A Tool for Supporting Game Design Education: Tower Defense Generator

José P. Zagal, Pitchatarn Lertudomtana, DePaul University, 243 S. Wabash Ave, Chicago IL 60604
Email: jzagal@cdm.depaul.edu, p.lertudomtana@gmail.com

Abstract: In education, game design is often used as a means to an end: for example, to learn computer programming. Inspired by the notion of constructional design, or the design of tools to support other's design activities, we are exploring the use of tools to support learning game design as an end in itself. We present a work-in-progress tool called *Tower Defense Generator* that allows student game designers to actively, and reflectively, explore a game's possibility space while developing a deeper understanding of the key features of the tower defense sub-genre, and how those features interact to produce meaningful gameplay experiences. TDG allows learners to use different heuristics to procedurally generate game levels that can then be analyzed, playtested, and modified. We argue that these (and other) features provide for an environment with the appropriate amount of scaffolding to encourage powerful and interesting design explorations in support of learning.

Introduction

Game design and development is often used to help students learn something else, for example, computer programming (e.g. Overmars, 2004). When doing so, learners may develop an "experience in design (i.e., planning, problem solving, researching, dealing with time constraints, [...], and bringing everything together into one project)" (Kafai, 1996). However, the end-goal isn't for students to learn and critically reflect on game design (see Games & Squire, 2008 for an exception). We are exploring ways to help students better learn and practice game design through the use of mixed-initiative design tools. A mixed-initiative approach is one where content (e.g. game levels or maps) "is created through iterative cycles between the human designer and procedural support" (Smith et al., 2010). The idea is to empower a designer by making it easy to rapidly examine alternative designs and automatically check for basic playability (e.g. is it possible to complete this level). We are also inspired by Resnick's notion of constructional design as "a type of meta-design: [that] involves the design of new tools and activities to support students in their own design activities" (Resnick et al., 1996). We believe that combining these ideas can help us develop better tools for practicing game design while also facilitating reflection and learning about game design. We introduce a work-in-progress tool to assist in the design of "tower defense" games. Our tool, *Tower Defense Generator* (TGD), encourages learners to actively, and critically, explore the design space of tower defense games while helping them develop a deeper understanding of the key genre features and how they interact.

Tower Defense Generator

In tower defense games players must position static defenses (i.e. towers) to prevent increasingly tougher waves of various kinds of enemies from reaching and/or destroying an endpoint. Players also collect resources (usually by killing enemies) that are used to purchase or upgrade existing towers. Successful play requires deciding which towers to build, where they should go, and when to upgrade them. In some games, tower placement is also strategic in that it can create paths or mazes that enemies must navigate, thus providing more time for the towers to attack. The pace of these games is moderate and does not require rapid button presses (Ta et al., 2009) and, from a player's perspective, they are easy to play even while they may require tactical and strategic sophistication (Avery et al., 2011). This makes them ideal candidates for learning game design; they are simple to play, but hard to understand deeply. Popular titles in this genre include the flash-based browser game *Desktop Tower Defense* and the multi-platform *Plants vs. Zombies*.

TDG is a tool for the designing tower defense games. It consists of a map generator, a map editor, and separate specifications for towers, enemies, and waves (in XML). Finally, there is a game engine that given a map, tower, enemy, and wave specifications, allows the player to play the game. Due to space, we will only describe the map generator.

TDG's procedurally generated map creator (see Figure 1) allows learners to explore the possibility space of playable maps. In addition to certain "baseline" parameters such as the dimensions of a map, TDG includes a variety of heuristics for map-generation that learners can select from, configure,

and then analyze the resulting output. For example, maps can be randomly generated after specifying general parameters such as density of certain terrain features (e.g. traversable vs non-traversable). Alternately, a branching algorithm can be used to create paths that enemies could follow. Each heuristic generates maps with a distinctive "feel", and providing these options encourages learners to reflect on the kinds of game experiences that result from different geographical layouts and how they may align (or not) with their design goals. Rather than focusing on the creation of a single map, our tool provides several for them to analyze, question, and explore. This avoids overwhelming learners with too much freedom and helps limit the scope of their explorations to a single idea. Our approach, in a nutshell, is to provide a "grey box" that avoids the inaccessibility of a "black box" and the complexity of a translucent "white box". By providing a way to rapidly test (play) a map and edit it, we also encourage the kind of iteration (and reflection) that is essential to design and learning. In other words, rather than help generate an "ideal" map, our goal is to help learners reflect on what features make a map successful (or not), learn to identify not-quite-there maps that could be successful after some changing, and be able to make and test those changes.

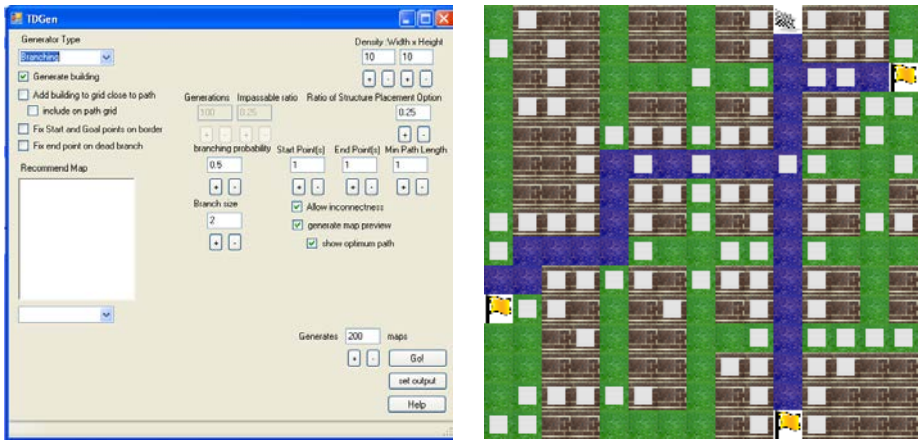

***Figure 1*: Map generator and sample map with one end point and three spawn points.**

Future Work

We are embarking on a formative assessment of our tool in a college educational setting. Additional developments include allowing users to define their own map-generation heuristics and tools for visualizing and modifying pacing and game balance. We are also exploring features for annotating maps with design goals, notable features, tracking the history of changes, and allow easier sharing.

References

Avery, P., Togelius, J., Alistar, E., & van Leeuwen, R. P. (2011). Computational intelligence and tower defence games. *2011 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1084-1091).

Games, I. A., & Squire, K. (2008). Design thinking in gamestar mechanic: The role of gamer experience on the appropriation of the discourse practices of game designers. *Proceedings of the 8th international conference on International conference for the learning sciences - Volume 1* (pp. 257-264). Utrecht, The Netherlands: International Society of the Learning Sciences.

Kafai, Y. (1996). Learning design by making games. In Y. Kafai & M. Resnick (Eds.), *Constructionism in practice*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Overmars, M. (2004). Teaching computer science through game design. *Computer, 37*(4), 81-83.

Resnick, M., Bruckman, A., & Martin, F. (1996). Pianos not stereos: Creating computational construction kits. *Interactions, 3*(5), 41-50.

Smith, G., Whitehead, J., & Mateas, M. (2010). *Tanagra: A mixed-initiative level design tool*. Paper presented at the FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games.

Ta, D.-N., Raveendran, K., Xu, Y., Spreen, K., & MacIntyre, B. (2009). *Art of defense: A collaborative handheld augmented reality board game*. Paper presented at the ACM SIGGRAPH Symposium on Videogames.