CMPT 276 - Assignment 3: Code Review Report

Michael Plunkett, mplunket@sfu.ca, 301396899
Salman Rafiei, salman_rafiei@sfu.ca, 301435217
Last updated: April 2, 2023, 8:46pm

Refactors:

Game Panel:

1) GamePanel::changeGameStates has repeated code
   - Package repeated code into new method: GamePanel::setupLevel
   - changeGameStates calls setupLevel, passing through the level number
   - Also automated the setting of keysNeeded
     - Used to be hardcoded for each level
     - Added numKeys variable to ItemFactory which is calculated when items are spawned
     - Now, in setupLevel, keysNeeded is set to numKeys (exactly the total amount of keys in the current level)
   - Commit address: 905fe63e1938263842042cdc2149117f7170e279

2) GamePanel::update has spaghetti code
   - Every tick of the game, update does a few things (given we are in playState):
     1. If the player is dead, end the game
     2. Drain the players health
     3. Attempt to spawn a potion
     4. Despawn expired potions
     5. Update the player
     6. Update all enemies
   - Instead of nesting tasks 2-6 inside of an else statement, i.e. if(dead){1} else{2-6}, if the player is dead we return immediately. Thus tasks 2-6 do not need to be nested within an else statement. This makes the method much more readable
   - Packaged the code which deals with draining the players health into a private function called drainHealth()
   - Packaged the code which deals with spawning potions into a private function called attemptSpawnPotion()
   - The other tasks are all simple and straightforward code which take only a few lines at most
   - Commit address: b94796ea578ffe85100dc0cf413dd785f5851e08

3) Refactor handling of gameStates and paused status
   - Change gameState and paused from public to private
   - Create methods getGameState(), isPaused(), and setPaused()
   - gameState already has changeGameState, which handles all changes to game state
   - Also create method inPlayState(), which returns true if gameState >= playState1 and gameState <= playState3

    - Since this is used in various places throughout the code (GamePanel, UI, and KeyInputs), having it packaged into one method is good coding practice and makes it much easier to add new levels in the future if need be.
- Also made some other minor refactors, such as changing consecutive if/else statements into one switch statement in UI::draw()
- Commit address: 98edfd2db7f8c5928a9700e0780bdf9fa760d064

KeyInputs:

KeyInputs instance variables are public

- Change upPressed, downPressed, leftPressed, rightPressed, and checkDrawTime to private
- Create getters for each and replace accesses with calls to the getter function in Player class
- Commit address: 9351299c764f8441f5424a06c9c4c49680b57298

Skeleton class

Instance variables have no access modifiers (all defaulted to public):

- Made all variables (ArrayList<String> path, int pathIndex, int nextX, and int nextY) private
- No getters or setters required as they are only used by the Skeleton class
- Commit address: 6d49da589c956e467edf1d58f8361528a9d6d7aa

Bat class

Bats change direction after collision with walls or other entities

- Previously, bats attempted to change directions every 50 ticks
- Added condition to Bat::setAction to check if its collision is on. If so, it will attempt to change directions immediately instead of waiting for the next 50 ticks to pass.
- Long delay between collision and changing direction removed
- Commit address: 808929d480bcd2790cd8ed9661fd9a70cbc21653

Items:

1) Item instance variables are all public:

- Change all variables to private
    - BufferedImage image,
    - String name
    - boolean collision
    - int worldX, worldY,
    - Rectangle hitBox
    - public int hitBoxDefaultX, hitBoxDefaultY,
- Create getter and setter methods for each variable in the Item class
    - Change name of the static method Key::getImage -> getSprite to allow for the creation of Item::getImage method as the getter method for image
    - This naming convention is also more intuitive and descriptive, as it follows suit with the setupSprite methods

- Change all accesses and modifications of these instance variables to call the respective getters and setters
    - Changes made to Door, Key, Potion, Spikes, CollisionChecker, and Player classes
- Commit address: 1e1672b4b7bcff88e632983255802555ff930eff

2) BonusReward instance variables are public
- Change birthTime and lifeTime to private
- Create getters for each and replace accesses with calls to the getter function in the GamePanel class
- Create protected setters (only subclasses should be able to set this variable) and implement them in the constructor of Potion
- Commit address: a487dd3c4ddfe81612a46cb2a8b940cc424cef11

<u>Fixing Potions</u>

Move the code which handles despawning of potions from the run() loop to the update() loop
- This is cleaner and more intuitive, as we are grouping all of the game state updates into the update() loop. This is the intended purpose of the update() loop.
- All edits were made in GamePanel
- Commit address: 7588aba63ac3a3072777308ddd67d5b40477a8a6