ASE 420 Team Project

Error 404: Name Not Found

Week 2 Progress Report – Tetris

📅 Week 2: Sept 15 – Sept 21

🎯 Focus: Refactoring Board Logic & Implementation

Team Overview

Team Members

- **Jeffrey Perdue** – Team Leader
- **Anna Dinius** – Board & Line Clearing
- **Cody King** – Pieces & Collision
- **Owen Newberry** – Rendering & Controls

Sprint Progress: 3/12 milestones completed (25% burn down rate)

Week 2 Goals Summary

Anna's Goals ✅

- ✅ Refactor and optimize the playing field grid
- ✅ Write unit tests for the playing field grid

Cody's Goals ✅

- ✅ Finalize requirements for piece representation and movement/rotation features
- ✅ Design the Piece class structure
- ✅ Start defining structure of figures.py to hold shape and rotation data

Week 2 Goals Summary (cont.)

Owen's Goals ✅

- ✅ Implement board rendering with Pygame
- ✅ Add ability to draw active piece from game state
- ✅ Confirm rendering updates each frame in the game loop

Statistics Overview

Lines of Code Added

- **Anna**: 605 LoC total
  - `src/game/board.py`: 39
  - `tests/test_board.py`: 71
  - Legacy code refactoring: 495
- **Cody**: 159 LoC total
  - `src/game/piece.py`: 141
  - `src/figures.py`: 18
- **Owen**: 77 LoC total
  - `src/game/game.py`: 36
  - `src/view/pygame_renderer.py`: 41

Anna's Progress: Board Implementation

Major Achievements

- **Board Class Creation**: Refactored board implementation into dedicated `Board` class
- **Import System**: Standardized imports with `from src.game.board import Board`
- **Legacy Code Cleanup**: Eliminated global `Field` variable, simplified board operations
- **Unit Testing**: Created 5 comprehensive unit tests for Board class

Key Changes

```
# Before: Global variables and functions
Field = [[0 for _ in range(Width)] for _ in range(Height)]

# After: Encapsulated Board class
GameBoard = Board(height, width)
GameBoard.clear_full_lines()
```

Cody's Progress: Piece System Design

Major Achievements

- **Piece Class Design**: Created comprehensive Piece class with movement/rotation
- **Data Organization**: Moved piece data to `figures.py` and colors to `constants.py`
- **Code Quality**: Improved readability with better variable names and structure
- **Modularity**: Eliminated magic numbers, centralized configuration

Key Changes

```python
# Before: Hard-coded piece data scattered throughout code
# After: Centralized, modular approach
from src.game.piece import Piece
from src.figures import FIGURES
from src.constants import COLORS
```

Owen's Progress: Rendering System

Major Achievements

- **Renderer Class**: Created dedicated `PygameRenderer` class
- **Separation of Concerns**: Moved rendering logic out of main game file
- **Frame Updates**: Confirmed rendering updates every frame in game loop
- **Clean Integration**: Simplified main loop with encapsulated rendering

Key Changes

```
# Before: Inline rendering in main loop
for i in range(len(board)):
    for j in range(len(board[i])):
        pygame.draw.rect(screen, colors[board[i][j]], ...)

# After: Clean renderer calls
renderer = PygameRenderer(screen)
renderer.draw_board(GameBoard)
renderer.draw_piece(current_piece)
```

Technical Improvements

Code Quality Enhancements

- **Modularity**: Separated concerns into dedicated classes
- **Testability**: Added comprehensive unit tests
- **Maintainability**: Eliminated global variables and magic numbers
- **Readability**: Improved variable names and code structure

Architecture Benefits

- **Encapsulation**: Board and Piece logic now properly encapsulated
- **Import System**: Standardized, reliable import paths
- **Separation**: Rendering, game logic, and data cleanly separated

Current State Assessment

Functionality Status

- ✅ **Board System**: Fully functional with proper class structure
- ✅ **Piece System**: Designed and implemented with movement/rotation
- ✅ **Rendering System**: Working with frame-by-frame updates
- ✅ **Testing**: Unit tests in place for core components

