Week 3 Progress Report: Refactoring Board Logic & Implementation

- Dates: 9/22 - 9/28

Week 3 Goals:

✅ Refactored and optimized the line detection and clearing logic
✅ Wrote unit tests for line detection and clearing

Additional Work Completed:

✅ Further optimized the playing field grid by refactoring it to use `Row` bitboards and a `LinkedList` to hold the rows in sequence

Statistics:

Summary of changes under src/

- Files changed:
  - `src/constants.py`
  - `src/game/board.py`
  - `src/game/row.py`
  - `src/utils/linked_list.py`
  - `src/starter_code/tetris_ver1.py`
  - `src/starter_code/tetris_code_explained.py`

Major Changes

1. `constants.py`
   - Purpose
     - Centralizes application constants: screen size, FPS, board dimensions, colors, and cell size.
   - Changes
     - Added board dimensions:
       - `HEIGHT = 20` (number of rows)
       - `WIDTH = 10` (number of columns)
   - Why
     - Provide a single canonical source for board dimensions so caller code (starter scripts) can refer to `HEIGHT` and `WIDTH` instead of local/legacy globals.

2. **`board.py`**
    - Purpose
        - Encapsulates the playing field in a `Board` class using `Row` bitboards and a `LinkedList` to hold the rows in sequence.
        - Provides board operations such as clearing, cell access, and clearing full lines.
    - Changes
        - Board is now fully encapsulated:
            - Constructor
                - `__height` and `__width` are set from `src.constants` (`HEIGHT` and `WIDTH`).
                - `Row` mask initialized via `Row.set_mask(self.__width)`.
                - Rows stored as `Row()` objects in a `LinkedList()` (`self._rows`).
                - Removed error handling statement that checked type of height &

3. **`linked_list.py`**
   - Purpose
     - Simple singly linked list implementation used by board.py to store `Row` objects (one node per row).
   - Changes
     - New file
     - Provides `Node` and `LinkedList` classes with methods:
       - `length()`
       - `append(value)`
       - `insert_top(value)`
       - `get_node_at(index)`
       - `delete_node(index)`

4. **`row.py`**
   - Purpose
     - Represents a single row using a bitmask for occupied cells and a color mapping for occupied columns.
   - Changes
     - New file
     - Stores bits/cells (`__bits`) and a `__colors` dict mapping column indices to colors.
     - Class-level mask `_mask` set via `Row.set_mask(width)`; used to determine row fullness.
     - Methods:
       - `set_mask(width)` sets `_mask` to `(1 << width) - 1`
       - `is_full()` determines whether a row is full by comparing `__bits` to the

5. **`tetris_ver1.py` & `tetris_code_explained.py`**
   - Purpose
     - Starter Tetris implementation.
   - Changes
     - Replaced local/legacy board globals (Height/Width) with the new global constants and `Board` API:
       - Uses `from src.constants import HEIGHT, WIDTH`
       - Draw logic:
         - Uses `GameBoard.get_height()` instead of `GameBoard.height`

           ```python
           for i in range(GameBoard.get_height()):
           ```

         - Uses `GameBoard.get_width()` instead of `GameBoard.width`

           ```python
           for j in range(GameBoard.get_width()):
           ```