COMP 504 Graduate Object-Oriented Programming and Design

Final Game Specification

Group Dog

Team lead: Lize Chen

Tech Lead: Huaminghui Ding

Documentation Lead: Tingting Li

Developer: Daolun Chen

Developer: Ruimin Li

Developer: Yichen Sun

Developer: Junhao Yu

Rice University December 1th, 2021

Content

1 Overview

1.1 introduction

People of all ages play video games for many different reasons. Video games serve a purpose and everyone plays for a reason. Novelty or variety keeps life interesting, fun, and engaging. Video games give us the opportunity to do something new, to become masterful, and to fail with freedom. To provide a place for people to benefit from video games, our team decided to build an online game app.

1.2 project description

The object of our team is to design and implement a user-friendly web app to allow people to have fun with the Pac-Man game. Users visit our Heroku app link and start playing under our guidance. Each online user manipulates the Pac-Man one the keyboard to eat dots and avoid ghosts. A user has three life and experience excitement while winning or failing the game. We apply Object-oriented Programming, design patterns to structure our code and meet requirements. We neither use a database nor front end frameworks. Technologies include Java, HTML, JavaScript, and Spark Java framework.

2 Use cases

(1) Initialize game

- The user decides the level of game - easy or hard.
- The user decides the number of ghosts and the type of fruits.
- Click easy button to start the game.

- The user sees 240 dots in the passage, including four blinking big dots at four corners.
- The user sees Pac-Man and moving ghosts at their starting positions.
- The score on the top center starts from zero.
- Lives on the top left starts from three.
- Ghosts move with their pre-defined chasing strategy.
- The hard level contains our team name.

(2) Pac-Man movement

- Pac Man is moved up, down, left, or right by user clicking on WASD or arrow keys
- Pac Man can only move in the passage.
- Pac-man starts with 3 lives, if Pac Man hits a ghost, this life is over; restart with one less life.
- User tries to collect all dots without touching a ghost to complete each level.
- When in contact with a wall, Pac Man will stop and be able to go in a different direction.
- Pac-Man can exit one side of the game board and enter in on the other side.
- Pac-Man can eat small dots, large dots and fruit to gain more scores.
- Pac-Man can hit ghosts if the ghost turns dark blue and starts flashing.

(3) Ghost movement

- Users cannot manipulate ghosts.
- In chasing mode, Ghosts will use different behaviors to move toward or away from Pac-Man.
- Ghosts will move randomly if they are frightened.
- Ghosts can only move in the passage.
- All ghosts go back to its born position if ghost collides with the Pac-man.
- Ghosts can exit one side of the game board and enter in on the other side.

- In this game, ghosts will switch between different modes.

Mode	**Description**
Chasing	The ghosts will actively pursue the Pac-Man using pre-defined behaviors.
Frightened	The ghosts turn dark blue and then start flashing using random strategy.
Dead	The ghosts become two eyes and travel quickly to the square box in the middle of the screen. The strategy will back to the born strategy after the ghost comes out again.

- Ghosts chase the Pac-Man in different pre-defined behaviors. Each ghost has a target. We calculate the nearest path between a ghost and the target using the A star algorithm every time we update the game. Ghost's movement in chasing mode is based on the nearest path.

Color	**Target**	**Strategy**	**Description**
Red	Pac-Man	ChaseStrategy	The red ghost will target at the location of Pac-Man and get closer to it.
Pink	Four units ahead of Pan-Man's direction	AmbushStrategy	The pink ghost will target ahead of Pac-Man four units and get closer to it.
Orange	Pac-Man/zone in corner	StupidStrategy	Within 8 squares: back to its zone in a pre-defined corner. More than 8 squares: same as red using chase strategy
Cyan	Based on distance	AvoidStrategy	Avoid the Pac-man basing on the distance from Pac-man to ghost. Pac-man will never meet Cyan.

