Assignment 3

Graded

Student

Longpeng Xu

Total Points

104 / 100 pts

Autograder Score
74.0 / 70.0

Failed Tests

Multiple crates (Advanced Events) (0/3.5)
Loading the game from a file (Filemenu) (0/2.5)

Passed Tests

play_game (Design) (2.5/2.5)
ExtraFancySokoban (Design) (0/0)
FancyGameView (Design) (0/0)
FancyStatsView (Design) (0/0)
Shop (Design) (0/0)
FancySokobanView (Design) (0/0)
ExtraFancySokoban (Design) (0/0)
Window Title (Basic Setup) (3/3)
Banner exists (Banner) (2/2)
Banner dimensions and positioning (Banner) (2/2)
FancyGameView exists (FancyGameView) (1/1)
Initial state (FancyGameView) (2/2)
Initial state maze 2 (FancyGameView) (2/2)
Initial state coin maze (FancyGameView) (2/2)
FancyStatsView exists (FancyStatsView) (1/1)
FancyStatsView is sized appropriately (FancyStatsView) (2/2)
FancyStatsView initial state (FancyStatsView) (3/3)
FancyStatsView draw_stats (FancyStatsView) (3/3)
FancyStatsView draw_stats 2 (FancyStatsView) (3/3)
Shop exists for maze 1 (Shop) (2.5/2.5)
Shop exists for maze 2 (Shop) (2.5/2.5)
FancySokobanView creates other views (FancySokobanView) (2/2)
Shop exist for maze 2 (FancySokobanView) (2/2)
Player can move (Basic Events) (3/3)
Player can move (Basic Events) (3/3)
Movement costs (Basic Events) (1.5/1.5)
Movement into wall (Basic Events) (2/2)
Pick up strength potion (Basic Events) (1.5/1.5)
Push crate (Basic Events) (2/2)
Push crate into wall (Basic Events) (2/2)
Push crate onto goal (Basic Events) (2/2)
Coin maze (Basic Events) (2/2)
Pick up coin (Basic Events) (2/2)
Buy strength potion (Basic Events) (2/2)
Buy fancy potion (Basic Events) (2/2)
Buy move potion (Basic Events) (2/2)
Pushing crate without strength (Basic Events) (2/2)
Filemenu is visible (Filemenu) (2.5/2.5)
Saving the game creates a file (Filemenu) (2.5/2.5)
Loading the game from a file (Filemenu) (2.5/2.5)

Question 2

Readability **6** / 6 pts

2.1 **Program Structure** **3** / 3 pts

✔ **+ 3 pts** All of the following criteria has been met:

- Vertical whitespace has been used appropriately to separate logical blocks of code.
- Horizontal whitespace has been used to avoid terse lines of code.
- There are no sections of code which create undue burden on the reader.
- All lines of code conform to PEP8 style rules such as a maximum line length of 80 characters.

+ 1.5 pts Most of the following criteria has been met:

- Vertical whitespace has been used appropriately to separate logical blocks of code.
- Horizontal whitespace has been used to avoid terse lines of code.
- There are no sections of code which create undue burden on the reader.
- All lines of code conform to PEP8 style rules such as a maximum line length of 80 characters.

+ 0 pts At least one of the following criteria has been majorly violated:

- Vertical whitespace has been used appropriately to separate logical blocks of code.
- Horizontal whitespace has been used to avoid terse lines of code.
- There are no sections of code which create undue burden on the reader.
- All lines of code conform to PEP8 style rules such as a maximum line length of 80 characters.

2.2 **Identifier Names** **3** / 3 pts

✔ **+ 3 pts** All of the following criteria has been met:

- All identifier names conform to the correct casing for python code.
- All non-counter variables have a meaningful name which describes the variable independent of its context.
- Variable types are not included in the name when the type does not describe the variable.

+ 1.5 pts Most of the following criteria has been met:

- All identifier names conform to the correct casing for python code.
- All non-counter variables have a meaningful name which describes the variable independent of its context.
- Variable types are not included in the name when the type does not describe the variable.

+ 0 pts At least one of the following criteria has been majorly violated.

- All identifier names conform to the correct casing for python code.
- All non-counter variables have a meaningful name which describes the variable independent of its context.
- Variable types are not included in the name when the type does not describe the variable.

Question 3

Documentation **6** / 6 pts

3.1 **Inline Comments** **3** / 3 pts

✔ **+ 3 pts** Inline comments are used to assist readability in all of the following cases:

- Single lines with complex logic.
- Blocks of code with a singular purpose which should be documented.

+ 1.5 pts Inline comments are used to assist readability in most of the following cases:

- Single lines with complex logic.
- Blocks of code with a singular purpose which should be documented.

