**Batgai-0 Sonic’s Revenge TDD/GDD**

**Game Design Overview**

* Punch enemies to knock them around.
* Knock enemies into designated scoring areas to score and kill them.
* Don’t touch enemies or score areas (you **will** die).

**Things to Implement**

* Content editor in the form of integration with Tiled map editor. Allows designers to easily create and import new levels.
* Basic AI for enemies.
* Scoring areas and scoring logic.
* Background with depth.
* Ripple shader for punches.
* Death explosion particle effect, trail particle effect for knocked enemies.
* Retro-style sound effects and music

**Stretch Goals**

* Highscore display in level select.
* Different and varied enemy types.
* Background customization integrated into the level editor.

**Risk Assessment**

* The content editor is high risk. I have never worked with Tiled (or any stand-alone level editor), nor have I worked with XML in C#. From the little research I have done, there are plenty of resources on both subjects.
* The AI is medium risk. I have not done any AI in C#, though I have done some in AS3. The AI for the basic enemy type is extremely simple as well.
* The scoring systems are low risk. I have done very similar things many times. I am just going to use simple collision methods for the scoring areas.
* The background with depth should be low-medium risk. The implementation will be fairly basic, so I don’t foresee many problems springing up.
* The ripple shader is medium risk. I was having difficulties implementing it before, so I do not know exactly how long it will ultimately take me to implement. On the other hand, I feel fairly comfortable with shaders in general and I will dedicate ample time to this important shader.
* The particle effects are low risk. I feel pretty comfortable with particle systems, and the particles I want to implement are not too complex.
* The sound effects are low-medium risk. I do not feel that the sound and music will be too difficult to implement, the game does not rely on synchronization with the music or anything like that. On the other hand, I have never dealt with sound in XNA.

**Timeline**

**Week 1 (10/30/13) - (11/06/13)**

* Scoring logic complete.
  + Multipliers.√
  + Display current score and multiplier.√
* Basic enemy AI complete.
  + Basic enemies come directly to player.√
  + Player dies on contact with enemies or scoring area.√
* Some sound effects implemented.
  + Make sounds in BFXR.√
  + Implement sound effects.√

**Week 2 (11/06/13) - (11/13/13)**

* Death explosion particle effect and knock-back trail particle effect complete.√
* Research map editors
  + Find out how importing maps would work.√
  + Find out how backgrounds, spawning, collision works with imported maps.√
  + Choose map editor.√
* Implement map editor (if possible)
  + Get basic map editing and importing working
  + Configure everything to make map creation and importing simple

**Week 3 (11/13/13) - (11/20/13)**

* Level Select Screen roughly working.
* Implement map editor (if not yet implemented)
  + Get basic map editing and importing working
  + Configure everything to make map creation and importing simple

**Week 4 (11/20/13) - (11/27/13) -- Thanksgiving Week**

* Ripple shader for punches implemented.
  + Create shader.
  + Attach shaders to each shot
* Backgrounds with depth implemented.
  + Implementation depends on level creation implementation

**Week 5 (11/27/13) - (12/04/13)**

* Finish potentially uncompleted work which was allocated to Thanksgiving week.
* Finish implementing sound effects, implement some music.
  + Create or edit sound effects which were lacking
  + Choose music for levels and menu (depends on level editor implementation)

**Week 6 (12/04/13) - (12/12/13)**

* Finish up everything from previous weeks which is not yet finished…
* Implement highscore display in level select (if possible).
  + Save high scores
  + Display high scores in level select screen
* Create different enemy types (if possible).
  + Create different AI types.
  + Differentiate enemies by sprite, particle effects, powers, etc...