Generating two-dimensional game maps with use of cellular automata

Michał Wolski

June 7, 2018

Contents

Chapter 1

Introduction

During recent years, presence of computer games in human lives has increased. Players demand for games has shown that playing games, both as a medium of expression and a means for entertainment, is a desirable form of activity. However, as the demand for games rises and the games become increasingly complex, there is also a rising need for game content such as believable maps, game objects, textures, sound and models, among other types. Studies such as [Hen+13] show where the evidence for insufficiency of manual content creation may be found:

TODO: no quoting, instead explain what authors presented

> (...) content production has grown to the point at which it has become a bottleneck in both game budgets and product time-to-market [Kelly and McCabe 2007; Lefebvre and Neyret 2003; Smelik et al. 2009; Iosup 2009].

TODO: ref https://www.isfe.eu/industry-facts/statistics

In order to reduce the cost of game development, allow for greater replay value or provide a feeling of vastness to the game worlds that designers aim to create, procedural content generation techniques can provide an attractive solution to the problem of content creation. Surveys such as [Hen+13], [Tog+11] and [DC+11] show what types of game content can be generated and are a good starting point for seeking methods of procedural generation.

TODO: write more on motivation for thesis - project UW
TODO: summarize why we are doing this
TODO: short paragraph on the choice of CA for game maps... here or later?

1.1 Objectives

TODO: WHAT are we doing here?

This work focuses on automated creation of 2-dimensional game maps using a cellular automata approach.

TODO: why maps and not other content?

TODO: why 2d and not other?

This thesis will investigate the existing methods for procedural generation of game maps which resemble cave structures. Then, an approach that may be used for automated creation of such maps will be selected and examined with a focus on implementing a working map generator. Main points of focus for this project are as follows:

- ...

- research on cellular automata

- research on generative grammars

- ...

    TODO: more pcg methods? ... refer to surveys

- selection of a promising approach to use

- design of a map generator

- implementing the solution in a programming language of choice

TODO: is that all?

1.2 Thesis scope and structure

TODO: scope - what we will do, what we dont do

TODO: short: what could be done instead

TODO: structure - describe the layout

The overall structure of this thesis includes introduction and three chapters. The first chapter serves as a study on possible mechanisms that could be used for procedural generation and specifically, for creation of 2D maps for games. The second chapter describes the experiments, design and implementation of a solution to the problem. Last chapter summarizes the findings and concludes the thesis.

1.3   Technology and tools

The following paragraphs summarize what tools were involved during the project of thesis preparation and performing the experiments.

1.3.1   Hardware

All experiments in this thesis have been performed using a laptop with an Intel x64 2.0 GHz multi-core processor, 16GB RAM and an nVidia GeForce GTX 560M card.

1.3.2   Software

Development environment for the purposes of thesis experiments has been set up under Windows 10 operating system with the following software installed:

- TODO: any IDE? perhaps VS2017, VSCode or CodeLite

- TODO: software for drawing diagrams?

- TODO: graphics?

- ...

Other configuration details include:

TODO: environment variables, configuration specifics...

This thesis has been prepared with LaTeX system for document typesetting and *TeXworks* editor.

Programming languages

TODO: language - C++ for implementation
TODO: which compiler was chosen?

Libraries

TODO: Dear ImGui, by Omar Cornut or Nuklear, by Micha Mettke - for User Interface in Immediate Mode (realtime)
TODO: OpenGL for rendering

Design patterns

1.3.3   Other tools

1.4   List of abbreviations and acronyms

The following terms, abbreviations and acronyms have been used in the thesis.

CA Cellular Automaton. A real-time simulation of cell objects.

PCG Procedural Content Generation. An automated process of creation.

PCGG Procedural Content Generation for Games

Moore's neighborhood In a context of a 2D square grid, Moore neighborhood includes the cell and its immediate neighbors - one to the north, south, east and west of the cell.

Von Neumann neighborhood In a context of a 2D square grid, Von Neumann neighborhood includes 8 closest neighbors of the cell - immediate and diagonal.

