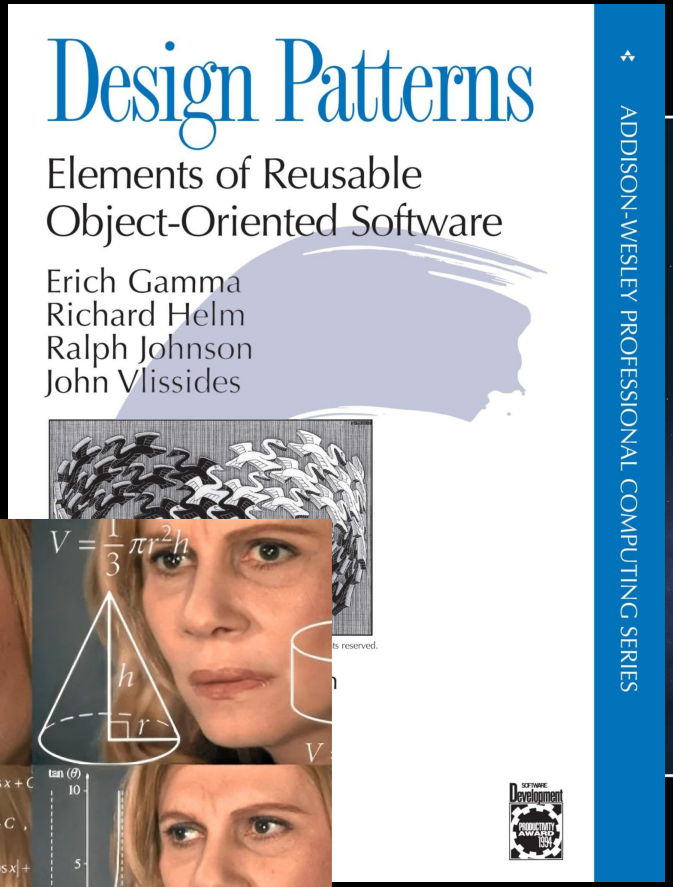
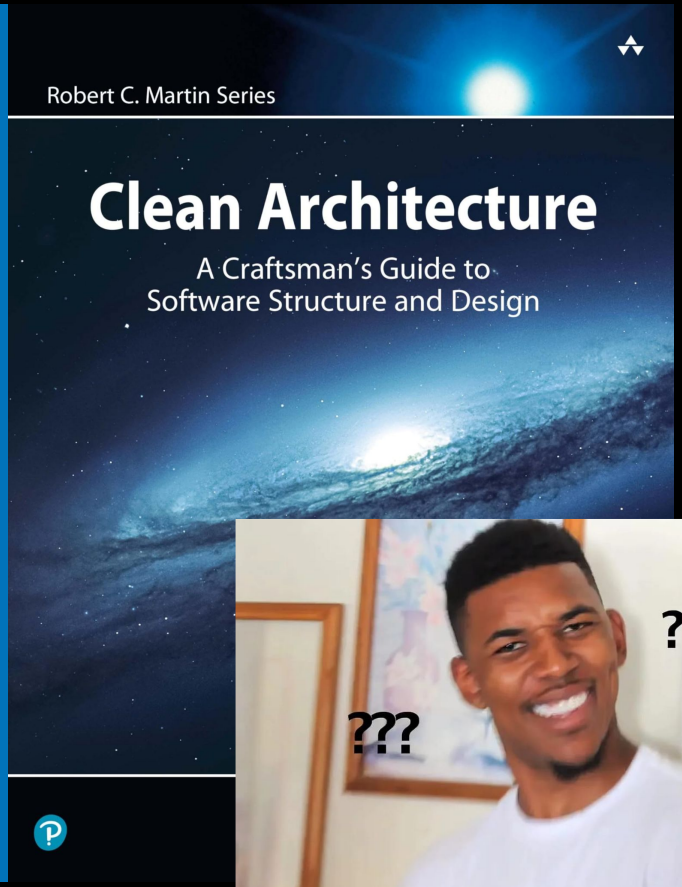
