Chip's Core Escape

Puzzle Maze

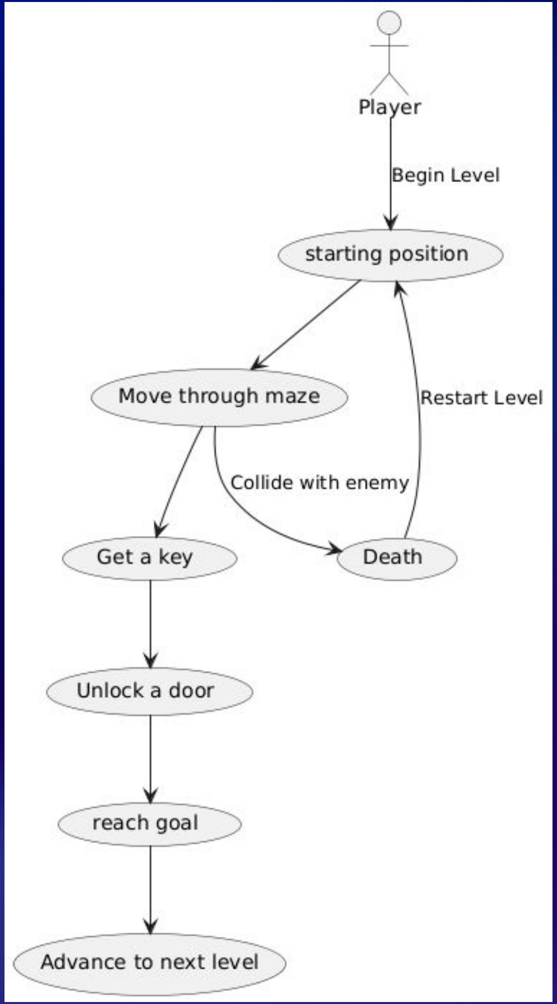
Creators:
Keegan Erickson
Jessica Story
Ethan Talbert

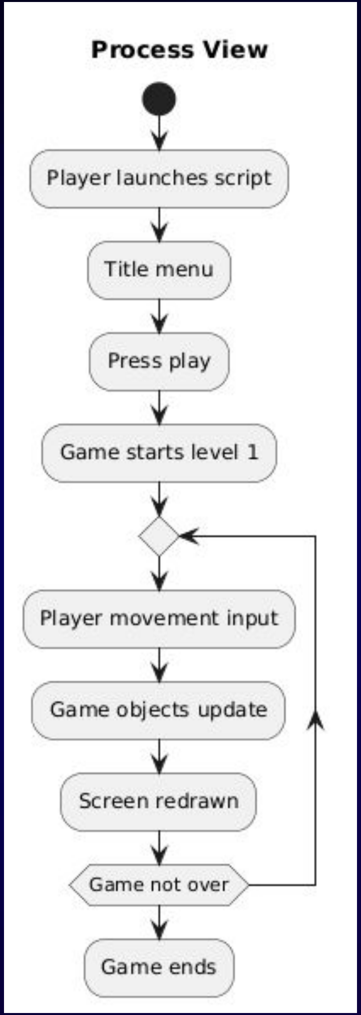
Advisor:
Dr. Basnet, Ram

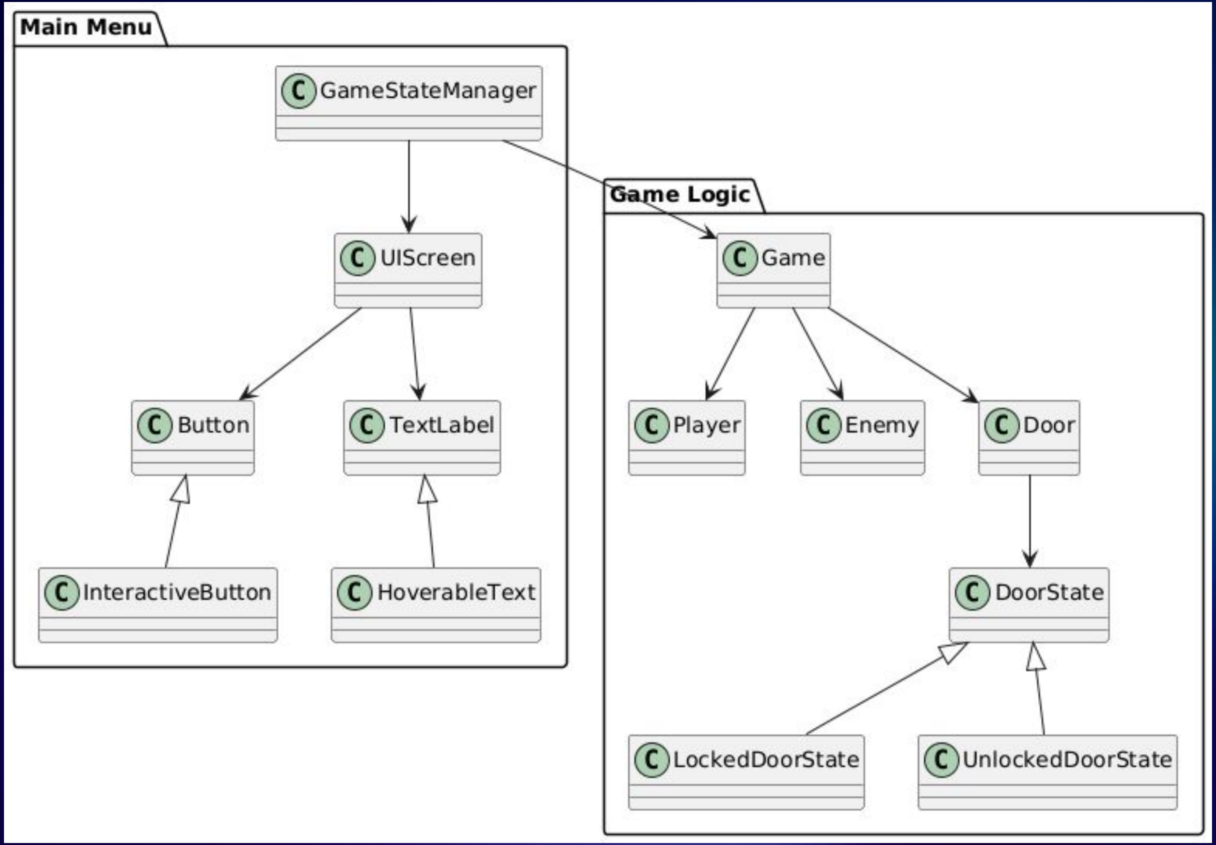
Object Oriented Programming Final Project

