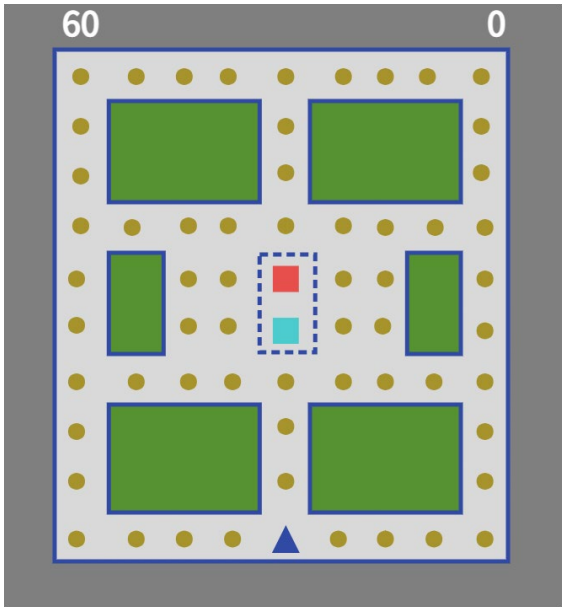
Computing Science CMPT 361 **Summer 2023**
Instructor: KangKang Yin **Simon Fraser University**

Programming Assignment #1 (20 marks)
Due on: Friday, June 23, at 23:59

Plagiarism is a serious academic offense: passing off someone else's work or ideas as your own in order to get a higher mark. Plagiarism is treated very seriously. The assignment you hand in must represent your own work. Submitting source code downloaded/copied from the WWW or your classmates' solutions as your own is deemed cheating, and an F grade will be awarded. However, reading and studying code from the web and discussing ideas with your classmates is allowed, but you must acknowledge their help, and still do your own work. In the README file you submit with your source code, you must list all the open-source code that you have studied, and all the people's names with whom you have discussed your assignment.

Late Policy: Late submissions are possible, but they will be penalized. One day late: 10% penalty; Two days late: 20% penalty; Three days late: 40% penalty; Four days late: 60% penalty; Five days late: 80% penalty; Six or more days late: 100% penalty.

Modified Pacman Game (20 marks):



You are to implement a simplified and **modified** version of the game Pacman as described below. You are advised to complete this assignment in several steps.

a. [4 marks] Game interface

Render the game window as shown in the image above. Blue lines are borders, light gray areas are corridors, and green blocks are obstacles. The center area surrounded by dashed lines is initially occupied by two square ghosts. Your blue triangle Pacman is initially at the center of the bottom. Ghosts can move out of the center, but Pacman cannot go in.

The top-left part of the game window shows a countdown timer, in seconds, initialized at 60. The top-right part of the game window shows the current score of Pacman, initialized as 0.

Please draw your interface in similar colors. The blue border lines can be omitted but all other items should be there.

b. [4 marks] Pacman movement

Pacman can move inside the maze in four directions with four direction keys. Every time a direction key is pressed, Pacman moves in the corresponding direction for one unit, which is the distance of two adjacent dots in the image, and then stop until the next key is pressed. Pacman cannot go into obstacles and the center area. Pacman cannot go out of the blue borders either.

c. [4 marks] Score and time

The countdown timer shows the time left in seconds. Pacman earns 100 points for each dot it eats on its way, so (score+=100). When the countdown drops to 0, or when all dots are eaten, the game ends and the final score is adjusted by the time left as score+=countdown*100.

d. [4 marks] Ghosts catch Pacman

Two ghosts first occupy the center area surrounded by the dashed lines and then move randomly inside the maze. Ghosts can overlap with each other, but cannot go inside obstacles or out of the blue borders. Each time a ghost catches Pacman, Pacman loses 500 points, so (score-=500). Then the ghost is put back to the center area and restarts its movement. If the score>0, Pacman revives where it was caught. If the score<=0, the game is over.

e. [4 marks] Additional game logic

1. Press s to start game, p to pause game, r to resume game, and shift+r to restart game. (1 mark)
2. One special item can be placed inside the maze which can be eaten by Pacman, after which Pacman can survive once without losing any points when caught by a ghost. (1 mark)
3. Improve your ghost movement AI, and/or the scores increased or decreased for Pacman after eating dots or eaten by ghosts, so that the game is balanced in the

sense that ghosts can win the game half of the time (score<=0), and your Pacman can win half of the time (score>0). (2 marks)

Note that the above steps build on top of each other, in order. You do not need to submit individual programs to correspond to these steps. If you can implement all the required parts, a single, complete program is sufficient. **No skeleton code** is provided.

Submission: Please submit a zip file with student number and your name (i.e., **`300000001_TerryFox.zip`**). The zip file contains an HTML, a JavaScript, and a README file. The README should acknowledge any help you have received and any discussion you have participated in. Please also document any steps not completed, the main idea of your e3 step as to how you tried to achieve a balanced game, and additional features/ideas/algorithms you have implemented. Your TA will be grading your assignment using the latest stable version of Chrome. If any extra instructions are needed for the TA to mark your assignment, please document them in README as well.

Note: You are not allowed to use any external packages, libraries, or tools other than the Angel's library (i.e. the files in the Common folder in the tutorial) for this assignment. You also have to use shaders to render the 2D shapes, and cannot use HTML CSS to draw them.