

Contents

1	Introduction	1
	Julian Togelius, Noor Shaker, and Mark J. Nelson	
1.1	What is procedural content generation?	1
1.2	Why use procedural content generation?	3
1.3	Games that use PCG	4
1.4	Visions for PCG	5
1.5	Desirable properties of a PCG solution	6
1.6	A taxonomy of PCG	7
	1.6.1 Online versus offline	7
	1.6.2 Necessary versus optional	8
	1.6.3 Degree and dimensions of control	8
	1.6.4 Generic versus adaptive	8
	1.6.5 Stochastic versus deterministic	9
	1.6.6 Constructive versus generate-and-test	9
	1.6.7 Automatic generation versus mixed authorship	10
1.7	Metaphors for PCG	10
1.8	Outline of the book	12
1.9	Summary	14
	References	14
2	The search-based approach	17
	Julian Togelius and Noor Shaker	
2.1	What is the search-based approach to procedural content generation?	17
2.2	Evolutionary search algorithms	18
	2.2.1 Other types of search algorithms	20
2.3	Content representation	20
2.4	Evaluation functions	22
	2.4.1 Direct evaluation functions	23
	2.4.2 Simulation-based evaluation functions	23
	2.4.3 Interactive evaluation functions	24

2.5	Example: <i>StarCraft</i> maps	24
2.6	Example: Racing tracks	25
2.7	Example: Board game rules	26
2.8	Example: <i>Galactic Arms Race</i>	27
2.9	Lab exercise: Evolve a dungeon	27
2.10	Summary	28
	References	29
3	Constructive generation methods for dungeons and levels	31
	Noor Shaker, Antonios Liapis, Julian Togelius, Ricardo Lopes, and Rafael Bidarra	
3.1	Dungeons and levels	31
3.2	Space partitioning for dungeon generation	33
3.3	Agent-based dungeon growing	38
3.4	Cellular automata	42
3.5	Grammar-based dungeon generation	45
3.6	Advanced platform generation methods	47
3.7	Example applications to platform generation	49
	3.7.1 Spelunky	49
	3.7.2 Infinite Mario Bros.	51
3.8	Lab session: Level generator for <i>InfiniTux</i> (and <i>Infinite Mario</i>)	52
3.9	Summary	53
	References	54
4	Fractals, noise and agents with applications to landscapes	57
	Noor Shaker, Julian Togelius, and Mark J. Nelson	
4.1	Terraforming and making noise	57
	4.1.1 Heightmaps and intensity maps	58
4.2	Random terrain	59
	4.2.1 Interpolated random terrain	59
	4.2.2 Gradient-based random terrain	61
4.3	Fractal terrain	62
4.4	Agent-based landscape creation	64
	4.4.1 Doran and Parberry's terrain generation	65
4.5	Search-based landscape generation	68
	4.5.1 Genetic terrain programming	68
	4.5.2 Simple RTS map generation	69
4.6	Lab session: Generate a terrain with the diamond-square algorithm	70
4.7	Summary	71
	References	71
5	Grammars and L-systems with applications to vegetation and levels	73
	Julian Togelius, Noor Shaker, and Joris Dormans	
5.1	Plants are everywhere	73
5.2	Grammars	74
5.3	L-systems	75

5.3.1	Graphical interpretation of L-systems	76
5.3.2	Bracketed L-systems	77
5.4	Evolving L-systems	78
5.5	Generating missions and spaces with grammars	78
5.5.1	Graph grammars	79
5.5.2	Using graph grammars to generate missions	81
5.5.3	Breaking the process down into multiple generation steps	82
5.5.4	Generating spaces to accommodate a mission	85
5.5.5	Extended example: ‘Dules	89
5.6	Grammatical evolution for <i>Infinite Mario Bros.</i> level generation . .	90
5.6.1	Backus-Naur form	91
5.6.2	Grammatical evolution level generator	92
5.7	Lab exercise: Create plants with L-systems	95
5.8	Summary	97
	References	97
6	Rules and mechanics	99
	Mark J. Nelson, Julian Togelius, Cameron Browne, and Michael Cook	
6.1	Rules of the game	99
6.2	Encoding game rules	100
6.3	Board games	102
6.3.1	Symmetric, chess-like games	102
6.3.2	Balanced board games	103
6.3.3	Evolutionary game design	104
6.3.4	Card games	109
6.4	Video games	110
6.4.1	“Automatic Game Design”: Pac-Man-like grid-world games	110
6.4.2	Sculpting rule spaces: Variations Forever	112
6.4.3	Angelina	112
6.4.4	The Video Game Description Language	116
6.4.5	Rulearn: Mixed-initiative game level creation	117
6.4.6	Strategy games	117
6.4.7	The future: Better languages? Better games? 3D games?	118
6.5	Exercise: VGDL	119
6.6	Summary	119
	References	120
7	Planning with applications to quests and story	123
	Yun-Gyung Cheong, Mark O. Riedl, Byung-Chull Bae, and Mark J. Nelson	
7.1	Stories in games	123
7.2	Procedural story generation via planning	124
7.3	Planning as search through plan space	125
7.4	Domain model	129
7.4.1	STRIPS-style planning representation	129

