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

Thomas A. E. Smith
taes1g09@ecs.soton.ac.uk
April 11, 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. In a reated field, 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 a prototype 'intelligent' PCG system (IPCG) that is capable of monitoring players' progress and fully 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 [5] 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.

1.1.1 Definitions

- **Procedural Content Generation** Given the variety of mechanisms used by games to serve content to players, it can be difficult to define exactly what PCG is and is not. Togelius *et. al.* define PCG as "*the algorithmical creation of game content with limited or indirect user input*" [**?**]. This definition deliberately does not specify whether randomness is required in a PCG system, as examples of both random (stochastic) and deterministic PCG systems exist. It also does not distinguish between 'offline' PCG, and adaptive 'online' PCG, as both have their uses. ¡TODO: more here¿
- **Dynamic Difficulty Adjustment** DDA is another broad term that could be applied to wildy differing systems in various games. In general, DDA is often viewed as the process of altering aspects of a game based on some part of the state of the game world - often a model of the performance of the player. Normally this is done with the intention of maintaining the player's ability to remain within a state of 'flow' [3]. It can range from the simple 'rubber-banding' used in basic racing games, to the more subtle alterations of timing, item placement and enemy frequency used by FPS games such as Resident Evil 5 [**?**].
- **Intelligent Procedural Content Generation** Intelligent Procedural Content Generation is an extension of typical PCG in that it forms a model of some aspect of the player or game state, and then uses this as an input to an adaptive

(parameterised) PCG system that modifies the generated output appropriately. Often this is done for the purposes of DDA (in contrast to typical non-PCG DDA, which generally makes minor adjustments to existing content, rather than generating new content), though it can also be applied for a range of other purposes, from providing more of the type of content a player favours [**?**], to directing the player towards unexplored areas [1]. ¡TODO use non-literature examples¿

1.2 Literature and Existing Systems

Though the focus of this project centres on the topic of IPCG, it can be seen from the definitions above that the fields of PCG and to a lesser extent DDA are closely related. An IPCG system cannot work without some mechanism existing to generate content appropriate for the model of the player that it has constructed, and even if the purpose of the system is customisation of a game aspect completely unrelated to challenge, the techniques and systems developed for the field of DDA are often relevant when attempting to model some aspect of the player. ¡TODO: write about the fact that PCG and DDA are mature fields, with plenty of academic literature, example systems and many commercial game product that use these techniques¿

1.3 Prototype

One of the aims of this project is to contruct a minimal prototype demonstrating the use of an IPCG system. By combining ideas from existing literature on PCG and DDA systems, it should be posible to construct a limited system that performs all of the stages necessary for it to function as an example of IPCG. That is, it should monitor the game state, form a conceptual model of some aspect of the player, and then use that model in order to generate content tailored to the particular player in some way. For the purposes of this project, the prototype will be necessarily basic, performing only the functions sufficient to display working IPCG. It should also provide some facility for player-provided evaluation of the system, to aid in final analysis. ¡TODO: more stuffs¿

1.4 Goals

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: fill out, listitems¿

- **Investigate the use of IPCG in computer games** To date, a remarkably small number of commercial games have contained systems that could reasonably classified as IPCG. In order to sucessfully research the current use of IPCG then, it will be important to view the field in the context of both PCG and DDA, more mature systems with more extensive bodies of existing literature and successful systems.
  - **Existing commercial products**
  - **Existing academic literature**

- **Construct a minimal prototype – Specify requirements**
  - **Design and Implement**
  - **Test and Evaluate**

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. In the case of Elite, this was done using a fully deterministic PCG system, and storing only the seeds used to generate the desired content - resulting in a game world that was identical each time it was generated, but that took very little memory to store. For Rogue, environments were generated pseudo-randomly, meaning a different play experience each time, but following strict constraints that ensured that levels were completable [**?**]. 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 [13]. 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 PGC mechanisms have matured, they are once again being used for the provision of entire play environments. Commenting on the use of PCG in a successful commercial title (Borderlands, [**?**]), A. Doull claims that "it points the way forward to a time where the current role of the level designer will be as obsolete as punch cards" [6].