- Task:
  - To create an object oriented programming (OOP) project
- Decided on:
  - Maze/puzzle game
  - Pygame library
- Divided Project into 3 sub-projects
  - Main Menu & state transition
  - Game loop & logic
  - Game objects & interaction
- Management/communication
  - Github, Github Issues
  - Discord

Conceptual Flow




Process View
Player launches script
Title menu
Press play
Game starts level 1
Player movement input
Game objects update
Screen redrawn
Game not over
Game ends


Main Menu
GameStateManager
UIScreen
Button
TextLabel
InteractiveButton
HoverableText
Game Logic
Game
Player
Enemy
Door
DoorState
LockedDoorState
UnlockedDoorState

Design Patterns and OOD Concepts

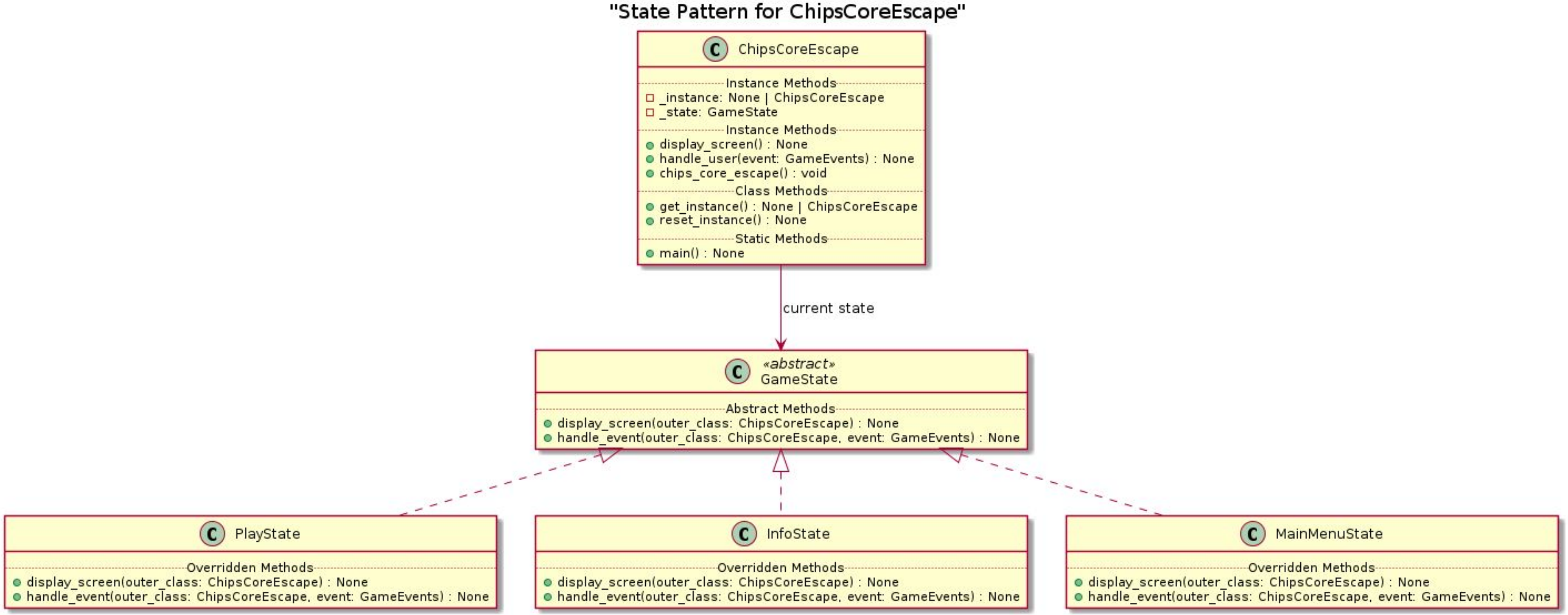
Design Patterns

- Template
  - Used in game.py
- State
  - Used in GameObjects.py,
  - Used in interactive_ui_elements.py,
  - Used in game_states.py
- Singleton
  - Used in game_states.py
- Decorator
  - Used in ui_elements.py

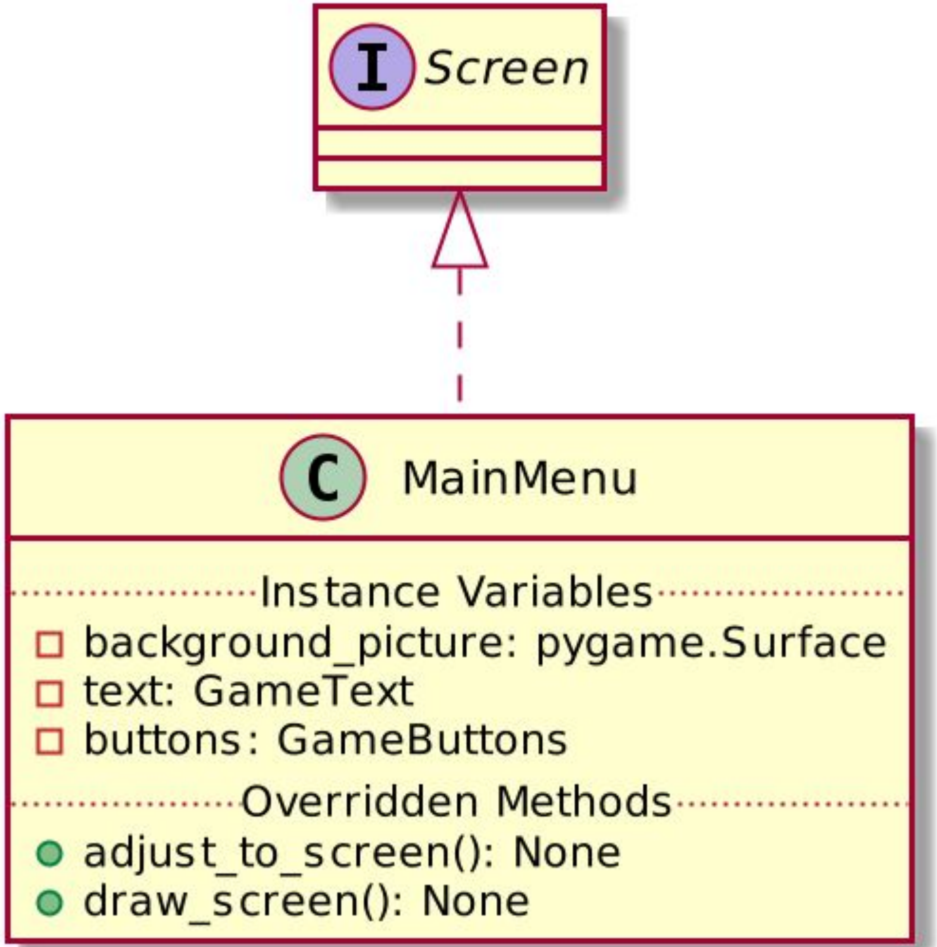
OOD Concepts

- Inheritance and Polymorphism
  - It was used in creating the Enemy and Player class
- Abstraction
  - The TileSet class uses this
- Getters and Setters
  - Used by the state class

