Renderer requirements

What is needed:

* Display 2D elements (tiles, sprites), with or without scale and rotation
* 2D (bitmap) font management
* Sprite sheet management
* { Effects – blur, light trail, bloom, fading (alpha) }
* Effects – lightning, sparkling particles
* { Render to texture (for effects) }
* Variable width lines (possibly using polygons)
* Lines with antialiasing (possibly using shaders)
* { Postprocessing, fragment shaders (for effects) }
* Color mapping for glow (that would be AWESOME)
* Lightning – lines (variable width) or sprites (variable size) for each segment
* Support for portrait & landscape orientations
* Support for iPhone 4\* and iPad\* resolutions (960x640, 1024x768)

Separate class for game table management:

* Management of tiles, coloring, clicks, sending, bombing
* Also manages state animations for tiles
* Management of worm (“virus”), power-ups, bombs
* Renders without special effects

Separate class for lightning effects:

* Lightning generation (main strike, branches)
* Lightning transition management
* { Renders to texture }

Separate class for sparkles:

* Sparkles generation
* Sparkles update (basic gravity / physics)
* { Renders to texture }

Separate class for Matrix-like background:

* Create and maintain table with green characters
* Update fading colors, random characters, “droplets” effect
* Renders without special effects

GAME REQUIREMENTS

Game Setup

* Game type
  + Practice (no time limits, some tutorial on various elements)
  + Challenge (levels with time limits, gather charges, upgrade, increasing difficulties)
  + Economic (left sources disappear after launch, can be collected as bonus items)
* **Level progression**
  + Time limit increase { time\_limit [seconds] }
  + Bomb, bonus items increase { bonus\_freq }
  + Target charges increase { target\_charges }
  + Worm starts to appear { worm\_level }
  + Worm increases length { worm\_length }
  + Worm starts to rotate tiles with increasing frequency { worm\_tile\_rotations }
  + Formula for each variable, as a function of { level }
* Difficulty (affects level progression) { ? }
* **Score { + BONUS – BONUS }**
* Bonuses

Level

* Time limit
* Target charges (upgrades) to gather
* Percentage of dead ends
* Worm presence (and length), and frequency of changes to tiles
* Bomb frequency
* Bonus items frequency

Animations

* SEND animation with branching lightning strikes (textured polygons with ALPHA\_ADD)
* BOMB / ARROW animation (explosion frames? Matrix effect?)
* MATRIX effect for background
  + 18x24 characters => 56x42 grid on 1024x1024 surface, 56x32 or 42x42 displayable, depending on screen orientation
  + Add some transparency to tile backgrounds
  + Write big text (certain letters to remain highlighted)
  + Write small text (letters to compose actual text and remain highlighted)

**TODO**

* Move send/bomb buttons (no hard-coding!)
* Move worm management to GameTable class { worm can be destroyed } **DONE!**
* Bonus items { CHARGES, bombs, clocks } put them into GameTable class
* Charges to gather
* Time limit { progression, pause, time display, time up }
* Level progression
* Textured polygons for lightning **DONE!**
* Splash screen with tiles / lightning { ELECTRO PIPES }
* Particle generators { with time limit } **DONE!**
* Large lighting “blobs” around lightning branches
* Lightning updates instead of re-generate
* ROTATING -- tap & drag tile to rotate (also actually rotating a tile) **DONE!**
* WORM -- make it destructible, make it smarter, make it move smoother (slide between tiles), move it to GameTable class **DONE!**
* SEND -- when connecting L-R, you have 48 frames to rotate tiles. If nothing happens, game auto-sends. meanwhile the connecting lightning changes color **DONE! With 96 frames**
* DESTRUCTION -- when tiles disappear (bomb, arrow, send), some lightning appears in place for a bit { Big lightning strikes when sending } **DONE!**
* MATRIX -- [3D] should have depth planes, big letters to small letters -- also you should scroll horizontally through it
* TRANSITIONS between states
* HELP page, menus
* Pause game