, as in the game Infinite Mario.TODO: mention occupancy regulated expansion / extension

2.1.1 Existing PCG Systems

Many interesting PCG systems have been developed, too numerous to mention. Of particular note:

- **Charbitat** ¡TODO: write description for charbitat. academic, possibly IPCG. look into this¿
- **Infinite Mario** A Java reimplementation of the original Mario, uses PCG to create an endless variety of potential levels [14]. Considered something of a standard in academic PCG implementations, the codebase is often used as a launching point for competitions of various kinds.
- **Speedtree** Possibly the most widely-used commercial PCG system, Speedtree is a middleware application that generates trees and other vegetation for use in games and some movies [**?**].

– **Borderlands** A commercially successful videogame, Borderlands was one of the first to extensively use and publicise a PCG system; in this instance to generate millions of different varied weapons - a selling point for the game.

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[12]. 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. By monitoring and then modelling the players' ability in some fashion, it can be possible to make informed changes to the play environment that satisfyingly help or hinder their progress. 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 [9]. 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)[3]. As DDA systems are given more control over additional aspects of the game environment, they can begin to cross the line and enter the realm of PCG, fundamentally altering the structure and pacing of the player's experience. Much DDA literature is relevant to the field of IPCG, as the the data-collection and model-forming portions of IPCG systems have existing parallels in DDA research.

2.2.1 Existing DDA systems

– **Resident Evil 5** A successful commercial game, Resident Evil 5 contained a ¡TODO: write about the adaptive difficulty where players pick a difficulty rating as usual, which limits them to a window on the 1 to 10 rating. the system monitors: the system adjusts:[**?**]
– **Hamlet** Hamlet is a system designed by R. Hunick to examine "basic design requirements for effective dynamic difficulty adjustment" [10]. It integrates with an SDK for an existing game engine in order to monitor aspects of the player's status, and modifies upcoming encounters based on a historically-trained map from player evaluation result to desired game world adjustments.
– **Polymorph** TODO: write about polymorph. Possibly it belongs in the IPCG section?

2.3 Intelligent Procedural Content Generation

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"[15]. 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 by Jennings-Teats *et al.* [11], who developed Polymorph - a system that generates 2D platformer levels on-the-fly. Approaching the topic of IPCG from another direction, Lopes' and Bidarra's survey of adaptivity challenges in games and simulations investigates the use of adaptivity in general in order to combat static and predictable content[12], including via PCG.

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 in commercial games 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'[3]. ¡TODO: comment: '??'¿

- **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, by tracking factors such as each player's current health and recent kills, proximity and deadliness of visible enemies, and separation from the main group. It then dynamically alters the placement of supplies and the generation of enemies of various types in order to control pacing and maintain flow. It follows a policy of encouraging intensity to build up to a peak, sustaining threat for a short time, and then allowing intensity to fade during a 'relax' period, thereby creating somewhat-unpredictable peaks and valleys during gameplay. In Left 4 Dead 2, the Director has additional control over the structure of the level [2].
- **Bethesda's Radiant Story** Another recent example of IPCG is the Radiant Story system used in Bethesda's game Skyrim. Rather than monitor the player's performance and aptitude, it evaluates their progress and history. When the player recieves a quest, the system looks for viable locations that the player has not yet explored, and customises details of the task so that it will take the player to this new location. This avoids requiring the player to return to already-completed areas, and instead force exploration of previously unknown regions, helping to increase immersion by avoiding repetition of content [1].
- **GAR's Weapon Evolution** Finally, the weapon evolution mechanism in the game Galactic Arms Race[7] is an unconventional application of an IPCG system. All of the weapons in the game are represented by procedurally generated particle systems, with a small collection of vairables that control their behaviour [8]. The IPCG system tracks only which weapons the player spends most time using, and then uses small neural networks to evolve new weapons

that are variations on the player's favourite weapons so far. This allows the player to experience more of the type of content that they prefer.

2.3.2 IPCG Academic Literature

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[16], 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 [11].

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. [**?**]

2.4 Problem Specification

An analysis and specification of the solution to the problem

3 Proposed System

3.1 Overview

The project is intended to provide the minimum functionality necessary to demonstrate a working IPCG system. This means that it will require the ability to monitor game state as the player progresses, form a model of some aspect of the state, and then generate further content based on the input from the model. 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.

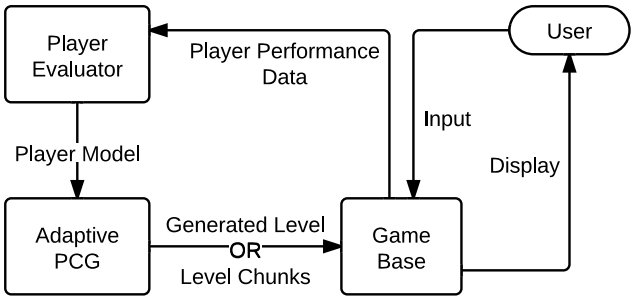

Figure 3.1: Data flow in proposed system.