+ 0 pts More than one case of missing inline comments in the following situations:

- Single lines with complex logic.
- Blocks of code with a singular purpose which should be documented.

  Or has needless inline comments.

3.2 **Informative Docstrings** **3** / 3 pts

✔ **+ 3 pts** All modules, classes, methods and functions are clearly and concisely described via informative and complete docstrings.

+ 1.5 pts Most modules, classes, methods and functions are clearly and concisely described via informative and complete docstrings.

+ 0 pts Few modules, classes, methods and functions are clearly described via informative and complete docstrings.

Question 4

Code Design **6** / 6 pts

4.1 **Single Instance of Logic** **3** / 3 pts

✔ **+ 3 pts** Almost no code has been duplicated in your program. You have well designed functions with appropriate parameters to modularise your code.

+ 1.5 pts Some code has been duplicated in your program. You have used some functions to modularise your code.

+ 0 pts Large amounts of code are duplicated in your program. You have made poor use of functions to modularise your code.

4.2 **Control Structures** **3** / 3 pts

✔ **+ 3 pts** Logic is structured simply and clearly through good use of control structures.

+ 1.5 pts A small number of control structures are unnecessarily complex.

+ 0 pts Many control structures are poorly designed (e.g. excessive nesting, overly complex conditional logic, loops with multiple unnecessary exit points, ...).

Question 5

Object-Oriented Program Structure **12** / 12 pts

5.1 Model View Controller **4** / 4 pts

✔ **+ 4 pts** The GUI's view and control logic is clearly separated from the model. Model information stored in the controller and passed to the view when required.

+ 2 pts A maximum of one of the following issues is present:

- View classes are storing too much modelling information.
- The view and model classes are in direct communication.
- Excessive model information is passed to update the view.
- Callbacks aren't used to setup GUI events.

+ 0 pts At least one of the following criteria has been majorly violated:

- View classes are storing too much modelling information.
- The view and model classes are in direct communication.
- Excessive model information is passed to update the view.
- Callbacks aren't used to setup GUI events.

5.2 Abstraction **3** / 3 pts

✔ **+ 3 pts** Public interfaces of classes are simple and reusable. Enabling modular and reusable components which abstract GUI details.

+ 1.5 pts The public interfaces of some classes are too rigid resulting in non-modular components.

+ 0 pts The public interfaces of most classes are too rigid resulting in non-modular components.

5.3 Encapsulation **3** / 3 pts

✔ **+ 3 pts** Classes are designed as independent modules with state and behaviour. Methods only directly access the state of the object on which they were invoked.

+ 1.5 pts Classes are almost always designed as independent modules with state and behaviour. At most once instance of breaking encapsulation is present.

+ 0 pts Multiple methods directly access or modify instance variables (or the state) of instances of another class.

5.4 Inheritance **2** / 2 pts

✔ **+ 2 pts** Subclasses extend the behaviour of their superclass without re-implementing behaviour, or breaking the superclass behaviour or design. Abstract classes have been used to effectively group shared behaviour amongst subclasses.

+ 1 pt Subclasses extend the behaviour of their superclass with only minor instances of re-implementing behaviour.

+ 0 pts Provided class interfaces have been modified in ways detrimental to the design, or there are several places in which behaviour has been re-implemented among subclasses.

Autograder Results

play_game (Design) (2.5/2.5)

Given the play_game function is defined
And play_game function takes 2 positional parameters

ExtraFancySokoban (Design) (0/0)

Given the ExtraFancySokoban class is defined
And ExtraFancySokoban.__init__ with 3 positional parameters is defined

FancyGameView (Design) (0/0)

Given the FancyGameView class is defined
And FancyGameView class inherits from tk.Canvas
And FancyGameView.__init__ with 4 positional parameters and 1 keyword parameters is defined
And FancyGameView.display with 4 positional parameters is defined

FancyStatsView (Design) (0/0)

Given the FancyStatsView class is defined
And FancyStatsView.__init__ with 2 positional parameters is defined
And FancyStatsView.draw_stats with 4 positional parameters is defined

Shop (Design) (0/0)

Given the Shop class is defined
And Shop class inherits from tk.Frame
And Shop.__init__ with 2 positional parameters is defined
And Shop.create_buyable_item with 4 positional parameters is defined

FancySokobanView (Design) (0/0)

Given the FancySokobanView class is defined
And FancySokobanView.__init__ with 4 positional parameters is defined
And FancySokobanView.display_game with 4 positional parameters is defined
And FancySokobanView.display_stats with 4 positional parameters is defined
And FancySokobanView.create_shop_items with 3 positional parameters is defined

ExtraFancySokoban (Design) (0/0)

