BigFoot's Game API spec doc

Use Case Design

User	Initialize Game	User starts the application and the main menu is displayed. The main menu contains: "Play Game", "View High Scores" and "Setting". On the board, there are "tunnels" on sides of the wall where if the pac-man goes in, it comes out of the other side. Each board will have approximately 240 small dots, 4 big dots in corners. In each game, a fruit worth 100 points will appear on the screen for 10s. Current score and life used are shown on the screen
User	View High Scores	User chooses "View High Scores" from the menu and a table listing the username entered with each corresponding high score is displayed. This screen is also displayed upon game completion as well as loss of all 3 lives. The top 5 high scores are kept.
User	Setting	User chooses "Setting" and the current game level, number of ghosts and lives are displayed. The user can change the game level, number of ghosts and lives in this menu item and then go back and play the game. In total, we have 2 levels for the game.
User	Pause / Continue Game	When users click Pause button, users are able to stop game progress while playing and then resume at a later time. The Pause button will change to the Continue button. When users click the Continue button, they can start where the saved game was stopped. The Continue button will change to the Pause button.
User	Pass level I	The game will be restarted as level II. The differences for two levels are the map, dots positions and the blue time of ghosts. In level two, the map changes and the blue time is shorter.
User	Win / Lose the Game	When all three lives are lost, you lose and the game exits. When all the dots are eaten, this level game is passed. When the final level is passed, you win and a completion screen is displayed and then the game exits. The Highest Score displays on the top left screen
Pacman	Meet small dots	Pacman eats dots and earns 10 points.

Pacman	Meet big dots	Pacman eats dots and earns 50 points. The ghost's color changes to blue and flashes for 10 seconds.
Pacman	Meet ghosts	Pacman loses 1 life. All ghosts go back to home base and pacman goes back to game start position. In total, pac-man has 3 lives. If it has no life to lose then the game is over.
Pacman	Meet blue ghosts	The first ghost Pacman collides with is worth 200 score, the second is worth 400, the third is worth 800, and the fourth is worth 1600. The meeted ghosts become two eyes and travel quickly to the home base. If one ghost was eaten, a new ghost will generate from the home base.
Pacman	Normal movement	Pacman moves in a direction based on directional keys. It will appear at the other side of the board after going through the tunnel. It cannot move through walls.
Ghost	Move pattern	a. Wander: wandering in the map, just random walk. b. Chase: chasing pacman all the time.
Ghost	Go out of home base	When the game starts, 4 ghosts in the middle will go out of home base and use some behavior to move toward or away from pac-man. When they die, they will turn into an eye and go back to the home base, and become a new ghost and go out of the home with the same moving pattern.

API Endpoints

GET /initialize Initialize the pac-man world.

POST /update Update the pac-man world.

GET /clear Clear the pac-man world.

POST /setGameParameters Set the game parameters.

Abstract classes

AObject

This class represents an object in the game.

AObject(String name, Point loc, String color)

* Constructor for the AObject class.
* @param loc The location of the object on the canvas.
* @param name The name of the object.

getColor()

* Get the ACharacter color.
* @return ACharacter color

setColor(String color)

* Set the ACharacter color.
* @param color The new ACharacter color.

getName()

* Get the object name.
* @return object name.

setName(String name)

* Set the object name.
* @param name The new object name.

getLoc()

* Get the object location.
* @return The object location.

setLoc(Point loc)

* Set the object location in canvas. The origin (0,0) is the top left corner.
* @param loc The object coordinate.

AItem

This class represents the unmoving items in the game (small dots, big dots, and fruits).

AItem(String name, Point loc, String color, int score)

* @param name The name of the AItem.
* @param loc The location of the AItem on the canvas.
* @param color The AItem color.

isEaten()

* Get if the AItem color is eaten.
* @return If AItem is eaten.

setEaten(boolean eaten)
* Set if the AItem color is eaten.
* @param eaten If AItem is eaten.

getScore()
* Get the AItem score.
* @return AItem score.

propertyChange(PropertyChangeEvent evt)
* Items respond to property change event.
* @param evt changed the event.

