ADVENTURES IN DATA-DRIVEN GAME CONTENT GENERATION

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Gabriella Alves Bulhões Barros

September 2018

Acknowledgments

First, I would like to thank my supervisor Julian Togelius for his support, guidance and encouragement. It was a privilege to study and collaborate with you. I also thank the members of my defense committee: Andy Nealen, Clara Fernández-Vara, R. Michael Young and Antonios Liapis. And I thank Michael Green and Antonios Liapis for co-authoring so many of the papers that built towards this thesis, and I thank all the other co-authors I've had the pleasure of working with: Ahmed Khalifa, Mark J. Nelson, Tiago Machado, Noor Shaker and Thorbjoern Nielsen. I'm also grateful to CAPES and NYU Tandon for the support during my research, and to the IT University of Copenhagen for the wonderful time I was affiliated to them. I thank Scott Lee, Fernando de Mesentier Silva, Philip J. Bontrager, Christoffer Holmgaard, Benedikte Mikkelsen, Kaho Abe, Chris diMauro, Michael Cook, Aaron Isaksen, Phil Lopes, Amy Hoover, Rodrigo Canaan, and the many other colleagues, collaborators and friends I've met along the way. I also thank Edmund McMillen and Florian Himsl for making The Binding of Isaac, which helped keep me sane throughout this PhD. I thank my parents for providing for me and giving me a good education. And I thank Marco Scirea and Anita Simonsen for their friendship, for all the fun we had and for Mr. Sleepy. I cherish the time we spent together and miss them terribly. Finally, I thank Bruno Batista Soares, for his patience, support, companionship, trust and love. Without him, I would not have had the strength to come this far.

Dedication

To Bruno with all my heart. You make me dream of a house, a corgi and two chubby cats.

ABSTRACT

ADVENTURES IN DATA-DRIVEN GAME CONTENT GENERATION

by

Gabriella Alves Bulhões Barros

Advisor: Dr. Julian Togelius

Submitted in partial fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science)

September 2018

This thesis explores the field of data-driven procedural content generation for games. Data-driven games, also called *data games*, employ the use of data, often freely available data, to automatically and algorithmically generate game content. We use artificial intelligence, in particular evolutionary computation, to automatically generate content for strategy and adventure games using open data. Artificial intelligence methods are also used to automatically curate acquired data in order to generate content. We describe four systems that do data-driven procedural content generation, three of which are narrative-focused and iterate upon each other. We quantitatively evaluated three systems and describe an user study performed on how users engage and perceive the fourth and last system. Furthermore, we discuss the design space of data-driven games and the potential benefits and challenges of using multiple data sources as seed to game content. We also propose two dimensions used to map the design space of data-driven games, and discuss how our projects fit within these dimensions. Additionally, we investigate the consequences of juxtaposing data when generating content, and how absurd content can emerge and be dealt with. Finally, we discuss what are the different purposes of data-driven game design and what are the possible ethical and legal issues that can arise from it.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is an immense amount of freely available information on the Internet. Every minute, users publish 600 new page edits on Wikipedia, post 46,740 photos on Instagram and send 456,000 new tweets on Twitter [62], and currently over a billion different active websites domains can be found online [97]. It is possible to peruse this information as is, but there exists the possibility of visualizing it in a different format. The data may be hard to understand, exhausting or boring to read, and it may be desirable to interact with this data in different ways. One such way is via games, thus seeing and playing with the information in the form of game content. If we base our games on real data, we might also learn about the world as we play. A game could showcase particular parts or facets of the world's geography, simply by tweaking which parts of the data to use. Interacting with the game could mean interacting with a faithful representation of aspects of the real world. In this way, games can act as visualization tools, providing more playful means to learn and interact with information that would otherwise be tedious or difficult to understand.

However, two problems can emerge when trying to create games from data. Ironically, the first emerges from the vast quantity of data itself, as it can be a gargantuan effort to manually select and process even part of this information. The second problem regards the subjective nature of the game design process itself, as the person creating this content can overlook parts of the information due to their own personal biases. A solution to these problems is the use of procedural content generation (PCG) techniques to automatically generate game content using data. PCG in games is by now a flourishing research field

and a well-received game design practice [177, 180]. There are several good methods for generating, for example, levels for platform games or maps for strategy games. Some research attempted to generate complete games [216], including the game rules themselves; this has been done successfully for board games [34, 32] and so far with limited success for card games [73, 74, 232] and action-arcade games [53, 185, 145, 216, 220]. Furthermore, the amount of information available on the Internet makes it a nearly inexhaustible source of raw material for PCG, and efforts to generate content using open data have been made previously [51, 80, 39, 213, 86], albeit only for the creation of simple representations of often no more than one data source at a time. Independently of how many sources they use, games that employ data to drive the PCG process are called *data games* [80]. Additionally, works that focus on complete game generation and works that focus on using data to automatically generate game content ignore what I believe is an intuitive manner of passing information: stories. Automatic generation of plot and stories, inside and outside games, has been studied for several decades [109], but few automatically generated from open data, thus often not done with the purpose of translating information into a narrative.

I intent to fill the gap of narrative-focused data-driven procedural content generation. Additionally, this thesis further explore the field of data-driven PCG. The remaining of this chapter specifies this thesis' thesis statement, research questions, main contributions and structure. It also lists publications made during my PhD studies and notes on the style chosen for writing this work.

1 Thesis statement and Research Questions

This work aims at exploring the generation of narrative-focused data games and strategy data games, research areas so far neglected and underexplored. Through the course of this thesis, I will show that:

It is possible to generate engaging playable adventure games from open data.

In order to do so, this thesis raises and answers the following research questions:

Q1 How can we computationally generate games from open data?

Q2 How do people experience adventure games generated from open data?

Q3 What are the challenges in curating open data for procedural content generated in games?

Q4 How is data (mis)represented in generated games?

Q5 What can we do about misrepresentation of data in automatically generated games?

Q6 What are the consequences of computationally juxtaposing data without human curation?

I developed one map generation system for a strategy data game and three adventure data games, which explore different aspects of data-driven generation, iteratively improving on each of their predecessor. These projects serve as test bed for a user study that aims at evaluating how players experience these games. Additionally, drawing from my experience with these projects, I propose a data-driven game design process, and discuss the challenges that emerge from this type of games.

2 Main Contributions

This thesis draws from and contributes to several areas of research, such as procedural content generation, automatic game generation and story and plot generation, artificial intelligence (in particular evolutionary algorithms), and data games. I consider the main contributions to be as follows:

- The first exploration of plot generation in data-driven procedurally generated games.
- The introduction and discussion of a data-driven game design process, and the introduction of two dimensions of this design process.
- An exploration of the two dimensions proposed through various iterations of an adventure data game.

- Three complete data-driven narrative-focused games and a content generator for a strategy game
- The description and results of an exploratory user study on how players perceive and engage with a narrative-focused data-driven procedurally generated game.

3 List of Publications

This thesis is the result of collaborative work and has been published in 7 peer-reviewed scientific papers, consisting of 1 journal paper, 5 conference papers and 1 workshop short paper. This section lists these publications that are directly related to my thesis.

1 Gabriella A. B. Barros and Julian Togelius. *Balanced civilization map generation based on open data*. In Evolutionary Computation (CEC), 2015 IEEE Congress on, pages 1482-1489. IEEE, 2015. (Chapters 4 and 9)

2 Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. *Data adventures*. In Proceedings of the FDG Workshop on Procedural Content Generation, 2015. (Chapters 5 and 9)

3 Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. *Playing with data: Procedural generation of adventures from open data*. In Proceedings of the First Joint Conference DIGRA-FDG, 2016. (Chapters 5 and 9)

4 Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. *Murder mystery generation from open data*. In Proceedings of the International Conference on Computational Creativity, 2016. (Chapters 6 and 9)

5 Gabriella A. B. Barros, Michael C. Green, Antonios Liapis, and Julian Togelius. *Who killed Albert Einstein? From Open Data to murder mystery games*. In IEEE Transactions on Computational Intelligence and AI in Games, 2018. (Chapters 6 and 9)

6 Michael C. Green, Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. *DATA Agent*. Accepted at the International Conference on the Foundations of Digital Games, 2018. (Chapters 7 and 9)

7 Gabriella A. B. Barros, Michael C. Green, Antonios Liapis, and Julian Togelius. *Data-driven Design: A Case for Maximalist Game Design*. Accepted at the International Conference on Computational Creativity, 2018. (Chapters 3 and 9)

3.1 Other Publications

Additionally, the following papers were published and are tangentially related to this work, but are not included in the thesis as they diverge too much from narrative-focused data-driven procedural content generation.

1 Gabriella A. B. Barros and Julian Togelius. *Exploring a large space of small games*. In Proceedings of the IEEE Conference on Computational Intelligence in Games, 2014.

2 Marco Scirea, Gabriella A. B. Barros, Noor Shaker, and Julian Togelius. *SMUG: Scientific music generator*. In Proceedings of the Sixth International Conference on Computational Creativity, 2015.

3 Thorbjørn S. Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J. Nelson. *General video game evaluation using relative algorithm performance profiles*. In Proceedings of EvoApplications, EVOStar, 2015.

4 Thorbjørn S. Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J Nelson. *Towards generating arcade game rules with VGDL*. In Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG), 2015. Received a best paper award.

5 Ahmed Khalifa, Gabriella A. B. Barros, and Julian Togelius. *DeepTingle*. In Proceedings of the Sixth International Conference on Computational Creativity (ICCC), 2017.

6 Michael C. Green, Ahmed Khalifa, Gabriella A. B. Barros and Julian Togelius. *"Press Space To Fire": Automatic Video Game Tutorial Generation*. EXAG workshop at AIIDE. 2017

7 Michael C. Green, Ahmed Khalifa, Gabriella A. B. Barros, Andy Nealen and Julian Togelius. *AtDELFI: Automatically Designing Legible, Full Instructions For Games*. Accepted at the International Conference on the Foundations of Digital Games, 2018.

8 Michael C. Green, Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius. *Generating Levels That Teach Mechanics*. Accepted at the Procedural Content Generation Workshop at the International Conference on the Foundations of Digital Games, 2018.

4 Organizational Outline

This thesis is structured as follows:

Chapter 2: Background presents a review of the several fields related to this thesis.

Chapter 3: Dimensions of data-driven game design introduces the data-driven game design process that will be followed and describes two dimensions of the data-driven design space. The following chapters of the thesis will attempt at explore parts of this design space.

Chapter 4: Generating balanced strategy game maps from open data describes an approach for the use of topological information to generate maps for strategy games. Additionally, it introduces the limitations of prioritizing data fidelity at the expense of decreasing how much the data is transformed in the game.

Chapter 5: Data Adventures presents the first iteration of the *Data Adventures* series of games, shifting the generation of content towards adventure games. It presents a still limited data transformation and discusses this approaches' advantages and shortcomings.

Chapter 6: WikiMystery describes the second iteration in the series, *WikiMystery*, taking a more transformative approach than that of its predecessors. Furthermore, this chapter introduces a discussion on a problem of data-driven games: the generation of absurd and/or undesirable content due to errors in the original data and/or the juxtaposition of the data from different sources.

Chapter 7: DATA Agent moves further towards focusing on data transformation than on data fidelity. It describes the latest installment of the series, *DATA Agent*, and how it attempts to solve part of the problem of suspension of disbelief and absurdity found in *WikiMystery* with a meta-narrative.

Chapter 8: Evaluation of engagement in *DATA Agent* presents the methodology and results of an exploratory user study aimed at understanding how players see and engage with *DATA Agent*.

Chapter 9: Data-driven game design builds on the previous chapters to discuss purposes, tradeoffs and challenges related to data-driven game design, such as ethical and legal issues.

Chapter 10: Conclusion summarizes this thesis and discusses its contributions and limitations. Moreover, it further explains how this thesis' work answers each of the research questions brought up in Section 1. Finally, it presents future work and expansions beyond what this thesis achieved.

5 Notes on Pronouns and Style

Unlike this first chapter, the remaining of this thesis will forgo the pronoun "I" for "we". While I am the first author in the majority of the papers related to this thesis, much of this work is the result of collaboration. As such, I chose to use the word "we" thought the next chapters, referring to both myself and collaborators as the authors, but also at times to us (the authors) and the reader, when necessary. Additionally, this thesis uses the gender-inclusive style of "they" and "their" as genderneutral singular pronouns, in order to avoid attributing gender unnecessarily.

Chapter 2

Background

This work builds on research from several research fields. We use multiple *evolutionary algorithms* to *procedurally generate content* for various parts of a game, leading to the full *automated game design* of the game itself. We also focus on *story, quest and dialog generation* and *adventure games*, as we believe that it is easier to visualize and understand the *open data* used to generate *data games* in some form of narrative. In this chapter, we will screen these topics in the order mentioned.

1 Evolutionary Computation

The term *evolutionary computation* refers to optimization algorithms inspired by Darwin's theory of evolution [58]. The general idea, called *evolution by natural selection* but often referred to as *survival of the fittest* [196], is that, within a population of individuals in a given environment, those more adapted to the environment are more likely to survive and reproduce. Thus, their offspring are also more likely to better adapt to the environment. As this cycle repeats, new generations will improve to better survive. Similarly, evolutionary computation has a **solution** that can be modified by **operators** and can be measured by **evaluation function**.

Evolutionary algorithms are a subject of the field of evolutionary computation. Evolutionary algorithms are global search algorithms, which means that they keep not one, but a group of solutions (individuals), thus searching in multiple points of the search space

Algorithm 1 Generic process of a evolutionary algorithm.

1: **Initialize** population with random solutions
2: **Evaluate** each solution
3: **while Termination condition** is not met **do**
4: **Select** parents
5: **Recombine** pairs of parents
6: **Mutate** the resulting offspring
7: **Evaluate** mutated solutions
8: **Select** individuals for the next generation
9: **Replace** population with selected individuals
10: **end while**

simultaneously [230]. This solutions, called *phenotype*, can be represented in a variety of ways, but a popular form is as a array of numbers. A solution's representation is called *genotype*. Additionally, evolutionary algorithms introduce the idea of **recombination**, or *crossover*, where one or more individuals are created by the combination of other individuals. Typically, some individuals are **selected** to **reproduce** in order to generate a new population. This population, or part of it, is **mutated** and evaluated by the fitness function. The fittest individuals from the original population and/or the offspring **replace** the previous population in the next iteration. Every iteration is called a **generation**, and the algorithm usually **terminates** when it reaches a specified number of generations or when a satisfactory solution is found. Algorithm 1 shows a generic evolutionary algorithm. In this dissertation, we used the following evolutionary algorithms: evolution strategies, genetic algorithms and multiobjective optimization.

1.1 Evolution Strategies

Evolution strategies (ES) are a family of evolutionary algorithms, typically easy to implement and capable of finding satisfactory solutions, specially when the phenotype consists of arrays of real numbers [230]. ES rely on mutation operations to create variations in the offspring, removing the recombination part of the process (i.e. copying the parents phenotype without any crossover). Algorithm 2 shows the pseudocode for a $\mu + \lambda$ evolutionary strategy, characterized for the use of two parameters: μ and λ. μ defines the best individuals in the population, the "elite", which survives the iteration without modifications. λ represents the

Algorithm 2 Generic $\mu + \lambda$ evolution strategy.

1: **Initialize** a population of $\mu + \lambda$ randomly generated individuals
2: **while Termination condition** is not met **do**
3: **Evaluate** each individual in the population
4: **Sort** the population by decreasing fitness
5: **Remove** worst λ individuals
6: **Copy** the μ best individuals
7: **Mutate** the offspring
8: **end while**

individuals that are generated during the iteration. Popular improvements of ES include *covariance matrix adaptation evolution strategy* (CMA-ES) [88], useful in continuous search spaces, and *natural evolution strategy* (NES) [226].

1.2 Genetic Algorithms

Unlike ES, *genetic algorithms* (GA) focus on the crossover operator, but can still have mutation, albeit at a smaller rate [173]. Another difference is that GAs traditionally used bitstring representation, while ES traditionally used real-valued representation. The simplest approach to recombination is a single-point crossover: given two vectors of numbers, select a random point in the array. Then, generate two offspring, one with the first half of the first parent and the second half of the second parent, and the other with the remaining parts.

A popular variation of the GA is the *feasible-infeasible 2-population (FI-2pop)* [106]. It holds two different populations, one of feasible solutions and one of infeasible ones (per case-to-case manually defined requirements), each with its own fitness function: the feasible population aims at improving valid genomes, while the infeasible population tries to transform invalid genomes into valid ones. Solutions can move from one population to the other at every iteration.

1.3 Multiobjective Evolution

Selecting a evaluation function for an evolutionary algorithm can be challenging. Often, it is necessary to evaluate different aspects of a solution, some of which may partially conflict with each other. While it may be possible to calculate a weighted sum, there are different

approaches better suited to handle this situation. *Multiobjective evolutionary algorithms* use multiple partially conflicting fitness functions when searching for a **Pareto front** [233]. These functions are partially conflicting because there may be no solution that optimizes all of them simultaneously, but there is a possibly infinite amount of Pareto optimal solutions. A Pareto front contain a set of Pareto optimal solutions, which are solutions that cannot be improved in relation to a fitness function without degrading another [46].

Cascading elitism is a multiobjective evolutionary algorithms [212]. It uses a percentage of elitism and a set of ordered fitness functions. For each fitness function, it sorts the population and removes part of the individuals from the population (keeping only the percentage of the elite), sequentially, until all fitness functions have been used. Then it duplicates and mutates the remaining genomes.

Another multiobjective algorithm is the *nondominated sorting genetic algorithm II* (NSGA-II) [59]. NSGA-II uses non-dominating sorting over possible fitness, followed by calculating the distances between solutions, to identify the pareto front.

2 Procedural Content Generation

Procedural content generation (PCG) refers to the algorithmic, automatic creation of game content, with limited or indirect input from users [177]. The game industry has been using PCG since the late 1970's and early 1980's. Driven by constrained memory spaces, game developers of the time had to come up with creative ways of providing more content within constraints. Perhaps the first game to ever employ PCG, *Beneath the Apple Manor* [63] is a dungeon crawling game where the player is an adventurer in search of the Golden Apple. They would descend dungeons generated on-the-fly, fight monsters and loot treasure. Similar, but much more well known, is *Rogue* [218], which would eventually name a whole genre of games (Roguelikes) such as *Spelunky* [138] and *The Binding of Isaac* [68], both of which also use PCG. Another famous early example is *Elite* [5], a space exploration game capable of storing a seed that could generate eight galaxies, each with 256 unique planets. The seed, generated offline, was manually curated by the designers prior to the game shipping.

The increase in computational power seen in the following decades did not stagnate the presence of PCG in commercial titles, opening possibilities for more complex and ambitious techniques. *Diablo* [26], an action role-playing hack-and-slash dungeon crawler, procedurally generated maps and the monsters and items inside them. *Spore* [8] generated planets, creatures and animations, sometimes acting as an assistant to the player. *No Man's Sky* [92] ambitiously generates whole galaxies, down to details such as the flora and fauna of their planets.

There are various different methods for generating content for games [177, 230], which can be, for example, categorized as constructive or generate-and-test methods. Constructive methods generate content only once, following rules or operations that guarantee said content will be satisfactory. Constructive PCG methods are quite popular, specially for dungeon or terrain generation: approaches include cellular automata [101], binary space partitioning [176] and agent-based simulation [64, 176]. Platform game level generation has been achieved through the use of smaller parts, grouped to generate larger sections of levels: Smith et al. [187] uses the notions of rhythm, time and repetition to aggregate parts of levels, while Mawhorter et al. [129] joins pre-authored chunks of level geometry to create a 2D platformer level. An interesting example of constructive PCG in the game industry is *Borderlands* [81], where weapons are generated based on parameters and capping their quality based on the player character's level [82].

On the other hand, generate-and-test methods create and evaluate content, iterating over these two steps until a final solution is generated. This evaluation of content typically addresses gameplay (e.g. Does this content break the game?) or technical concerns (e.g. Does the content meet specified minimum requirements?). Unsatisfactory results are usually discarded at the end of an iteration, so new content can replace it. It is not uncommon to find examples of approaches that mixed these constructive and generate-and-test methods, as seen in the game *Dwarf Fortress* [21]. The next subsections highlight some constructive and generate-and-test methods: noise and fractals, cellular automata, grammar-based, search-based, solver-based methods, and machine learning approaches.

2.1 Noise and Fractals

A common approach for heightmap and terrain generation, noise algorithms, specially fractal algorithms, popular due to their easy use and fast running time. By representing the content as a two- or three-dimensional matrix (for example, for textures and heightmaps, respectfully), it is easy to create a randomized version of the content, interpolating over it to generate smoother surfaces, or use Perlin noise instead [162]. Perlin noise is a type of gradient noise that has a pseudo-random appearance, commonly used to create procedural textures. Fractals, on the other hand, subdivide a whole space over and over, selecting the middle points of the current space and altering their values. Fractals are typically used for terrain generation [135, 175]. A famous example is the Diamond-Square algorithms, commonly used for generating cloud textures or terrains[22, 103].

2.2 Cellular Automata

Cellular automata (CA) are constructive methods that are based on n-dimensional set of cells, which can have a finite number of states (e.g. on and off). A cell x is surrounded by a collection of cells, called x's neighborhood, and its state is affected by the neighboring cells' states. A mathematical function (typically Moore or von Neumann for two-dimensional cellular automata) defines the state of a cell at a time t based on its neighborhood's state at $t - 1$ [223, 230]. Using this method, it is possible to generate fast structures that are somewhat continuous and organic-like. For example, Johnson et al. uses this technique to generate infinite cave-like dungeons, creating small two-dimensional portions of the dungeon whenever the player moves to a new screen [101]. A similar approach combines CA to fractals to generate dungeons for the game *Galak-Z* [4] [7].

2.3 Grammar-based methods

Grammars are useful to generate content that can be specified in a set of rules. Rules define how nonterminal symbols generate terminal and nonterminal symbols, and can be expanded (from a initial rule) until a terminal symbol is generated for every nonterminal one expanded [44]. Originally, the rewriting of nonterminal symbols for their production

rules was made sequentially (i.e. left-to-right or right-to-left), but L-systems, a variation of this method, perform parallel rewriting. This means that, for a given state (e.g. string), every nonterminal symbol is expanded at once, given place to a new state independent of the previous [123]. These methods have been used to generate maps and levels [6, 65], and foliage [123, 167].

Althout it is not only aimed at games, *Tracery*'s [48, 49] flexibility and easy-of-use made it a powerful and popular tool for generating text content in and out of games. *Tracery* is a library for expanding grammar-based text, being capable of generating artifacts such as story [48, 49], poetry [49] and images [154], among others.

2.4 Search-based methods

A popular subsection of generate-and-test are search-based methods [217], which employ evolutionary algorithms or stochastic search/optimization approaches to create content. A search-based PCG solution consists of a *search algorithm* (e.g. evolutionary strategy, swarm optimization), a *content representation* (e.g. levels, textures, plot lines) and *evaluation functions*, which indicate the quality of a given content representation. Search-based techniques are versatile, and have been used to generate a variety of types of artifacts. Shaker et al. [178] used a neural network trained on players' input as the fitness function for level generation in a plaftormer game. *Ludi* evolved board game rules via a variant of genetic algorithm, evaluating board games through minimax playing agents [34]. Its representation of board games as strings allowed for simple description of games to be interpreted as expression trees. Aditionally, variations on this method allow for human interaction during the search. *Sentient Sketchbook*, a mixed-initiative game level design tool, provides suggestions for maps, treated as low-resolution sketches, through the use of various search approaches, such as novelty search and FI-2pop [118, 120, 165]. Hastings et al. [90] generate weapons for *Galactic Arms Race* based on how much users interact with previous weapons during gameplay, Cardamone et al. [38] evaluate racing tracks based on the player's preferences, and Ølsted et al. [149] create levels for a first-person shooter also using user's ratings. Other examples include generation of levels for platformers [100, 157, 195], maps

for RTS games [214, 215], racing tracks [212], terrain [10, 75, 76, 77], missions [65], grammar systems [150, 151], puzzles [9, 153], board game rules [94] and card game rules [73].

2.5 Solver-based methods

Solver-based methods use constraint solvers to search for satisfactory solutions [230]. Arguably the most famous solver-based approaches involve the use use of answer set programming (ASP). ASP solvers use a domain logic and constraints to explore a search space that satisfy said constraints [186]. *Variations Forever* is a game and research project that generates, using ASP, descriptions for game mechanics of two-dimensional arcade games [185]. The game *Refraction* also uses ASP to generate its puzzles [35]. However, methods are not limited to ASP. *Tanagra*, a mixed-initiative platform level design tool, uses a Java library as its solver [188, 189]. Its constraints focus on the solvability and aesthetic of platform levels.

2.6 Machine learning

Machine learning techniques for generating content have increased in popularity lately. The use of machine learning in the PCG pipeline is not new, but was mostly delegated to evaluate content generated via other methods [117, 179]. However, content have actively been generated using artificial neural networks and Markov models. Markov models variations, such as n-grams, have created levels for platform games by breaking down the levels into parts and building conditional probability tables based on them [57, 190, 191]. Summerville et al. [200] expanded these methods using Monte-Carlo tree search (MCTS) [33], employing the learned probabilities during the generation of the levels.

While Markov models tend to be simple and fast, artificial neural networks can be more complex and slower to train [230]. Summerville and Mateas [199, 198] and Hoover et al. [95] generate levels for *Super Mario Bros* [147]. Summerville and Mateas used Long-Term Short Memory Recurrent Neural Networks, which is able to recall information for a number of time steps, as opposed to classic neural networks. Hoover et. al used NeuroEvolution of

Augmenting Topologies to evolve their neural networks, using a representation borrowed from musical composition.

3 Automated Game Design

Among the multitude of game content that can be generated lies the possibility of generating the whole game itself. Complete game generation is an active area within PCG, focused on the automation of the rules and mechanics that define traditional or video games. Here we survey generation of rules and aspects that directly affect how the game is played. Thus, we do not focus on, for example, board generation that affects the board representation but has no bearing in the fundamental rules of the game.

3.1 Card- and board-game generation

The main examples of traditional game generation are METAGAME [158] and Browne's *Ludi* [34, 32]. METAGAME was the first work to ever tackle game rule generation [158], focused on generating chess-like games. It created two-player symmetric turn- and grid-based games, which resembled variants of chess. METAGAME used a constructive approach, randomly sampling rules from a game grammar that contained a set of pre-encoded rules, then checking for playability. Browne's *Ludi* is perhaps the most successful automatic game design example. It uses search-based PCG (most specifically, genetic programming) to generate rules for two-player combinatorial board games [34]. It represents games as sets of *ludemes*, which are units of game information, and evaluates them by having an agent play against itself. *Ludi* is considered so successful because it was the first system to automatically generate full commercial games: *Yavalath* [31] and *Ndengrod*.

Additionally, there has been some research in card game generation. Font et al. defined a description language for card games, such as *Uno*, *Texas Hold'em* and *Blackjack*. They then tried to evolve games using this language, most of which have been deemed to be unplayable [73, 74]. In a smaller context, Zaidan and Góes developed Evolvestone, a generator of *Hearthstone* [28] variants [232]. While *Hearthstone* is a digital card game, its rules can easily be transposed to a traditional one. Zaidan and Góes first defined a series

of parameters to describe different aspects of the game, then explored this space using a genetic algorithm, evaluating individuals through MCTS simulations.

3.2 Video game generation

When we transition from traditional to video games, the possible *generative space of games* expands significantly, and generating a complete game becomes significantly harder. A generative space of games consist of all games that can be specified using a given rule encoding [177]. A popular approach to counter a large generative space is to limit which games to generate, such as only create two-dimensional arcade games. For example, Togelius and Schmidhuber [216] restricted the search space to only two-dimensional games similar to *Pac-Man* [12]. All generated games occupy a 15x15 grid, with either free space or walls. They can terminate after a defined number of timesteps or after a certain amount of score is gathered, and contain a player character (a cyan dot) and several *things* (red, blue or green dots), that can represent a variety of concepts, like food, enemies and so on. Collision between the player and things, or between things, can result in nothing happening, one or both objects dying, and one or both objects teleporting to a different position in the grid. Togelius and Schmidhuber evolved games using a $5 + 5$ ES, where the fitness function calculated how "learnable" a game is by playing it with a random agent and an evolved agent that could better play the game as it improved.

Variations Forever [185] has a similar, but larger, generative space. It uses ASP to search for two-dimensional games that satisfy game designer's specific constraints, such as "it should be possible to win by having no red things". These games are played on a grid of size determined by the generator, with objects of different colors that behave in preprogrammed ways, which are inspired by *Pac-Man*, *Asteroids* [11] and *Rogue*. Part of the player's goal is not only to beat the game, but also to figure out how to play it. Unlike Togelius and Schmidhuber [216], however, *Variations Forever* as a generator is more focused on exploring a slice of a generative space, rather than generating one singular game with a specific purpose.

Another arcade-like game generator is *ANGELINA* [53, 54]. Its first few iterations

focused on two-dimensional arcade-like games, using a cooperative coevolutionary system to generate, separately, the game's map, ruleset and the entities [50, 53]. Each part is evolved separately, each with its own fitness functions, but they are tested together to evaluate how well the general game is. While *ANGELINA*$_1$ (i.e. the first version of the system) focused on arcade-like games, and *ANGELINA*$_2$ and *ANGELINA*$_3$ focused on the Metroidvania subgenre of arcade games, subsequent versions expanded towards treedimensional games [52, 54].

Game-O-Matic distinguishes itself from the aforementioned works by focusing less on the specific lower-level mechanical aspects of the game, and more on the general relationship between entities present in the game [219]. It generates games in the form of a concept map, where nodes represent actors and edges represent their relationships and how they affect each other. This concept map is seeded by an user, treating nodes as nouns and edges as verbs. The system has a set of micro-rethorics, representational units of gameplay from which one can derive meaning [220] (e.g. a "heal" micro-rethoric can be derived from the mechanics of two objects colliding and one or both receiving health points), and each valid verb is associated to a subset of micro-rethorics. For every verb in the map, *Game-O-Matic* selects one of the possible micro-rethorics that that verb has, and adds modifiers and conditions to further define their relation (e.g. their position on screen) and the game's win/lose conditions.

3.2.1 Game description languages and game generation

It is possible to use a game description language (GDL) to constrain the generative space, similar to what Font et al. did for card games [73, 74]. While using a GDL obviously constrain the space fo possible games to those that the language can describe, it can allow for algorithms to more easily navigate in that space. One such GDL is *PuzzleScript* [112], aimed at describing two-dimensional puzzle games similar to *Sokoban* [204]. Lim and Harrell describes a evolutionary approach to generating *PuzzleScript* games, more specifically 3 out of the 8 parts of a *PuzzleScript* game file: rules, levels and win conditions. It evaluated games via simulation, looking for aspects such as feasibility, repetitiveness and difficulty of solutions.

Another GDL is the Video Game Description Language (VGDL) [67], part of the General

Video Game — AI (GVG—AI) framework [161]. VGDL can represent two-dimensional arcade, action and/or puzzle games. VGDL contains two main parts: a game description and a level description. The game description has information about the game entities (sprites), their interactions, the game's win/lose conditions (termination set) and how each entity is represented in a level. GVG—AI is a Java implementation of VGDL, originally created for the GVG—AI competition. Barros and Togelius was the first to attempt to generate VGDL descriptions [19], using a grammar to explore the possible space of VGDL games. Nielsen et al. [145, 146] expanded on this work, using relative algorithm performance profiles (RAPP), which is the relative difference in performance between given AI agents [145], to create VGDL descriptions. They hypothesized that a better designed game would have a higher difference in relative performance between a good agent and a bad agent, while a worse designed game would have a lower difference. That essentially means that, given two games and two agents of different skill level, the different between the worse and better agent is much higher in the better game of the two. Unlike Nielsen et al. [145, 146], Khalifa et al. [105] focused on generating only the rules for VGDL games, ignoring level descriptions. It generated games in three different ways: randomly, using constructive methods and evolving with FI-2pop. Finally, Gow and Corneli [85] used conceptual blending on two game descriptions (*Frogger* and *Zelda*), attempting to blend their sprites and rules and creating variations that drew elements from one another.

Our work differs from the aforementioned works for a) focusing on a genre that is often overlooked in automatic game design, which is that of adventure games; and b) making large use of open data in the generation process.

4 Story and Plot Generation

This section focuses on story and plot generation in games. It highlights the use of story generation in adventure games, describes briefly the broad field of narratology, and depicts some methods used to automatically or semi-automatically create stories.

4.1 Narratology

Narratology has been of great influence to story and plot generation. Narratology is "(...) the ensemble of theories of narratives, narrative texts, images, spectacles, events; cultural artifacts that 'tell a story'" [16]. Perhaps the most used example in story generation is Propp's morphology of a folktale [166], which has been used extensively. Propp analyzed Russian folktales and described 31 narrative elements (*functions), e.g. villainy or acquisition of a magical agent*) and 7 archetypes, such as hero, villain and helper.

Another narratology influence is Campbell's Hero's Journey [36], used by Champagnat et al. [41] to generate an interactive plot. The Hero's Journey, or *monomyth*, consists of three main sections (the separation, the initiation and the return), and is briefly defined by Campbel [36] as:

> "A hero ventures forth from the world of common day into a region of supernatural wonder: fabulous forces are there encountered and a decisive victory is won: the hero comes back from this mysterious adventure with the power to bestow boons on his fellow man".

4.2 Adventure, Role-Playing and Board Games

Adventure games are a popular game genre where players control a character in the game world, interacting with it through object manipulation in order to solve puzzles, focusing heavily on space and action exploration and story-driven experiences [71]. The last feature (story-driven experience) encourages that greater attention be taken towards how the story is conveyed, either via dialogues, scenarios, visuals or any other form of expression. The genre originated with *Colossal Cave Adventure* [228], but later improvement in graphics and user-interfaces led to several variations, including the classic text-based interactive fiction [137] like *Zork* [193], point-and-click adventures like *Day of the Tentacle* [125] and *Thimbleweed Park* [203], visual novels like *Phoenix Wright: Ace Attorney* [37] and *Zero Escape: Virtue's Last Reward* [45], and action-adventures such as *The Legend of Zelda* [148] and *Tomb Raider* [69].

