Tanagra: An Intelligent Level Design Assistant for 2D Platformers

Gillian Smith, Jim Whitehead, Michael Mateas

Expressive Intelligence Studio
University of California Santa Cruz
Santa Cruz, CA, USA
{gsmith, ejw, michaelm}@soe.ucsc.edu

Abstract

We present a demonstration of Tanagra, an intelligent level design assistant for 2D platformers. Tanagra integrates reactive planning and constraint programming to automatically generate levels and respond to designer changes in realtime. A reactive planning language, ABL, allows us to easily express hierarchical patterns of level geometry, from simple patterns such as a jump over a gap, to more complex patterns such as staircases and pits. A model of player movement and constraints within and between geometry components ensure that the levels created by our system are always playable.

Introduction

Creating a good level is a time consuming and iterative process: designers will typically play a level themselves a number of times before showing it to anyone else, simply to check that it is playable and meets their expectations (Castillo and Novak 2008). Making a change to a small section of a level, such as moving a single piece of geometry, can have a wide impact and require much of the rest of the level to be modified as well.

A mixed initiative-approach to level design, in which content is created through iterative cycles between the human designer and procedural support, offers a solution to this problem. Designers could take advantage of the many benefits procedural content generation has to offer: rapidly showing alternative designs, regenerating specified portions of a level, and enforcing playability. However, such a tool requires a new approach to procedural content generation. Evolutionary algorithms, which have seen success in offline level generation and adaptive content creation (Togelius, DeNardi, and Lucas 2007; Hastings, Guha, and Stanley 2009), are inappropriate for a realtime tool. Grammar and rule-based approaches show a great deal of promise (Lipp, Wonka, and Wimmer 2008; Golding 2010); however, these systems are confined to creating non-interactive structures that do not need a model of timing and physics constraints.

We demonstrate Tanagra, an intelligent level design tool for a 2D platformer that integrates reactive planning and constraint programming. This tool allows the designer to manipulate geometry in the level, both in terms of individual level components and more sophisticated patterns of geometry. Tanagra also supports modifying the pacing of the level itself, without needing to manipulate geometry at all.

System Overview

We use a reactive planning language, ABL (Mateas and Stern 2002), to easily express hierarchical patterns of geometry that can be incorporated into the level, and also monitor and react to designer changes. The geometric relationships between level components are given to a constraint solver, Choco (Choco Team 2008), as a set of constraints that must be satisfied, thus ensuring that the generator will never produce an unplayable level. A diagram describing our system's infrastructure is shown in Figure 1.

Beat Representation

A fundamental concept in our system is the use of beats to represent frequency of player actions. Each beat corresponds to an action that the player takes in the level, such as jumping over a gap or killing an enemy. We have previously shown that beats are a useful abstraction for dexterity-based platformers, and can be used to model rhythm and pacing in a level (Smith, Cha, and Whitehead 2008). Beats are the glue that holds our levels together: without them, the generator would not know how to assign length to a geometry pattern, or how to tie geometry patterns together.

A Behavior Language (ABL)

ABL is a Java-based reactive planning language, designed for authoring agents that can react to a rapidly changing

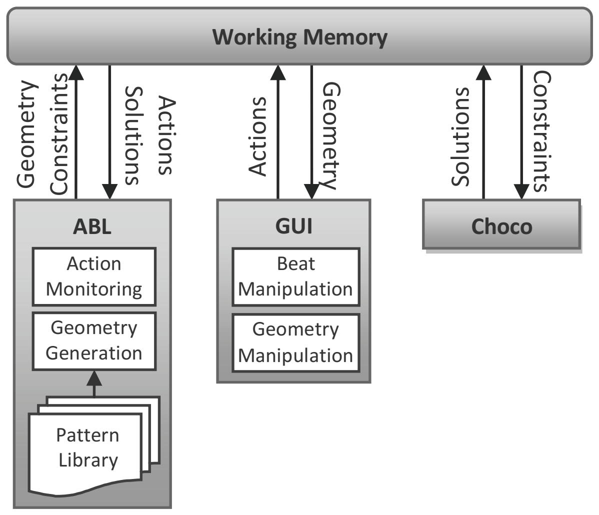

Figure 1. An architecture diagram explaining the interplay between ABL, Choco, and the designer's GUI.

world state. Agents are authored as a set of hierarchical behaviors.