ACharacter

This class represents the moving characters in the game (ghost and pacman).

ACharacter(String name, Point loc, Point vel, String color, IUpdatePacmanStrategy updateStrategy, int direction, int size)
* Constructor.
* @param name The name of the ACharacter.
* @param loc The location of the ACharacter on the canvas.
* @param vel The ACharacter velocity.
* @param color The ACharacter color.
* @param updateStrategy The object updateStrategy.
* @param direction The character direction.

getUpdateStrategy()
* Get the ACharacter updateStrategy.
* @return The ACharacter updateStrategy.

setUpdateStrategy(IUpdatePacmanStrategy updateStrategy)
* Set the updateStrategy of the ACharacter.
* @param updateStrategy The new updateStrategy.

getVel()
* Get the velocity of the ACharacter.
* @return The ACharacter velocity.

setVel(Point vel)
* Set the velocity of the ACharacter.
* @param vel The new ACharacter velocity.

getDirection()
* Get the ACharacter direction.
* @return The ACharacter direction.

setDirection(int direction)
* Set the direction of the ACharacter.
* @param direction The new direction.

getOriginalLoc()
* Get the ACharacter original location.
* @return The ACharacter original location.

getSize()
* Get the size of the ACharacter.
* @return The ACharacter size.

setSize(int size)
* Set the size of the ACharacter.
* @param size The ACharacter size.

detectCollisionWithWalls(int direction, int[][] layout)
* Detects collision between an ACharacter and a wall in the ACharacter world. Change direction if ACharacter collides with a wall.
* @return if it collides with a wall within a step.

propertyChange(PropertyChangeEvent evt)
* Character responds to property change events.
* @param evt changed the event.

Interfaces

IUpdateGhostStrategy

The interface for updating the strategy of ghosts. The ghosts will use corresponding update strategies (chase or random walk) to update its behaviors in the game.

updateState(ACharacter pacman, ACharacter ghost)

* Update the state of the Ghost.
* @param pac-man The pacman character to compare to.
* @param ghost The ghost to apply the strategy to.

String getName()

* Get the name of the strategy.
* @return The strategy name.

IUpdatePacmanStrategy

The interface for update strategy of pacman. The pacman will use this update strategy to update its location in the game.

updateState(ACharacter pacman)

* Update the state of Pacman.
* @param pacman The pac-man to apply the strategy to.

getName()

* Get the name of the strategy.
* @return The strategy name.

IUpdatePacmanStrategy

The ICharacterCmd is an interface used to pass commands to characters in the Pacman. The character will execute the command when updating the state, interacting with another character or switch strategy (for ghosts).

execute(ACharacter context)

* Execute the command.
* @param context The receiver ACharacter on which the command is executed.

execute(ACharacter pacman, ACharacter context)

* Execute the command.
* @param context The ACharacter on which the command is executed.
* @param pac-man The pacman on which the command is executed.

Design Pattern

Command design pattern:

- Update state: update ghosts and pac-man state. The request is wrapped under an object as command and passed to the invoker object.
- Interact with items: detect collision with dots and fruits in the canvas. The request is wrapped under an object as command and passed to the invoker object.

Strategy design pattern:

- Ghost movement: ghosts have chase strategy and wander strategy.
- Collision: show the eat strategy of pac-man.
- Return home base: make dead ghosts go back to home base.

Factory design pattern:

- Strategy: ghosts have chase strategy and wander strategy. For these concrete classes, using factory design pattern can help to create objects.
- Cmd: decide which strategy to implement for both ghosts and pac-man to update state and the actions while collision. For these concrete classes, using factory design pattern can help to create objects.

Singleton design pattern:

- Each factory class uses make and only to create one instance to save spaces.