1.5   Related work (?)

Chapter 2

Research on 2D map generation methods

2.1 Automated creation of content

TODO: write about PCG in general, short
TODO: PCG types of content
TODO: PCG methods
TODO: focus on maps

2.2 Definitions

TODO: what is a map?
TODO: 2d map types?
TODO: what do we mean by generation?

2.3 Existing solutions for 2D map generation

TODO: HOW it was done until now? options?
TODO: ref survey with table of 2d dungeon gen

2.3.1 Cellular automata

A cellular automaton is a simulation in which every object in a mathematically defined space is being updated at every step of a simulation. Historically, cellular

automata and their properties have been studied since the time of first electronic computers [Sar00].

One of the most complete sources on cellular automata is a book summarizing research on CA carried out by Stephen Wolfram since 1980s [Wol02], where a classification of cellular automata is shown along with examples for each kind of CA.

TODO: write on use of CA for generation of content and then specifically maps

2.3.2   Generative grammars

2.3.3   L-systems

2.3.4   ...

2.3.5   ...

2.3.6   ...

2.4   Evaluation of existing methods for project purposes

In order to effectively judge the value that a working map generator may bring to a game development project, we need to consider what characteristics should be evaluated. First, a useful generator must be effective at map generation.

TODO: how to measure effectiveness? time of map generation, map shape, desirable map features?

Another point to consider is how easy to use such generator can be. Game designers may ultimately decide to use manual methods of map creation if the method of map generation requires too much effort to include in their project.

TODO: how to measure such ease of use? accessibility?

The third aspect of choice what a generation method could be used is to consider how much value it brings to the designer versus what development costs it can reduce.

TODO: how to measure cost?

The following subsections describe how each of the mentioned aspects can influence the choice of a generation method.

2.4.1   Effectiveness

TODO: study on generation time
TODO: desired characteristics of generated content?

2.4.2   Accessibility

TODO: study on what makes generation easy to include in game development projects

2.4.3   Cost

TODO: examples of development costs - human resources, machine resources
TODO: which of these costs can be reduced by PCG

2.5   Chosen approach: cellular automata for 2D map generation

TODO: why we chose CA for mapgen?
TODO: what are pros and cons of such choice?

Chapter 3

Application for visualizing procedural generation of levels

Having chosen cellular automata as a method for generating maps, we need to have a clear idea about how to approach building a program that could simulate a cellular automaton. One of the helpful resources on the topic of building cellular automata simulations is chapter 7 in Nature of Code, a book by Daniel Shiffman [Shi12], where we can find a short tutorial to build our first CA simulation. There, author describes elementary concepts needed to construct a basic CA, explains how to implement a working simulation and provides helpful exercises. The tutorial is quite useful as a guide, since examples presented in New Kind of Science [Wol02] are implemented in the Wolfram language and would require familiarity with it.

3.1   Basic CA simulation

In order to build a map generator based on a cellular automaton, we must first understand the basics. As stated in the tutorial by Daniel Shiffman [Shi12], a CA simulation would need the following key elements:

- Cell state - every cell has a state updated on each simulation step,

- Grid - a space on which cells are placed,

- Neighborhood - each cell needs to know the state of its neighbors to update its state.

Another observation is that whenever cell states need to change (the simulation moves to a later step), the state change is applied to every cell in the grid before simulation step ends and cells do not need to be updated in sequence if only all cells will be changed before the next step. Hence, cell state updates could be applied in parallel to reduce the time needed to compute the simulation step.

TODO: more on CA, 2dim CA

3.2 Generating content for a game

Since the goal of this work is not to just implement a working cellular automaton simulation, we need to find a way to generate maps using CA simulation. One of possible proposed approaches is the work of L. Johnson, G. Yannakakis and J. Togelius from IT University of Copenhagen [JYT10].

Authors describe an

TODO: 1 random image
TODO: 2 apply CA steps as in article cavegen
TODO: 3 voila, maps!

3.3 Analysis of requirements

Having gathered the abstract constructs needed to build a CA map generator, we may proceed to state the requirements formally.

