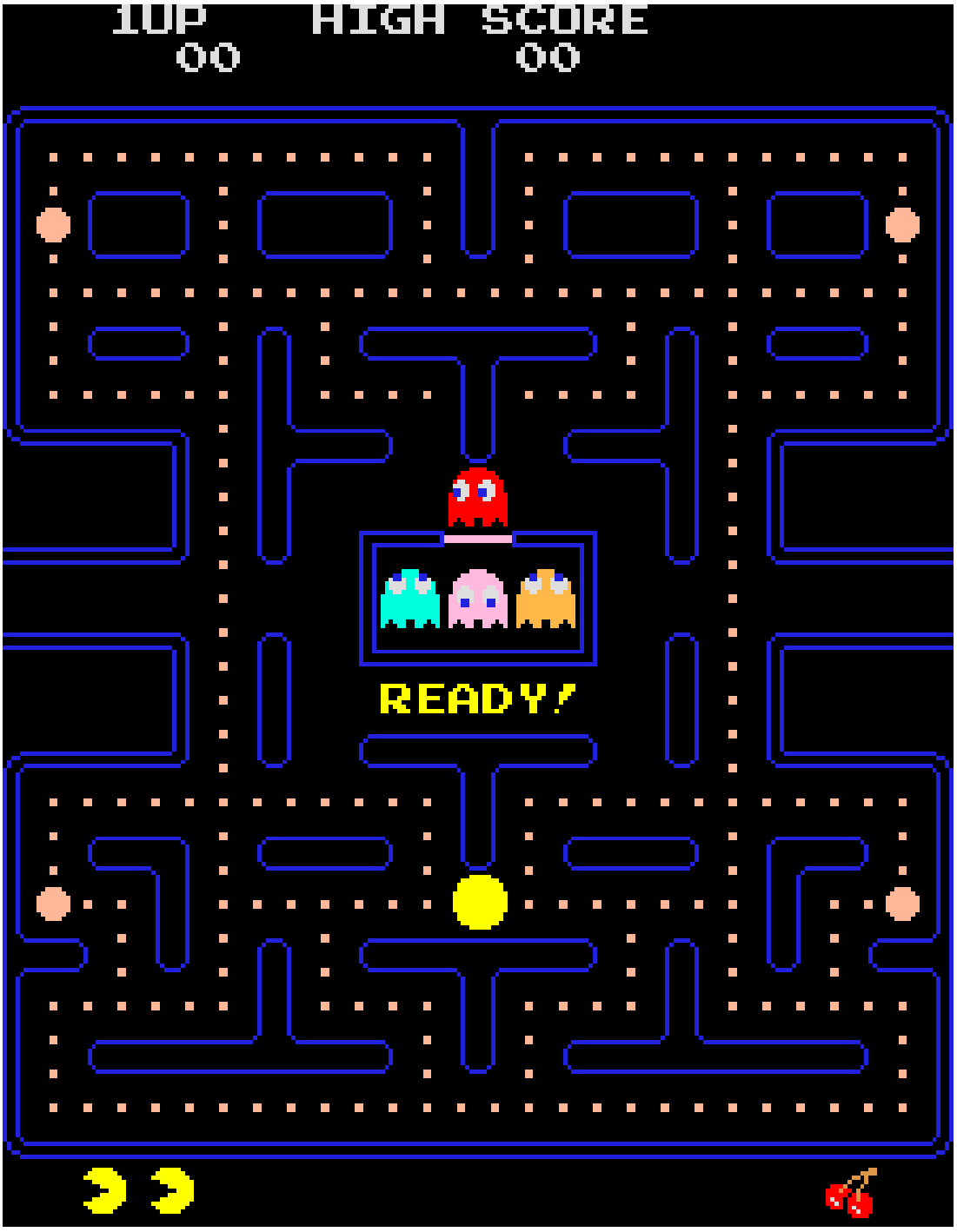
Beijing-Dublin International College

COMP2005J - Object Oriented Programming
Group Assignment: PAC-MAN



Assignment Details

Group size: Between 2 and 4
Due date: 13^{th} of December 2019 (No Exceptions)
Language: Solution must be completed in Java

Game Description

PAC-MAN is a classic arcade game, where a pac-man moves through a maze consuming the dots and avoiding the coloured ghosts. The player has a limited number of lives and loses one whenever a ghost catches them. The player gets points for eating the dots, power pellets (these give special powers to pac-man) and ghosts. The player can get another life every time their score goes over a multiple of 10,000.

