Computer Engineering Department

Procedural Dungeon Generation

Project Advisor: Richard Sinn

Akash Mattaparthy (MS Artificial Intelligence)
Ganesh Tulshibagwale (MS Software Engineering)
Shyam Nalluri (MS Computer Engineering)
Srikari Veerubhotla (MS Computer Engineering)

Introduction

Procedural content generation (PCG) has revolutionized video game development by increasing variety and replayability while reducing manual design effort. Our project focuses on procedural dungeon generation, a classic application of PCG that provides players with dynamic and engaging environments to explore.

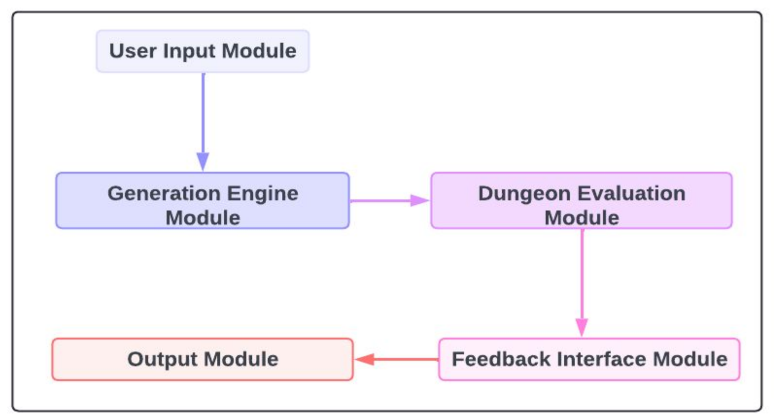

Fig 1. Flow Chart of Procedural Dungeon Generation

Project Goals:

- Develop a flexible, user-driven dungeon generation tool that adapts to player requirements.
- Incorporate player feedback and in-game behavior analysis to enhance and customize dungeon layouts.

Methodology

Map Design:

- **User Input:** Set dungeon parameters (size, complexity).
- **Graph Generation:** Algorithm creates dungeon flow with design patterns.
- **Placement:** Rooms and corridors are positioned for connectivity.

Smoothing & Refinement:

- **Terrain Smoothing:** Cellular automata reduce irregularities.
- **Refinement:** Heuristics ensure accessibility and playability.

User Interface:

- **Input Fields:** For parameters.
- **Generate Button:** Starts dungeon creation.
- **Visualization Panel:** Displays the dungeon map.

Feedback Collection:

- **AI Testing:** Uses Monte Carlo Tree Search to evaluate map playability.

Implementation and Results

1. Algorithm Performance

Graph Rewriting Algorithm:

- Transforms graph nodes and edges based on predefined rules.
- Facilitates the creation of intricate and interconnected dungeon designs.

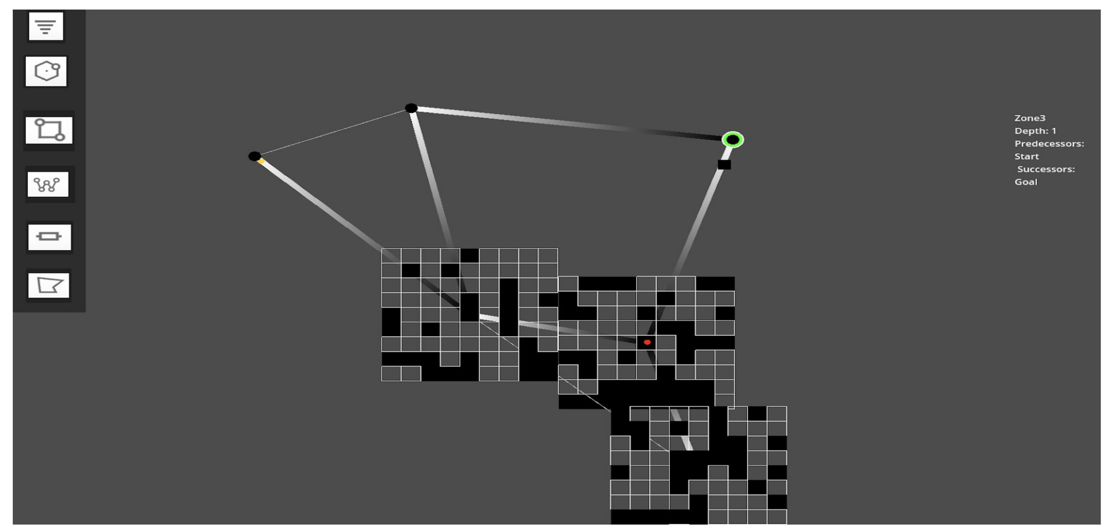

Fig 2. Graph Rewriting Algorithm Implementation

Minimum Spanning Tree (MST) Algorithm:

- Ensures all rooms are interconnected with the shortest possible paths.
- Optimizes dungeon layout by minimizing total path length.
- Guarantees that every room is reachable from every other room.

