29/09/2017

PG5200-1 17H Tools programmering

Game Editor - Post-Mortem 1

Project

The editor is meant for the creation of a world which could be used for an adventure or role-playing game. The world is divided into maps, each of which consists of 8x8 tiles which all have at least one graphical sprite asset (no empty tiles). The editor should contain:

- A world editor **//To do**
- A map editor **// WIP**
- A character / creature editor **// WIP**

Structure

The editor is currently divided into tabs (one tab per tool), which will probably be subject to review before long as it is not very user friendly. Each tool is separated in its own XAML and is designed according to the MVVM design pattern through WPF and C#. For now, a lot of the view is very much directed by the code-behind, which is against how WPF is supposed to be used. The only reason for this is because I'm not yet sure how to make efficient UserControl / ItemsControl templates and components to encapsulate the data-to-view flow **all the while** keeping control of data flow between components. But my aim is to have the code-behind only handle events in the view, and link to the backend (asset loaders, DB / file managers, etc…).

I focused most of my efforts on the Map editor and tried to have a hierarchy of Tiles, each of which have a dedicated role. I have tried to design the hierarchy with separation of concerns in mind, but I am worried that this might lead to code rigidity because of an immovable inheritance tree of Tiles. I have thought about using the Decorator pattern, where each tile could have several roles at the same time (eg: a tile that can trigger both an enemy to spawn and a door to close).

I might change how the tiles are stored. At the moment, I feel that there is too high coupling between the Map editor and the Tile manager, and way too much responsibility given to the MapEditorControl.

In the next versions I would like to allow for characters / entities created within the editor to be directly available for placing in the Map, but that will be once I overcome the issues with Tile loading, and after I have created a basic World editor so that maps can link together.

Challenges - feedback request

- I would love some pointers on how to manage scope for design steps, because I feel like I’m drowning in a feature-creep nightmare while trying to learn WPF at the same time. I feel like the solution is an event-based architecture which would keep code separated in tight small classes, but I don’t have much experience with custom-made event design patterns.
- I would also like to know how I can go about designing custom behaviour for Tiles in a serialisable fashion - by which I mean being able to save a tile’s type, coordinates, arguments and action inside the JSON file.
- Any feedback on anything particularly horrible or great in the present solution would be awesome as well, and what could (should?) be removed or added.