PyDrums – Digital Beatmaker Workstation

Project Report

Author: Dyka Adlero Clifzier Holy Covenant
Course: Algorithm and Programming
Institution: BINUS INTERNATIONAL

---

1. Project Specification

1.1 Project Overview

PyDrums is a Python and Pygame-based desktop beatmaker. It emulates a step sequencer akin to professional Digital Audio Workstations (DAW) such as Ableton, FL Studio, etc. Users can create custom drum patterns by toggling a grid, set their own tempo (BPM), adjust the length of the beat, mute/enable a specific instrument track, load presets, and save or load their own beat.

1.2 Objectives

The objectives of this project are:

- To enable musicians and non-musicians express their creativity in beat making
- To apply OOP and best practices in Python learned throughout the course
- To ensure precise timing in audio playback
- To emulate professional step-sequencer
- To implement storage for saved beats

1.3 Scope

The application supports:

- Simultaneous/Individual drum instrument playback
- Adjustable BPM and beat length
- Preset beat loading
- Saving and loading custom beats
- Real-time playback and editing

Out of scope:

- MIDI export
- Advanced audio effects

- Volume level adjustment per instrument
- External hardware integration

---

2. Solution Design

2.1 Overall Architecture

The program follows a modular structure where each class handles a specific feature

Module	Responsibility
`main.py`	Primary logic and main loop
`sequencer.py`	Beat progression and timing
`sound_manager.py`	Audio loading and playback
`ui_manager.py`	Drawing and UI interaction
`preset_manager.py`	Preset beat management
`storage_manager.py`	Saving and loading beats
`menus.py`	UI for menus

2.2 Design Approach

- **Top-down control:** `PyDrumsApp` coordinates all subsystems
- **Encapsulation:** Each module manages its own data and logic
- **Polymorphism:** Menu classes share a common interaction pattern
- **Separation of concerns:** UI, audio, logic, and storage are isolated

---

3. Implementation & How It Works

3.1 Core Algorithm – Step Sequencer

The sequencer operates on a fixed time step:

1. A clock ticks at a fixed FPS
2. BPM determines when a beat advance
3. The active beat index moves forward
4. If a grid cell is active, its sound is played

Timing is deterministic, ensuring consistent rhythm playback.

3.2 Data Structures Used

3.2.1 Beat Grid

```
grid[instrument][step]
```

- 2D list
- `1` = active beat
- `-1` = inactive beat

3.2.2 Presets

```
{
  "Preset Name": {
    "beats": int,
    "bpm": int,
    "pattern": list[list[int]]
  }
}
```

Stored in dictionaries and deep-copied to prevent mutation.

3.2.3 State Flags

Boolean flags control UI and program flow:

- `playing`
- `save_menu`
- `load_menu`
- `load_preset`

3.3 Key Class Responsibilities

`PyDrumsApp`

- Owns all managers
- Runs the main loop
- Handles user input
- Synchronizes UI and sequencer state

`Sequencer`

- Advances beats based on BPM
- Maintains the active beat
- Controls timing logic