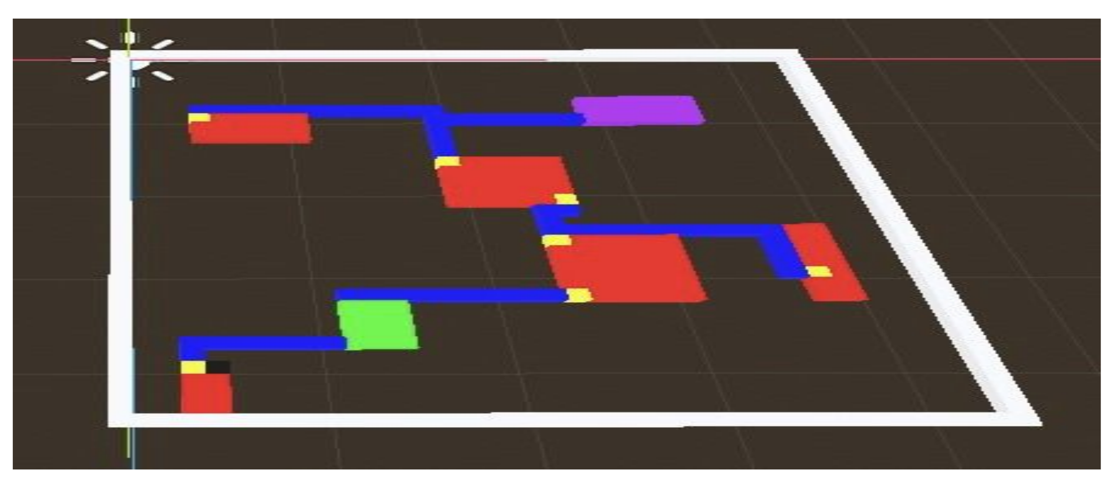

Fig 3. MST Algorithm Implementation

Feedback through agents:

- Uses an AI agent to play through the generated map.
- Builds and updates a decision tree through back-propagation.
- Runs for 1000 iterations (adjustable).
- Provides metrics to assess map complexity and gameplay experience.

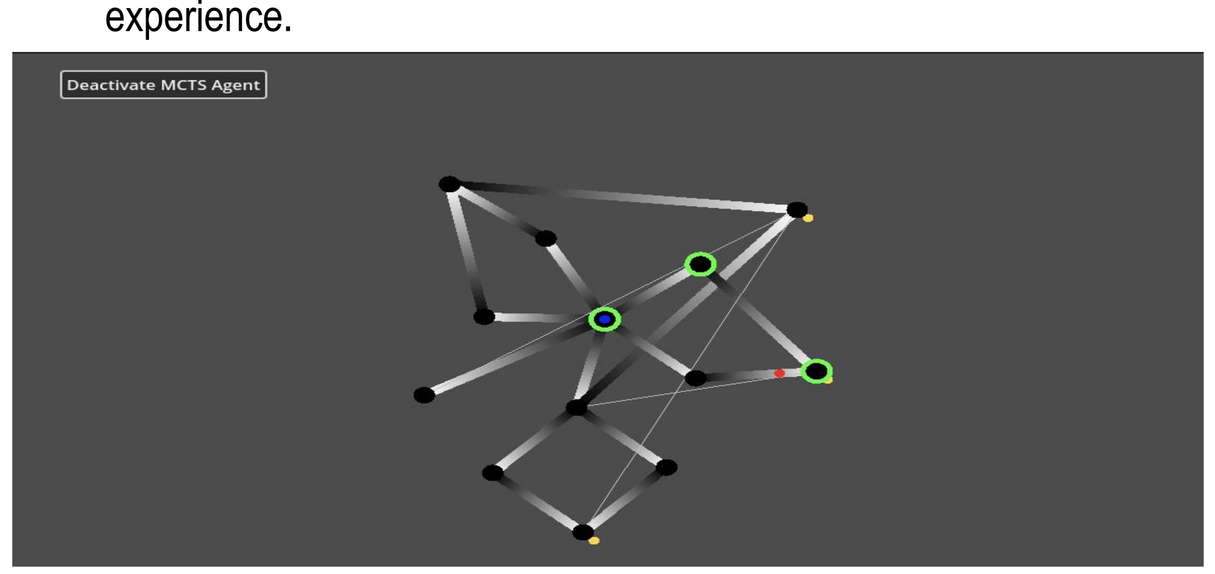

Fig 4. Feedback Collection UI

User Interface Implementation

The user interface (UI) is designed to be intuitive and user-friendly, facilitating easy input of dungeon parameters and visualization of generated dungeons.

Input Fields: Allow users to specify parameters such as border size, room number, room size and complexity.

Generate Button: Initiates the dungeon generation process.

Visualization Panel: Displays the generated dungeon map.