Movement

Pac-man will always move in the direction that he is pointed. He cannot speed up or slow down and when the game starts, he will begin moving. In the maze pac-man can only turn a direction when the way is open, he cannot turn towards a wall.

The only buttons used in this game are the directional arrows.

Gameplay

In the game there is the following:

- A maze that pac man must move through
- Dots that pac-man must eat
- Four power pellets, that make pacman invincible for a short amount of time during which he can eat the ghosts (they come back to life a short time later)
- Four Ghosts that come out and chase after pac-man
- Fruit that randomly appears for a short amount of time and can be eaten for extra points

Points

Pacman gets points for eating each of the following

- Dots - 10 points each
- Power pellets - 50 points each
- Fruit - 500 points each
- Ghosts - 200 points for the first, 400 points for the second, 800 points for the third and 1600 points for the fourth (count is reset each time you eat a power pellet)

Levels

A level is complete when all of the dots and power pellets are eaten. When a level is complete, it is reset and the player starts again but the difficulty increases a little each time.

With each increasing level, the ghosts should increase in speed and usually get a bit better at catching the player.

Our version of the game should allow the layout of the level to be loaded from a text file. The different parts of the maze should be shown using the following characters

% - This character means that this part is a wall

P - This character means that this part is where pac-man spawns at the beginning of the level or when the player dies

. - This character represents that there is a dot in this location

***** - This character represents that there is a power pellet at this location

F - This character represents that fruit will occasionally appear here

G - This character represents that the ghosts spawn here at the beginning and after they die (there will be one for each ghost)

- - This character represents an empty square

Each maze design will be stored as a single text file with each individual character separated by spaces. The text file is laid out like a 2d grid and each line is the next part of the maze. An example of the format is here:

```
% % % % % % % % % % % % % % % % % % % %
% . . . . . . . . . . . . . . . . . . %
% * % % . % % % % . % % % % . % % % * %
% . % % . % % % % . % % % % . % % % . %
% . . . . . . . . . . . . . . . . . . %
% % % % % % % % % % % % % % % % % % . %
% . . . . . . . . . . . . . . . . . . %
% . % % % % % . % % % % % % . % % % . %
% . . . . . . . . . . . . . . . . . . %
% . % % - % % % . % % % % % . % % % . %
% . % G G G G % . % . * . . . % % % . %
% . % % % % % % . % . % % % . % % % . %
% . . . F . . . . . . . . . . % % % . %
% . . . % % % % . % . % . % . % % % . %
% % % . . . . . . % . % . % . % % % . %
% . . . % % % % . % . % . % . % % % * %
% * % . % . . . . % . % . % . % % % . %
% . . . % . % % . % . % . % . % % % . %
% % % . % . % % . % . % . % . % % % . %
% . . . . . . . . % . % . % . % % % . %
% * % . . . . . . % . % . % . % % % . %
% . % . % % % % % % . % . % . % % % . %
% . . . . . . P . . . . . . . . . . %
% % % % % % % % % % % % % % % % % % % %
```

The maximum width (in number of characters not including spaces) of any level will 20 and the maximum height (number of lines) will be 24. I will be testing with files that I have created myself, so your code must be able to generate levels based on them.

Ghosts

The four ghosts are named Blinky, Pinky, Inky and Clyde and are coloured Red, Pink, Cyan and Orange. All 4 ghosts should move in different directions (not all follow the same path). When the player eats a power pellet, the ghosts should try and run away from the player.

Example

To get an example of the how the pac-man game plays, go to the `http://pacman.platzh1rsch.ch/` and play the game there. Note that only the motion of the characters is required, animation is optional.

Submission

It is required that you use the Computer Science gitlab website (`https://csgitlab.ucd.ie/`) to manage and submit your project. This will require each student in the group to create an account on the website and be set up with either Developer, Maintainer or Owner permissions on the project. The accounts must use your **UCD student number** as the **username**. If you do not use your student number as your username, I will have no way of connecting your submission to the correct person and will not be able to give you your grade for the assignment.

One member of the each group must email me (sean.russell@ucd.ie), giving me their account details from csgitlab (full name and username/student number). I will then create a group for each project which you must add your other team members to. Inside this group you will have the ability to add projects (such as for storing the documentation and code).

Assessment

This section gives a breakdown of the approximate marking criteria for the assignment. The final marking scheme may vary slightly but will be relatively similar.