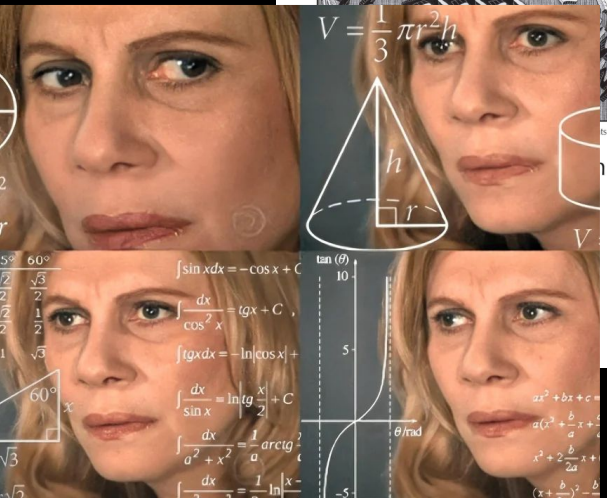
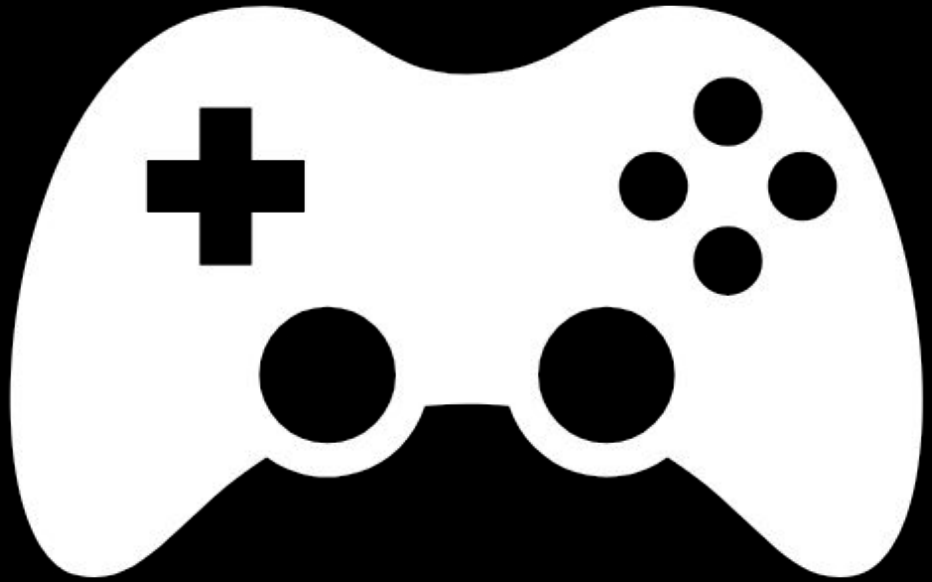