Game State Control

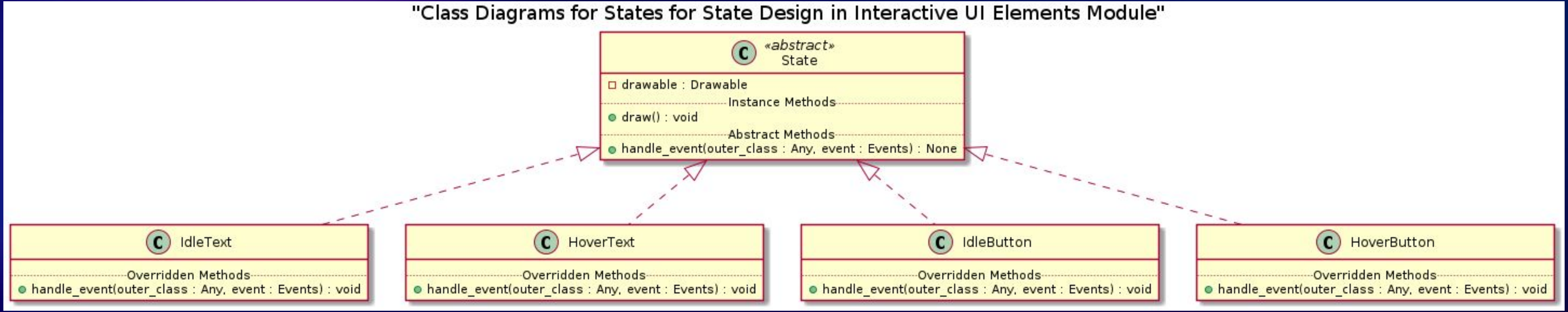

"State Pattern for ChipsCoreEscape"
ChipsCoreEscape
Instance Methods
_instance: None | ChipsCoreEscape
_state: GameState
Instance Methods
display_screen() : None
handle_user(event: GameEvents) : None
chips_core_escape() : void
Class Methods
get_instance() : None | ChipsCoreEscape
reset_instance() : None
Static Methods
main() : None
current state
«abstract»
GameState
Abstract Methods
display_screen(outer_class: ChipsCoreEscape) : None
handle_event(outer_class: ChipsCoreEscape, event: GameEvents) : None
PlayState
Overridden Methods
display_screen(outer_class: ChipsCoreEscape) : None
handle_event(outer_class: ChipsCoreEscape, event: GameEvents) : None
InfoState
Overridden Methods
display_screen(outer_class: ChipsCoreEscape) : None
handle_event(outer_class: ChipsCoreEscape, event: GameEvents) : None
MainMenuState
Overridden Methods
display_screen(outer_class: ChipsCoreEscape) : None
handle_event(outer_class: ChipsCoreEscape, event: GameEvents) : None

"Main Menu Class Diagram"

