A better recipe for game jams: using the Mechanics Dynamics Aesthetics framework for planning

Paris Buttfield-Addison
Secret Lab Pty. Ltd.
Hobart
Tasmania, Australia
paris@secretlab.com.au

Jon Manning
Secret Lab Pty. Ltd.
Hobart
Tasmania, Australia
jon@secretlab.com.au

Tim Nugent
University of Tasmania
School of Engineering and ICT
Tasmania, Australia
tim@lonely.coffee

ABSTRACT

This position paper explores the application of the Mechanics Dynamics Aesthetics (MDA) framework to the game ideation and design process, as employed during time-limited game jams and hackathons.

MDA is a framework for understanding games that attempts to interlink the related disciplines of game design, game development, game criticism, and game research. MDA suggests that the components of a game can be broken down into mechanics, which describe processes and actions available in a game, dynamics, which describe the dynamics of actions reaction to each other in game, and aesthetics, which describe the emotional responses evoked in the player by the game.

This paper describes how MDA can be applied in a game jam context in order to focus the design and ideation process, and direct team activities into building a cohesive, playable end result.

Keywords

Game jams; game design; hackathons.

1. INTRODUCTION

In this paper, the authors report on their experiences over the course of six separate game jams[1] in which the authors, in combination with other participants, have consciously arranged the early stages of production in such a way that individual groups focus on the mechanics, dynamics, and aesthetics of the game.

The Mechanics, Dynamics, Aesthetics (MDA) framework by Hunicke, LeBlanc and Zubek [2] describes a separation of components in an operating game:

- Mechanics are game rules and logic, as devised by the game creator
- Dynamics are the observed results of the mechanics interacting with each other and with the player during gameplay
- Aesthetics are the qualitative impacts upon the player that are felt as a result of participating in the dynamic system, and reinforced by the game's art, sound, and other ”polish” elements

While the MDA framework is primarily intended for designing and working with the gameplay design of a game[2], when rapidly constructing a game in the context of a game jam, the authors have found it useful to take inspiration from MDA in multiple other areas than gameplay. In particular, we have found it useful to use the underlying philosophy of MDA—that is, game rules are separate to game behaviours are separate to game perception[2, 5]—in the division of responsibilities in teams, in the way that the game's rules are devised, and in the way that the game is built.

Using MDA provides game jam teams with a more coherent process to follow at the start of the game jam, and allows them to generate a straightforward plan of development that can guide them to the conclusion of the jam. Other research has investigated the process undertaken during game jams [4, 1]; however the use of the MDA framework to guide the team dynamics in a game jam—as opposed to as a game *design* tool in a game jam—is, to the authors' knowledge, unexplored.

This paper proceeds as follows: first, we discuss the division of roles, taking inspiration from the MDA framework. We then discuss the use of MDA in design at a conceptual level, and then at a more practical lower level during the technical creation of the game, while exploring the details of what happens in a team that follows an MDA-conscious structure. Finally, we discuss the strengths and limitations of the MDA-based approach to game jams, and report on our conclusions.

2. MDA-INSPIRED ROLES

In a game jam process drawing from MDA, almost immediately after the game jam begins, the team divides up into groups, such as the following:

- Programmer(s), who work on game code
- Game designer(s), who work on gameplay
- Artist(s), who work on visuals and audio

[1]GovHack Australia 2013-2015, TasJam 2015, TasJam Health 2015, and the Qantas Codeshare Hackathon.

GJH&GC '16, March 13 2016, San Francisco, CA, USA
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4083-0/16/03...$15.00
DOI: http://dx.doi.org/10.1145/2897167.2897183

The fact that game programmers do not take on the responsibilities of game design is key, for two reasons: 1) it improves the inclusivity of the team, by allowing members who do not program or produce art to work on the project; while game design is a specialised skill, it is intuitive and non-technical enough to allow participants with no prior game development experience to contribute at some level, often under the guidance of a more experienced designer, and 2) it allows for the development of any code that the programmers anticipate will be needed regardless of the specifics of what the game designers finally decide upon.

The events being drawn on for this paper all involved groups of 4 or more, though there are typically fewer programmers—often only a single person—due to the overhead of programmer collaboration, and a lack of willingness that the authors have observed, in both other teams and in themselves, to devote much time to ancillary tasks like preparation of infrastructure to reduce this overhead.[2]

Implementation almost always takes the most time, followed by art and game design; however, art and implementation cannot effectively begin until game designers have completed sufficient work for the other team members to use. It is common for game jam teams to begin preliminary work while waiting for game designers to provide their designs.

