Electronics and Computer Science
Faculty of Physical and Applied Sciences
University of Southampton

Thomas A. E. Smith
taes1g09@ecs.soton.ac.uk

April 9, 2012

Intelligent Procedural Content Generation for Computer Games

Project supervisor: E. Gerding – eg@ecs.soton.ac.uk
Second examiner: C. Cirstea – cc2@ecs.soton.ac.uk

A project report submitted for the award of
MEng Computer Science with Artificial Intelligence

Abstract

Increasingly, as the demand for ever larger and more varied computer game environments grows, procedural content generation (PCG) is used to ensure that content remains 'fresh'. However, many of the opportunities to use these systems to generate truly personalised content have so far been largely overlooked. When content is generated manually or algorithmically during the design phase of a game, it can only be created according to the designers' expectations of the players' needs. By instead generating content during the execution of the game, and using information about the player(s) as one of the system's inputs, PCG systems should be able to produce more varied content that can be far more tailored to enhance individual players' experiences than anything manually created. Much has been written about the generation of player models from observed data, including for the purposes of adaptivity or dynamic difficulty adjustment (DDA), and literature exists examining the problem of generating satisfying game environments via challenge adjustment. This project looks at combining these fields to create an intelligent PCG system (IPCG) that is capable of monitoring players' progress and dynamically generating upcoming challenges to best suit their abilities. TODO: specify that this is a prototype, investigating a specific aspect of the many that IPCG is able to cover.

Contents

Statement of Originality

- You are strongly encouraged to include a one or two paragraph statement of originality this is all my own work is rarely true you should acknowledge the help you have received - Was the idea for the project yours, or was it based on an earlier project, or your supervisors research? - The examiners will assume that the analysis, design, implementation, testing, . . . are your own work - So tell them where this is not true the design of component X follows a standard technique/pattern described in [source] this is my own code except for ¡package/class/method¿ which I have copied from ¡Internet site/author¿

1 Project Description

1.1 Introduction

The aim of this project is to investigate the use of intelligent procedural content generation (IPCG) in computer games, by looking at existing products, research in related areas and constructing a minimal prototype. ¡TODO: explanation of IPCG¿ Due to the increasing demand for both detail and variety within computer game environments, various aspects of in-game content are now often generated procedurally (that is, algorithmically rather than manually), using techniques that are frequently just refinements of algorithms used in the early days of computing, for games such as Elite or Nethack.However, one of the strengths of modern procedurally generated content (PCG) is that (within reasonable limits) it may be performed at runtime, allowing it to also use information about the player in order to dynamically generate content on-the-fly in response to the player's actions. In general, this will involve making use of algorithms from the field of Artificial Intelligence in order to evaluate the information available and condense it to a form suitable for input to a PCG system; hence such systems might reasonably be termed 'intelligent' procedural content generators (IPCG). As shown later in the literature review and background research sections, IPCG techniques can be applied to many different scenarios, for many reasons. For the purposes of this project, the prototype will be restricted to using IPCG for dynamic difficulty adjustment (DDA) in basic 2D platformer levels.

Definitions:

Procedural Content Generation

Dynamic Difficulty Adjustment

Intelligent Procedural Content Generation

1.2 Literature and Existing Systems

1.3 Prototype

1.4 Goals

2 Project Background

Lot of academic work in related fields Paucity of actual shipping systems

2.1 Procedural Content Generation

Procedural Content Generators have been used since the early days of gaming. Well-known games such as Elite and Rogue made extensive use of PCG in order to present the player with expansive game worlds far larger than could have been fully stored on the distribution media that was available at the time. As technologies improved, focus shifted more towards hand-crafted environments as it was easier to ensure that these provided value and did not feel sparse [9]. However, with the further progress of technology attention has returned to procedural generation. Modern game worlds contain vast amounts of detail, and procedural content generation algorithms are ideally suited to producing large numbers of variations on a theme, be that trees, clouds, textures, or even sounds. Producing each of these items individually by hand would take many hours of labour and much disk space, but by defining specific sub elements and assembly rules, variation can be almost endlessly reused, as in the game Infinite Mario.TODO: mention occupancy regulated expansion / extension

