A PROCEDURAL GENERATION PLATFORM TO CREATE RANDOMIZED GAMING MAPS USING 2D MODEL AND MACHINE LEARNING

Nathan Lee[1], John Morris[2]

[1]Northwood High School, 4515 Portola Pkwy, Irvine, CA 92620
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

As a video game developer, the most difficult problem I ran into was creating a map for the game, as it was difficult to create non repetitive and original gameplay [1]. My project proposes a solution to this problem as I use Answer Set Programming to create a program to procedurally generate maps for a video game [2]. In order to test its reliability, I allowed it to generate around 10,000 maps, stored the data of each of the maps, and used the common trends I find in the data to find problems with the program and fix it in the future.

In developing a video game, level creation consumes a major portion of the development total time and level procedural generation techniques can potentially mitigate this problem. This research focused on developing a VVVVVV style level generator using Answer set programming for the game Mem. experiment which was developed at the same time. VVVVVV is a 2D puzzle platformer that uses changes in direction of gravity instead of jumping for the player's vertical movement [3].

During the development of the level generator, 10,000 levels were created. I found out that the average total time it took between generations is 45 seconds, and the average time for ASP to generate a map is 12 seconds [4]. This means that the process of displaying the generation took between 2x - 3x longer than generating the ASP solution.

KEYWORDS

ASP, Procedural Generation, 2D Game

1. INTRODUCTION

As technology became more advanced, video games began to popularize in our culture as an entertainment activity. Video game design has become more relevant in society nowadays and has become a popular career pathway to many [5]. Although being a Video game developer sounds like an interesting job to do, there are still very tedious tasks, including and not limited to map design. Map design for large scaled games is very time consuming and will take weeks and even months to complete.

Road signs are crucial to the safety of drivers and pedestrians getting from one place to another safely. Red signs such as stop signs, yield signs, and do not enter signs organize traffic into something manageable for drivers to use the roads safely. Yellow signs serve as precautions to drivers in order to prevent accidents. If a sign is identified incorrectly, it could result in a dangerous situation or a costly ticket, which makes it essential that signs are accurately deduced and followed. Advanced detection technologies such as CCTV cameras allow engineers to receive real-time traffic data and access live traffic video, which allows them to make the appropriate adjustments in order to keep traffic going smoothly [6]. Moreover, autonomous cars need to be capable of detecting their environment without human navigation.

PROCEDURAL CONTENT GENERATION (PCG) is a game-design technique that involves creating game content via automated processes rather than via hand-authoring [7]. ASP or Answer Set Programming is a type of declarative programming language used to solve difficult search problems. (Smith and Mateas, 2011) ASP can be used to select solutions that match the desired properties that the developer wants, which is applicable to procedural content generation, or in this case, the procedural generation of maps in video games.

Research Procedurally generated games

- Terraria
  - Perlin Noise
- Minecraft
  - Perlin Noise

No man's sky

- Spelunky

  - https://www.youtube.com/watch?v=RiDy6CgBKqs&ab_channel=GDC
  - Spelunky is a 2D platform game created by Derek Yu. Spelunky also uses procedural content generation in order to create its maps.
  - The maps are generated with different rooms, and each room is selected from a set of predetermined template rooms.

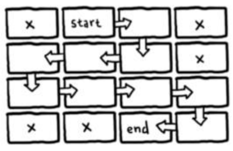


Fig. 3.13: Level generation in *Spelunky*. Adapted from [10]

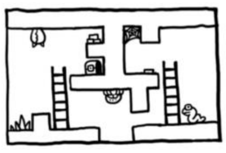


Fig. 3.14: Example room design in *Spelunky*. Adapted from [10]

Figure 1. Spelunky

Each of the boxes on the 3.13 are chunks, and the chunks are replaced by a random number, a program then sets a number to be the start, and a number to be the end. Then the obstacles are generated within it as presets, and the monsters are generated within the level randomly given there is enough space between them.