When one thinks about story and plot generation, it is easy to overlook traditional venues,

such as tabletop role-playing games (RPGs) [126] and adventure board- or card-games [197]. In tabletop RPGs, a dungeon master will typically prepare an adventure or use a pre-made one, to be played by the remaining group of players. The dungeon master can follow the story or be flexible and create plots as diversions happen, but the actions and choices of players greatly influence the final narrative.

4.2.1 Story and plot generation in board- or card-games

Sullivan and Salter [197] described four types of adventure board- or card- games: *Story crafting games*, *unordered story games*, *ordered story games*, and *story exploration games*. In *story crafting games*, players are responsible for creating the majority of elements in the story, being given only a general narrative structure and/or specific elements that must appear in the narrative. An example is the card game *Gloom* [13], where each player takes control of a set of individuals in a family, and the goal is make them as miserable as possible in order to kill, as only dead characters awards points at the end of the game. The player with the most point wins. Players must play event cards, which make character happier/sadder, and death cards, which kill them and can only be played on characters when they are miserable. To play a card, however, the player must continue the story of the character affected, remembering whatever events happened to them before in order to keep the story consistent.

Unordered story games, on the other had, have elements and events on the story specified, and the player's part in generating the narrative is to specify in which order they appear. *Betrayal at House on the Hill* [229], for example, is a board game where players both cooperate and compete against each other. In the first part, they cooperate to explore a haunted house, setting up the layout of the house with randomized tiles and interacting with event cards. When a specific trigger happens, the game switches and one of the players become "the traitor", receiving a new backstory, new goals and rules. The game contains different scenarios that can be played out, and as the players work towards their end goals (both the traitor and the cooperative group), their actions and the events they trigger contribute to form a narrative.

On the other hand, *ordered story games* allow players to create most events and elements,

but the order in which these must appear is specified by the game itself. Sullivan and Salter [197] found only one example of this type: *Mysterium* [121], a game about a ghost trying to help a group of mediums to find who murdered them, where and how. One player takes the role of the ghost, while the others are mediums. The ghost can only communicate with cards, each of which contains abstract, dream-like images. In the first half of the game, the ghost must lead each player to a separate set of suspect, place and weapon, which are randomized. In front of the players lie a number of options for each category, and the player gives cards to each of them separately, hoping they can link the content of the card with their options. The ghost must always give clues in a specific order (first suspect, then place, then weapon), and if a player cannot find their correct card, this player cannot move forward and receives other clues for the same category on the following round. While the order of the categories is fixed, the narrative of the game changes based on which cards are laid out as options, and which cards the ghost draws and chose to give as clues.

Finally, *story exploration games* contain a pre-made story and a set plot order, but it is up to the players how to explore the space in order to advance the story. An example is *T.I.M.E. Stories* [3], a cooperative board game that sets players as time agents eliminating paradoxes in the past and future, and each expansion of the board game tells a different story. Each player takes the role of one character in the plot, and the story is told via cards in the deck. Depending on which locations the players visit and what actions they take, they uncover different parts of the narrative.

4.3 Computational Story and Plot Generation Methods

This section presents a non-exhaustive discusson on highligthed plot and story generation systems. Kybartas and Bidarra [109] presented an extensive survey on the topic, where they define plot and story as follows: **plot** is the set of events tied together in temporal ordering and causality, typically consisting of one or more actions that are caused and/or affect the **space**. The **space** is the set of all characters, settings, objects, etc, that exist, physically or abstractly, in the narrative. The **story**, on the other hand, encompasses what happens in the narrative, including its plot and space. This section will not focus on **discourse** (i.e. the way

a story is told [109]), but rather on how the general plot and/or story is created. For example, while *BRUTUS* excels at how well-written its generated stories are, we are more interested in how it uses a grammar-based system to plot out the betrayal stories it creates [30].

Novel writer is possibly the earliest know example of a computational system generating stories [108]. It is a simulation-based system that creates murder stories happening within a weekend houseparty. Motives for the murder arise as the simulation unfolds, driven by the story's events. Characters are modeled in the simulator, and a set of grammar rules define how they behave as time passes, in a fictitious universe represented by a semantic directed graph, where nodes are elements and edges are relations.

4.3.1 Planner-based systems

Several works use planning techniques to generate plot or stories. *TALE-SPIN* [133] is one of the first to do so, creating stories by planning with simulating characters, based on character goals and an initial state in the world, which can be modified by changing the rules that underlie the system's solver. It models characters, adding to them personality traits (i.e. kindness, vanity, intelligence and honesty), and their relationships to each other.

UNIVERSE [113] uses a similar approach to *TALE-SPIN*, but instead focus on *author goals*, as opposed to *TALE-SPIN*'s *character goals*. *UNIVERSE* uses planning to create plots for a melodrama, aiming for interesting and extended stories. In this case, it viewed conflict between characters as an indication of how interesting a story is. Similarly, *GADIN* [18] also uses planning to create soup-opera-like stories, but differs from *UNIVERSE* because it is interactive, putting the user in the position of one of the characters.

While *TALE-SPIN* and *UNIVERSE* focus on different goals, *Fabulist* generates stories that attempt at respecting character believability while satisfying human author goals [171, 169, 170]. It uses a combination of two planners to find character actions that are coherent to author goals, and when necessary infer information that was not determined by the author.

Story Canvas [184] builds upon the *Wide Rule* [183] system, which is a visual story authoring tool inspired by *Universe*, and uses planning to generate interactive stories. *Prevoyant* is a planning-based system that focus on generating stories that elicit surprise in the reader [15], while Ware et al. emphasized conflict instead of surprise [225]. Barros

and Musse's *Fabulator* prioritizes tension arcs while trying to maneuver interactions that deviate from the plot [20]. Porteous et al. [43, 164] also used planning to adapt the story of an interactive version of Shakespear's *Mechant of Venice* in order to fit user's actions that deviate from the original story. Chang and Soo [42] used a different Shakespeare play, *Othello*, as the setting for their planning-based system. Finally, *Mimesis* [231, 168] contains a divided architecture for interactive storytelling, where part of it, the *Mimesis Controller*, uses planning techniques to generate a narrative as a sequence of character actions.

4.3.2 Grammar-based systems

Another popular technique for plot generation are grammars, used by *BRUTUS* [30]. Lakoff [111] used a grammar based on Propp's morphology [166] to generate plots. Similarly, one of the approaches proposed by Gervás [83] expanded used grammars automatically extracted from an appendix in Propp's work to generate plots. On a different domain than Propp's folk tales, *GESTER* generates plots in the style of medieval French epics [159].

Colby [47] created a grammar inspired by Eskimo folktales, that builds on top of three different categories: motivation, engagement and resolution. Motivation defines what is the protagonist's motivation, such as "being banished from house for refusal to marry" or "a relative is taken and held captive". Engagement consists of actions taken by the protagonist or other characters, such as "protagonist is invited to a house where hospitality is given" or "protagonist engages in a magical contest". Resolution is a solution to one or multiple engagements, e.g. "adversary escapes" or "protagonist pretends to heal a wound". Colby's grammar creates sequences of moves, which consist of a motivation, one or more engagement and a resolution. A unit of each of this category replaces the category once the sequence is fully generated.

ReGEN [110] is a gramar-based system that uses graph rewriting. It contains two separate directed graphs, one that represents the story world and one that represents the characters and their relations to each other. It searches the world graph for a potential story, which is defined by a initial graph that effectively is a pattern that can be transformed into another pattern, similar to a grammar. When it finds a pattern existing in the story world, it generates it rewrites the plot with what it found. It repeats this process with a different set of

rewriting rules, which aim at making the story "more interesting". Finally, it simulates the plot generated to find how it affects the story world graph.

4.3.3 Case-based reasoning-based systems

Case-based reasoning has been applied to story and plot generation, but typically involves a high level of human authoring to build its knowlede base. For example, ProtoPropp [61, 84] uses case-based reasoning over an ontology based on Propp's work [166] to generate a plot for a folk tale.

MINSTREL [221] creates stories revolving around Arthur of Camelot and his knights using case-based reasoning. *MINSTREL*'s solution works well for both character goals and author goals, by, when approached with a problem, recalling previous solved problems and adapting similar solutions, but uses a massive about of authored knowledge to generate its stories.

MEXICA [160] uses case-base reasoning to generate plots about Aztecs. *MEXICA* uses a database of schemas, i.e. stories parts, to identify possible actions that can be taken in the current setting, incrementally building the plot with them. The system also models each character separately, but takes into consideration their relationships with one another. Whenever necessary, the system alternates between two stages: *engagement*, when forwarding the plot using schemas, and *reflection*, undoing parts of the story that present inconsistencies of fail to satisfy certain criteria.

4.3.4 Evolutionary systems

An example of evolutionary narrative system is *HEFTI* [152], which uses a genetic algorithm to create story plots. It contains a large set of human authored building blocks (plot templates and plot elements), and evolves a sequence of story events. The representation of the gene is a list integers that represent XML templates, and it uses one- and two-point crossovers and mutation to reproduce individuals. McIntyre and Lapata [132] also used GAs to generate plots. Unlike Ong and Leggett [152], their representation and operators use graphs, and generates the story templates and elements via Natural Language Processing.

Alternatively, Nairat et al. [139, 140] evolved not the plot, but the characters in the story. These characters are represented via resources (e.g. wealth, health), emotions (e.g. happiness, fear), and feelings towards each other (e.g. love/hate, friendship). Additionally, their system contains a series of actions, both general and agent-specific, from which the agents can chose during simulation. The evolution is interactive, relying on a human to select which agents to reproduce or not. The system outputs a storyboard of the sequential actions simulated by the characters at the end of the evolution.

4.3.5 Search-based systems

Scheherazade [115, 114] is unique in the sense that it applies a search-based algorithm to find plots within a plot-graph, which is created from crowdsourcing stories about in a specific domain. The system works by crowdsourcing stories to generate story graphs, and then creating stories from said graphs. The first step consists of gathering several story variations manually written in a simple format, i.e. simple sentences with only one verb and no pronouns. The system uses these stories to identify elements and events, and in which order they appear. This allows *Scheherazade* to build a plot-graph, which can be traversed on the second step by a search algorithm. This algorithm tries to identify a path between the beginning of the story and one of the possible finales, giving the user the option of using a deterministic way or a stochastic one.

4.3.6 Simulation-based systems

Perhaps the most successful interactive system, *Façade* [127, 128] is an interactive story game that puts the player in the role of a close friend of a couple, Trip and Grace, during a evening get-together at said couple's apartment. *Façade* has a virtual simulated world, but updates which behaviors Grace and Trip can have and what context they are in at every *story beat*. These beats are small authored units of dramatic action that more the plot forward.

PASSAGE [205, 206] uses player modeling to adapt the story and quests to better suit the user's playstyle. It divides playstyles into five types (i.e. Figher, Method Actor, Storyteller, Tactician, Power Gamer), and weights players actions to find their preferences. When PASSAGE generates part of its stories, it draws from an human authored library of possible

events, placing the output story into the game world in the way it believes will better suit the player.

Prom Week [130, 131] uses the *Comme Il Faut* [131] social engine system to model the game's world, including its characters and their relations, in a high-school setting. In the game, the player controls one of the many characters modeled, and the goal is to manipulate the social space to achieve specific world states, often involving altering the relationships between two characters. The system requires human authored space definition, including main events that happen at specific moments (such as the prom that titles the name itself), but the plot is defined by the player through their interactions.

Another system that heavily relies on social interaction is *Versu* [70], an agent-driven simulation system/game. Unlike *Façade*'s approach of defining behaviors for specific characters, *Versu* tries to author behaviors in a character-agnostic fashion, making it possible to switch characters in the simulation at play time without altering the underlying system. On a similar approach, Virtual Storyteller [202, 201] also uses multi-agent simulation in its plot generation pipeline. It uses a graph consisting of six story elements (i.e. Setting, Event, Internal Response, Goal, Attempt and Outcome) to implicitly define a hierarchical structure for part of the story, which consists of several graphs. The simulation allows agents (i.e the characters) to modify the space of the story, in a sense prioritizing character goals. Additionally, it uses a drama manager capable of adding characters and altering character goals in order to stir the plot on specific directions.

5 Data & Games

A large part of this work involves the transformation and (re)use of data. In a world of ever-more ubiquitous technology, the amount of data we consume daily is rapidly increasing. Rob Kitchin [107] defines data as:

> "(...) the raw material produced by abstracting the world into categories, measures and other representational forms — numbers, characters, symbols, images, sounds, electromagnetic waves, bits — that constitute the building blocks from which information and knowledge are created. Data are usually representative

> in nature (e.g., measurements of a phenomena, such as a person's age, height, weight, colour, blood pressure, opinion, habits, location, etc.), but can also be implied (e.g., through an absence rather than presence) or derived (e.g., data that are produced from other data, such as percentage change over time calculated by comparing data from two time periods), and can be either recorded and stored in analogue form or encoded in digital form as bits (binary digits)."

Data can also appear in numerous forms [107], such as **quantitative / qualitative** — Quantitative data is numeric, while qualitative is not (e.g. photos, text, etc) –; **structured / semi-structured / unstructured** — Structured data obeys a specific format (e.g. information in a relational database), while unstructured has no defined general presentation (e.g. social media posts). Semi-structured lies in the middle of these both (e.g. XML) —; and **primary / secondary / tertiary** — data is primary if generated by a researcher within a research design of their own, secondary if made by someone else and available to reuse or analyze, and tertiary if it is derived from other data. Independent on its type, data acts as an abstraction of the real world, serving as foundation for a deeper undertanding of reality [72].

5.1 Open Data

Open data is a movement based on the idea that data should be freely available to all, and aims to popularize the production of knowledge and information, as opposed to restrict the use of data to those who can produce or buy it [107]. Famous examples are *Wikipedia* [227] and *OpenStreetMap* (OSM) [87]. *Wikipedia* is a web-based, openly-editable encyclopedia, while the OSM project aims to create and provide maps that are free to view, user and modify. Several governments and organizations have engaged into efforts for providing transparency of information to (their) citizens, such as the United States [1], New Zealand [2], the United Kingdom [3], Kenya [4], the United Nations [5] and the World Bank [6]. The ever-increasing amount

[1] https://www.data.gov/
[2] https://www.data.govt.nz/
[3] https://data.gov.uk/
[4] http://www.opendata.go.ke/
[5] http://data.un.org/
[6] https://data.worldbank.org/

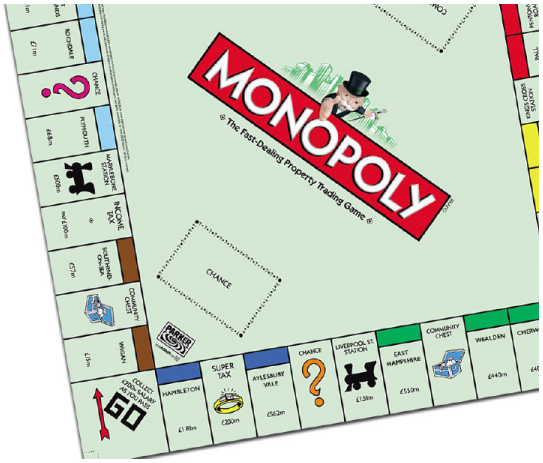
of data on the Web led to the creation of *Linked Data*, data "(...) published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets" [25]. *DBpedia* [14] is such an example: its goal is to extract, structure and make available information from *Wikipedia*, linking it to other (mostly semantic) datasets. Linked data uses the resource description framework (RDF) format [155]. RDF links describe data using triples of *subject*, *predicate* and *object*. The *subject* identifies a resource, the *object* identifies either a resource (in the form of uniforn resource identifiers (URI) [23]) or a string value, and the *predicate* defines the relation between the two. For example, DBPedia contain a tuple <*Albert Einstein*, *birthDate*, "1879-3-14">, where the subject (*Albert Einstein*) and predicate (*birthDate*) are resources, and the object ("1879-3-14") is a string.

5.2 Data Games

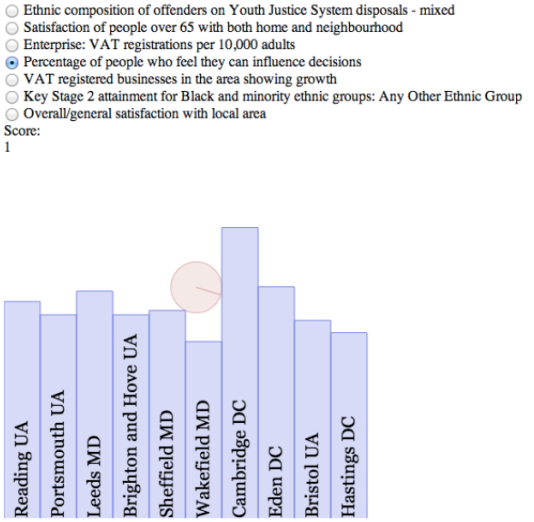
Data games use information derived from outside the game, often real world open data, to generate in-game content [79, 78, 80]. The term, first proposed by Friberger and Togelius [79, 78], refers to a class of games that often acts as a form of interactive visualization, allowing players to see, interact with and explore the information in new, playful ways, as opposed to simply e.g. reading it in an article or spreadsheet. While this area is still fairly underdeveloped, in comparison to other fields, this subsection will highlight some efforts made into developing data games.

Alongside a definition of what "data game" are, Friberger and Togelius [79] also used geographic and demographic data from the United Kingdom to create a *Monopoly* [89] board generator, called *Open Data Monopoly*. It requested user input to select which (among 72) indicators of prosperity to use, such as criminality level or inequality gaps, and whether these indicators affect positively or negatively the wealth of a place. They used a $50 + 50$ ES, where the fitness function was a weighted sum that took into account the places' prosperity, geographical distribution and notability. Figure 2.1a shows a graphical representation of a generated board.

Still related to traditional games, *Open Trumps* [80, 39] is a data game version of *Top*



(a) A graphical representation of a board generated by Open Data Monopoly. Source: [79].



(b) Screenshot of Bar Chart Ball. On top, a series of indicators, of which one is selected. On the bottom, a ball sits on top of the game chart. Source: [213].

Figure 2.1: Examples of data games.

Trumps [66], a card game that uses themed decks. Each deck has a list of categories with numerical values present in every card, and players chose categories based on the cards current at hand, in order to best their opponents. *Open Trumps* uses a $1+1$ ES over a dataset provided by the user, and its fitness was based on how many generated cards could outdo all other cards using one of the selected categories.

Moving on from traditional games to video games, there is *Bar Chart Ball* [213, 80], a game that uses the same data set as *Open Data Monopoly*, but gameplay happens on a graph, as shown in Figure 2.1b. A ball is placed over a bar chart. Players can select different indicators, which modify the state of the chart, allowing the ball to move. The goal is to move the ball where the player wants (e.g. off the chart, to the middle, etc), but to do so the player must understand how the indicators relate to different bars in the graph, which represented different locations from the dataset.

While it does not claim to be a data game, *A Rogue Dream* [51] relies heavily on information obtained online. The game itself is a roguelite (a less punishing version of a roguelike), but the player's character is defined by the user. At the start of the game, it prompts the player with "Last night, I dreamt I was a...", and it is up to players to complete

the sentence with a noun, which defines the playable character. The system uses Google autocomplete to find completions for variations of "Why do [noun]...", which identify the character's abilities, enemies, goals and items. Additionally, for everything that needs a sprite (e.g. the playable character, items, enemies), the system uses *Spritely* [51], a tool for querying the web for images and creating small sprites out of them, to query Google Images, Open Clipart and Wikimedia Commons and generate images to be used in the game.

Most notably are commercial games that can, to some extent, be categorized as data games — often through their use of real word maps. Niantic's *Ingress* [143] and *Pokemon Go* [144] both use geological information in their gameplay loop, rendering maps based on players' locations and adding, for example, portals (in the former) or monsters (in the later) depending on where they are. *911 Operator* [102] is a strategy game that also uses real world locations as its in-game maps. Additional examples include *Geocaching* [86], *BotFighters* [98] and *Turf Wars* [56].

Chapter 3

Dimensions of data-driven game design

The unprecedented availability of digital data impacts most human endeavors, including game design. In particular, freely available data can be combined with procedural content generation (PCG) and computation creativity to create systems that can generate games (or game content) based on open data. These games have previously been identified as "data games" [80].

This chapter explores some of the aesthetic challenges, particularities and concerns associated with games that are created from data. We start from the idea that the use of data games is in many ways similar to notions in art such as collage, sampling, and remixing. We draw on content from many different sources, causing creative collisions between them. This lets us apply some of the same conceptual apparatus to study data games as has been applied to these types of art.

We address the following questions:

- What does it mean for games designed from/for data to be maximalist?
- What is the tradeoff between transforming data and staying true to the source in terms of generating games?
- What are the characteristics of game content that can be generated from data?

1 Data-driven design and data games

This age of data sharing (whether sharing is free or not) has certainly been advantageous to research in computational creativity. While computational creativity does not necessarily need to emulate human creativity [156], freely available human-annotated data can be exploited as an inspiring set [172] to any creative software.

While using existing game data —often annotated with human notions of quality— has been explored in computational game creativity [119], most efforts perform minor adjustments to existing games. Game generators such as *ANGELINA* [55], *A Rogue Dream* [51], and *Game-O-Matic* [219] use data outside the game domain, enhancing their outcomes with human-provided associations (and content such as images). Even so, the core gameplay loop is simple: in *ANGELINA*, for example, the player performs the basic actions of a platformer game (e.g. jump, run); in *A Rogue Dream* the player moves along 4 directions and perhaps uses one more action. Gameplay in all these games is mechanics-heavy, relying on fast reactions to immediate threats rather than on high-level planning or cognitive ability. Many *data games* take an existing game mechanic and generate new content for that game from open data [80, 79, 39]. In some cases, such as the game *Bar Chart Ball*, a new game mechanic is added to an existing data visualization [213]. To play even simple data games, the player must have some understanding of the underlying data. Playing data games requires some mental effort, deduction or memory; not only dexterity.

While most data-driven game generation software focus on a simple and tight gameplay loop, there is considerable potential in using and re-using information outside of games to create more complex game systems and more involved experiences. We argue that data-driven game generation can allow for a new gameplay experience. Using the *Data Adventures* series of game generators, which will be extensively discussed in this dissertation, as a concrete example, we articulate the tenets of maximalism in game design inspired by the art movement of the same name. Moreover, we discuss two possible dimensions of maximalist game design, and how it can start from the raw data on one end or from the gameplay experience on the other. Finally, we envision the potential uses and issues of maximalist game design.

2 Maximalism in data-driven design

We are inspired by the notion of maximalism in the arts, rather than in the game design sphere. In music, for example, maximalism "embraces heterogeneity and allows for complex systems of juxtapositions and collisions, in which all outside influences are viewed as potential raw material" [99]. We similarly embrace the use of heterogeneous data sources as notes (i.e. the individual components) and melody (i.e. the overarching game or narrative structure) to produce a game as an orchestration of dissimilar instruments [116]. In that sense, maximalism in data-driven design is likened with mixed media in art, where more than one medium is used. De facto, the heterogeneity of the data, its sources, and the people who contribute to its creation and curation will insert juxtapositions and collisions. This may not always be desired, and several catastrophic, inconsequential or seemingly random associations should be redacted. However, the "grain" of data-driven design [104] is built on the collision and absurdity of different elements that find their way into the game.

It should be noted that maximalism in the artistic sphere refers to materials or identities of elements within an image, song, or novel. We refer to maximalist game design in that sense, focusing on how game elements originating from different data sources (or transformed in different ways) are visualized, combined and made to interact together, thus not directly opposed to minimalist game design. Nealen et al. [141]'s minimalist game design encourages removing the unnecessary parts of the design, highlighting the important bits. Sicarts's [181] approach refers to the game loop; minimalist games have a simple core game loop which is largely unchanged throughout the game. Sicart uses *Minecraft* [136] as an example where the simple core loop gather→craft→build remains relevant and unchanged (except from the specific materials worked) throughout the game.

A data-driven game, maximalist in the artistic sense, can also be minimal in the gameplay loop sense. *Data Adventures* (explained at length on Chapter 5) has a simple core gameplay loop of traveling to a new location, talking to a non-player character in that location, learning the clue for the next location. Games that we would define as maximalist on the design sense, on the other hand, have the broadest mechanics of options for solving a problem — e.g. killing a dragon with stealth, magic, followers, swords, fists, poison, etc. in *Skyrim* [24] — or

subsystems that are so elaborate or numerous that the player becomes unable to distinguish a core game loop — e.g. the diverse driving, shooting, spraying, running, etc. minigames in *Saints Row IV* [182] which are the main ways to progress in the game. While certainly data-driven design can offer the latter form of maximalism, e.g. with individual minigames where different forms or sources or data are presented and interacted with in each, not all data-driven games need to have maximalist game loops.

3 Designing Games for Maximalism

A major challenge of maximalist game design is deciding what to prioritize. One can shape data in order to fit the game, or modify the game design to better showcase the original data. One can also have data ingrained in the game mechanics, directly affecting gameplay, or show the data in a decorative manner. Maintaining a balance between data transformation to fit other data and the game itself, or staying faithful to the original data while providing an engaging experience is challenging. This section describes two dimensions of maximalist game design: *Data Transformation versus Data Fidelity* and *Functional versus Decorative*. Fig. 3.1 shows these dimensions and our interpretation of how some games fit into them. The X-axis represents Data Transformation versus Data Fidelity, while the Y-axis represents Functional versus Decorative.

3.1 Data Transformation versus Data Fidelity

The tension between data fidelity and data transformation is rooted in the priorities of a maximalist designer: the original game design or the original data. When using open data, designers may wish to adapt that data to the game, or to keep the data as it is and mold the game around it. Extensive data transformations may improve the game experience, but are also susceptible to loss of information or inaccuracies.

Transforming data gives designers more freedom and might be preferred if they have an inflexible idea, or if the data itself is malleable. On the other hand, designers may instead wish to stay faithful to the original data, molding the game to the data instead. This way, information present in the data is more likely to be clearly presented within game content.

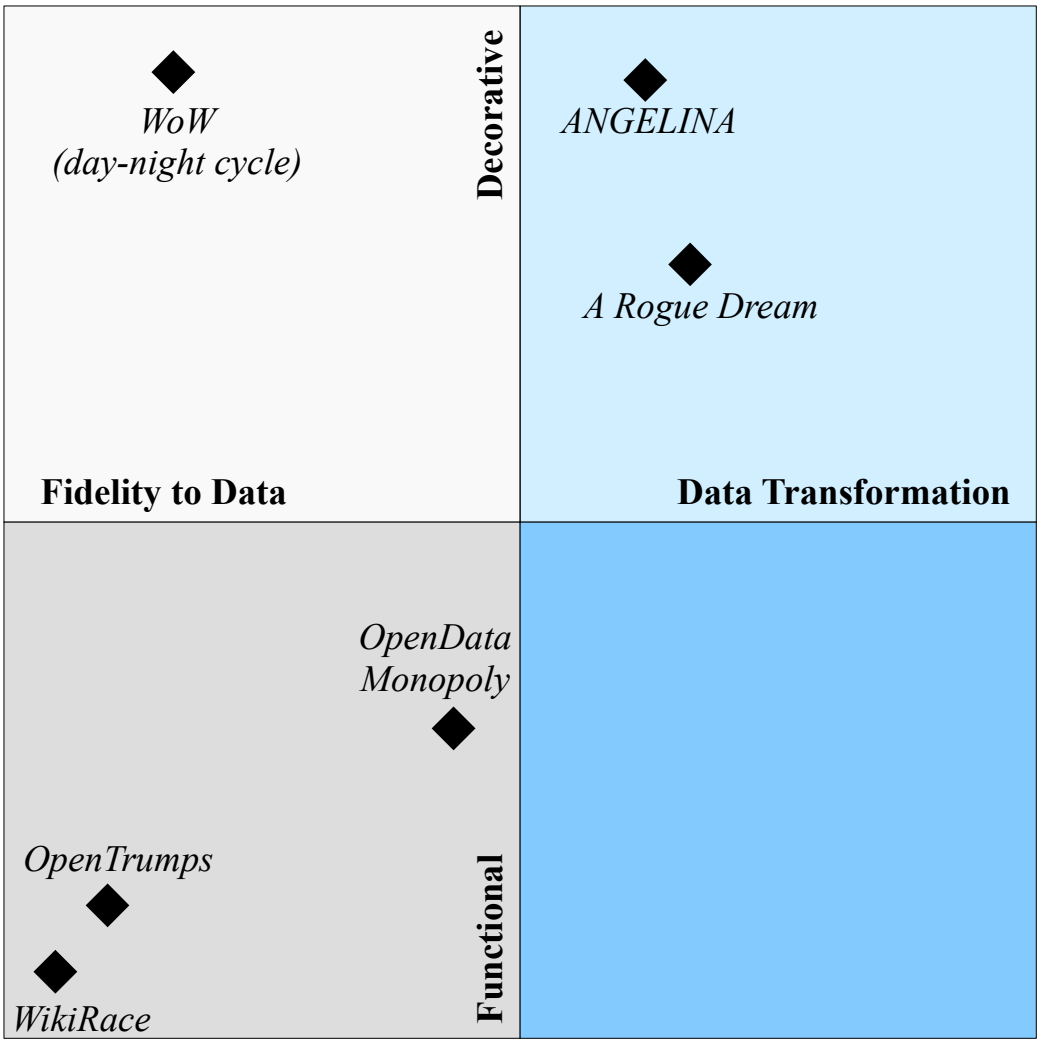


Figure 3.1: Examples of games within the two dimensions.

The rigidity of data restricts what kind of game elements can be used, or forces designers to be creative in their implementations. While less time might be spent cleaning and translating data, more time will likely be spent on raw game and mechanic design. An example can be found in *WikiRace* [1], where the game navigation is built from Wikipedia links.

[1] http://2pages.net/wikirace.php

3.2 Functional versus Decorative

Another dimension pertaining to maximalist game design is functionality versus decoration. We define data being *functional* when it has a strong impact on gameplay. If the player does not have to interact with the data, or the data does not impact gameplay in a significant way, then it is *decorative*.

In order to be functional, data can be incorporated in a variety of ways. In *Open-Trumps* [39], the raw mechanics of the game come from open data, as the cards themselves are created from it. On the other hand, any data that does not serve a functional purpose is decorative. Data can serve a decorative purpose in many ways. *A Rogue Dream* [51] uses open data to name player abilities, however the in-game effects of the abilities are not affected by their names or the underlying data, and *World of Warcraft* [27] uses real-world time to create an aesthetic day-night cycle in-game, which has no affect on actual gameplay.

4 Summary

This chapter discussed an approach to game design inspired by the notion of maximalism in the arts. It encourages the reuse and combination of heterogeneous data sources in the creative design process. Maximalist game design embraces the generation of game content using different data sources, re-mixing them in order to achieve something new.

We propose a mapping of the maximalist game design space along two dimensions, *data transformation versus data fidelity* and *functionality versus decoration*. The former focuses on the extent that the data is transformed from its original form, while the latter refers to the actual role of the data in the game.

Chapter 4

Generating balanced strategy game maps from open data

This chapter explores an approach that is highly functional and focused on data fidelity as opposed to data transformation. It describes how we used topological information to generate maps for an open version of the classic epic strategy game *Civilization* [134]. These maps reproduce parts of the real world, any part the player chooses, at any scale. This system also combines resources deposit information to place resources in the map and evolves players' starting positions.This chapter begins by giving a brief overview of the games used, then it describes how we acquired data and how the system transforms it into maps and evolves the players' initial positions. Afterwards, it highlights results it can generate and what are the its limitations.

1 Civilization and FreeCiv

Civilization [134] is an epic turn-based strategy game designed by Sid Meier originally released in 1991. In it, the player takes on the role of the leader of a civilization through 6000 years of history. The game features military conflict and can be won in various different ways, and it encourages players to explore the game world, founding and growing cities, planning productions, balancing budgets, and conducting scientific research. A game of *Civilization* is heavily influenced by the topology of the map and which other civilizations

and resources (e.g. coal, gold, etc) are available and where. The game design, which rewards exploration and only reveals certain resources once a particular level of technology has been reached, allows for the emergence of interesting conflicts and complex gameplay from resource competition. At the beginning of a new game, a complete new map is generated, and the conditions under which the player builds their civilization differs radically between playthroughs.

We chose to use *Civilization* for this project because the generation of maps from topological information is a fairly direct transformation, as the original content is much closer to the final project than if we generated, for example, mechanics out of real-world maps. Additionally, maps in *Civilization* can directly impact the gameplay, as the player's positioning and the resources available directly affect how the game is played. We did not, however, used the actual *Civilization* game. Instead, we used *FreeCiv*, an open source turn-based strategy game inspired by the *Civilization* series and comparable to *Civilization* I and II [1, 2].

2 Data Acquisition

This project uses two different categories of data to generate maps: terrain information and resource deposits' locations. The first one is acquired right before the actual map generation, using *OpenStreetMap* (OSM) [87] and *JMapViewer* to render a map. OSM is a community based world-mapping project [87]. *JMapViewer* is a Java component that can incorporate an OSM map into a Java application as an interactive map[1].

We developed a tool to assist the process of importing and managing resource deposit map images from the user's computer. It also allows for the selection of areas to generate maps using *JMapViewer*. While it would be better to fully automatize the process of importing maps and resource information, our goal was to explore how the data is imported and transformed. We were not able to find a single source of information about all (or most of) resource deposits, and map images can differ greatly in design, making it impossible for a single pattern to be used for recognition. Thus, we decided to create a mixed-initiative

[1] http://wiki.openstreetmap.org/wiki/JMapViewer

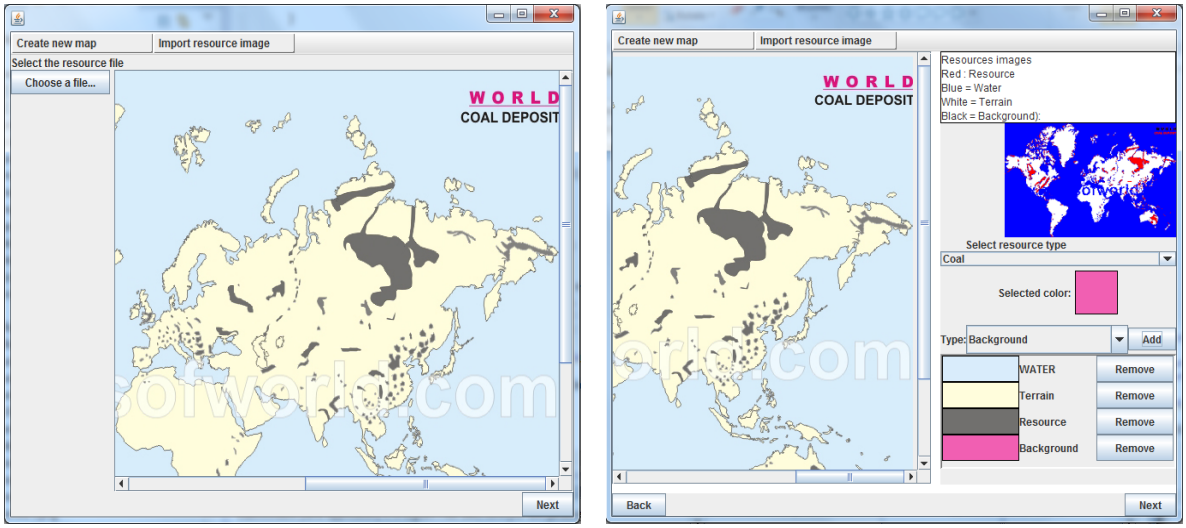


Figure 4.1: **Left:** Screen used to import resource deposit images. The user can select a file from their computer in this screen. **Right:** The second screen used when importing resource data. The user can select colors by clicking on the image on the left side of the screen. On the upper right corner, a preview of the resource image version is shown. On the bottom right, they can see each selected color and its associated resource type and can remove colors from the list.

tool.

