A Tool for Evaluating, Adapting and Extending Game Progression Planning for Diverse Game Genres

Katharine Neil, Denise De Vries, Stéphane Natkin

A Tool for Evaluating, Adapting and Extending Game Progression Planning for Diverse Game Genres

Katharine Neil[1,2], Denise de Vries[2], Stéphane Natkin[1]

[1]CEDRIC-CNAM, Paris, France
katharine.neil@gmail.com, stephane.natkin@cnam.fr
[2]Flinders University, Adelaide, Australia
denise.devries@flinders.edu.au

Abstract. Game progression design is a demanding, data-intensive design activity that is typically performed by game designers without even basic computational support. To address this, a concept for tool-supported "progression planning" has been proposed and implemented by Butler, Smith, Liu & Popovic for the design of their educational puzzle game *Refraction*. *Refraction* is a game that has relatively undemanding progression design needs. Further tool development and practice-based evaluation is needed to establish whether – and if so, how – a generic, tool-supported progression design process can address the diverse range of often complex progression design challenges that game designers find themselves engaging with. In this paper we describe how we used three game design case studies in contrasting game genres to inform the development of a tool that adapts and extends the progression planning approach.

Keywords: game design · progression planning · design tools

1 Introduction

Designing game progression, commonly understood as a structure consisting of serially introduced unique challenges [1], can be a demanding task for a game designer. To design a game's progression is to design the way that game is experienced by the player over time; the way gameplay elements are introduced is largely responsible for its aesthetics of pacing, challenge and variety.

Currently computational support is not used for this task, despite the increasing importance and sophistication of progression within game design. In response to the need for tools to aid progression design thinking, Butler, Smith, Liu & Popovic have proposed a general architecture for "progression planning" tools which they have implemented within their level authoring tool for their educational game *Refraction* [2]. As a demonstration of this concept, the Refraction tool ably hints at its potential. However, its strength as a tool to help solve challenging progression design problems remains largely untested; as Butler et al acknowledge, Refraction game's genre and scope as a puzzle game with modest progression design needs limits its applicability to other games.

adfa, p. 1, 2011.

We have sought to implement, evaluate and build upon this progression planning approach, by orienting it towards more demanding and more varied progression design problems. To this purpose we used three game design case studies in contrasting genres, all at the early progression design phase of their development: a top-down shooter with puzzle elements, a casual strategy game and an adventure game. In this paper we describe how we adapted and extended Butler et al's progression planning concept to support progression design for these games.

2 Research Objectives

We have two main research goals for this work. The first is to evaluate and explore how *Refraction*'s progression tool approach can be applied to the progression design challenges of games other than *Refraction* – particularly its universality in terms of servicing the needs of games of contrasting genres. The second, which foreshadows the first, is to discover and reveal the ways in which Butler et al's approach requires adapting or extending in order to apply it. This paper focuses on the results of our second objective.

2.1 Our Approach

There exists a wide gulf between the primitive document-editing and non-standardised diagramming practices that many game designers currently use and the advanced automated design and reasoning based research being applied to the problem of game design tools in the research community. While mixed initiative game design research is exploring the potential of computers to participate in design thinking and serve not merely as "tool" but as collaborator or expert [3][4], practitioners still do their design thinking typically without any computational support.

With a "first, do no harm" philosophy in mind, it is the "low-tech" elements of Refraction's progression planning tools - its simple constraints editing and visualisations – that interest us the most. We see them as a possible first step forward from the nascent formal approaches being explored within current practice, in that they support what many designers are more or less already doing. In building upon this aspect of the Refraction tool we also serve our aim of working at a sufficiently generic level so as to adapt the approach to other games. Accordingly, when implementing Refraction's approach we have in certain cases chosen to simply visualize the data and allow the designer to conduct their own analysis and policing of design moves (as done with pen and paper) rather than add computational support, choosing and thereby necessarily restricting the form that support should take.