My tool is a program using ASP to procedurally generate maps for a sample game I created.
I also created a script that allows me to infinitely generate a map, save the data of the map, and then delete and start over. This helps with data collection as I would have a large sample size to work with. I will be using the collected data to find problems within my program to improve/fix it in the future.

I collected data from around 10,000 maps that were generated with the ASP solver. The data includes totalTime (time between the generations), aspSolveTime (time the ASP program took to create a complete solution), satisfiableCount (amount of rooms that meets the criteria), unsatisfiableCount (amount of rooms that does not meet the criteria and had to be regenerated), and roomCount (the total number of rooms in the maps).

Things to note: the ASP solver first takes some time to generate a general layout of the map , and then generates a map based on the paths of the layout. The generation of the map is not included in the aspSolveTime.

The map is also divided into multiple rooms, which are generated one at a time. Each green box is a room.

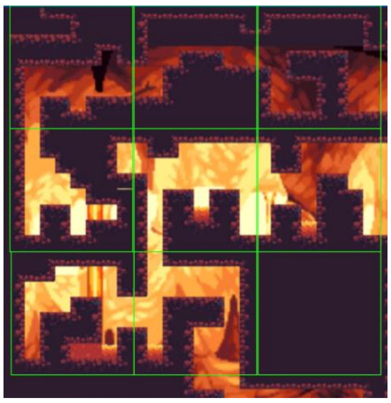


Figure 2. Map

Section 1 is the introduction to the problem and my project; Section 2 is the steps I took to begin the project and research; Section 3 is the data collection; Section 4 is the explanation of the significance of the data collected from section 3; Section 5 is how I can use the data to solve the problem with the program.

2. Challenges

In order to build the project, a few challenges have been identified as follows.

2.1. Learning How to Use ASP

The lack of research and information in this field:

ASP was developed in the late 1990s but was overshadowed by many other programming languages such as C#, Java, and C++ [15]. The usable branches of ASP like Clingo made by potassco (potsdam answer set solving collection) were only developed a little over 2 years ago [8]. When I tried to do research and learn more about ASP, I struggled a lot to find information related to ASP.

2.2. How to Ensure the Players can Utilize Clingo on their Own Computer

Clingo is a separate program that is not always compatible:
Utilized a server which hosted clingo allowing players to generate levels without having to install clingo on their computers [9]. how to deal with server interruptions: ensure the player does not get stuck in the generated level.

3. SOLUTION

1.Created base game as a template to test the ASP program.
2.Used ASP to create simple tunnels from 1 point to another in a map.
3.Added variations and randomization to the maps.
4.Added more limitations to the map to keep the results desirable.
5.Created code to store data of the map generations and allowed the program to create around 10,000 maps.
6.Create a box plot for the data, check data for errors and for any trends that may have occurred and caused these errors.
7.Implement a plan to solve the error.

Mem.Experiment is a game that uses a solver made with ASP (answer set programming) to procedurally generate maps. This project focuses on optimizing the program by running it multiple times and saving the data. I then create a graph to display all of the data in order to notice the trends. I then try to make a conclusion based on the data and attempt to improve the program.

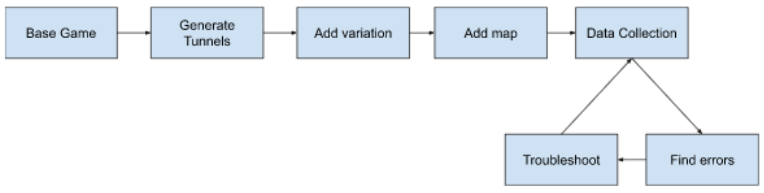


Figure 3. Overview of the solution

- ASP level generation
  - ■ Each level is made from rooms each of which are defined by a grid of tiles that are either filled or empty.
  - ■ Each room has a defined entrance and exit.
    - ◆ the asp solver (asp code) randomly selects the entrance and exit tiles along the corresponding edge.
  - ■ Each room either has a path obstacle or is just a traversal room

- Path obstacles can be either lava or spikes which require the path entrance to exit of the player to cross the obstacle
- lava must be on ground tiles and contained with filled tiles or lava on either side
- spikes can be on any filled tile

- Unsatisfiable rooms

  - When a built room results in a neighboring room being unsatisfiable, that room is removed from the solved list and regenerated after the unsatisfiable room is generated.

