Wumpus: Network
Object Oriented Programming Assessment Task

Hamzah Ahmed

Contents

1. Overview

This document provides documentation for the Preliminary Object Oriented Programming Project. The appendixes provide external documentation and context useful for those studying the code.

1.1. Hunt the Wumpus: Background

Hunt the Wumpus is a classic computer game originally created in 1972 by Gregory Yob. The game is set in a series of interconnected caves, where the player must hunt a creature known as the Wumpus. The player navigates the cave system, avoiding hazards such as bottomless pits and super bats, while attempting to deduce the Wumpus's location and shoot it to win the game.

1.2. Wumpus: Network

Wumpus: Network is the result of this project. It is a game inspired by Hunt the Wumpus, adding original features. It is a graphical game with 3D and 4D levels, expanding the scope of the original game with more challenging and interesting levels.

2. Journal

2.1. 05/05/2025 git commit hash: `19c52af565583322ebfb73220f7fd8ead5119bbe`

2.1.1. Progress

- Competed text based implementation of the game
- This implementation loads the game map from a JSON file
  - This is used to create the `Level` class, which holds the position of `Caves` and `Hazards`
  - `Hazard` has various subclasses for `Wumpus`, `BottomlessPit` and `Superbats`, which manipulate the `PlayerController` and `Level` on events such as `on_player_enter` (implemented as methods)
  - `PlayerController` has the actual player controlling functionality implemented in subclasses such as `TextPlayerController`
    - This will ease migration to a graphical implementation

2.1.2. Future improvements

- Add unit tests and documentation
- Change `Level` to an event-based system, where entities yield or return events that control the level, rather than having a tight coupling to the `Level` and `PlayerController`

2.2. 22/05/2025 git commit hash `854cebf7a5eb777694642c36225013d7ddd866ef`

2.2.1. Progress

- Added event based architecture
- This helped to decouple the Hazards from the Level
- Handling of player interaction with Hazards was also moved to Level.

2.3. 13/06/2025 git commit hash `9003447fea0d29cc30092adac94c328893a5294f`

2.3.1. Progress

- Complete migration to event based architecture
- Separate text-based game completely from core logic
- Add automated black-box testing
- Fix various edge cases (eg: shooting to room not adjacent to current room)
- Install pygame to nix shell

2.3.2. Future tasks

- Create graphical package

2.4. 20/05/2025 git commit hash `6a933ba5beb2a14ae5db1195709b73e29b06c576`

2.4.1. Progress

- Created graphical package
  - Has a stack of scenes (Main Menu, Level Select, How to Play, Playing, Paused)
  - Modal scenes can be pushed onto stack and pop'ed to preserve game state
- Used force directed graph drawing to draw level onto scene
  - However, force directed graph drawing prefers to have all edges same length
  - So level map converges onto a perspective-like map with crossings, rather than a flattened map

2.4.2. Future tasks

- I've decided against continuing with a flat 2D approach, rather using perspective projection of the actual polyhedral graph
- Hunt the Wumpus uses a dodechedral graph, so similar levels modelled after Platonic solids can be made

2.5. 20/07/2025 git commit hash `106c2d65c672cbe5ac7d59964eb6f37db517d4f2`

2.5.1. Progress

- Playable graphical version
  - Uses Renderer class to draw level, Caves and Player are Drawable instances that get sorted by depth
  - Arbitrary dimension perspective projection allows for 3D and 4D levels (or potentially higher in the future)
  - Created 4 levels (3 3D levels and 1 4D)
  - Animations when moving player or shooting to rotate view – so the player doesn't constantly have to reposition the camera
  - Spinning level graphics on main menu
  - Death counter and screen tint to indicate player death
    - I do this since this game will take a few deaths to complete due to randomness
    - Deaths are a level statistic that players can try to optimise as a *high score*