Given the ExtraFancySokoban class is defined
And ExtraFancySokoban.__init__ with 3 positional parameters is defined
And ExtraFancySokoban.redraw with 1 positional parameters is defined
And ExtraFancySokoban.handle_keypress with 2 positional parameters is defined

Window Title (Basic Setup) (3/3)

Given I open the game, using maze 1
Then the window title is "Extra Fancy Sokoban"

Banner exists (Banner) (2/2)

Given I open the game, using maze 1
Then I see the banner

Banner dimensions and positioning (Banner) (2/2)

Given I open the game, using maze 1
Then I see the banner
And it is 75 pixels tall
And it is 650 pixels wide
And it is above all other widgets

FancyGameView exists (FancyGameView) (1/1)

Given I open the game, using maze 1
Then a FancyGameView instance should be packed within the GUI
And it is 450 pixels wide
And it is 450 pixels tall

Initial state (FancyGameView) (2/2)

Given I open the game, using maze 1
Then a FancyGameView instance should be packed within the GUI
Then the maze tile should contain
```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | | | |W|
|W|W|W|W|W|W|W|W|
```
And the maze crates should contain
```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
```
And the player should be at (1, 1), facing DOWN
And there are no potions
And there's no funny business

Initial state maze 2 (FancyGameView) (2/2)

Given I open the game, using maze 2
Then a FancyGameView instance should be packed within the GUI
Then the maze tile should contain
```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | | | |W|
|W|W|W|W|W|W|W|W|
```
And the maze crates should contain
```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
```
And the player should be at (1, 1), facing DOWN
And the maze potions should contain
```
| | | | | | | | |
| | | | | | | | |
| | |S| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | |M| | | | |
| | | | | | | | |
```
And there's no funny business

Initial state coin maze (FancyGameView) (2/2)

Given I open the game, using coin maze
Then the maze coins should contain
```
| | | | | | | | |
| | |$| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | |$| | |
| | | | | | | | |
| | | | | | | | |
```

FancyStatsView exists (FancyStatsView) (1/1)

Given I open the game, using maze 1
Then a FancyStatsView instance should be packed within the GUI
And the FancyStatsView should contain 0 buttons
And the FancyStatsView should contain 0 labels

FancyStatsView is sized appropriately (FancyStatsView) (2/2)

Given I open the game, using maze 1
Then a FancyStatsView instance should be packed within the GUI
And it is 650 pixels wide
And it is 75 pixels tall

FancyStatsView initial state (FancyStatsView) (3/3)

Given I open the game, using maze 1
Then an FancyStatsView instance should be packed within the GUI
And the FancyStatsView should have the appropriate heading text
And the FancyStatsView should have the appropriate heading font
And I have 12 moves remaining
And I have 1 strength remaining
And I have $0

FancyStatsView draw_stats (FancyStatsView) (3/3)

Given I open the game, using maze 1
Then an FancyStatsView instance should be packed within the GUI
When I redraw stats on FancyStatsView with (10 moves, 5 strength, $5)
Then I have 10 moves remaining
And I have 5 strength remaining
And I have $5

FancyStatsView draw_stats 2 (FancyStatsView) (3/3)

Given I open the game, using maze 2
Then an FancyStatsView instance should be packed within the GUI
When I redraw stats on FancyStatsView with (14 moves, 10 strength, $10)
Then I have 14 moves remaining
And I have 10 strength remaining
And I have $10

Shop exists for maze 1 (Shop) (2.5/2.5)

Given I open the game, using maze 1
Then 1 Shop instances should be packed within the GUI
And Shop correctly display title
And all Shop Frames correctly display the potion name
And I have 1 "Strength Potion"
And I have 1 "Move Potion"
And I have 1 "Fancy Potion"
And Shop have 3 buttons
And Shop have the correct buttons text

Shop exists for maze 2 (Shop) (2.5/2.5)

Given I open the game, using maze 2
Then 1 Shop instances should be packed within the GUI
And Shop correctly display title
And all Shop Frames correctly display the potion name
And I have 1 "Strength Potion"
And I have 1 "Move Potion"
And I have 1 "Fancy Potion"
And Shop have 3 buttons
And Shop have the correct buttons text

FancySokobanView creates other views (FancySokobanView) (2/2)

Given I open the game, using maze 1
Then I see the banner
And the window title is "Extra Fancy Sokoban"
And a FancyStatsView instance should be packed within the GUI
And 1 Shop instances should be packed within the GUI
And a FancyGameView instance should be packed within the GUI

Shop exist for maze 2 (FancySokobanView) (2/2)

Given I open the game, using maze 2
Then 1 Shop instances should be packed within the GUI
And Shop correctly display title
And I have 1 "Strength Potion"
And I have 1 "Move Potion"
And I have 1 "Fancy Potion"
And Shop have 3 buttons
And Shop have the correct buttons text