For example, when a team decides to make a card game, programmers can immediately start working on code that allows for manipulation of cards, decks, drag-and-drop, and so on, without knowing any specifics about the card game itself will be. At the same time, artists (both visual and audio) are able to devise an initial concept of the game's visual and aural tone of the game, by creating a representative sample of the game's sound design and music, and by creating concept art that is indicative of where the developers see the game ending up.

3. MDA FOR DESIGN

With the basic division of roles established, the paper can now proceed to discuss how the MDA framework can be applied both at a conceptual level, and in more detail at a practical level.

At the conceptual level, the MDA framework is useful in assisting the decision-making process for what the game can contain; at the practical level, the MDA framework is useful in determining the responsibilities and tasks of the individual members of the team in a time-constrained environment.

The developers and players of a game do not experience the game in the same way: the developer is more focused on mechanics, and observes the dynamic and aesthetic effects on the player, while the player is primarily focused on the aesthetics of the game as they are evoked by the dynamics that result from the game's mechanics.[2].

The developer is intimately involved in the selection and implementation of game mechanics, in the form of the game's source code and user interface. It must be remembered that 'mechanics' means not only high-level decisions, but low-level ones as well. When developing a first-person shooting game, for example, the developer must decide whether the player's health is automatically restored over time or not, which is a high-level, structural decision; the developer must also decide the speed at which health is restored, which is a lower-level decision.

When the game is played, both the high-level and low-level selections of mechanics result in different game dynamics. In a game that has automatic restoration of health, the player will be less conservative, because they are free to repeatedly risk taking damage in the game without making future play more perilous. If automatic restoration of health is available, recharging more slowly will result in more cautious play than rapid recharging.

These dynamics have their aesthetic effects on the player. When cautious play is forced by the game's mechanics, players will feel more apprehensive when confronted with combat; when incautious play is permitted (or even encouraged), players will feel more reckless and powerful.

These dynamic and aesthetic effects are only influenced by the developer in an indirect manner. As a result, the primary view of the game from the developer's perspective is on the mechanics. The player, however, interprets a game from the opposite direction: a player who plays a game and feels apprehensive does so through the game's dynamics, which were set in place by the developer. Therefore, when considering how to build a game, it behooves the developer to first consider the game's aesthetics, and work backwards through the game's dynamics to arrive at the game's mechanics.

The MDA framework is subject to continued contemporary debate; Frank Lantz [3] criticises the term "aesthetic", suggesting that it is too closely linked to visual concepts, and instead suggests "affect" as a replacement; in the same article, Lantz further criticises the choice of the word "mechanics", and argues that game designers have a very different and much more precise definition of the term than even the original MDA paper's [2] own definition provides. However, despite its flaws, MDA provides a useful lens for thinking about the various parts-in-motion that comprise gameplay.

It is important to note that this approach to game mechanic selection applies to all mechanics. One does not begin the process with the goal of making the player feeling clever, and work backwards to arrive at "card game". The approach does apply after a very high-level architectural decision has been selected, typically by selecting a genre such as "fighting game", "top-down strategy game" or "point-and-click adventure game".

4. MDA FOR TASKS

Games are rarely designed for the benefit of the developer, but rather for the player. Because the player perceives the game through the lens of its aesthetics, the selection of which aesthetics should be evoked by the game makes for a useful starting point for the game developer.

The first goal of a team participating in a game jam is to agree on the high-level structure of the game. Typically, this is done by deciding on a game genre, which serves as the foundational set of metaphors and norms that both players and developers can understand and build from. In game jams that the authors have participated in, this involves the entire team; after a decision has been made, team breaks into the three groups described earlier, with each group assigned specific goals of refining the game's content.

This aligns closely with Zook's [6] study of inspiration sources for game jams, which found that the majority of

[2]This raises another topic: how can necessary infrastructure for collaborative game design be rapidly provisioned so as to provide maximum utility while minimizing distraction from the more enjoyable work of actually building the game?

