Zoom Coders
Pac-Man API Specification

Alex Liapis (Team Lead) • Yang Mi (Tech Lead) • Dingyu Wang (Doc Lead) • Minyu Jiang (Dev) • Dingjia Li (Dev) • Jiamin Liu (Dev)

Introduction:

This document includes comprehensive details of our pac-man application. Included are use cases, abstract classes, interfaces, and design decisions.

Use Cases

***Pac-man*:**

1. Will move in a direction based on what directional keys the user presses.
2. The pac-man's mouth will change directions.
3. The pac-man does not move when hitting a wall.
4. The pac-man will appear at the other side of the board after going through the tunnel.
5. Fruit disappears when the pac-man eats it.
6. When pac-man eats large dots, the ghosts turn dark blue and then start flashing (blue and white) for a period of time.
7. If pac-man collides with a flashing ghost, the ghosts become two eyes and travel quickly to the square box in the middle. 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, etc.
8. If pac-man collides with a non dark blue or non-flashing ghost, Pac-Man loses 1 life.
9. Pac-man has 3 lives total.

Ghost:

1. Ghosts will use some behavior to move toward or away from pac-man.
2. Ghosts will have different behavior.
3. Ghosts have different colors and velocities.
4. Ghosts start in the middle box.

Board:

1. There will be multiple levels a user can choose and each will be a different board.
2. There are "tunnels" on sides of the wall where if the pac-man goes in, it comes out of the other side.
3. Each board will have approximately 240 small dots worth 10 points each.
4. Periodically a fruit worth 100 points will appear briefly on the screen.
5. Each corner has 4 large blinking dots worth 50 points each.
6. The ghosts start in the middle of the board in a box.
7. The paths in the board are at width of a pac-man.
8. There will be info like a score, dots consumed, lives left in the board.
9. Game ends and will display "game over" on the board when the pac-man loses 3 lives.
10. Changes level (different board) when pac-man eats all dots.
11. Each level (board) becomes more difficult.

Abstract Classes and Interfaces

ACharacter (abstract class):
An abstract character of the pac-man game. Could be a ghost or pac-man.

	Member fields
vel (Point)	The velocity of the character.
updateStrategy (IUpdateStrategy)	The movement strategy of the character.
collideStrategy (ICollideStrategy)	The collision strategy of the character.
color (String)	The color of the character.
direction (int)	The direction the character is facing
originalLoc (Point)	The original location of the ACharacter.
size (int)	The size of the ACharacter.
	Methods
ACharacter(String name, Point loc, Point vel, String color, IUpdateStrategy updateStrategy, ICollideStrategy collideStrategy, int direction)	Constructs a new character based on the supplied information.
getColor()	Get the color of the ACharacter. Return: String color.
setColor(String c)	Set the color of the ACharacter. Parameter: String color.
getOriginalLoc()	Get the location of the character in the game. Return: Point originalLoc.
setLocation(Point loc)	Set the location of the character in the game. Parameter: Point loc.
getVel()	Get the velocity of the character in the game. Return: Point vel.
setVel(Point vel)	Set the velocity of character in the game. Parameter: Point vel.
getStrategy()	Get the strategy of the character in the game.

	Return: IUpdateStrategy updateStrategy.
setStrategy(IUpdateStrategy updateStrategy)	Set the strategy of the character in the game. Parameter: IUpdateStrategy updateStrategy.
getCollideStrategy()	Get the collision strategy of the character in the game. Return: ICollideStrategy collideStrategy.
getDirection()	Get the direction of the ACharacter. Return: int direction
setDirection(int direction)	Set the direction of the ACharacter. Parameter: int direction
getSize()	Get the size of the ACharacter. Return: int size.
setSize(int size)	Set the size of the ACharacter. Parameter: int size.
setCollideStrategy(ICollideStrategy collideStrategy)	Set the collision strategy of the character in the game. Parameter: ICollideStrategy collideStrategy.
detectCollisionWithWalls()	Detects collision between a character and a wall. Changes direction if the character collides with a wall. Return: Boolean: True if collision, false otherwise.

