Made using: python 3.9.6 and pygame 2.0.2

Setup:

-Add images/music to the assets folder (inside project).

-In your project folder (e.g. demo), add new chapter directories with your scripts to scene_list.py. Scene classes require an __init__() with 3 variables, and run().

-To change the names of the project folder, the starting chapter, or the starting scene, edit the variables in main.py.

-Use the configuration file for various general settings, as well as setting character images and map coordinates (to get user input).

Script Command Options:

* Background
* Dialog Box
* Narration Box
* Menu
* Portraits/Images
* Map
* Change Scenes
* Save game
* Music, Sound Effects, Shake Screen, Pause, variables
* Coordinator
* Scene transitions

Backgrounds:

-Background images are scaled to: 1130 x 700 (30px width remains offscreen, and is used for shake_screen)

Set the background in the __init__:

```
self.background = 'background_image.jpg'
```

-If self.background = None, then a solid color is used.
-You can also set the background image or color in the __init__.

Add text on background:

```
self.game.display_bg_text(text, x, y, color='White', font='Georgia', size=75, bold=0)
```

-The x, y can be the int pixel coordinates, or a percent of screen size as a str: "40%". You can also use 'center' for the x coordinate.

Remove the background text:

```
self.game.clear_bg_text()
```

Dialog Box:

-Dialog box takes a list of quotes. Each quote is a list of 3-4 items: the character name, character image index, quote-string, and (optional) choices. Choices must be in the last quote:

```
self.game.display_dialog([
    ['name', 0, 'quote'],
    ['name', 0, 'quote', ['Yes', 'No']]
])
```

-To disable certain choices, add a set_disabled argument with a list of booleans of the same length as the choices list:

```
self.game.display_dialog([['Name', 0, 'quote', ["Yes", "No", "Maybe"]]], set_disabled=[True, True, False])
```

Color:

-Text color escape code (for a single quote only):

Add '<RED>' to the start of the quote. (Use any pygame color names)

-Text color (dialog box setting):

```
self.game.dialog_boxes['default'].color = 'Red'
```

Note: any pygame color can be used, or RGB value (255,0,0)

-Background color (dialog box setting):

```
self.game.dialog_boxes['default'].surface.background_color = 'Black'
```
(or: (0,0,0))

Move the dialog box to top or bottom of screen:

```
self.game.dialog_box.move_dialog_up()
self.game.dialog_box.move_dialog_down()
```

-These can also take optional arguments x and y for placement other than the default set in configuration.py.

Prevent automatic removal of dialog box:

```
self.game.display_dialog(quotes, remove=False)
self.game.dialog_box.remove_dialog_box()
```

-If you want no name tag, use an empty string '', same with image and quote (latter of which is useful with animations by keeping it up with remove=False):

```
self.game.display_dialog([['', '', '']], remove=False)
```

Advanced Dialog Box options:

-Change settings of dialog box in the configuration file. Use the below options to change mid script or the box dimensions.

Font:

```
self.game.dialog_boxes['default'].change_font(
    font_name=None,
    text_size=None,
    color=None,
    name_tag=False)
```

-To change for the nametag, use: name_tag=True

Toggle typing animation:

```
self.game.dialog_boxes['default'].gradual_typing = False
```

Typing speed:

```
self.game.dialog_boxes['default'].typing_speed = 15
```

-This is the default in milliseconds. A higher number will increase the delay time between characters.

To change the size, location, or background color of the box and choices menu, use any of the optional arguments:
-Note: The first 4 arguments can take either an integer or percent (of screen size) as a string, ex: "10.5%"

```
self.game.dialog_boxes['default'].config_surface(
        x=None,
        y=None,
        width=None,
        height=None,
        background_color=None,
        border_color=None,
        border_width=None,
        transparency=None,
        choice_menu_x=None,
        choice_menu_y=None,
        txt_wrap='auto',
        max_lines='auto',
        x_txt_padding=None,
        y_txt_padding=None,
        image_size=None,
        image_border_width=None
)
```

To change the name tag, add 'name_tag=True' and any of the other optional arguments:

```
self.game.dialog_boxes['default'].config_surface(
        x=None,
        y=None,
        background_color=None,
        border_color=None,
        border_width=None,
        name_tag=True
)
```

Note: width and height of the name tag are set automatically according to the size of the text.

