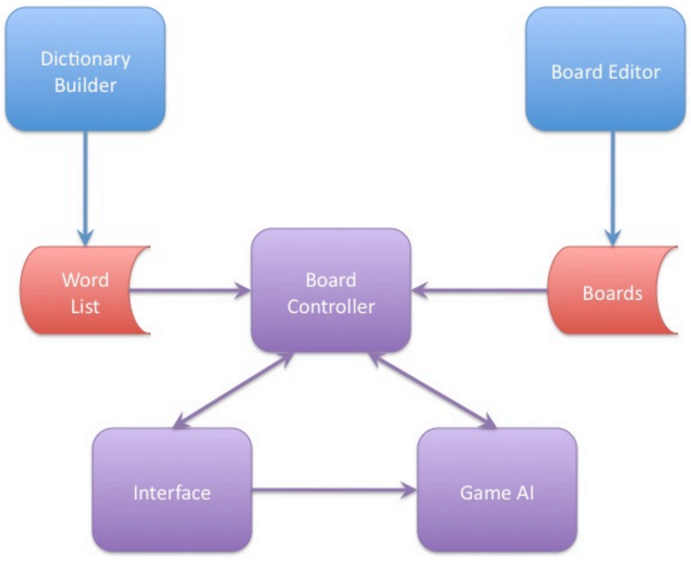
Scryptic Technical Design Document

Version 1.8 – 25 March 2011

The components of the Scryptic system and their points of interaction appear in the following diagram:



The Dictionary Builder and Map Editor are stand-alone systems, which provide, respectively, a set of boards and list of acceptable words to the main application. All other components form part of the main Scryptic game.

Dictionary Builder

This component is a stand-alone program for creating and managing the list of acceptable words. In particular it is designed to automate as far as possible the process of assigning connotations to words. The dictionary builder is based on the SOWPODS list of acceptable SCRABBLE words that is widely used in international Scrabble tournaments. It contains words up to 15 letters in length, including plurals.

- Language – Python
- File Format – the word list is a plain text file, with each word on a separate line in the following format:
  <word in capitals> <zero or more space separated optional specifiers from [A, C, D, S]> <EOL>
  where A = word with attacking connotation, D = word with defending connotation, S = word with stealthy connotation, C = word with chopping or forest destroying connotation. So a typical entry might be: AXE A C
- Functionality –
  - Word lists can be loaded or saved as required.
  - A word can be entered by the user and flagged as having certain connotations (a combination of A, C, D, S). The word is checked to see

if it exists in the current word list and all its synonyms and derivatives (e.g., plurals) will be extracted. In the default case, this new set of synonyms will be given the same connotations as the root word, but the user will be allowed to edit the connotations of the synonyms on an individual basis.

Board Editor

A stand-alone component that allows the design of game boards from both a functional and visual perspective. It reads in a set of images representing different tiles and allows these to be arranged in a matrix format. The visual aspects need to be sufficiently compelling to allow repetitive patterns to be spotted.

- Language – something cross-platform
- File Format – boards are text files with the following content:
  <boardname> <hori> <vert> <overlaytile(0,0)> <overlaytile(0,1)> ....
  <overlaytile(h-1, v-1) <underlaytile(0,0)> <underlaytile(0,1)> ....
  <underlaytile(h-1, v-1)>

  Where <boardname> is a string in double quotation marks providing a name for the board (e.g., "The Plains of Arken"), <hori> and <vert> are integers specifying the horizontal and vertical dimensions of the board, <tile n,m> is a row major list of tile specifiers with the following format:
    - F1, …, F20 – forest tiles
    - P1, …, P20 – plains tiles
    - M1, …, M20 – mountain tiles
    - R1, …, R20 – river tiles
    - I1, …, I20 – ice tiles
    - C0 – unaligned city tile
    - C1, C2 – city tile aligned to either player 1 or 2
    - F0, P0, M0, R0 – special power tiles (e.g. earthquake) located in forest, plains, mountain or river cells, respectively
    - E0 – specifies an empty cell

  There is both an overlay set of tiles (which specifies the initial appearance of the board) and an underlay set (which specifies the appearance in areas where tiles can be removed, such as ice and forest). The underlay is also specifies the appearance of terrain underneath city tiles and power tokens. In all other areas it is set to empty (E0). For each power token there is a power underlay, which indicates the change that occurs to the overlay when that power is activated.
- Functionality –
    - A board can be loaded or saved as required, even if it is in a partially finished state.
    - The user can initialize a board with a specific name and dimensions. These cannot be changed once editing begins.
    - A user can drag and drop tile images into cells on the board, which provides a visual display of the current design. The image (and associated type) in a particular cell can be changed later.

    - The board can be displayed at different zoom levels to simulate the level of detail facility in the main game.