AItem (abstract class):
An abstract item in the game. Could be a dot, big dot, fruit, or a wall.

	Member fields
color (String)	The color of the item.
score (int)	How much score the item gives.
isEaten (boolean)	Whether the item has been eaten.
	Methods
AItem(String name, Point loc, String color)	Constructs a new item with the supplied information.

getColor()	Get the item's color. Return: String color.
setColor(String c)	Set the color of the item. Parameter: String c.
getScore()	Get the score of the item. Return: int score.
isEaten()	Whether the item is eaten. Return: boolean
setEaten(boolean eaten)	Set if an item is eaten. Parameter: eaten

ICollideStrategy (interface):
Interface for ACharacter strategies that determine the collision behavior between characters in the game.

	Methods
getName()	Get the name of the collision strategy. Return: String name.
collideState(APaintObj other)	Update the state of collision for the character. Parameters: APaintObj other.

IUpdateStrategy (interface):
IUpdateStrategy interface is used to determine the behavior of a paint object in the canvas over time.

	Methods
getName()	Get the update strategy name. Return: String name.
updateState(ACharacter context)	Update the state of the ACharacter. Parameters: ACharacter context.
updateState(ACharacter pacman, ACharacter context)	Update the state of the ACharacter. Context is the character to apply strategy to.

GameLevel:
A game level in the pac-man game.

	Member fields
levelCount (int)	The number of the level.
walls (List<Walls>)	The list of walls of the game board for the level.
fruits (List<fruits>)	The list of fruits for the level.
dots (List<Dot>)	The list of dots for the level.
bigDots (List<bigDots>)	The list of big dots for the level.
zoom (Zoom)	A special zoom item.
	Methods
GameLevel(int levelCount, List<Wall> walls, List<Fruit> fruits, List<Dot> dots, List<BigDot> bigDots, HashMap<Point, Boolean> fruitLocations, Zoom zoom)	Constructs a level with the supplied information.
getLevelNum()	Get the level number. Return: int levelNum.
setLevelNum(int levelNum)	Set the level number. Parameter: int levelNum.
getLevelName()	Get the name of the level. Parameter: String levelName.
setLevelName(String levelName)	Set the name of the level. Parameter: String levelName.
getWalls()	Get the wall of the level. Return: List<Wall> walls.
setWalls(List<Wall> walls)	Set the walls of the level. Parameter: List<Wall> walls.
getFruits()	Get the fruits of the level. Return: List<fruits> fruits.

setFruits(List<Fruit> fruits)	Set the fruits of the level. Parameter: List<Fruit> fruits.
addFruit(Fruit fruit)	Add a fruit to a level. Parameter: Fruit fruit.
removeFruit(Fruit fruit)	Removes a fruit from the level. Return: Fruit fruit.
getDots()	Get all the dots in the level. Return: List<Dot> dots.
setDots(List<Dot> dots)	Set the dots in the level. Parameter: List<Dot> dots.
removeDot(Dot dot)	Removes a dot in the level. Parameter: Dot dot.
getBigDots()	Get the big dots in the level. Return: List<bigDots> bigDots.
setBigDots(List<BigDot> bigDots)	Set the big dots in the level. Parameter: List<BigDot> bigDots
removeBigDot(BigDot bigDot)	Removes a big dot in the level. Parameter: BigDot bigDot.
removeZoom()	Removes the zoom object.
getZoom()	Get the zoom object. Return: Zoom zoom.
generateZoom()	Generates a zoom object. Return: returns a zoom object.
getRandomDotLocation()	Generates random location for dot. Return: Point.

IPacmanCmd (interface):
IPacmanCmd interface that is used to make change to the pac man world.

	Member fields
	Methods
execute(APaintObj context)	Executes the command for APaintObj context. Parameter: APaintObj context.
execute(APaintObj pacman, APaintObj context)	Executes the command for APaintObj pacman and APaintObj context. Parameter: APaintObj pacman, APaintObj context

APaintObj (abstract class):
An object that will be drawn in the pac-man game world.