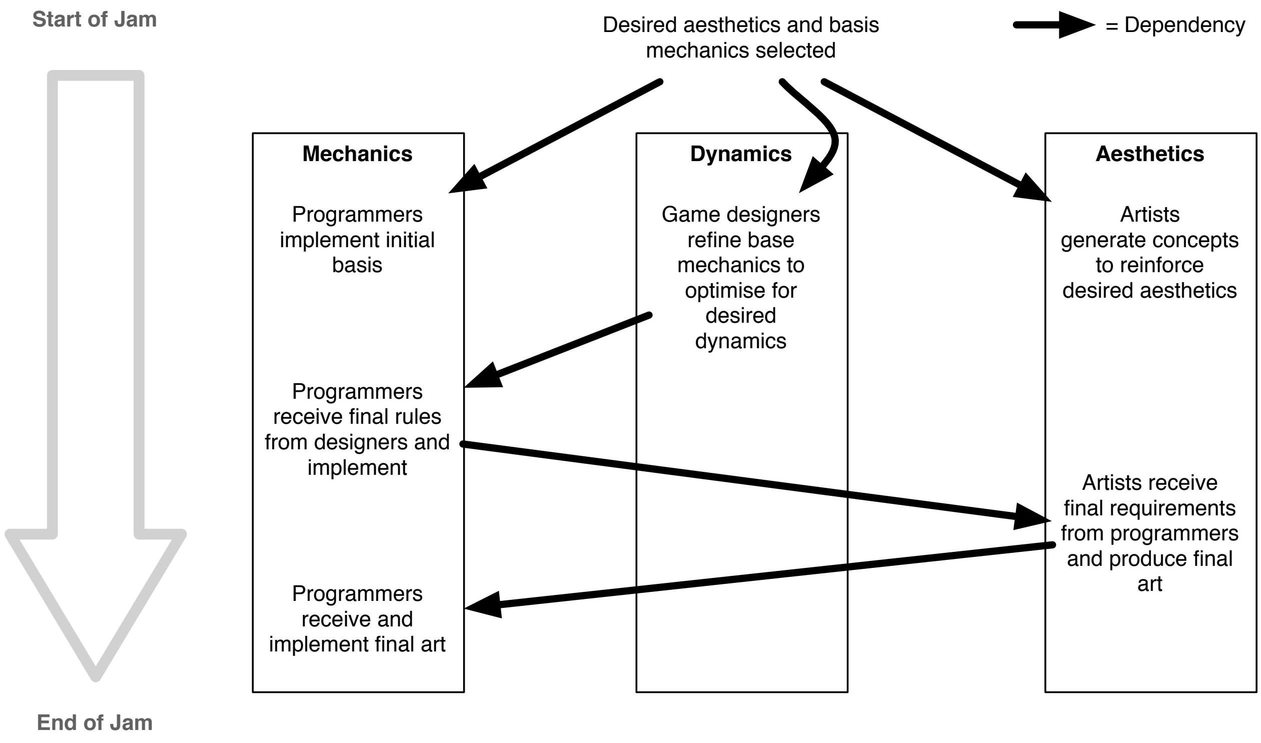
Start of Jam
End of Jam
Desired aesthetics and basis mechanics selected
= Dependency
Mechanics
Programmers implement initial basis
Programmers receive final rules from designers and implement
Programmers receive and implement final art
Dynamics
Game designers refine base mechanics to optimise for desired dynamics
Aesthetics
Artists generate concepts to reinforce desired aesthetics
Artists receive final requirements from programmers and produce final art

Figure 1: A visualisation of the team structure and process described in this paper.

interviewed jam participants start their game from a desired theme.

To reiterate, the game designers at a jam are responsible for devising the game, and providing rules for the game to the developers and artists. The developers are responsible for implementing the game as it has been designed, while the artists are responsible for reinforcing the game's aesthetic impact through the use of visual and aural media.

This division of labor is not only based on relative skill sets; rather, it is done with a conscious understanding of the MDA-driven design process of the game. Immediately after the style and high-level approach have been decided, a rough aesthetic is agreed upon by the team. The team's groups then focus on improving as much as possible the mechanics, dynamics and aesthetics of the game:

- The team's programmers focus on the foundational mechanics that will underpin the more complex and specific mechanics developed by the game designers.
- The team's game designers develop the complete set of game rules from the foundational mechanics decided upon by the entire team at the start of the jam, with an eye to creating interesting dynamics that reinforce the desired aesthetics.
- The team's artists refine the desired aesthetics from the necessarily broad overview that was selected at the start of the jam.

Figure 1 shows a visual representation of this team structure and process.

The bottleneck at this point is the game's designers; while the programmers and artists can perform useful work in the service of the game's development, they cannot produce the final game until they have the final set of game mechanics from the designers. However, by splitting the team into groups that focus on the broad mechanics, central dynamics and key aesthetics of the game, the production process is able to make the best use of the available time to make the game.

The downsides to this approach are that is does require a great deal of trust amongst the team to ensure that a product matching the initial agreed upon style and aesthetic occurs due to the groups working in a highly independent manner and by siloing the members of the team this approach also puts limits, albeit small ones, on the potential for emergent design after the initial aesthetic is determined.

5. STRENGTH AND LIMITATIONS

The authors have observed both strengths and limitations while applying MDA to our team formation in this manner.

5.1 Strengths

One of the first tasks of any game jam team is to determine what tasks each member of the team should perform. These generally fall along skill-based lines, in which members who can code cluster together and form a programming team, while artists form an asset-producing cluster. However, doing this does not provide guidance on the specifics of what each group of team members should be doing.