(4) Score calculation

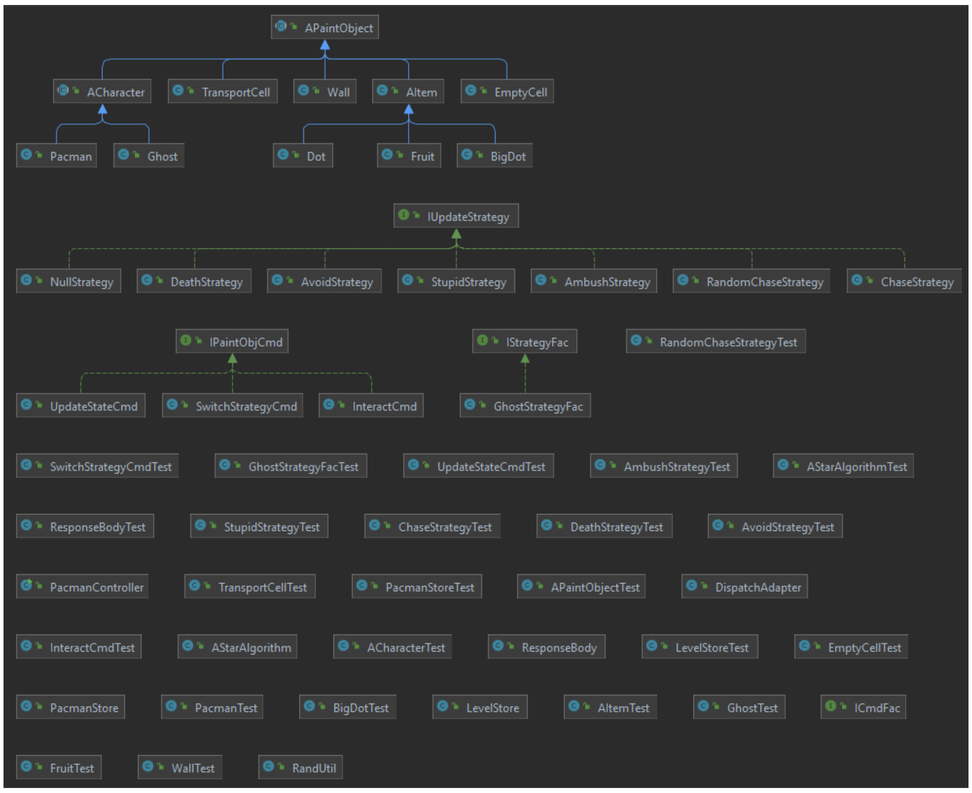
- The score starts from zero.
- If Pac Man hits a small dot, the dot disappears, and 10 points are awarded and displayed.
- Periodically, a piece of fruit will appear that will be worth 100 points. When Pac-Man eats the piece of fruit, the fruit disappears.
- When Pac-Man eats large dots, the ghosts turn dark blue and then start flashing (blue and white colors) for a small period of time. Each large dot is 50 points.
- If Pac-Man collides with a dark blue or flashing ghost, the ghosts become two eyes and travel quickly to the square box in the middle of the screen.
- For a single large dot, 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.
- If Pac-Man collides with a non-dark blue or non-flashing ghost, Pac-Man loses 1 life.
- The score becomes zero if the user enters the next level.
- The lives come back to three if the user enters the next level.

