AUTOMATIC GAME SCRIPT GENERATION

By

Ahmed Abdel Samea Hassan Khalifa

A Thesis Submitted to the
Faculty of Engineeringat Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
JUNE 2015

AUTOMATIC GAME SCRIPT GENERATION

By

Ahmed Abdel Samea Hassan Khalifa

A Thesis Submitted to the
Faculty of Engineeringat Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

Under the Supervision of

Prof. Magda Fayek
Professor of Artificial Intelligence
Computer Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
JUNE 2015

AUTOMATIC GAME SCRIPT GENERATION

By

Ahmed Abdel Samea Hassan Khalifa

A Thesis Submitted to the
Faculty of Engineeringat Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

Approved by the Examining Committee:

Prof. First S. Name, External Examiner

Prof. Second S. Name, Internal Examiner

Prof. Magda Fayek, Thesis Main Advisor

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
JUNE 2015

Engineer's Name:	Ahmed Abdel Samea Hassan Khalifa
Date of Birth:	22/02/1989
Nationality:	Egyptian
E-mail:	amidos2002@hotmail.com
Phone:	+2 0100 671 8789
Address:	20 Mahmoud El-Khayaal St, Haram, 12561 Giza
Registration Date:	1/10/2010
Awarding Date:	14/7/2015
Degree:	Master of Science
Department:	Computer Engineering

Insert photo here

Supervisors:

Prof. Magda Fayek

Examiners:

Prof. First S. Name	(External examiner)
Prof. Second S. Name	(Internal examiner)
Prof. Magda Fayek	(Thesis main advisor)

Title of Thesis:

Automatic Game Script Generation

Key Words:

Procedural Content Generation; General Video Game Playing; Genetic Algorithm; Computational Creativity; Rule Generation; Level Generation; Video Game Description Language

Summary:

Summary

Table of Contents

List of Tables

List of Figures

Acknowledgements

In this section, you may provide acknowledgements to those who gave you support and encouragement to complete your thesis. Acknowledgement of funding from local and international funding agencies must be clearly stated.

Starting from the acknowledgements page, pages are numbered using the Roman numerals i, ii, iii âĂęetc. Starting from Chapter 1, pages must be numbered using Arabic numerals. Page numbers are at the bottom of the page, preferably centered.

Dedication

You may include this section if you wish to dedicate your thesis to someone.

Abstract

This file is provided to help graduate students at the Faculty of Engineering, Cairo University in preparing their theses according to the regulations and format guidelines defined by the graduate committee. Students are required to consult the regulations for thesis preparation available at the department of graduate studies besides using this template.

In this template, different styles are defined which start with âĂIJFECU ThesisâĂİ phrase. You may use these styles to quickly format your text throughout the thesis. You may also change these styles as long as they comply with the regulations for thesis preparation.

Chapter 1: Introduction

Video Games were originally created by small groups of people in their spare time. As time passes Video Games evolved to be a huge multi-billion industry where thousands of people are working everyday to create new games. Automation doesn't play huge role in creating games so most of the work is done by humans. Hiring humans ensures producing of original high quality games but it costs a lot of money and time.

1.1 Motivation

During the early days of Video Games, games were created by a couple of people in their spare time. Most of the time was spent in programming the game, while a small portion was dedicated for graphics, sounds, and music because of the technical limitations of the devices at that time. As these limitation are no more, producing a game takes more time than before. Most of that time is spent on creating content for the game (graphics, music, sounds, levels, and ...etc)[12]; for example creating graphics for a huge main stream game may take hundreds of artist working for a year or two. That is why the production cost of a huge game may reach millions of dollars[7].

That huge production cost caused lots of Game Studios to shut their doors [3] and others became afraid of creating new game ideas. In spite of all these technological advancement, we could not reduce the cost and time for creating good game contents because creation process heavily depends on the creative aspect of the human designer. Automating this creative process is somehow difficult as there is no concrete criteria for judging creativity which raises the main question: Can technology help to reduce the time and money used in producing games?

1.2 Problem Statement

Creating content for games or any creative medium, e.g. Art, Music, Movies, and ...etc, is really a tough problem and take very long time. For example a famous popular game called *Threes* took around 14 month working on the concept itself [10]. Game market is growing fierce as there is always a demand for new games with new contents but that can not be accomplished very fast because creating new innovative games requires long time.

Games consists of lots of aspects ranging from game graphics to game rules. It is hard to generate all these aspects at the same time, so we are going to address just two of them which are Levels and Rules. Level Generation has been in industry for decades since early days of games [1] but it has always used lots of hacks and it has never been generalized. Same applies to Rule Generation except that their is a very small contribution in automatically generating rules.

