Pac-Man API Specification Document

Team: Bullet(Team J)

Member:
Siyu Xie(team Lead)
Huijia Zhu (doc lead)
Pengyu Zhang (dev)
Su Zhang (dev)
Suyue Zhang (dev)

Table of Contents

Design Decision

1. Following is the tech Stack we use in this project:

- Frontend: JavaScript
- Backend: Java Spark
- Communication protocol: Http Protocol

2. Following are the design patterns we use in our implementation:

- Strategy design pattern
- Singleton design pattern

3. The user extensible of our project is adding a study mode. By clicking the button on frontend, we can switch game interface to study interface, in case users get caught by teacher or boss.

Use Case

Use Case Name	Pac-Man Movement
Participating Actors	User
Flow of Events	1. User type keyboard up/down/left/right to change Pac-Man's movement direction; 2. Pac-Man moves straightly toward the current direction. 3. Movement updates.
Alternative Flows 1	1. If User types a keyboard up/down/left/right, Pac-Man will set up a new direction according to the user's input. 2. Movement updates.
Alternative Flows 2	1. If Pac-Man's forward board is a wall, then Pac-Man will stay and not move. 2. If Pac-Man's forward board is a ghost, then Pac-Man will do Collision with Ghost flows. 3. If Pac-Man's forward board is a bonus (fruits, small dots and big dots), then Pac-Man will get corresponding scores and bonus objects will be set "invisible". 4. Movement updates.
Entry Conditions	Update times meet pac-man movement cycle

Use Case Name	Pac-Man Collision with Ghost
Participating Actors	Pac-Man, Ghost
Flow of Events	1. Get the status of Ghost (normal, flash or "eyes" status) 2. Pac-Man will lose life/get scores or do nothing according to the status.
Alternative Flows 1	1. If Ghost is in normal status, Pac-Man will lose one life. Game will end if Pac-Man loses all lives. 2. All the Ghost and Pac-Man will return to its initial positions.
Alternative Flows 2	1. If Ghost is in flash status, Pac-Man will get a current ghost score. 2. All the Ghost scores will increase. (the first ghost Pac-Man collides with is worth 200, the second is worth 400, the third is worth 800, and the fourth is worth 1600) 3. Ghost will change its status to eyes status, and set up its movement strategy to GoHomeStrategy.

Alternative Flows 3	1. If Ghost is in “eyes” status, Pac-Man and Ghost will do nothing.
Entry Conditions	Ghost and Pac-Man are in same position

Use Case Name	Ghost Movement
Participating Actors	Ghost
Flow of Events	1. Ghost will have its movement strategy, every update will call its strategy update function. 2. Movement update.
Alternative Flows 1	1. If the Ghost has a RandomStrategy, then the Ghost will randomly move up/down/left or right. 2. If the Ghost's next move will collide with the wall, Ghost will stay and do nothing. 3. Movement update.
Alternative Flows 2	1. If the Ghost has a FollowingPacManStrategy, then the ghost will call SearchPacMan function to determine the next move. 2. Movement update.
Alternative Flows 3	1. If the Ghost has a GoHomeStrategy, then the ghost will call SearchHome function to determine the next move. 2. Movement update.
Entry Conditions	Update times meet Ghost movement cycle

Use Case Name	Init
Participating Actors	/
Flow of Events	1. Build a board matrix with “0” of wall and “1” of passaway. 2. Generate Pac-Man, 4 Ghost and dots. Place them in the correct place. 3. Return Game object to front-end (including board matrix, Ghost lists, Dots lists and Pac-Man).
Alternative Flows 1	/
Entry Conditions	/

Use Case Name	Update
Participating Actors	/
Flow of Events	1. Call game object's function update function. 2. For all Pac-Man and Ghosts in the lists, update its position according to Pac-Man movement and Ghost-Movement. 3. For Pac-Man, update it's status (life or score), all ghosts status, bonus status according to Pac-Man Collision with Ghost workflow. 4. Package these scores/difficulty/pac-man/ghosts/bonus and return response.
Alternative Flows 1	/
Entry Conditions	/

Use Case Name	Control
Participating Actors	User
Flow of Events	1. User type with keyboard up/down/left/right. 2. Pac-man will call function setDirection to set its direction to the user input.
Alternative Flows 1	/
Entry Conditions	/

Interface and Class Description

Interface

GhostStrategy:

GhostStrategy works as the interface of the concrete moving strategies.

Method	Description
String getName()	Get the strategy name.
void update(Ghost ghost)	Update the ghost using the behavior defined by the strategy.

Class

PaintObject:

PaintObject works as the abstract class of the concrete object classes painted on the game map.

Field	Description
protected Point pos	Position of the object.
protected Integer cycle	The update cycle of different objects.
protected Integer t	Game playing time.

Method	Description
void updatePos()	Update the position of the object according to the direction.
Point getPos()	Get the position of the object.
void setPos(Point newPos)	Set the position of the object.

Game:

Game works as the concrete class to show the game status and the status of the objects on the game map.

Field	Description
private Integer life	The times you can play in every game.
private Integer difficulty	The difficulty of the game.
private Integer score	The score you get in every game.
private Integer[][] map	The status of the map, 0 means wall, 1 means passageway.
private Pacman pacman	The pacman object we manipulate.
private List<Ghost> ghosts	The list of ghosts on the map.
private List<Bonus> bonuses	The list of bonuses on the map.
final Point exitLeftPos	The exit on the left side of the map.
final Point exitRightPos	The exit on the right side of the map.
final String mapFileName	The map file name.
static private Game ONLY	The singleton object.

Method	Description
private Game()	The constructor.
static public Game getOnly()	Get the singleton object.
private void newGame()	New game, initialize necessary members.

private void levelUp()	Pacman eats all the dots and game difficulty levels up.
private void buildMap()	Build the map and crucial parts according to the map file.
private void loseLife()	The pacman loses one life.
public void addScore(int scores)	Gain scores.
public UpdateResponse update()	Update the game and wrap up the response
public void updatePositions()	Update the positions of all objects.
public void updateStatus()	Update the status of all objects.
Public Point getExitLeftPos()	Get the left exit.
Public Point getExitRightPos()	Get the right exit.
public Boolean isWall(Point p)	Tell if the position is a wall.
public String randomDir(Point p)	Choose a random direction from position p.
public String findPath(Point src, Point dst)	Find the shortest path from position src to position dst.
public Point getPacmanPos()	Return the position of the pacman.

Response:

Response works as the concrete class to let the frontend get relative data to paint the map each interval.

Field	Description
int life	Current pacman life.
int difficulty	Current game difficulty.
int score	Current game game score.

Pacman pacman	Current pacman information.
Ghost[] ghosts	Current information of all ghosts.
Bonus[] bonuses	Current information of all bonuses.