3.3.1 Functional requirements

- user interface allowing playing with parameters

- rendering each generation step

- exporting generated maps

- ...

3.3.2 User requirements

- allow changing parameters by user

- format of exported maps must be easy to understand and use

- ...

3.3.3   Constraints

3.4   Design

3.4.1   Data structures, persistence

TODO: how do we store data?
TODO: exporting data from generator?
TODO: how designers can get a complete map model?

3.4.2   Application logic

TODO: how a generator will work
TODO: diagrams

3.4.3   User interface

TODO: mention Bret Victor talks
TODO: imgui immediate mode user interface

3.5   Implementation

3.6   Tests

3.6.1   Performance test

3.7   Deployment (?)

Chapter 4

Conclusions

4.1   Evaluation of results

4.2   Perspectives for usage

4.3   Further work

Bibliography

[Sar00]   Palash Sarkar. "A Brief History of Cellular Automata". In: *ACM Comput. Surv.* 32.1 (Mar. 2000), pp. 80–107. ISSN: 0360-0300. DOI: `10.1145/349194.349202`. URL: `http://doi.acm.org/10.1145/349194.349202`.

[Wol02]   Stephen Wolfram. *A new kind of science*. Vol. 5. Wolfram media Champaign, 2002.

[JYT10]   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*. ACM. 2010, p. 10.

[DC+11]   Daniel Michelon De Carli et al. "A survey of procedural content generation techniques suitable to game development". In: *Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on*. IEEE. 2011, pp. 26–35.

[Tog+11]   Julian Togelius et al. "Search-based procedural content generation: A taxonomy and survey". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (2011), pp. 172–186.

[Shi12]   Daniel Shiffman. *The Nature of Code: Simulating Natural Systems with Processing*. Daniel Shiffman, 2012.

[Hen+13]   Mark Hendrikx et al. "Procedural content generation for games: A survey". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9.1 (2013), p. 1.

List of Figures

List of Tables

Chapter 5

Attachments

1. Załącznik 1

2. Załącznik 2

3. Załącznik 3

5.1   Full source code: main.cpp

```cpp
1  #include <iostream>
2
3  #include <GL\gl3w.h>
4  #include <GLFW\glfw3.h>
5
6  #include "UserInterface_MapGenerator.h"
7
8  GLFWwindow* window;
9
10 void glfw_error_callback (int error, const char* description)
11 {
12   std::cerr << "GLFW Error " << error << ": " << description << std::endl;
13 }
14
15 bool glfwSetupWindow (unsigned int width, unsigned int height, const char* title)
16 {
17   glfwSetErrorCallback (glfw_error_callback);
18   if ( glfwInit () )
19   {
20     glfwWindowHint (GLFW_CONTEXT_VERSION_MAJOR, 3);
21     glfwWindowHint (GLFW_CONTEXT_VERSION_MINOR, 2);
```

```cpp
22    glfwWindowHint (GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
23    glfwWindowHint (GLFW_MAXIMIZED, GLFW_TRUE);
24  #if __APPLE__
25    glfwWindowHint (GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
26  #endif
27    window = glfwCreateWindow (width, height, title, NULL, NULL);
28    glfwMakeContextCurrent (window);
29    glfwSwapInterval (1); // Enable vsync
30    gl3wInit ();
31    return true;
32  }
33    return false;
34  }
35
36  int main (int, char**)
37  {
38    if ( !glfwSetupWindow (800, 600, "Cellular Automata Map Generator 152017") ) return 1;
39    else
40    {
41    UserInterface_MapGenerator missionControls = UserInterface_MapGenerator (window);
42    while ( !glfwWindowShouldClose (window) ) // Main loop
43    {
44      glfwPollEvents ();
45      {
46        missionControls.Update ();
47        missionControls.Render ();
48      }
49      glfwSwapBuffers (window);
50    }
51  }
52    glfwDestroyWindow (window);
53    glfwTerminate ();
54    return 0;
55  }
```

Listing 5.1: main.cpp Source Code