To create and switch to a new dialog box:

```
self.game.create_dialog(
        'new_name',
        x=None,
        y=None,
        width=None,
        height=None,
        background_color=None,
```

```
            border_color=None,
            border_width=None,
            transparency=None,
            name_tag_x=None,
            name_tag_y=None,
            choice_menu_x=None,
            choice_menu_y=None
    )

    self.game.dialog_boxes['new_name'].change_font(
            font_name=None,
            text_size=None,
            color=None,
            name_tag=False)

    self.game.switch_dialog('new_name')
```

-'new_name' becomes new default. Use self.game.switch_dialog('default') to switch back.
-Default options: 'default' and 'default2' (A transparent version)

Narration Box:

Narration Boxes take a string of any length. Long text will continue on new pages.

```
    self.game.display_narration(
            text,
            choices=[],
            remove=True,
            set_disabled=[]
    )
```

-A newline in the text will automatically indent a new paragraph. See configuration.py to change indent length.
-'text' can be either a string, or a list of strings. A list will start a new page for each item, regardless of length.
-To disable certain choices, add an optional set_disabled argument with a list of booleans of the same length as choices.

To get user input, simply use a choices argument:

```
    self.game.display_narration(text, choices=["Yes", "No"])
```

The optional 'remove=True' will keep the Narration box with text up after the user clicks to continue.
To remove it, display another narration box or use:

```
    self.game.narration_box.remove_narration_box()
```

Scrolling text:

```
self.game.display_narration_scroll(
    text,
    speed=1,
    speedup=True,
    y_offset=0,
    font_name=None,
    size=35,
    color=None,
    bold=False
)
```

-speed can be a float. Useful to reduce the speed below 1.
-speedup toggles fast forward option.
-y_offset adjusts the start and vanish points for the text lines.
-font_name and color will use the current Narration Box settings if None given.

Color:
-Text color escape code (single paragraph only):

Add "<Red>" to the start of the paragraph. (Use any pygame color names)

-Text color (all):

```
self.game.narration_box.color = 'Red'
```

Note: Any pygame color can be used, or RGB value: (255,0,0)

-Background color (all):

```
self.game.narration_box.surface.background_color = 'black'
```
(or RGB: (0,0,0))

Advanced:

-Change settings for Narration Box in the configuration file: To change mid script or box dimensions, use the below options:

Toggle typing animation:

```
self.game.narration_box.gradual_typing = False
```

Typing speed:

```
self.game.narration_box.typing_speed = 2
```

-This is the default in milliseconds, a higher number will increase the delay between characters.

To change the font:

```
self.game.narration_box.change_font(font_name=None, text_size=None, color=None)
```

To change the size, placement, and background colors, fill in any of the optional arguments:

```
self.game.narration_box.config_surface(
        x=None,
        y=None,
        width=None,
        height=None,
        background_color=None,
        border_color=None,
        border_width=None,
        transparency = None,
        max_lines=None,
        txt_wrap=None,
        x_txt_padding=None,
        y_txt_padding=None
)
```

-Can use int or percent (of screen size) ex: "10.5%"

Menus

Create a menu:

```
self.game.create_menu(
        name,
        items,
        x,
        y,
        color="Black",
        size=50,
        spacing=10,
        align='left',
        bold=0,
        font='georgia'
)
```

-items example with three options: [["First", True], ["Second", True], ["Third", True]]
-Replacing True with False will show the item but disabled so it cannot be selected.

To add a background box behind menu text:

```
self.game.menus['menu_name'].add_bg(
    padding=20,
    bg_color='white',
    border_color='Grey',
    border_width=0,
    transparency=255
)
```

Display menu:

```
self.game.menus['menu_name'].show_menu(remove=True, default=0)
```

-remove=False will keep menu up (unusable as a background) after a selection has been made. Useful with transitions.
-default is the index for which item is highlighted. -1 for none, 0 is first.