The Geometry Pattern Library. Our system builds levels out of pre-specified geometry patterns, which are defined by ABL behaviors. The simplest of these patterns span a single beat: for example, two platforms separated by a gap or an enemy patrolling a platform. Other, more complex patterns are constructed from these single-beat patterns, with optionally added constraints. For example, the multi-beat "staircase" pattern is built out of multiple gap-jump patterns, with the constraint that the height of each gap in the staircase must be equal.

Geometry patterns are made up of level components—platforms, gaps, enemies, stompers, and springs—and constraints that define how these components fit together. Level components have parameterizable properties such as position and dimension; which are expressed as constraint variables. A simple model of avatar movement poses additional constraints on these patterns, as their length must correspond to the duration of the beat they belong to.

Tanagra understands five different single-beat patterns: a simple long platform, a gap between two platforms, an enemy on top of a platform, a stomper on top of a platform, and a spring that propels the player over a gap. There are also two multi-beat patterns that incorporate a combination of these single-beat patterns: staircases, and pits/mesas.

Constraint Programming with Choco

Choco is an open source Java library for describing and solving constraint satisfaction problems. Constraints can be expressed on variables that are Booleans, integers, or real numbers. We use Choco to model constraints for the geometric relationships within and between level components in geometry patterns. For example, a gap between two platforms has a number of constraints associated with it concerning the width of the gap's impact on the start and end positions of the platforms. A gap also has internal constraints: making sure that if the width of a gap is zero, its height cannot also be zero, and vice versa. These constraints are created by ABL, which communicates them to Choco for solving. In turn, Choco informs ABL of the solutions it finds, so that ABL can continue generating geometry for the remainder of the level.

Demonstration

Figure 2 shows an example of a level created using Tanagra. Dark platforms are those that the user has modified from the generated level; note that there are two high-level patterns visible in this level: a staircase at the beginning, and a pit in the middle. Further details on Tanagra are available online[1].

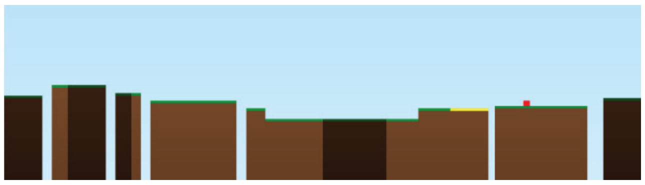

Figure 2. An example of a level created using Tanagra.

References

Castillo, T. and Novak, J. 2008. *Game Development Essentials: Game Level Design*. Delmar Cengage Learning.
Choco Team. 2008. Choco: an open source Java constraint programming library. White Paper, CPAI08 Competition.
Golding, J. 2010. Building Blocks: Artist Driven Procedural Buildings. Session at *Game Developers Conference 2010*. San Francisco, CA.
Hastings, E. J., Guha, R. K., and Stanley, K. O. Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Transactions on Computational Intelligence and AI in Games*. 4(1), 245-263.
Lipp, M., Wonka, P., and Wimmer, M. 2008. Interactive visual editing of grammars for procedural architecture. In *Proc. of ACM SIGGRAPH 2008*. Los Angeles, CA.
Mateas, M. and Stern, A. 2002. A behavior language for story-based believable agents. *IEEE Intelligent Systems*, 17(4), 39-47.
Smith, G., Cha, M., and Whitehead, J. 2008. A framework for analysis of 2D platformer levels. In *Proc of ACM SIGGRAPH Sandbox Symposium*. Los Angeles, CA.
Togelius, J., De Nardi, R., and Lucas, S. 2007. Towards automatic personalised content creation for racing games. In *Proc. of the IEEE Symposium on Computational Intelligence and Games (CIG '09)*. Honolulu, HI.

[1] http://eis.ucsc.edu/tanagra