7.4.2	ADL, the Action Description Language	130
7.5	Planning a story	131
7.6	Generating game worlds and stories together	131
7.6.1	From story to space: Game world generation	133
7.6.2	From story to time: Story plan execution	137
7.7	Lab exercise: Write a story domain model	139
7.8	Summary	140
	References	140
8	ASP with applications to mazes and levels	143
	Mark J. Nelson and Adam M. Smith	
8.1	What to generate and how to generate it	143
8.2	Game logic and content constraints	144
8.3	Answer set programming	145
8.4	Perfect mazes	147
8.5	Playable dungeons	149
8.6	Constraining the entire space of play	153
8.7	Exercises: Elaborations in dungeon generation	156
8.8	Summary	156
	References	157
9	Representations for search-based methods	159
	Dan Ashlock, Sebastian Risi, and Julian Togelius	
9.1	No generation without representation	159
9.2	Representing dungeons: A maze of choices	160
9.2.1	Notes on usage	162
9.3	Generating levels for a fantasy role-playing game	162
9.3.1	Required content	164
9.3.2	Map generation	165
9.3.3	Room identification	165
9.3.4	Graph generation	166
9.3.5	Room population	166
9.3.6	Final remarks	168
9.4	Generating game content with compositional pattern-producing networks	168
9.4.1	Compositional pattern-producing networks (CPPNs)	168
9.4.2	Neuroevolution of augmenting topologies (NEAT)	171
9.4.3	CPPN-generated flowers in the Petalz videogame	171
9.4.4	CPPN-generated weapons in Galactic Arms Race	172
9.5	Generating level generators	175
9.6	Summary	178
	References	178

10	The experience-driven perspective	181
	Noor Shaker, Julian Togelius, and Georgios N. Yannakakis	
10.1	Nice to get to know you	181
10.2	Eliciting player experience	183
10.3	Modelling player experience	184
	10.3.1 Model input and feature extraction	184
	10.3.2 Model output: Experience annotation	186
	10.3.3 Modelling approaches	187
10.4	Example: <i>Super Mario Bros.</i>	188
	10.4.1 Player experience modelling	190
	10.4.2 Grammar-based personalised level generator	190
10.5	Lab exercise: Generate personalised levels for <i>Super Mario Bros.</i> .	192
10.6	Summary	192
	References	193
11	Mixed-initiative content creation	195
	Antonios Liapis, Gillian Smith, and Noor Shaker	
11.1	Taking a step back from automation	195
11.2	A very short design-tool history	196
	11.2.1 Mixed-initiative interaction	197
	11.2.2 Computer-aided design and creativity support	198
	11.2.3 Requirements, caveats, and open problems for mixed-initiative systems	199
	11.2.4 Examples of CAD tools for games	201
11.3	Interactive evolution	205
	11.3.1 User fatigue and methods of combating it	206
	11.3.2 Examples of interactive evolution for games	208
11.4	Exercise	211
11.5	Summary	212
	References	212
12	Evaluating content generators	215
	Noor Shaker, Gillian Smith, and Georgios N. Yannakakis	
12.1	I created a generator, now what?	215
12.2	Why is evaluation important?	217
12.3	Top-down evaluation via expressivity measures	218
	12.3.1 Visualizing expressive range	218
	12.3.2 Choosing appropriate metrics	220
	12.3.3 Understanding controllability	220
12.4	Bottom-up evaluation via players	221
	12.4.1 Which questionnaire should I use?	222
	12.4.2 Ways around the limitations of self-reporting	222
12.5	Summary	223
	References	223

A	Game-designer interviews	225
A.1	Andrew Doull	225
A.2	Ed Key	228
A.3	Michael Toy	230
A.4	Richard Evans	232
A.5	Tarn Adams	234