2.1.1 Existing PCG Systems

2.2 Dynamic Difficulty Adjustment

Another game design concept receiving increasing attention is dynamic difficulty adjustment (DDA). Typically, challenge adjustment within video games has consisted of user choice between one or more discrete challenge settings that have been painstakingly balanced at production time. However, this solution is far from ideal - typically, if a game is begun with a certain difficulty it is difficult to later change; and this upfront decision also alienates players that are unfamiliar with the terminology or expectations, or uncertain how to classify themselves[8]. Furthermore, since game difficulty is typically a continuous function of multiple parameters, it should be possible to precisely match each player to their ideal level of challenge rather than enforcing adherence to low-resolution skill profiles. Typically, DDA is achieved by altering values that are hidden from the player, such as enemy health, accuracy, or the amount of ammo and health-kits available in the world [6]. Often, the intention is to do this invisibly, and merely ensure that the player remains optimally challenged. By manipulating values behind the scenes, it is possible to ensure that the player is neither overchallenged (leading to frustration), or underchallenged (leading to boredom)[2]. As DDA systems are given more control over additional aspects of the game environment, they can begin to enter the realm of PCG, fundamentally altering the structure and pacing of the player's experience.

2.2.1 Existing DDA systems

2.3 Intelligent Procedural Content Generation

2.3.1 Existing IPCG Systems

IPCG can be (and has been) used for a wide range of purposes, almost as varied as PCG itself. Three very different such uses are detailed below. It is unsurprising that many of the existing applications of IPCG are used to tackle some of the current key challenges in game design: maintaining players' engagement with the game via enhancing immersion and controlling flow[2].

Valve's 'AI Director'

One of the most well-known such applications is used in Valve's games Left 4 Dead and Left 4 Dead 2. Known as the 'AI Director', the system monitors the "emotional intensity" of each players' gameplay experience, and dynamically alters the placement of supplies and the generation of enemies of various types in order to control pacing and maintain flow. In Left 4 Dead 2, the Director has additional control over the structure of the level[1].

Bethesda's Radiant Story

Another recent example of IPCG is the Radiant Story system used in Bethesda's Skyrim. Rather than monitor the player's performance, it evaluates their progress and history, and deliberately generates in-game tasks designed to force exploration of previously unknown areas; in order to increase immersion.

GAR's Weapon Evolution

Finally, the weapon evolution mechanism in the game Galactic Arms Race[4] is an unconventional application of an IPCG system, as it tracks only which weapons the players prefer, and then uses small neural networks to evolve new variations on the favourite weapons, which themselves are all simply procedurally generated particle systems[5].

2.4 Problem Specification

An analysis and specification of the solution to the problem

Though mechanisms fulfilling the definitions of IPCG systems have already started appearing in games, little has so far been written specifically on the subject - "personalized and player-adaptive PCG ... is a new research direction"[10]. However, existing literature in related areas borders on the topic: in some cases DDA algorithms are being used to generate entire levels; thus qualifying as IPCG. One of the most thorough papers on this area is "Polymorph: Dynamic Difficulty Adjustment Through Level Generation", by Jennings-Teats *et al.* [7], which generates 2D platformer levels on-the-fly. Approaching the topic of IPCG from another direction, Lopes' and Bidarra's "Adaptivity Challenges in Games and Simulations: A Survey" investigates the use of adaptivity in general in order to combat static and predictable content[8], including via PCG.