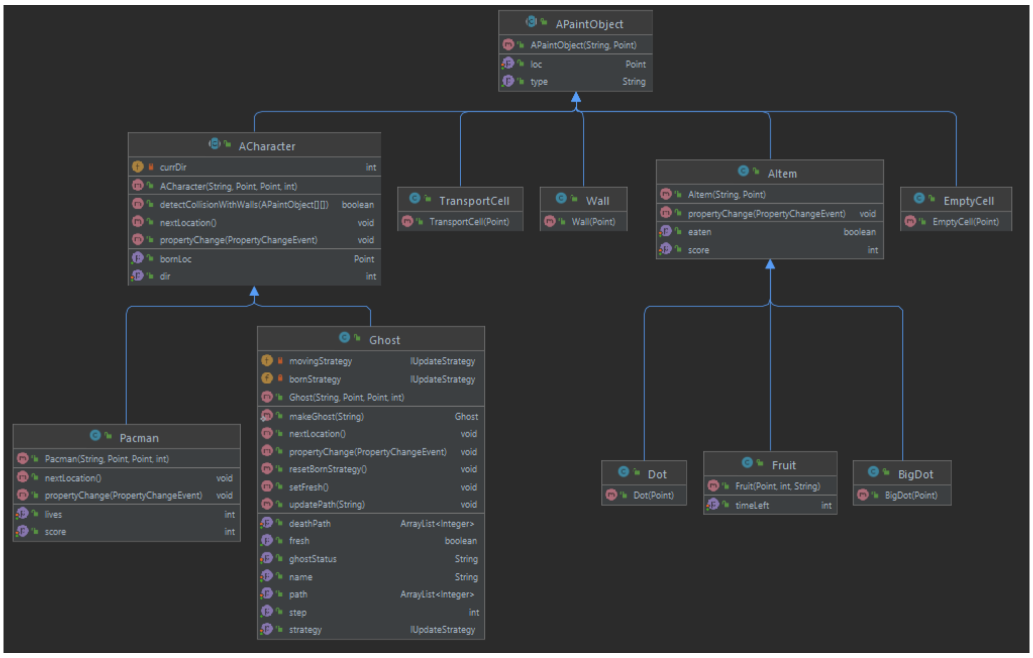
(5) Game over

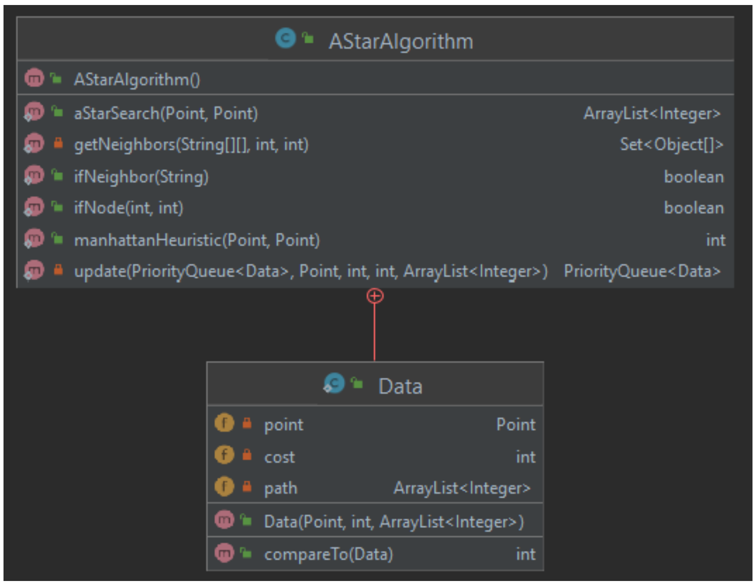
- If the last life is used, the user sees a game over warning shown on the screen.
- Pac-Man advances to the next level if Pac-Man eats all the dots before losing 3 lives. The next level is more difficult.
- Game is complete when final level is defeated.