When a team has followed the structure outlined in this paper, all members have had a very clear task in mind from the very start of the jam. Occasional moments of dead time do occur—for example, when programmers run out of foundational tasks to complete while waiting for the designers to provide the game rules—but these tend to happen later in the jam, and never at the start.

This structure also acts as a useful lens for keeping the project focused on what the game is "for", as decided by the team at the start of the jam. When team members are uncertain or in disagreement about the direction of the game they are making, keeping the desired aesthetics of the game that were initially decided upon at the start of the jam has tended to act as a useful guideline. This does not mean that the team is rigid with regard to the game's direction; rather, by placing a high value on the aesthetic goals of the project, it becomes easier to make decisions. Zook's study [6] noted that the most common change to the game's design during a jam was the decision to cut features; where features need to be cut in order to still produce a completed game, it is best to attempt to use the game's aesthetics and theme to guide *which* features to cut.

5.2 Limitations

The first limitation observed is that dividing teams into groups, each with a fixed focus, reduces the flexibility of individual team members. In the structure described in this paper, programmers have minimal jurisdiction over game mechanics, and are beholden to the game designers in this respect. Meanwhile, the game designers are made to focus only on creating game dynamics that match the overall goal, and are discouraged from spending time creating more detailed aesthetic content.

The team members are not *prohibited* from working outside their area, but we have found that after being given a domain of responsibility, team members tend not to leave it. This may result in team members who have strong skills in more than one area not contributing as much as they can.

Secondly, dividing up the responsibilities of the team in this way creates dependencies. Programmers cannot implement the entire game until they have the full set of rules from the game designers, and this further delays the production of art assets as a result of their having to know precise requirements from the programmers. In practice, this has not resulted in significant problems, because the preliminary, preparatory work done by both the programmers and the artists while waiting for the designers is both necessary for quality work later, and also takes roughly the same amount of time as the designers need. However, this may not continue to be the case.

Finally, while the approach to team structure described in this paper provides clear guidelines for the early stages of the game jam, the structure loses its rigidity towards the end of the jam. Once the game designers have delivered their game rules to the rest of the team, the designers have less of a defined role.

In past game jams, the authors have observed the designers turning to a jack-of-all-trades role, supplementing and assisting other team members with other tasks, such as documentation, playtesting and management of tasks. This may be a consequence of the time-limited nature of jams, because we have also observed this diffusion of task focus occurring in all team members as the conclusion of the jam approaches.

6. CONCLUSIONS

The MDA framework is useful in game design, but it is rare to see it discussed in the context of game production. We speculate that this is due to the fact that the concept may not effectively scale to productions that have an increased team size or have more complex requirements; additionally, the requirement to have all team members being continuously productive on the same project at the same time may be more common in the time-constrained environment of a game jam.

The use of MDA to drive game production during hackathon-type environments shows great potential. The structured approach provided by MDA makes decisions more straightforward, and guides the project. At the same time, dividing teams into MDA-based groups potentially reduces the flexibility of individual team members, and their ability to float between various roles in the project.

The authors look forward to seeing where this exploration of MDA in this context leads, and will continue to critically evaluate the application of this development methodology as they attend and participate in more jams.

7. ACKNOWLEDGMENTS

Thanks to the various hackathons and game jams where the authors applied the approach reported in this paper: GovHack 2013-2015, the Qantas Codeshare Hackathon, TasJam 2015, and TasJam Health. Thanks also to our colleagues, friends, and collaborators at all the hackathons and game jams!

8. ADDITIONAL AUTHORS

Additional author: Jason Imms (Freelance Games Journalist, email: `jason@allegedlyinteresting.com`).

9. REFERENCES

[1] A. Arya, J. Chastine, J. Preston, and A. Fowler. An international study on learning and process choices in the global game jam. *International Journal of Game-Based Learning (IJGBL)*, 3(4):27–46, 2013.

[2] R. Hunicke, M. LeBlanc, and R. Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, 2004.

[3] F. Lantz. MDA. *Game Design Advance*, 2015.

[4] J. Musil, A. Schweda, D. Winkler, and S. Biffl. Synthesized essence: what game jams teach about prototyping of new software products. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, volume 2, pages 183–186. IEEE, 2010.

[5] K. Neil. Game design tools: Time to evaluate. In *Proceedings of the DiGRA Nordic Conference*, 2012.

[6] A. Zook and M. O. Riedl. Game conceptualization and development processes in the global game jam. In *Workshop Proceedings of the 8th International Conference on the Foundations of Digital Games*, 2013.