2.4.1 Polymorph: DDA Through Level Generation

Polymorph is a progression of prior work in both DDA and PCG: like previous DDA systems it alters challenge during play, and as with traditional PCG for 2D platformers it generates levels algorithmically. However, the DDA is effected via structural differences in the level design rather than 'numerical tweaking', and the level is generated 'online' out of small rhythm segments[11], rather than fully ahead of time. The authors present an interesting statistical model of difficulty, along with some of the issues and solutions found while evaluating the system. [7]

2.4.2 Adaptivity Challenges in Games and Simulations: A Survey

The survey is an investigation of present research into and existing commercial implementations of adaptivity in games. The topic is broken down into three areas: purpose, target and method, with a wide range of examples given for each point raised. In contrast to other papers on the subject, the authors look beyond challenge as the sole steering purpose for adaptivity, and then discuss the wide range of types of content that may be adapted or generated algorithmically. Finally, they look at the methods by which content may be adapted or generated, and conclude that PCG is one of the most promising for offline generation, and also may increasingly be suited to online adaptation. [8]

3 Proposed System

3.1 Overview

Following the separation of concerns (SoC) design practice, the project may easily be modularised into three principle components: a 'host' or base module, an adaptive PCG system, and a means of taking data about the player and converting it into a model usable by the PCG. The proposed flow of information is shown in Figure 3.1.

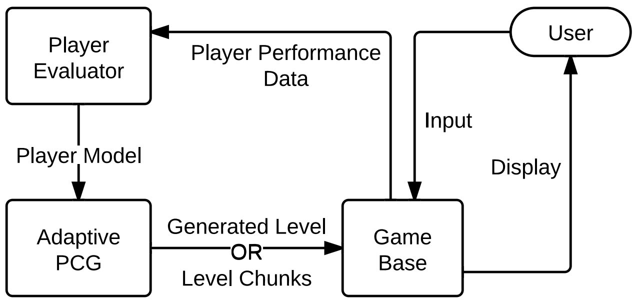

Figure 3.1: Data flow in proposed system.

3.1.1 Modules

Game Base

A simple 2D platformer engine. This module should handle all input and rendering activity, and should follow the MVC architecture. In addition to presenting the user with the output of the IPCG system, the Base module should provide the basic input handling, physics and game functionality necessary to play the platformer, while also logging multiple types of information about the players performance for the Evaluator module.

Adaptive PCG

An adaptable 2D platforming level generator. Building on the work of Compton *et al.*[3], this module should maintain a context-free grammar (CFG) of obstacles available, along with weights representing the estimated challenge of each element (terminal obstacle or combination). By taking these weights into account when deriving a string of obstacles from the CFG, sequences of a desired difficulty can be produced - or alternatively, the estimated challenge of existing sequences may be evaluated. A PCG system designed in this way should be able to generate entire levels 'offline', by maintaining pre-determined maxima and local variations in difficulty, but should also be capable of generating levels on-the-fly, by ensuring that short-term future difficulty levels match those requested by the Evaluator.

Player Evaluator

Initially, a multi-class classifier. Given the varied inputs from the Base module, the Evaluator should form a belief about the player's skill relative to the current challenge of the level. By running the player's data through a previously-trained classifier, this module should obtain a model that can be passed to the PCG module and acted upon.

3.1.2 Approach

The modules above are presented in logical order of development: none of the IPCG system will be testable without the Base program (which can be tested standalone if given a hand-crafted level), but the PCG may be run and tested using specimen models, and finally the Evaluator can be tested once the other systems are in place.

Classifier Training

The development of the classifier will involve initially collecting data on as many potentially relevant features of the player's performance as possible, and then performing principle component analysis (PCA) upon the data-set in order to identify the maximally variant features. These can then be retained and used as the input to a K-means discretisation algorithm, which can finally be used to train a One-vs-All SVM classifier.

3.1.3 Justification

An alternative approach to the problem of generating a 2D platforming environment for use with DDA is presented by Sorenson *et al.*[12], who detail a more general top-down approach using genetic algorithms. However, their system is also more complex and provides an unneeded degree of generality for this project. This system as proposed should be able to fulfill the requirements given, and demonstrate intelligent variation in output based upon the skill of the player. TODO: mention the other student's one, with cannons

3.2 Requirements

The main aim of the project requirements will be to constrain the problem to an achievable scale, and inform future evaluation of the final solution. In this instance, the functional requirements are generic to any IPCG system (with refinements specific to this problem in brackets, like so), while the non-functional requirements are constraints that should serve to encourage feasibility and quality.