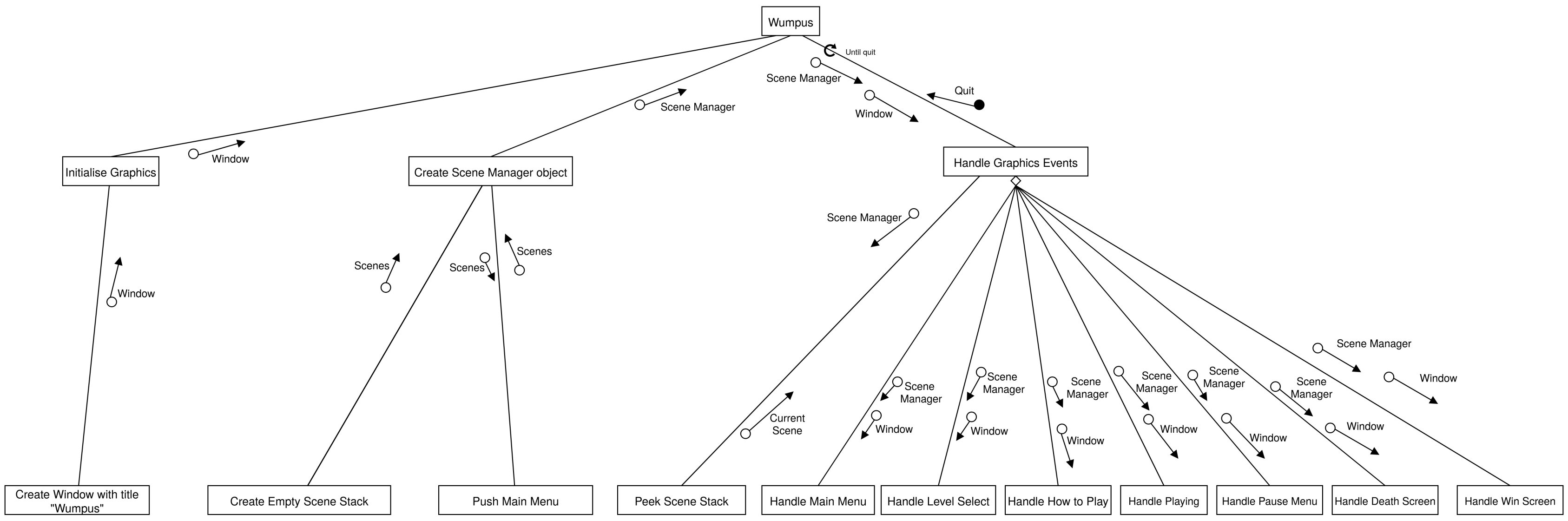
2.5.2. Future tasks

- Scene transitions
- How to play
- Credits menu
- Win screen (with high score)
- Level 5
- Level unlocking
- Sound effects