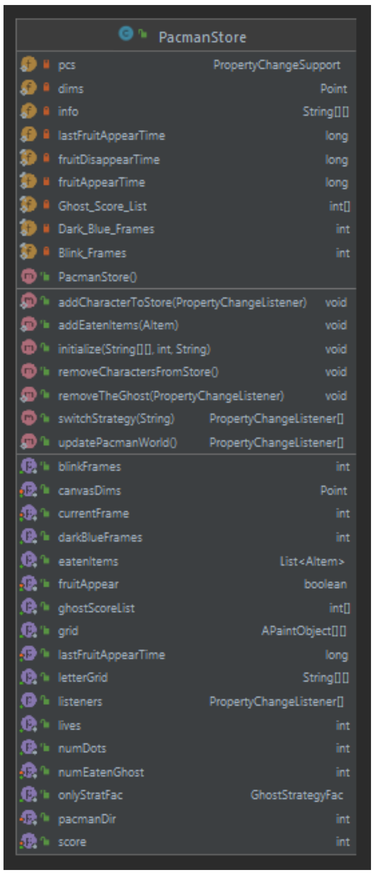
3 Project Design

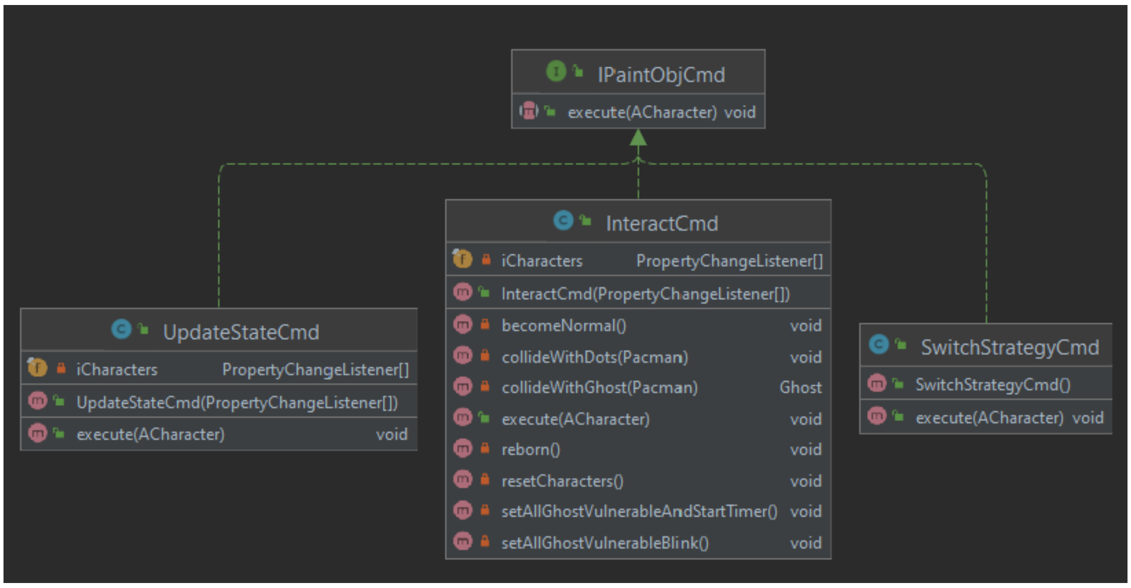
3.1 UML diagram

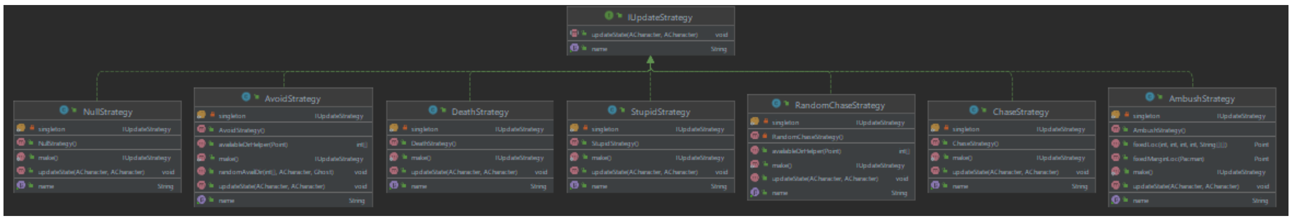


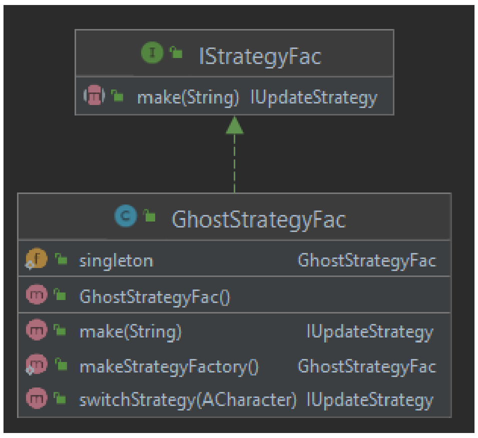


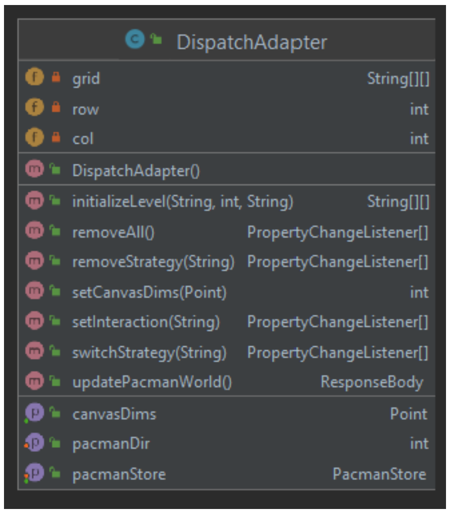

AStarAlgorithm
AStarAlgorithm()
aStarSearch(Point, Point) ArrayList<Integer>
getNeighbors(String[][], int, int) Set<Object[]>
ifNeighbor(String) boolean
ifNode(int, int) boolean
manhattanHeuristic(Point, Point) int
update(PriorityQueue<Data>, Point, int, int, ArrayList<Integer>) PriorityQueue<Data>
Data
point Point
cost int
path ArrayList<Integer>
Data(Point, int, ArrayList<Integer>)
compareTo(Data) int

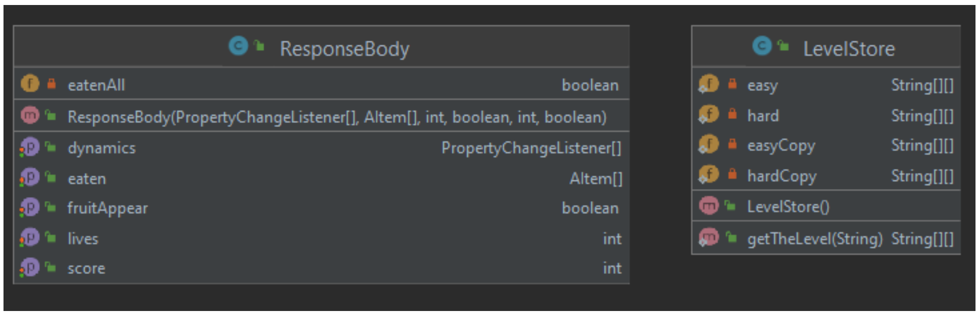

PacmanStore
pcs PropertyChangeSupport
dims Point
info String[][]
lastFruitAppearTime long
fruitDisappearTime long
fruitAppearTime long
Ghost_Score_List int[]
Dark_Blue_Frames int
Blink_Frames int
PacmanStore()
addCharacterToStore(PropertyChangeListener) void
addEatenItems(AItem) void
initialize(String[][], int, String) void
removeCharactersFromStore() void
removeTheGhost(PropertyChangeListener) void
switchStrategy(String) PropertyChangeListener[]
updatePacmanWorld() PropertyChangeListener[]
blinkFrames int
canvasDims Point
currentFrame int
darkBlueFrames int
eatenItems List<AItem>
fruitAppear boolean
ghostScoreList int[]
grid APaintObject[][]
lastFruitAppearTime long
letterGrid String[][]
listeners PropertyChangeListener[]
lives int
numDots int
numEatenGhost int
onlyStratFac GhostStrategyFac
pacmanDir int
score int


IPaintObjCmd
execute(ACharacter) void
InteractCmd
iCharacters PropertyChangeListener[]
InteractCmd(PropertyChangeListener[])
becomeNormal() void
collideWithDots(Pacman) void
collideWithGhost(Pacman) Ghost
execute(ACharacter) void
reborn() void
resetCharacters() void
setAllGhostVulnerableAndStartTimer() void
setAllGhostVulnerableBlink() void
UpdateStateCmd
iCharacters PropertyChangeListener[]
UpdateStateCmd(PropertyChangeListener[])
execute(ACharacter) void
SwitchStrategyCmd
SwitchStrategyCmd()
execute(ACharacter) void


IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
NullStrategy
singleton IUpdateStrategy
NullStrategy()
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
AvoidStrategy
singleton IUpdateStrategy
AvoidStrategy()
availableDirHelper(Point) int[]
make() IUpdateStrategy
randomAvailDir(int[], ACharacter, Ghost) void
updateState(ACharacter, ACharacter) void
name String
DeathStrategy
singleton IUpdateStrategy
DeathStrategy()
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
StupidStrategy
singleton IUpdateStrategy
StupidStrategy()
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
RandomChaseStrategy
singleton IUpdateStrategy
RandomChaseStrategy()
availableDirHelper(Point) int[]
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
ChaseStrategy
singleton IUpdateStrategy
ChaseStrategy()
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String
AmbushStrategy
singleton IUpdateStrategy
AmbushStrategy()
fixedLoc(int, int, int, int, String[][]) Point
fixedMarginLoc(Pacman) Point
make() IUpdateStrategy
updateState(ACharacter, ACharacter) void
name String


IStrategyFac
make(String) IUpdateStrategy
GhostStrategyFac
singleton GhostStrategyFac
GhostStrategyFac()
make(String) IUpdateStrategy
makeStrategyFactory() GhostStrategyFac
switchStrategy(ACharacter) IUpdateStrategy


DispatchAdapter
grid String[][]
row int
col int
DispatchAdapter()
initializeLevel(String, int, String) String[][]
removeAll() PropertyChangeListener[]
removeStrategy(String) PropertyChangeListener[]
setCanvasDims(Point) int
setInteraction(String) PropertyChangeListener[]
switchStrategy(String) PropertyChangeListener[]
updatePacmanWorld() ResponseBody
canvasDims Point
pacmanDir int
pacmanStore PacmanStore



3.2 Interface/Abstract class

3.2.1 DispatchAdapter

Class description

This adapter communicates with the view (paint objects) and the controller.

Method Details

Method	Description
public DispatchAdapter()	Constructor call.

public String[][] initializeLevel(String level, int ghostNum, String fruitType)	Initialize the game. The filename is named by the level of the game. Number of ghosts and type of fruit is decided by the user.
public ResponseBody updatePacmanWorld()	Update the pacman world.
public int setCanvasDims(Point p)	Set the canvas.
public PropertyChangeListener[] setInteraction(String interaction)	Set the interaction between the ghosts and pacman.
public PropertyChangeListener[] switchStrategy(String strat)	Switch the strategy for switcher objects.
public PropertyChangeListener[] removeStrategy(String strategy)	Remove the strategy for switcher objects.
public PropertyChangeListener[] removeAll()	Remove all ghosts and pacman from listening for property change events for a particular property.
public Point getCanvasDims()	Get the canvas dims.
public PacmanStore getPacmanStore()	Get the Pac-Man store.
public void setPacmanDir(int dir)	Set the direction of Pac-Man

3.2.2 Package agent

Class description

class	Description
ACharacter	Abstract class extends the APaintObject and implements the PropertyChangeListener.
Ghost	Subclass extends from ACharacter, the ghost drawn in the pacman world.

Pacman	Subclass extends from ACharacter, the pacman drawn in the pacman world.

3.2.3 Package cmd

Class description

class	Description
InteractCmd	Concrete class to handle interactions.
SwitchStrategyCmd	Concrete class to switch strategies of ghosts.
UpdateStateCmd	Concrete class to update Pac-Man and ghosts.

Interface description

Interface	Description
ICmdFac	A factory that makes commands.
IPaintObjectCmd	Pass commands to objects in the pacman world. The objects must execute the command.

3.2.4 Package item

Class description

class	Description
AItem	Class that extends APaintObject and implements PropertyChangeListener. Each item has score and eaten status.
APaintObject	Objects drawn in the pacman world.
BigDot	Big dot with the score of 50.
Dot	Small dot with the score of 10.
EmptyCell	The empty passage.
Fruit	Fruit with the score of 100.

TransportCell	The exit sides on the game board that can transport the Pac-man and ghosts.
Wall	The wall.