2.1 Resource deposits locations

Information about resource locations was used during the generation process, but had to be semi-automatically obtained and processed before. This process occurred in three steps: In the first step, several images were manually collected using the Google search engine, with sentences such as “oil deposit + maps”. These images were saved and fed to the developed tool, as shown in Fig. 4.1.

In step 2, the tool displays this image on screen and allows for the selection of the resource’s type (i.e. coal, oil, gems, gold or iron), as well as which colors from the image represent what (as shown in Fig. 4.1). For each color selected, it is necessary to define it as either resource, background, terrain or water. The selected colors are used to infer those not selected using a color similarity algorithm. This goes as follows: first, a new blank image is created with the same dimensions as the original one. For each color c_i in the

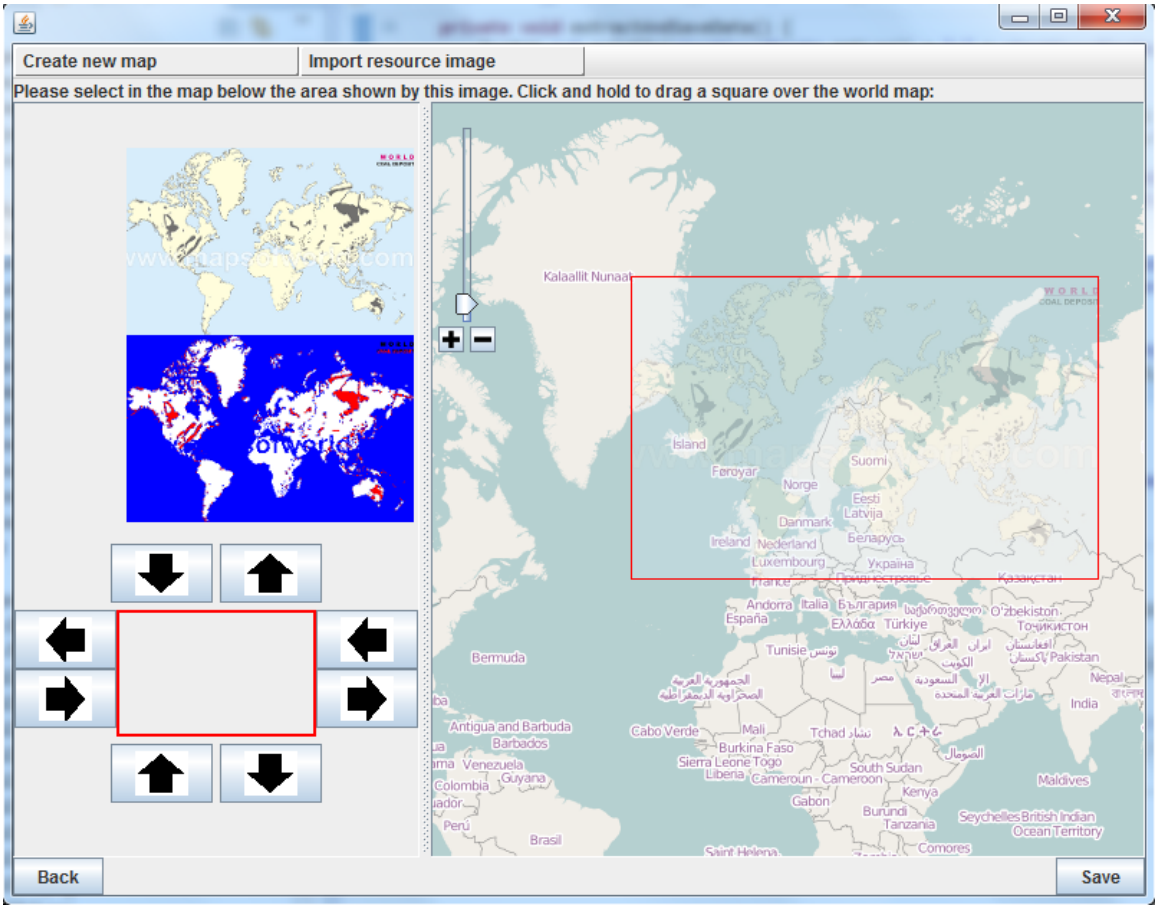


Figure 4.2: Third and final resource importing feature screen. The user can select a rectangle from the map and save the file.

original image, a distance is calculated between c_i and all selected colors. The color with the smallest distance is, then, applied to the new image. The distance is calculated using DeltaE 1994, or ΔE 1994 [122], a distance metric between two colors in the CIE Lab color space. The CIE Lab is a color representation based on an vertical L* axis ("lightness"), that ranges from 0-100, and two horizontal axis a* and b* (green and red, respectively) [17].

Finally, in step 3, the user can select a rectangle in an actual world view rendered with *JMapViewer*, indicating were the image would fit inside the world, as seen in Fig. 4.2. Additionally, they can better adjust the positions of lines using buttons on the left side of screen.

2.2 Terrain information

Terrain information is obtained immediately prior to the actual map generation. Using the interface shown in Fig. 4.3, the user can zoom in and out of the map and select a portion of it, as well as the resolution and name of the final *FreeCiv* map. The selected portion is then saved as an image to be processed during data transformation.

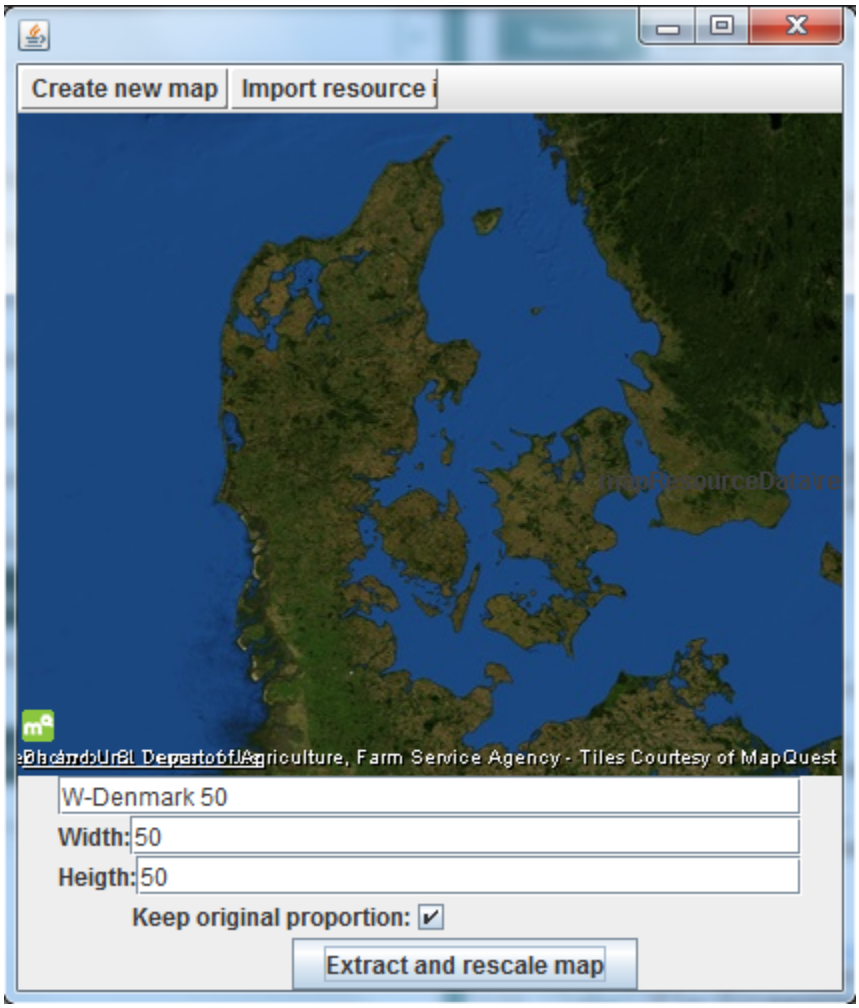


Figure 4.3: Screen of world place selection for map generation.

3 Data Transformation

The map generation process consists of four steps: map selection and terrain transformation, resources deposits intersection and creation, placement of players' initial positions, and

lastly post-processing. These are described below.

3.1 Map selection and terrain transformation

In the map selection, a world image is extracted and saved, as explained in Section 2.2. Longitude and latitude coordinates from the top left and the bottom right corners are gathered for later use. Furthermore, the user can select the final map's name and dimensions. Then the extracted original image is rescaled to the desired dimensions. An integer matrix, *mapTerrain*, with the same size is also created and initialized with zero (0) values. For each pixel in the original image, a value is attributed to the matrix's relative position, using the color of that pixel as terrain type (i.e. green pixel is a forest, blue pixel is ocean or lake, orange is desert, etc). As images are resized, some information about the original data is lost in the process. Data loss ratio depends on the desired map size and the original space dimension. If the original image is far larger than the output, parts of it are be comprised into chunks of equal size.

3.2 Resources deposits intersection and creation

Subsequently, the coordinates of the original extracted image corners are used to identify which resources can appear in that area. An array of images, resourcesImgs, is used to represent the resources. At first, all images are initialized with the same size as the map, and they contain only black pixels. Each image represents a resource type, such as coal or oil. For each resource imported, the areas of the resource image that intersect with the extracted map are copied from the first and merged to the second. Areas with resources are painted white, but water, terrain and background are ignored.

Afterwards, a character matrix, mapResources, is created with similar dimensions as mapTerrain, and initialized with blank spaces (' '). For each image in resourceImgs, and for each tile in mapResources, the system extracts the resource type and its character. If a pixel in resourceImgs indicates that there can be a resource there, the resource has a 0.35 percent change of being added to the map. Resources that the system does not have information about (i.e. the user did not import any map in the data acquisition first step,

Algorithm 1 Resource creation

```
for all Images img ∈ resourceImgs do
  for j = 0 to mapResources.height do
    for i = 0 to mapResources.width do
      rType ← resource type of img
      rChar ← character representation of rType
      if img[j][i] = WHITE and randomdouble < probRes and mapTerrain[j][i]
      is compatible with rType then
        mapResources[j][i] = rChar
      end if
    end for
  end for
end for
for all resourceType ∉ resourceImgs do
  rType ← resource type ∉ resourceImgs
  rChar ← character representation of rType
  for j = 0 to mapResources.height do
    for i = 0 to mapResources.width do
      if mapResource[j][i] is EMPTY and random double < probRes and
      mapTerrain[j][i] is compatible with rType then
        mapResources[j][i] = rChar
      end if
    end for
  end for
end for
```

Figure 4.4: Pseudo-code for resource position selection.

shown in Section 2.1) are randomly placed in the empty spaces if they are compatible with the terrain and the space is currently empty. This algorithm for resources insertion is shown in Algorithm 4.4.

3.3 Placement of players' initial positions and post-processing

The third step evolves initial positions on the generated map using an evolutionary algorithm. The algorithm and fitness functions are discussed in detail in Section 4. Finally, in the fourth and last step, other minor information is randomized, such as country and chosen leader. All data is saved in a .sav text file, which can be opened in *FreeCiv* as a save game.

4 Evolutionary Balancing

Our strategy for balancing maps focused on the initial base's position. Given a terrain map and resource map, the positions for n given players were evolved using a $50 + 50$ evolution strategy. Individuals were represented as vectors of $2n$ spaces, where n is the amount of players. This vector contained values x and y representing the coordinates of each player's base.

Firstly, an initial population of 100 individuals was created. For every different individual, players positions were chosen at random. This population was then submitted to 50 iterations of evolution. In each iteration, a new population was created. One point crossover between two parents of the first population resulted in two other individuals. This was repeated until the new population size reached a threshold. All parent were then added to this population, and cascading elitism [212] was applied using the three fitness functions described bellow, in the following order: Initially, fitness fDB_i was used to sort the population, and the lower rated half of it was eliminated. Then, remaining population was resorted using fOR_i fitness, and again half of it was eliminated. The last individuals were, once again, sorted, now using fitness $fBPR_i$, and the worse half was eliminated. The survivors were, finally, cloned repeatedly, until the population returned to its original size. The population was subsequently mutated. There would be 20% of chance of mutation for altering the vector, i.e. changing bases' x and y values. Then, the mutated population would be passed to the next iteration. In total 50 iterations passed before the algorithm terminated. The order of the fitnesses (fDB_i, fOR_i and $fBPR_i$) was chosen after testing with all possible combinations.

Fitness calculation takes into account **exploration** and **fairness**. By exploration, we define the necessity of searching new places and how far can one player go before encountering another one. It is measured by a value fDB_i, short for fitness of distance between bases, as shown in Eq. 4.1.

$$
\begin{aligned}
fDB_i &= \begin{cases} 1 - \left(\sqrt{\frac{\sum_{i=0}^{n} (dist_{ij} - mean_{dist})^2}{n}} \right), if dist_{min} > threshold \\ 0, \text{ else} \end{cases} \\
mean_{dist} &= \left(\sum \frac{dist_{ij}}{dist_{max}} \right) \div n
\end{aligned} \tag{4.1}
$$

where n is the amount of players, $dist_{ij}$ is the distance between two bases i and j, $dist_{max}$ is the maximum distance in map (i.e. its diagonal) and $dist_{min}$ is the minimum distance found. All distances are normalized based on the $dist_{max}$ and calculated using A* [173]. $mean_{dist}$ represents the average normalized distance between all bases. Thus, $fitDB$ is the standard deviation of distances between bases, multiplied by -1, plus 1. We do this inversion to maximize the value, so we can search for a maximized fitness. In truth, the lower the standard deviation, the better, since we would obtain a more equal distance among all bases.

Fairness, on the other hand, implies giving similar opportunities to every player of obtaining resources. It is measured by the values $fBPR_i$ and fOR_i (short for fitness of bases per resources and fitness of owned resources), as shown in Eq. 4.2 and 4.3, respectively.

$$
\begin{aligned}
fBPR_i &= 1 - \left(\sqrt{\frac{\sum_{i=0}^{n} (mean_i - bigMean)^2}{n}} \right) \\
bigMean &= \frac{\sum_{i=0}^{n} mean_i}{n} \\
mean_i &= \sum_{j=0}^{qR} \frac{dist_{i \to j}}{dist_{max}}
\end{aligned} \tag{4.2}
$$

where n is the amount of players, $dist_{i \leftarrow j}$ is the distance between bases i and resource j, $dist_{max}$ is the maximum distance in map, qR is the total quantity of resources. This represents the standard deviation of the average of the average distances between resources and bases.

$$fOR_i = \begin{cases} 1 - \sqrt{\frac{\sum_{i=0}^{n}\left(\left(\frac{\sum_{i=0}^{n} s_i}{qOR}\right) - s_i\right)^2}{n}} & \text{if } qOR \neq 0 \\ 0 & \text{if } qOR = 0 \end{cases} \tag{4.3}$$

where qOR is the total quantity of owned resources in map, n is the amount of players, and s_i is the amount of safe resources owned by base i. A safe resource is a resource that is closest to i than to all other bases, and that is guarded by base i itself, in the sense that any other player who tries to get to it has to pass by i first. If there is no owned resource, the fitness returned is 0.

5 Results and Discussion

This section discusses some of our results.

5.1 Balancing Results

Three different series of experiments, of 10 runs each, were performed using random maps, populated with water, grassland and resources. The first batch had maps of 100x100 dimension and 10 players, the second had the same dimensions, but 10 bases; and the last was 250x250 with 5 bases. Initial positions were evolved in 200 iterations using the method described in Section 4. Fig. 4.6 shows, for each fitness, the average between the best individuals of that iteration (in said feature) in all runs. All three features show an increase over time, but the outcome is higher when using a larger map, especially for fOR_i. $fBPR_i$ show a smaller difference between tests, probably due to being the last feature used in the cascading process.

Another experiment was done using a *NSGA-II*, a multi-objective optimization algorithm [59]. It attempts at minimizing the inverse of each fitness (fDB_i, fOR_i and $fBPR_i$) without one fitness decreasing another. It used 10 experiments with maps of 100x100 dimension and 5 bases, over 250 iterations. Results are shown in Fig. 4.5, scaled between 0 to 0.4 on the y-axis. It is possible to notice that the fitness fOR_i seems to optimize much faster than the others, and that fOR_i and $fBPR_i$ converge fairly quickly, within 20

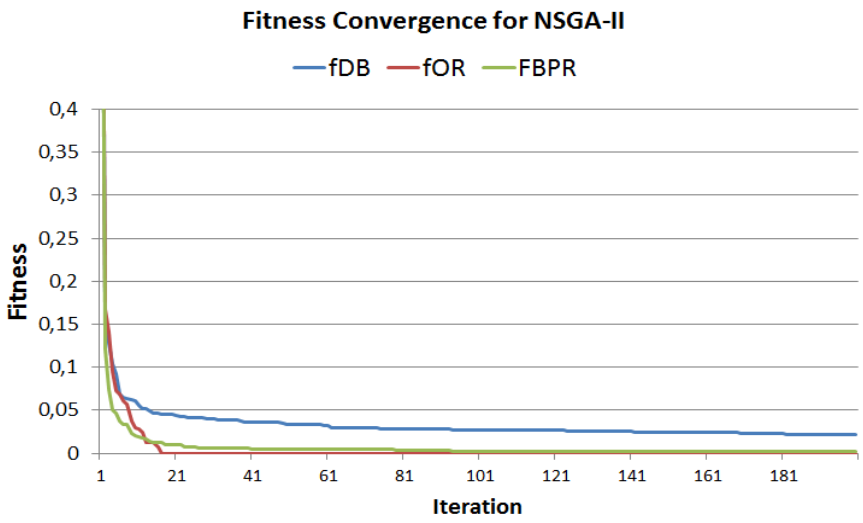


Figure 4.5: Fitness convergence of the NSGA-II algorithm.

iterations.

Figure 4.6: Average convergence of the algorithm with different map sizes and quantity of bases. **Left:** Using 100x100 map and 10 bases. **Middle:** 100x100 map and 5 bases. **Right:** 250x250 map and 5 bases.

5.2 Map creation

Some of the maps generated are shown below. Fig. 4.7 shows an in-game map of Denmark, while Fig. 4.8 shows maps from Europe, Africa, and North and South America. It is possible to notice that the generated maps retain some amount of the real-world geographical information. Fig. 4.9 compares loss of quality in a level of 25x20, 50x40 and 150x141

dimensions, showing a small amount of difference in comparison with the original, which is the exact same as the one shown in Fig. 4.3.

One problem encountered was that some maps of resource deposits that were imported had the wrong proportions or required tilting and/or warping to properly fit the OSM view. This led to some erroneous placing, like putting coal mines in places that do not have coal deposits in the real world.

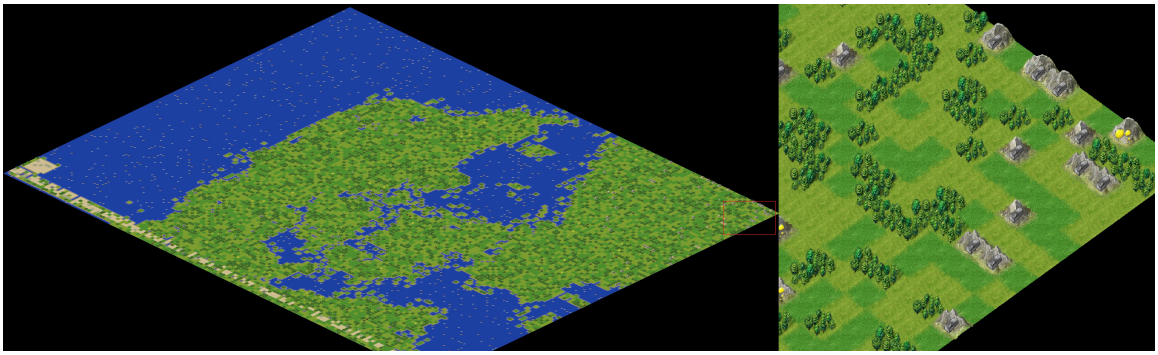


Figure 4.7: Map of Denmark, in-game, on the left. On the right, the right upper are (highlighted in red) of map is zoomed in. Note that it has isometric topology, thus is "inclined" to the right in comparison to the original map image.

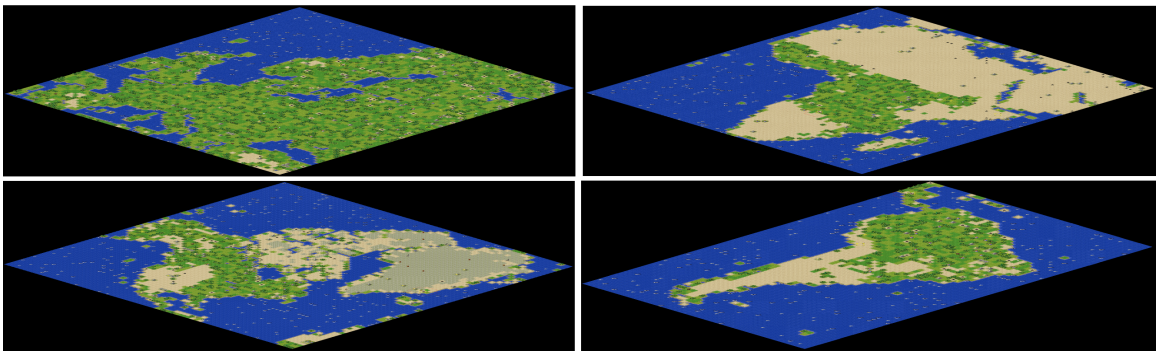


Figure 4.8: Maps generated from different inputs. **Top left:** Europe. **Top right:** Africa. **Bottom left:** North America. **Bottom Right:** South America.

6 Summary

This chapter described a method for creating complete playable maps for a clone of the popular strategy game *Civilization*, using open data about the real world. The method creates maps that present a low degree of topological transformation, thus being somewhat faithful

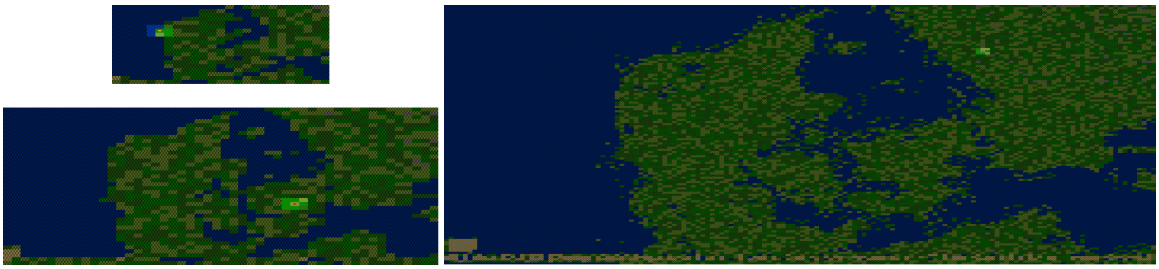


Figure 4.9: Maps generated using Denmark as input, with 25x30 (top left), 50x40 (bottom left) and 150x121 (right) dimensions.

to the original data. Resource information, on the other hand, adds some inaccuracies due to the actual data itself. Additionally, as the content generated from the data directly affects the gameplay, this transformation is more functional than decorative. The method is incorporated into a framework which lets the user select any part, of any size, of a world map and create a playable *Civilization* map out of this area.

Moreover, we believe that the passage of information can be further explored in a narrative-focused game. In this sense, what we can accomplish on this system is limited by the actual design of *FreeCiv*. As such, the next sections further explore the data-driven maximalist design space proposed in Chapter 3 using adventure games, as opposed to strategy games.

Chapter 5

Data Adventures

In this chapter, we consider the problem of generating adventure games — or perhaps more properly, adventures for adventure games. Compared to arcade and strategy games, the nature of the content that needs to be generated is somewhat different for adventure games. Many adventure games rely on places (to explore), people (to interact with), social relationships (to explore and/or exploit) and items (to move or use). These types of content are frequently both textual and graphical in nature, e.g. a person needs both an image and textual description or dialogue. Adventure games can involve less functional data transformation than strategy ones, as character images may have a more decorative role. Building a system capable of creating believable and interesting non-player characters, items and places from scratch would be a tall task; one can easily compare with the many generators that have tried this, such as the world generator in *Dwarf Fortress* [21], where the formulaic nature of names and characteristics can quickly become apparent.

By developing ways to automatically (or pseudo-automatically) generate adventure games from open data, we hope to be able to facilitate the process of designing and developing adventure games, but also to make such games more personalizable (a player might be more interested in playing a game involving characters or places they have a connection to) and enhance replay value, as the games could be made to be effectively infinite. The process of designing such games, which is user-directed, could also be seen as a novel way of playful data exploration and visualization (and could possibly provide a form of edutainment). On a more ambitious note, we are interested in exploring the aesthetic

induced by data-generated adventure games; the first examples of data adventure games shown in this chapter, coupled with [51], can be the basis for a genre of "living", contextual games which are playable e.g. only concurrently with the news or data that create them, or only by individuals with a deep knowledge of the domain (e.g. science, politics) where the data originates from. More generally, the adventure game generator which forms the end-goal of the work presented here would constitute an exemplar of computational game creativity [119], as the computer must find ways of exploring the vast volumes of data, find which data can be combined together and finally transform the game's rules and aesthetics appropriately to the data at hand: this draws direct parallels to the exploratory, combinatorial, and transformational capacities of computational creativity [29].

In the current chapter, we describe *Data Adventures*, the first in a three-game series, and its mechanism for generating a particular kind of adventure game content: non-player character groups suitable for the search for a missing person, or "manhunt". In designing this, we were inspired by the classic adventure game *"Where in the World is Carmen Sandiego?"* [192], which focuses on finding the titular game character by arresting criminals, traveling the world, talking to non-playable characters and collecting clues. The system described here is a functional "horizontal slice" of an adventure game focused on finding a missing person through the simple mechanics of traveling to places, talking to NPCs and consulting items. Every time the game is started, the player can decide on the characters used as starting and finishing points, and a complete game is generated by selecting characters and places from the real world and constructing items. All the information is drawn from *Wikipedia*, *Wikimedia Commons* and *OpenStreetMaps*.

1 Data Adventures' game design

Adventure games are a popular genre of games where players interact with virtual environments in order to solve conflicts in a given context [40]. Adventure games vary greatly, as they have been in circulation for over 30 years, yet a trait they share is a strong story-driven experience. It is thus necessary to focus on how the story is conveyed, either by dialogues, scenarios, visuals or any other form of expression. In this project, the story will be repre-

sented mostly by dialogues and description texts, in a similar fashion to classical adventure titles.

The game is defined by a pair of Wikipedia articles about two people, which become the initial and the final NPCs. Using these, it is possible to find other articles that can, through links, lead from the initial to the final article. These articles and can refer to people, locations, categories, etc, and each would act as a plot point, thus in this sense the story would be the full path connected by plot points. The player's goal is to find the location of the final NPC, at which point the game ends. To do so, the player must travel between different cities and countries in search for clues. These clues can either be a piece of dialogue or an object (e.g. a note with a list of NPCs) that direct the player towards the next clue. Since the experience is about freely exploring data, there is no losing condition.

The game draws inspiration for its visuals (as well as for other aspects) from the original *"Where in the World is Carmen Sandiego?"* [192], an adventure game released in 1985. In it, the player is a detective searching for criminals around the globe. The player uses a simple interface to travel between different cities, issue search warrants used to interrogate people and learn clues, which appear as puns in the dialogue. The gameplay consists of clicking buttons, thus making *Carmen Sandiego*'s mechanics simpler than those of other adventure games. For instance, instead of having to walk to a door to leave one place and then move to the next, the player only needs to click on the map to choose the next location. Our game is built similarly: the interface contains dialogue screens, menu options and static images. Additionally, unlike *Where in the World is Carmen Sandiego?*, *Data Adventures* has no inventory or puzzles, which are common traits of adventure games.

Our game starts at the world map screen, where the player can travel between cities. Once in a city, the player can travel to different buildings, such as hospitals, schools or houses. In a building, it is possible to view information about the NPCs and objects therein, as well as to select an NPC to talk to, which leads to a dialogue screen. A more elaborate example of the game progress and the interface is provided in the Results Section.

2 Implementation

Several steps are necessary in order to create a game from open data: the acquisition of data, the selection of appropriate data, the transformation of raw data into playable content and the presentation of content to the player. In this project, the adventure is generated via the following steps, which are described in detail below:

1. **Plot mining:** Collect data to establish a plot defined from the links between two people.

2. **Game objects generation:** Generate NPCs, objects and locations from nodes in the plot, transforming data into game objects.

3. **Postprocessing:** Generate clues and set up preconditions and effects for creating a game progression.

2.1 Plot mining

In *Data Adventures*' generator, the game's plot is constructed starting from two people who have *Wikipedia* articles. The steps connecting one person to another are obtained from *Wikipedia* via the *DBpedia* endpoint. *DBpedia* is a project which extracts information from *Wikipedia* in a structured manner[1], recording it as RDF links, as described in Section 5.1. In this context, an endpoint is a service in the *DBpedia* server that allows communication between our system and *DBpedia*'s database. Possible links between these individuals are obtained via a crawler heavily inspired by the *RelFinder* tool [91], a visualization and exploration tool for the web-semantic data.

Given two people, an *origin* O and a *goal* G, the crawler creates multiple queries and passes them through the DBpedia endpoint. The queries seek different relations between the goal and the origin, as shown in Fig. 5.1: P_i are predicates between two subjects (e.g. "reside in" or "affiliation"). The objective is to find different configurations of connections. The results are represented as directed paths, which are then evaluated; the best one is used

[1] http://dbpedia.org/

$$\mathrm{O}\ \overrightarrow{\langle P_1 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overleftarrow{\langle P_1 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overrightarrow{\langle P_1 \rangle}\ \mathrm{obj}_1\ \overrightarrow{\langle P_2 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overrightarrow{\langle P_1 \rangle}\ \mathrm{obj}_1\ \overleftarrow{\langle P_2 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overleftarrow{\langle P_1 \rangle}\ \mathrm{obj}_1\ \overrightarrow{\langle P_2 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overrightarrow{\langle P_1 \rangle}\ \mathrm{obj}_1\ \overrightarrow{\langle P_2 \rangle}\ \mathrm{obj}_2\ \overrightarrow{\langle P_3 \rangle}\ \mathrm{G}$$
$$\mathrm{O}\ \overrightarrow{\langle P_1 \rangle}\ \mathrm{obj}_1\ \overrightarrow{\langle P_2 \rangle}\ \mathrm{obj}_2\ \overrightarrow{\langle P_3 \rangle}\ \mathrm{obj}_3\ \overrightarrow{\langle P_4 \rangle}\ \mathrm{G}$$

Figure 5.1: Examples of possible relations discovered by the crawler. The difference between the predicate direction (e.g. "Albert Einstein has a notable student Leó Szilárd" versus "Leó Szilárd is a notable student of Albert Einstein") is indicated by the arrow above the predicate.

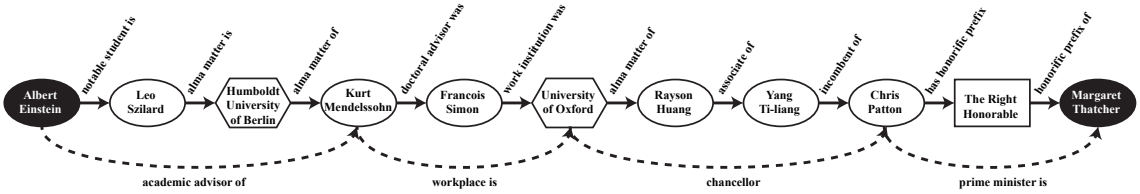


Figure 5.2: One of the possible paths between Albert Einstein and Margaret Thatcher. Dotted arrows represent major paths, which are connected indirectly with minor paths (black arrows). People are placed in a circle, locations in a hexagon, and categories in a square.

to flesh out the adventure's plot. An example path between Albert Einstein and Margaret Thatcher, found by the crawler, is shown in Fig. 5.2.

As the path is a direct representation of the story, and the story in *Data Adventures* is a single linear adventure, we wanted to prioritize larger paths, thus providing a longer experience. We used a brute-force approach: first, the system queries DBpedia for paths. It creates every possible combination of path queries, up until a certain threshold, which represents the maximum length of the longest possible path. Each query represents a relation between two or more resources (i.e. articles), as shown in Fig. 5.1. For example, for 2 given nodes O and G, one query would search for relations that fit $O\overrightarrow{\langle P_1 \rangle}G$, while another would search for $O\overleftarrow{\langle P_1 \rangle}G$. Given a path with n nodes in the path and $e = n - 1$ edges, the possible combinations of relations amount to 2^e. However, we do not want to exclude the chance of selecting a smaller, more interesting path. Therefore, the number $q(e)$ of queries for e edges is $\sum_{i=1}^{e} 2^i$.

Nodes (Edges)	2 (1)	3 (2)	4 (3)	5 (4)	6 (5)	7 (6)	8 (7)
Time	0.08	0.26	0.5	1	140	469	249
Queries	2	4	6	8	10	12	14

Table 5.1: Average amount of time, in seconds, taken to search all possible queries with n nodes and $n-1$ edges from Alan Turing to Nikola Tesla.