Complete automation of levels and rules for all types of games seems beyond the reach at the present time, so In this work we focus on automatic generation of Puzzle Games because they are accessible by all players, they do not require much time to be played,

and do not have any boundaries on their ideas. From there, we found an urgent need to research for new ways in Level and Rule Generation for Puzzle Games.

1.3 Objectives

The aim of this work is to try to understand how our brain works in creating creative content, helping game designers and developers to think outside the box, and minimizing their time for searching for new content by providing them with good diverse seeds for levels and rules. It is not intended to create an Artificial Intelligence that can create the whole game from scratch and sell it afterwards because its too early to think that computers can do that on its own and people are not prepared yet to deal with that kind of Intelligence that can replace them at work.

1.4 Organization of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, we explain further the background needed to understand Procedural Content Generation in Video Games, conducted by Chapter 3 explaining any previous work done in that area. In Chapter 4, we show different methodologies and techniques we are using to reach our objectives followed by our results from applying these techniques in Chapter 5. Finally, Chapter 6 presents our conclusion and future work.

Chapter 2: Background

This chapter presents required background information needed to understand this thesis. It starts with explaining Puzzle Genre in Video Games, followed by description of Procedural Content Generation and a special case called Search Based PCG. It is conducted by describing Video Game Description Language and why it is important. Finally, it explains General Video Game Playing and its relation to this research.

2.1 Puzzle Genere in Video Games

In 1950 a small group of academic people started developing Video Games on the main frames of their schools. This has continued to till the 1970s when Video Games reached Main Stream industry. In 1980s lots of technicals problems were solved which brought new types of Video Games. Game Genres were invented to differentiate between different game types so players can refer to their favorite games by genres, for example there is Adventure genre like Legend of Zelda, Fighting genre like Street Fighter, Platformer genre like Super Mario Bros, Shooter genre like Space Invaders, Puzzle Genre like Bomberman and ...etc. In 2000s people started to go toward being mobile and making games for cellular phones and one of the most important genres on these mobile devices is Puzzle Genre[6].

Puzzle Genre is a style of Video Games that emphasize solving puzzles, it always focuses on logical or conceptual challenges. Puzzle games can be described using the following aspects:

- **Graphics:** How the game looks like.
- **Inputs:** How input is given to the game.
- **Rules:** How the game behaves to events.
- **Levels:** How objects are placed.
- **Win Condition:** How to end the game.

2.2 Procedural Content Generation

Procedural Content Generation (PCG) means generating game content using a computer. It was first developed due to technical limitations (small disk space) and the need to provide a huge amount of content. *Akalbeth* was the first game using PCG, the game starts asking the player to provide it with a lucky number that is used as a seed to a random number generator which generates everything in the game from the player statistics to dungeon maps. Due to that the game has a very small footprint around 22 Kb so it does not need to be loaded from lots of disks [1]. Although technical difficulties become history and storage is no longer a problem, PCG is still one of the hot topics in Video Games Industry and Research. PCG helps us reduce development time and cost, be creative, and understand the process of creating game content. PCG can be used to generate different game aspects for example Textures, Sounds, Music, Levels, Rules, and ...etc. In this thesis we will focus only on the Levels and Rules Generation which is the core aspect for any Puzzle Game.

2.2.1 Level Generation

Levels are different combination of game objects. These combinations control the game flow and difficulty. For example in Super Mario, as the game advances the levels become longer and include more enemies and gaps. Level Generation has been in industry since dawn of games in order to decrease the game size, but it is now used to introduce huge amount of levels that humans can not generate manually in reasonable time. Level Generation has always been done for a specific game using lots of hacks to improve the output result. These hacks cause the output levels to follow certain guidelines which may cause elimination of huge amounts of possible levels but on the other hand these guidelines ensure that the output levels are all playable (can reach goal of the game) and satisfactory by all players [4].

2.2.2 Rule Generation

Rules are what governs game behavior. Based on them you can know how the game will be played; for example what to collect, what to avoid, how to kill enemies ...etc. Rules are very different from one game genre to another, for example in Board Games (e.g. Monopoly) all rules are applied after player plays his turn, while in Platformer Games (e.g. Super Mario) the rules are applied with every time step (not depending on the player movement). Due to that huge difference between rules in different genres, Rule Generation is one of the most difficult contents to generate. That is why we need to find a way to represent and describe rules. The only way to represent all different genres is using Coding Languages because Coding is the only common thing between all games. Generating new Rules using Coding Languages may take years because Coding Languages are used to create any computer related software such as plug-ins, programs, viruses, games and ...etc. Still some researchers managed to use it but it will need more guidance due to the huge search space of Coding Languages [13]. A better solution is to use a description language for the rules that is more specific for a certain genre which we are going to discuss in Section 2.4 [21].