Player can move (Basic Events) (3/3)

Given I open the game, using maze 1
Then the player should be at (1, 1), facing DOWN
When I press d
Then the player should be at (1, 2), facing DOWN

Player can move (Basic Events) (3/3)

Given I open the game, using maze 1
Then the player should be at (1, 1), facing DOWN
When I press d
Then the player should be at (1, 2), facing DOWN
When I press s
Then the player should be at (2, 2), facing DOWN
When I press a
Then the player should be at (2, 1), facing DOWN
When I press w
Then the player should be at (1, 1), facing DOWN

Movement costs (Basic Events) (1.5/1.5)

Given I open the game, using maze 1
Then I have 12 moves remaining
When I press d
Then I have 11 moves remaining

Movement into wall (Basic Events) (2/2)

Given I open the game, using maze 1
When I press the a key, 3 times
Then the player should be at (1, 1), facing DOWN
And I have 12 moves remaining
When I press the s key, 20 times
Then the player should be at (5, 1), facing DOWN
And I have 8 moves remaining

Pick up strength potion (Basic Events) (1.5/1.5)

Given I open the game, using maze 2
Then the player should be at (1, 1), facing DOWN
And I have 1 strength remaining
When I press d
Then the player should be at (1, 2), facing DOWN
When I press s
Then the player should be at (2, 2), facing DOWN
And I have 3 strength remaining

Push crate (Basic Events) (2/2)

Given I open the game, using maze 1
When I press d
Then the player should be at (1, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
```

When I press the s key, 2 times
Then the player should be at (3, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
```

Push crate into wall (Basic Events) (2/2)

Given I open the game, using maze 1
When I press d
Then the player should be at (1, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
```

When I press the s key, 2 times
Then the player should be at (3, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
```

When I press the s key, 20 times
Then the player should be at (4, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
```

Push crate onto goal (Basic Events) (2/2)

Given I open the game, using maze 1
When I press d
When I press the s key, 2 times
When I press a
When I press s
When I press the d key, 3 times
Then the player should be at (4, 4), facing DOWN
When I press s
When I press d
Then the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | |C| | |
| | | | | | | | |
| | | | | | | | |
```

And the maze tile should contain

```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | | | |W|
|W|W|W|W|W|W|W|W|
```

When I press w
Then a messagebox should be displayed
And the messagebox should say "You won! Play again?"

Coin maze (Basic Events) (2/2)

Given I open the game, using coin maze
Then the maze coins should contain

```
| | | | | | | | |
| | |$| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | |$| | |
| | | | | | | | |
| | | | | | | | |
```

And I have $0

Pick up coin (Basic Events) (2/2)

Given I open the game, using coin maze
Then I have $0
When I press d
Then the player should be at (1, 2), facing DOWN
And the maze coins should contain
```
  | | | | | | | | |
  | | | | | | | | |
  | | | | | | | | |
  | | | | | | | | |
  | | | | | |$| | |
  | | | | | | | | |
  | | | | | | | | |
```
And I have $5

Buy strength potion (Basic Events) (2/2)

Given I open the game, using coin maze
Then I have 1 strength remaining
And 1 Shop instances should be packed within the GUI
When I buy 1 Strength Potion
Then I have 1 strength remaining
When I press d
Then I have $5
When I buy 1 Strength Potion
Then I have 3 strength remaining
And I have $0

Buy fancy potion (Basic Events) (2/2)

Given I open the game, using coin maze
Then I have 1 strength remaining
And I have 10 moves remaining
When I buy 1 Strength Potion
Then I have 1 strength remaining
When I buy 1 Move Potion
Then I have 10 moves remaining
When I buy 1 Fancy Potion
Then I have 1 strength remaining
And I have 10 moves remaining
When I press d
Then I have $5
When I press d
When I press the s key, 3 times
When I press the d key, 2 times
Then I have $10
Then I have 1 strength remaining
And I have 3 moves remaining
When I buy 1 Fancy Potion
Then I have 3 strength remaining
And I have 5 moves remaining
And I have $0

Buy move potion (Basic Events) (2/2)

Given I open the game, using coin maze
Then I have 1 strength remaining
And I have 10 moves remaining
When I buy 1 Strength Potion
Then I have 1 strength remaining
When I buy 1 Move Potion
Then I have 10 moves remaining
When I buy 1 Fancy Potion
Then I have 1 strength remaining
And I have 10 moves remaining
When I press d
Then I have $5
When I press d
When I press the s key, 3 times
When I press the d key, 2 times
Then I have $10
Then I have 1 strength remaining
And I have 3 moves remaining
When I buy 1 Move Potion
Then I have 1 strength remaining
And I have 8 moves remaining
And I have $5