	Methods
loc (Point)	Location of the object on the canvas.
String name	Name of the object.
	Methods
APaintObj(String name, Point loc)	Constructs an APaintObj.
getLoc()	Get the location of the APaintObj. Return: Point loc.
setLoc(Point loc)	Set the location of the APaintObj. Parameter: Point loc.
getName()	Get the name of the APaintObj. Return: String name.
setName(String name)	Set the name of the APaintObj. Parameter: String name.

GameContext:
The pac-man context and game board for the game.

	Member fields
levelInstance (GameLevel)	Level instance either 1 or 2.
pacman (Pacman)	The pacman object.
Ghosts (List<Ghost>)	The list of the ghosts for the game.
Status (int)	The status of the game, whether it's paused.
maxLives (int)	The maximum number of lives for the pac-man.
currentScore (int)	The current score of the game.
highestScore (int)	The highest score of the game.
levelCount (int)	Total counts of the levels.
numberOfGhosts (int)	The number of ghosts.
numberOfFruits (int)	The number of fruits.
isZoomAvailable (boolean)	Whether the zoom item is available.
ghostScore (int)	The ghost score.
nextFruitIndex (int)	The index of the next fruit.
fruitsActivated (int)	The number of fruits activated so far.
fruitActiveTime (int)	How long fruits are active for.
zoomEffectTime (int)	How long zoom items are active for.
inEffect (boolean)	Whether the item is in effect.
effectLeftTime (int)	Time left on the effect
scoreFactor (int)	The multiply factor for the score.
ghostFlashingTime (int)	How long ghosts keep flashing.
maxGhosts	Maximum number of ghosts.
	Methods
GameContext(int gameLevel, int numberOfGhosts, int maxLives, boolean isZoomAvailable, int highestScore)	Constructs new GameContext.
init()	Initialize the board with the game settings.

getLevelCount()	Get the Level Count. Return: int levelCount.
setLevelCount(int levelCount)	Set the level Count. Parameter: int levelCount.
getLevel()	Get the level. Return: Alevel level.
setLevel(ALevel level)	Set the level. Parameter: Alevel level.
getPcs()	Get the property change support. Return: Return PropertyChangeSupport pcs.
getLives()	Get the number of lives. Return: int lives.
setLives(int lives)	Set the number of lives. Parameter: int lives
getCurrentScore()	Get the current score. Return: int currentScore.
setCurrentScore(int currentScore)	Set the current score. Parameter: set currentScore.
getHighestScore()	Get the highest score. Return: highestScore.
setHighestScore(int highestScore)	Set the highest score. Parameter: highestScore.

Design Decisions

- Our game will be rendered inside a canvas, with buttons & menus outside of the canvas to set the level, reset, etc.
- The canvas will update according to the user settings when the game is ready to be started. Highest score, current lives, etc will be updated as those events occur on the canvas.

- Pacman & ghosts are both subclasses from ACharacter, since they both have directions, velocity, strategies, etc.
- Character direction is decoupled from velocity so that the magnitude of the character speed can be separately changed from the directions. And velocity can be used to store the next moving direction.
- Pacman & ghosts have two versions of the same sprite in order to create the animation of movement. These animation frames are cycled between in the view.
- Previous location data & redraw parameters are used to ensure that old ghost positions are cleared & ghosts do not erase previous dot, big dot, & fruit drawings on the canvas when passing over those items.
- Each ghost has its own unique default chasing strategy. This strategy is changed to a run away strategy when pacman eats a big dot or return to base strategy when eaten.
- PacmanContext class holds the game information: level, score, etc. Particular level information is stored in the concrete subclasses of GameLevel. The DispatchAdapter ties these all together for each game session.
- Collisions between objects are assigned a specific command according to the type of collision.
- A general update state command is used to refresh the character locations & parameters.