Senior Project Proposal

The Legend of Zelda – 3D

The Problem:

The idea is to fully re-implement the Nintendo's original The Legend of Zelda game from 1987 in a fully 3-Dimensional format using a modern game engine.

New features will hopefully include complete environmental effects (sun rise/set, weather etc.) a deeper and more animated combat system, and if time permits, a "third quest" completely designed by me.

The Engine:

- Torque Game Engine
  - Produced by GarageGames (garagegames.com)
  - Supports Open/GL and Direct-3D

The Programming Language:

- C++
- Torque Scripting Language
- Open/GL

This choice is dictated by the chosen core engine technologies. The engine itself is written in C++ so any "low-level" changes to it must be made in the same.

The majority of a Torque game can actually be written in the C style "Torque Scripting Language". These files are compiled at run-time by the Torque Engine and are more programmer friendly than hacking into the real engine code.

Torque also abstracts the choice of Open/GL or Direct3D but I have been testing in Open/GL because it seems to be performing >-= D3D.

The Platform:

- Currently developing on windows due to time/budget/technology constraints.

- The Torque Engine is Cross-Platform (*nix, Mac, Windows) so in theory any resulting product should be cross-platform.
- The decision to primarily support Open/GL should maximize the chance of cross-platform compatibility.

The User Interface:

- The game will be played out in a fully 3D environment with an outdoor Overworld and 9 Underworld Labyrinths.
- The game will use standard FPS(First Person Shooter) Keyboard controls but will hopefully support Playstation-styled Logitec USB game controllers.
- The game will primarily be played with a third-person "chase-cam" view of the main character.
- First-person views to look around the environment should be available.
- Whenever possible the UI-functionality will be made to resemble/match the UI for the original game.
- For UI-functionality pertaining to 3-D aspects, Playstation2 games will be a model.
    - Prince of Persia – The sands of time (1st)
    - Grand Theft Auto – San Andreas (2nd)
    - Star Ocean – Till The End of Time (*what not to do)

The Approach:

- Torque will be used to define the Overworld environment, the GUI and tie the entire project together.
- 3D-Studio Max, Maya, or similar will be used for character modeling and animation.
- Blender will be used for modeling DTS objects, aka. Trees, rocks, buildings (including Labyrinths) and other large objects/structures.
    - Underworld Labyrinths will be set in individual Torque Levels with a non-descript terrain. The labyrinths will be the proverbial 'Room with no entrances or exits' and the spawn point will be in the first room of the levels.
    - Torque Level switching will have to be used to go between the Overworld and Underworld. Commercial games suggest the involved load time is a socially acceptable “cost of business” but perhaps a pre-loading optimization is possible to speed this up.
- GIMP is the photo-editor of choice for all artwork.
- I have begun building a library of photographed textures. Hopefully by running these through GIMP, I can (painlessly?) create suitable textures for the game.

The Details:

I have already started defining the Overworld environment. Setting up dynamic weather and lighting conditions has proven to be a great starting point for learning the vast Torque Engine.

Many low-level functions for environmental effects come built in. There are several TSL(Torque Scripting Language) Mods from the Torque-community that abstract these functions to create feature filled dynamic-systems for managing the environment in a Torque world.

Since some of the community resources are out dated, I'm hoping to re-package my enhanced versions of the mods and re-release them to the community. The weather system, for example, I had to implement about a dozen bug-fixes and features already listed in the resource's comments on the web-site. I found at least one more bug and have already added at least one new feature that is wholly my own.

I am also considering writing a tool to help place objects on the 3-D map from a 2-D top-down perspective.

Lastly I may attempt to write a quick tool to generate realistic trees based on older works I've seen. (Specifically have in mind: *www.infa.abo.fi/~jweeriks/treegen/*)