3.2.1 Functional

In order to properly implement IPCG, the system should:

- present the user with an interactive game environment (basic 2D platformer)
- record data on the player's behaviour (in this case, performance)
- evaluate this data according to specific criteria (a trained classifier)

- form a model of some aspect of the player (skill relative to current difficulty)
- use this model to inform further PCG activities (level chunk generation)

3.2.2 Non-Functional

In order to remain at a manageable scale, the system should:

- be written in Java
- be confined to 2D
- be presented as a basic platformer
- limit the user to move and jump actions

but also:

- remain responsive
- properly maintain the challenge of generated content

TODO: make this FAR longer and more detailed. Write about evaluation method using swing

4 System Design

o Which process did you follow: waterfall, iterative, evolutionary? o How does this show in your plan? o Why did you choose this process? o How well did it work out for you?

TODO: mention Java

5 System Implementation

see testing

6 System Testing

see appendix

7 System Evaluation

o Comparative evaluation(cf.competition) performance graphs, feature lists o Criticalevaluation with respect to your project goals and plan

8 Project Evaluation

One or more Gantt charts showing the planned schedule and the actual progress

o Criticalevaluation with respect to your project goals and plan o Reflection in hindsight, did you use the right tools, techniques, metrics and methods? what did you learn? were your goals and plan sensible? how could you have done it better/differently?

Project Management and Planning o You should account for your time this is the major expense for most projects o Compare your initial plan with how things actually went perhaps include project diary as an appendix o If you fell behind how did you catch up or decide which features to drop o Did you consider and allow for risks illness, equipment failure or delays