3 Method

It was not practical to use the *Refraction* progression planning tool itself as a starting point for our work, as it is integrated with the game's level editor and generation system. We began, therefore, by building our own progression planning tool based on *Refraction*'s approach. We then adapted and extended our tool with a view to simultaneously servicing the specific needs of three game design case studies. These case studies were all at the early progression design point of their design cycle. The games have been under development by the primary author of this paper for between six months and three years. The designs represent three game genres: a casual strategy game, an adventure game and a top-down shooter. Usefully, these genres vary quite markedly in the nature of their progression. Servicing these contrasting design needs helps enforce a degree of universality in our tool design. Our games cannot be representative of all game genres, however, and like *Refraction*'s system, the design of our tool was inevitably driven by the needs of specific games rather than the needs of all possible games. Intuitively, the progression planning requirements of all three of our games go beyond the concept-based progression units used for the *Refraction* game. Most notably, unlike *Refraction* the games are all, to some degree, non-linear in their progression structure. This feature invokes the challenge of how to plan progression that does not take the form of linear sequence.

4 System Model

Here we describe our tool and how it builds upon, modifies and extends the model used by Refraction's progression planning tool.

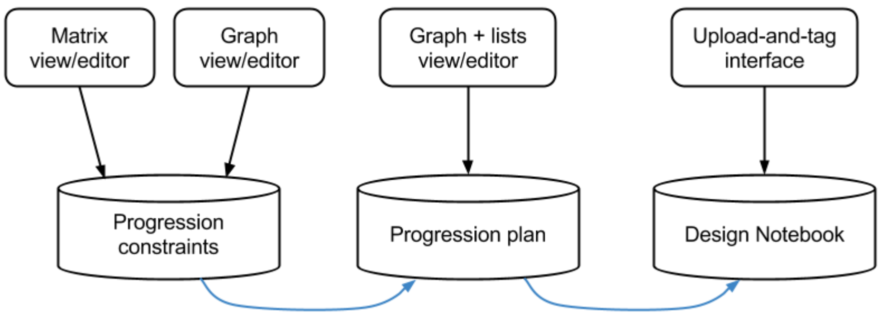

Fig. 1. System model.

Our system, inspired by *Refraction*'s, comprises constraints, plan and idea repository components (see Fig. 1). While our system performs a similar role, our differing workflow approach is manifest in some important functional differences. *Refraction*'s system uses its simple constraints calculations to drive generative features and analyze design moves, generating the progression plan and, optionally, the levels themselves. Our system removes the generative features and much of the automated analysis performed by their system, and instead focuses on presenting and organizing the results of our calculations to the designer. Most notably, our system, created to serve all three of our case studies, does not include a procedural level generation system or an integrated level editor. We have replaced this component with a designer's notebook style feature for storing and tagging ideas, design patterns and work-in-process levels, which can then be filtered according to data created in the progression plan component. We have also extended the constraints and progression plan components, adding additional calculations and visualization features, including a graph traversal algorithm to the progression plan component in order to apply constraints to non-linear level progressions.

4.1 Progression Constraints Component

As in *Refraction*'s tool, our progression constraints component allows the game designer to create game elements and constraints by defining some elements as explicit prerequisites and co-requisites of other elements. An element can be as concrete and quantifiable as an ammo pickup, for example; or a quality the level might contain like "intense combat". As well as providing a matrix-style editor, the tool automatically infers transitive prerequisites (as does *Refraction*'s tool). This is a non-trivial calculation for a designer to perform manually.

Graph and Matrix Interfaces.
To this component we have added an editable graph-based view as an alternative to the matrix view. We chose to include a graph-based view upon discovering that the progression plan for our adventure game could be in fact better described and supported by taking the form of progression constraints, leaving the progression structure itself for managing the higher level episodic narrative. As in matrix view, graph edges are automatically added when constraints are transitively inferred by the tool. This kind of computational support is useful for managing the binary but complex progression logic of an adventure game.

4.2 Progression Plan Component

Graph Interface.
The most signification modification we have made to *Refraction*'s approach is expanding the progression plan out from their linear table form into a graph-based editor and analyzer. A graph allows for games that have a non-linear progression structures to be modeled. The designer uses the graph interface to create levels or mission stages as nodes and define connections between them as edges. He or she can select a node in the graph to edit the properties of the associated level.