Pushing crate without strength (Basic Events) (2/2)

Given I open the game, using coin maze
When I press d
Then I have $5
When I press a
When I press the s key, 2 times
When I press the d key, 20 times
Then I have 6 moves remaining
And the player should be at (3, 1), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
```

When I press w
When I press d
When I press s
Then the player should be at (3, 2), facing DOWN
And the maze crates should contain

```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | | | | | | | |
```

Multiple crates (Advanced Events) (0/3.5)

Given I open the game, using maze 3
When I press d
When I press s
When I press a
When I press the s key, 3 times
When I press the d key, 2 times
Then the maze crates should contain
```
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | |C| | | | |
| | | | | | | |
| | | | |C| | | |
| | | | | | | |
```
And the maze tile should contain
```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |G| |W|
|W|W|W|W|W|W|W|W|
```
When I press d
Then the maze crates should contain
```
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | |C| | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
```
And the maze tile should contain
```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |X| |W|
|W|W|W|W|W|W|W|W|
```
Assertion Failed: Sokoban tile contains:

```
# |W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |G| |W|
|W|W|W|W|W|W|W|W|
```

```
# Expected:

# |W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |X| |W|
|W|W|W|W|W|W|W|W|

#
```

Filemenu is visible (Filemenu) (2.5/2.5)

Given I open the game, using maze 1
Then the file menu is displayed
And I can see a Save menu option
And I can see a Load menu option

Saving the game creates a file (Filemenu) (2.5/2.5)

Given I open the game, using maze 1
When I get prompted I will say "save1.txt"
When I select the Save menu option
Then a file named "save1.txt" should exist

Loading the game from a file (Filemenu) (2.5/2.5)

Given I open the game, using maze 3
When I get prompted I will say "save.txt"
Then the player should be at (1, 1), facing DOWN
And the maze crates should contain
```
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
| | |C| | | | | |
| | | | | | | | |
```
And the maze tile should contain
```
|W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |G| |W|
|W|W|W|W|W|W|W|W|
```
When I select the Save menu option
Then a file named "save.txt" should exist
When I press d
When I press s
When I select the load menu option
Then the player should be at (1, 1), facing DOWN

Loading the game from a file (Filemenu) (0/2.5)

```
Given I open the game, using maze 3
When I get prompted I will say "save.txt"
Then the player should be at (1, 1), facing DOWN
And the maze crates should contain
  | | | | | | | |
  | | | | | | | |
  | | | | | | | |
  | | |C| | | | |
  | | | | | | | |
  | | |C| | | | |
  | | | | | | | |
And the maze tile should contain
  |W|W|W|W|W|W|W|W|
  |W| | | |W| | |W|
  |W| | | |W| | |W|
  |W| | | |W|G| |W|
  |W| | | | | | |W|
  |W| | | | |G| |W|
  |W|W|W|W|W|W|W|W|
When I select the Save menu option
Then a file named "save.txt" should exist
When I press d
When I press s
When I press a
When I press the s key, 3 times
When I press the d key, 3 times
Then the maze crates should contain
  | | | | | | | |
  | | | | | | | |
  | | | | | | | |
  | | |C| | | | |
  | | | | | | | |
  | | | | | | | |
  | | | | | | | |
And the maze tile should contain
  |W|W|W|W|W|W|W|W|
  |W| | | |W| | |W|
  |W| | | |W| | |W|
  |W| | | |W|G| |W|
  |W| | | | | | |W|
  |W| | | | |X| |W|
  |W|W|W|W|W|W|W|W|
   Assertion Failed: Sokoban tile contains:

# |W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
```

```
|W| | | | | |W|
|W| | | |G| |W|
|W|W|W|W|W|W|W|


# Expected:

# |W|W|W|W|W|W|W|W|
|W| | | |W| | |W|
|W| | | |W| | |W|
|W| | | |W|G| |W|
|W| | | | | | |W|
|W| | | | |X| |W|
|W|W|W|W|W|W|W|W|

#
```

Submitted Files

a3.py 1 Comment Download

```
import tkinter as tk
from tkinter import messagebox, filedialog
from typing import Callable, Union
from model import *
from a2_support import *
from a3_support import *



class FancyGameView(AbstractGrid):
    """ A grid displaying the game map, incl. all tiles, entities and player.
        Inherits from AbstractGrid
    """
    IMAGES = {
        WALL: 'images/W.png',
        FLOOR: 'images/Floor.png',
        GOAL: 'images/G.png',
        CRATE: 'images/C.png',
        PLAYER: 'images/P.png',
        STRENGTH_POTION: 'images/S.png',
        MOVE_POTION: 'images/M.png',
        FANCY_POTION: 'images/F.png',
        COIN: 'images/$.png'}

    def __init__(self, master: tk.Frame | tk.Tk, dimensions: tuple[int, int],
                 size: tuple[int, int], **kwargs) -> None:
        """ Initialize FancyGameView. Set up appropriate dimensions, size, and
            an empty dict as the image cache