- Graph Generation

  - ASP generated graph that defines how each room is connected to reach and what, if any, obstacles that room has.
  - the graph ensure that the player does can reach every room and does not get stuck

I started with creating a base game for testing the ASP program, which was inspired by a game created in September 2010 called VVVVVV. For the development of the ASP program, I started with just generating a tunnel from one side to the other [14]. Then I began adding variations and randomizations to make it look more like a cavern. I then decided to subdivide the entire map into different rooms, which allowed me to use ASP to create a random graph (Shown in first image below) and then make a map around it. I then implemented a data collection system which allows me to store data of over 10,000 rooms that I generated. I then imported the data to Microsoft Excel in order for me to create a box plot. (Shown in second image below)

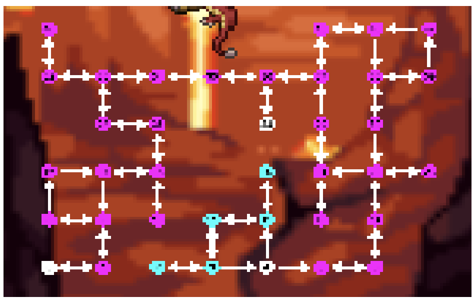


Figure 4. Rooms generated

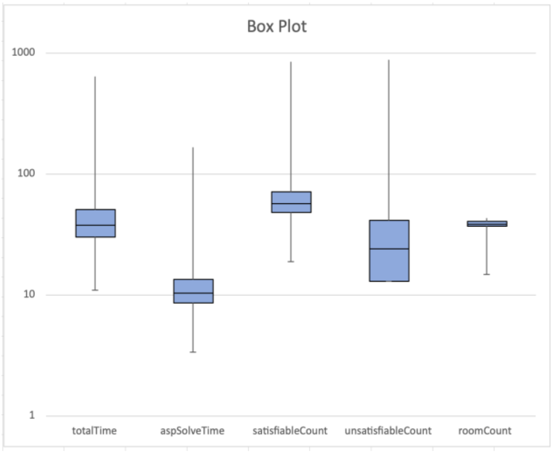


Figure 5. Boxplot

4. EXPERIMENT

Experiment 1 Runtimes: analysis and causes of values

lava percent

Experiment 2: Pathfinding and verification

There is a room where the player could fall out of the level, If the player enters from a specific side, then the opening leading to outside the map could never be reached.
Error from the graph generation

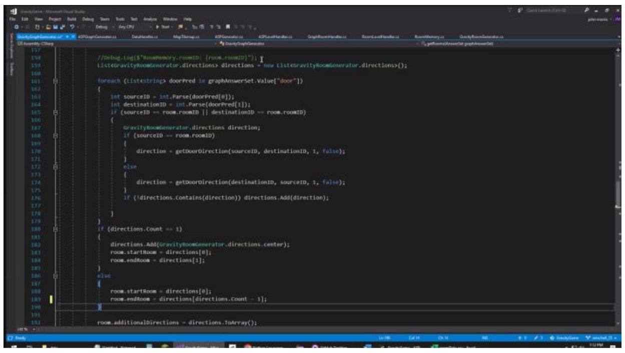
Experiment 3: Level quality: what parameters make a good/fun level

Expressive Range:

lava percent: lavaTiles / allTiles, lava/empty, lavaRooms/allRooms, percentLava/UNSATIFIABLE rate

percent empty:

Reachable:



```
            //Debug.Log($"RoomMemory.roomID: {room.roomID}");
            List<GravityRoomGenerator.directions> directions = new List<GravityRoomGenerator.directions>();

            foreach (List<string> doorPred in graphAnswerSet.Value["door"])
            {
                int sourceID = int.Parse(doorPred[0]);
                int destinationID = int.Parse(doorPred[1]);
                if (sourceID == room.roomID || destinationID == room.roomID)
                {
                    GravityRoomGenerator.directions direction;
                    if (sourceID == room.roomID)
                    {
                        direction = getDoorDirection(sourceID, destinationID, 1, false);
                    }
                    else
                    {
                        direction = getDoorDirection(destinationID, sourceID, 1, false);
                    }
                    if (!directions.Contains(direction)) directions.Add(direction);

                }
            }
            if (directions.Count == 1)
            {
                directions.Add(GravityRoomGenerator.directions.center);
                room.startRoom = directions[0];
                room.endRoom = directions[1];
            }
            else
            {
                room.startRoom = directions[0];
                room.endRoom = directions[directions.Count - 1];
            }

            room.additionalDirections = directions.ToArray();
```

Figure 6. screenshot of code 6

The Room asp only verifies that there is a valid path from the start to the end, but doesn’t ensure that the path is accessible from both directions. Because of that, the player can potentially fall out of the map if the player enters from the exit side

4.1. Experiment 1

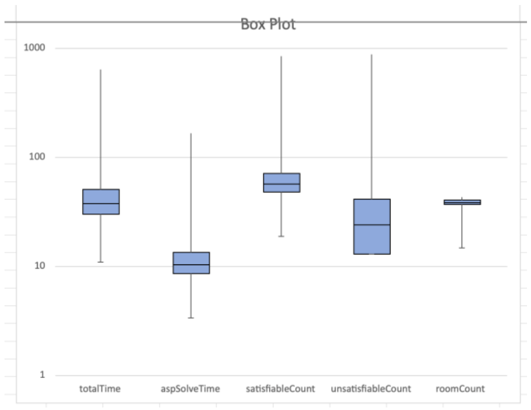


Figure 7. Boxplot of experiment 1

The table above shows the total time between trials, the total time when asp is running, the amount of satisfiable rooms generated (rooms that were generated without problem), unsatisfiable rooms (rooms that had to be re-generated because of error), and the total amount of rooms in the map.

The totalTime seemed to be on average, higher than the aspSolveTime. Since the graph generation isn't included in the aspSolveTime, we can assume that the total time is mostly composed of aspSolveTime + graph generation.

4.2. Experiment 2

I used data from experiment 1 to manually check the maps that had the highest number of unsatisfiableCount. Then I looked at the process of generation and try to see what type of room caused the most re-generations.

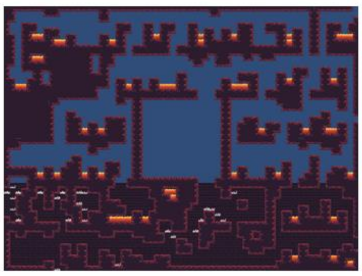


Figure 8. Graph of experiment 2

Levels that contain more lava-rooms generally have a higher solve time. The reason for that is because lava has an unique property of being forced to generate on the ground, and it must be contained by blocks on both sides, and each lava-rooms must contain at least one tile of lava. That means that if multiple lava rooms generate on top of each other, there has to be paths going through the ground of the lava rooms, and since each rooms are only 8 tiles on each side, this severely limits the region where the lava could spawn and a room could cause other rooms to also fail to generate, resulting in a higher solve time.

5. Related Work

This paper talks about various information about Answer Set Programming and how many properties are combined to create randomized solutions with given constraints [10].

In order to understand ASP, you must have some knowledge about AnsProlog. AnsProlog refers to a concrete syntax one uses to write answer set programs, this is derived from Prolog, which is a deductive logic programming language. Terms are terminated with a period.

The chapter shows us sections of ASP codes and roughly translated them to english terms to explain some properties of ASP.

Integrity constraints let programmers express what can not be possible for the program to accomplish.

The ASP solver is used to create a solution based on the answer sets that are fed into it.