2.3 Search Based PCG

There are many approaches to generate content in Video Games. The search based approach is one famous technique that has been used a lot in recent research. The search based approach can be divided into three main parts Search Algorithm, Content Representation, and Evaluation Function [22].

2.3.1 Search Algorithm

This is the *Engine* for generating game content. There are different searching algorithm for example Simulated Annealing, Hill Climbing, and Genetic Algorithm (GA) . GA is a search algorithm based on Darwin Theory *"Survival for the Fittest"*. GA uses techniques inspired from natural evolution such as selection, crossover and mutation in order to find solution to large space optimization problems [5].

2.3.2 Content Representation

There are several ways for modeling different aspect of the game content during the generation process. The way of representing the content can affects a lot on the output of the generation. For example levels may be represented as:[22]

- 2D matrix where each value represent a tile location in the level.
- Sequence of position of objects in level.
- Sequence of level Patterns.
- Level properties such as Number of Objects in map, Number of Players and ...etc.
- Random number seed.

2.3.3 Evaluation Function

Without Evaluation Function the Search Algorithm will be blind. The evaluation function leads the Search Algorithm to find better content in the solution space. The evaluation Function can be either one of the following:[22]

- **Direct evaluation function:** which utilizes some understanding about the generated content and evaluates it accordingly.
- **Simulation-based evaluation function:** Use Artificial Intelligence agent to test the generated content and based on his behavior it estimates its quality.
- **Interactive evaluation function:** Evaluate generated content based on the interaction with human.

2.4 Puzzle Script as Video Game Description Language

Referring to Section 2.2.2 we can not generate game rules without having a methodology to describe it. Video Game Description Language (VGDL) was originally invented to help on the work for General Video Game Playing (discussed later in Section 2.5) at Standford University which will be discussed in the following section. The idea behind VGDL was to provide a simple description language that can be used to describe simple 2D games. Researchers insist that any VGDL language must have the following aspects:[14]

- **Human Readable:** It must be easy for human user to understand games written with it and formulate new one.
- **Unambiguous and Easy to parse:** It must be easy to parse using a computer that is why it must be clear.
- **Searchable:** The games formulated using it can be drawn in form of tree so its easy to use Search Algorithm (discussed in Section 2.3.1) to find new games.
- **Expressive:** It can be used to express more than one game.
- **Extensible:** It can be extended to add more game types and dynamics.
- **Viable for Random Generation:** Each component of the language have 'sensible defaults' so its easier to search for new values for certain components without worrying about other ones.

Puzzle Script (PS) is a VGDL created by Stephan Lavelle to help game designers and developers to create puzzle games [9]. Games generated by PS are time stepped games

similar to Sokoban game; Sokoban is an old Japanese puzzle game where your goal is to push some crates towards certain locations[11]. PS file starts with some meta data like game name, author name, and website then it is divided into 7 sections objects, legend, sounds, collision layers, rules, win conditions, and levels. Figure 2.2 shows an example of the PS file for Sokoban game. It is refereed to it in the next sections.

2.4.1 Objects

It is the first section of the PS file and it contains a list for all the object names and colors being used in the game for example in Figure 2.2 we had Background, Target, Wall, Player, and Crate associated with a color.

2.4.2 Legend

It is a table of one letter abbreviation and one or more group of objects for example in Figure 2.2 the Wall is assigned "#" symbol and Crate and Target together are assigned "@" symbol.

2.4.3 Sounds

It consists of group of game events associated with a certain number. PS engine use these numbers as parameter to BFXR to generate sounds in runtime[2] for example in Figure 2.2 Move event for Crate object is assigned to value 36772507.

2.4.4 Collision Layers

It defines which objects can not exists at same location during running the game, Objects on the same line can not exist together. Each line is considered a new layer, for example in Figure 2.2 Player, Crate, and Wall can not exists at same location as they are assigned to the same layer.