Due to limitations in the DBpedia endpoint, it is not practical to make very large queries. Table 5.1 shows the time required for querying all combinations of nodes for paths of size between 1 and 7 nodes between Alan Turing and Nikola Tesla (chosen arbitrarily). To avoid large queries, we separated the crawling process into two steps. Firstly, we search for a path p_{major} between O and G with up to 5 nodes. We choose this value because it does not take as long to compute but still gives us reasonable paths that can be extended in the next step. Secondly, for each pair of nodes in p_{major}, we search another path p_{minor}, size 4, which will be incorporated into the main path (see Fig. 5.3). Path sizes were chosen empirically.

2.1.1 Path evaluation

Evaluating paths discovered by the crawler is not straightforward: two desirable properties are *length* and *uniqueness*. The paths returned by the crawler are used to flesh out the game's plot, to instantiate the NPCs, locations and clues and ultimately to determine the game's progression. Favoring longer paths leads to more NPCs and locations, and makes for a longer, arguably more challenging game session.

Moreover, the links between game elements (objects, NPCs) should be somewhat obscure — to challenge the player in discovering these links — but also specific enough to avoid guesswork. Therefore, discovered paths are evaluated on their uniqueness, both for nodes and links. Uniqueness of links is evaluated based on how often they appear in all discovered paths between origin O and goal G; links that are common in most paths are less favored. As an example, in paths between Alan Turing and Nikola Tesla, many paths include links such as "influenced by" and "influenced" (scientists influencing each other), while far fewer paths include links such as "thesisYear" (scientists who handed in their PhD thesis on the same year); thus links such as "thesisYear" are more 'unique' and therefore would provide a more specific, if obscure, clue. Similarly, uniqueness of nodes is evaluated on how often

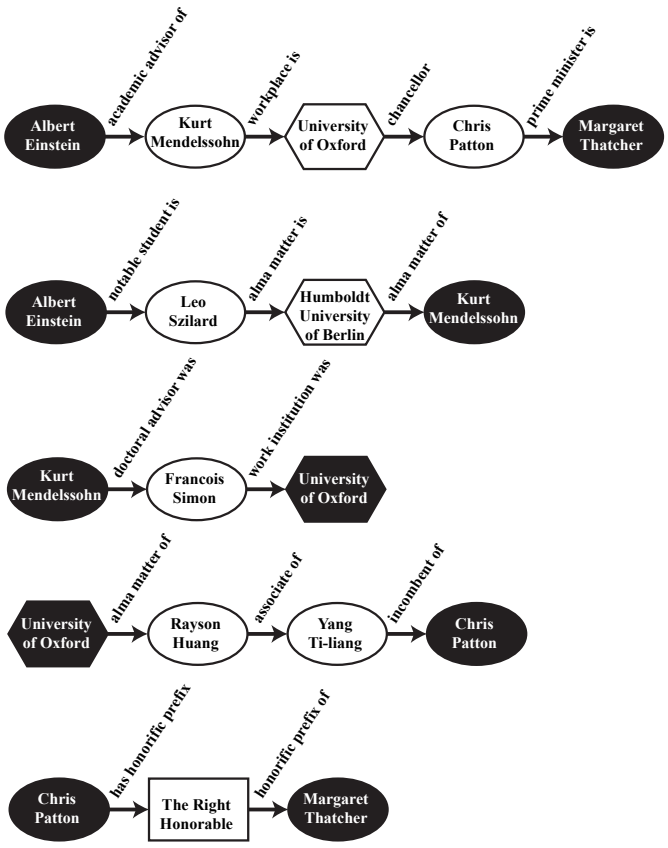


Figure 5.3: The process of expanding a major path of depth 5 (top) connecting Albert Einstein with Margaret Thatcher with minor paths. Each node in the major path becomes a start- and end-point for the crawler, which returns the 4 minor paths (at the bottom). These are combined to create the plot of Fig. 5.2.

specific nodes appear in all discovered paths; for instance, nodes such as "Austrian Empire" are less unique than "Mathematicians who committed suicide" (evidently, many individuals lived or died in the Austrian empire). Given a node i and a path p, let X^i be the amount of times that node appear in all paths, and x_p^i the amount of times that node appears within path p. The uniqueness value u of a path p is calculated as shown in Equation 5.1, where l_p is the length of p (the number of nodes in the path) and n is the total amount of nodes.

$$u(p) = \frac{1}{l_p} \sum_{i=1}^{n} \left(\frac{x_p^i}{X^i} \right) \tag{5.1}$$

Calculation of uniqueness of links is done similarly, only changing the variables from nodes to links (edges). The uniqueness of paths and the uniqueness of links are normalized

and added to the path length (normalized to the maximum possible length) in order to evaluate the path. The highest scoring path is used to flesh out the components of the game, i.e. its locations, NPCs and objects.

2.2 Game objects generation

Similar to *"Where in the World is Carmen Sandiego?"*, the game revolves around travelling the globe in order to discover clues. Therefore, both NPCs and objects in the generated adventure need to be placed in locations. The next sections describe the process behind location, NPC and item generation.

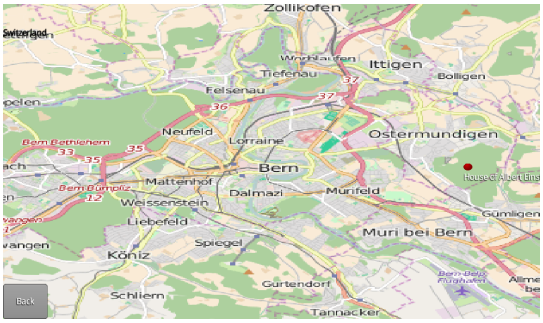
2.2.1 Location generation

Locations in the game can be cities or buildings within those cities. Locations are generated from nodes in the path discovered by the crawler. Each node may have coordinate information (i.e. latitude and longitude) or may be linked to data with coordinate information, for instance via a "placeOfBirth" link (see Fig. 5.4a). If such coordinate information is a city or state (based on the DBpedia ontology), it instantiates a city location. The city is placed on the world map via its DBpedia coordinates (see Fig. 5.4b), and a map of the city itself is created using OSM with *JMapViewer*, in a similar fashion as that described in Chapter 4. The map is centered at the DBpedia coordinates given, exported as an image and stored in the game for re-use (see Fig. 5.4c). If the node does not contain coordinate information or its coordinates are not a city, state or country, the node creates an in-game building, which is a generalized location (e.g. a plaza, a university, a hospital or a house). Buildings are placed in existing cities; NPCs and objects generated from the node in question are placed in their respective locations i.e. if an NPC has a city, they are placed in a random building within that city. Images for buildings (see Fig. 5.4d) are obtained using Spritely [2] [51], a tool that automatically searches within *Wikimedia Commons* for images, ripping and compressing them – in this case, it searches for the location's name or type (e.g. "university" or "house").

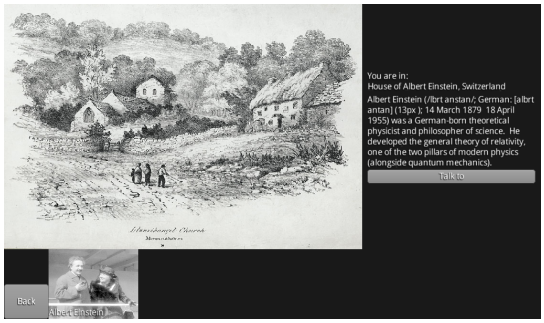
[2] http://www.gamesbyangelina.org/2012/12/spritely/

(a) Snippet of the DBpedia entry of Einstein, showing Switzerland under 'citizenship'.

(b) World screen when Einstein is added as a start NPC: Switzerland is placed at the right coordinates in the world map.



(c) City screen of Switzerland "city": the city of Bern from OpenStreetMaps is used as visual backdrop, and Einstein's house is placed as a building in it.



(d) Building screen for Einstein's house: Albert Einstein (with an appropriate portrait) can be interacted with via the icon at the bottom of the screen.

Figure 5.4: Generated locations (world locations, city locations, and building descriptions) for the Albert Einstein NPC of the adventure in Fig. 5.2.

2.2.2 NPC & item generation

The system transforms nodes from the path discovered by the crawler into NPCs or items in a similar fashion to the process for location generation. If the node is of type "Person" (based on the *DBpedia* ontology), an NPC is generated in-game and some of its stats (e.g. name, date of birth) are instantiated from its *DBpedia* entry. NPCs are placed in new buildings within cities which match their residency information, place of birth or place of death. in *Data Adventures*, an image of the real person is obtained using Spritely (see Fig. 5.4d for an image of the NPC instantiated from Albert Einstein). If no image is found, a picture of a random person is used.

Nodes which are not of type "Person", e.g. categories such as "The Right Honourable" in Fig. 5.2, are transformed into items, specifically books. Pieces of information linked to

the node (e.g. “subject of” or “abstract” in *DBpedia*) are added as readable content for the book. Generated items are placed in new buildings within existing cities, chosen randomly.

2.3 Setting up clues and creating the story flow

Once all game objects (NPCs, locations and items) have been generated, it is necessary to guarantee that they are presented in a sequential order, according to the rudimentary plot discovered via *DBpedia*. For each node in the plot’s path, a clue and a condition are added: if the node is a person, the system instantiates a template dialogue tree for that NPC, adapting it to point to the game object of the next clue. If the node is a location, a new NPC (with random characteristics and picture) is placed in that location and a template dialogue tree is added. If the clue is neither of the above, the clue is added to the text within the book item. For instance, if the book represents a category, such as “Mathematicians in the 20th century”, the book text could be something like: “Mathematicians in the 20th century: ...”, followed by a list of names found in that DBpedia entry. Finally, a condition is added so that the game object of the next node on the path can only be accessed after the current node’s clue has been seen, either in the NPC dialogue or in the book.

In addition to clues that point to (and unlock) the next node in the plot, the clue set-up can also provide false clues to confuse the player and enhance the exploration element common in adventure games. There is a 50% chance of giving a vague or false clue that can lead to a dead end. If the next node in the plot is an NPC, the current clue may point to a location that contains the right NPC but also other random ones who provide no useful information. If the next node in the plot is a building, the clue can be the name of the city that building is in, and new random buildings in that city are added with random NPCs (or no NPCs at all). However, if the next node is a city, no false clue can be given.

3 Results

To test the acquisition and transformation algorithm, we used the list of 100 Most Important People of the 20th Century by TIME Magazine [209, 208, 211, 207, 210]. Albert Einstein, Franklin Roosevelt, Aretha Franklin and Anne Frank are examples of names in this list. The

Result	Number of runs
Runs with Errors	32
Runs with no Errors	819
Playable runs	827
Type of error	**Amount of errors**
No path found	10
Search error	9
Parsing error	12
Other errors	1

Table 5.2: Results of game generation runs.

game was executed 851 times using randomly selected pairs of people in the list, with no repetition. However, the article sometimes considered multiple people as one (e.g. The Beatles or the Kennedy Family). In such cases, we use only one individual of the group (e.g. John Lennon for The Beatles). This person was chosen based on the first reference in the group's Wikipedia article (e.g. The Beatles's article). The choice of people is not as important for this test's purposes, as long as they are reasonably "famous" and thus well-referenced by Wikipedia. One entry on the list, "American GI", was removed because we could not choose a single person to represent it. Therefore, pairs were built from a list with 99 people.

Table 5.2 shows the number of playable generated games: 96% of runs generated a complete adventure game without errors. Of the failed ones, 31% were due to no paths being found for the specified distance. This does not necessarily mean that there is no connection between them, but rather that no connection within 5 nodes could be found. Most errors (37%) occurred during parsing (i.e. the transformation between data and game objects), which sometimes could be traced back to lack of information in *DBpedia*. For instance, creating an NPC depends on the data type "Person" in the *DBpedia* page; however, some pages (such as the one on Kurt Gödel) were incomplete at the time and led to parsing errors. Search errors include any other error not caused by the program itself during the search process, such as a bad gateway error caused by the DBpedia endpoint being under maintenance or offline. It should be noted that some errors were not catastrophic, leading to playable games as other queries were still able to find a plot.

Table 5.3 shows information about the objects created using data from the paths. An

	Average (sd)
Number of cities	3.82 (1.39)
Number of buildings	7.86 (2.36)
Number of NPCs	5.33 (2.32)
Number of items	4.14 (2.36)
Ratio of real NPCs (over all NPCs)	74% (20%)
Average path length	8.34 (2.45)
Minimum path length	3
Maximum path length	13

Table 5.3: Top: Average amount and standard deviation of game objects generated. Bottom: Minimum, maximum and average length of paths selected by the crawler.

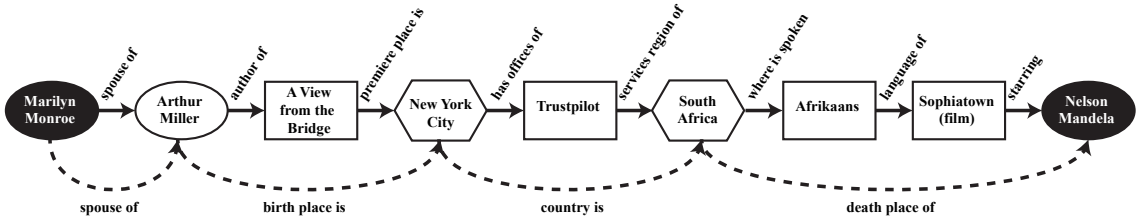


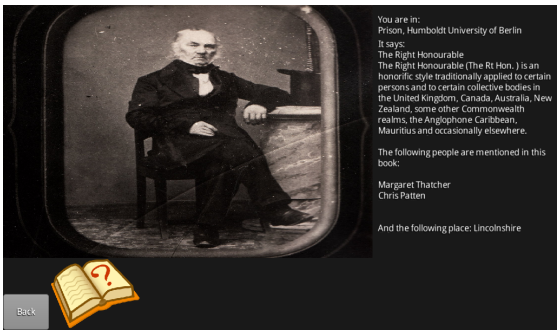
Figure 5.5: Path between Marilyn Monroe and Nelson Mandela.

interesting finding is that most NPCs are based on real people from *Wikipedia* articles, which is desirable from a design perspective as more data finds its way into the game (compared to randomly generated, and thus inherently less interesting, NPCs). Overall, the results show that, in order to discover the goal NPC, several locations must be visited and NPCs interacted with. The amount of game content generated (buildings, items, NPCs) suggests that playing through such adventure games would be an involved, prolonged process; however, user tests with human players was not performed to validate this hypothesis.

3.1 Examples of generated games

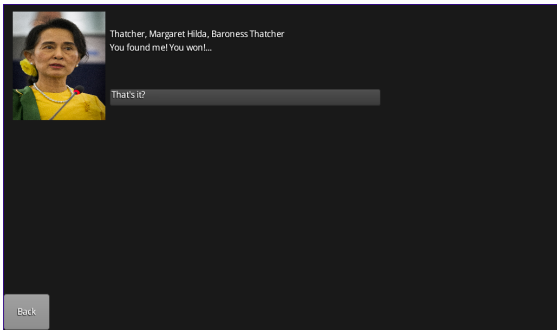
To better understand the results of the data-based adventure generation algorithm, this section provides two paths and some images from the final generated games of the previous experiment. The first path was calculated between Albert Einstein and Margaret Thatcher, shown in Fig. 5.2. The game starts with a world map screen containing one city, “Switzerland”, where the player can visit the house of Albert Einstein and talk to him (see Fig. 5.6a).

(a) Screenshot of a building (Albert Einstein's house), with an NPC (Albert Einstein). On the right, a description of Einstein and a button that leads to a dialogue screen.



(b) Screenshot of the book titled "The Right Honorable" (which covers the same category as that in Wikipedia), pointing to Lincolnshire, where players can find Thatcher.

(c) Another building (a bar), where there is a book about the Humboldt University of Berlin.



(d) The NPC dialogue screen at the end of the game, when the player finds Margaret Thatcher and wins the game. Unfortunately, the image of her NPC is clearly not Thatcher.

Figure 5.6: In-game screenshots of the adventure of Fig. 5.2 connecting Einstein with Thatcher.

Talking to Einstein leads the player to a police station in Graham, where a book about Leó Szilárd can be found. This book cites the Humboldt University of Berlin, and takes the player to a small bar in Switzerland (Fig. 5.6c), where another book about this university is, with data about Kurt Mendelssohn — who can be met in his house in the United Kingdom. This continues until the player finally meets Margaret Thatcher, in Grantham (Fig. 5.6d). This playthrough included 4 cities, 8 buildings, 4 NPCs (all based on real people) and 4 items.

Another game was generated between Marilyn Monroe and Nelson Mandela. The path generated is shown in Fig. 5.5. The path contains misinformation as it states that New York

City is part of South Africa. This can be traced back to the DBpedia resource page[3] for New York City, where several places exist in the field "dbp:country", including not only United States and South Africa, but also Israel, Italy and Egypt.

Note that *Data Adventures*' generator generates games that have only rather shallow dead ends; individual NPCs that are not tied to the story exist, but there are for example no subplots or hidden items or places. Essentially, the generated games can be played and won by simply visiting all places and talking to all NPCs. However, it is not possible to find the target NPC without having talked to the other NPCs first, meaning that any generated game has a given minimum play length.

4 Discussion

The processes described in this chapter choose data from vast repositories of open data (primarily *Wikimedia* projects) and transform them into game elements which can be interacted with and, after being experienced sequentially, lead to a goal state. The data which form paths between starting and goal NPC are dependent on the quality and breadth of information available in *Wikimedia*, and can lead to associations which are obscure (e.g. Margaret Thatcher and Chris Patton sharing the title "The Right Honorable" in Fig. 5.2) or unintuitive (e.g. New York City in Fig. 5.5 referring to a South African city rather than the obvious U.S.A. city of the same name). Moreover, when generating pictures for NPCs or places based on their names, the crawler may not find appropriate images (or in the case of Margaret Thatcher in Fig. 5.6d, use a picture of Aung San Suu Kyi). This leads to a varying degree of absurdity in the results, and one could argue that the generative system has several issues that need to be fixed; the next Section highlights necessary changes to improve the output of the generator.

However, the very decision of using open data which are freely edited and updated daily by millions of users (in the case of *Wikimedia*) or susceptible to bias from latest user searches and newly appeared content (in the case of Google searches) comes with a degree of inherent absurdity in the outcomes. While the unintended or catastrophic absurd outcomes should

[3] http://dbpedia.org/page/New_York_City

be hedged against through careful engineering, it is both unavoidable and desirable that a degree of absurdity in the resulting adventures remains. As with several games which rely on transformations of rhetoric [219] or concepts [51] into game mechanics or objects, absurdity is a desirable side-effect which evidences the data used to instantiate them — the player *should* be feeling "like [they are] playing a videogame against The Internet" [51]. Absurdity on the modern worldwide web is a reality (and largely an appealing quality for message boards such as `reddit`) and therefore can not be (and should not be) absent from the data adventures which transform them into playable experiences. As noted in [51], information collected from open data "is often tainted by popular belief, misconception, stereotype and prejudice, as opposed to purely factual information". This is a strength (as they provide contextual associations based on current events or current interests of the userbase) and a weakness (as they can be volatile or offensive or absurd); regardless of whether it is more a strength or a weakness, this fact can not be cut off from data games.

5 Summary

This chapter presented the first installment of the *Data Adventures* series, a game and system that uses open data to generate content for an adventure game. It describes the developed crawler used to discover a path that connects two people, using *Wikipedia* and *DBpedia* as source of information. This path represents the story of the adventure, undergoing a parsing and transformation process to generate the game's locations, NPCs and items.

The gameplay of the generated adventures in *Data Adventures* is still too restricted. This is mostly due to the lack of diversity and interactivity of game objects. The system uses a larger variety of data sources than the one described in Chapter 4, but most of the content generated in *Data Adventures* affects less the game mechanics than the previous system, thus being more decorative in nature e.g. images and descriptions for books are created, but do not change the underlying form of how the game is played. On the other hand, the original data undergoes more transformation in this system: text-based information is transformed into characters, dialogue and descriptions, and both text- and image-based data are used to create locations and map images.

Additionally, the game still lacks a cohesive story. Although there is a series of relations between NPCs, places and items, this relation is not yet conveyed to the player in a clear manner. Furthermore, it is never explained why the player is searching for the goal NPC — with no motive, the player is simply running around searching for someone for no reason. The next game generator in the series tries to overcome this limitation, adding a layer of narrative to the game.

Chapter 6

WikiMystery

Games that cast the player in the role of a detective, where the gameplay and main challenge revolve around solving a crime or mystery, have been popular for many decades. Some games, such as *Where in the World is Carmen Sandiego?* [192], task the player with finding a fugitive criminal. Other games, such as *Indiana Jones and the Fate of Atlantis* [124] or the *Tomb Raider* [69] series, see the player embark on an adventure to solve ancient mysteries in the face of opposition from shadowy goons. It is common for these games to feature frequent in-game travel to exotic locales around the world to interact with colorful people and to gather clues, solve puzzles and overcome resistance.

This chapter presents *WikiMystery*, the second game in the *Data Adventures* series, a framework for generating complete, playable point-and-click adventure games with minimal human input: in this case, the name of a person who has a *Wikipedia* page. Unlike its predecessor, *WikiMystery* offers a much more engaging, coherent and complete experience with a clear goal to arrest the culprit of a murder. This is facilitated by story branching towards several suspects, enhanced ludic elements as game objects that unlock certain locations, and enriched dialog elements that allow Non-Player Characters (NPCs) to share facts both about the mystery and about themselves (based on open data).

1 Overview of the game and the generator

WikiMystery is a data-driven procedurally generated point-and-click adventure game. In the game, the player assumes the role of a detective trying to solve a murder case. The victim is the central point of the story, and suspects are based on people related —somehow— to her. We use *Wikipedia* to identify possible suspects, out of which five are selected. The game plot is tree-structured: the victim is the root, each suspect is a leaf, and the path between them is a representation of the hyperlinks between the victim's and each suspect's *Wikipedia* articles.

The gameplay is divided into two parts: (1) gathering clues and finding suspects, and (2) providing evidence that all but one suspect is innocent to arrest the culprit. Initially, the only location available is the victim's house, where the player can talk to people related to the victim. The player also becomes aware of who the five suspects for the murder are. As they interact with people inside the house, new locations, objects and NPCs become available. As the player explores and interacts with the world, they collect information about suspects' characteristics (e.g. year of death, occupation etc.). Every suspect except the culprit has a value for one characteristic (e.g. "died in 1980" for suspect 1 or "birth place in Cleveland" for suspect 2), which acts as an evidence of innocence (identified in the dialog with a note "... couldn't have done it."). The player receives no such information about the culprit; the culprit also does not share the same value in a characteristic which is evidence of innocence for other suspects (i.e. the culprit did *not* die in 1980 and was *not* born in Cleveland). The game ends when the player issues an arrest warrant, identifying the culprit and specifying the values acting as evidence of innocence of the other suspects. If the player correctly finds the culprit and provides the correct evidence for the other suspects, the game is won. If the player does not specify the right culprit, or if the evidence for the innocence of any other suspect is incorrect, then the game is lost.

The game generation involves several steps, as shown in Fig. 6.1, the first of which is selecting the victim. From the victim, the system uses *DBpedia* (as did *Data Adventures*, for different purposes) to find a set of five suspects via artificial evolution presented in Section 2.1 and to generate paths between each suspect and the victim via a constructive algorithm

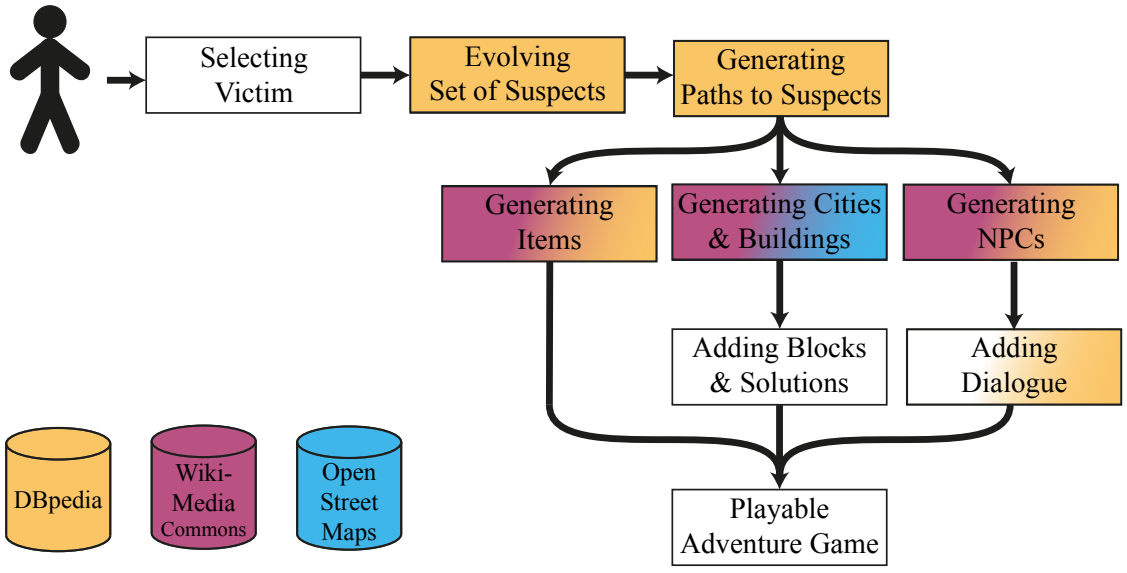


Figure 6.1: Flowchart of *WikiMystery* and its open data sources.

presented in Section 2.2. Once all paths are generated, the system creates locations, items and NPCs through constructive processes covered in Section 3. Finally, it generates puzzles for accessing locations and dialog options for learning about clues or general information from NPCs; we discuss the latter in Section 4.

2 Crawling *DBpedia*

WikiMystery creates a plot from a series of hyperlinks in *Wikipedia*, generated using several consecutive queries to *DBpedia*. Once a victim has been introduced, the system tries to find suspects and pinpoint a culprit among them. Then, it searches for paths between the victim and the suspects.

2.1 Finding suspects and a culprit

Selection of a set of suspects involves identifying who is related to the victim, and out of those, which subset is the most interesting. Given a *Wikipedia* article about a person, the system queries *DBpedia* to find anyone who has something in common with the victim. It can be as common as living in the same place, or as specific as being in the same band. This list is our *pool of suspects*. For each one, the system queries again *DBpedia* to find

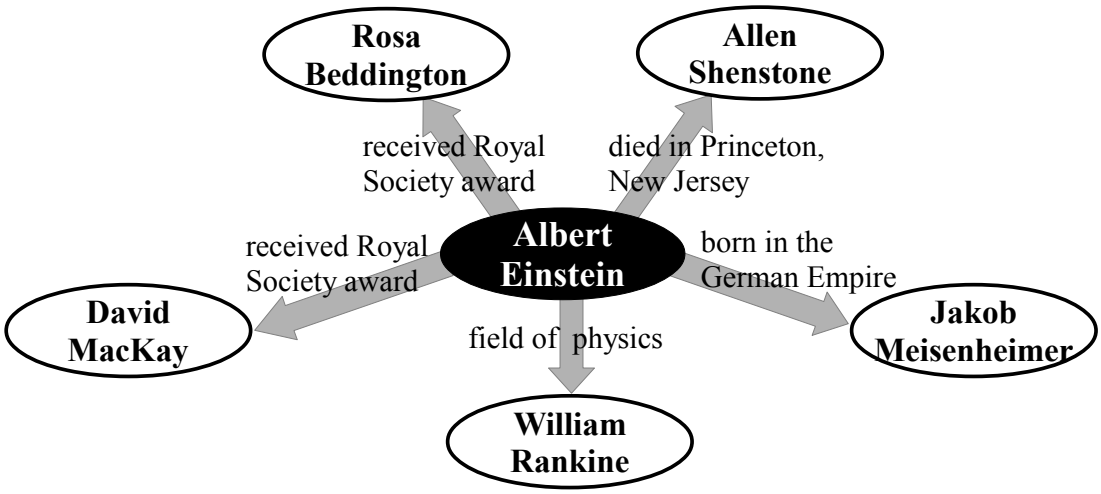


Figure 6.2: Selecting suspects from *DBpedia* to use in the game and finding relations between victim and suspects. Initially, the system has a single node: the victim (black node). Suspects related to the victim are selected with a genetic algorithm (white nodes), and paths between the victim and suspects are created from *DBpedia* (see Fig. 6.3). All suspects share direct connections with the victim, shown on the arrows.

everything known about them. At this point, it has a list of suspects, each containing a list of characteristics. Each characteristic can have multiple values, for example, a suspect could be "Albert Einstein", who would have the characteristic "Field" with values "Physics" and "Philosophy". Fig. 6.2 shows a simplified selection of suspects. From a victim (black node), the system finds suspects (white nodes), who are related somehow to the victim (arrows). The system must also find a set of characteristics that can single out the culprit among suspects. A characteristic and a value of that characteristic, together, form an *evidence of innocence*, which is used to identify the culprit (by clearing all innocents) and issue an arrest warrant.

The list of possible subjects, characteristics and values can be very large, at times. Characteristics may have multiple values available, of which the system will only use one per characteristic, and suspects can have multiple characteristics. For example, as of 2017, there were 15,300,451 distinct people related to Albert Einstein in some way in one or two steps, constituting his possible suspect pool. Each one had, on average, at least five characteristics, which may or may not have had multiple values. Selecting five suspects, four characteristics and their values is therefore challenging. To select which subset of this list is interesting, we turn to an $\mu + \lambda$ evolutionary algorithm [230]. Our goal was to have a finite

set of suspects n (typically, 5) and a finite set of characteristics $n-1$, such that we can pair each characteristic to a person. The leftover person is the culprit, and the characteristics are evidences of innocence for the $n-1$ suspects. This allows the player to eliminate innocent suspects by finding the clue paired to them. The remaining suspect who does not have the same value with any evidence of innocence must be the killer.

Our fitness function evaluated solvability and diversity. *Solvability* favors complete solutions, where the player can identify the culprit by excluding the $n-1$ characteristics they know the killer does not have. The system uses Depth-First Search: for every chromosome, it marks one of the suspects in the chromosome as the killer. The search states are characteristics in the chromosome, and they are paired with one suspect once they are visited. Valid states have three properties: (a) the killer has at least one value for the characteristic; (b) one or more suspects have value(s) for the characteristic; (c) at least one suspect has one value different from the killer. The algorithm tries to pair each suspect to one characteristic if the suspect has at least one value that is different from that of the killer (for that specific characteristic). If it cannot match a pair, it backtracks and tries a different suspect. No characteristic can be paired with more than one suspect, and vice-versa. The optimal solution is a leaf where all characteristics are paired successfully to suspects; the fitness is the depth of the leaf.

Diversity evaluates how different the characteristics and values are. The game only outputs one value per characteristic/suspect pair, so it is necessary to optimize which value to use. For example, in a game with 5 suspects and 4 characteristics, a solution with 20 values, one per pair, is better than one where one suspect has no value for any characteristic but the one matched to him. Additionally, a solution where all suspects have the same job and live in the same city is less diverse than a solution where they all have different jobs and live in different cities. Even though they still use the same characteristics ("job" and "residence"), the second one has more diversity of values. The actual fitness value is given by:

$$f_D = \sum_{i=0}^{P} \left(Q_i \times \left(- \sum_{j=0}^{V_i} p_{ij} \times log_2(p_{ij}) \right) \right) \tag{6.1}$$

where P is the number of characteristics; V_i is the number of values for characteristic i; Q_i is the number of people with characteristic i. We multiply with Q_i to reward more suspects sharing characteristic i. p_{ij} is calculated as the number of people that have value j in characteristic i divided by Q_i.

The system uses cascading elitism [212] over a population of 100 individuals for 500 generations, with a mutation chance of 20%. Cascading elitism uses both fitness functions: it sorts the population using the solvability fitness, removes the worse 50% individuals, and then sorts the remaining using the diversity fitness. The highest 25% of the population is duplicated and mutated until the new population is filled. It is far more important that games are solvable for playability's sake (rather than diverse); solvability is applied first during cascading elitism so that it introduces a stronger genetic bias.

2.2 Finding paths to suspects

Once the system has a victim, suspects and clues as evidence of innocence, it weaves them into a plot by searching *DBpedia* for a path of hyperlinks between the victim and each suspect. As mentioned in Chapter 5, a path consists of nodes (*Wikipedia* articles) and edges (links between them). The set of all paths can be seen as a tree if we merge the initial node in the paths (i.e. the victim). Therefore the root of the tree is the crime scene, and each branch leads to a possible suspect. This tree represents most plot points the player is able to unravel in the game in order to move the plot forward, such as locations, NPCs and clues. The only clues not present in this tree are those of evidence of innocence.

For each suspect, the system queries *DBpedia* multiple times, searching all possible paths between the victim and said suspect. It rates each path based on how diverse it is, i.e. the type of articles and links in the path. For example, a path that only has articles about locations is less diverse than one with an even number of articles about people and locations. This process can be computationally expensive, since it is necessary to create one query

per node in the path, and per direction of the edge. In practice, using paths longer than 5 nodes has proven to be too time consuming (see Section 2.1 for further details). To bypass this, we divided the search into two steps. The first finds a path of length no longer than 5 nodes, as described above, which we called the *major path*. Then, for each consecutive pair of nodes in the major path, the system searches for a *minor path* between those nodes. The minor path replaces the edge between the two nodes in the major path. Fig. 6.3 shows an example path between the victim (Albert Einstein) and a suspect (William J. M. Rankine), identifying major and minor paths.

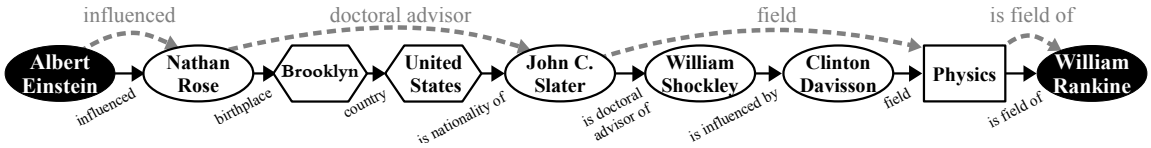


Figure 6.3: The major and minor paths between Albert Einstein and William J. M. Rankine. Major paths have dotted arrows, minor paths have black arrows. Locations are represented as hexagons, NPCs as circles and items (books or photographs) as squares.