3.2.5 Package strategy

Class description

class	Description
AmbushStrategy	Use for pink. Advanced four units before the Pac-man direction.
AStarAlgorithm	A star Algorithm with Manhattan heuristic. Calculate the shortest path from a starting point to an ending point.
AvoidStrategy	Use for Cyan. Avoid the Pac-man basing on the distance from Pac-man to ghost.
ChaseStrategy	Use A*. This is for the red ghost. Use the location of the current Pac-man and the ghost. Death strategy is the same as this strategy - back to the born strategy
DeathStrategy	A ghost moves back to its born location in the nearest path.
GhostStrategyFac	A factory to make different strategies.
NullStrategy	Use for initialize.
RandomChaseStrategy	A strategy to update a ghost position randomly.
StupidStrategy	Use for orange. Within 8 units, go back to the corner; otherwise use chaser.

Interface description

Interface	Description
IStrategyFac	A factory that makes strategies.
IUpdateStrategy	An interface for ball strategies that determine the ball behavior.

3.2.6 PacmanStore

Class description

A store containing the current Pac-man world.

Method Details

Method	Description
public PacmanStore()	Constructor call.
public void initialize(String[][] info, int ghostNum, String fruitType)	Initialize the game. Info is the letter map.
public static int getNumDots()	Get the number of dots.
public static APaintObject[][] getGrid()	Get the 2-D array of the game.
public static String[][] getLetterGrid()	Return the letter grid.
public static void addEatenItems(AItem item)	Add an eaten item to the list.
public static PropertyChangeListener[] updatePacmanWorld()	Call the update method on all the observers to update their position in the pacman world.
public PropertyChangeListener[] getListeners()	Get the listener list.
public static void setPacmanDir(int curDir)	Set the current direction of pacman.
public static void addCharacterToStore(PropertyChangeListener pcl)	Add a character that will listen for a property change
public void removeCharactersFromStore()	Remove all characters from listening for property change events for a particular property.
public static boolean isFruitAppear()	Whether the fruit disappears or not.

public static void setFruitAppear(boolean fruitAppear)	Set the fruit appears.
public static int getScore()	Get current score.
public static void setScore(int score)	Set current score.
public static List<AItem> getEatenItems()	Get the list of eaten items.
public static int[] getGhostScoreList()	Get ghost score lists.
public static int getLives()	Get the number of remaining lives.

3.2.7 LevelStore

Class description

A store containing the different levels.

Member Details

Member	Description
private static String[][] easy	Store the game layout of easy level
private static String[][] hard	Store the game layout of hard level
public static String[][] getTheLevel(String level)	Get a specific level

3.2.8 ResponseBody

Class description

A store containing the dynamic information of the game.

Member Details

Member	Description
dynamics	The current listener list.
eaten	The eaten dots
score	The score

fruitAppear	The status of a fruit
lives	Remaining number of lives
eatenAll	the status of eaten dots

3.3 API design

(1) Server API

endpoint	verb	payload	response	description
/level	post	level	String[][]	Initialize the game.
/update	get	pacmanDirection	PropertyChange Listener[]	Update the pacman world based on pacman's direction.
/clear	get	---	---	Clear the ghosts and pacman.
/canvas/dims	post	Height, wdith, ghostNum, fruitType	---	Initialize the game. Extensible for ghost num and fruit type

(2) User extensible

The game can be extensible in the following ways.

- The game map will change to a different layout basing on the user's selection.
- The number of ghosts can change from one to four. The more ghosts, the harder for Pac-man to avoid the ghosts.
- The type of fruit can be changed based on selection like apple and strawberry.

3.4 Design Pattern

3.4.1 Template Design Pattern

To design API that meet all the use cases and complete implement, we analyzed variant and invariant operations for the project. Interfaces and abstract class set up invariant skeleton of the algorithm, and concrete subclasses implement specific variant details.

3.4.2 MVC Design Pattern

Our group used the PacmanController to instantiate the DispatchAdapter and communicate with the model and view.

3.4.3 Factory design pattern

To develop a robust and scalable application, the controller and DispatchAdapter should not care how an object is created. In this project, our team used the factory to handle details. In this game, a ghost follows and switches different strategies to move. So it is necessary to design the GhostStrategyFac to make strategies.

3.4.4 Strategy design pattern

In chasing mode, scattering mode, frightened mode, and dead mode, a ghost need to move toward/away from Pac-Man. Our group will implement different types of ghost-related movement strategies and implement the updateState functions. Therefore we can update the ghost position to players. The strategy design pattern is appropriate for different behaviors.