Design Patterns
Elements of Reusable
Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

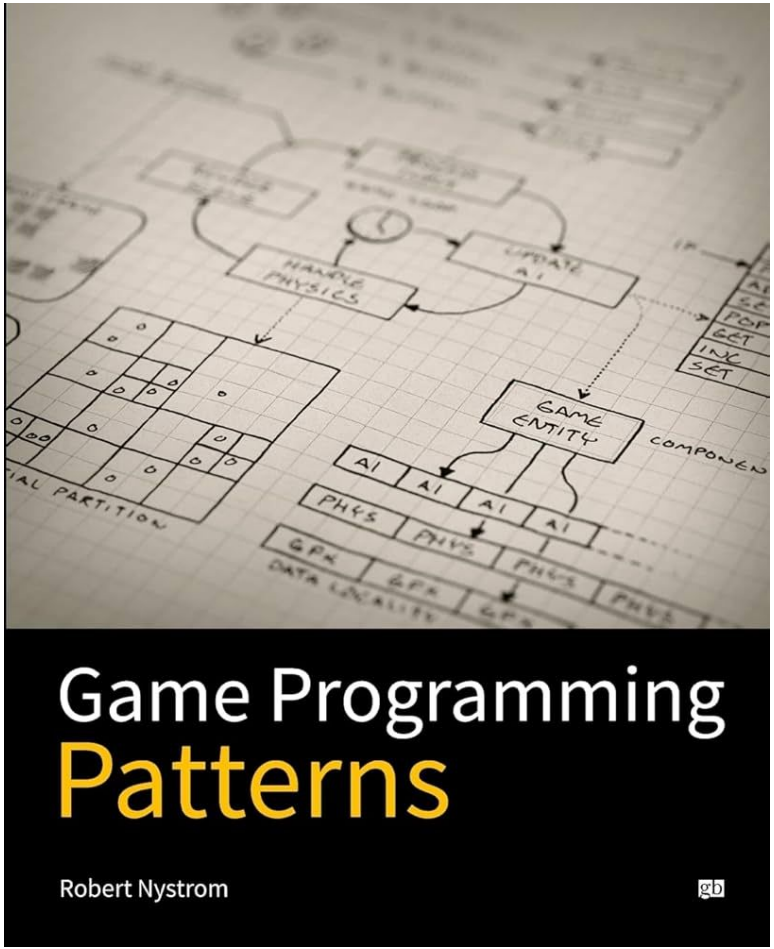

Robert C. Martin Series
Clean Architecture
A Craftsman's Guide to
Software Structure and Design


$V = \frac{1}{3}\pi r^2 h$
h
r

???
???

What makes Design Patterns and Software Architecture so difficult to learn or teach?





Research

I. Introduction

1. Architecture, Performance, and Games

II. Design Patterns Revisited

2. Command
3. Flyweight
4. Observer
5. Prototype
6. Singleton
7. State

III. Sequencing Patterns

8. Double Buffer
9. Game Loop
10. Update Method

IV. Behavioral Patterns

11. Bytecode
12. Subclass Sandbox
13. Type Object

V. Decoupling Patterns

14. Component
15. Event Queue
16. Service Locator

VI. Optimization Patterns

17. Data Locality
18. Dirty Flag
19. Object Pool
20. Spatial Partition

Singleton

"Ensure a class has one instance,
and provide a global point of access to it."[1]

[1] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland GmbH. (p. 127)

Command

“Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.”[2]

[2] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland GmbH. (p. 233)

Observer

“Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.”[3]

[3] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland GmbH. (p. 293)

State

“Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.”[4]

[4] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of Reusable Object-Oriented Software*. Pearson Deutschland GmbH. (p. 305)

Which Game Engine should I use

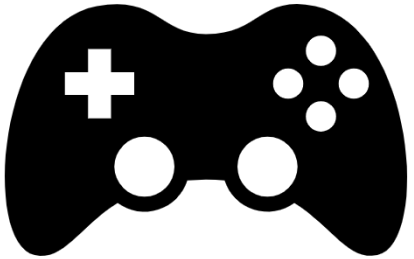

lib
GDX

Game Design

Setting

Art Style

Character Controls



The PlayerController

```
public interface PlayerState {
    void update(float delta);
    void render(SpriteBatch batch, float delta);
    void onEnter();
    void onExit();
}
```

