ASE 420 Team Project

Error 404: Name Not Found

Week 4 Progress Report – Tetris

📅 Week 4: Sept 29 – Oct 5

🎯 Focus: Code Quality & Integration Preparation

Team Overview

Team Members

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

Sprint Progress: 7/10 milestones completed (70% total, 20% per week)

Week 4 Goals Summary

Anna's Goals ✅

- ✅ Generate tests for edge cases and error handling
- ✅ Continue optimizing and cleaning up the code
- ✅ Add code comments and documentation
- ✅ **BONUS**: Added comprehensive error handling for all core classes

Cody's Goals ✅

- ✅ Implement Piece class in current code
- ✅ Define all piece shapes and rotation states in figure.py
- ✅ Write unit tests to verify proper piece initialization
- ✅ **BONUS**: Completed board collision detection methods

Statistics Overview

Lines of Code Added

- **Anna**: 226 LoC total
  - `src/game/board.py`: 95 (error handling & optimization)
  - `src/game/row.py`: 70 (validation & robustness)
  - `src/utils/linked_list.py`: 71 (exception handling)
  - `tests/test_board_core.py`: 88 (comprehensive tests)
  - `tests/test_board_edge_cases.py`: 101 (new edge case tests)
  - `tests/test_linked_list_core.py`: 57 (updated tests)
  - `tests/test_linked_list_edge_cases.py`: 37 (new

Burn Down Rates

Week 4 Performance

- **Anna**: 100% total (3/3 goals completed)
- **Cody**: 75% total (3/4 goals completed, 1 pending)
- **~14% per day** progress rate
- **Strong milestone alignment**

Sprint 1 Progress

- **69% total** (9/13 milestones completed)
- **23% per week** average
- **~3.3% per day** overall progress

Major Technical Achievements

Code Quality & Robustness (Anna)

- **Error Handling**: Comprehensive exception handling for all core classes
- **Input Validation**: Robust validation for all public APIs
- **Test Coverage**: Complete edge case testing with 41 passing tests
- **Documentation**: Clear code comments and API documentation

Piece System Implementation (Cody)

- **Piece Class**: Complete implementation with movement and rotation support

Key Architecture Changes

1. Board Class Enhancement

```python
# src/game/board.py - 95 LoC
class Board:
    def __init__(self, height, width, row_factory):
        self.__height = height
        self.__width = width
        self._rows = LinkedList()
        # Factory pattern for row creation
```

- **Factory Pattern**: Dependency injection for row creation
- **Validation**: Index bounds checking with clear error messages
- **API**: Clean properties for height/width access

2. Robust Error Handling

```python
# Enhanced validation across all classes
def _check_row_index(self, row):
    if not 0 <= row < self.__height:
        raise IndexError(f"Row index {row} out of bounds")

def _check_column_index(self, col):
    if not 0 <= col < self.__width:
        raise IndexError(f"Column index {col} out of bounds")
```

- **Fail-Fast**: Clear exceptions instead of silent failures
- **Debugging**: Descriptive error messages with context
- **Safety**: Prevents invalid operations

3. Piece Class Implementation

```python
# src/game/piece.py - 26 LoC (updated)
class Piece:
    def __init__(self, piece_type, x, y, rotation=0):
        self.piece_type = piece_type
        self.x = x
        self.y = y
        self.rotation = rotation
        self.shape = FIGURES[piece_type][rotation]
```

- **Encapsulation**: Complete piece state management
- **Integration**: Ready for board collision detection
- **Extensibility**: Support for all Tetris piece types
- **Architecture**: Movement and rotation logic moved to board class
- **State Management**: Piece instance passed to board methods for operations

4. Movement & Rotation Architecture

```
# src/game/board.py - 160 LoC (updated)
class Board:
    def move_piece(self, piece, direction):
        # Movement logic with collision detection
        # Piece instance passed as parameter

    def rotate_piece(self, piece, direction):
        # Rotation logic using figures.py states
        # Visual state updated after rotation

    def grid_position_to_coords(self, row, col):
        # Fixed bug causing incorrect piece placement
```

- **Architecture**: Movement/rotation functions moved to board class
- **Integration**: Piece instances passed to board methods
- **State Management**: Visual state updated after each

Testing Coverage Analysis

Test Statistics

- **Total Test Files**: 6 (3 core + 3 edge case)
- **Total Test LoC**: 345 lines
- **Test Coverage**: 100% for new functionality
- **Passing Tests**: 41/41 (100% success rate)

Test Quality Improvements

- **Edge Cases**: Comprehensive error condition testing
- **Exception Handling**: Validation of all error scenarios
- **Integration**: Board-Piece interaction testing
- **Performance**: Fast core tests for development

Code Quality Improvements

Before vs After

```python
# Before: Silent failures and unclear errors
def get_cell(self, row, col):
    return self._rows.get_node_at(row).value.get_bit(col)

# After: Clear validation and error handling
def get_cell(self, row, col):
    self._check_row_index(row)
    self._check_column_index(col)
    row_obj = self.get_row_object(row)
    return row_obj.get_bit(col)
```

