ASE 420 Team Project

Error 404: Name Not Found

Week 8 Progress Report – Tetris

📅 Week 8: Nov 3 – Nov 9

🎯 Focus: Start Screen UI, Scoring System Finalization & Pause Display

Team Overview

Team Members

- **Jeffrey Perdue** – Team Leader
- **Anna Dinius** – Scoring & UI
- **Cody King** – Preview, Pause/Resume
- **Owen Newberry** – Rendering & Controls

Sprint 2 Progress: Continuing feature buildout (start screen, scoring polish, pause UX)

Week 8 Goals Summary

Anna's Goals ✅

- ✅ Display current score in the UI and persist until reset *(completed in Week 7)*
- ✅ Display high score of the current session in-game *(completed in Week 7)*
- ✅ Finalize scoring system unit test coverage
- ✅ **BONUS**: Render start screen with title, controls, and prompt *(Week 9 goal completed early)*
- ✅ **BONUS**: Implement transition from start screen into the game loop *(Week 9 goal completed early)*

Cody's Goals ✅

- ✅ Display pause indicator when game is paused
- ✅ Prevent piece movement and gravity when paused
- ✅ Test pause functionality with all game mechanics

Owen's Goals ✅

- ✅ Adjust piece fall speed based on level
- ✅ Render ghost piece with distinct visual style
- ✅ Update ghost piece position in real-time

Statistics Overview

Lines of Code Added

- **Anna**: 384 LoC total
  - `app.py`: 49
  - `src/game/game.py`: 31
  - `src/ui/button.py`: 29 (new)
  - `src/ui/button_manager.py`: 21 (new)
  - `src/ui/start_screen_layout_utils.py`: 10 (new)
  - `src/ui/start_screen_render_utils.py`: 23 (new)

- `src/view/pygame_renderer.py`: 73
- `src/constants/__init__.py`: 14 (new)
- `src/constants/colors.py`: 15 (new)
- `src/constants/game_dimensions.py`: 9 (new)
- `src/constants/game_states.py`: 3 (new)
- Tests: `tests/test_scoring.py`: 13, `tests/test_session_score.py`: 94 (new)

- **Cody**: 10 LoC total
  - `src/view/input.py`: 2
  - `tests/integration/test_pause_toggle.py`: 8

- **Owen**: 10 LoC total
  - `src/game/game.py`: 2
  - `src/view/pygame_renderer.py`: 26

Total: 422 lines of code

Burn Down & Velocity

- **Anna**
  - Week 8 milestones: 3/3 (100%)
  - Early Week 9 milestones: 2/3 completed
  - Sub-requirements completed (Sprint 2): 9/14 (~64%)
  - Total requirements completed: 10/17 (~59%)
- **Cody**
  - Week 8 goals: 3/3 (100%)
  - Sprint 2 milestone completion: 9/12 (~75%)
- **Owen**
  - Week 8 goals: 3/3 (100%)
  - Sprint 2 milestone completion: 9/12 (~75%)

Major Technical Achievements

Start Screen UI & State Management (Anna)

- Complete start screen with interactive button system
- State-based game management using constants (`START_SCREEN`, `PLAYING`, `GAME_OVER`)
- Visual key binding displays with arrow keys and spacebar images
- Layered rendering with semi-transparent overlays
- Proper state transitions between menus and gameplay

Pause Display & Input Integration (Cody)

- Pause indicator display when game is paused
- Dual-key support for pause (`ESC` and 'p' key)
- Comprehensive pause functionality testing
- Input handling prevents piece movement when paused

Key Architecture Changes

1) State Management System Refactor

```python
# src/game/game.py - updated
class Game:
    def __init__(self, ...):
        self.state = START_SCREEN  # Using constants instead of booleans

    def start_new_game(self):
        # Proper initialization/reset of game state
        ...
```

- **Purpose**: Centralized state management using constants
- **Benefits**: Clear state transitions, easier debugging
- **Integration**: Seamless transitions between start screen, gameplay, and game over

2) Start Screen UI System (New)

```python
# src/ui/button.py - 29 LoC (new)
class Button:
    def __init__(self, x, y, width, height, text, ...):
        # Configurable colors for normal, hover, clicked states
        # Built-in color brightening/darkening effects
        ...

# src/ui/button_manager.py - 21 LoC (new)
class ButtonManager:
    def __init__(self):
        # Centralized button collection management
        # Automated cursor state updates
        ...
```

- **Interactive Buttons**: Hover and click effects with visual feedback
- **Button Management**: Centralized system for UI interaction
- **Layout Utilities**: Reusable functions for centering and positioning
- **Render Utilities**: Standardized rendering helpers for UI elements

3) Main Game Loop Restructure

```python
# app.py - updated
def main():
    # State-based rendering system
    # Improved event processing for keyboard and mouse input
    # Separated rendering logic based on game state
    ...
```

- **State-Based Rendering**: Different screens based on game state
- **Input Handling**: Combined keyboard and mouse input support
- **State Transitions**: Proper transitions between menus and gameplay

4) Start Screen Rendering

```python
# src/view/pygame_renderer.py - updated
def draw_start_screen(self, buttons, ...):
    # Visual key binding displays using arrow keys and spacebar images
    # Interactive button system with hover and click effects
    # Layered rendering with semi-transparent overlays
    # Cursor state management for UI interaction
    ...
```