        Inputs:
            master: The master frame of the game, tk.Frame | tk.Tk
            dimensions: Dim of the game grid as # rows and # columns,
                tuple[int, int]
            size: width and height in pixels for the game grid, tuple[int, int]
        """
        super().__init__(master, dimensions, size=(MAZE_SIZE, MAZE_SIZE),
            **kwargs)
        self._cache = dict()

    def display(self, maze: Grid, entities: Entities, player_position:
                Position) -> None:
        """ Display the grid of the game map

        Inputs:
            maze: A grid with only tiles on it, Grid
            entities: Entities to be placed on the grid, Entities
```

```
            player_position: Position of the player, Position
        """
        self.clear()
        cell_size = self.get_cell_size()

        # Tiles
        for row in range(self._dimensions[0]):
            for col in range(self._dimensions[1]):
                image = get_image(self.IMAGES[maze[row][col].get_type()],
                    cell_size, self._cache)
                self.create_image(self.get_midpoint((row, col)), image=image)

        # Entities
        for key in entities.keys():
            image = get_image(self.IMAGES[entities[key].get_type()], cell_size,
                self._cache)
            self.create_image(self.get_midpoint(key), image=image)

        # Player
        image = get_image(self.IMAGES[PLAYER], cell_size, self._cache)
        self.create_image(self.get_midpoint(player_position), image=image)

    def reset_cache(self) -> None:
        """ Reset the cache for FancyGameView
        """
        self._cache = dict()



class FancyStatsView(AbstractGrid):
    """ A grid displaying the stats of the player, incl. moves remaining,
        strength, and money. Inherits from AbstractGrid
    """
    def __init__(self, master: tk.Tk | tk.Frame) -> None:
        """ Initialize FancyStatsView. Set up appropriate dimensions and size

        Inputs:
            master: The master frame of the game, tk.Frame | tk.Tk
        """
        super().__init__(
            master,
            dimensions=(3, 3),
            size=(MAZE_SIZE + SHOP_WIDTH, STATS_HEIGHT))

    def draw_stats(self, moves_remaining: int, strength: int, money: int) \
        -> None:
        """ Display the grid of the player's stats

        Inputs:
```

```
96          moves_remaining: # moves remains for player, int
97          strength: # units of strength remains for player, int
98          money: # units of money remains for player, int
99      """
100     self.clear()
101
102     # Annotate cells
103     self.annotate_position((0, 1), 'Player Stats',
104         font=('Arial', 18, 'bold'))
105     self.annotate_position((1, 0), 'Moves remaining:', font='TkDefaultFont')
106     self.annotate_position((1, 1), 'Strength:', font='TkDefaultFont')
107     self.annotate_position((1, 2), 'Money:', font='TkDefaultFont')
108     self.annotate_position((2, 0), f"{moves_remaining}",
109         font='TkDefaultFont')
110     self.annotate_position((2, 1), f"{strength}", font='TkDefaultFont')
111     self.annotate_position((2, 2), f"${money}", font='TkDefaultFont')
112
113
114
115 class Shop(tk.Frame):
116     """ A frame displaying a shop where player can buy potions, incl. strength
117         potion, move potion, and fancy potion. Inherits from tk.Frame
118     """
119     def __init__(self, master: tk.Frame) -> None:
120         """ Initialize Shop. Set up the shop to act as tk.Frame and to have a
121             title label
122
123         Inputs:
124             master: The parent class for the shop frame, tk.Frame
125         """
126         super().__init__(master)
127         label = tk.Label(self, text="Shop", font=('Arial', 18, 'bold'))
128         label.pack(side=tk.TOP)
129
130     def create_buyable_item(self, item: str, amount: int, callback:
131                             Callable[[], None]) -> None:
132         """ List a buyable item in the frame of shop, incl. a label and a button
133
134         Inputs:
135             item: a global constant for the string repr of a potion, str
136             amount: the amount of money required to buy that potion, int
137             callback: a function-like callable where its arguments can be told
138                 somewhere else
139         """
140         # One frame for one specific potion
141         item_frame = tk.Frame(self)
142         item_frame.pack(side=tk.TOP, fill=tk.X)
143
144         item_names = {
```