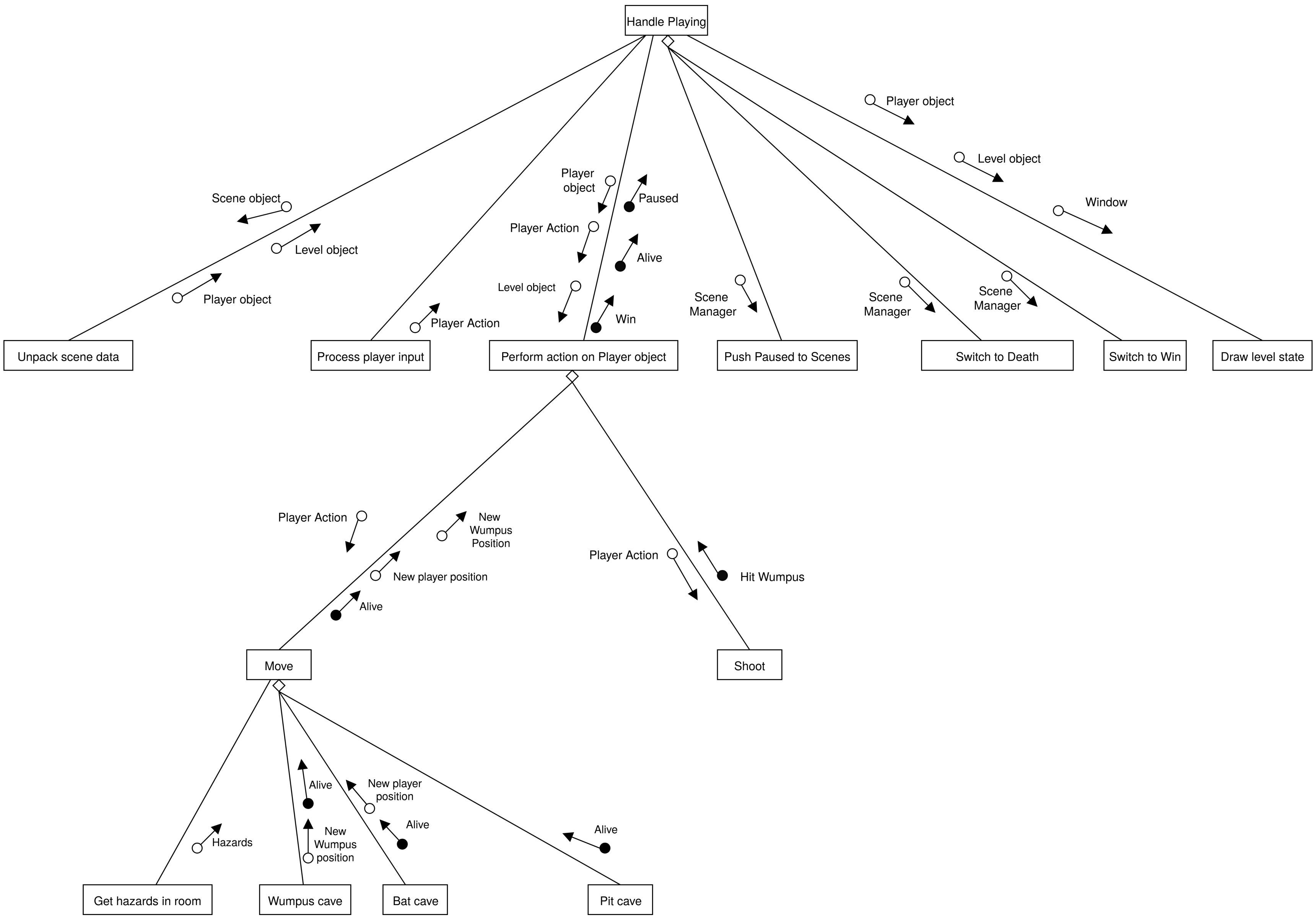
3. System Modelling

3.1. Structure Charts

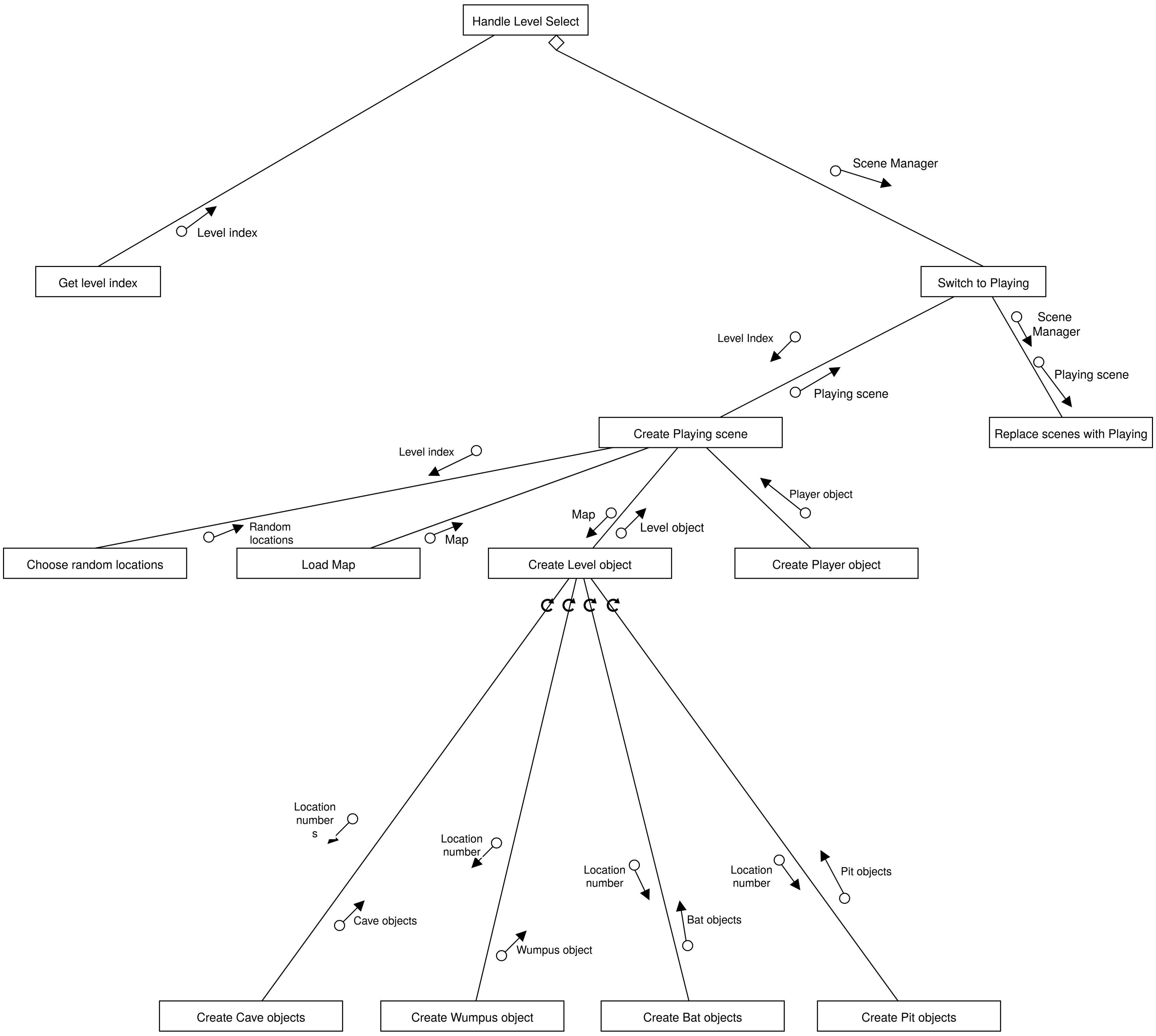
3.1.1. Wumpus



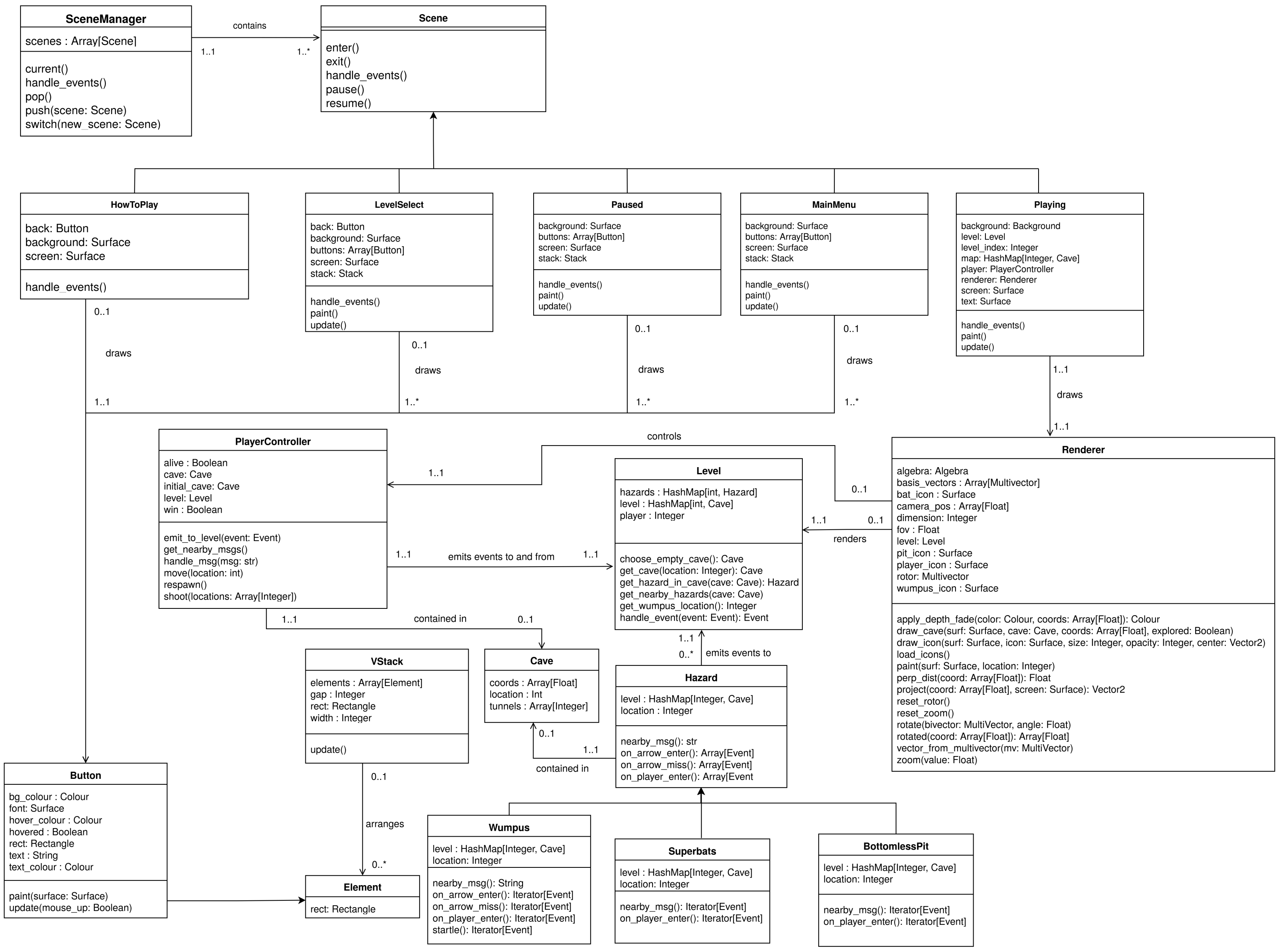
3.1.2. Handle Playing



3.1.3. Handle Level Select



3.2. Class Diagram



4. Data Dictionary

4.1. Internal represenation

Data Structure (type)	Attributes	Data type	Max length	Description
PlayerController (object)	alive	Boolean	1	Whether the player is alive
	cave	Cave object	EOF	Current cave the player is in
	initial_cave	Cave object	EOF	Cave where the player first spawned in (used for respawn)
	level	Level object	EOF	Level object containing caves and hazards
	win	Boolean	1	Whether the player has won

Data Structure (type)	Attributes	Data type	Max length	Description
Level (object)	hazards	Array of hazard objects	EOF	The hazard objects in the level
	level	Hashmap of Integer to the Cave objects	EOF	Maps cave location number to cave objects
	player	Integer	3	Cave location number of the player

Data Structure (type)	Attributes	Data type	Max length	Description
Cave (object)	location	Integer	3	This cave's location number
	tunnels	Array of integers	EOF	Array of cave location numbers this cave connects to
	coords	Array of reals	EOF	3D coordinates of cave

Data Structure (type)	Attributes	Data type	Max length	Description
Hazard (object)	level	Hashmap of Integer to Cave object	EOF	Maps cave location number to cave objects