Benefits Achieved

- **Reliability**: Fail-fast behavior prevents silent bugs
- **Maintainability**: Clear error messages aid debugging

Sprint Progress Analysis

Completed Milestones (7/10)

- ✅ **Week 2**: Board refactoring and testing (Anna)
- ✅ **Week 2**: Piece class design and implementation (Cody)
- ✅ **Week 2**: Rendering system implementation (Owen)
- ✅ **Week 3**: Line clearing logic and optimization (Anna)
- ✅ **Week 3**: Keyboard input mapping and event processing (Owen)
- ✅ **Week 4**: Edge cases, cleanup, and documentation (Anna)
- ✅ **Week 4**: Piece implementation and collision detection (Cody)

Performance Metrics

Development Velocity

- **Week 4**: 1,242 LoC in 7 days = **177 LoC/day**
- **Sprint Average**: 23% milestone completion per week
- **Quality**: 100% test coverage maintained
- **Reliability**: 41/41 tests passing

Code Distribution

- **Core Logic**: 45% (board, piece, row, movement, rotation enhancements)
- **Testing**: 25% (comprehensive test suite expansion)
- **Starter Code**: 20% (reference implementations and documentation)

Technical Challenges Overcome

1. Error Handling Strategy

- **Challenge**: Balancing robustness with performance
- **Solution**: Fail-fast validation with clear error messages
- **Result**: Improved debugging and maintainability

2. Test Organization

- **Challenge**: Managing growing test suite complexity
- **Solution**: Separated core and edge case tests
- **Result**: Fast development tests + comprehensive validation

3. Integration Preparation

Week 4 vs Week 1 Milestones Analysis

Anna - PERFECT ALIGNMENT ✅

- **Week 1 Milestone (Wk4)**: "Edge cases, cleanup, docs"
- **Week 4 Reported:** ✅ Edge cases + cleanup + docs + **BONUS** error handling
- **Assessment**: **100% on track** - Anna completed exactly what was planned for Week 4

Cody - PERFECT ALIGNMENT ✅

- **Week 1 Milestone (Wk3)**: "Implement Piece + shapes/rotations"
- **Week 3 Status**: Completed in Week 3
- **Week 4 Reported:** ✅ Piece implementation + collision detection + tests
- **Assessment**: **100% on track** - Cody completed Week 3 goals and advanced to Week 4 work

Team Performance Assessment

Strengths Demonstrated

- **Code Quality**: Comprehensive error handling and validation
- **Testing Excellence**: 100% test coverage with edge case validation
- **Documentation**: Clear code comments and API documentation
- **Integration Readiness**: Factory patterns and clean contracts
- **Milestone Adherence**: Perfect alignment with original planning

Team Positioning for Week 5

- **Anna**: Advanced board system with robust error handling ready for integration
- **Cody**: Complete piece system with movement/rotation logic ready for final integration
- **Owen**: Input system complete, ready for main loop integration
- **Overall**: Strong foundation for final integration and polish phase

Sprint Progress Update

Milestone Completion Rate

- **Week 1**: 0/10 (0%)
- **Week 2**: 3/10 (30%)
- **Week 3**: 5/10 (50%)
- **Week 4**: 7/10 (70%)
- **Projected Week 5**: 10/10 (100%)

Velocity Analysis

- **Consistent Progress**: 20% milestone completion per week
- **Quality Focus**: 100% test coverage maintained throughout
- **Innovation**: Advanced error handling beyond basic requirements
- **Team Coordination**: Multiple members contributing simultaneously

Integration Readiness Assessment

Component Status

- **Board System**: ✅ Complete with error handling and optimization
- **Piece System**: ✅ Complete with collision detection
- **Input System**: ✅ Complete with keyboard mapping
- **Rendering System**: ✅ Complete with frame updates

Integration Preparation

- **API Contracts**: Clear interfaces between all components
- **Error Handling**: Consistent exception patterns across modules
- **Testing**: Comprehensive test coverage for integration

Week 5 Focus Areas

Final Integration Goals

- **Main Loop**: Integrate all components into cohesive game loop
- **Movement Logic**: Implement piece movement and rotation
- **Game Over**: Add game over detection and overlay
- **Polish**: Final testing, documentation, and presentation preparation

Success Metrics

- **Functional Game**: Complete Tetris gameplay experience
- **Code Quality**: Maintained test coverage and

Questions & Discussion

Ready for final integration phase?

Any concerns about component compatibility?

Final testing and presentation strategy?

🎯 Week 5 Focus

Priority: Final Integration & Polish

Goal: Complete working Tetris game with all features

Success Metrics:

- Functional gameplay with piece movement and rotation
- Complete line clearing and scoring
- Game over detection and overlay
- Final presentation preparation
- 100% milestone completion