Similar to *Data Adventures*, *WikiMystery* measures path quality based on its *length* and *uniqueness*. Longer paths are preferred because they extend the game: each node will be transformed into a city, NPC or item in the story. Uniqueness is calculated as the entropy of each node/edge in the path, compared to all nodes and edges found in all paths in that particular search. Thus a path where the type of edge is not found in other possible paths is better than typical edges (e.g. for scientists it is typical to have edges of type “influenced by” or “influenced”).

3 Enriching the data

The system transforms the set of paths (tree) obtained from *Wikipedia* into gameplay objects that the player can interact with. Each node in the tree becomes a location, an item or a NPC. To do so, it creates all necessary game objects, then generates dialogs and links between them, verifies if all objects appear in the correct order, and add puzzles.

Nodes in the tree can be roughly categorized into places (e.g. “London” or “Canada”), people (e.g. “Albert Einstein”) and everything else (e.g. “Mathematicians of the 20th

Century"). The system begins by transforming the nodes into the simplest objects possible: locations, NPCs and items. For each node based on an article about a place, it generates a city (if the place contains a geographic coordinate) or a building. In the game's logic, the world contains cities, and buildings are places inside cities. Buildings can also contain items and characters. If the system generates a building, it tries to place it into its respective city. If it cannot find any city related to that building, it will randomly pick a place from *Wikipedia* and generate a city for it, placing the building in it.

After buildings and cities are created, the system takes all nodes based on real people and generates one NPC for each. The NPC gets the original person's name and a small description. Any node that is not a person or a location is transformed into an item: either a book, a list, a letter or a photograph. Depending on the type of item, different text templates are generated to explain it.

It is not possible to transform the tree's root into an NPC, because he/she/they is supposed to be murdered. *WikiMystery* attempts to solve this by adding people related to the victim instead. For each suspect, it searches for a person directly connected to the victim, and transforms they into an NPC. If it cannot find enough people, it generates "random" NPCs, whose sole purpose is to give a clue about the following node.

Once all objects needed for the plot have been generated, it is necessary to create a logical sequence of steps from the victim to each suspect. The system traverses each branch in the tree, and adds clues and conditions from one node to the next. If the current node is a location, an NPC or item is generated and placed in it. If it is a person, dialog is created directing the player to the next node. We discuss dialog generation in more detail in Section 4. Otherwise, the clue is added to the item's text description. Additionally, at random times the game may generate a "fake" NPC, the sole purpose of which is to provide a red herring. It is given a random name, no description, and dialog that is less than helpful. A condition manager guarantees that non-root game objects are only available after they have been triggered by another object.

Finally, the system adds "puzzles". One of the most well-known puzzles in adventure games is the "lock-and-key" i.e. a location that is inaccessible unless the player uses some specific item to unlock it. *WikiMystery* generates this kind of puzzles by creating items that

are able to unlock buildings, such as flashlights for dark places and crowbars for chained gates. Puzzle objects are placed using a variation of the Breadth-First Search algorithm. First, nodes in the tree are separated by their depth, so depth of 0 will have only the location of the root NPCs, depth of 1 would contain all locations available after talking to the root NPCs, and so on. We simulate a playthrough to perform said separation. It also maintains an array with all possible keys (e.g. “keys”,“crowbars”, etc) and an initially empty stack of locks. At every depth, it randomly chooses whether to put a key in a building of that depth. If it does so, it adds the respective lock to the stack of locks. Additionally, it may randomly pop a lock from the stack and add it to another building. For example, at depth 0 it may chose to put the “flashlight” key in the root building. It will automatically add the lock “darkness” to the stack. Because the victim’s house is the only building at depth 0, and it has already been chosen, the algorithm goes to depth 1. It randomly chooses to put no key and no lock, so it skips straight to depth 2, where it finds a Church and a House. It randomly decides to not put any keys, but decides to pop the lock from the stack (i.e. the “darkness” lock) and adds it to the Church. Adding the key before the lock guarantees that the puzzle will be solvable.

4 Dialog generation

The game’s dialog has two goals: to advance the game by giving hints and evidence needed to win, and to provide a sense of depth and immersion, which can be hard to capture in a data game. Each NPC has their own dialog tree, i.e. the lines of dialog that both the player and the NPC use when interacting, along with the dialog options for the player. The root of this tree is a simple “Hello”, and the choices that follow are called dialog branches. There are two types of branches: the *main branch* containing information necessary to complete the game, and the *side branch* (of which there are several subtypes), which contains information that is not necessary to complete the game but increases immersion.

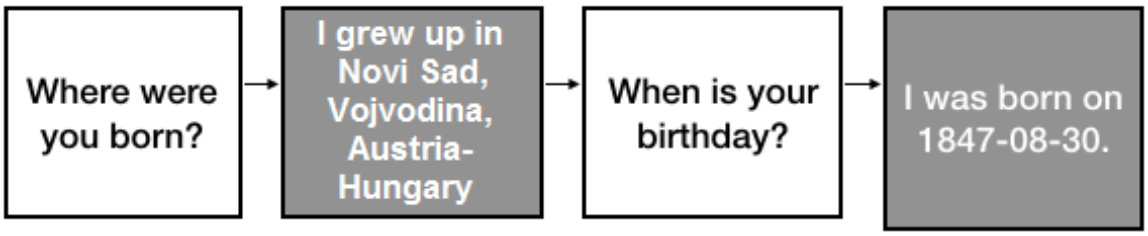


Figure 6.4: An example of a dialog side branch. The player is speaking to Hermann Einstein in the Albert Einstein game.

4.1 Main branch

The main dialog branch contains hints that will allow the player to advance in the game. For every NPC, the generator parses data and stores information from this person's *DBpedia* page: anything from places, persons, items, and concepts associated with the person, as well as personal information like birthplace and birthday, is stored. The dialog generator then takes sentence templates and replaces placeholder text. For example, Rosa Beddington's person object might have "Jamaica" stored within it as an associated place. When a player talks to another NPC who is associated with Rosa Beddington, they might have a dialog node telling the player where they think Rosa Beddington is.

> I saw `PERSON` in `PLACE`. You should probably look there.

will now become:

> I saw Rosa Beddington in Jamaica. You should probably look there.

After packaging a sentence of dialog into a dialog node, the generator adds this node as a child of the dialog root. Dialog choices are hidden by default unless its parent has been visited. Thus, the player must select the root "Hello" option before any branches are revealed to them.

4.2 Side branches

Beyond the main dialog branch, the generator also selects randomly from a set of "side branches", which have no effect on the overall story. These branches provide extra information about the NPC that the player is speaking to. There are future plans to use these for

an educational purpose, where players can learn more about characters' backgrounds by talking to them. Currently, there are 3 possible side branches: birth-dates and birth-places, current residence, and lifetime achievement. When data is originally parsed from *DBpedia*, birth, current residency and overview information are stored. When creating a side branch, this data replaces the placeholder words in templates (as in the main branch); see Fig. 6.4 for an example side branch. After the main branch is created, the generator randomly selects up to two topics for which to generate side branches, or none. If generated, the system shows an option on the dialog screen that leads the user to these branches.

5 Example playthrough

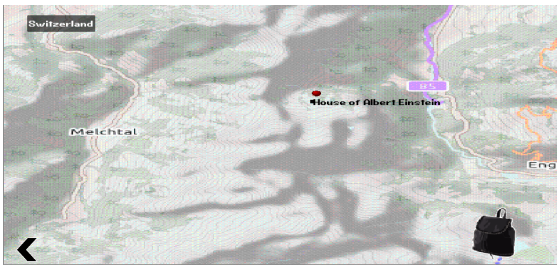
As an indicative playthrough, we describe the first few minutes of *WikiMystery* gameplay; this game uses as input the text "Albert Einstein", identified by the TIME Magazine [209, 208, 211, 207, 210] as the "Person of the [20th] Century". Once the game launches, the user can load any of the 100 most influential people of the 20th century, which were pre-generated for the purposes of the analysis of Section 6.

The game starts at the world map (see Fig. 6.5a), where only one point can be visited: Switzerland, chosen because it is the birthplace of Albert Einstein. Clicking on that point of interest, the user moves to a map of a location in Switzerland collected from OpenStreetMap[1], where a single location titled "House of Albert Einstein" (see Fig. 6.5b) can be visited. The player also has access to a backpack in this screen (bottom right of Fig. 6.5b), which is currently empty but can store items that can be used to access locked locations. When the player clicks on the house of Albert Einstein, they move to the building screen which shows a background of a single-story house, coupled with informative text about Switzerland in the bottom area (see Fig. 6.5c) and six different game icons that can be interacted with to the right. The first five icons are NPCs, while the last icon displays a crowbar which can be stored in the inventory by clicking on the hand button on the crowbar icon.

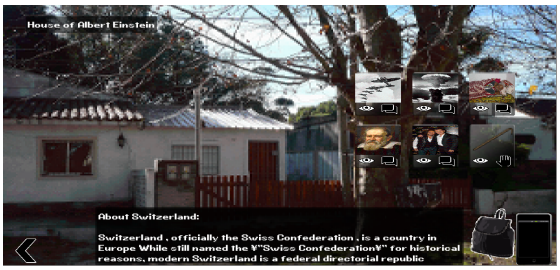
As noted above, in the House of Albert Einstein there are five NPCs which the player

[1] Apparently what is shown in Fig. 6.5b is an open area near the Melchtal Valley, as the *DBpedia* entry places the coordinates of the country of Switzerland at its center.

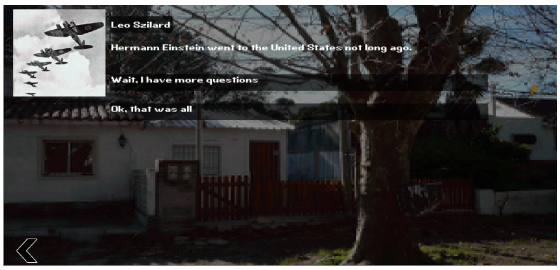
(a) Initial World Map



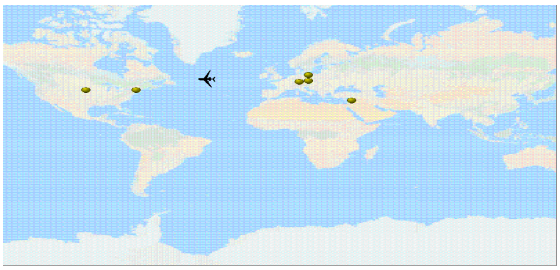
(b) Initial city map of Switzerland



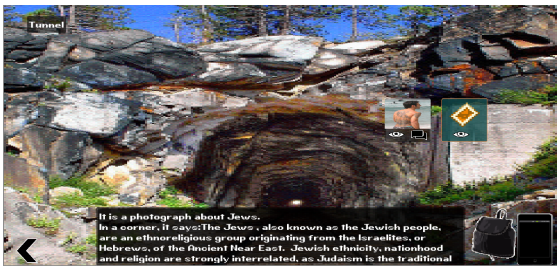
(c) House of Albert Einstein



(d) Clue given during dialog



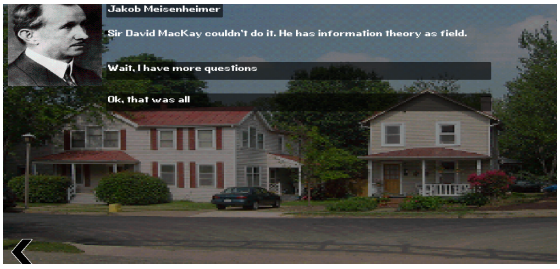
(e) Travel between cities



(f) Icon of the Photograph item

(g) Zoomed-in Photograph item

(h) Dialog with a Suspect



(i) Finding an Evidence of Innocence

(j) Issuing a warrant

Figure 6.5: In-game screenshots of the mystery around the murder of Albert Einstein.

has the option of observing (eye button under each icon in Fig. 6.5c) or talking to (dialog button under each icon in Fig. 6.5c). These NPCs are Leo Szilard, David Joseph Bohm, Jean Gebser, Riazuddin, and James; the last NPC was randomly generated and given a random name, while the remaining NPCs are physicists except Gebser who is a philosopher. Clicking on the eye button gives information about each of these NPCs (for the random NPC James the text says "There is no information available for this character"). Note that the images chosen for all NPCs are not images of people in the case of Szilard and Bohm (instead the images are related to the atomic bomb), and not images of the correct people in the case of Gebser and Riazuddin.

Clicking on the dialog button of any NPC icon moves the user to the dialog overlay (see Fig. 6.5d), where the NPC's name is followed by their response text, followed in turn by a set of dialog options. The general dialog sequence for any NPC in the house of Albert Einstein revolves around first asking for help, then asking their name, and then about any information they might have and where the player should look for it. The responses of each NPC depend on which path they are on, and towards which suspect they will guide the player. Indicatively, if the player initiates dialog with Leo Szilard then the NPC will respond to the player's question "Please state your name." with "Maybe... You can call me Leo Szilard. I was one influence."; in this case, Leo Szilard was an influence to Albert Einstein, which (subtly) explains why this character is in this game. When the player asks "Is there something you think I should know?", Leo Szilard responds "He talked to Hermann Einstein a lot."; the player can then ask "Where is Hermann Einstein?" to which Leo Szilard responds "Hermann Einstein went to the United States not long ago." (see Fig. 6.5d). This immediately adds "the United States" as a location in the world map, puts the house of Hermann Einstein on its city map, and an NPC named Hermann Einstein within it.

After the player has talked to all five NPCs in the House of Albert Einstein, there are five more locations in the world map that they can visit by clicking on them, at which point a small plane will be shown traveling from the player's current location to the selected one (see Fig. 6.5e). These locations on the world map are 'the United States'[2] (containing the house of Hermann Einstein), 'Princeton, New Jersey' (containing a Tunnel building), 'Israel'

[2]The marker based on *DBpedia* information is again placed at the center of the U.S.A.

(containing the house of Nathan Rosen), 'Wrttemberg' (containing a Stadium building) and 'Swiss Federal Institute of Technology in Zurich' (containing a building of the same name, and placed in Zurich on the map). Similarly to the house of Albert Einstein, there is one or more NPCs or clues in each building listed above. For instance, in the Tunnel building of Princeton, New Jersey there is a random NPC named Vlad and a photograph icon (left-most in Fig. 6.5f). Clicking on the photograph shows an image of Jewish people (see Fig. 6.5g), and its description says:

> It is a photograph about Jews.
>
> In a corner, it says: The Jews, also known as the Jewish people, are an ethnoreligious group originating from the Israelites, or Hebrews, of the Ancient Near East. Jewish ethnicity, nationhood and religion are strongly interrelated, as Judaism is the traditional faith of the Jewish nation, while its observance varies from strict observance to complete nonobservance.
>
> There are names written behind:
>
> Canada
>
> Israel

This information is based on the abstract of the *Wikipedia* article[3] regarding the 'category: Jews' (as stored in *DBpedia*) which is used to link different NPCs in the mystery together. These NPCs are Nathan Rosen in Israel and a random NPC (also named Vlad) in the University building in a world map location named 'Canada'; Vlad reveals that Allen Goodrich Shenstone is located in his house in Princeton, New Jersey.

After an extensive investigation taking the player to many different cities around the globe, and slowly revealing more and more buildings, NPCs and clues in previously visited cities, the player finds the five suspects. In this mystery the suspects are Sir David MacKay (whose dialog and pleas for innocence are shown in Fig. 6.5h), Allen Shenstone, William John Macquorn Rankine, Jakob Meisenheimer, Rosa Beddington. Each of these five names are also provided — after questioning — by one of the five NPCs in the player's starting location: the house of Albert Einstein. Of those suspect NPCs, Rosa Beddington (linked

[3] `https://en.wikipedia.org/wiki/Jews`, accessed 13 March 2017.

to Einstein as a fellow scientist and having been awarded by the Royal Society) is the culprit. Other suspects such as Sir David MacKay can be absolved by finding an evidence of innocence, in this case provided by chemist Jacob Meisenheimer (see Fig. 6.5i). Once the player is confident they have collected enough evidence, they can click on the cellphone (bottom left corner of Fig. 6.5d) to choose the guilty person as in Fig. 6.5j. The player chooses the guilty person and once they do so, the remaining suspects are placed in another window (bottom half of Fig. 6.5j); the player must then specify one characteristic and the correct value for each person which make them incapable of having committed the murder. The characteristics and values that absolve all suspects except Rosa Beddington are included in Table 6.1. If the player selects the culprit and chooses values for the remaining suspects, they can click on the "arrest" button (bottom right of Fig. 6.5j) at which point the game ends with a message of success or failure.

6 Evaluation

While the playthrough of Section 5 provides a glimpse of what it means to play a generated murder mystery, this section evaluates the content generated from a broader set of murdered *Wikipedia* persons. The goal is two-fold: estimate the number of interactions afforded in each game (e.g. dialogs with NPCs, visits to cities, item pickups), and assess the sensitivity of the system to different inputs (i.e. *Wikipedia* persons). For the former, several metrics regarding instances of specific elements (cities, NPCs, dialog lines) per generated game are listed. For the latter, we describe which *Wikipedia* persons were murdered in games with the highest and lowest values in these metrics. While we did not perform an user playtest of *WikiMystery*'s generated games to assess e.g. how intuitive the connections between NPCs are, the provided evaluation is vital in understanding how complex the generated games are and which of the generated gameplay elements contribute most to this complexity. This evaluation is thus a first step prior to a playtest, to assess for instance the minimum number of player clicks (via the tree size metric combined with the dialoge nodes metric) for a game to be completed. Such metrics can then be compared with actual metrics derived during playtests, but can also inform changes to the generative algorithms before such playtests can

Table 6.1: Solution from games generated with the top 3 most influential people: Albert Einstein, Franklin D. Roosevelt and Mahatma Gandhi. Each innocent suspect is paired to one characteristic (blue italics) that differentiates them from the killer (last in each list, with an asterisk). Empty values appear in-game as "Unknown".

Albert Einstein

Suspects	Death Place	Field	Subject	AlmaMater	Direct Connection
Jakob Meisenheimer	*Nazi Germany*		1934 deaths	Ludwig Maximilian University of Munich	Born in the German Empire
Sir David MacKay		*Information theory*	Living people	California Institute of Technology	Received the Royal Society award
William J. M. Rankine	Glasgow	Physics	*Thermodynamicists*	University of Edinburgh	Physicists
Allen G. Shenstone	United States	Physics	Fellows of the Royal Society	*Princeton University*	Died in Princeton, New Jersey
Rosa Beddington*	Great Tew	Developmental biology	20th-century women scientists	Brasenose College, Oxford	Received the Royal Society award

Franklin D. Roosevelt

Suspects	Term end	Party	Office	AlmaMater	Direct Connection
Kevin Cahill	*1994-12-31*	Democratic Party (US) from the 103rd District	Member of the New York Assembly at New Paltz	State University of New York	Part of Democratic Party
Gwendolyn Garcia	2013-06-30	*One Cebu*	Governor of Cebu	University of the Philippines Diliman	Politicians
Johnny Ellis	1993-01-11	Democratic Party (US)	*Majority Leader of the Alaska Senate*	Claremont McKenna College	Part of Democratic Party, politicians
Jane Griffiths	2005-04-11	Labour Party (UK)	Member of parliament	*Durham University*	Politicians
Daniel Poulter*	2015-05-12		Member of parliament	University of Bristol	Politicians

Mahatma Gandhi

Suspects	Death year	Birth place	Subject	Occupation	Direct Connection
Tex Avery	*1980*	Taylor, Texas	Articles containing video clips	Animator, cartoonist, voice actor, director	Appear in the Wikipedia category of articles containing video clips
Stanley Rosen	2014	*Cleveland*	Jewish American writers		20th century philosophers
Volker Zotz		Landau	*20th-century philosophers*	Writer	20th century philosophers
Jhunnilal Verma	1980	Damoh, India	People from Damoh	*Lawyer*	Indian lawyers
Eddie Lyons*	1926	Beardstown, Illinois, USA	Articles containing video clips	Actor, director, screenwriter, producer	Appear in the Wikipedia category of articles containing video clips

take place.

To assess a broad range of games, based on persons with a strong presence in *Wikipedia*, we used the list of the TIME magazine's 100 most influential people of the 20th century by TIME Magazine [209, 208, 211, 207, 210] as input, in the same way we did in Chapter 5. Each person in the list became the victim in a procedurally generated game, some after preprocessing, excluding two: "American G.I." and "Unknown Rebel". The system was not able to generate games with them, as the first represents a whole category (we could not choose a single person that represented this category), and the latter represents an unknown person who does not contain the tag "Person" in his *DBpedia* page. Additionally, the system cannot process groups of people, so inputs such as "The Kennedy Family" had to be transformed into a single individual. Entries about groups were transformed into one of the most known people of the group. For example, "The Beatles" became "John Lennon" and "The Kennedy Political Family" became "John F. Kennedy".

The system generated a total of 98 games, one per input. Table 6.2 shows the quantitative results.

6.1 Game Content

Based on Table 6.2, the average tree size of the generated games is 60.3 nodes. The game with the smallest tree size had "Robert H. Goddard" as input and 20 nodes. "Marlon Brando", "Martin Luther King, Jr.", "Richard Rodgers" and "Willis Carrier" tied for the most nodes in the tree, with 65. On average, the length of paths between victim and each suspect was 12 nodes. Six games had the lowest path length with 5 nodes, while 39 had the highest path length with 13 nodes.

Each game had on average around 18 cities and 46 buildings, with approximately 2.9 buildings per city. The most common cities amongst all games was "The United States", appearing in 73 out of the 98 games, followed by "New York City" (43) and "District of Columbia" (37). North American locations dominated the top 10 most common cities, with 8 locations. The remaining two were "London" and "Germany". Note that while the game only represents locations as cities and buildings, the in-game city category may include

Table 6.2: Average metrics of all generated adventure games for the 98 most influential people.

Location metrics	
Cities	18.07
Buildings	46.37
Average buildings per city	2.88
Item & Puzzle metrics	
All Items	23.91
Books	9.45
Photographs (torn or not)	7.24
Torn Photographs	2.55
Key items	2.71
Locked buildings	2.67
NPC metrics	
All NPCs	46.53
NPCs based on real people	24.03
Average ratio of real NPCs over all NPCs	52%
Average NPCs per building	1.02
Dialog metrics	
All dialog nodes	208.33
Average dialog nodes per NPC	4.45
All side-branches	35.47
Achievement side-branches	9.05
Residence side-branches	8.56
Birth side-branches	17.86
Complexity	
Average length of paths	12.07
Tree size	60.34

countries (e.g. The United States), states and actual cities.

On average, 24 items were generated per game, mostly books (9.45). Games with most and fewest books were created from, respectively, “Le Corbusier” (20 books) and “Theodore Roosevelt” (1). Key items and locked buildings tend to appear together, with an average of 2.71 key items and 2.67 buildings. Every game had at least one key and one locked building, while at most there were three keys and three locked buildings in a single game. The number of keys was always equal or higher to that of locked buildings, ensuring solvability.

An average of 46 NPCs were created per game, and on average 24 were based on real people (ratio of 52%). While this ratio is not optimal, we believe it can be improved in future versions by being more lenient in the NPC generation: now, we only look at people

with one-degree distance from the article that originated the node. If there is no person, we could expand the search to 2 or 3 degrees distances, which we believe can improve this ratio. We believe that increasing the percentage of NPCs based on real people over "random" NPCs would provide more interesting, full-fledged characters and interactions. The ratio of NPCs based on real people ranged from 28% to 76% of all NPCs. The game with most real NPCs was generated from "Lech Wałesa" with 41 real NPCs, and the one with the least from "Walter Reuther" with 9.

Based on Table 6.2, there are 208.33 dialog nodes on average in a *WikiMystery* game, distributed across all NPCs in the game. Results show an average of 4.45 dialog nodes per person. Every person has a main branch in their dialog tree, so the number of main branches is equal to the number of NPCs. In addition to those, there are on average 35.5 side-branches in a game. Of those, an average of 9 side-branches refer to the NPC's personal achievements, 8.6 side-branches concern the NPC's current residence, and 17.9 side-branches are associated with the person's birth. There are nearly twice as many side-branches on birth than the other two types, since the generator creates two branches (birth date and birth place) when selecting a side-branch on birth.

6.2 Suspects, direct connections and evidence

Each generated game must have a set of suspects, evidence of innocence and direct connections between suspect and victim (i.e. the reason for selecting those suspects). Table 6.1 shows the set of suspects, evidence of innocence and direct connection between victim and suspect of the three most influential people on TIME's list: Albert Einstein, Franklin D. Roosevelt and Mahatma Gandhi. Values in italics are used as evidence for innocent suspects, allowing the player to differentiate between them and the culprit. Notice that when the culprit has no value (e.g. in the game generated from Franklin Roosevelt, Daniel Poulter has no value for the "Party" characteristic), any value would fit to differentiate between any suspect, but the game will only check this characteristic for the specific paired suspect (in the game generated from Franklin Roosevelt, Gwendolyn Garcia is paired up with the "Party" characteristic). Additionally, if two innocent suspects share the same value for a

characteristic, it is used as evidence for one of them, as long as it is different from the culprit. That merely means that it is evidence that only one suspect is innocent, but does not absolve the other suspect. An example is shown in Mahatma Gandhi's game (see Table 6.1) where Tex Avery and Jhunnilal Verma both died in 1980, but this is only evidence of innocence for Avery (note that Eddie Lyons, the culprit, died in 1926). In some cases, the actual value appeared as a consequence of *Wikipedia*'s own organization. In Mahatma Gandhi's game, the primary reason for selecting Tex Avery and Eddie Lyons as suspects is them belonging (like Gandhi) to the category "Articles containing video clips", indicating that they both appear in the *Wikipedia* list of articles containing video clips. The secondary reason was that this set of these five suspect allowed for a solvable and somewhat diverse game, according to the GA.

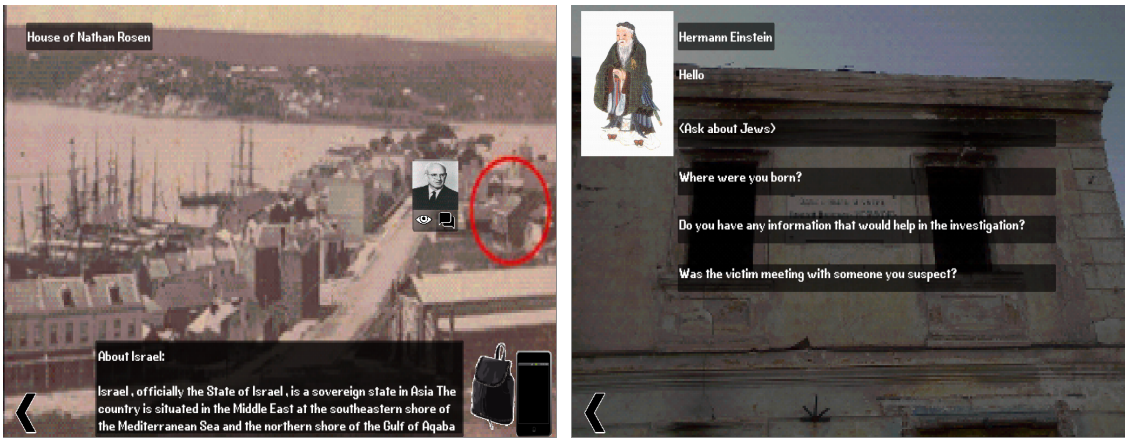
Direct connections are relations between the victim and each suspect. Since they depend on hyperlinks from the victim's *Wikipedia* page, they can only be as varied as the article itself. Therefore, games usually have an emergent underlying theme. In the game generated for Albert Einstein, both he, Allen Shenston and William J. M. Rankine were in the field of Physics, and he and Shenston both also died in Princeton, New Jersey; Einstein, David Mackay and Rosa Beddington received the Royal Society award, and both Einstein and Jakob Meisenheimer were born in the German Empire. In Franklin D. Roosevelt's game, Roosevelt and all suspects except Kevin Cahill are politicians[4]. Roosevelt, Cahill and Johnny Ellis were all part of the U.S.A. Democratic Party. Finally, in Mahatma Gandhi's game, both he and Jhunnilal Verma were Indian lawyers; Gandhi, Volker Zotz and Stanley Rosen were 20th century philosophers; and he, Eddie Lyons and Tex Avery all appear in the *Wikipedia* category of articles containing video clips. The last connection is arguably poorer than others, demonstrating how some of the source data can be difficult to tailor to our needs; this is discussed extensively in Section 7.

[4]Cahill actually *is* a politician, he is not tagged as one in *DBpedia*.

7 Discussion

The sample playthrough of Section 5 and the numerical evaluations of Section 6 provide a high-level overview of the types of games generated by *WikiMystery*. Contrary to our early attempts at adventure generation, which created one path between two people (Chapter 5), a murder mystery is far less linear and includes more dialog and gameplay options. The fact that paths can be traversed non-sequentially (inevitably so, as it is difficult to keep track of which NPC or object forms a path towards which suspect) increases the exploration and branching factor in terms of decision-making on the part of the player. This in turn leads to more interesting gameplay as it gives a greater sense of player agency. The gameplay has been improved with more branching, better visual presentation of results, more interesting dialog options, and a concrete winning condition.

Based on the metrics of Table 6.2, *WikiMystery* has strengthened its link to real-world data accessed via open data repositories, with each game containing a multitude of cities placed in their real-world locations and with a city map showing their street view (based on *OpenStreetMap*). Additionally, because it encompasses this data into a fake narrative (i.e. a murder), it is less faithful to the original data than *Data Adventures*. On the other hand, unlike *Data Adventures*, the data used to instantiate game objects in *WikiMystery* directly affects the game mechanics, in special in the case of the arrest mandate portion of the game. As it is, the data is more functional than its predecessor, but less so than the FreeCiv map generator presented in Chapter 4. The ratio of random NPCs to "real" NPCs, which are based on *Wikipedia* articles, is also kept in balance, while the introduction of photograph in-game objects increases the modes through which open data can be experienced (i.e. through images rather than text information found in book in-game objects). Most importantly, the improved NPC dialog allows not only for a more engaging and intuitive way to solve the mystery but also allows for yet another way to present open data, as a player can choose what questions to ask of the NPC (e.g. regarding their life achievements) rather than being presented the data as a large chunk of text when observing the NPC, for example.



(a) Darling house of early colonial Australia highlighted... and placed in Israel.

(b) The image of Confucius chosen for Hermann Einstein, and the 'ask about Jews' dialog are... unfortunate.

Figure 6.6: Absurd and potentially offensive combinations of data can occur with *WikiMystery*.

Although there have been substantial improvements in the presentation of content from *Data Adventures*, the very nature of generating games from open data hinges on the uncontrollable nature of such data. This allows for nigh-infinite expressivity, as any person with a *Wikipedia* presence can potentially star in a generated game (where he is murdered), but this lack of control can lead to unexpected, unintended or even unwanted outcomes. On the one hand, the ongoing efforts of the *Data Adventures* line of research focus on controlling this vast repository of data and transforming it into intuitive and playable objects; for instance, attempting to find unique connections between people rather than trivial ones such as "they are both human". It is, however, impossible to ever fully control or constrain the experience, as doing so would obfuscate its origins from a living, vast knowledge base rooted deeply in the real world. It is that very absurdity that makes the outcomes appealing in their own way; as the user of another data-based game titled *A Rogue Dream* states, it feels "like playing a videogame against The Internet" [51], and at least in the case of *WikiMystery* that is intentional.

This absurdity, however, causes some hilarious, and sometimes appalling, outcomes. It has been noted in the playthrough of Section 5 that most NPCs' images were not correct, which is either due to the lack of appropriate images for those people in WikiMedia Com-

mons, or flaws in the image parsers currently at hand. In such cases, a random search for an image of a man (for male NPCs) and a woman (for female NPCs) is used instead. For buildings, moreover, the image search is based on the name of the building without context of its geographical location. This can lead to results such as that of Fig. 6.6a, where not only is the building's background an old photograph with an actual highlighted building with a red circle, but on closer inspection the chosen building (result of a search for "house of Nathan Rose") is the Darling House, which holds historical significance for early colonial Australia, but in the game is used as a domicile for Nathan Rosen in Israel. Our choice of using only freely available sources, such as *Wikimedia Commons*, complicates the retrieval of specific images. A source such as Google Images could improve results, but contradicts the scope of freely available solutions.

Additionally, more problematic are instances where an unforeseen combination of content and their transformation can lead to insensitive or offensive results. As an example, Fig. 6.6b shows the dialog with Hermann Einstein as part of the playthrough of Section 5 where the player is seeking the culprit of Albert Einstein's murder. The image, unsurprisingly, is not that of Hermann Einstein; instead, the random search for an image of a man serendipitously ended up being a drawing of Confucius. On the other hand, the dialog has chosen to highlight that the connection between this person and the next along the path is the 'category: Jews'; this category was also cued by the photograph of Fig. 6.5g as discussed extensively in Section 5. In this case, the player interacts with Hermann Einstein with the dialog line "⟨Ask about Jews⟩". It is certainly true that the actual story of Albert Einstein was deeply affected by him being Jewish and the events of World War II, so the category and the path found is accurate (perhaps desirable), however the random choice of this dialog line [5] and the random assignment of an image of Confucius for avatar are a very unfortunate, insensitive and likely offensive combination. It is difficult to envision how such instances could be avoided, as it was largely an issue with simple transformations of data and their combination going awry. While it is not the case here, one should also not underestimate that the nature of open online data "is often tainted by popular belief, misconception, stereotype and prejudice, as opposed to purely factual information" [51],

[5] Consider how inoffensive a similar line saying "⟨Ask about Physics⟩" would be.

and thus such unfortunate instances may actually occur due to prejudice in the source data before they are even transformed.

8 Summary

This chapter presented *WikiMystery*, the second installment of the *Data Adventures* series, and detailed its complex generation pipeline, from the name of a person with a *Wikipedia* article to a full interactive murder mystery game. *WikiMystery*'s generator was the first to create fully playable adventure games with minimal human authorship and curation. Open data is used in a multitude of ways in order to find NPC suspects for an in-game murder of a specified person, to find paths linking these NPCs, to place them in locations around the globe and to provide a way for the player to absolve innocents and deduce the culprit. Moreover, open data is used to create the 'levels' (i.e. cities and buildings) in which NPCs are found, to create in-game objects with photographs and books that act as clues, and to enhance dialog options of NPCs beyond the merely functional needs of completing the game.