8.1 Gantt Charts

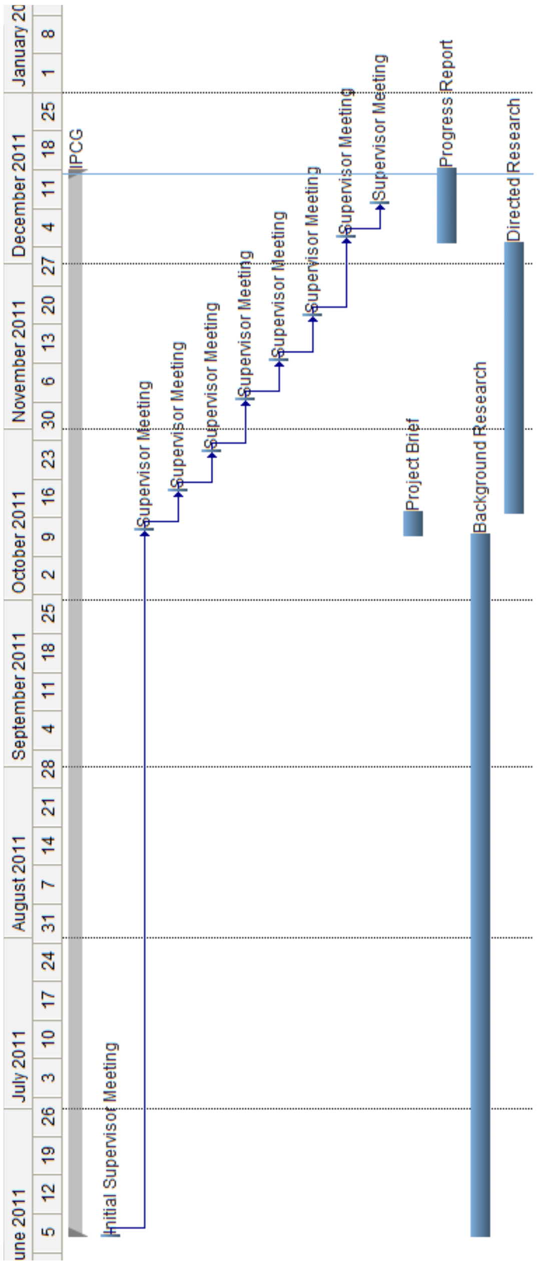

Figure 8.1: Work Completed

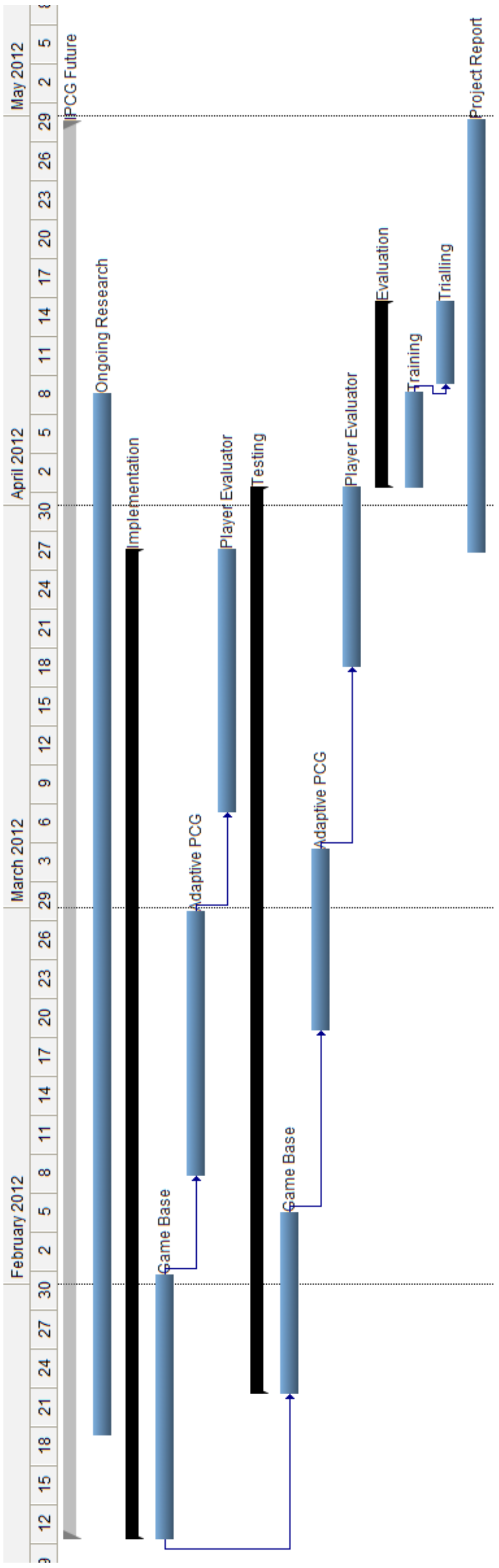

Figure 8.2: Work Remaining

9 Conclusion

9.1 Possible Extensions

One drawback of the system as proposed is that it must still reduce all of the data about the player's performance into a single discrete decision: whether to continue generating the level at the current difficulty, increase the difficulty, or decrease it. There is no opportunity for granularity representing player aptitude at a particular type of challenge. One possible extension would be to divide the obstacles by type (stationary hazard, timed hazard, projectile etc.), and evaluate the player's skill on particular classes individually, then use this more detailed model to inform a slightly more sophisticated PCG module. This approach would be an ideal candidate for a collaborative filtering algorithm, which with a large enough dataset would further allow the system to predict a player's aptitude at obstacle types that had not yet been seen by that player.

References

[1] M. Booth. The AI systems of Left 4 Dead. In *Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE '09)*, Stanford, CA, October 14 - 16, 2009.

[2] Jenova Chen. Flow in games (and everything else). *Commun. ACM*, 50:31–34, April 2007.

[3] K. Compton and M. Mateas. Procedural level design for platform games. 2006.

[4] E. Hastings, R. Guha, and K. Stanley. Demonstrating automatic content generation in the Galactic Arms Race video game. In *Artificial Intelligence for Interactive Digital Entertainment Conference*, 2009.