2.4.5 Rules

Its a set of production rules that govern how the game will be played. Production rules have the following formate

Tuple ... Tuple -> Tuple ... Tuple

where the number of tuples on the right must be equal to number of tuples on the left, and each tuple must be in the following format

[Objects | ... | Objects]

Each tuple has a group of Objects separated by "|" symbol which means that these objects are beside each other for example [Player | Crate] means there is a Player and Crate object beside each other. The number of "|" in both left and right side of each tuple must be equal. Objects must be written in the following format

Direction Object ... Direction Object

Object must be selected from the Objects section and must be from different collision layer for example [Player Target] means Player and Target exists on same tile. Last thing is the Direction, it must be one of the following ">", "<", "∧", and "∨". The first one means in same as the moving direction, the second one means opposite to moving direction, while the third means 90° from the moving direction, and the final one is −90° from the moving direction. Let's take a real example on the rule, for example in Figure 2.2 the rule is

[> Player | Crate] -> [> Player | > Crate]

means if there is a Player and Crate beside each other, and the Player moves towards the Crate, then both the Player and the Crate will move in the same direction.

2.4.6 Win Conditions

It identifies when the level should end. It consists of 2 objects separated by on keyword. For example in Figure 2.2 "all Target on Crate" means that the level ends when all Targets are on same location with Crates. The possible keywords that can be used are "all", "some", and "no". "all" means every single object must be at same location, while "some" means at least one object at same location, but "no" means that the two objects must not be at same location.

2.4.7 Levels

They are 2D matrices showing the current configuration for each game level using the abbreviations found in the Legends section. Each level is separated from the next level by a new line. In Figure 2.2 you can see 3 designed Sokoban levels. By replacing each symbol by the corresponding object from the Legend section then the color from Objects section, so you will get the following levels in Figure 2.1.

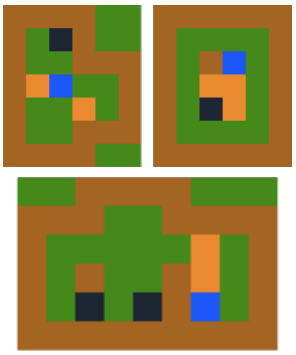
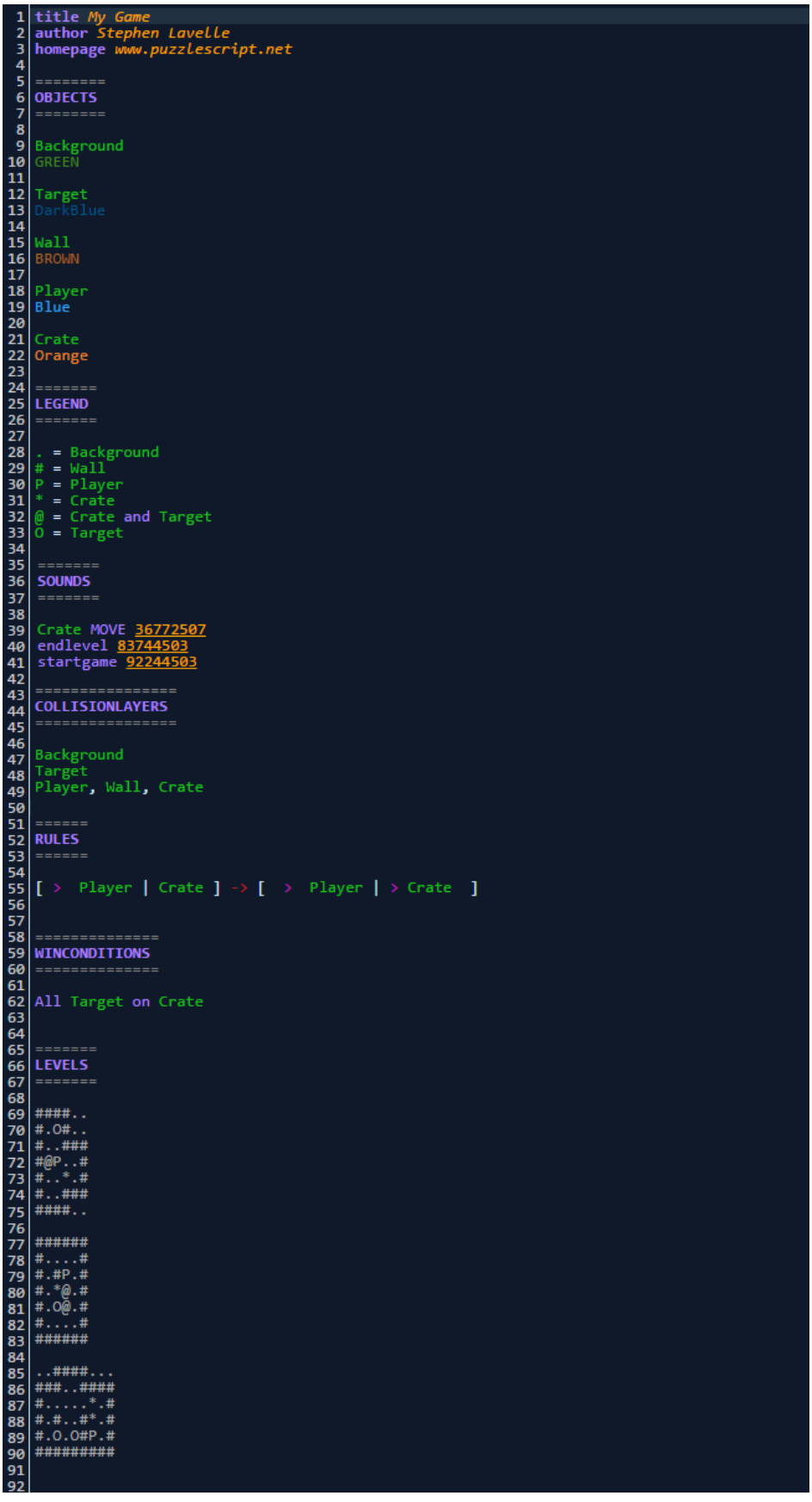