3.1.1 Modules

Game Base

The game base need be nothing more than a simple 2D platformer engine. This module should handle all input and rendering activity, and should follow the Model-View-Controller architecture in order to facilitate monitoring and live updating. 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

Forming the first portion of the IPCG system, this should be an adaptive (parameterised) 2D platforming level generator. Building on the work of Compton *et al.*[4], 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

The second portion of the IPCG system should be essentially 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.

Evaluation

In order to perform retrospective evaluation of the system, it would be useful to integrate some facility for users to provide deliberate feedback outside of the player modelling system.

3.1.3 Justification

Justify: 2D platformer as simplest possible approach Use of Java, as it is my most well known language Custom-built base module, as will allow to be kept simple An alternative approach to the problem of generating a 2D platforming environment for use with DDA is presented by Sorenson *et al.*[17], 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. The 'functional'

requirements that follow specify functionality that the system must provide. The majority of them are generic to any IPCG system, as this project intends to cover that alone, and refinements specific to this project are specified where necessary. The 'non-functional' requirements define qualities that the system must adhere to, and in general 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.** Without at least a basic game environment in place, there will be no player interaction to collect data on, and nothing to generate content for. The system should provide a simple 2D platforming environment, with enough complexity to demonstrate working IPCG. Typical game mechanisms such as score or powerups are unnecessary in this context.
- **Record data on the game state and player's behaviour.** The system will need to be able to monitor and record data on many aspects of the game environment. Statistics such as number of mistakes, number and average width of gaps jumped, and time to completion of level will all be needed for the player evaluator. In addition, non-player data such as the length of the level may also need to be taken into account.
- **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.** Finally, the system will need to be able to make use of this model in order to generate future level chunks at a difficulty suitable for the player. To do this, the PCG must be able to evaluate the difficulty of its own output, and ensure that this matches the desired difficulty indicated by the evaluator.

In addition, in order to facilitate evaluation of the system itself, it should:

- **Keep a usable record of the data used to generate a level.** This serves two purposes: it should allow re-generation of a previous level when given the same input, for inspection of the generation process. It will also allow more accurate re-calibration of the classifier, should that be necessary.
- **Request feedback from the player.** In order to evaluate the effectiveness of the system, it would be useful to have feedback from the actual users. Rather than require the use of an external platform, an inbuilt feedback facility could store responses alongside the ingame data stored above.

3.2.2 Non-Functional

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

- **be written in Java.**
- **be presented as a basic platformer.** Though IPCG could be applied in some fashion to most genres, adjusting jump and obstacle placements is likely to be one of the most basic applications.

- **be confined to 2D.** As the degree of complexity of the generated content grows, so too does the complexity of the PCG required to generate it. A simple 3D terrain is relatively easy to generate, but in order to keep the complexity of a platforming game to a minimum it should be restricted to two dimensions.
- **limit the user to move and jump actions.** Many 2D platformers provide the player with novel interaction methods, which in this context would complicate the both the level generation and player evaluation systems. By restricting the player avatar's moveset, complexity is minimised.

In order to function satisfactorily as an interactive experience for the users, the system should 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