	location	Integer	3	Hazard's location number

4.2. Graphical

Data Structure (type)	Attributes	Data type	Max length	Description
SceneManager (object)	scenes	Array of Scene objects	EOF	A stack where the top scene is the one that is shown

Data Structure (type)	Attributes	Data type	Max length	Description
MainMenu (Scene)	buttons	Array of Button objects	EOF	Buttons that are shown on screen

Data Structure (type)	Attributes	Data type	Max length	Description
HowToPlay (Scene)	back	Button object	EOF	Button to close the menu

Data Structure (type)	Attributes	Data type	Max length	Description
LevelSelect (Scene)	buttons	Array of Button objects	EOF	Buttons to enter each menu
	back	Button object	EOF	Button to return to main menu

Data Structure (type)	Attributes	Data type	Max length	Description
Playing (Scene)	level	Level object	EOF	Level object containing caves and hazards (see above)
	player	PlayerController object	EOF	PlayerController to control actions of player (see above)

Data Structure (type)	Attributes	Data type	Max length	Description
DeathMenu (Scene)	buttons	Array of Button objects	EOF	Buttons that are shown on screen

Data Structure (type)	Attributes	Data type	Max length	Description
WinMenu (Scene)	buttons	Array of Button objects	EOF	Buttons that are shown on screen

Data Structure (type)	Attributes	Data type	Max length	Description