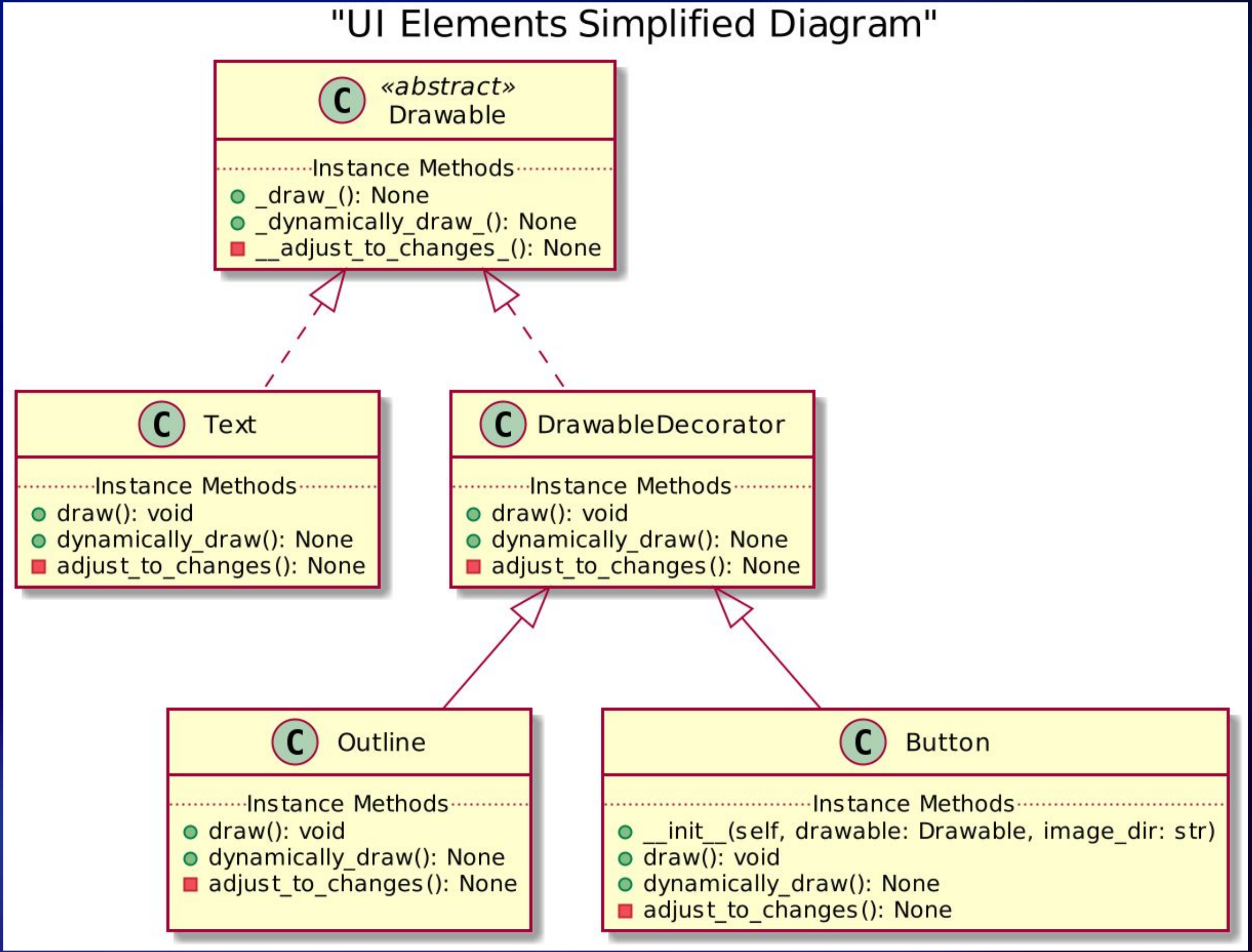


Main Menu

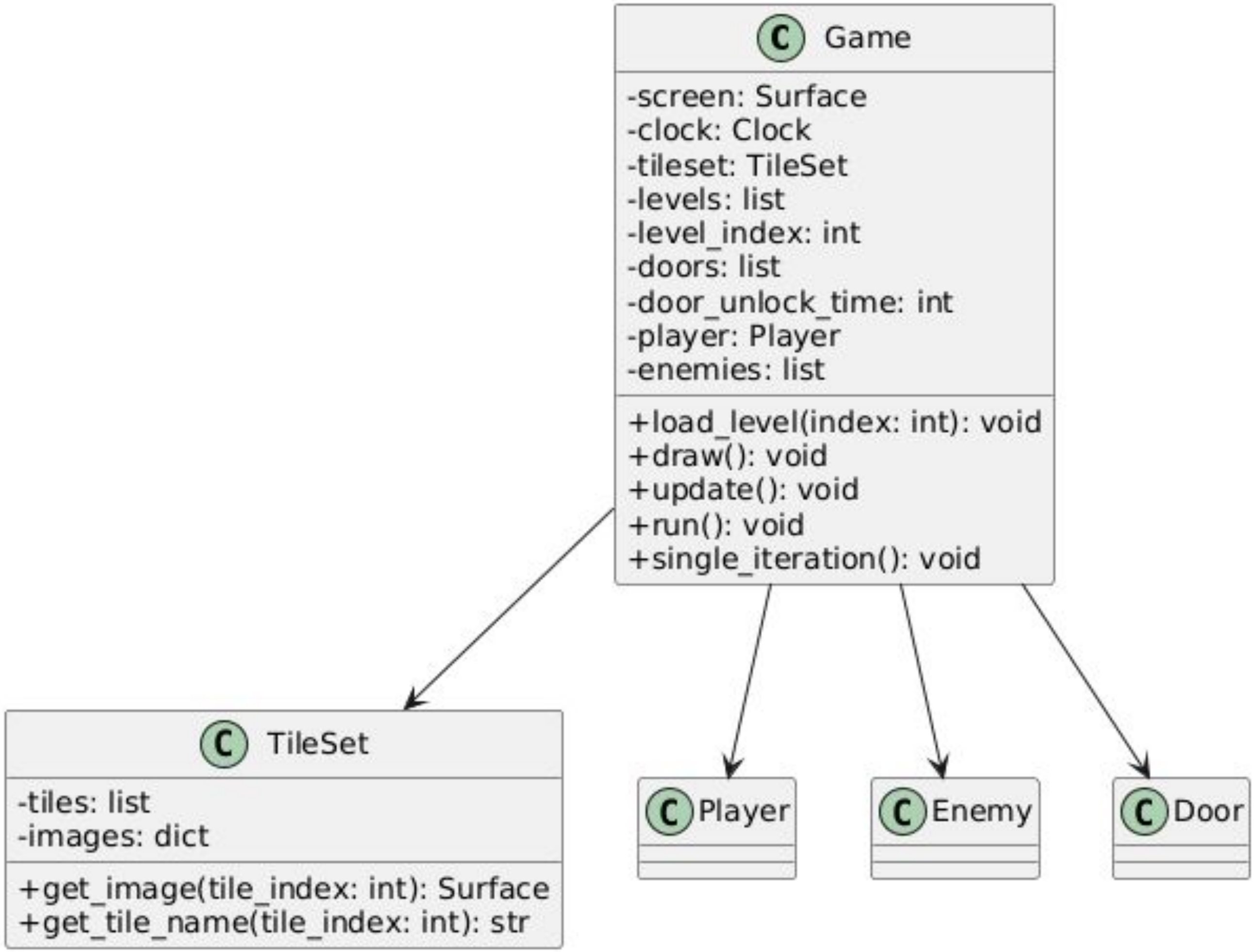
Interactive Elements


"Class Diagrams for States for State Design in Interactive UI Elements Module"
«abstract» State
drawable : Drawable
Instance Methods
draw() : void
Abstract Methods
handle_event(outer_class : Any, event : Events) : None
IdleText
Overridden Methods
handle_event(outer_class : Any, event : Events) : void
HoverText
Overridden Methods
handle_event(outer_class : Any, event : Events) : void
IdleButton
Overridden Methods
handle_event(outer_class : Any, event : Events) : void
HoverButton
Overridden Methods
handle_event(outer_class : Any, event : Events) : void