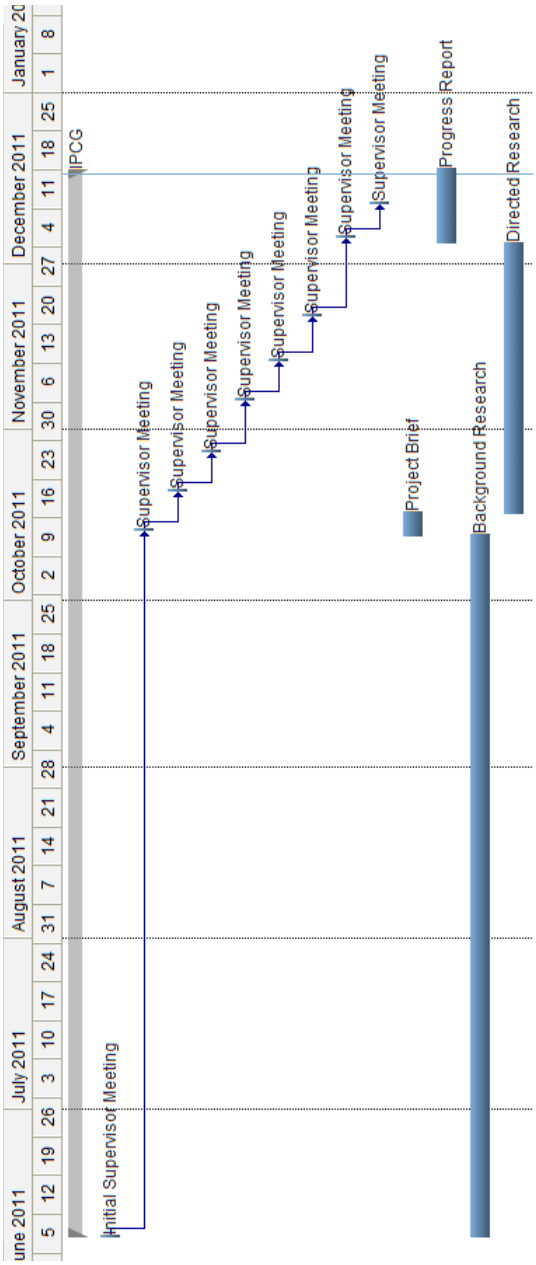

Figure 8.1: Work Completed

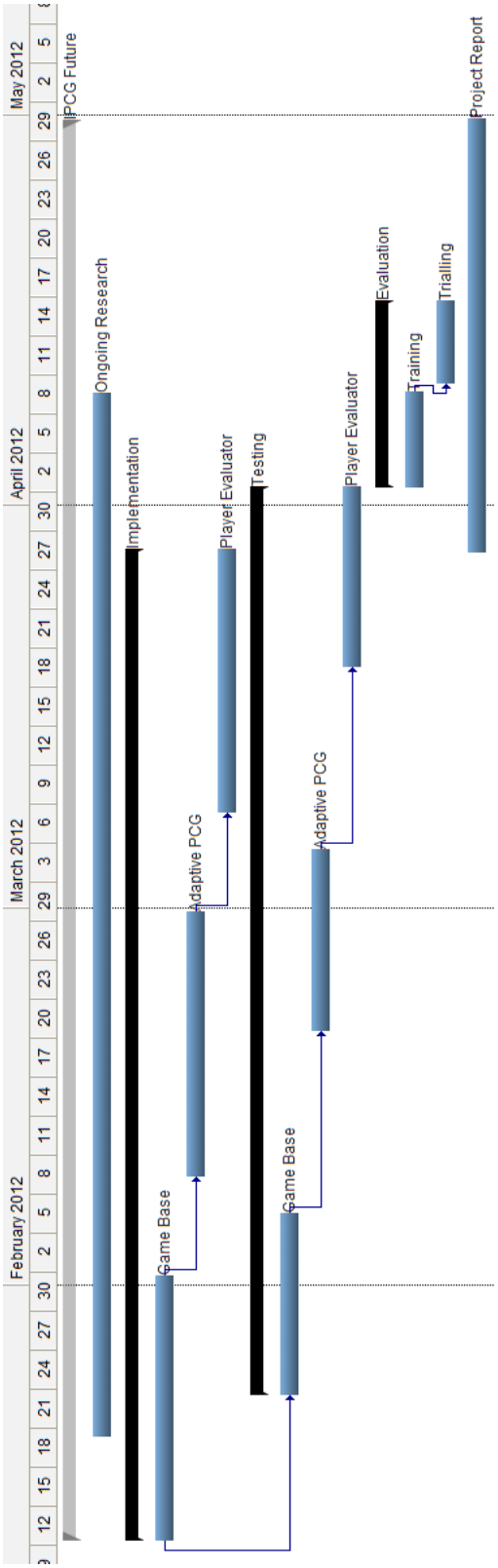

Figure 8.2: Work Remaining

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] Radiant story - the elder scrolls wiki.

[2] 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.

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

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

[5] I. Bell D. Braben. Elite, 1984.

[6] Andrew Doull. The death of the level designer. *ASCII Dreams*, Jan 2008.

[7] 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.

[8] 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.

[9] 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.

[10] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACE '05, pages 429–433, New York, NY, USA, 2005. ACM.

[11] 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.

[12] 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.

[13] 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.

[14] M Persson. Infinite mario bros! (online game), 2008.

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

[16] 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.

[17] 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