Chapter 7

DATA Agent

The iterations of the *Data Adventures* series described in the previous chapters produced games that successfully integrated data from various sources, but were somewhat lacking in playability. This was deemed to be partly due to the user interface, but also due to the core game design patterns that the generator used. In order to convincingly demonstrate the potential for data-driven adventure game generation, the generated games need to be enjoyable and intuitive, necessitating a redesign of both the game interface and the game generator. The following game in the series, *DATA Agent*, attempted at improving on these facets while further exploring the design space of data-driven games. *DATA Agent* is system that combines a murder mystery game generator, which generates games from open data, and a game, in which the player must identify a time-traveling doppelganger who has committed a murder and is now masquerading as a historical person. In the game, the player must travel between places and interact with historical people (NPCs), in order to collect information that will allow them to identify the culprit among a lineup of suspects.

This paper presents the outcome of that process, *DATA Agent*. *DATA Agent* has a redesigned backstory, trying to create coherent scenarios while acknowledging the frequently absurd results of building on *Wikipedia* data; a new story generation mechanism to work with the new story pattern; and a redesigned user interface. The games produced by the *DATA Agent* generator, while having in common with previous Data Adventures games that they are adventure games generated from open data, play very differently. Below, we detail the game design and generator design of *DATA Agent*.

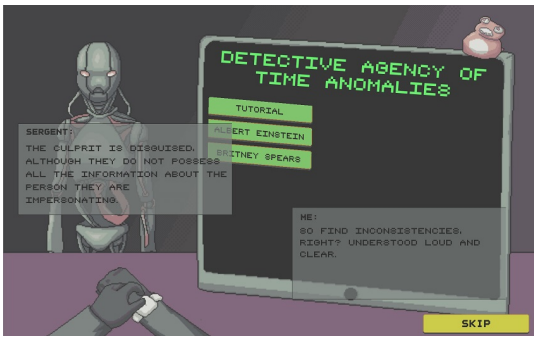
1 Game design

The design of *DATA Agent* is divided into a client and a generator system. The client (i.e. the actual game) was built using the Unity game engine (Unity Technologies, 2005), while the generator (i.e. the generator of adventures) was built in Java. The generator creates adventures using an evolved version of past *Data Adventures* generators. These changes are discussed at length in the following two subsections. The following sections describe the narrative and visual design of *DATA Agent*, as well as a writeup of how the game is played.

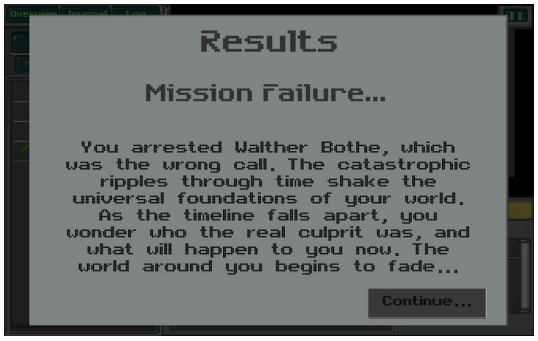
1.1 Backstory

DATA Agent introduces a level of narrative to explain the context and links between entities in the game. This serves both to explain certain expected inconsistencies (e.g. people from different time periods existing in the same game) and to clarify that both the murderer and the culprit do not refer to real events or people, respectively. The inconsistencies arise from the messiness of open data, as mentioned in Section 7. The player takes the role of a detective working for the Detective Agency of Time Anomalies (DATA). In the game's universe, criminals have the ability to go back in time and murder famous people, altering the time line and creating inconsistencies. To prevent this, DATA sends agents to the past to catch the killer and prevent the murder. Since the very act of going back in time messes up time lines, DATA has incomplete information about the past, and can only tell the player who the suspects are. It is the job of the player to learn facts about the suspects from other NPCs and objects, then catch the culprit, who is attempting to impersonate one of the suspects. Thankfully, the culprit has incomplete information about the person they are impersonating, and therefore will lie to the player about who they are. If the player has all the correct information about that individual, they can catch the culprit red-handed and save the day. All of this is clarified in the introductory cut-scene (which can be skipped), during the introductory dialogue before the player starts the game, and at the end of the game: see Fig. 7.1 for clarifications on how this exposition is displayed.

(a) Snapshot of the intro video



(b) Start Menu



(c) Game Lost Screen

Figure 7.1: An extensive introductory cutscene (Fig. 7.1a) frames the time anomaly backstory and DATA agency. The dialog on the start menu screen (Fig. 7.1b) provides a short reminder on the context and winning conditions of the game. The winning or losing screens (Fig. 7.1c), at the end of the game, also tie in with the time anomaly backstory.

1.2 User interface

The Unity client, which allows players to interact with the game, was designed to allow for ease-of-use, so that players have as much information gained throughout the game available to them at any point. Fig. 7.2 shows several screenshots of the game. The game interface is split into three different displays: the game display (top right) for most game interactions, the description panel (bottom right) for showing information relevant to the current interaction, and the side-panel (left) which has several tabs displaying all information gathered so far.

- The *Game Display* shows the player's current location and represents the current activity, e.g. talking to a person (see Fig. 7.2a), looking at a city map (see Fig. 7.2b), or reading a book (see Fig. 7.2c).

- The *Description Panel* contains a description of the last clicked on person, place, or

object, either accessed from the Game Display or from a tab of the side-panel. If the player clicked on a place, they can travel to this place by clicking the "Travel To" button (see Fig. 7.2b). Places also display a list of known individuals the player found at this location. If the player clicked on a suspect from the Overview Panel, a list of known facts is displayed instead (see Fig. 7.2f).

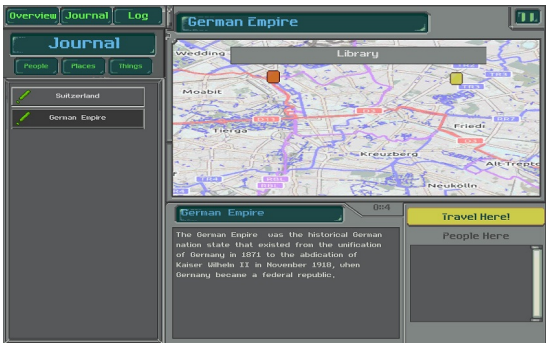
- The *Journal Tab* marks all people (see Fig. 7.2a), places (see Fig. 7.2b), and objects (see Fig. 7.2c) that the player has uncovered, distinguishing those that have not been fully explored.

- The *Activity Tab* displays in-game actions the player has taken, including traveling to locations, talking to people, and inspecting items for clues. This allows the player to remember if they have already performed some actions and/or exhausted the possible actions in a location. The side-panel of Fig. 7.2d shows the activity tab.

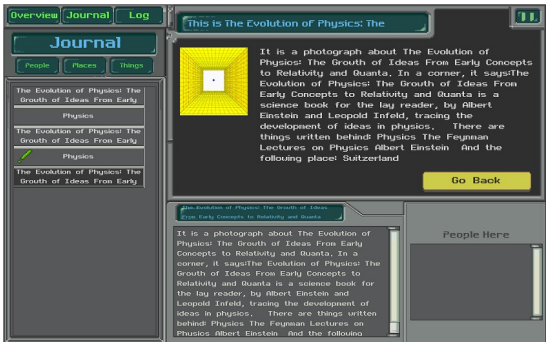
- The *Overview Tab* displays a list of suspects and current items in the player's inventory (top and bottom of the side-panel in Fig. 7.2c). If the player clicks on a suspect, the Description Panel will display all known information about this suspect as well as their known whereabouts (see Fig. 7.2f).

As noted above, *DATA Agent* provides the player with a journal containing notes about people, places and objects that they encounter in the game, as shown in Fig. 7.2a. When a player encounters something for the first time, the journal provides a glowing notification and an exclamation mark near the newly added journal entry for the player's convenience. Clicking on an entry will display information in the Description Panel. This was designed to act as an easily-referenced encyclopedia in case the user wants more information about a specific game item. Any fact discovered about a suspect is also written to the journal for the player to easily access during an *interrogation*, described in the Section 1.3. The addition of the journal and the activity log allows the user to more easily keep track of their past actions and discoveries, as well as the tasks that they still have to do (i.e. new journal entries); this addresses usability concerns in past games in the *Data Adventures* series.

(a) NPC Dialog

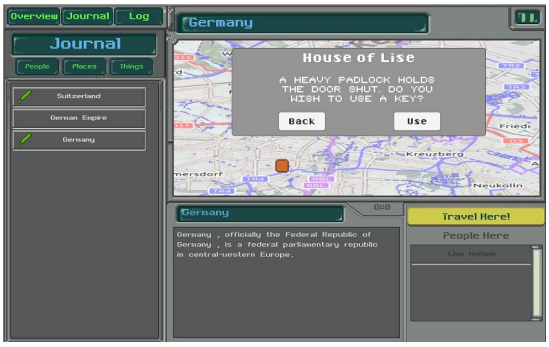


(b) City screen, where buildings can be selected.

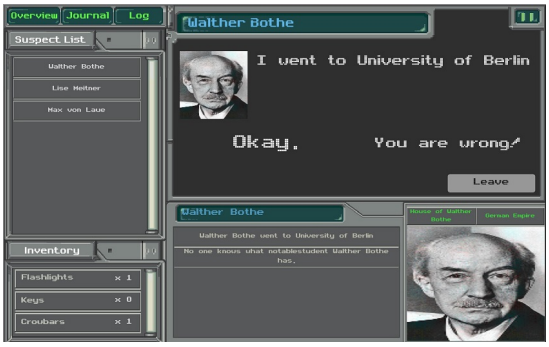


(c) Item interaction

(d) Collectible key



(e) Building that can be unlocked with the key



(f) Interrogation screen

Figure 7.2: Different screenshots of the *DATA Agent* User Interface, from the playtested Albert Einstein game.

1.3 Game loop

Players must talk to people and inspect items to learn clues and facts about suspects in order to win the game. In this version of *DATA Agent*, dialogue reveals facts about suspects, and both dialogue and items reveal new cities and buildings to explore. Along the way, the player may encounter buildings they are unable to initially visit, unless they spend a consumable item to unlock it. For instance, the player may come across a locked building (see Fig. 7.2e)

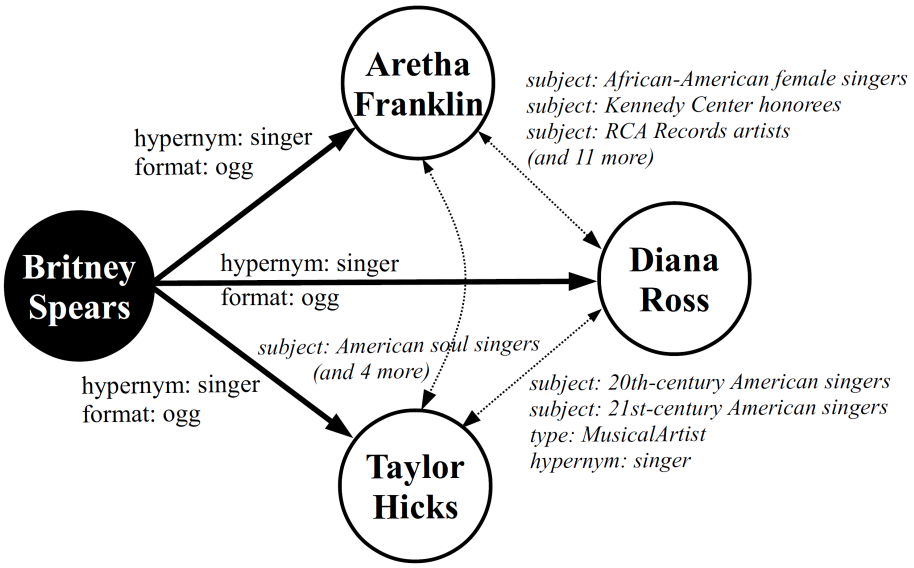


Figure 7.3: Example for suspect selection and its fitness calculation, in the Britney Spears testbed game. The fitness attempts to maximize the common predicates and objects between the victim and the suspects (bold arrows) and between the suspects themselves (dotted arrows, italics). Some of the links are shown in the format "⟨predicate⟩: ⟨object⟩".

that needs a key. Only after finding a key (see Fig. 7.2d) and using it at the locked location will the player be able to enter and inspect the items and people within.

After finding a suspect (initially their locations are hidden from the player), they may talk to him/her to start an *interrogation*. During an interrogation, the suspects list facts about themselves, to which the player can respond "Okay" or "You are Wrong!" (see Fig. 7.2f). Facts mentioned by the suspect are the same ones that the player will have acquired from other NPCs in the game. If a player presses "Okay", the suspect continues to list another fact; if the player presses "You are Wrong!" then a pop-up allows the player to "Arrest This Person." The culprit of the murder always lists an incorrect fact (borrowed from another suspect). If the player chooses to arrest the suspect, the game ends. If this suspect was the culprit, a message appears congratulating the player, otherwise another message mentions the collapsing time line (and the end of the world) caused by arresting the wrong suspect (see Fig. 7.1c). We decided to use a suspect/culprit mechanic because of the interface it creates for the player to interact with the raw data, since, by having players fact check each suspect, they have to collect and verify each fact.

2 Generating adventures

The system generates an adventure seeded by a person's name, similarly to *WikiMystery*. A character representing this person's *Wikipedia* article becomes the victim of the murder on which the plot builds around. Initially, the system finds a large pool of possible suspects, and narrows it down to a select group of suspects that will appear in the game. Then, it finds paths of links between suspects and the victim, which eventually will shape the game's plot. The following sections further explain this process.

2.1 Suspect selection

Similar to *WikiMystery*, at the beginning of the suspect selection process, the system performs a series of queries to find people related to the victim. These queries use *DBPedia*, and the people it finds become a pool of possible suspects. Suspects must have a direct link to the victim, sharing at least one characteristic value (object) with the them. The queries used in *DATA Agent* are improved versions of those used in *WikiMystery*. See Fig. 7.3 for a sample of shared predicates and objects.

The system also uses an $\mu + \lambda$ evolution strategy with cascading elitism[212] over the suspect pool to select an optimal group of suspects. However, it has different representations, operators and fitness function. Based on informal playtests of *WikiMystery*, we noticed that players would often be either overwhelmed by the amount of information or bored by the repetitiveness of the gameplay. Our solution was to decrease the amount of suspects from 5 to 3, essentially removing two paths from the plot tree. Thus, each genome consists of a set of n suspects, specifically 3 suspects per case. The new mutation operator, then, simply swaps one or more suspects with another from the pool.

The system uses two fitness functions in a population of 50 individuals. The first function, the sum of the amount of direct links between the victim and the suspects in the genome, sorts the population first, removing the worst half of it. Then it uses the sum of the amount of direct links between the suspects to sort the remaining population, eliminating another half and leaving only 25% of the original population. This 25% is then copied and mutated to return the population to its original size. The first fitness function encourages

individuals that have more connections between suspects and the victim, while the second favors suspects that are deeply linked between themselves. The process ends after 500 iterations, and the trio of suspects with the highest fitness is chosen as suspects for the game. Out of these, one is marked at random as the culprit.

Fig. 7.3 illustrates how fitness is calculated. We theorized that this fitness will result in a mystery where suspects have more common traits with the victim and themselves. We experimented with a different function for the second sorting process, which rewarded suspects with different characteristics, but the resulting sets were deemed too random based on a brief subjective evaluation.

2.2 Path generation

The generator creates each game element, be it an NPC, item or location, from articles linking the victim to each suspect. For each suspect, a series of search queries is made to *DBPedia* in order to find paths of articles that link, respectively, the victim to the suspect. Due to computational limitations related to querying the *DBPedia* endpoint, we limited the maximum path length to 4 edges (i.e. 5 articles) per path. Types of articles in one path are compared to all types found in all paths within a given pair of victim and suspect. All paths are sorted based based on their diversity and length, favoring those that are longer *and* contain articles of different types (e.g. places, people, etc). For example, a path with 4 articles is preferred to one with 2; and a path with an article about a person and one about a place is preferred to one about two places. After sorting, the best path is selected for that victim and suspect. This approach presents small, but important, deviations from the previous versions of *Data Adventures*. The old systems repeat the path generation process once, essentially finding a minor best path for each pair of articles in the main (major) best path that leads from a victim to the suspect. This results in paths that are far longer, thus increasing the gameplay time. For example, if you assume that the system will always select a path with the highest possible length l, let the minor paths cap at length $l - 1$, and let n be the amount of suspects, a *DATA Agent* adventure will have roughly $n * l$ minimal game objects, while a *WikiMystery* one will have $n * l * (l - 1)$ objects. For 3 suspects

and 5 articles in a path, *DATA Agent* would have 15 objects, and the *WikiMystery* generator (Chapter 6), which preceded *DATA Agent*, would have 60. While one may argue that longer games may be more interesting, playtesting showed us that when the player's interaction options are not varied enough, the game becomes repetitive and boring.

Once every suspect has a best path, the system generates in-game objects for each node in the paths: articles about locations are transformed into in-game cities and buildings; those about people become NPCs, and anything else is made into a game item, such as a book or a letter. Items and NPCs are placed in buildings within a city and are given descriptions, images, names, dialogue lines as applicable. Items' descriptions and NPCs' dialogues give clues that can reveal new buildings in the current city or a previously seen (or unseen) city, until finally the player can reach a suspect. When necessary, new NPCs or items are generated to fill gaps in the path (e.g. when an article is about a city, an NPC related to it is created and added in a building within that city). Finally, lock and key puzzles are added to the game paths using a variation of a breadth first algorithm that guarantees that each building can be unlocked with keys found before it.

2.3 Dialogue generation

DATA Agent uses the *Tracery* grammar system [48] to create dialogue. Dialogue can be of three types: *essential* dialogue, *fact-giving* dialogue or *immersion* dialogue; each requires their own unique grammar. Essential dialogue gives clues about game objects, progressing the story and making the locations containing them accessible. Fact-giving dialogue reveals information about suspects which is important when the player is interrogating a suspect, as the suspect may give false information during that stage. Immersion dialogue gives information about the NPC that the character is talking to, such as their birth date or what they are known for, and it is not essential to the progression of the story. Not every NPC has fact-giving or immersion dialogue, and suspects that do not have dialogue at all: their conversation triggers an interrogation, following a different mechanic than previous character interactions. Table 7.1 defines several sentences used in the game, possible example lines that could be generated for this sentence, the sentence's dialogue type, and the speaker of

the sentence.

3 Summary

This paper presented the *DATA Agent* system, which generates adventures based on *Wikipedia* that task the player with solving the fictional murder of a real-world person. The system utilizes linked data to discover appropriate suspects for the murder within *Wikipedia*, and places these real-world people, locations, and encyclopedic subjects in a fictional setting where time and space are relative. *DATA Agent* is the latest installment in the *Data Adventures* series, and comes with an extensive overhaul of the user interface and the exposition of the game's backstory. This system, however, was not evaluated. The new narrative pushes the system further away from the data fidelity spectrum of data-driven dimensions, but we believe it provides a better overall experience. Unlike its predecessors, *DATA Agent* aims at providing an engaging and fun experience. To validate that, we ran a user study on the game, which will be described in the next chapter.

Table 7.1: Example dialogue lines for different sentences. "Type" defines whether the dialogue is *Essential*, *Fact-giving*, or *Immersion*, and "Speaker" designates who speaks the line in-game (Player or NPC). Words in italics are instantiated in each game based on information for this NPC or the plot of the adventure.

Sentence	**Example Lines**	**Type**	**Speaker**
Greeting-to-Player	"Hello? Can I help you?", "Hi. Do you need something?", "Hi."	Essential	NPC
Central-Hub	"Greetings, I'm a private-eye. I'm sorry to disturb, could I ask you some questions?", "Greetings, I'm a detective. Can I ask you some questions?"	Essential	Player
Central-Hub Response	"Okay. Ask away.", "Sure. I'll do my best.", "Of course. I can try."	Essential	NPC
Name-Query	"Please, tell me your name.", "Who are you?", "Would you tell me your name?"	Immersion	Player
Name-Response	"Yes, I'm $personName$", "I am known as $personName$.", "I am called $personName$."	Immersion	NPC
Clue-Query	"Is there something you think I should know about?", "Do you have something that might be useful to the investigation?"	Essential	Player
Clue-Response-Building	"The victim often traveled to a place named $building$.","The victim often visited a place known as $building$.", "I believe the victim talked about $building$."	Essential	NPC
Clue-Response-Concept	"If I remember correctly, the victim talked about a thing called $thing$.", "If I recall correctly, the victim mentioned $thing$.", "I think the victim mentioned a thing named $thing$."	Essential	NPC
Clue-Response-Place	"Last I thought, the victim went to $place$.", "Last I heard, I heard the victim visited $place$."	Essential	NPC
Clue-Response-Person	"The victim was friends with $personObject$.", "Last I heard, the victim was acquaintances with $personObject$. Perhaps they know about something."	Essential	NPC
Suspect-Fact-Query	"Would you give me any information about one of the suspects?","I need some help. Would you tell me some information about one of the suspects?"	Fact-giving	Player
Suspect-Fact-Response	"$suspectName$ is an "$attribute$."	Fact-giving	NPC
Residence-Query	"Where do you currently reside? It might help with the investigation."	Immersion	Player
Speculation-Response	"I live in $place$.", "That's none of your business!"	Immersion	NPC

Chapter 8

Evaluation of engagement in *DATA Agent*

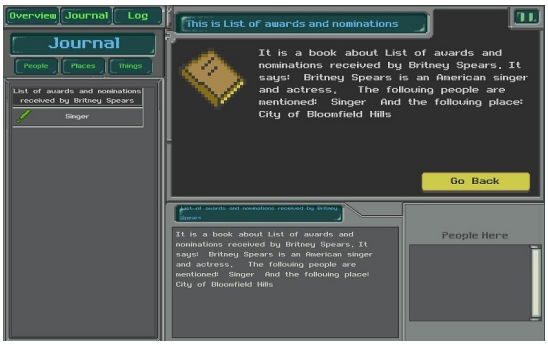
This chapter describes the methodology and results of an user study on *DATA Agent*. In it, 30 individuals played two games generated by *DATA Agent* and answered a questionnaire. We will discuss the findings of this study. Key findings from the user study include the fact that players were curious to play more mysteries with different people, the interface and general interactions were intuitive, and that finding the solutions to the mysteries was straightforward.

1 User study

In order to test the games generated via the *DATA Agent* system, a user study was conducted with 30 participants playing two *DATA Agent* games. The protocol followed, the games tested, the participants' demographics, and an analysis of their gameplay and feedback is provided in the following sections. Any statistical significance analysis reported is made with $\alpha = 0.05$ threshold.

(a) Singers at the house of the deceased, in McComb, Mississippi (the birthplace of Britney Spears)

(b) A suspect makes a claim which can be falsified



(c) An item consisting of a very sparse list

Figure 8.1: Screenshots of the playtested Britney Spears game.

1.1 Experiment design

Subjects were asked to individually playtest two games generated by *DATA Agent* and give their opinions about the game. Participants were found through use of school email lists and social media, through which we asked for volunteers. All participants were brought to a quiet classroom setting. Before beginning the test, the participants were informed about open data, data games, and the motivations behind the research. It was stressed that everything in the *DATA Agent* games had been generated from open data using artificial intelligence and had not been hand-authored.

Subjects first filled out a pre-game questionnaire to obtain demographic information (gender, age), play habits (frequency and preference for games) and experience with adventure games (preference and three example adventure games). The subjects were then asked to play two *DATA Agent* mysteries: **The Case of Albert Einstein**, and **The Case of Britney Spears**. These two cases were selected for their difference in terms of gender,

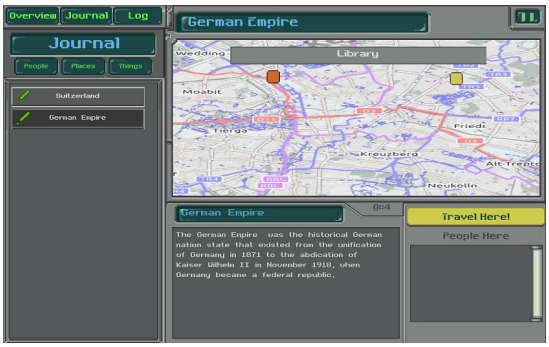
type of celebrity, and contemporaneity. The order of games was randomly selected for each participant. During play, participants were asked to freely voice their opinions about the game and the data, as all playthroughs were audio recorded. Researchers were available in case the player needed help or had questions about gameplay. All in-game player actions were logged, and specific game events (e.g. cities visited) were logged and analyzed in Section 1.4. After the participants completed both mysteries (regardless of whether they won or not), participants were asked to fill out a post-game survey. The survey contained 11 questions regarding the gameplay and the broader appeal: answers were on a 5-scale Likert scale from "strongly disagree" to "strongly agree". Additionally, the survey included 4 questions regarding broader suggestions and reflections: answers were free-form text. After the experiment was over, participants were compensated with a $10 Amazon Gift Card.

1.2 Tested Games

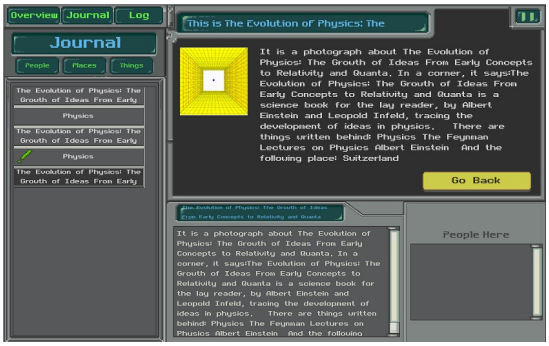
In **The Case of Albert Einstein** (see Fig. 8.2), players set out to solve the murder of Albert Einstein, for which the suspects are Walther Bothe, Lise Meitner, and Max von Laue; Lise Meitner is the (doppelganger) culprit. This game involves regular NPC's such as Langston Hughes, Franklin D. Roosevelt, and Harry Emerson Fosdick. The player can travel between three cities: Switzerland, The German Empire, and Germany. In total, there are 10 buildings (e.g. "House of Lise Meitner" in Fig. 8.2e, "Library" in Fig. 8.2b), 9 items (e.g. "a photograph about The Evolution of Physics: The Growth of Ideas From Early Concepts to Relativity and Quanta" in Fig. 8.2c), 7 people, and 41 dialogue nodes to explore.

In **The Case of Britney Spears** (see Fig. 8.1), players set out to solve the murder of Britney Spears, for which the suspects are Aretha Franklin, Diana Ross, and Taylor Hicks; Diana Ross is the (doppelganger) culprit. This game involves regular NPCs such as Whitney Houston, Jamie Lynn Spears, and Katy Perry. The player can travel between five cities: "McComb, Mississippi"; "New York City"; "Birmingham, Alabama"; "City of Bloomfield Hills"; and "Detroit, Michigan". In total, there are 10 buildings, 7 items (e.g. "a book about List of awards and nominations received by Britney Spears" in Fig. 8.1c), 10 people, and 52 dialogue nodes to explore.

(a) NPC Dialog

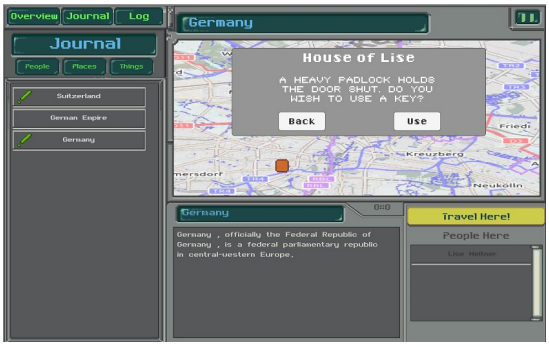


(b) City screen, where buildings can be selected.

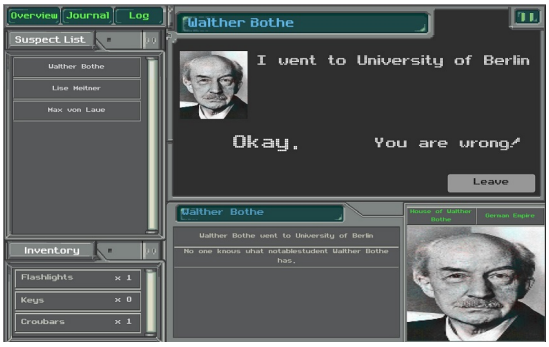


(c) Item interaction

(d) Collectible key



(e) Building that can be unlocked with the key



(f) Interrogation screen

Figure 8.2: Different screenshots of the *DATA Agent* User Interface, from the playtested Albert Einstein game.

1.3 User Data

In total, 30 adults (22 identifying as male, 7 as female, 1 as other) participated in the user study. The median participant age was 27.5, but most participants (53%) were between 19-25 years old. An overwhelming majority (93%) of playtesters reported that they enjoyed playing games, and a slightly smaller majority (83%) reported enjoying playing adventure games specifically. Two thirds of playtesters claimed to play video games at least as often as

2-3 times a week, while only 13% of players played less often than once a week.

1.4 Gameplay data

Overall, 23 participants managed to find the culprit in both games while 28 participants found the culprit in at least one game. There was no clear difference between the number of people who "won" the Albert Einstein game (25 players) over the Britney Spears game (26 players). How users interacted with different game entities in each game is shown in Table 8.1. Obviously, players interacted more with people and cities in the Britney Spears game where there were more people and cities. Normalizing interactions based on the number of game objects of that type shows that the differences between the two games were not as pronounced. However, it is obvious that the Britney Spears game required more player attention in general; based on general feedback and the average interaction ratios shown in Table 8.1, that game required a longer playtime as well. In general, it was surprising to us that the average interactions with NPCs is smaller than the number of NPCs of each game, meaning that some NPCs were not interacted with at all. Indeed, the NPC interactions were fewer than the NPCs in 73% of Einstein games and in 63% of Spears games. It is not surprising that players visited cities more than once, however, as each new clue often revealed a new building in existing cities (or a new NPC in an existing building) and required players to travel back to previously visited cities. It should be noted that players had very different interaction patterns: for instance, in the Albert Einstein game players interacted between 3 and 15 times with NPCs. It should be noted that the two players who had only 3 NPC interactions managed to win the game. In general a correlation analysis between the raw or normalized gameplay values of Table 8.1 and whether the game was won showed no significant correlations.

1.5 Usability Data

For the sake of brevity, we discuss only the responses of users which were provided in a Likert scale. The results of these responses are summarized in Table 8.2. We use the number of positive responses ("agree" or "strongly agree") versus the number of negative

Table 8.1: Average interaction ratios for all playthroughs, along with their 95% confidence intervals. The normalized (nrm.) values are divided by the number of respective game objects in each game.

Values	**City**	**Building**	**Item**	**People**	**Dialogue**
Albert Einstein game					
Raw	8.4±2.4	15.5±2.8	9.7±0.8	5.8±0.9	43±5.7
Nrm.	2.8±0.8	1.6±0.3	1.1±0.1	0.8±0.1	1±0.1
Britney Spears game					
Raw	10.6±2	16.1±2.6	7.3±0.4	9±1.1	66.7±8.8
Nrm.	2.1±0.4	1.6±0.3	1±0.1	0.9±0.1	1.3±0.2

Table 8.2: Weighted average (1 for strongest disagreement, 5 for strongest agreement) of user's responses to specific post-game questions and its 95% confidence interval. Also shown are the number of positive (agree and strongly agree) versus negative (disagree and strongly disagree) responses.

Question	**Average**	**Pos./Neg.**	
DATA Agent is an adventure game.	3.77±0.35	21	3
DATA Agent is fun, relative to other adventure/role-playing games I have played.	3.4±0.26	15	2
DATA Agent is challenging, relative to other adventure/role-playing games I have played.	2.37±0.35	4	17
It is easy to understand how to play *DATA Agent*.	3.87±0.35	22	4
I learned something new playing *DATA Agent*.	3.53±0.34	20	5
I would get *DATA Agent* if it were a fully developed as a mobile or computer game.	3.4±0.39	16	6
I would recommend *DATA Agent* to a friend if it were fully developed as a mobile or computer game.	3.6±0.32	19	3
I would like to play *DATA Agent* again with the same mystery.	2.3±0.39	5	20
I would like to play *DATA Agent* again with a different mystery.	3.83±0.38	24	5
Finding the culprit in *DATA Agent* was straightforward and made sense to me.	3.9±0.37	21	4
Finding the culprit in *DATA Agent* requires mental effort.	2.6±0.36	7	15

responses ("disagree" or "strongly disagree") as a granular measure that shows the inter-rater agreement. Overall, the number of "neutral" answers were low, with a notable exception regarding whether "*DATA Agent* is fun…" (with 43% of "neutral" answers). Similarly, participants tended to agree in terms of either positive or negative responses. Considering the

usability assertions where users' positive responses were more than quadruple the number of negative responses (and vice versa), we can safely conclude the following users' assertions:

1. *DATA Agent* **is** an adventure game.
2. *DATA Agent* **is** fun, relative to other adventure/role-playing games I have played.
3. *DATA Agent* **is not** challenging, relative to other adventure/role-playing games I have played.
4. It **is** easy to understand how to play *DATA Agent*.
5. I **would** recommend *DATA Agent* to a friend if it were fully developed as a mobile or computer game.
6. I **would** like to play *DATA Agent* again with a different mystery.
7. Finding the culprit in *DATA Agent* **was** straightforward and made sense to me.

It should be noted that the above conclusions ignore neutral responses. If we consider assertions with more than quadruple positive responses than negative responses and quadruple positive responses than neutral responses (and vice versa for negative), only assertions (4), (6) and (7) hold.

Similarly to gameplay data, we investigated whether the user's responses correlate with their winning streak. The assertions "I learned something new…" and "finding the culprit […] was straightforward…" had a significant positive correlation with the number of games won by that player ($r = 0.36$ and $r = 0.46$ respectively). Moreover, significant positive correlations were found between the usability assertion "It is easy to understand how to play *DATA Agent*." and agreement regarding "In general, I like to play games." ($r = 0.44$) and "I like to play adventure games" ($r = 0.53$). These correlations should not be surprising, as the interface design of *DATA Agent* was inspired by classic point-and-click adventure games; users familiar with games and especially with adventure games would be more likely to find this game easier to play.