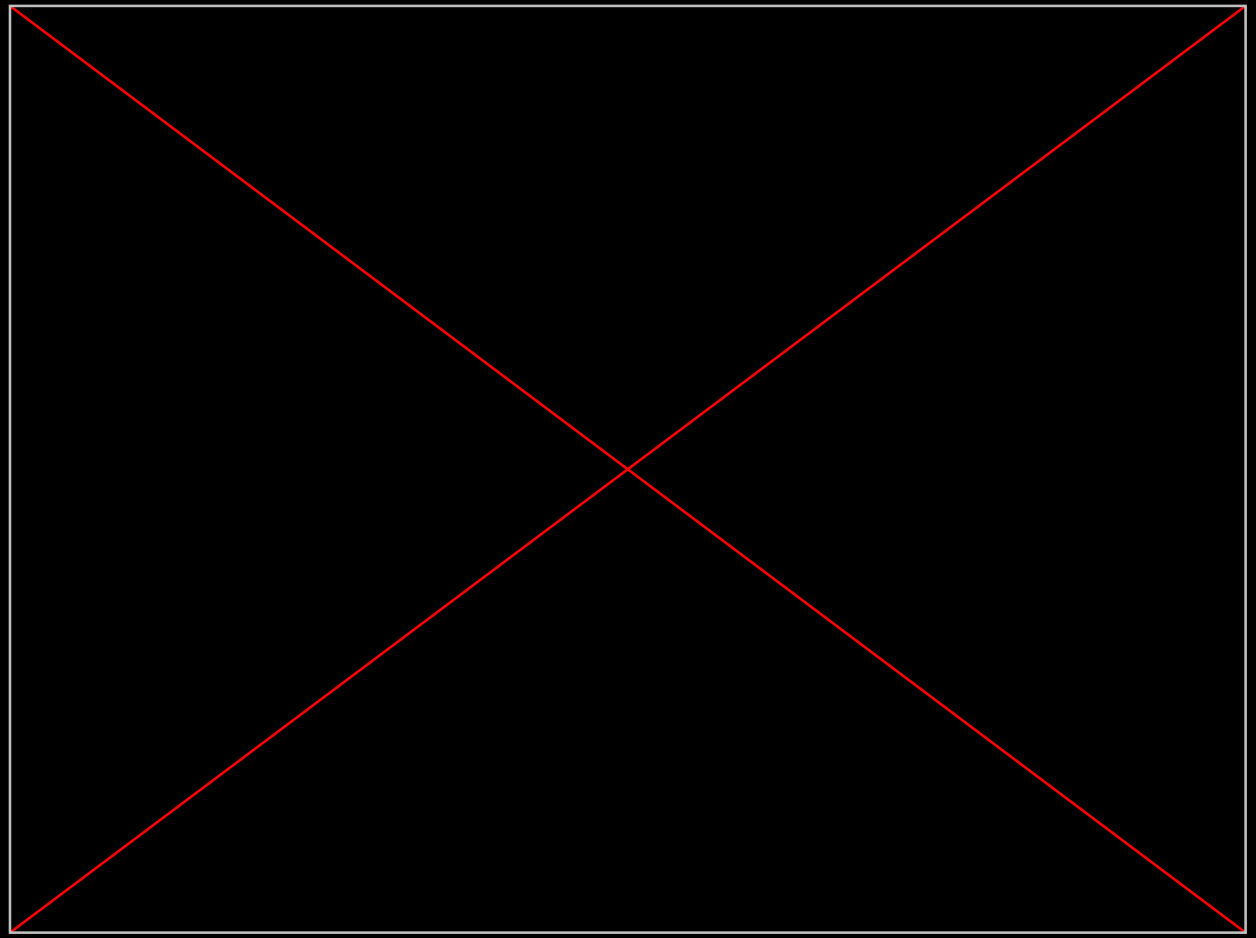
Input Handling

```
public class InputHandler extends InputAdapter {
      private final Map<Integer, Command<PlayerController>> keyDownMappings;

      private final Command<PlayerController> jumpCommand = new Command<PlayerController>() {
              @Override
              public void execute(PlayerController playerController) {
                  playerController.jump();
              }
      };


      public InputHandler(PlayerController playerController) {
            this.playerController = playerController;
            keyDownMappings = new HashMap<Integer, Command<PlayerController>> ();
            keyDownMappings.put(Input.Keys.UP, jumpCommand);
      }

      @Override
      public boolean keyDown(int keycode) {
           Command<PlayerController> playerControllerCommand = keyDownMappings.get(keycode);
           if (playerControllerCommand != null) {
            playerControllerCommand.execute(playerController);
           }
           pressedKeys.add(keycode);
           return true;
      }
}
```

Interaction System

Audio

```
public class AudioController {

    private static AudioController instance;

    private final EnumMap<MusicTrack, Music> musicMap = new EnumMap<MusicTrack, Music>(MusicTrack.class);

    private AudioController() {}

    public static AudioController getInstance() {
        if (instance == null) {
            instance = new AudioController();
        }
        return instance;
    }

    public Music getMusic(MusicTrack musicTrack, boolean looping, float volume) {
        Music music = musicMap.get(musicTrack);
        if (music == null) {
            music = Gdx.audio.newMusic(Gdx.files.internal(musicTrack.getPath()));
            musicMap.put(musicTrack, music);
        }
        music.setLooping(looping);
        music.setVolume(volume);
        return music;
    }
}
```



kringel.games