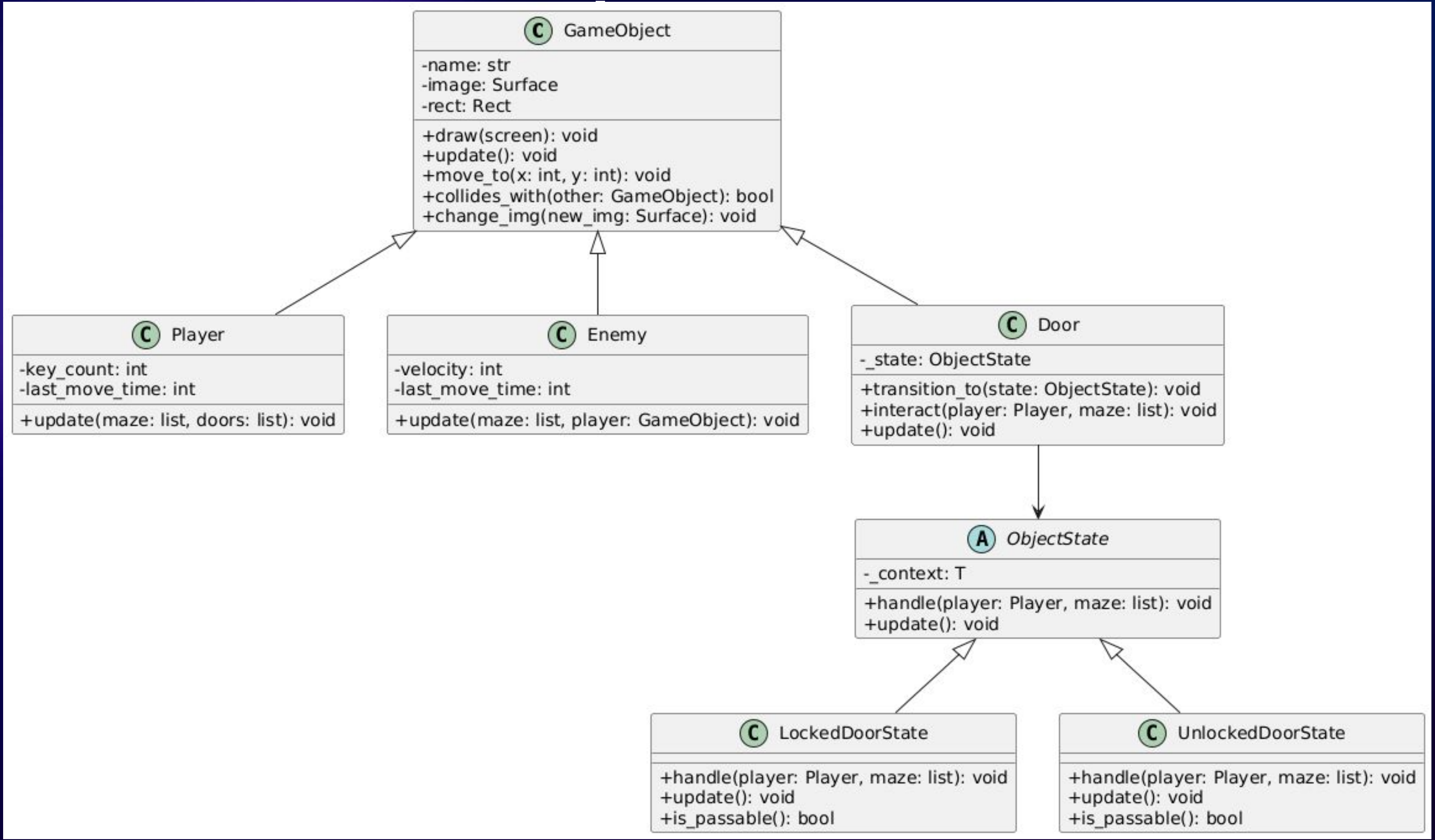
UI Elements


"UI Elements Simplified Diagram"
C «abstract» Drawable
Instance Methods
● _draw_(): None
● _dynamically_draw_(): None
■ __adjust_to_changes_(): None
C Text
Instance Methods
● draw(): void
● dynamically_draw(): None
■ adjust_to_changes(): None
C DrawableDecorator
Instance Methods
● draw(): void
● dynamically_draw(): None
■ adjust_to_changes(): None
C Outline
Instance Methods
● draw(): void
● dynamically_draw(): None
■ adjust_to_changes(): None
C Button
Instance Methods
● __init__(self, drawable: Drawable, image_dir: str)
● draw(): void
● dynamically_draw(): None
■ adjust_to_changes(): None