Board Controller

This sub-component of the Scryptic game executes on the iPod Touch and iPhone. It keeps track of the state of the board, manages combat, reports on illegal words and provides possible word combinations for the AI. Its functionality is accessed by the Interface and AI via method calls.

- Language – C++
- File Format – loads the word list and a set of boards. Can save current game status in the format:
  <saved game name> <board name> <EOL>
  PLAYER1 <score> <letter_1> <r_1> <c_1> <letter2> <x2> <y2> … <EOL>
  PLAYER2 <score> <letter_1> <r_1> <c_1> <letter2> <x2> <y2> … <EOL>
  Where <letter_n> is a single capital letter and <r_n>, <c_n> are integers specifying the coordinates of the cell in which it appears. There is an integer score recorded for each player.
- Functionality –
    - Load and Save – the current state of a game can be saved and loaded.
    - Cell queries – a cell can be queried for its specific terrain type and letter content as well as any intersecting words and their status (connotation, connectivity and validity).
    - Word permutations – given a rack of letters, the controller will return a list of valid words, their placements and associated scores. If relevant it will also report on the combat and terrain issues associated with a word. The list of words should also consider possible redeployment of existing words.
    - Combat adjudication – the controller lists words under attack and their combat scores. It also deals with letter removal for defeated words.
    - Letter Removal – should a player's crossword be in an invalid state at the end of their turn, the controller will be responsible for removing letters.
    - Pot management – the controller will keep track of the remaining letters and provide letters for each player's rack.
    - Redeployment – the controller should rule on the validity of a request for redeployment of a word.
    - Score – the controller keeps track of the score of both players.

Game AI

The Game AI is a sub-component of the Scryptic game, executing on the iPlatform. It is responsible for managing the play of the computer opponent. For this purpose there should be at least four levels of AI difficulty – easy, moderate, difficult and master.

- Language – C++
- File Format – The AI does not require any external files. It communicates by method calls with the Board Controller, which will provide the board state and a list of possible word placements.
- Functionality –
    - Word placement – the AI will manage placement of words during the computer's turn, with consideration of both tactical and strategic issues. In doing so it will make calls to the word permutation and cell query functionality of the board controller.
    - Defense – In the event of an attack by the human player, the AI will order the list of its defending words based on concerns such as minimizing losses and retaining a strong strategic position.

Interface

The interface is responsible for managing game display and user interaction. It interprets touch-screen gestures, manages the levels of display detail and makes calls to the board controller.

- Language – C++ and Objective C
- File Format – load GIF images representing board surrounds, icons, tiles, and letters. The board controller will manage other file I/O. It will also load sound files for in-game effects such as tile placement, icon presses, etc.
- Functionality –
    - Interaction – gesture-based controls for letter placement, button selection, board movement and zooming.
    - Display – queries the board controller and displays the current board with consideration given to the zoom level and display position. Also manages the rack and its letters as well as the information displayed in the snap zone.
    - Menu Structure – out-of-game controls for settings such as difficulty level, play mode (computer opponent or pass-and-play), saved games and campaign progress. Also includes animations

Iterative Development

The system will be developed and tested in a number of iterations.

1. Dictionary builder (complete functionality), board editor (without terrain images, using text entry for cells), board controller (without word permuations, letter removal and redeployment). No interface and no game AI. - COMPLETE
2. Board editor (with terrain images), board controller (for player-based game, including redeployment and combat), interface (for player-on-player game mode) – COMPLETE
3. Complete game with Game AI – Timeline below

Art Assets

All of these assets are in a PNG format. Assets that are not finalized marked in red.

Resource Type	Dimensions	Number	Version
Silver Letters	64x64	26	4
Gold Letters	64x64	26	4
Letter Connectors	64x64	8	2
Board Background	480x320	1	3
Commit Sigil (active, inactive)	64x64	2	3
End Turn Sigil (active, inactive)	64x64	2	3
Recall Sigil (active, inactive)	64x64	2	3
Spy Sigil (active, inactive)	64x64	2	3
Menu Sigil (active, inactive)	64x64	2	3
Forest Tiles	64x64	20	2
Mountain Tiles	64x64	20	2
River Tiles	64x64	20	2
Ice Tiles	64x64	20	2
Lava Tiles	64x64	20	1
Plains Tiles	64x64	20	1
City Tiles	64x64	3	5
Power Tiles	64x64	12	3
Overlay Background	480x52	1	1
Message Background	200x120	1	1
Small Button Background (active, inactive)	80x40	2	2
Large Button Background (active, inactive)	135x40	2	2
Long Button Background (active, inactive)	300x40	2	2
Combat Strength Background	200x180	1	1
Radio (Gem) Button (inactive, 4 active)	64x64	5	1
Attacker Gem	64x64	1	1
Defender Gem	64x64	1	1
Instigate Combat Sigil	64x64	1	1
Start Screen Background	480x320	1	1
Small Icon	57x57	1	2
Large Icon	512x512	1	3
Menu Background	420x260	1	1
Strength Modifier Symbols (attacking, defending, mountains, disconnected, city, stealth, weakened)	64x64	7	2
Achievement Icons (easy, med, hard)	150x150	3	1

Audio Assets

File formats should preferably be .wav (but .caf or .aif are also acceptable). The data formats (audio encoding) must be linear PCM (e.g., LEI16) or IMA4. Sound effects are required to be less than 30 seconds in duration. These are restrictions imposed by the Core Audio System Sound framework.

