Games as Research / MDM

stands for Method for Design Materialisation
stands for Materialising Design Moves
stands for Mapping Design Moments
etc.

Summary

The method uses the affordances of Git to track the design process of a project from initial concept through design through development (and iteration) through testing to release, support, and reception. Perhaps most centrally, Git commits are used not just to keep track of technical progress in the implementation of the project, but to track the design thinking behind each “move” made in the project. This is alongside a Process Journal that is used to capture more longform reflections about the project direction as a whole.

Requirements

- Basic knowledge of and comfort with Git and associated technologies. A typical constellation of tools might be SourceTree as a local client with GitHub for hosting, for example. If you don’t use or understand Git, it’s imperative to learn before working with MDM.

Components

- Most fundamentally the method requires the use of a *git repository* as the core of a project. This repository will store the technical matter of a project (e.g. source code, assets, release builds, etc.) but also the processual information about the project as well. The repository will, ideally, be public rather than private (at least upon release). Example: Tanks!?

- A *README* at the base level of the repository that explains it in language suited for someone who is not overly familiar with Git, programming, or MDM. It should link directly to key aspects of the documentation. Example: Tanks!?

- Within the repository there should be at least a *process journal* document that is used to regularly write diary-style entries that involve reflection on the status of the project, references, images, and anything else that captures the normal

elements of design and development as it happens. This can and should be written in a fairly casual language to avoid artificiality and avoidance. We recommend writing this documentation in Markdown or a similar plain-text based format to avoid fiddling with stylistic details. Services such as GitHub and Bitbucket provide wikis as part of repositories which is a fairly straightforward way of including this sort of document. Ideally you would include screenshots and other embeddable media in this journal to illustrate points. Example: Tanks!?

- A possible inclusion (with permission) is any email correspondence or other capture-friendly *conversations* you have about the design and development of the project. This may be especially helpful in team-based projects where discussions of design move fast and have multiple stakeholders, making it more difficult and inconvenient to write singular process journal entries that capture the debate. This is also possibly true of more casual playtesting of the project. Example: Sibilant Snakelikes

- As per practices by artists such as Robert Yang and the theatre group Coney, we are also recommending the presence of a *manifesto*, *artist's statement*, *research statement* document in the repository. This should be a statement of the purpose and nature of the project and should constantly revised during the work. This is vital in part of accountability to a larger vision, but also as a place to work out what that vision is as it inevitably changes through the design process. Example: It is as if you were making love

Workflow

The basic *commit-based workflow* is as follows:

- You perform some *defined block of work* on the project. In early times this may be a writing a process journal entry, a draft of the manifesto, including reference images, etc. When development begins this would also often refer to a specific block of implementation with an intelligible outcome and (ideally) a working build of the game.

- You write a *commit message* for this block of work which should consist of. *A brief explanatory title* that specifically states what the work is at a technical/basic level. *A design reflection* that involves writing about how the work just performed relates to the design of the project (especially with reference to the manifesto/artists's statement/research statement). When work was of a technical nature, this implies you need to reflect on how the technical informs and is informed by the design/theoretical/conceptual. Example: Sibilant Snakelikes

- **Note:** There are many personal styles involved in when to commit and how to write commit messages appropriately. The concern within MDM is for legibility of the design process within the repository's commit history. Ideally, all commits would therefore involve some design-relevant reflection, but this is not necessarily realistic. As such, consider a highly visible notation preceding commits that do have design reflections in order to signal them in the commit history. Likewise, it could be good to avoid a style that involves many tiny commits. Another approach might be to maintain a "design branch" in which you only commit design-relevant commits such that this branch could be browsed as a reflection of design process.

- You *commit* your work to your local repository and *push* it to your remote repository, creating a specific *moment of commitment* that is recorded in your repository as: the diff representing the work done itself, the related build of the project (if there is a build at this point), and the design reflection represented in the commit message.

At the end of the project you *make your repository available* as a direct reflection of and reification of the design and development process of the project. There would therefore be:

- the *final release* of the project itself,
- the *commit history* of the project (including reflective commit messages for each iterative build),
- the *finished process journal* reflective a unified and coherent story of development,
- records of *conversations* about the design and evaluation of the project,
- the *final artist's statement/manifesto* reflecting your end-point in terms of the research question.
- We also suggest writing and including a *press kit*.

As part of finishing the project we recommend the writing of a *final reflective essay* that is either in the process journal or perhaps separate from it and easily accessible to a reader (linked from the README). The intent here is to provide a higher level interpretation of the process followed (possibly with links to relevant builds, commits, diary entries) for those who do not necessarily have the time to investigate the full repository.

Branches?

At present our ideas around using branches as part of MDM are underdeveloped. We perceive a clear opportunity to use branches as a way to potentially reflect other trajectories in design space than the more linear progression to a final release. We imagine that a disciplined developer could start a new branch to explore a major design reflection and then either merge it to the master branch if it works out and

otherwise return to the master, leaving the exploratory branch in place as documentation of that design (and development) foray.

Accessibility?

Git? We have concerns about the accessibility of Git repositories for non-expert audiences, such as the hypothesised academic who wishes to use the unprecedented access to design and development process to perform analysis. The base-line response here is to make sure that a) when linking someone to the repository there is at least some explanatory text to prepare them for it, and b) the base-level README in the repository is written to facilitate navigation for non-expert users.

Builds? This is especially true/problematic in the case of playing the builds associated with specific commits, as such an undertaking requires actually downloading the repository itself, reverting to specific commits, and then building the project at that point. Projects in interpreted languages such as JavaScript are fractionally simpler in that there is no step of building the project required, but likely still remain outside the reach of non-experts. We have in mind a tool to facilitate this process, but there is no question this is a major challenge.

Why?

- To effectively document videogame design and development we need to *capture design moments as they happen* rather than try to predict them (as with early design documents) or recall them (as with postmortem approaches to design writing).

- By using Git we are able to capture *playable builds* of the project at each moment of “design and/or technical commitment” in its history. This means that later when we need to analyse or think about the project, all of its previous states are recoverable and thus usable as evidence in a larger argument about or from design.

- The “*moment of commitment*” in which we chose to commit and push a change to our version control repository represents a welcome opportunity to explicitly reflect on our design work throughout a project. This is true when the commit is of early design ideation, but it is especially advantageous once commits represent implementation work, as they invite us to make explicit what Schön called our “conversation with materials” - we can write about the commit in design terms and with specific reference to the relationship between technical work and design work.

- This leads to a key *formal process* of design and development that encourages/demands good habits when we work on our projects in that we

simple *must* include design reflection with each block of work we do, making it possible even for those of us who are forgetful about such practices to be encouraged to do it. This is supplemented by the process journal.

- There is a huge *dissemination* win involved in this approach because our entire design process becomes available as a trace of the trajectory our work takes through a design space. Arguments can thus be formed on the basis not just of the final artefact but based on the entirety of the process.

- As academic practicing research-creation/research-through-design/arts-based practice/etc. we must be *accountable* in reporting and carefully examining our creative process rather than offering post-hoc explanations of our work or, worse, no explanations at all.