2 Discussion

The conclusions drawn from an user study with 30 participants are generally positive. Most participants agree that *DATA Agent* is easy to understand, and solving the mystery was straightforward. The major overhaul of the user interface, compared to previous titles in the *Data Adventures* series, seems to result in an UI that is intuitive to use, especially by users more familiar with games in general and adventure games in particular. While the majority of players would not try the game again with the same mystery, they would like to try the game with another mystery. This is hardly surprising, as adventure games in general are rarely replayable. With the small set of interactions (few suspects, people, and cities), participants in this version of *DATA Agent* likely had explored all possible aspects of each mystery on their first try. It can be assumed, on the other hand, that a generator such as the one in *DATA Agent*, which can create infinite mysteries, can keep players engaged for a long period of time. We can assume that a large part of this appeal is not the fun gameplay itself (which we discuss later), but the players' natural curiosity regarding which associations the generator will make and how those will be presented to them in terms of visuals and dialog.

On the other hand, the participants' responses highlighted that the game now is perhaps too easy. Most participants agree that the game is not challenging. We could also interpret the large number of neutral responses in terms of how fun the game is, or the agreement that finding the culprit is straightforward as further proof that the straightforward solutions offered by the game make it feel "bland". To a degree, many of the design decisions taken when generating the two testbed games were explicitly to reduce the length and complexity of gameplay. The smaller number of suspects (three versus five in earlier efforts (Chapter 6)) and the shorter paths (without minor paths (Chapter 6)) all contribute to a shorter playtime and a more direct way to reach each suspect. This reduces the time investment of interested participants and allows us to receive more feedback from more users faster. Additionally, it highlights important information, that would otherwise be lost amidst a large amount of content: *WikiMystery* generates, on average, 135 game objects per adventure (Chapter 6), including NPCs, locations and items; while *DATA Agent* cuts that down to 30 game objects. However, it is not easy to estimate how to increase the challenge of the game without making

it unplayable or banal. Re-introducing more suspects would increase the number of people, buildings, cities and items that the player has to interact with (prolonging playtime) and may increase the challenge as players will need to interrogate and find facts for more suspects. On the other hand, this could result in more "trivial" interactions that feel inconsequential, and could clutter the interface (e.g. by having 30 or more people in the peoples tab with little ability for the player to understand which one has been recently added (or may hide more information). Towards increasing the "mystery" aspect of the game, the vital facts for interrogating suspects could be hidden away in more locations (possibly locked ones) instead of being offered by NPCs at the house of the deceased. Moreover, having multiple ways to reach the same fact or suspect could make the choices of the players more meaningful; for instance, the game could have one key that can open two locks (i.e. the player will only have access to one but not both of those buildings) so the game can be solved with the clues hidden in either but not both of those locations. Other additions, such as a broader set of dialogue options for immersion, are likely to enhance the curiosity and learning potential that motivates and benefits the player respectively.

In terms of broader directions for taking *DATA Agent* forward, the game design and algorithmic implementation should capitalize on the strengths noted by participants in the user study and minimize the negative factors such as the perceived lack of challenge. As noted, more extensive dialogue could lead to more interesting interactions: such dialogue generation could take more advantage of open data (allowing users to ask more from each NPC beyond their name and the subject they are known for), or follow authored templates to introduce more murder mystery tropes such as other NPCs offering potential motives or alibis for the suspects based on their relationship with the suspect (which can be calculated from open data, or not). Facts about the suspects could be chosen more wisely, since certain facts are nonsensical (e.g. in Fig. 8.1b one of the facts for Diana Ross is "Diana Ross is a soundrecording of dianaross 1") or trivial to ascertain the veracity of (e.g. "I was a singer" in Fig. 8.1b). This finding is corroborated by the user survey, as players managed to solve each game despite talking to a very small subset of NPCs. Finally, the cities used in the game could be improved, both in terms of name (e.g. players can travel to "The German Empire" in the Einstein game, which is hardly a city) and in terms of the city map used to

display them (e.g. in Fig. 8.2b). Information from OSM could be used directly to place actual buildings in their correct location on the map, rather than at random as it is currently. The buildings from OSM could show a generic background (e.g. a generic "Supermarket" background if the OSM location says that this location is a shop of any type) or could be specific to the location based on a search for images with that specific location keyword (e.g. an image of the "Coliseum" at its exact location in Rome).

Since the previous installment of the *Data Adventures* series, described in the Chapter 6, there have been substantial improvements with how the data is presented to the player. However, generating games from open data is still heavily dependent on the uncontrollable nature of said data. Games often include facts that are nonsensical, as pointed out above ("Diana Ross is a soundrecording of dianaross 1"). Sometimes the image for a character is not correct, and several times throughout the user study, players commented on this fact. We, however, find it nearly impossible to ever fully control or constrain these occurrences, since doing so would involve handpicking data, defeating the purpose of open data games. In the long run, an important question is to what extent this type of data-driven game generators can ever be trusted to be fully autonomous, given the bizarre and sometimes even offensive situations and characters that can occasionally emerge even from correct underlying data.

3 Summary

This chapter presented a user study on *DATA Agent*. The resulting games are shown to be easy to grasp, straightforward to solve, and generally fun to play. Based on a user study with 30 participants, two games generated for *DATA Agent* were deemed fairly easy to solve, based on a high completion ratio as well as user feedback in the end of the experiment.

Chapter 9

Data-driven game design

This chapter attempts to explore the design space of maximalist data-driven games (and other data games) in order to form an initial understanding of it. It is also an attempt to systematize reflections from our own and others' attempts at creating such games. We discuss how the games described in this thesis fit in the dimensions of data-driven game design described in Chapter 3. Additionally, we address the following questions:

- For what purposes can data-driven maximalist games be designed and how does that affect their character?
- What new legal and ethical issues, including copyright issues and the potential for generating offensive, misinforming and biased content, are raised by this type of game design?

1 Instances of Data-driven Design

Fig. 3.1 shows an updated version of Fig. 3.1, despicting the two dimensions described previously in Chapter 3. The X-axis represents Data Transformation versus Data Fidelity, while the Y-axis represents Functional versus Decorative. Games where the goal is to preserve the original data, adapting the game to do so, are on the leftmost side of the figure: *WikiRace* and *OpenTrumps* exemplify this. Data in these games is also extremely functional: all mechanics in these games rely on a direct interaction with and understanding of the data

(in *WikiRace* through reading the articles, and *OpenTrumps* through the values that affect deck superiority). Less faithful to the data, but similarly functional, is the *FreeCiv* map generation where geographic data is used as-is; resource placement is based on the original data but also adapted (i.e. transformed) for playability.

DATA Agent (Chapter 7) is the most extreme example of data transformation we have. In the game, the engine transforms individual facts about separate people into a murder mystery. The facts are also transformed into dialog lines, used by NPCs when prompted by the player. Some facts are altered purposely, in order to "lie": the culprit's dialog differs from reality in order to point to the time-agent (and thus, the player) that he or she is guilty. *WikiMystery* (Chapter 6), on the other hand, uses proof of innocence in a similar manner but never misrepresents the actual data: all proofs given by NPCs are true, and the player must memorize them in order to use them in the game's accusation sequence. *Data Adventures* (the game; Chapter 5) is even less transformative, but not as faithful to the data as the *FreeCiv* map generator: for example, houses of characters are said to be where they were born or where they died, which may not necessarily be correct, and characters of different time periods appear at the same time.

Moving upwards along the Y-axis, we find *Open Data Monopoly* [79] and *WikiMystery*. Both use data in a decorative manner, the former as names for lots on the board and the latter as images, but some of the original data is also functionally translated (e.g. lots prices in *Open Data Monopoly* and proof of evidence in *WikiMystery*). On the far end of the Y-axis we have *ANGELINA* [55], which uses visuals as framing devices but without affecting the core platformer gameplay. The visuals are based on text of a newspaper article: the source data is transformed via natural language processing, tone extraction, and image queries.

While our projects were, in relation to the X-axis, progressively less faithful to the data and more transformative, the same linear progression cannot be seen in the Y-axis. The first project, the *FreeCiv* map generator, was the most functional of all our experiments, partially because of the game genre. It was easier for the data we used to affect the mechanics of a strategy game than those of a point-and-click adventure one. *Data Adventures*, on the other hand, was the furthest up towards the decorative spectrum of games, with *WikiMystery* and *DATA Agent* occupying spaces in-between.

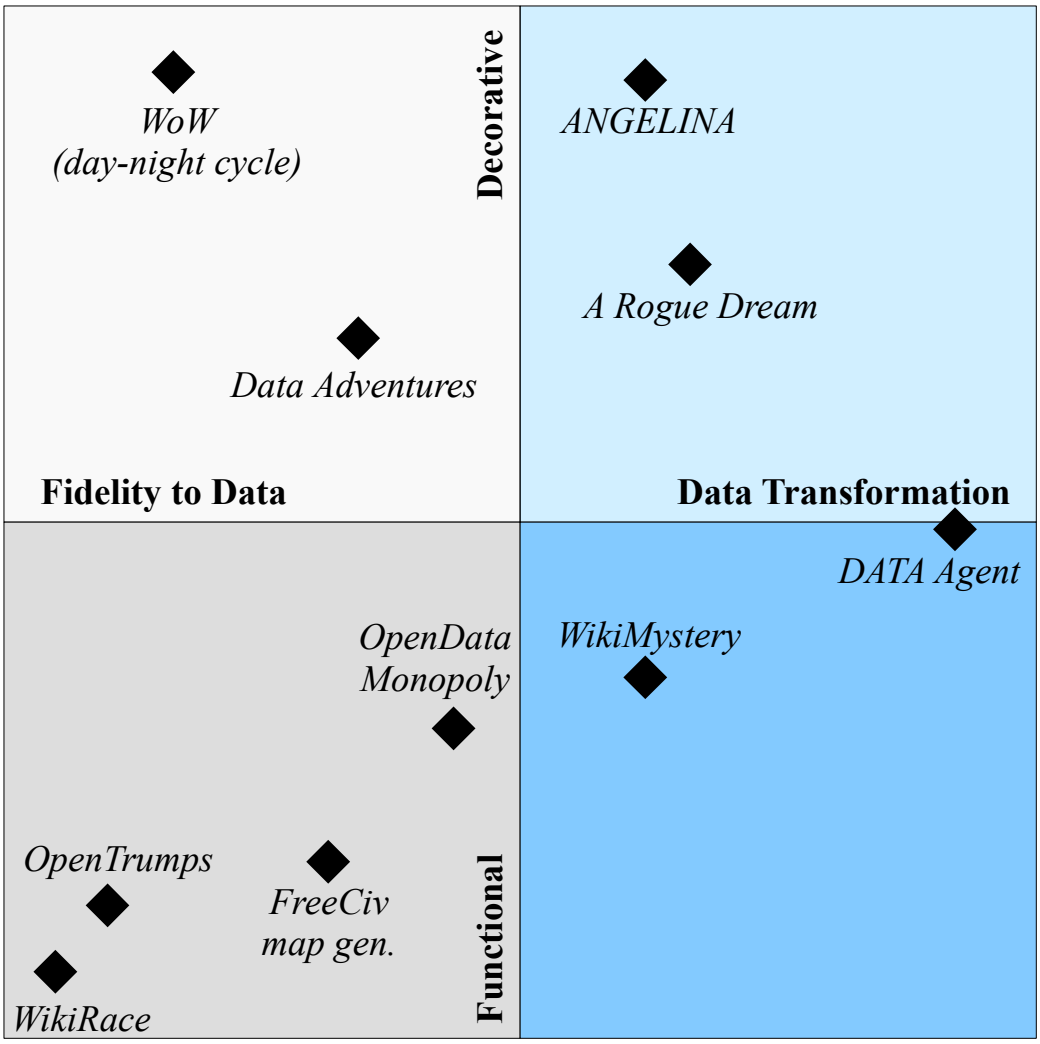


Figure 9.1: Examples of games within the two dimensions.

2 Purposes of Maximalist Games

While we attempt to highlight the principles and directions for designing maximalist game experiences, it is important to consider the purpose of such a game. Maximalist game design can be desirable for many different reasons, which we highlight in this section. Depending on the purpose, moreover, the design priorities could shift in the spectrum of data transformation versus data fidelity or decorative versus functional.

2.1 Learning

Modern-day practice sees students of all levels refer to *Wikipedia* for definitions, historical information, and tutorials. When browsing such knowledge repositories, it is common to access linked articles that may not have been part of the topic of inquiry. Maximalist game design that exploits open sources of knowledge, such as *Wikipedia*, can be used as a tool for learning in a playful context. One strength found in games is their ability to motivate a player for long periods of time. They also allow several ways to engage players, which can vary based on decisions and learning goals of game designers. Furthermore, games present failure as necessary in the course of learning [163], causing players to explore and experiment more, since failing in game is less consequential than in the real world [93]. Studies have also shown that when games immerse the player in a digital environment, they enhance the player's education of the content within the game [60]. Games, unlike raw open data, are adaptive to players' skill level and are the most fun and engaging when they operate on the edge a player's *zone of proximal development* [224]. Thus, we believe players can learn facts within open data during gameplay.

Data-driven maximalist games intended for learning can highlight and allow the player to interact with the data playfully. DATA Agent, to a degree, builds on this concept by creating NPCs out of articles about people, whether they are historical or fictional. These NPCs can then answer questions about their birth date or life's work. More relevant to the game progression, each NPC leads to an object (NPC, item or location) about another article, which can be interacted with and is associated to the current NPC somehow. The *Data Adventures* series was not designed with the explicit purpose of learning in mind; possibly, alternative design priorities could build on more "educational" principles. It would be possible, for example, to check for understanding by asking the player questions relating to the data in a diegetic manner — not unlike *DATA Agent*, where the player must interrogate suspects and cross-check with their own obtained knowledge to detect falsehoods.

Information used to instantiate data should be fairly transparent to the player-learner. Therefore, transformation of data should not be convoluted, and perhaps even textual elements from original articles can be used as "flavor text". The veneer between encyclopedic

content and game content does not need to be thick, in order to ensure that the right information is provided. In terms of function versus decoration, maximalist games for learning tend to edge closer towards data that influences the outcome of a game session in order to motivate learners to understand and remember the data. Such checks for understanding, however gamified they may be, will have an impact on the success or failure of the game.

2.2 Data exploration

Data transformed into interactive game content, forming a consistent whole that goes far beyond the sum of its parts, can allow human users to explore the data in a more engaging way. Data visualization has been used extensively with a broad variety of purposes — far beyond the ones listed here — to take advantage of how most humans can more easily think through diagrams [222]. In that vein, gameplay content originating from data can act as a form of highly interactive data visualization. The fact that data from different sources is combined together based on associations imagined by an automated game designer allows players to reflect on the data and make new discoveries or associations of their own. Due to the potential of emotional engagement that games have beyond mere 2D bar plots, the potential for lateral thinking either through visual, semantic, gameplay or emotional associations [174] is extraordinary. In order for a game to offer an understanding of the data that is used to instantiate it and allow for that data to be re-imagined, the transformation into game content should be minimal. Examples of data games which already perform such a highly interactive data visualization are *BarChartBall* [213] or *OpenTrumps* [39]. However, a more maximalist approach could benefit games like the above by providing a more consistent storyline and progression, as well as a stronger emotional investment in the data.

2.3 Contemporaneity

Automated game design has been always motivated, to a degree, by the desire to create perpetually fresh content. With data-driven design, this can be taken a step further by generating a new game every day. Such a game could be contextually relevant based on

the day itself, e.g. building around historical events which happened on this day (such an extensive list can be found on `onthisday.com` and *Wikipedia*) or people who had important personal events on that day (e.g. date of birth, date of death, graduation day). Moreover, the social context can be mined and used to drive the automated design process by including for instance trending topics of Twitter or headlines of today's newspapers. Early examples of such data-driven process have been explored for example by *ANGELINA* [55] which used titles and articles from The Guardian website and connected them with relevant visuals and the appropriate mood. It is expected that a more maximalist data-driven design process would strengthen the feeling of contemporaneity by including more data sources (i.e. more data to transform) or stronger gameplay implications (i.e. broader transformations and functional impact).

Contemporaneity can make games generated on a specific day appealing to people who wish to get a "feel" for current issues but not necessarily dig deeply. On the other hand, the plethora of games (30 games per month alone) and the fact that each game is relevant to that day only could make each game itself less relevant. Contemporaneity and the fleeting nature of daily events could be emphasized if each game was playable only during the day that it was produced, deleting all its files when the next game is generated. This would enhance the perceived value of each game, similarly to *permadeath* in rogue-like games as it enhances nostalgia and the feeling of loss when a favorite gameworld is forever lost.

Any maximalist game could satisfy a contemporaneity goal, but such games can be more amenable to data transformation. For example, data could be transformed to more closely fit the theme of the day, e.g. query only female NPCs on International Women's Day. Contemporaneous data can be functional (to more strongly raise awareness of issues) but can also easily be decorative, e.g. giving a snowy appearance to locations during the Christmas holidays.

2.4 Personalization

When game content is generated from data, it is possible to highlight certain bits of information. When the game takes player input as part of the data selection process, it personalizes

their experience. If player information is available in the form of interests, important personal dates such as birthdays, or even social networks, the potential data sources that can be selected to form the game can be narrowed down. Presenting game content which is personally relevant (e.g. adventures with NPCs based on people living before Christ for an archeology student), or contextually relevant (such as solving the murder of an NPC born on the player's birthday) could contribute to a more engaging experience. It might also be possible to tailor the game's source repositories based on such personal interests. There are numerous online wikis, most of which follow a common format; therefore a user can implicitly (via personal interests) or explicitly (by providing a direct URL) switch search queries of a data-driven maximalist game to a specific wiki of choice.

2.5 Opinion & Critique

Often designers want to make a statement through their games. For instance, *Game-o-Matic* [219] creates games from manually defined associations (as *micro-rhetorics*). *September 12th: A Toy World* [142] makes a political statement about the futility of America's War on Terror. Open data could similarly be used in a game to critique some aspect of culture by adding a weight of relevance and realism. For instance, a game such as *September 12th* could use the real map or skyline of Baghdad, or data on daily deaths in Iraq, to instantiate the challenge of the game. Similarly, if a designer wishes to critique the unprofessional use of social media in the White House, they could use real tweets to form dialog lines rather than generating them as in *DATA Agent*.

2.6 Entertainment

Ostensibly, all games have entertainment as a (primary or secondary) purpose. This includes maximalist games, even if they have an additional purpose as listed in this work. It is meaningful therefore to investigate what data-driven maximalist design has to offer to the entertainment dimension of any such game. Since maximalism —as we define it— does not necessarily apply to the mechanics of a game, a more relevant dimension is the end-user aesthetic that such games facilitate, following the mechanics-dynamics-aesthetics

framework of Hunicke et al. [96]. Data-driven maximalist games primarily enhance the aesthetic of *discovery*, similarly to data exploration via such a game, and *expression* if it can be personalized to a user based on provided input such as birthday, hometown or interests. In many ways, data-driven games can enhance the aesthetic of *fantasy* by using and transforming real-world information. DATA Agent, for example, describes an alternate history setting where a famous historical figure has been murdered (often by colleagues). The fantasy aesthetic is further enhanced by having a player take the role of a detective traveling through time and space to interrogate suspects. Other possible aesthetics that can be enhanced through data are *sensation* if the data comes from sources of high quality video, audio, or visuals (e.g. paintings of the National Gallery of London), or *fellowship* if the data comes from other users (e.g. anonymous users' trending tweets or social media postings of the player's friends). Evidently, games geared primarily towards entertainment can be fairly flexible in terms of data transformation, and can adapt the data to the intended game mechanics and game flow. While data can act as a decoration in such games (if intended to enhance the sensation aesthetic), in general games intended primarily for entertainment are fairly focused in the mechanics and feedback loops, and thus data would primarily be transformed into functional elements.

2.7 Human Computation

Presenting hand-picked results from a vast database in an engaging, playful way is not only relevant for humans to consume. The human-computer interaction loop can be closed if human users provide feedback on the quality of the data itself. This human feedback can be used internally by the game, adapting its criteria in order to avoid unwanted data repositories, queries, associations or transformations made to the data. For instance, a future DATA Agent version could re-compute the set of suspects for the next games (removing one or more suspects from the pool of possible suspects) if a player provides negative feedback explicitly (e.g. via a 'report' button) or implicitly (e.g. by being unable to solve the mystery). More ambitiously, the positive or negative feedback of players engaging with the playable —transformed— data can be fed back to the source repositories which instantiated

the game. This can allow for instance misinformation found in *Wikipedia* to be flagged, alerting moderators that either a human error (e.g. a wrong date or a misquote) or malformed data (e.g. unreadable titles) exists and must be corrected. Whether these corrections should be made by an expert human curator, or directly based on player interactions with the game could be a direction for future research.

3 Issues with data-driven game design

Accomplishing *good* data-driven maximalist game design is a challenge. While the previous sections presented ways of doing so, there are still many implementation- or game-specific details which affect the design process. Beyond the core challenge of a good game design, there are several peripheral challenges to the design task itself which however spring from the practice of data-driven design. We elaborate on those peripheral challenges here.

3.1 Legal & Ethical Issues

Any software which relies on external data that it cannot control may be prone to legal or ethical violations. Privacy of personal information may be a concern for a game generated from the social media profile of a user, especially if that game can then be played by a broader set of people. Using results from Google Images may lead to direct infringements of copyrights; using results from models built from text mining, on the other hand, may or may not result in such copyright infringements depending on whether the model returns actual copyrighted material. The issue of copyright becomes more complex when the data is transformed: relevant to data mining, a judge has ruled for fair use for Google Books as "Google Books is also transformative in the sense that it has transformed book text into data for purposes of substantive research, including data mining and text mining in new areas" [194]. One can only assume that transformations of data into game content, depending on the fidelity to the original data and the purpose (e.g. data exploration and education), would make for a clearer case of fair use.

Game content built on fair use or open data combined into an interactive experience may lead to unexpected issues. This is especially true in cases where the player has sufficient

agency to interpret or act upon content of high fidelity with the original data in an open-ended fashion: consider, for example, a violent shooter game where opponents' visual depictions (3D models or faces) are those of Hollywood celebrities. Even in *Data Adventures*, where player interaction is fairly "curated", a generated game featured solving the murder of Justin Bieber (Chapter 6). Apart from the fictional narrative of a popular celebrity's death, the game identifies another celebrity as the murderer: both of these decisions may cause concern to highly visible people (be they depicted murdered, murderers, or suspects). A disclaimer that the game characters are fictional can only alleviate that much of the ethical responsibility of game designers for such data-driven games.

3.2 Misinformation & Bias

Connected to the concerns of misrepresenting contemporary or historical celebrities are the inherent issues of error in the source data. Before data is transformed into game content, open repositories that can be edited by anyone can be saturated by personal opinion and perhaps deliberate misinformation. As noted previously, not all data provided by different stakeholders in the information age are factual; this may be more pronounced in certain repositories than others. Beyond deliberate misinformation, an inherent bias is also present even in "objective" data. For example, algorithms for Google query results or image results are based on machine learned models that may favor stereotypes (based on what most people think of a topic). Even though *WikiMystery* uses what we arguably consider "objective" repositories such as *Wikipedia*, the 8 most popular locations in 100 generated games were in North America (Chapter 6), pointing to a bias of the articles or the DBpedia entries chosen to be digitized. Other cases where misinformation may arise is when different content is combined inaccurately: examples from the *Data Adventures* series include cases where an image search for a character named Margaret Thatcher resulted in an image of Aung San Suu Kyi (Chapter 5). When data-driven design uses social network data such as trending topics on Twitter, then the potential for sensitive or provocative topics to be paired with inappropriate content or combined in an insensitive way becomes a real possibility. If data-driven maximalist games are intended towards critique or opinion, the

misinformation or misappropriation could be deliberately inserted by a designer (by pairing different repositories) or accidentally introduce a message that runs contrary to the intended one.

4 Outlook

Maximalist game design encourages creation through reuse and combination. If one imagines its most powerful form, it would likely involve taking any mixture of information, pouring it into any game content cast, and reveling in its results. It would provide a freedom to interact with any data in the best, most personalized way possible.

Current PCG techniques allow for unlimited playability for a large variety of games. However, they can lack a level of contemporaneity and relevance that could be provided by open data. Additionally, research has suggested that concepts can be effectively learned through gameplay [60]. Using games as a method of interacting with open data may create a novel way for learning about the data in a fun way. Rather than use *Wikipedia* to learn about specific people and places for the first time, players could play games where they can talk to these people and visit these places.

Open data is available to all, to create as well as consume. Sometimes the data is inaccurate. The idea of visualizing this information in any form can provide means to "debug" the original data, in a more engaging way than just browsing *Wikipedia* or poring through a massive database.

4.1 Summary

Finally, we described how maximalist game design can serve different purposes in the design process and which tradeoffs emerge from each purpose.

Chapter 10

Conclusion

This thesis set out to explore the space of data-driven game design. Prior to the work described here, there was little research in this field, and none whatsoever in the subspace of adventure data games. Our contributions come in the form of several algorithms and methods for data-driven content generation for games, tested in one strategy game and three iterations of adventure games. This final chapter discusses our contributions, drawing back from the thesis hypothesis and research questions laid out in Section 1, lays out the limitations of this work and describes open possibilities.

1 Contributions, thesis statement and research questions

The main contributions of this work are, in no specific order:

- **The first exploration of plot generation in data-driven procedurally generated games**. Our *Data Adventures* series of games are the first data games to focus on plot generation and narrative.
- **The introduction and discussion of a data-driven game design process, and the introduction of two dimensions of this design process**. We described two dimensions of data-driven game design (Data Fidelity versus Data Transformation and Functional versus Decorative), and discussed how data-driven games can be designed for different purposes, as well as issues that may arise in designing games like these.

- **An exploration of the two dimensions proposed through various iterations of an adventure data game**. Our strategy map generation and our iterations of the *Data Adventures* series explore various parts of the dimensions we proposed.

- **The description and results of an exploratory user study on how players perceive and engage with a narrative-focused data-driven procedurally generated game**. The user study performed with 30 participants playing *DATA Agent* helped us shed a light on how users engage with and perceive an adventure data game.

Furthermore, Section 1 introduced the following thesis statement:

> ***It is possible to generate engaging playable adventure games from open data.***

This thesis statement influenced our work, turning its focus on data-driven adventure games. The results of the user study presented in Chapter 8 indicate that this hypothesis is indeed true, as players found *DATA Agent*, the last game developed during this work, to be both an adventure game (in their definition of what an adventure game is) and fun, in relation to other adventure/role-playing games played by them.

Additionally, Section 1 also raised several research questions, which we answered throughout this work, summarized as follows:

Q1) How can we computationally generate games from open data?

Chapters 3, 4, 5, 6, 7, 9 attempted to answer this question, both in a broader sense (Chapters 3, 9) and with specific examples via projects (Chapters 4, 5, 6, 7). Chapter 4 presents a method to content for a strategy game using topological information, while Chapters 5, 6, 7 and 9 use several different sources of data to automatically generate adventures for adventure point-and-click games. The methods described in these chapters cover not only the general pipeline for computationally generating game content from open data, but also specify how to acquire the data from specific sources and how acquired data is algorithmically selected and transformed using artificial intelligence.

Throughout this thesis, we explained the different purposes a game that uses open data can have (Chapter 9), such as, for example, learning, opinion and critique and entertainment. Given one or more of this purposes, it is possible to transform data in a more faithful manner, such as in our map generator for FreeCiv, or in more abstract ways, as in the Data Adventures series. The generated content, then, can directly affect the mechanics of the game and the way players engage with it or serve as mere decoration. Our map generator created a database of images via a human-assisted tool and image processing, that it then used to evolve maps for a open-source strategy game. Data Adventures increased the amount of data sources, adding Wikipedia, DBpedia and Wikimedia Commons to the mix to generate linear adventure games. It selected articles that connected two people by querying DBpedia and sorting paths based on their diversity and length, transforming the selected articles into characters, places and locations. *WikiMystery* built on this to generate a branching murder story, adding an evolutionary algorithm capable of selecting suspects related to an Wikipedia article, breadth-first variants for the placement of puzzles, and improving on several fronts. Finally, *Data Agent* added a meta narrative to the game, improved both the suspect selection and the path querying system, and added grammar-based dialogue with *Tracery*.

The variety in our projects also let us explore different facets of data-driven content generation. We showed how to transform demographic and topological information in content for a strategy games, using evolutionary algorithms and image processing in an user-assisted tool. Parts of these data was transformed in different ways, in addition to other information, in content for adventure games. The change in genre allowed to highlight other aspects of the original information.

In sum, we can, by gathering information from sources such as Wikipedia and using AI methods such as evolutionary computation, automatically create games based on the data.

Q2) How do people experience adventure games generated from open data?

Chapter 8 described a user study performed with 30 participants playing *Data Agent*. Participants found the adventures to be fun to play, relative to other adventure/role-playing

games they have played. They also thought that finding the culprit in the adventures' murder cases was straightforward and sensible to them, and would play other adventures if available. It is interesting that, while participants indicated they would not purchase the game (if it was released commercially), they would recommend it to their friends. This might indicate they thought that the interaction (the game) was interesting, but not as a recurring experience, or might be the result of personal preferences. Additionally, some participants that did not play games often had more engaged responses to the game. In unofficial playtests, participants would often play in groups, sharing elaborate meta-stories on what they believed was happening in the game and the NPCs motives and beliefs. While the narrative was often wrong, they seemed to enjoy themselves much more than the user study participants, which we believe is dues to their shared experience. Furthermore, players' opinions on the game were much higher once they were told that the system was automatic and the design input was minimal. They were also more forgiving and interested in the adventures when told so. However, although we believe players enjoyed their interaction with the game, it is hard to generalize this findings. Our user study was limited to a single genre of data-driven game, and it is possible that other styles of games or other types of generated content provide better – or worse – experiences.

Therefore, people can enjoy generated data games and appreciate how the information is presented, but the adventures may not offer any replayable value.

Q3) What are the challenges in curating open data for procedural content generated in games?

While most of this thesis briefly discusses these challenges (Chapters 4, 5, 6, 7), this question is tackled in more detail in Chapter 9. Our map generation tool showed us the problem of lacking a specific data source to gather from, requiring a intermediary human in the process to overcome missing a unified source of information. The *Data Adventures* series, on the other hand, challenged us with the juxtaposition of different data sources and formats. How to automatically select parts of the large amount of data that make would allow for interesting content to be generated, or even sensible content, a problem we faced when

trying to generate adventures from *Wikipedia* (Chapters 5, 6 and 7). *Data Adventures*, the game, tackle this challenge through the use of diversity metrics, in order to find paths of articles that would be varied and interesting. *WikiMystery*, on the other hand, expanded this by adding diversity and solvability of group of suspects, which led to larger games and a bigger space to be curated. This space proved to too large, in fact, and we believed that *Data Agent*'s approach to the problem provided the best solution: a smaller set of suspects evolved by maximizing the relations to the victim and the other suspects. This required, however, that the game design be simplified, removing mechanics previously present in *WikiMystery*. Furthermore, perhaps the most difficult challenge is how to avoid the creation of offensive content created by mixing data that, when separate, is inoffensive. The map generation tool and the first *Data Adventures* have few misrepresentations, which come from the original data being incorrect or slightly mismatched when juxtaposed, but *WikiMystery* and *Data Agent* had the potential of selecting information that was highly controversial or insensitive in the right context, such as, for example, the case of an NPC based on Adolf Hitler suspecting another NPC for being jewish.

In short, at times the data themselves are wrong, and even data that is correct can be misrepresented in some ways, when juxtaposed with conflicting or problematic data.

Q4) How is data (mis)represented in generated games?

This question is answered throughout the whole thesis, with different examples appearing in different projects. Misrepresentation could be generated in result of juxtaposing data (as with the case of the offensive content being generated from mixing two different contexts), errors in the original data (wrong coordinates or categorizations) or artistic liberties taken when designing the game (e.g. creating a fictional murder). Our map generator tool had fewer misrepresentations, as it was more faithful to the data than any other project, and the data itself had few inconsistencies or errors. The worst misrepresentation that could occur in that case was resource maps being slightly off-put. *Data Adventures* (the game) also had few misrepresentations, brought from the small changes made when designing the game, such as the fact that putting two names of people that did not live in the same period as input yield

a game where both were alive simultaneously, but also from errors in the data, as was the case from wrong information in DBpedia dataset that lead to places being added to wrong geographic coordinates or articles transformed into NPCs instead of places, among others. *WikiMystery* increased the misrepresentation derived from the game designed when it further transformed the data, now implying a fictional murder and fictional suspects, which was also present in *Data Agent*.

Thus, data can be misrepresented by generating content from juxtaposing information or transforming the data in order to better fit the game's design.

Q5) What can we do about misrepresentation of data in automatically generated games?

Similar to **Q4**, this question is answered throughout the thesis with examples. Some of the misrepresentation can be ignored if it doesn't significantly affect the final product or is not offensive in some way, as was the case of our map generation tool for *FreeCiv*. The most significant case is the evolution of the *Data Adventures* series, as the game had several more misrepresentations due to wrong data in the first installments of the series. While human curation would solve this problem, we wanted to use minimal human input. One of our solutions was to improve the selection of data, decreasing inconsistencies, and improving the algorithms used to acquire this data. In *Data Adventures*, the game, we only used *Spritely* with simple searches for image gathering, which proved to be insufficient and result in bad matches of images and characters than in later games. We improved on this search on *WikiMystery* and *Data Agent*, creating better searches and going as far as using images from the Wikipedia article itself, not relying just on Wikimedia Commons. *Data Agent* also used more precise querying algorithms to create the initial pool of suspects, resulting in a smaller pool, less algorithmic time and suspects that appear to be more related. Even so, the game could generate insensitive results through juxtaposing content, and we believe it is impossible to fully prevent this with constructive methods such as those we used. A possible approach would be to add a trained model that could identify sensitive content generated, raising flags and triggering changes in the content.

Therefore, it is possible to ignore minor representations or improve curating algorithms, through stronger restrictions or better classification techniques, to deal with data misrepresentation in data games.

Q6) What are the consequences of computationally juxtaposing data without human curation?

As seen in all chapters, creating content from data without human curation can lead to both interesting insights and offensive or incorrect content. This juxtaposition can benefit the game, adding layers of information that were not planned for (for example, in *WikiMystery*, when using a particular picture to represent a character that should be mourning, but appears smiling in the picture), adding new levels to the narrative of the game. The use of topological data and resource deposit information in our map generation had the potential of teaching players about where certain resources could be acquired in the real world, while also directly affecting the gameplay and what strategies players used. *Data Adventures*, however, added inaccurate information (such as indicating that an NPC lived where he died, which might be untrue) for the sake of gameplay, a trend repeated in all games of the series to varying degrees. In the case of *WikiMystery*, the lack of a human curator and the large amount of possible suspects created set of NPCs that seemed unrelated to the players, a problem countered in *Data Agent* by improving our selection process and decreasing the pool size.