Figure 2.1: Puzzle Script Levels



```
title My Game
author Stephen Lavelle
homepage www.puzzlescript.net

========
OBJECTS
========

Background
GREEN

Target
DarkBlue

Wall
BROWN

Player
Blue

Crate
Orange

=======
LEGEND
=======

. = Background
# = Wall
P = Player
* = Crate
@ = Crate and Target
O = Target

=======
SOUNDS
=======

Crate MOVE 36772507
endlevel 83744503
startgame 92244503

================
COLLISIONLAYERS
================

Background
Target
Player, Wall, Crate

======
RULES
======

[ >  Player | Crate ] -> [  >  Player | > Crate  ]


==============
WINCONDITIONS
==============

All Target on Crate


=======
LEVELS
=======

####..
#.O#..
#..###
#@P..#
#..*.#
#..###
####..

######
#....#
#.#P.#
#.*@.#
#.O@.#
#....#
######

..####...
###..####
#.....*.#
#.#..#*.#
#.O.O#P.#
#########
```

Figure 2.2: Puzzle Script File

2.5 General Video Game Playing

General Video Game Playing (GVGP) is an area in Artifical Intelligence (AI) where people try to develop an intelligent agent that is able to play more than one game. Most of Game Playing Agents are not general players, they are designed for a certain game like Chess, Poker, Backgammon, or ...etc. At the beginning the AI is supported with game rules in form of VGDL Section 2.4 with current game state and possible actions and the agent should choose the next action. After several play-outs the agent should learn how to maximize his score or reach goal and avoid death. Researchers used different methods to try to create general AI using Monte Carlo Tree Search, Neural Networks, and Reinforcement Learning methods. GVGP is very important for PCG specially for rule and level creation because in order to generate good random level or rule, it is better to test it out using a GVGP Agent to have a better judgment on how the random content will be played.[15]

Chapter 3: Literature Review

This chapter will provide a review of the past work on Procedural Content Generation. It will highlight different efforts towards generating levels and rules for games.

3.1 Level Generation

This section will present some of the work related to level generation from academia and industry. Most of previous work didn't talk a lot about generating levels for Puzzle Games but it focus more on Platform genre and Arcade games. The work found about Puzzle Games was limited toward certain games such as Sokoban [20, 16], Cut The Rope [19], and Fruit Dating [8]. Even the generic work on Puzzle Games need prior human understanding of the game rules [18, 17].

Generating levels for Puzzle Games seems interesting but it has lots of problems. Levels must have at least one solution. Every element in the level should be used, adding unused items is a bad design. There is no rules govern Puzzle Games ideas which is different from FPS Genre or any other genre. For example some games may have continuous space where all game actions depend on player skills (Angry Birds, Cut the Rope, and Where's my Water?) while others have discrete space where game actions depend (Fruit Dating, Sokoban, and HueBrix).

3.1.1 Puzzle Level Generation

As there is nothing before like this work, this section will show all previous work that can be slightly related. One of the earliest research in Puzzle Games was by Murase et al.[16] trying to generate well designed solvable levels for Sokoban game. His system consists of 3 stages generate, check, then evaluate.

- **Generate:** The system generates levels layout using predefined templates and placing them at random positions. Goal areas are placed afterwards at random position avoiding placing it at places where player can't push object towards it. For boxes placement more processing is done, as he choose random position that is reachable from the goal area.
- **Check:** The system then uses Breadth First Search (BFS) to check for solution for the generated levels an remove all unsolvable levels.
- **Evaluate:** The system check all the legal candidate levels and remove levels with very short solution, less than four direction changes in the solution, or no detours in the solution.