Resource Type	Duration	Version
Coin Placement	0.1-0.3s	1
Coin Lift	0.1-0.3s	1
Attack Start	0.4-1s	1
Attack End	0.4-1s	1
Coin Destroyed (4)	0.1-0.3s	1
Sigil Activated	0.4-0.6s	1
End Game	1-15s	1
Ice Breaking	2s	1
Forest Destroyed	2s	2
Firestorm Power	2.5s	1
Drought Power	2.5s	1
Flood Power	2.5s	1
Volcano Power	2.5s	1
Achievement	2s	1

Achievements

There will be 12 achievements possible, gradiated into 4 x easy (40 points), 4 x medium (60 points), 4 x hard (80 points).

Name	Description	Level
Combatant	Win an Attack	Easy
Novice Strategist	Beating easy AI at a game	Easy
Elementalist	Use a power token outside tutorial	Easy
Spy Master	Use all 3 spies in a game	Easy
Wordsmith	Create a word with score > 15	Medium
Journeyer Strategist	Beating medium AI at a game	Medium
Conqueror	Place words on three corners of a board	Medium
Besieger	Capture an opponent's city	Medium
Sun King	Finish campaign	Hard
Master Strategist	Beating hard AI at a game	Hard
Lexist	Weighted score > 150	Hard
Raconteur	More than 2 attacking and 2 defending connotations in a game	Hard

Game Boards

Board size in number of tiles is matched to the aspect ratio of the display area so that a board fits perfectly when zoomed out to the fullest extent.

Board Type	Size	Quantity
Tutorial	9x6	3
QuickPlay	12x9	5
Short	16x12	3
Medium	20x15	7
Long	24x18	2

Glossary

Connotation – the semantics of a word (attacking, defending, sneaking, forest destroying) that modifies its effect, typically in combat situations.
Connectivity – whether a crossword is properly connected in the sense that there is a continuous path of connected words linking a particular word to a source city.
Validity – whether a word appears in the list of acceptable words. For various reasons a players crossword may have invalid words during their turn but at the end only valid words may remain.
Redeployment – lifting a word from the board into the player's rack so that the letters can be re-used.

Interaction Scripts

REDEPLOY INTERACTION

Algorithm:

```
User presses Redeploy Button
Popup Redeploy Explanation provided
User touches letter [touches on non-letter squares allow conventional
board interactions such as pan and zoom]
Call redeployQuery with letter position to return redeployable words
IF no redeployable words THEN
    popup Redeploy Failure Explanation, allow interaction cancel
ELSE
    Selection dialog from redeployable words, allow interaction cancel
    Call redeployWord with chosen word returning list of lifted letters
```

Board Controller Methods:

- *redeployQuery*: input 2D board position, output list of redeployable words that can be removed without leaving the word network in an illegal state.
- *redeployWord*: input 2D board position and word for redeployment, removes letters from the board that do not intersect other words, output list of removed letters to be placed onto rack.

WORD COMMIT INTERACTION

Algorithm:
(After user has placed a number of letters on the board)

```
User presses Word Commit Button
IF call placeWord with list of newly placed letters returns an invalid
placement code THEN
    Popup Invalid Word Explanation using returned placement code
ELSE
    IF placement code indicates an attack THEN
        Call combatants to get the attacker (and its strength) and all
        defenders (and their strengths)
        Ordering dialog to allow defender to prioritize words
        REPEAT
            resolveAttack between attacker and next word in
            defense list
            Call removeWord on defeated word
            Display outcome of combat in popup and on board
        UNTIL attacker or all defenders defeated
    ELSE
        Change rendering of letters for successful commit
```

Board Controller Methods:

- *placeWord*: input list of letters and their board positions, output placement code (-1, -2, -3 = invalid word with type of error (disconnected, more than one word, not in dictionary, etc.), 0 = valid non-attacking word, 1 = attacking word).
- *combatants*: input index of attacking word, return strength of attacking word and a list of defending words (overlapped or adjacent to attacker) and their strengths.
- *resolveAttack*: input index of attacker and its current strength and index of single defender, output winner and loser status and new strength of attacker (which could be <= 0 if it loses)
- *removeWord*: input index of word, and remove all its letters that do not form part of another word. If this is the attacker then all its letters should be removed. (may be better to incorporate this into the *resolveAttack* method) Return list of letters and locations to allow animation.