Open 2Dimensional Development (o2d) Project

Cory Petosky
Department of Computer Science
Hood College, Frederick MD 21701

Problem Description

Arguably the most popular hobbyist tool for creating top down perspective video games is RPG Maker XP (http://www.enterbrain.co.jp/tkool/RPG_XP/eng/); however, this tool comes with several significant limitations. First and foremost, it is useful primarily for turn based role playing games, as it is designed only for this purpose. Secondly, it is designed for a Japanese audience; the English support for the tool is minimal, at best. Thirdly, it runs only on Windows 98/NT; it does not run on any variant of Linux, BSD, Mac OS, or even on Windows Vista. Finally, there is a $60 license fee to use the program, which both prevents widespread adoption by hobbyist game developers but also results in significant piracy, with a variety of incompatible cracked versions of the original tool currently being used across the world.

The o2d project intends to provide a compelling, open source replacement for RPG Maker. Compatibility with RPG Maker resources and similarity to the mechanics of that development environment will be retained unless there is a good reason to do otherwise. Point by point, the o2d project will:

- Provide an intuitive IDE to create multilayer, tile based maps with full transparency support.
- Allow simple editing of all game resources, including tile palettes, events, and entities.
- Integrate with the Ruby scripting engine to allow complex user created
- scripts for games.
- Allow users to import resources from RPG Maker XP, when the format for these resources are known.
- Run on any platform with the project dependencies installed.

Proposed Work

The o2d project will be written in ANSI C++, so as to provide maximum compatibility with all hardware. The target compiler is g++, though as no warnings will be allowed in the final code, it should work with all standards compliant C++ compilers. The standard library will be used extensively. The o2d project depends on the Simple Direct media Layer (http://www.libsdl.org) and the related SDL_Image and SDL_TTF libraries for the game engine. The development environment, however, depends on gtkmm (http://www.gtkmm.org), the C++ wrapper for GTK+, a popular cross platform windowing environment.

Finally, the o2d project depends on libxml++ (http://libxmlplusplus.sourceforge.net/) for XML handling capability. Because two different renderer libraries are used, some effort is involved in writing the core classes in a renderer neutral manner. An abstract rendering API must be written to facilitate this – after that, specific renderer classes can inherit from the abstract API to support the rendering engines in question. A side effect of this design is that, in the future, other rendering engines could be supported with a minimum of effort. Scripting is probably the biggest single task that needs to be implemented. The editor should maintain a database of scripts that can be used as map based triggers, AI behaviors, or event callbacks.

The Ruby interface should provide accessors and mutators to game data, which are backed by C++ objects and methods. Most other functions, however, should be as pure Ruby as possible, so that the user can easily edit default behaviors. The integration of the scripting engine to the C++ code base to be fairly straightforward; however, sandboxing the Ruby engine, both to ensure my C++ object integrity and to prevent game scripts from running wild on user's machines, will probably take some time. Map editing is he second biggest challenge. Maps should have three editing layers, to facilitate overlaying tiles upon each other. However, when rendering the maps, these layers are almost irrelevant—instead, the *priorities* of the individual tiles should be taken into account.

Tile priority, stored in the tile database maintained by the editor, is essentially an integer representing the relative height of the tile to the ground. Thus, while ground tiles would have priority 0, the guardrail on a bridge might have priority 1, so the game entities would appear behind it. Vertically higher tiles, like the branches of a tree, might have priority 3 or 4, so even big entities appear under the image. The game engine itself is the third challenge in this project. The engine must be able to handle a significant number of entities on screen, all of which will be running their own scripts, and all of which might have their own graphics. The engine must bind button presses to scripts created in the editor, and of course it must handle any other script that should take effect in any given location.

Additionally, it has to handle the smooth loading of adjacent map files, so that the world can appear more or less seamless. Collision detection will be easier than in most modern games, as all my resources are two dimensional. Finally, the o2d project editor must handle game databases for each game project a user wishes to work on. Beyond a minimal set of standard graphics and sound, it must maintain the states of all the individual maps, scripts, images, sounds, tiles, palettes, and variables a user creates to make his game. These databases will be stored via XML files in a predefined directory structure, to make them easy to maintain and import by other tools in the future.