3.4.5 Observer design pattern

According to the use cases regarding Pac-Man eating and interactions with ghosts, information about characters (Pac-Man and ghosts) should able to be updated when some event occurs. Therefore our group implemented the function propertyChange(PropertyChangeEvent evt) of PropertyChangeListener interface in concrete classes Pacman and Ghost. All these Pac-Man world objects will be added to the listener list.

3.4.6 Command design pattern

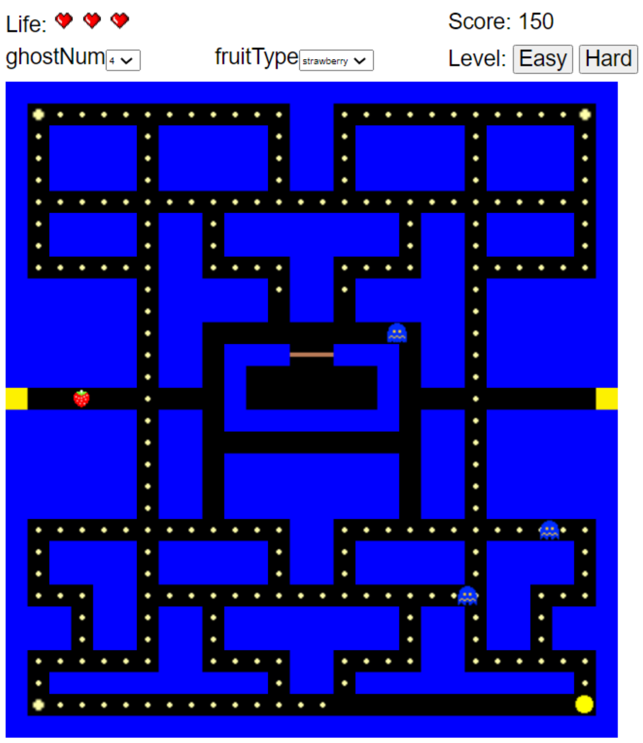
We designed an interface IPaintObjectCmd to pass commands to receivers in the pacman world. The receiver does not care about what the command is and just execute the command. Commands include interaction between items, update location, switch strategies, and so on.

3.4.7 Singleton Design Pattern

Some class contains invariant attributes and information. We only need one instance of the class to save space, instead of allocating space for objects again and again. Therefore, we used the singleton design pattern for concrete classes GhostStrategyFac and ghost moving strategies.

4 Result

https://pacman-final-team-dog.herokuapp.com/

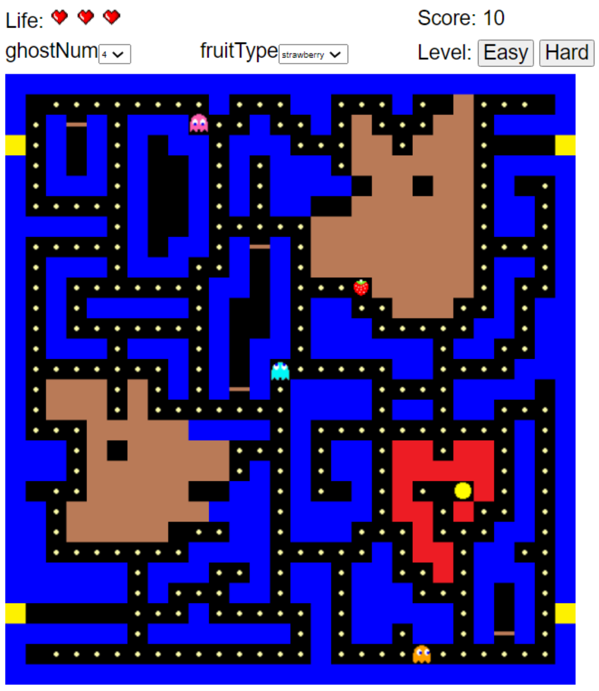

Life:

ghostNum 4

fruitType strawberry

Score: 150

Level: Easy Hard


Life:

ghostNum 4

fruitType strawberry

Score: 10

Level: Easy Hard