Design Overview for “Advance Wars” clone

Name: Jordan Boekel
Student ID: 101625077

Summary of Program

I would like to create a clone of an old DS/GBA game series called advance wards (or Famicon wars, depending on how far back you go). In this game, players take on the role of controlling an army, and must destroy the forces of the other player(s). It is a 2D, top down, tactical turn based game.

(Credit: https://en.wikipedia.org/wiki/Advance_Wars)

I intend to use Splashkit to draw the game map.

Initially, I would like to construct a basic game map and units, and check that this works properly. This would involve units being able to *move* and *attack.*

I will then take any lessons learnt along the way to expand the implementation to include things like money, factories, units that can pass over specific tile types (eg planes over water), turns, etc.

Required Roles

UML provided below

The intent is for the Map class to manage the map of the game. It would tell tiles to draw themselves, which would then tell units to draw themselves. It would maintain a 2D array of tiles, representing the map, and would draw these in the correct order to build a map of the game. I am wondering whether I should give any information to the tiles about where they should draw, or whether this should be kept to the map class (eg. Map passes in coordinates to tiles for drawing every time they are drawn, rather than giving tiles information about where they should be drawn and then continuously telling them to redraw using internal state)

Tiles contain information about game tiles. This would include terrain, movement costs, defence bonuses, things like that. Different terrain types would inherit from this class and

apply the necessary costs and textures. They contain information about what unit sits on the tile. They will also draw themselves. I have added information about neighbouring tiles, to simplify some feature implementations (like pathfinding). The basic tile would be abstract.

Units would be able to draw themselves. They contain information like health, attack, movement points, etc. The basic unit would be abstract.

Mechanically, selecting a unit and telling it to move consists of asking it to move. I'm not entirely sure of the best way to implement this – While the map can see everything, it seems outside of its duties to calculate a route between two locations. Ideally, we tell the unit to move, and the unit then figures out how to get to the required location.

Table 1: Map details

Responsibility	Type Details	Notes
Map: Manage game state	_rows : Horizontal map width _cols : Vertical map width _map[_rows, _cols] : tile : Stores the map layout arrangement. Map() : instantiates map Draw() : Draws the game board. To start with will be called at the end of each game loop to update visual output	
Tile: Manage individual effects	_unit : stores the unit on the tile _location : Location of tile on the map _moveCost : cost for unit movement Tile(location) : Initialises tile with it's map coordinates	A bit of repetition of information between map and tile, will think about factoring it out. Inheritors are different types of tile (eg. Draw different textures)
Unit : Manage actors on the map	_health, _attack etc: Individual unit characteristics, to be initialised on unit creation according to type of unit (eg. Tank) _selected : Indicate if a unit has been selected, and change what it does if so (eg. Draw possible movement shadow) _team : indicate unit team	Inheritors are different types of units. Class is abstract, will only exist as child classes
ICanAttack	Attack()	Defines which units can attack – some don't, such as transport units.
IDraw	Draw()	Defines which types can draw.

Implementation notes and questions for tutorial:

Pathfinding

Unit needs to recursively check where it can move (eg. Movement cost – tile cost for each adjacent tile). Ignore drawing a path for now. Tiles with a unit from a different team present are infinite cost.

If the above returns true, allow unit to move, and reduce movement points available until turn refresh

- Turn refresh
    - Refreshes unit movement
        - Unit movement must subtract after moving
    - Allows unit to attack again
        - Units need hasAttacked property or something
    - Add funds (later)
- Units need teams
- Units need to know where they are in the array to do this themselves.