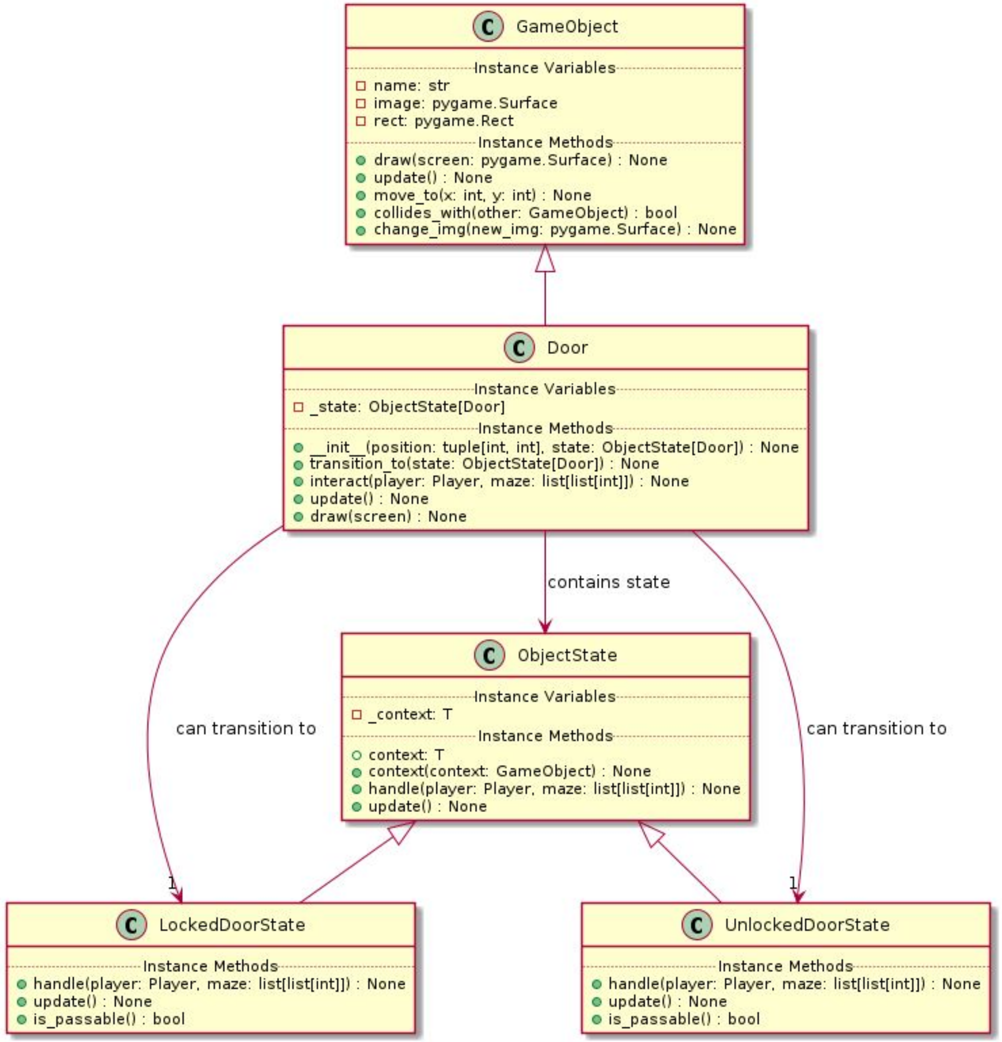
Game Loop


Game
-screen: Surface
-clock: Clock
-tileset: TileSet
-levels: list
-level_index: int
-doors: list
-door_unlock_time: int
-player: Player
-enemies: list
+load_level(index: int): void
+draw(): void
+update(): void
+run(): void
+single_iteration(): void
TileSet
-tiles: list
-images: dict
+get_image(tile_index: int): Surface
+get_tile_name(tile_index: int): str
Player
Enemy
Door

Game


GameObject
-name: str
-image: Surface
-rect: Rect
+draw(screen): void
+update(): void
+move_to(x: int, y: int): void
+collides_with(other: GameObject): bool
+change_img(new_img: Surface): void
Player
-key_count: int
-last_move_time: int
+update(maze: list, doors: list): void
Enemy
-velocity: int
-last_move_time: int
+update(maze: list, player: GameObject): void
Door
-_state: ObjectState
+transition_to(state: ObjectState): void
+interact(player: Player, maze: list): void
+update(): void
ObjectState
-_context: T
+handle(player: Player, maze: list): void
+update(): void
LockedDoorState
+handle(player: Player, maze: list): void
+update(): void
+is_passable(): bool
UnlockedDoorState
+handle(player: Player, maze: list): void
+update(): void
+is_passable(): bool

"State Design for Door Object Diagram"