At times, however, juxtaposing data resulted in absurd content, such as pictures and information about characters being mixed and identified by the player as wrong (as in the example of Margareth Thatcher and Aung San Suu Kyi in *Data Adventures*, the game, or the case of Confucius in *Data Agent*). There is a level of absurdity that emerges from this juxtaposition, and whether this is beneficial or not to the game can vary on a case-to-case basis. However, sometimes it is possible to present some of this absurdity in an interesting manner, if the design of the game allows it. Our map generator and the first *Data Adventures* did not allow for a suspension of disbelief, as the game presented the data in a more truthful manner. *WikiMystery*, on the other hand, introduced the murder narrative, more clearly moving away from the real data and asking players to engage with a more

absurd setting. *DATA Agent* pushes this even further, trying to explain errors sometimes created from juxtaposing data with a time-travel narrative. Additionally, some players are willing to ignore weird juxtapositions when presented with a fully automated game designer or engaging in the game as a group.

In sum, juxtaposing information can lead to interesting or absurd content creation, which may improve or hurt the game depending on how offensive the content generated is and how aware of the inner workings of the generator players are.

2 Limitations

This section discusses the limitations, which emerged from this work, related to design decisions and implementation of specific algorithms.

A part of the generated content presents errors when compared its real-world counterpart. Part of this problem came from assumptions made when designing and developing the games. For example, when we designed the first *Data Adventures* game, we assumed that places where people resided in, were born in or died in would fit well as houses for the NPCs that represented them, but it is fairly possible these people were born in places that had little significance in their lives (e.g. Freddie Mercury was born in Tanzania, but it makes more sense to have him appear in the United Kingdom instead). On the other hand, we encountered several instances of mistakes in the original data that lead to information being wrongly translated, as our systems were not prepared to deal with that much noise.

Our user study focus on engagement of players when playing *DATA Agent* but our findings may not be generalizable to other adventure games. It also never measure how much can players learn from this game, as we are still unsure of how to measure learnability in the game. Measuring how much the players can learn from the original data is a hard problem, as it must be done while trying not to bias the players.

Another limitation of this work is the type of adventure game we used. Murder mysteries are only a slice on a far larger genre, and we barely scrapped the surface of the genre. It is possible that, as more research is done in this field, some of our findings prove to be more specific to murder mystery adventure games, and less generalizable to other adventure

games. Furthermore, we ignored part of the content that could be generated in these stories, such as murder weapons or puzzle mechanics created from the actual data.

3 Future Work

This section highlights possibilities we see for future work. Some of these opportunities emerge from limitation described previously.

We only explored four points in the space delimited by the two dimensions proposed in this work. However, there are large areas still unexplored, specially in the quadrant representing the intersection of *Functional* and *Data Transformation*. We believe further work, especially related to the genres of role-playing, strategy or physics-based games, could lead to exciting new findings in this space. The generation of, for example, skill trees in a role-playing game from open data could lead to content that directly affects the mechanics of the game and is highly functional, but also has to be less faithful to the data in order to create an engaging and balanced experience. Furthermore, in the *Data Adventures* series, we explored different parts of the space with similar games, but it would be interesting to analyze how different types of data affect content generated for the a same game, in variations of this game. Additionally, the two dimensions hereby proposed are new and it is possible that there are other dimensions we have yet to identify. Thus, another possibility is to develop a more detailed and deeper taxonomy of this design space.

Focusing only on the adventure aspect of this work, it is possible to notice that the *Data Adventures* series does not employ some narrative aspects such as misdirection and motivation (for the murder mystery settings). It would be interesting to explore how to generate misdirection in the story, perhaps by analyzing the final story graph and add or remove information in specific areas, encouraging certain player behaviors and specific encounters. Another possibility is to generate the story without knowledge of the data, using generative methods for narrative design, then attempt to fit the data into the created story. That might lead to more coherent narratives and interesting juxtapositions of characters, settings and props. However, it is also possible that murder mystery stories are less suited for data-driven design than other adventure genres.

Player engagement and perception of data-driven games also needs to be further tested, as our experiments only focus on a specific genre of data games (i.e. adventure data games). Furthermore, there are several other subgenres of adventure games that our work doesn't begin to interact, such as interactive fiction, in addition to several other game genres that haven't been explored yet.

Regarding the problem of curating the data itself, we focused on evolutionary computation to curate data, but different techniques can be applied to this problem. We believe that machine learning in particular would be an interesting approach: it is possible to train a model to identify when and where the content generated is undesirable, and use this trained model during the generation process to automatically curate the data.

Data-driven content generation for games also opens the possibility of debbuging the data through gameplay. While we never explored this idea, it would be possible to allow players to pinpoint data that they know is incorrect, such as a the city of Oregon being placed in the middle of the sea. Even without players, the algorithms used to procedurally generate content could pinpoint discrepancies in the data they use. For example, in the case above more than one geographic location is available, and they are too distant for it to be physically possible. This situation could be flagged by the system and late verified by a human. This could help clean and fix large data sources, where manual fixing can be difficult and boring.

Finally, this dissertation doesn't focus on one of the major strengths of data-driven games: the possibility of learning the underlying data in a fun, playful way. While we believe that players learned facts from our projects, we cannot properly measure what or how much they learned. The use of data-driven games for educational purposes is very much still an open problem.

4 Summary

This chapter brings this dissertation to an end, discussing the contributions and limitations of this work towards future research. In this work, we discussed the space of data-driven game design, and introduced two dimensions used to map this design space. We also described four different projects that used data in various ways to generate content. Three of these

projects were part of a series of adventure games called *Data Adventures*, consisting of *Data Adventures*, *WikiMystery* and *DATA Agent*. These games built from one another to explore different parts of the space of data-driven game design, focusing on its narrative aspect.

Throughout the iterations of the projects described in this thesis we encountered answers to questions before unanswered, in addition to several new problems. Perhaps the most intriguing one consist of the issues that emerge from juxtaposing data to generate new content. However, there are several other game genres where data hasn't been used to procedurally generate content for, and where data games are still an unexplored field full of possibilities.

Bibliography

[1] Freeciv. https://play.freeciv.org/.

[2] Freecivwikia. http://freeciv.wikia.com/.

[3] Space Cowboys . T.I.M.E. Stories, 2015.

[4] 17-bit. Galak-Z, 2016.

[5] Acornsoft. Elite, 1984.

[6] David Adams. Automatic generation of dungeons for computer games. Bachelor's thesis, University of Sheffield, UK, 2002.

[7] Zach Aikman. Galak-Z: Forever: Building Space-Dungeons Organically. https://www.gdcvault.com/play/1021999/Galak-Z-Forever-Building-Space, 2015. Game Developers Conference.

[8] Electronic Arts. Spore, 2008.

[9] Daniel Ashlock. Automatic generation of game elements via evolution. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 289–296. IEEE, 2010.

[10] Daniel A. Ashlock, Stephen P. Gent, and Kenneth Mark Bryden. Evolution of l-systems for compact virtual landscape generation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 2760–2767. IEEE, 2005.

[11] Atari. Asteroids, 1979.

[12] Atari. Pac-Man, 1980.

[13] Atlas Games. Gloom, 2004.

[14] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, pages 722–735. Springer, 2007.

[15] Byung-Chull Bae and R Michael Young. A computational model of narrative generation for surprise arousal. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):131–143, 2014.

[16] Mieke Bal. *Narratology: Introduction to the theory of narrative*. University of Toronto Press, 2009.

[17] Patel Janakkumar Baldevbhai and R. S. Anand. Color image segmentation for medical images using l* a* b* color space. *Journal of Electronics and Communication Engineering*, 1(2), 2012.

[18] Heather Barber and Daniel Kudenko. Generation of dilemma-based interactive narratives with a changeable story goal. In *Proceedings of the 2nd International Conference on Intelligent Technologies for Interactive Entertainment*, INTETAIN '08, pages 1–10, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] Gabriella A. B. Barros and Julian Togelius. Exploring a large space of small games. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–2. IEEE, 2014.

[20] Leandro Motta Barros and Soraia Raupp Musse. Towards consistency in interactive storytelling: Tension arcs and dead-ends. *Computers in Entertainment (CIE)*, 6(3):43, 2008.

[21] Bay 12 Games. Dwarf fortress, 2006.

[22] Farès Belhadj. Terrain modeling: a constrained fractal model. In *Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 197–204. ACM, 2007.

[23] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic syntax. Technical report, MIT/LCS, U.C. Irvine, Xerox Corporation, 2005.

[24] Bethesda. Skyrim, 2011.

[25] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data — the story so far. *International journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[26] Blizzard. Diablo, 1996.

[27] Blizzard. World of warcraft, 2004.

[28] Blizzard. Hearthstone, 2014.

[29] Margaret Boden. *The Creative Mind*. Abacus, London, 1992.

[30] Selmer Bringsjord and David Ferrucci. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press, 1999.

[31] Cameron Browne. Yavalath. In *Evolutionary Game Design*, pages 75–85. Springer, 2011.

[32] Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

[33] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[34] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.

[35] Eric Butler, Adam M. Smith, Yun-En Liu, and Zoran Popovic. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 377–386. ACM, 2013.

[36] Joseph Campbell. *The hero with a thousand faces*, volume 17. Princeton University Press, 1949.

[37] Capcom. Phoenix Wright: Ace Attorney, 2001.

[38] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th Conference on Genetic and Evolutionary Computation*, pages 395–402. ACM, 2011.

[39] Andrew Borg Cardona, Aske Walter Hansen, Julian Togelius, and Marie Gustafsson Friberger. Open Trumps, a data game. 2014.

[40] Beth Cavallari, John Heldberg, and Barry Harper. Adventure games in education: A review. *Australasian Journal of Educational Technology*, 8(2), 1992.

[41] Ronan Champagnat, Guylain Delmas, and Michel Augeraud. A storytelling model for educational games: Hero's interactive journey. *International Journal of Technology Enhanced Learning*, 2(1-2):4–20, 2010.

[42] Hsueh-Min Chang and Von-Wun Soo. Planning-based narrative generation in simulated game universes. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(3):200–213, 2009.

[43] Fred Charles, Julie Porteous, Marc Cavazza, and Jonathan Teutenberg. Timeline-based navigation for interactive narratives. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, pages 1–37. ACM, 2011.

[44] Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.

[45] Chunsoft. Zero Escape: Virtue's Last Reward, 2012.

[46] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[47] Benjamin N Colby. A partial grammar of eskimo folktales. *American Anthropologist*, 75(3):645–662, 1973.

[48] Kate Compton, Benjamin Filstrup, and Michael Mateas. Tracery: Approachable story grammar authoring for casual users. In *Proceedings of the 7th Intelligent Narrative Technologies Workshop*, pages 64–67, 2014.

[49] Kate Compton, Ben Kybartas, and Michael Mateas. Tracery: an author-focused generative text tool. In *Proceedings of the International Conference on Interactive Digital Storytelling*, pages 154–161. Springer, 2015.

[50] Michael Cook and Simon Colton. Multi-faceted evolution of simple arcade games. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 289–296. IEEE, 2011.

[51] Michael Cook and Simon Colton. A Rogue Dream: Automatically generating meaningful content for games. In *Proceedings of the AIIDE Workshop on Experimental AI and Games*, 2014.

[52] Michael Cook and Simon Colton. Ludus ex machina: Building a 3d game designer that competes alongside humans. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 54–62, 2014.

[53] Michael Cook, Simon Colton, and Jeremy Gow. The angelina videogame design system — part i. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(2):192–203, 2017.

[54] Michael Cook, Simon Colton, and Jeremy Gow. The angelina videogame design system — part ii. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):254–266, 2017.

[55] Michael Cook, Simon Colton, and Alison Pease. Aesthetic considerations for automated platformer design. In *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*, 2012.

[56] Daggermouth. Turf Wars, 2007.

[57] Steve Dahlskog, Julian Togelius, and Mark J. Nelson. Linear levels through n-grams. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*, pages 200–206. ACM, 2014.

[58] Charles Darwin. *The Origin of Species by means of Natural Selection*. 1859.

[59] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[60] Chris Dede. Immersive interfaces for engagement and learning. *Science*, 323(5910):66–69, 2009.

[61] Belén Díaz-Agudo, Pablo Gervás, and Federico Peinado. A case based reasoning approach to story plot generation. In *European Conference on Case-Based Reasoning*, pages 142–156. Springer, 2004.

[62] Domo. Data Never Sleeps 5.0. `https://www.domo.com/learn/data-never-sleeps-5`, 2017.

[63] Don Worth. Beneath the Apple Manor, 1978.

[64] Jonathon Doran and Ian Parberry. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):111–119, 2010.

[65] Joris Dormans. Adventures in level design: generating missions and spaces for action adventure games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–8. ACM, 2010.

[66] Dubreq. Top Trumps, 1976.

[67] Marc Ebner, John Levine, Simon M. Lucas, Tom Schaul, Tommy Thompson, and Julian Togelius. Towards a Video Game Description Language. In Simon M. Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius, editors, *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 85–100. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013.

[68] Edmund McMillen. The Binding of Isaac, 2011.

[69] Eidos Interactive. Tomb Raider, 1996.

[70] Richard Evans and Emily Short. Versu — a simulationist storytelling system. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):113–130, 2014.

[71] Clara Fernández-Vara. From open mailbox to context mechanics: shifting levels of abstraction in adventure games. In *Proceedings of the 6th International Conference on Foundations of Digital Games (FDG)*, pages 131–138. ACM, 2011.

[72] Luciano Floridi. *Information: A very short introduction*. OUP Oxford, 2010.

[73] Jose M Font, Tobias Mahlmann, Daniel Manrique, and Julian Togelius. A card game description language. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications*, pages 254–263. Springer, 2013.

[74] Jose M. Font, Tobias Mahlmann, Daniel Manrique, and Julian Togelius. Towards the automatic generation of card games through grammar-guided genetic programming. In *Proceedings of the 8th International Conference on the Foundations of Digital Games (FDG)*, pages 360–363, 2013.

[75] Miguel Frade, Francisco Fernandez De Vega, and Carlos Cotta. Modelling video games' landscapes by means of genetic terrain programming-a new approach for improving users' experience. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, pages 485–490. Springer, 2008.

[76] Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. Evolution of artificial terrains for video games based on accessibility. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, pages 90–99. Springer, 2010.

[77] Miguel Frade, Francisco Fernandez de Vega, and Carlos Cotta. Automatic evolution of programs for procedural generation of terrains for video games. *Soft Computing*, 16(11):1893–1914, 2012.

[78] Marie Gustafsson Friberger and Julian Togelius. Generating game content from open data. In *Proceedings of the 7th International Conference on the Foundations of Digital Games (FDG)*, pages 290–291. ACM, 2012.

[79] Marie Gustafsson Friberger and Julian Togelius. Generating interesting monopoly boards from open data. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 288–295. IEEE, 2012.

[80] Marie Gustafsson Friberger, Julian Togelius, A Borg Cardona, Michele Ermacora, Anders Mousten, M Møller Jensen, V Tanase, and Ulrik Brøndsted. Data games. In *Proocedings of the 4th Workshop on Procedural Content Generation*, pages 1–8. ACM, 2013.

[81] Gearbox Software. Borderlands, 2009.

[82] Gearbox Software. Inside the Box: The Borderlands 2 Loot System. `http://www.gearboxsoftware.com/2013/09/inside-the-box-the-borderlands-2-loot-system/`, 2013.

[83] Pablo Gervás. Propp's morphology of the folk tale as a grammar for generation. In *Proceedings of the 2013 Workshop on Computational Models of Narrative*. Dagstuhl, Germany, 2013.

[84] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on cbr. *Knowledge-Based Systems*, 18(4-5):235–242, 2005.

[85] Jeremy Gow and Joseph Corneli. Towards generating novel games using conceptual blending. In *Proceedings of the 11th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 15–21, 2015.

[86] Groundspeak. Geocaching, 2000.

[87] Mordechai Haklay and Patrick Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[88] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[89] Hasbro. Monopoly, 1935.

[90] Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Evolving content in the Galactic Arms Race video game. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 241–248. IEEE, 2009.

[91] Philipp Heim, Steffen Lohmann, and Timo Stegemann. Interactive relationship discovery via the semantic web. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 303–317, Berlin/Heidelberg, 2010. Springer.

[92] Hello Games. No man's sky, 2016.

[93] Bobby Hoffman and Louis Nadelson. Motivational engagement and video gaming: a mixed methods study. *Educational Technology Research and Development*, 58(3), 2010.

[94] Vincent Hom and Joe Marks. Automatic design of balanced board games. In *Proceedings of the 3rd AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 25–30, 2007.

[95] Amy K. Hoover, Julian Togelius, and Georgios N. Yannakis. Composing video game levels with music metaphors through functional scaffolding. In *Proceedings of the First Computational Creativity and Games Workshop*, pages 1–7, 2015.

[96] Robin Hunicke, Marc Leblanc, and Robert Zubek. MDA: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on the Challenges in Games AI*, 2004.

[97] Internet Live Stats. Total number of websites. `http://www.internetlivestats.com/total-number-of-websites/`, 2017.

[98] It's Alive! Botfighters, 2001.

[99] David Jaffe. Orchestrating the chimera-musical hybrids, technology, and the development of a 'maximalist' musical style. *Leonardo Music Journal*, 5, 1995.

[100] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. Polymorph: A model for dynamic level generation. In *Proceedings of the 6th Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 138–143, 2010.

[101] Lawrence Johnson, Georgios N. Yannakakis, and Julian Togelius. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–4. ACM, 2010.

[102] Jutsu Games. 911 Operator, 2017.

[103] K Raiyan Kamal and Yusuf Sarwar Uddin. Parametrically controlled terrain generation. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 17–23. ACM, 2007.

[104] Rilla Khaled, Mark J. Nelson, and Pippin Barr. Design metaphors for procedural content generation in games. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1509–1518, 2013.

[105] Ahmed Khalifa, Michael Cerny Green, Diego Perez-Liebana, and Julian Togelius. General video game rule generation. In *Proceedings of the 2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 170–177. IEEE, 2017.

[106] Steven Orla Kimbrough, Gary J Koehler, Ming Lu, and David Harlan Wood. On a feasible–infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*, 190(2):310–327, 2008.

[107] Rob Kitchin. *The Data Revolution: Big Data, Open Data, Data Infrastructures & Their Consequences*. Sage, 2014.

[108] Sheldon Klein, J. Aeschlimann, D Balsiger, Steven L. Converse, Claudine Court, Mark Foster, Robin Lao, John D. Oakley, and Joel Smith. Automatic novel writing: A status report. Technical report, University of Wisconsin-Madison, 1973.

[109] Ben Kybartas and Rafael Bidarra. A survey on story generation techniques for authoring computational narratives. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(3):239–253, 2017.

[110] Ben Kybartas and Clark Verbrugge. Analysis of regen as a graph-rewriting system for quest generation. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):228–242, 2014.

[111] George Lakoff. Structural complexity in fairy tales. *The Study of Man*, 1:128–150, 1972.

[112] Stephen Lavelle. Puzzlescript. `http://puzzlescript.net`, 2013.

[113] Michael Lebowitz. Story-telling as planning and learning. *Poetics*, 14(6):483–502, 1985.

[114] Boyang Li. *Learning knowledge to support domain-independent narrative intelligence*. PhD thesis, Georgia Institute of Technology, 2015.

[115] Boyang Li, Stephen Lee-Urban, and Mark Riedl. Crowdsourcing interactive fiction games. In *Proceedings of the 8th International Conference on the Foundations of Digital Games (FDG)*, pages 431–432, 2013.

[116] Antonios Liapis. Creativity facet orchestration: the whys and the hows. In Simon M. Lucas, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius, editors, *Dagstuhl Reports*, volume 5. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[117] Antonios Liapis, Héctor Pérez Martínez, Julian Togelius, and Georgios N. Yannakakis. Transforming exploratory creativity with delenox. In *Proceedings of the 4th International Conference on Computational Creativity*, pages 56–63, 2013.

[118] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Sentient Sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th International Conference on Foundations of Digital Games (FDG)*, pages 213–220, 2013.

[119] Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Computational game creativity. In *Proceedings of the International Conference on Computational Creativity (ICCC)*, pages 46–53, 2014.

[120] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. Designer modeling for Sentient Sketchbook. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2014.

[121] Libellud. Mysterium, 2015.

[122] Bruce Lindbloom. Delta e (cie 1994). `http://www.brucelindbloom.com`, 1994.

[123] Aristid Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

[124] LucasArts. Indiana Jones and the Fate of Atlantis, 1992.

[125] LucasArts. Day of the Tentacle, 1993.

[126] Daniel Mackay. *The fantasy role-playing game: A new performing art*. McFarland & Company, 1974.

[127] Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game developers conference*, volume 2, pages 4–8, 2003.

[128] Michael Mateas and Andrew Stern. Procedural authorship: A case-study of the interactive drama façade. *Digital Arts and Culture (DAC)*, page 27, 2005.

[129] Peter Mawhorter and Michael Mateas. Procedural level generation using occupancy-regulated extension. In *Proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 351–358. IEEE, 2010.

[130] Josh McCoy, Mike Treanor, Ben Samuel, Aaron Reed, Noah Wardrip-Fruin, and Michael Mateas. Prom week: Social physics as gameplay. In *Proceedings of the 6th International Conference on the Foundations of Digital Games (FDG)*, pages 1–8, 2011.

[131] Josh McCoy, Mike Treanor, Ben Samuel, Brandon Tearse, Michael Mateas, and Noah Wardrip-Fruin. Authoring game-based interactive narrative using social games and comme il faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization: Archive & Innovate*, pages 1–8, 2010.

[132] Neil McIntyre and Mirella Lapata. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572. Association for Computational Linguistics, 2010.

[133] James R. Meehan. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, volume 77, pages 91–98, 1977.

[134] MicroProse. Sid meyer's civiliation, 1991.

[135] Gavin SP Miller. The definition and rendering of terrain maps. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 39–48. ACM, 1986.

[136] Mojang. Minecraft, 2011.

[137] Nick Montfort. *Twisty Little Passages: an approach to interactive fiction*. Mit Press, 2005.

[138] Mossmouth Games. Spelunky, 2009.

[139] Malik Nairat, Palle Dahlstedt, and Mats G. Nordahl. Character evolution approach to generative storytelling. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1258–1263. IEEE, 2011.

[140] Malik Nairat, Palle Dahlstedt, and Mats G. Nordahl. Story characterization using interactive evolution in a multi-agent system. In *Proceedings of the International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 168–179. Springer, 2013.

[141] Andy Nealen, Adam Saltsman, and Eddy Boxerman. Towards minimalist game design. In *Proceedings of the 6th International Conference on the Foundations of Digital Games (FDG)*, pages 38–45. ACM, 2011.

[142] Newsgaming. September 12th: A toy world, 2003.

[143] Niantic. Ingress, 2004.

[144] Niantic. Pokemon Go, 2016.

[145] Thorbjørn S. Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J. Nelson. General video game evaluation using relative algorithm performance profiles. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, pages 369–380. Springer, 2015.

[146] Thorbjørn S Nielsen, Gabriella A. B. Barros, Julian Togelius, and Mark J. Nelson. Towards generating arcade game rules with VGDL. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 185–192. IEEE, 2015.

[147] Nintendo. Super Mario Bros, 1985.

[148] Nintendo. The Legend of Zelda, 1986.

[149] Peter Thorup Ølsted, Benjamin Ma, and Sebastian Risi. Interactive evolution of levels for a competitive multiplayer FPS. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 1527–1534. IEEE, 2015.

[150] Michael O'Neill and Anthony Brabazon. Evolving a logo design using lindenmayer systems, postscript & grammatical evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3788–3794. IEEE, 2008.

[151] Michael O'Neill, John Mark Swafford, James McDermott, Jonathan Byrne, Anthony Brabazon, Elizabeth Shotton, Ciaran McNally, and Martin Hemberg. Shape grammars and grammatical evolution for evolutionary design. In *Proceedings of the 11th Conference on Genetic and Evolutionary Computation*, pages 1035–1042. ACM, 2009.

[152] TeongJoo Ong and John J. Leggett. A genetic algorithm approach to interactive narrative generation. In *Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 181–182. ACM, 2004.

[153] David Oranchak. Evolutionary algorithm for generation of entertaining shinro logic puzzles. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, pages 181–190. Springer, 2010.

[154] Johnathan Pagnutti, Kate Compton, and Jim Whitehead. Do You Like This Art I Made You: Introducing Techne, a creative artbot commune. pages 1–8, 2016.

[155] Jeff Z. Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009.

[156] Alison Pease and Simon Colton. On impact and evaluation in computational creativity: A discussion of the turing test and an alternative proposal. In *Proceedings of the AISB symposium on AI and Philosophy*, 2011.

[157] Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling player experience in super mario bros. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 132–139. IEEE, 2009.

[158] Barney Pell. Metagame in symmetric chess-like games. pages 1–30, 1992.

[159] Lyn Pemberton. A modular approach to story generation. In *Proceedings of the 4th Conference on European chapter of the Association for Computational Linguistics*, pages 217–224. Association for Computational Linguistics, 1989.

[160] Rafael PÉrez Ý PÉrez and Mike Sharples. MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139, 2001.

[161] Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M. Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):229–243, 2016.

[162] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[163] Jan L. Plass, Bruce D. Homer, and Charles K. Kinzer. Foundations of game-based learning. *Educational Psychologist*, 50(4):258–283, 2015.

[164] Julie Porteous, Marc Cavazza, and Fred Charles. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 1(2):10, 2010.

[165] Mike Preuss, Antonios Liapis, and Julian Togelius. Searching for good and diverse game levels. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2014.

[166] Vladimir Propp. *The morphology of the fairy tale*. University of Texas Press, 1968.

[167] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[168] Mark Riedl, Cesare J. Saretto, and R. Michael Young. Managing interaction between users and agents in a multi-agent storytelling environment. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 741–748. ACM, 2003.

[169] Mark O. Riedl and R. Michael Young. Story planning as exploratory creativity: Techniques for expanding the narrative search space. *New Generation Computing*, 24(3):303–323, 2006.

[170] Mark O. Riedl and Robert Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.

[171] Mark Owen Riedl et al. *Narrative planning: Balancing plot and character*. PhD thesis, North Carolina State University, 2004.

[172] Graeme Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17, 2007.

[173] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[174] T. Scaltsas and C. Alexopoulos. Creating creativity through emotive thinking. In *Proceedings of the World Congress of Philosophy*, 2013.

[175] Jens Schneider, Tobias Boldte, and Rüdiger Westermann. Real-time editing, synthesis, and rendering of infinite landscapes on gpus. In *Proceedings of Vision, modeling, and visualization*, page 145, 2006.

[176] Noor Shaker, Antonios Liapis, Julian Togelius, Ricardo Lopes, and Rafael Bidarra. Constructive generation methods for dungeons and levels. In *Procedural Content Generation in Games*, pages 31–55. Springer, 2016.

[177] Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.

[178] Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. Towards automatic personalized content generation for platform games. In *Proceedings of the 7th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 63–68, 2010.

[179] Noor Shaker, Georgios N. Yannakakis, and Julian Togelius. Crowdsourcing the aesthetics of platform games. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(3):276–290, 2013.

[180] Tanya X. Short and Tarn Adams. *Procedural Generation in Game Design*. CRC Press, 2017.

[181] Miguel Sicart. Loops and metagames: Understanding game design structures. In *Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG)*, 2015.

[182] Deep Silver. Saints Row IV, 2013.

[183] James Skorupski, Lakshmi Jayapalan, Sheena Marquez, and Michael Mateas. Wide ruled: A friendly interface to author-goal based story generation. In *International Conference on Virtual Storytelling*, pages 26–37. Springer, 2007.

[184] James Skorupski and Michael Mateas. Novice-friendly authoring of plan-based interactive storyboards. In *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 174–179, 2010.

[185] Adam M. Smith and Michael Mateas. Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 273–280. IEEE, 2010.

[186] Adam M. Smith and Michael Mateas. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):187–200, 2011.

[187] Gillian Smith, Mike Treanor, Jim Whitehead, and Michael Mateas. Rhythm-based level generation for 2d platformers. In *Proceedings of the 4th International Conference on the Foundations of Digital Games (FDG)*, pages 175–182. ACM, 2009.

[188] Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: A mixed-initiative level design tool. In *Proceedings of the 5th International Conference on the Foundations of Digital Games*, pages 209–216. ACM, 2010.

[189] Gillian Smith, Jim Whitehead, and Michael Mateas. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):201–215, 2011.

[190] Sam Snodgrass and Santiago Ontanón. Controllable procedural content generation via constrained multi-dimensional markov chain sampling. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 780–786, 2016.

[191] Sam Snodgrass and Santiago Ontanón. Player movement models for platformer game level generation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 757–763. AAAI Press, 2017.

[192] Brøderbund Software. Where in the World is Carmen Sandiego?, 1985.

[193] Personal Software. Zork, 1980.

[194] Barry Sookman. The Google Book project: Is it fair use? *Journal of the Copyright Society of the USA*, 61:485–516, 2013.

[195] Nathan Sorenson and Philippe Pasquier. Towards a generic framework for automated video game level creation. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, pages 131–140. Springer, 2010.

[196] Herbert Spencer. *The Principles of Biology*, volume 1–2. New York, 1864.

[197] Anne Sullivan and Anastasia Salter. A taxonomy of narrative-centric board and card games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG)*, page 23. ACM, 2017.

[198] Adam Summerville, Matthew Guzdial, Michael Mateas, and Mark O Riedl. Learning player tailored content from observation: Platformer level generation from video traces using LSTMS. In *Proceedings of the 12th Conference on Artificial Intelligence and Interactive Digital Entertainment (DIGRA)*, pages 107–113, 2016.

[199] Adam Summerville and Michael Mateas. Super Mario as a string: Platformer level generation via LSTMS. *Proceedings of the Conference on Artificial Intelligence and Interactive Digital Entertainment (DIGRA)*, pages 1–16, 2016.

[200] Adam James Summerville, Shweta Philip, and Michael Mateas. MCMCTS PCG 4 SMB: Monte Carlo tree search to guide platformer level generation. In *Proceedings of the 11th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 68–74, 2015.

[201] Ivo Swartjes, Edze Kruizinga, and Mariët Theune. Let's pretend I had a sword. In *Proceedings of the Joint International Conference on Interactive Digital Storytelling*, pages 264–267. Springer, 2008.

[202] Ivo Swartjes and Mariët Theune. A fabula model for emergent narrative. In *Proceedings of the International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 49–60. Springer, 2006.

[203] Terrible Toybox. Thimbleweed Park, 2014.

[204] Thinking Rabbit. Sokoban, 1982.

[205] David Thue, Vadim Bulitko, and Marcia Spetch. Passage: A demonstration of player modeling in interactive storytelling. In *Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 226–227, 2008.

[206] David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive storytelling: A player modelling approach. In *Proceedings of the 3rd Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 43–48, 2007.

[207] TIME. Time 100: Artists and entertainers [special issue]. June 1998.

[208] TIME. Time 100: Builders and titans [special issue]. December 1998.

[209] TIME. Time 100: Leaders and revolutionaries [special issue]. April 1998.

[210] TIME. Time 100: Heroes and icons [special issue]. June 1999.

[211] TIME. Time 100: Scientists and thinkers [special issue]. March 1999.

[212] Julian Togelius, Renzo De Nardi, and Simon M. Lucas. Towards automatic personalised content creation for racing games. In *Proceedings on IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 252–259. IEEE, 2007.

[213] Julian Togelius and Marie Gustafsson Friberger. Bar Chart Ball, a data game. pages 1–2, 2013.

[214] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, and Georgios N Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 265–272. IEEE, 2010.

[215] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, Johan Hagelbäck, Georgios N. Yannakakis, and Corrado Grappiolo. Controllable procedural map generation via multiobjective evolution. *Genetic Programming and Evolvable Machines*, 14(2):245–277, 2013.

[216] Julian Togelius and Jurgen Schmidhuber. An experiment in automatic game design. In *Proceedings of the IEEE Symposium On Computational Intelligence and Games (CIG)*, pages 111–118. IEEE, 2008.

[217] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186, 2011.

[218] Toy and Wichman. Rogue, 1980.

[219] Mike Treanor, Bryan Blackford, Michael Mateas, and Ian Bogost. Game-O-Matic: Generating videogames that represent ideas. In *Proceedings of the 3rd workshop on Procedural Content Generation in Games*, pages 1–11. ACM, 2012.

[220] Mike Treanor, Bobby Schweizer, Ian Bogost, and Michael Mateas. The micro-rhetorics of Game-O-Matic. In *Proceedings of the 7th International Conference on the Foundations of Digital Games (FDG)*, pages 18–25. ACM, 2012.

[221] Scott R. Turner. *Minstrel: a computer model of creativity and storytelling*. PhD thesis, University of California at Los Angeles, 1993.

[222] Adam Vile and Simon Polovina. Thinking of or thinking through diagrams? The case of conceptual graphs. In *Thinking with Diagrams Conference*, 1998.

[223] John Von Neumann. The general and logical theory of automata. *Cerebral mechanisms in behavior*, 1(41):1–2, 1951.

[224] Lev Vygotsky. Interaction between learning and development. *Readings on the development of children*, 23(3), 1978.

[225] Stephen G. Ware, R. Michael Young, Brent Harrison, and David L. Roberts. A computational model of plan-based narrative conflict at the fabula level. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):271–288, 2014.

[226] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 3381–3387. IEEE, 2008.

[227] Wikimedia Foundation. Wikipedia. `https://en.wikipedia.org/`, 2001. [Online; last accessed 10-April-2018].

[228] Will Crowther. Colossal Cave Adventure, 1976.

[229] Wizards of the Coast. Betrayal at House on the Hill, 2010.

[230] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer, 2018. `http://gameaibook.org`.

[231] R. Michael Young. An overview of the mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. In *The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Environment. Technical Report SS-01-02*, pages 77–81, 2001.

[232] Tiago Zaidan and Luıs Fabrício W Góes. Evolvestone: An evolutionary generator of balanced digital collectible card games. In *Proceedings of the 15th Brazilian Symposium on Games (SBGames)*, pages 11–17, 2016.

[233] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.