```
            STRENGTH_POTION: "Strength Potion",
            MOVE_POTION: "Move Potion",
            FANCY_POTION: "Fancy Potion"}
        name = item_names.get(item, "")

        tk.Label(item_frame, text=f"{name}: ${amount}", font='TkDefaultFont')\
            .pack(side=tk.LEFT)
        tk.Button(item_frame, text="Buy", command=callback).pack(side=tk.RIGHT)



class FancySokobanView:
    """ View of the game, wrapping the smaller GUI widgets incl.
        FancyGameView, FancyStatsView, and Shop
    """
    def __init__(self, master: tk.Tk, dimensions: tuple[int, int],
                 size: tuple[int, int]) -> None:
        """ Initialize FancySokobanView. Create a title banner, set the title on
            the window, and instantiate and pack the three widgets

        Inputs:
            master: The master frame of the game, tk.Frame | tk.Tk
            dimensions: Dim of the game grid as # rows and # columns,
                tuple[int, int]
            size: width and height in pixels for the game grid, tuple[int, int]
        """
        # Instantiate the three widgets
        self._fancy_game_view = FancyGameView(master, dimensions, size)
        self._fancy_stats_view = FancyStatsView(master)
        self._shop = Shop(master)
        self._cache = dict()

        # Create and pack the title banner
        banner_width = MAZE_SIZE + SHOP_WIDTH
        banner = get_image(
            'images/banner.png', (banner_width, BANNER_HEIGHT), self._cache)
        banner_label = tk.Label(master, image=banner)
        banner_label.pack(side=tk.TOP)

        # Pack the three widgets
        self._fancy_stats_view.pack(side=tk.BOTTOM)
        self._fancy_game_view.pack(side=tk.LEFT)
        self._shop.pack(side=tk.TOP)

        master.title("Extra Fancy Sokoban")

    def display_game(self, maze: Grid, entities: Entities,
                     player_position: Position) -> None:
        """ Display the game grid
```

```
194
195         Inputs:
196             maze: A grid with only tiles on it, Grid
197             entities: Entities to be placed on the grid, Entities
198             player_position: Position of the player, Position
199         """
200         self._fancy_game_view.display(maze, entities, player_position)
201
202     def display_stats(self, moves: int, strength: int, money: int) -> None:
203         """ Display the stats grid
204
205         Inputs:
206             moves: # moves remains for player, int
207             strength: # units of strength remains for player, int
208             money: # units of money remains for player, int
209         """
210         self._fancy_stats_view.draw_stats(moves, strength, money)
211
212     def create_shop_items(self, shop_items: dict[str, int], button_callback:
213                           Callable[[str], None] | None = None) -> None:
214         """ Create all the buyable items in the shop, where a lambda function
215             callback, which calls button_callback, is given to
216             create_buyable_item in Shop
217
218         Inputs:
219             shop_items: Maps item's string repr to price, dict[str, int]
220             button_callback: A callable on an item (key) in shop_items,
221                 Callable[[str], None] | None = None
222         """
223         for item, amount in shop_items.items():
224             # A lambda function calling button_callback on item
225             if button_callback:
226                 callback = lambda item=item: button_callback(item)
227                 self._shop.create_buyable_item(item, amount, callback)
228
229             else:
230                 self._shop.create_buyable_item(item, amount, None)
231
232     def reset_view(self, new_dims: tuple[int, int]) -> None:
233         """ Reset the cache and the dimensions of FancyGameView
234
235         Inputs:
236             new_dims: the updated dimensions of FancyGameView, tuple[int, int]
237         """
238         self._fancy_game_view.reset_cache()
239         self._fancy_game_view.set_dimensions(new_dims)
240
241
242
```

```
class ExtraFancySokoban:
    """ Controller of the game, which creates, maintains and communicates the
        instances of the model and the view classes
    """
    def __init__(self, root: tk.Tk, maze_file: str) -> None:
        """ Initialize ExtraFancySokoban. Create instances of SokobanModel and
            FancySokobanView. Create shop items. Bind keypress events to the
            relevant handler. Redraw the display

        Inputs:
            root: The master frame of the game, tk.Tk
            maze_file: The directory of a raw maze txt file, str
        """
        # Model of the game
        self._sokoban_model = SokobanModel(maze_file)

        # View of the game
        self._root = root
        dimensions = self._sokoban_model.get_dimensions()
        size = (MAZE_SIZE+SHOP_WIDTH, BANNER_HEIGHT+MAZE_SIZE+STATS_HEIGHT)
        self._sokoban_view = FancySokobanView(self._root, dimensions, size)

        shop_items = self._sokoban_model.get_shop_items()
        self._sokoban_view.create_shop_items(shop_items, self.buy_effect)

        self._root.bind("<KeyPress>", self.handle_keypress)

        self.redraw()

    def buy_effect(self, item: str) -> None:
        """ Helper function to be passed to create_shop_item when creating shop
            items in __init__
        """
        self._sokoban_model.attempt_purchase(item)
        self.redraw()

    def redraw(self) -> None:
        """ Redraw the game view and stats view based on the current model state
        """
        self._sokoban_view.display_game(
            self._sokoban_model.get_maze(),
            self._sokoban_model.get_entities(),
            self._sokoban_model.get_player_position()
        )
        self._sokoban_view.display_stats(
            self._sokoban_model.get_player_moves_remaining(),
            self._sokoban_model.get_player_strength(),
            self._sokoban_model.get_player_money()
        )
```