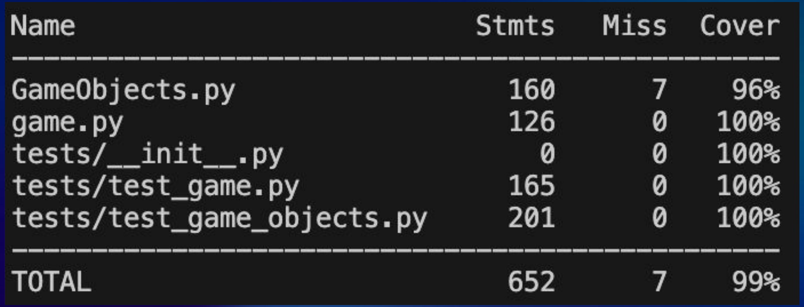


GameObject
Instance Variables
name: str
image: pygame.Surface
rect: pygame.Rect
Instance Methods
draw(screen: pygame.Surface) : None
update() : None
move_to(x: int, y: int) : None
collides_with(other: GameObject) : bool
change_img(new_img: pygame.Surface) : None
Door
Instance Variables
_state: ObjectState[Door]
Instance Methods
__init__(position: tuple[int, int], state: ObjectState[Door]) : None
transition_to(state: ObjectState[Door]) : None
interact(player: Player, maze: list[list[int]]) : None
update() : None
draw(screen) : None
contains state
can transition to
can transition to
ObjectState
Instance Variables
_context: T
Instance Methods
context: T
context(context: GameObject) : None
handle(player: Player, maze: list[list[int]]) : None
update() : None
1
1
LockedDoorState
Instance Methods
handle(player: Player, maze: list[list[int]]) : None
update() : None
is_passable() : bool
UnlockedDoorState
Instance Methods
handle(player: Player, maze: list[list[int]]) : None
update() : None
is_passable() : bool

Demo

Testing

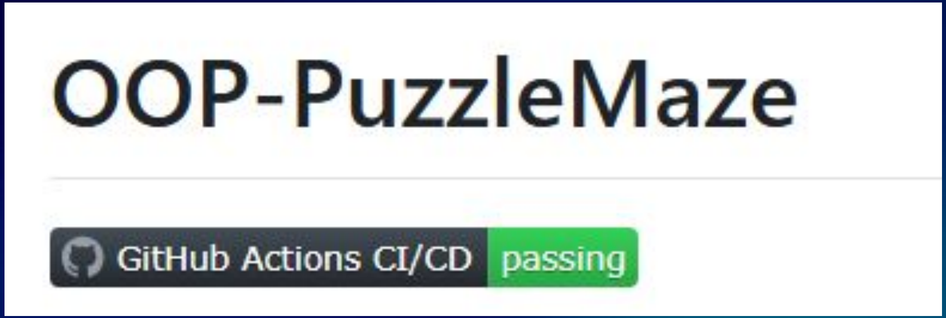
- ❖ Property-Based Testing
  - ➢ Generated random start positions and movements to validate player and movement logic across the grid
- ❖ Dependency Isolation
  - ➢ Use simple dummy objects Instead of real graphics so tests run fast and anywhere



```
Name                          Stmts   Miss  Cover
-------------------------------------------------
GameObjects.py                  160      7    96%
game.py                         126      0   100%
tests/__init__.py                 0      0   100%
tests/test_game.py              165      0   100%
tests/test_game_objects.py      201      0   100%
-------------------------------------------------
TOTAL                           652      7    99%
```

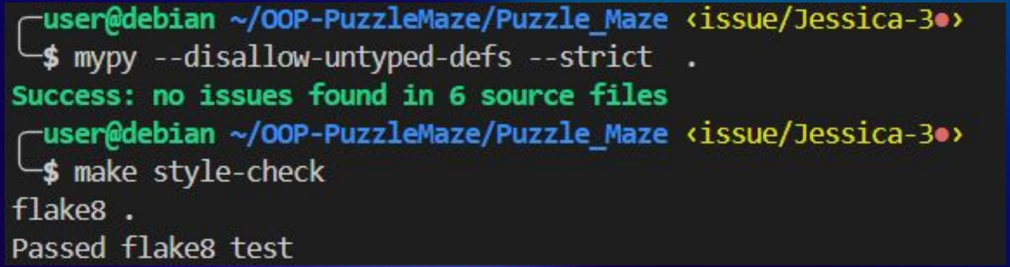
Best Practices

- CI/CD Results
  - Passed CI/CD tests on Github by following CI/CD pipeline
- Mypy and Flake8 Results
  - Adhered to Pep8 conventions throughout program



OOP-PuzzleMaze

GitHub Actions CI/CD passing



```
┌user@debian ~/OOP-PuzzleMaze/Puzzle_Maze ‹issue/Jessica-3●›
└$ mypy --disallow-untyped-defs --strict  .
Success: no issues found in 6 source files
┌user@debian ~/OOP-PuzzleMaze/Puzzle_Maze ‹issue/Jessica-3●›
└$ make style-check
flake8 .
Passed flake8 test
```

Takeaways

- Object interaction logic is tricky in a game
- Getting a door to work harder than players and enemies
- Learning to manage a multi-person project on Github
  - Merge conflicts
- Testing a GUI is difficult
  - Initially done manually
- Good opportunity to apply OOP

Questions