Integration Status

- **Import System**: Standardized import patterns established in each branch
- **Dependencies**: Encapsulated classes ready for integration
- **Code Quality**: Significantly improved from original implementation

Challenges Overcome

Technical Challenges

- **Import Resolution**: Fixed module path issues when running from different directories
- **Legacy Code**: Successfully refactored complex global state into clean classes
- **Branch Coordination**: Team members working on separate branches with consistent patterns

Process Challenges

- **Coordination**: Team members working on separate branches with overlapping concerns

Next Steps (Week 3)

Anna's Plan

- Implement line clearing logic with comprehensive tests
- Handle edge cases and cleanup
- Prepare Board class for integration with other components

Cody's Plan

- Integrate Piece class into existing game code
- Implement collision detection
- Add unit tests for movement and rotation

Owen's Plan

- Implement keyboard input mapping
- Complete rendering system integration
- Begin work on main game loop

Integration Goals

- **Merge all branches** into main branch
- **Resolve any conflicts** between Board, Piece, and Renderer implementations
- **Test integrated system** to ensure all components work together

Sprint Progress Update

Completed Milestones (3/12)

- ✅ Board refactoring and testing (Anna)
- ✅ Piece class design and implementation (Cody)
- ✅ Rendering system implementation (Owen)

Upcoming Milestones

- **Week 3**: Branch integration, line clearing logic, keyboard input mapping
- **Week 4**: Movement & rotation logic, main loop integration, edge cases
- **Week 5**: Game Over overlay, final testing, documentation, presentation preparation

Week 2 Progress vs. Week 1 Milestones Analysis

Anna - PERFECT ALIGNMENT ✅

- **Week 1 Milestone (Wk2)**: "Refactor grid, write tests"
- **Week 2 Reported**: ✅ "Refactor and optimize the playing field grid" + ✅ "Write unit tests for the playing field grid"
- **Assessment**: **100% on track** - Anna completed exactly what was planned for Week 2

Cody - AHEAD OF SCHEDULE ✅

- **Week 1 Milestone (Wk2)**: "Requirements & design Piece class"
- **Week 2 Reported**: ✅ "Finalize requirements for piece representation and movement/rotation features" + ✅ "Design the Piece class structure" + ✅ "Start defining structure of figures.py"
- **Assessment**: **Ahead of schedule** - Cody completed Week 2 goals PLUS started Week 3 work:
  - ✅ Week 2: Requirements & design (completed)
  - ✅ Week 3: Implement Piece + shapes/rotations (partially completed - created `piece.py` with 141 LoC and `figures.py`)

Owen - PERFECT ALIGNMENT ✅

- **Week 1 Milestone (Wk2)**: "Board rendering, draw active piece"
- **Week 2 Reported**: ✅ "Implement board rendering with Pygame" + ✅ "Add ability to draw active piece from game state" + ✅ "Confirm rendering updates each frame in the game loop"
- **Assessment**: **100% on track** - Owen completed exactly what was planned for Week 2

Team Performance

Strengths Demonstrated

- **Collaboration**: Effective coordination on separate branches with consistent patterns
- **Code Quality**: Significant improvements in maintainability and testability
- **Problem Solving**: Successfully overcame technical challenges
- **Documentation**: Maintained clear progress tracking
- **Milestone Adherence**: Strong alignment with original Week 1 planning

🎯 Week 3 Focus

Priority: Individual Milestone Completion + Branch Integration
Goal: Complete Week 3 milestones while preparing for integration
Success Metrics:

- Anna: Line clearing logic implemented and tested
- Cody: Piece class integrated into existing code
- Owen: Keyboard input mapping functional
- All branches ready for Week 4 integration

Questions & Discussion

Ready for branch integration challenges?

Any concerns about merging separate implementations?

Integration strategy and timeline?