Game AI, Spring 2013
Project III

Procedural Content Generation is the use of algorithms (procedures) to create novel - and sometimes customized - game content from scratch. Examples of PCG include generation of levels, maps, tree, cityscapes, weapons, monsters, and quests. PCG is often used as a design-time tool to roughly sketch out level content to be refined by human designers. PCG can also be done at run-time to incorporate individual player differences such as skills or preferences. In this project, we look at run-time PCG to create Mario Bros. game levels customized to individual players' play styles. This includes (a) learning a model of the player's play style, and (b) using the model to create a custom level. Fortunately, the first part is already done for you. You must focus on designing and implementing algorithms that use the player information to create something that will evaluate well.

This project will be using the IEEE Super Mario Bros. competition infrastructure for the Level Generation Track. The description and distribution of the software can be found here:http://www.marioai.org/

The rules for level-creating algorithms will follow the rules of the competition. **This is a team project.** You are responsible for obtaining the code and using the resources provided with the competition. Please use Piazza for all online conversations and questions for the class community. Note that there is an active development community here that is a valuable resource: http://groups.google.com/group/mariocompetition/topics

You will be given 3.5 weeks for this assignment with weekly-suggested milestones:
W1: Get the game running, test the RandomLevel, CustomizedLevel, and MyLevel. Make some simple changes to MyLevel.
W2: Design your mapping from player model (read in from "player.txt" to a GamePlay object) and start implementing your design.
W3-4: Implement, and tweak.

Getting up and running

1. Acquire and install apache ant (http://ant.apache.org/)

2. Unzip MarioLevelComp2011_cs4731.zip (this one provided is already built, has an ant build file, and several sample level generators.)
3. Modify the following files: MarioLevelComp2011_cs4731/src/dk/itu/mario/level/MyLevel.java and MarioLevelComp2011_cs4731/src/dk/itu/mario/level/generator/MyLevelGenerator.java (if necessary)
4. Build the project: in the MarioLevelComp2011_cs4731 directory, type "ant"
5. Run your level generator from the MarioLevelComp2011_cs4731 directory: java -cp bin dk.itu.mario.engine.PlayCustomized

Evaluation

Level generation is highly subjective. To control for subjectivity, we will create several distinct test player profiles with the expectation that, by running the code several times with each profile, we will be able to see discernable consistency among runs with the same profile and discernable differences among runs with different profiles. While your solution will likely have stochastic elements, the solutions should not appear totally random. Your writeup will be key in understanding how to interpret your design.

Hints

- You can do all your work in MarioLevelComp2011_cs4731/src/dk/itu/mario/level/MyLevel.java. It comes with some handy routines for laying out common Mario elements. The original MyLevel just creates a repeating pattern of straight, straight with hill, pit, straight with tubes, straight with cannons. It is very easy to see how this is done, so you might want to check this code out first.
- You can also look at RandomLevel.java and CustomizedLevel.java. CustomizedLevel.java creates some probability tables for liklihood of certain common elements. To run the RandomLevel, use java -cp bin dk.itu.mario.engine.Play
- Your level generator should eventually make use of the information stored in a player model. The player model is passed in as a GamePlay object (called playerMetrics)
- Consider implementing an optimization search routine that tries to maximize some given evaluation/fitness function.
- Some papers to read on level generation for Mario: http://www.marioai.org/levelGenerationCompetition.pdf, http://noorshaker.webs.com/images/AIIDE.pdf, http://julian.togelius.com/Pedersen2009Modeling.pdf. Here is a paper on "rhythm" in platform games: http://games.soe.ucsc.edu/rhythm-based-level-generation-2d-platformers. There is also a paper uploaded on T-square in resources called "PCG.levels.pdf" to read.

Writeup

Include a .doc or .txt file that describes the following:

- Exact instructions for how to run your agent (if these are not included, you may receive an F).

- The algorithmic approach you used, with particular attention to how you chose to map the player model features to decisions about content and layout. Be thorough, we will expect to see how the mapping occurs in practice. This is key in being able to grade your design.
- The major challenges you met during the project and how you overcame them.
- Make clear what your goal was in creating your algorithms, why that goal was a reasonable one, and how well you achieved it.

Submission

Submit via T-Square everything needed to run your agent, including the native SMB code, and your writeup.