Game Elements List.

Alongside the graph interface we display a list of all game elements. Any element in the list may be selected and added to the level's game elements list. The elements in the list are displayed differently, according to their eligibility for use in the currently selected level: eligible, ineligible and potentially eligible. Eligibility is based on the constraints defined using the Progression Constraints component: it is calculated using a graph traversal algorithm (our progression plan analyzer) to determine whether a given element satisfies the constraints governing the level. The inclusion of the "potentially eligible" type is due to the non-linear progression structure. It services the case where one or more, but not all, paths leading to the level satisfy the constraints associated with the listed element. Being "potentially eligible" may render the element appropriate or inappropriate for inclusion in the selected level, depending on the nature of the game or the element itself. The designer is left free to make an informed decision as to whether they wish to include the element based on this contextual information.

Level Game Elements.

The level's game elements list is where the designer defines the contents of the level, using game elements from the game elements list. The designer may simply indicate the presence of a concept used (e.g. "ranged combat") or they may specify the number of instances of the element in the level (e.g. 5 ammo pickups).

Progression Histories.

A designer may wish to know which game elements the player has experienced or has potentially experienced by the time they reach the selected level, based on the content of the levels that may or must be completed prior to it. This can be viewed for all possible paths to the level, or for a single path selected in a "Paths to selected level" list. For example, the designer might see that the player has encountered a minimum 4 ammo pickups but potentially a maximum of 16. Selecting a path also highlights all the levels along that path in the graph view.

Graph Element Visualization.

Refraction's tool includes a constraint editor and visualizer that plot the density and frequency of elements in the progression plan, in order to regulate progression considerations such as game pacing. In line with our approach, our tool computes this for the purpose of visualization only: we highlight all nodes of the progression graph that contain a selected game element, thus affording the designer a broad overview of where instances of a given element are used in their game.

4.3 Filtered Design Notebook Component

Our third component of the system, as noted above, is an alternative to Refraction's integrated level editor. Design process is commonly understood to be notoriously non-linear.[1] Here we provide a home for level design fragments or design patterns that do not yet have a home within the progression plan itself. Ideas can be tagged with one or more of the game elements it includes, and the notebook can be filtered by the level's game elements list to display all ideas that contain only the game elements relevant to that level.

5 Conclusion

We have found that is possible to apply, adapt and extend Butler's progression planning approach to other game genres, by using three contrasting case studies with concrete progression planning challenges to discipline and inform this process. It being unlikely that we have anticipated all the needs of our three design case studies, further additions and modifications will probably need to be made during this process. Our next step is practice-based evaluation: we will integrate the tool into the progression design process of our three case studies, and diarize the experiences of the designer for analysis. This analysis will then be used towards developing a generic tool-supported progression design process that can be refined and tested with a wider group of designers and design cases.

6 References

1. Juul, J.: The open and the closed: Games of emergence and games of progression. In: Mäyrä, F. (ed.) Computer Games and Digital Cultures Conference Proceedings. pp. 323–329. Tampere University Press (2002).
2. Butler, E., Smith, A., Liu, Y., Popovic, Z.: A mixed-initiative tool for designing level progressions in games. ACM Symposium on User Interface Software and Technology (2013).
3. Khaled, R., Nelson, M.J., Barr, P.: Design metaphors for procedural content generation in games. Proc. SIGCHI Conf. Hum. Factors Comput. Syst. - CHI '13. 1509 (2013).
4. Smith, G., Whitehead, J., Mateas, M.: Computers as design collaborators: Interacting with mixed-initiative tools. Proc. Work. Semi-Automated …. (2011).
5. Winograd, T.: Bringing design to software. ACM Press (1996).

[1] According to Donald Schon, "unpredictability is a central attribute of design - it is not necessarily the defining one, but it is important. It means that there is no direct path between the designer's intention and the outcome"[5].