[5] Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Interactive evolution of particle systems for computer graphics and animation. *Trans. Evol. Comp*, 13:418–432, April 2009.

[6] R. Hunicke and V. Chapman. AI for dynamic difficulty adjustment in games. In *Papers from the 2004 AAAI Workshop on Challenges in Game Artificial Intelligence*, volume Technical Report WS-04-04, Menlo Park, CA., 2004. The AAAI Press.

[7] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, pages 11:1–11:4, New York, NY, USA, 2010. ACM.

[8] R. Lopes and R. Bidarra. Adaptivity challenges in games and simulations: A survey. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(2):85 –99, june 2011.

[9] Michael Nitsche, Calvin Ashmore, Will Hankinson, Rob Fitzpatrick, John Kelly, and Kurt Margenau. Designing procedural game spaces: A case study. In *FuturePlay 2006*, October 2006.

[10] N. Shaker, G. Yannakakis, and J. Togelius. Towards automatic personalized content generation for platform games. 2010.

[11] Gillian Smith, Mike Treanor, Jim Whitehead, and Michael Mateas. Rhythm-based level generation for 2D platformers. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 175–182, New York, NY, USA, 2009. ACM.

[12] N. Sorenson, P. Pasquier, and S. DiPaola. A generic approach to challenge modeling for the procedural creation of video game levels. *Computational Intelligence and AI in Games, IEEE Transactions on*, 3(3):229 –244, sept. 2011.

Appendix A: Project Brief

Intelligent Procedural Content Generation for Computer Games

Thomas Smith
Supervisor: Enrico Gerding

Problem: In modern computer game development, content production accounts for a large proportion of the initial (and in some cases, ongoing) outlay. As both budgets and in-game worlds get larger, there is increasing demand to offload some of these production efforts to automated systems. The concept of procedural content generators (PCG) has been around for some time, and they have been used in many successful games, but many of the advantages made available by these systems have so far been largely overlooked. When content is generated manually or algorithmically during the design phase of a game, it can only be created according to the designer's expectations of the players' needs. By instead generating content during the execution of the game, and using information about the player(s) as one of the system's inputs, PCG systems should be able to produce more dynamic experiences that can be far more tailored to enhance individual player's experiences than anything manually created. An intelligent procedural content generator (IPCG) should therefore consist of two parts: some means of evaluating (some aspect of) the player and generating a model, and a PCG system that is able to accept this model as an input and dynamically generate variants on its standard output based on the contents of the model.

Goals: As specified above, an intelligent PCG should consist of two subsystems: an evaluator and its companion generator. The aim of the project will be to create a simple game-like application that uses an IPCG system to produce dynamically variable content based on the player's behaviour. I will begin by creating a variable PCG that is able to produce content based on specimen player models, and then use environments created in this way to create and tune a player evaluator for further generation.

Scope: In order to attempt to ensure that the project goals remain achievable, the scope should be restricted to the simplest possible system. Based on inital inspection of the problem space and existing literature, it appears that this would be adjustment due to player skill in a 2D platforming environment. The project will be coded in Java, as that comprises the majority of my recent coding experience, and it has a wealth of 2D graphics drawing support which will simplify the less-relevant areas of coding. Similarly, many of the peripheral components traditionally included in computer games are irrelevant to the project and will not be needed.

Appendix B: Questionnaire Script

These questions will be presented to participants during the course of the experiment:

1) At the start of the experiment, rating agreement on a 5-point Likert item:

I am familiar with videogames in general.

strongly disagree		neutral		strongly agree
○	○	○	○	○

I am familiar with 2D platforming games.

strongly disagree		neutral		strongly agree
○	○	○	○	○

2 - 5) After each of the four interactive sections, evaluating the previous section on a 7-point Likert item:

The previous level presented a level of challenge that was:

too easy			about right			too hard
○	○	○	○	○	○	○

6) At the end of the experiment:

This data has been collected during the course of this session:

```
< collected data >
```

Submit >>

Appendix C: DVD Contents