`SoundManager`

- Loads sound files
- Plays sounds by instrument index

`UIManager`

- Draws grid and controls
- Returns clickable regions

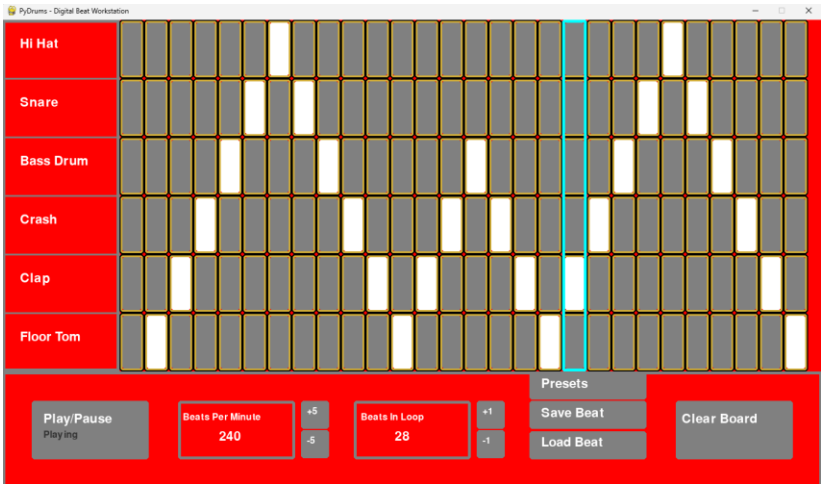
---

4. Evidence of Working Program

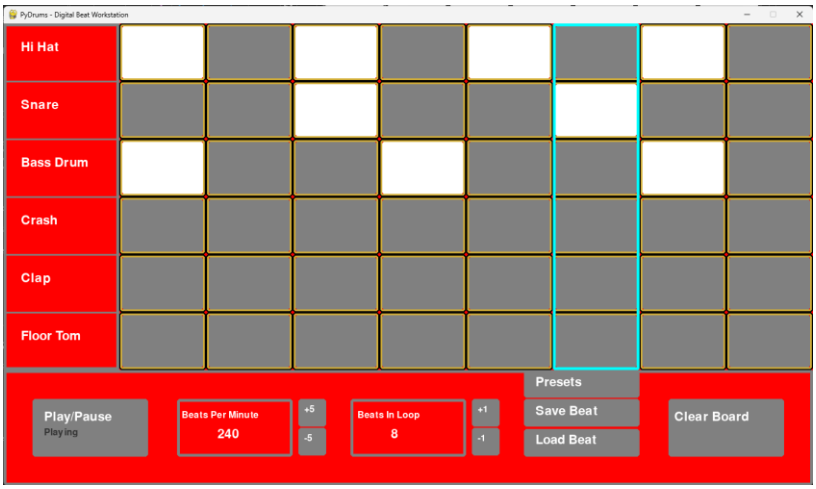
4.1 Screenshots to Include

📸 Insert screenshots under each heading

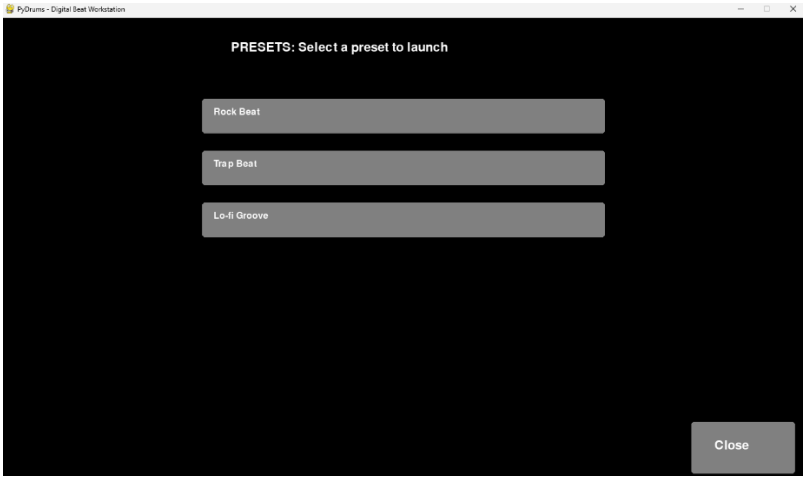
1. **Main Sequencer Interface**
   - Grid with active beats
   - BPM and beat controls visible



2. **Playback in Action**
   - Highlighted active beat column

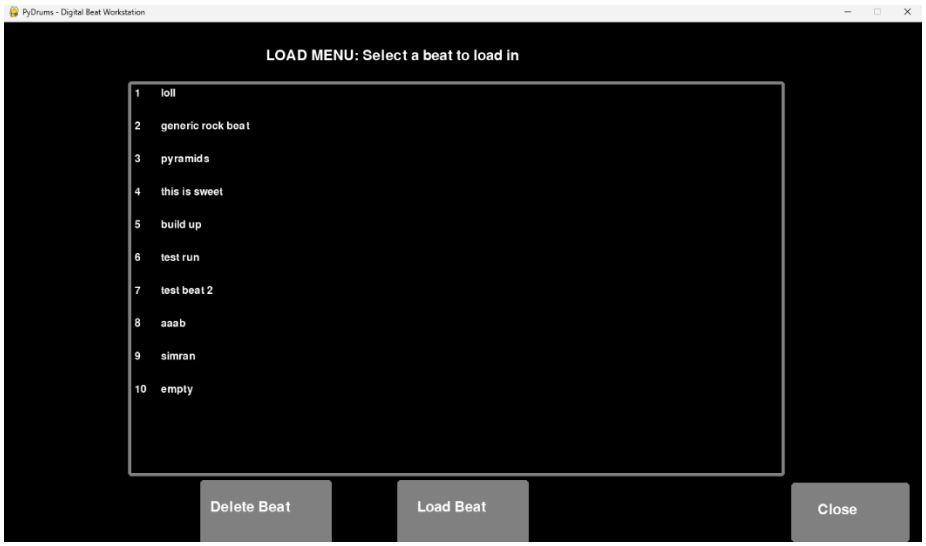


3. **Preset Menu**
    - Preset list displayed



4. **Save Menu**
    - Text input for beat name

5. **Load Menu**
    - List of saved beats



---

5. Discussion & Reflection

5.1 What Works Well

- Clear separation of logic
- Predictable timing behavior
- Easy to extend with new instruments
- Clean, readable code structure

5.2 Challenges

- Synchronizing UI state with sequencer state
- Managing deep copies to prevent data corruption
- Handling multiple modal menus without conflicts

5.3 Future Improvements

- Add more instruments dynamically
- Implement MIDI or audio export
- Improve UI scaling for different screen sizes

---

6. Statements on The Use of Artificial Intelligence

Project Title: PyDrums – Python-Based Drum Sequencer Application

In the development of the PyDrums application, Artificial Intelligence tools were used exclusively as a learning and development assistant. The AI support was limited to the following activities:

Explaining Python programming concepts, libraries, and syntax relevant to audio playback and application logic

Assisting in understanding and debugging specific code segments, including state management, user input handling, and feature implementation (e.g., save/load, presets)

Providing guidance on best practices for code organization, modularization, and readability

Supporting conceptual problem-solving during feature development without directly replacing student work

All core logic, system structure, feature design, implementation, testing, and final integration were performed and controlled by the student. The AI tool was not used to automatically generate a complete application, nor to circumvent learning outcomes, assessments, or academic integrity requirements.

Prompt example:

- Explain to me the concept of polymorphism in a nutshell.
- Roleplay as a veteran programmer and explain to me why this snippet works.
- Check for indentation errors
- Debug this code

7. Conclusion

This project successfully demonstrates a functional digital beatmaker using Python and Pygame. Through modular design and object-oriented principles, PyDrums achieves real-time interaction, precise timing, and maintainable code structure.