The system generated levels around 500 levels but only 14 are marked as good levels. The generated levels have short solution sequence due to the usage of BFS.

Taylor and Parberry[20] followed Yoshio Murase et al.[16] work in generating levels for Sokobon game. He followed a system similar to previous consists of 4 stages generating empty room, placing goals, finding the farthest state, then evaluating generated levels.

- **Generating empty room:** The system generate the size of empty rooms then using predefined 3x3 templates, it starts creating the layout of the level. Generated levels are discarded if they are not completely connected, they have huge empty space, the number of empty floors is less than planned number of boxes and goals, there
- **Placing goals:** The system places goal areas using brute force algorithm to try every possible combination for goals.
- **Finding the farthest state:** The system tries to find farthest position from goal area using a similar algorithm to BFS. The algorithm continues till no more possible locations can be found. It returns the farthest locations using a box line metric heuristic which calculates the number of unrepeated moves after each other.
- **Evaluating generated levels:** The system evaluates the generated levels and give each one score based on some heuristics. For example: number of box pushes in the solution, number of levels found at same depth, number of box lines, number of boxes, and ...etc.

The generated levels by system using single box are generally uninteresting and as the number of boxes increased they become more interesting and more difficult but takes more time to generate.

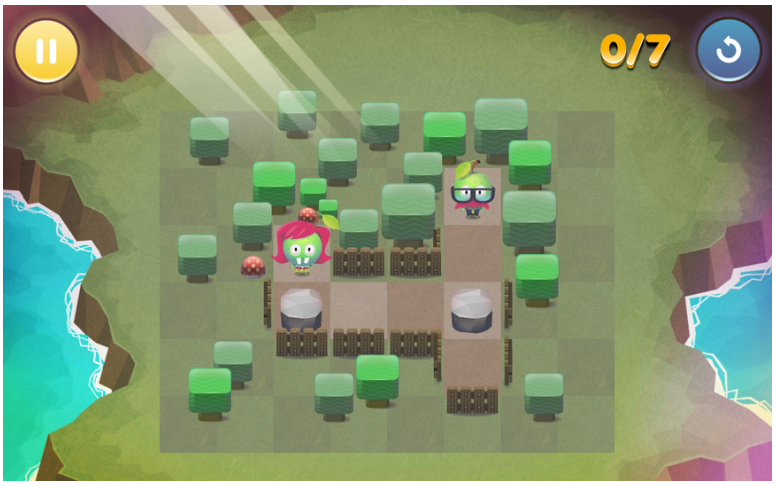


Figure 3.1: Screenshot from Fruit Dating

Rychnovsky worked on generating puzzle levels layout for his new game Fruit Dating[8]. As shown in Figure 3.1, Fruit Dating is a puzzle game where the player need to make all similar fruits meet each other. In order to move objects in the level player can swipe in one of the four direction (up, down, left, right). Swiping in any direction cause all level object to move toward that direction. The game sounds simple but with different kind of objects, it become very difficult. His system starts with generating level layout utilizing symmetry when generating two walls at a time. The system generates fruits at random locations based on predefined weights. For example placing fruits is more preferable at the end of corridors, followed by corners, followed by corridors, and last thing is open area. Other game objects are placed with the same strategy but using different weights for different locations. The system tries to solve the generated level and return the solution using an algorithm similar to BFS. Rychonovsky made a level editor to edit the generated levels and test for playability using autosolver algorithm. The technique

doesn't take more than couple of minutes to generate a level but the main problem is there is no way to know the difficulty of level before generating it or even influence difficulty in the generation.