```
292
293     def handle_msgbox(self, msg: str) -> None:
294         """ Handle the game replay behavior by a messagebox when a win or a lost
295             is in place. If True, the game will be reset; if False, the program
296             will terminate
297
298         Inputs:
299             msg: The binary message indicating win or loss of the game, str
300         """
301         msg_box = messagebox.askyesno(title=None, message=msg)
302
303         if msg_box == True:
304             self._sokoban_model.reset()
305             self.redraw()
306         else:
307             self._root.destroy()
308
309     def handle_keypress(self, event: tk.Event) -> None:
310         """ A keypress event handler. When a keypress event occurs, the model
311             attempts move as per the event, and the view is redrawn. If a game
312             is won or lost, ask player if he/she will replay it
313
314         Inputs:
315             event: A keypress event, tk.Event
316         """
317         self._sokoban_model.attempt_move(event.char)
318         self.redraw()
319
320         # Message box after win or lost
321         if self._sokoban_model.has_won() == True:
322             self.handle_msgbox("You won! Play again?")
323         elif self._sokoban_model.has_won() == False \
```

Instructor | 11/08 at 10:56 pm

> control structures: has_won returns a boolean value, so it doesn't make sense to test if it's true or false. This could be, for instance, "elif not ...has_won() and ..."

```
324             and self._sokoban_model.get_player_moves_remaining() <= 0:
325             self.handle_msgbox("You lost! Play again?")
326
327     def save_file(self) -> None:
328         """ Save the current game state incl. tiles and entities on maze, and
329             player's strength and moves remaining, to a txt file
330         """
331         filepath = filedialog.asksaveasfilename(
332             defaultextension=".txt",
333             filetypes=[("Text files", "*.txt")])
```

```
334
335         # If a filename is provided
336         if filepath:
337             with open(filepath, 'w') as file:
338
339                 # Write player stats to txt
340                 file.write(
341                     f"{self._sokoban_model.get_player_strength()}" + " " +
342                     f"{self._sokoban_model.get_player_moves_remaining()}\n")
343
344                 # Write entities, player, tiles to txt
345                 maze = self._sokoban_model.get_maze()
346                 maze_rows, maze_cols = self._sokoban_model.get_dimensions()
347                 entities = self._sokoban_model.get_entities()
348                 for i in range(maze_rows):
349                     for j in range(maze_cols):
350                         if (i,j) in entities.keys():
351                             file.write(str(entities[(i,j)]))
352                         elif (i,j) == self._sokoban_model.get_player_position():
353                             file.write(PLAYER)
354                         else:
355                             file.write(str(maze[i][j]))
356                     file.write("\n")
357
358     def read_file(self) -> None:
359         """ Read and restore a saved game state from a txt file, formatted as
360             the line 0 of two integers (strength, moves remaining), and a
361             maze from line 1 (tiles, entities, player position)
362         """
363         filename = filedialog.askopenfilename(
364                 title="Select file",
365                 filetypes=[("Text files", "*.txt")],
366                 defaultextension=".txt")
367
368         # All info from txt file are handled by SokobanModel
369         self._sokoban_model = SokobanModel(filename)
370
371         dimensions = self._sokoban_model.get_dimensions()
372         self._sokoban_view.reset_view(dimensions)
373
374         self.redraw()
375
376
377
378 def play_game(root: tk.Tk, maze_file: str) -> None:
379     """ Construct the controller instance, set up the file menu for saving and
380         reading current game state, and ensure the root window stays listening
381         for events
382
```

```
    Inputs:
        root: The master frame of the game, tk.Tk
        maze_file: The directory of a raw maze txt file, str
    """
    controller = ExtraFancySokoban(root, maze_file)

    # File menu
    menu = tk.Menu(root)
    root.config(menu=menu)
    file_menu = tk.Menu(menu)
    menu.add_cascade(label="File", menu=file_menu)
    file_menu.add_command(label="Save", command=controller.save_file)
    file_menu.add_command(label="Load", command=controller.read_file)

    controller._root.mainloop()



def main() -> None:
    """ Set up the root of the game and play the game
    """
    root = tk.Tk()
    play_game(root, 'maze_files/coin_maze.txt')



if __name__ == "__main__":
    main()
```