So I’m a big fan of turn based Strategy/RPG hybrids or “Tactical RPGS” (<http://en.wikipedia.org/wiki/Tactical_role-playing_game>) . I’ve played through quite a few over the years but a few of my favorites are Fire Emblem (<http://en.wikipedia.org/wiki/Fire_Emblem_(video_game))>, Final Fantasy Tactics Advance (<http://en.wikipedia.org/wiki/Final_Fantasy_Tactics_Advance>), Fallout Tactics (<http://en.wikipedia.org/wiki/Fallout_Tactics>), Silent Storm (<http://en.wikipedia.org/wiki/Silent_Storm>), and most recently Valkyria Chronicles(<http://en.wikipedia.org/wiki/Valkyria_Chronicles>).

Each of these handled the genre slightly differently but at the core you are the Commander of a squad of people and you take turns to explore the world and defeat the enemy. Essentially, it’s a more complicated version of chess.

Looking at these types of games, there are a lot of very interesting technical feats that I think would be great to distill and breakdown to see exactly how they work and more importantly, translate them to the Flash environment.

When you look at the vast majority of Flash games out there, they’re usually quite gimmicky and very shallow from a gameplay perspective. This isn’t a knock against those developers or the game design, I’ve spent many a time kicking small animals across the screen just to watch that magical score number grow ☺ and I completely understand the business realities behind it. A client isn’t going to pay you to develop a large in depth game over 2 years, they want something quick, fun and viral.

Indie developers who do choose to develop their own large in depth game, generally choose a platform other than Flash for the main reason that they can actually sell it. Especially with digital distribution models like Steam and Xbox live, it’s easy for a new/small studio to cut their teeth and get their game out there.

That all being said, I want to see one in Flash. For no other reason than to see if I can, over the coming months, I’ll be creating a brand new Tactical RPG completely developed in AS3 from scratch and documenting the entire process on BreakTryCatch. I intend to release the source of the engine, editor and game at periodic (hopefully stable) milestones to correspond with each post so you can see exactly what’s going on and have a tangible reference for what I’m talking about.

**Design Goals:**

The design goals of this project are as follows and obviously might change as we go through this:

* Create an Isometric (actually diametric but we’ll get into that later) engine and editor with support for variable height.
* Create a Tactical RPG with the editor using the engine.
* Ensure usability and clarity both in the editor UI and readability of the code.

**The RoadMap:**

The roadmap for development will roughly take the following form and jump across the Engine, Editor and Game as different dependencies are worked out.

* **Framework**
  + Identify Challenges and Establish a Framework.
* **Engine**
  + Environments and Actors
  + Texture Mapping to Polygons
  + Camera System
* **Editor**
  + Interactive Builder
  + Merge/Splice/Extrude/Add/Delete/Save/Load
  + UV Editor
* **Engine**
  + Culling (Backface, View Volume, Possible Occlusion?)
  + Depth Sorting
  + Scene Graph
  + LightMaps
  + Dynamic Lighting
  + Shadows
  + Zoning (Water, Fog, etc)
  + Particles and Special Effects
* **Editor**
  + Actor Placement
  + Scripting Support
* **Engine**
  + 3D Pathfinding
  + Collisions
  + Triggers
* **Game**
  + Engine Integration
  + Player Character
  + Player Controlled Actors
  + Turn Based System
  + Enemy Commander AI
  + Enemy Units AI
  + Line Of Sight Detection
  + Audio Detection
  + Squad Tactics
  + Possible Multiplayer Support?

As you can see, there’s a lot of work to be done, and the later phases are much more general and fuzzy at this point, but I think it’s a good starting point and plan to follow. We’ll get right into the code next time setting up the framework and talking about some of the technical challenges I expect to run into.

Obviously, I’ll be coding in advance of the posts, so here’s a peek at Texture Mapping to Polygons. It’s a 512 by 512 (in terms of iso width/depth) with 8 sub sections each way. Each vertex is randomly given a height value between 0 and 32 to give it a bit of a random terrain look. Debugging outlines are turned on.