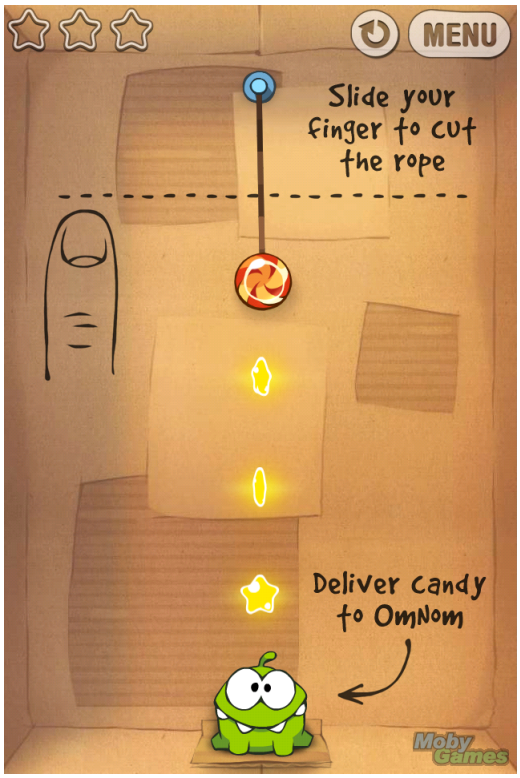


Figure 3.2: Screenshot from Cut The Rope game

Shaker et al.[17] worked on generating levels for physics based puzzle games and applied his output on Cut The Rope (CTR) . As shown in Figure 3.2, CTR goal is to cut some ropes to free the candy to reach OmNom (a frog monster fixed at certain position on the screen). The game is not as easy as it sounds as levels advanced more components are introduced which affect on the movement of the candy to redirect it to OmNom. Shaker used Grammar Evolution to generate levels for the game where he enforced at least one object from each game component. Shaker used a fitness function completely depending on how CTR levels are designed. For example: Candy should be placed higher than the Om Nom, Om Nom must be placed below the lowest rope, and ...etc. He generated 500 levels and then show some analysis on the expressive range of the generator such as frequency, density, and ...etc. Since there is nothing to make sure the level is playable, so Shaker modified the fitness function with a score generated from random player plays the generated level. If the random player couldn't solve the generated level in 10 trials or less, the fitness is decreased by large number to indicate that level is unplayable. Shaker generated 100 playable levels using and show there expressive range which didn't change a lot from the previous.

Shaker et al. conducted their research on CTR in an another paper[19] and developed a AI to simulate game playing better than random player in the previous paper[17]. He introduced two kind of AI, the first one takes actions based on the current properties of

components in the level, while the second one depends on reachability of the components by the candy based on its current position and velocity and direction. The levels generated from this paper are more diverse from the previous because random agent may discard some potential levels.

Shaker et al. started working on reducing the time for his generating levels in 2015[18]. He introduced a new search technique named Progressive Approach. Shaker uses GA to evolve timeline of game events. This evolved timeline is evaluated by simulating the game and following the timeline. An intelligent agent is used during the simulation utilizing prior knowledge about the game to map the timeline to a possible level layout. If the agent succeed it return the layout, else it returns invalid. This technique can be used on any kind of games to improve generation time, so he tested it on CTR. The results was better than simulation approach as this technique took on average 9.79 seconds to generate a level while previous one [19] took around 82 seconds, the only drawback that you can't know the difficulty of the level from the timeline, you need to simulate it to evaluate level difficulty.

3.1.2 Platformer Level Generation

3.1.3 Dungeon Level Generation

3.2 Rule Generation

3.3 General Video Game Playing

Chapter 4: Methodology

You are free to use any font type and size within your figures and tables provided that they are clear enough to the reader. For large size tables, you may separate them into several pages. For large size figures that cannot be resized to fit an A4 size paper without the loss of clarity, you may use larger size papers provided that they are properly folded to the A4 size and firmly bonded with the rest of the thesis.

4.1 Rule Analysis

Figures and tables should be included in the main text as close to the point of their introduction as possible. For figure captions use 12 point Times New Roman, bold, centered; place below the figure, use spacing of 12 points above and 24 points below. Leave two blank lines between the figure and the text above it.

Figure 4.1 shows ….

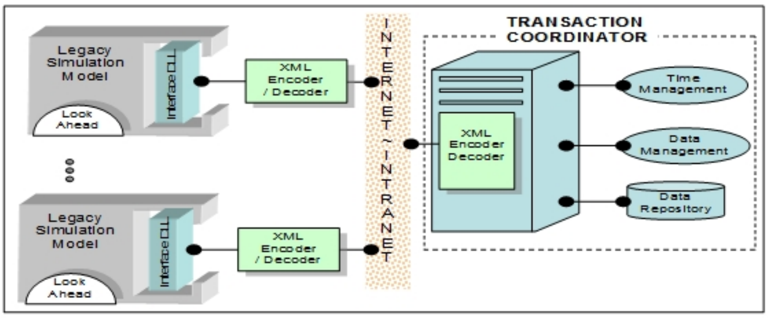


Figure 4.1: Example figure for demonstration

4.2

Body text. Body text Body text. Body

Table 4.2: Example table for demonstration

xxxx	1233	ccccc
uuuuu	2323	gggggg

text. Body text. Body text. Body text. Body text. Body text. For table captions use 12 point Times New Roman, bold, centered; place above the table, use spacing of 24 points above and 12 points below. Leave two blank lines between the figure and the text above it.

Table 4.1 demonstrates ….