All submissions are to be completed by groups of at least 4 and no more than 6. Submissions by larger or smaller groups will **not be accepted** (unless I have given you permission).

The completed projects will be collected from the repository on gitlab after the deadline expires. Please ensure that your repository includes the following;

- All of the source files (.java) associated with the project (preferably
- Compiled javadoc documentation for the whole project explaining each of the classes, interfaces and methods and stating which member of the group wrote them (Should include name and student number)
- A README document explaining who was in the group (including names and UCD student numbers) and what you have achieved for each of the criteria in the marking scheme (estimate fail, pass or excellent).

If any of these are committed after the deadline, I will not be accepting them.

Marking Scheme

There are two parts to the marking scheme, one based on the code and features of the project and a second based on the teams ability to work together on the project using gitlab.

The marking scheme shown in table 1 is subject to change and is only about how the game is assessed. This means that it may be changed at any time without notice if I feel some parts were too easy or too hard and additional criteria may be added. The final grade from this section will be based on a weighted sum of the individual parts. The weights for each part will be based on the difficulty and importance. For example, design and cohesion is very important and will likely be weighted much higher than any other component.

The text explaining each can be used as a guide to the amount of work expected for the different parts of the assignment.

The marking Scheme given in table 2 shows the criteria for teamwork and coordination. Some of these are assessed based on the whole group and some are assessed based on the individual contribution of each student (judged based on the commit history of the members of the group).

Final Grade

The final grade for each student will be calculated based on the project grade (calculated based on table 1). This project grade can then be increased or decreased based on the first 4 criteria in table 2 (reduced if there was poor teamwork and use of git, or increased if there was excellent teamwork and use of git).

Finally, the final criteria or table 2 will be used increase or decrease the grade of a student as much as to 2 grades down or up. This will be based on analysis of the commit history of the project, so make sure that everyone contributes something to the project regularly (particularly in the final weeks of the project).

Item	Fail (E/F)	Pass (D/C)	Excellent (A/B)
Design and Cohesion	Poor use of classes such as only using a very small number of classes	OK use of classes but perhaps overusing them, such as the putting too many responsibilities into classes	Excellent use of classes, each class is used only for a sensible use and has good cohesion (right amount of sensible responsibilities)
Input	User input is not completed	User input is completed but not all actions are implemented	User input is completed and all actions are implemented
Display	No shapes are drawn on the game screen	Some of the shapes are drawn on the screen, or they are implemented using images loaded from files	All required drawable objects are implemented in code by specifying the coordinates that individual pieces should be drawn at
Menu	No menu or other options are shown, game moves straight into gameplay	Main menu is shown before gameplay starts	Game contains a main menu, hall of fame display (high scores) and an info screen showing the controls for the game
High Scores	No High scores are recorded or scoring is not calculated correctly	Previous high scores are loaded from file, but new scores are not saved, or opposite	Previous high scores are loaded from a file and any new high scores are saved in the file when a game is completed
Mazes	Mazes cannot be loaded from a file, there is a single default maze used instead	Mazes can be loaded from a file, but it does not work well for all mazes	Maze Loading works perfectly
Ghosts	Ghosts all move together or do not move well	Ghosts move in different directions or follow different paths when chasing the player	Ghosts more in different directions, get faster at higher levels and run away when player gets a power pellet

Table 1: Indicative Marking Scheme for Assignment

Item	Fail (E/F)	Pass (D)	Good (B/C)	Excellent (A)
Timeliness	Not everything was submitted by the deadline	The project and documentation was submitted, but only at the end and was not committed as the work was done or only by a single member of the group		Code and documentation was committed continually over the course of the term by all members in the team and within the time line expected
README	No readme documentation is provided or it is very basic	The readme documentation contains only contains a list of the students		The readme documentation contains a full description of the functionality of the game
Appraisal	No Appraisal was included in the README	The appraisal was very inaccurate, and claimed the project was better than the code showed		The appraisal was very accurate, and highlighted areas where the project met or exceeded the project specifications as well as areas where it did not meet them (if applicable)
Documentation	No Javadoc documentation is provided the javadoc was not compiled	Some Javadoc documentation is provided, but it does not cover all of the classes or public methods		Javadoc is provided explaining the classes and public methods thoroughly.
Individual Contribution	This individuals contribution was little/none		This individual shows a contribution that is significant, but only to some sections of the project	This individual shows a contribution that is significant to all sections of the project

Table 2: Indicative Marking Scheme for the overall Teamwork and Coordination