- **Visual Controls**: Key binding displays with images
- **Interactive UI**: Button hover and click effects
- **Professional Polish**: Semi-transparent overlays and drop shadows

5) Pause Input Enhancement

```
# src/view/input.py - updated
def process_keyboard_event(event):
    # Added 'p' key as alternative pause command binding
    # Both ESC and 'p' keys emit PAUSE intent
    ...
```

- **Dual-Key Support**: `ESC` and 'p' key for pause control
- **User Preference**: Flexible input options for better UX
- **Testing**: Comprehensive pause functionality verification

Testing Coverage Highlights

Scoring System Tests (Anna)

- Finalized unit test coverage for scoring system
- Session score tracking tests (`test_session_score.py`: 94 LoC)
- Expanded scoring tests with edge cases
- Integration checks for score display and persistence

Pause Functionality Tests (Cody)

- Updated test suite to verify pause with both `ESC` and 'p' key inputs
- Comprehensive pause state management testing
- UI rendering verification for pause indicator
- Integration with all game mechanics

Constants Module Organization

New Constants Structure

```
# src/constants/__init__.py - 14 LoC (new)
# Centralized constants organization

# src/constants/colors.py - 15 LoC (new)
# Color definitions for UI and game elements

# src/constants/game_dimensions.py - 9 LoC (new)
# Screen and layout dimension constants

# src/constants/game_states.py - 3 LoC (new)
START_SCREEN = "start_screen"
PLAYING = "playing"
GAME_OVER = "game_over"
```

- **Organization**: Modular constants structure
- **Maintainability**: Single source of truth for game configuration
- **Extensibility**: Easy to add new constants as needed

Week 8 vs Sprint 2 Planning

- **Anna – Ahead of plan**: Completed Week 8 goals and 2/3 Week 9 goals early
- **Cody – On plan**: Completed Week 8 goals including pause display and testing
- **Team**: Strong momentum toward full Sprint 2 completion

Sprint 2 Progress Update

Requirement Completion Rate

- **Week 6**: 5/37 (14%)
- **Week 7**: 12/37 (32%)
- **Week 8**: 22/37 total requirements (76%)

Velocity Analysis

- **Current Progress**: ~20% per week average
- **Quality Focus**: 100% test coverage maintained
- **Innovation**: Advanced UI features beyond basic requirements
- **Team Coordination**: Parallel development with no conflicts

Major Code Quality Improvements

Before vs After

```python
# Before: Boolean-based state management
self.is_playing = True
self.is_game_over = False

# After: Constant-based state management
self.state = START_SCREEN
self.state = PLAYING
self.state = GAME_OVER
```

Benefits Achieved

- **Clarity**: Explicit state constants instead of boolean flags
- **Maintainability**: Easy to add new states
- **Debugging**: Clear state transitions in code
- **Type Safety**: Constants prevent invalid state values

UI System Architecture

Component Structure

```
src/ui/
├── button.py                     # Interactive button class
├── button_manager.py             # Button collection management
├── start_screen_layout_utils.py # Layout calculation helpers
└── start_screen_render_utils.py # Rendering utilities
```

- **Modularity**: Separate concerns for layout, rendering, and interaction
- **Reusability**: Utility functions for common UI patterns
- **Testability**: Isolated components for easier testing

Week 9 Focus (Looking Ahead)

- **Anna**: Complete remaining Week 9 goals (game over screen)
- **Cody**: Continue pause UX polish, expand automated tests
- **Owen**: Continue with difficulty levels and ghost piece logic
- **Team**: Maintain test coverage and UI consistency

Technical Challenges Overcome

1. State Management Refactoring

- **Challenge**: Migrating from boolean flags to constant-based states
- **Solution**: Created game states constants module and refactored game loop
- **Result**: Clear, maintainable state management system

2. Start Screen UI Implementation

- **Challenge**: Creating interactive UI with buttons and visual controls
- **Solution**: Modular UI system with button manager and layout utilities
- **Result**: Professional start screen with smooth interactions

3. State Transition Logic

- **Challenge**: Proper transitions between start screen, gameplay, and game over
- **Solution**: Centralized state management with clear transition points
- **Result**: Seamless user experience across all game states

Code Distribution Analysis

Anna's Contributions (384 LoC)

- **UI System**: 83 LoC (22%) - Button system and utilities
- **State Management**: 31 LoC (8%) - Game state refactoring
- **Rendering**: 73 LoC (19%) - Start screen rendering
- **Constants**: 41 LoC (11%) - Organized constants structure
- **App Integration**: 49 LoC (13%) - Main loop restructure
- **Testing**: 107 LoC (28%) - Scoring and session tests

Cody's Contributions (10 LoC)

- **Input Enhancement**: 2 LoC (20%) - Dual-key pause support
- **Testing**: 8 LoC (80%) - Pause functionality tests

Week 8 Takeaways

Priority: UI Polish & State Management

Outcome: Start screen complete, scoring finalized, pause display enhanced

Next: Game over screen, remaining Sprint 2 items, final polish

🎯 Week 8 Summary

Achievements:

- ✅ Start screen with interactive UI
- ✅ State management system refactored
- ✅ Scoring system finalized
- ✅ Pause display and dual-key support

Progress:

- 59% of Sprint 2 total requirements completed
- 64% of Sprint 2 sub requirements completed
- Ahead of schedule on Week 9 goals