Table 4.1: Example table for demonstration

Use two blank lines below the table. In some cases you may need to include a wide table using the full available paper height with a 90^o clockwise rotation. The following page shows one such table, Table 4.3.

Table 4.3: Another example wide table for demonstration

4.2 Auto Player

Body text. Body text Body text.

4.3 Level Generation

4.3.1 Linear Algorithm

4.3.2 Chromosome Representation

4.3.3 Fitness Function

4.4 Rule Generation

4.4.1 Chromosome Representation

4.4.2 Fitness Function

Chapter 5: Results and Evaluation

In this research, the common industrial problem of …. As extension to this work, the following points are recommended for the future work;

5.1 Data Description

5.2 Level Generation Problem

5.3 Rule Generation Problem

Chapter 6: Conclusion

In this research, the common industrial problem of …. As extension to this work, the following points are recommended for the future work;

References

[1] Akalabeth. http://www.filfre.net/2011/12/akalabeth/. [Accessed: 2015-01-18].

[2] Bfxr. http://www.bfxr.net/. [Accessed: 2015-01-19].

[3] Every game studio that's closed down since 2006. http://kotaku.com/5876693/every-game-studio-thats-closed-down-since-2006. [Accessed: 2015-01-20].

[4] Generate everything. http://vimeo.com/92623463. [Accessed: 2015-01-19].

[5] Genetic algorithm. http://en.wikipedia.org/wiki/Genetic_algorithm. [Accessed: 2015-01-18].

[6] History of video games. http://en.wikipedia.org/wiki/History_of_video_games. [Accessed: 2015-01-18].

[7] How much does it cost to make a big video game? http://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649. [Accessed: 2015-01-20].

[8] Procedural generation of puzzle game levels. http://www.gamedev.net/page/resources/_/technical/game-programming/procedural-generation-of-puzzle-game-levels-r3862. [Accessed: 2015-02-24].

[9] Puzzle script. http://www.puzzlescript.net/. [Accessed: 2015-01-19].

[10] The rip-offs and making our original game. http://asherv.com/threes/threemails/. [Accessed: 2015-01-21].

[11] Sokoban. http://en.wikipedia.org/wiki/Sokoban. [Accessed: 2015-01-19].

[12] What is the budget breakdown of aaa games? http://www.quora.com/What-is-the-budget-breakdown-of-AAA-games. [Accessed: 2015-01-21].

[13] Cook, M., Colton, S., Raad, A., and Gow, J. Mechanic miner: Reflection-driven game mechanic discovery and level design. In *Applications of Evolutionary Computation - 16th European Conference, EvoApplications 2013, Vienna, Austria, April 3-5, 2013. Proceedings* (2013), pp. 284–293.

[14] Ebner, M., Levine, J., Lucas, S. M., Schaul, T., Thompson, T., and Togelius, J. Towards a Video Game Description Language. In *Artificial and Computational Intelligence in Games*, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds., vol. 6 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013, pp. 85–100.

[15] Levine, J., Congdon, C. B., Ebner, M., Kendall, G., Lucas, S. M., Miikkulainen, R., Schaul, T., and Thompson, T. General Video Game Playing. In *Artificial and Computational Intelligence in Games*, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds., vol. 6 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013, pp. 77–83.

[16] Murase, Y., Matsubara, H., and Hiraga, Y. Automatic making of sokoban problems. In *PRICAI'96: Topics in Artificial Intelligence, 4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia, August 26-30, 1996, Proceedings* (1996), pp. 592–600.

[17] Shaker, M., Sarhan, M. H., Naameh, O. A., Shaker, N., and Togelius, J. Automatic generation and analysis of physics-based puzzle games. In *CIG* (2013), IEEE, pp. 1–8.

[18] Shaker, M., Shaker, N., Togelius, J., and Abou Zleikha, M. A progressive approach to content generation. In *EvoGames: Applications of Evolutionary Computation* (2015).

[19] Shaker, N., Shaker, M., and Togelius, J. Evolving playable content for cut the rope through a simulation-based approach. In *AIIDE* (2013), G. Sukthankar and I. Horswill, Eds., AAAI.

[20] Taylor, J., and Parberry, I. Procedural generation of Sokoban levels. In *Proceedings of the 6th International North American Conference on Intelligent Games and Simulation (GAMEON-NA)* (2011), EUROSIS, pp. 5–12.

[21] Togelius, J., Shaker, N., and Nelson, M. J. Rules and mechanics. In *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2015.

[22] Togelius, J., Shaker, N., and Nelson, M. J. The search-based approach. In *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2015.