Remove the menu (after displayed with remove=False):

```
self.game.menus['menu_name'].remove_menu()
```

To enable or disable a menu item using the index number:

```
self.game.menus['menu_name'].disable_enable_menu_item(0)
```

Move in from sides (animation):

```
self.game.menus['menu_name'].move_in_left(remove=True, default=0)
self.game.menus['menu_name'].move_in_right(remove=True, default=0)
```

Portraits/Images

Add an image to screen:

```
self.game.portraits['example'].blit_image(self, x=0, y=0, dialog='left')
```

-x can be: 'right' 'center' 'left' or int
-y can be: 'bottom' or 'top' or int
-dialog can be: 'left' 'right' or 'center' (automatic x y placement above dialog box)

Animations:

```
self.game.portraits['example'].move_in_left(y='default')
self.game.portraits['example'].move_in_right(y='default')
self.game.portraits['example'].move_out_left()
self.game.portraits['example'].move_out_right()
```

Change size:

```
self.game.portraits['example'].upscale(-20)  # 20% decrease in size
```

Remove image:

```
self.game.portraits['example'].remove()
```

Maps

Display a map:

```
self.game.display_map('map_name')  -> 'name of location'
```

-input returned is stored in self.game.input_return

No input, reset the previously chosen location, or set a starting location:

```
self.game.display_map('map_name', display_only=True, reset_loc=True, set_loc='loc_name')
```

Show/hide a location during script:

```
self.game.show_map_loc('map_name', 'loc_name')
self.game.hide_map_loc('map_name', 'loc_name')
```

Remove map:

```
self.game.remove_map()
```

To disable input only:

```
self.game.remove_map(keep_display=True)
```

Configure a map's settings:

```
self.game.maps['map_name'].config_map(
    x_y=None,
    width_height=None,
    border_color=None,
    border_width=None,
    dot_radius=None,
    dot_color=None,
    highlight=None,
    pad_multi=None,
    txt_color=None,
    txt_bg=None,
    font_size=None,
    bold=None,
    font=None,
    dot_transparency=None,
    txt_bg_transparency=None
)
```

Scenes

To change scene:

```
self.game.change_scene('chapter_name', 'scene_name')
```

-'chapter_name' is the directory name inside project folder. 'scene_name' is the name for the class specified in scene_list.py.

Save game

To save:

```
self.game.save(index=None, name=None)
```

-This saves the chapter, scene, and flag variables. No parameters will use the first unused number for the name, e.g. 'Save 3'

Get a list of saved game names:

```
self.game.get_saved_file_names(menu=True) -> e.g. [['Save 0', True], ['Save 1', True]]
```

-The returned list-boolean pairs are used for creating Menus. Use menu=False to return a simple list of names without booleans: ['Save 0', 'Save 1']

Load a game:

```
self.game.load_game(saved_game_name)
```

-Game loads once current scene ends (run() method ends), same as switching scenes. Use return to break out of the run() method mid scene.

Remove saved game:

```
self.game.delete_saved_game(saved_game_name)
```

Various

Music:

```
self.game.play_song(
        song_name,
        volume=None,
        fade=False
        stop=False,
        pause=False,
        unpause=False,
        repetition=-1
)
```

-volume is a float between 0 and 1.
-fade is time in milliseconds to fade until stop.

Check if a song is currently playing using pygame:

```
pygame.mixer.music.get_busy() -> T/F
```

Sound Effect:

```
self.game.sounds['sound_name'].play()
```

Shake Screen:

```
self.game.shake_screen(pause=False)
```

A pause without input:

```
self.game.game_loop_input(num)  # num is number of loops. 30 = 1 second
pygame.time.delay(1000)  # 1 second
```

To create a variable to use, can use the flag_vars dictionary (can also be added in configuration file):

```
self.game.flag_vars["new_var"] = False
```

Transitions

```
self.game.fade_out()
self.game.fade_in()
self.game.slide_left('background_image.jpg')
self.game.slide_right('background_image.jpg')
```

Coordinator

The map locations in configuration.py uses percent of screen rather than pixels for coordinates, so that it will scale to screen size changes. To easily get decimal locations, you can use the Coordinator program. Simply put the code at the start of your script and run the game. Then click on the map where you want the location, and the coordinates will print to the console. (Coordinator.py can also be run directly.)

```
self.game.coordinator.coordinate('map_name')
```