Button (Scene)	bg_colour	RGB colour	6	Background colour of button
	font	Font file	EOF	Font file to text
	hover_colour	RGB colour	6	Background colour of button when hovered
	hovered	Boolean	1	Whether the button is currently hovered

	rect	Array of integers	4	Top, left, width, height coordinates of button
	text	String	100	Label to show on the button
	text_colour	RGB colour	6	Colour to draw the text

5. Testing Strategies

6. Evaluation

6.1. Implementation of Object Oriented Programming concepts

6.2. Functionality

- TODO: write about how tunnels always draw behind nodes and how that is technically wrong.

6.3. Originality

6.4. Documentation

Appendix A. Code structure

The use of LLMs was allowed in this project.

```
wumpus/
├── graphical/     # Graphical user interface: scenes, icons, GUI utilities
├── level_gen/     # Level generation logic and algorithms for different map types
├── text/          # Text-based interface and logic for terminal gameplay
└── wumpus/        # Core game logic: cave structure, events, hazards, player, levels
```

A.1. Commands

1.1.1. Running Unit Tests

```
python -m unittest
```

1.1.2. Playing graphical

```
python -m graphical
```

1.1.3. Playing text

```
python -m text
```

1.1.4. Exporting folio to pdf

```
cd folio
typst compile main.typ
```

Appendix B. Arbitrary Dimension Perspective Projection and Rotation