This paper talks about the implementation of graph based map generation for ASP inside action-adventure games, more specifically in ‘dungeons ’(A type of experience which features variety of obstacles, exploration, increasing access to areas, bosses, and more [11]. It mentions how a play through of an action-adventure game can be simplified into a graph that clearly portrays intended progression of the game, and the ASP can generate a map based around the graph.

My project also uses ASP to generate a graph, which is then used as a template to generate the map.

This paper talks about softlocking, an error with game design that causes the player to be stuck from progression permanently [12].

My ASP code specifically takes note of the player’s movement limitations and creates a map to make sure the player wouldn’t get soft locked.

6. CONCLUSIONS

I created a program using ASP to procedurally generate maps for 2D video games [13]. I then implemented a data collection system to collect data from over 10,000 generated maps, which are used to locate problems that affect the efficiency of my program. I can then use this information to troubleshoot the problems.

The current limitations of my program is that the speed of the trials can be greatly optimized by fixing some of the problems I mentioned. I am also sure that there are a lot of other problems affecting the speed and efficiency of the program, and some of them might require a different approach.

In the future, I plan to fix the current problems, which would make the generation faster. Then I would generate even more maps to find any new problems. This will essentially be a cycle of constant improvement to my program.

REFERENCES

[1] Weststar, Johanna. "Understanding video game developers as an occupational community." Information, communication & society 18.10 (2015): 1238-1252.
[2] Brewka, Gerhard, Thomas Eiter, and Mirosław Truszczyński. "Answer set programming at a glance." Communications of the ACM 54.12 (2011): 92-103.

[3] Smith, Gillian, Mee Cha, and Jim Whitehead. "A framework for analysis of 2D platformer levels." Proceedings of the 2008 ACM SIGGRAPH symposium on Video games. 2008.

[4] Erdem, Esra, Michael Gelfond, and Nicola Leone. "Applications of answer set programming." AI Magazine 37.3 (2016): 53-68.

[5] Dondlinger, Mary Jo. "Educational video game design: A review of the literature." Journal of applied educational technology 4.1 (2007): 21-31.

[6] Welsh, Brandon C., and David P. Farrington. "Public area CCTV and crime prevention: an updated systematic review and meta-analysis." Justice quarterly 26.4 (2009): 716-745.

[7] Hendrikx, Mark, et al. "Procedural content generation for games: A survey." ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 9.1 (2013): 1-22.

[8] Kaminski, Roland, Torsten Schaub, and Philipp Wanko. "A tutorial on hybrid answer set solving with clingo." Reasoning Web. Semantic Interoperability on the Web: 13th International Summer School 2017, London, UK, July 7-11, 2017, Tutorial Lectures 13 (2017): 167-203.

[9] Gebser, Martin, et al. "Multi-shot ASP solving with clingo." Theory and Practice of Logic Programming 19.1 (2019): 27-82.

[10] Smith, Adam M., 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 (2011): 187-200.

[11] Smith, Thomas, Julian Padget, and Andrew Vidler. "Graph-based generation of action-adventure dungeon levels using answer set programming." Proceedings of the 13th International Conference on the Foundations of Digital Games. 2018.

[12] Mawhorter, Ross, and Adam Smith. "Softlock Detection for Super Metroid with Computation Tree Logic." Proceedings of the 16th International Conference on the Foundations of Digital Games. 2021.

[13] Roettl, Johanna, and Ralf Terlutter. "The same video game in 2D, 3D or virtual reality–How does technology impact game evaluation and brand placements?." PloS one 13.7 (2018): e0200724.

[14] Brain, Martin, et al. "Debugging ASP programs by means of ASP." Logic Programming and Nonmonotonic Reasoning: 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007. Proceedings 9. Springer Berlin Heidelberg, 2007.

[15] Lano, Kevin, Sobhan Yassipour Tehrani, and Krikor Maroukian. "Case study: FIXML to Java, C# and C++." Transformation Tool Contest (TTC 2014). 2014.