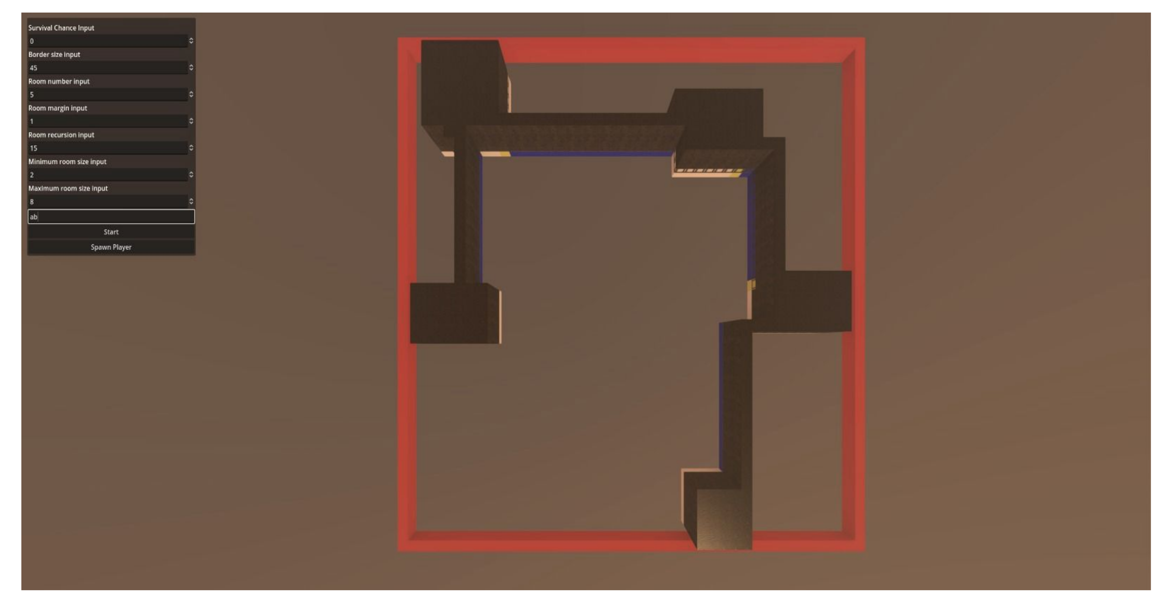

Fig 5. Procedural Dungeon Generator Interface

2. Scalability

In our project, scalability is evaluated through Frames Per Second (FPS), which measures the game's performance as the scene complexity, or the number of active objects increases.

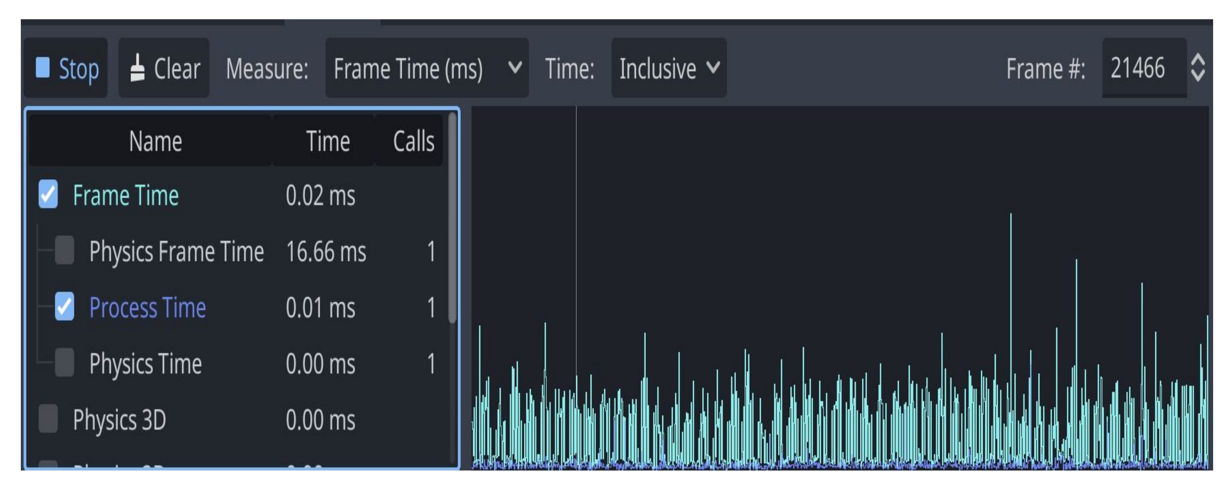

Fig 6. Scalability through Frames Per Second

Resource Utilization

Memory Usage

- **Static Memory**: 84.66 MiB
  This is the current memory usage for static allocations. The graph below shows fluctuations, with a recent peak approaching the maximum observed.
- **Static Max**: 93.03 MiB
  The peak amount of static memory used, representing the maximum static allocation.
- **Msg Buf Max**: 4.00 KiB
  Maximum size of the message buffer, which is much smaller than static memory, indicating efficient management.

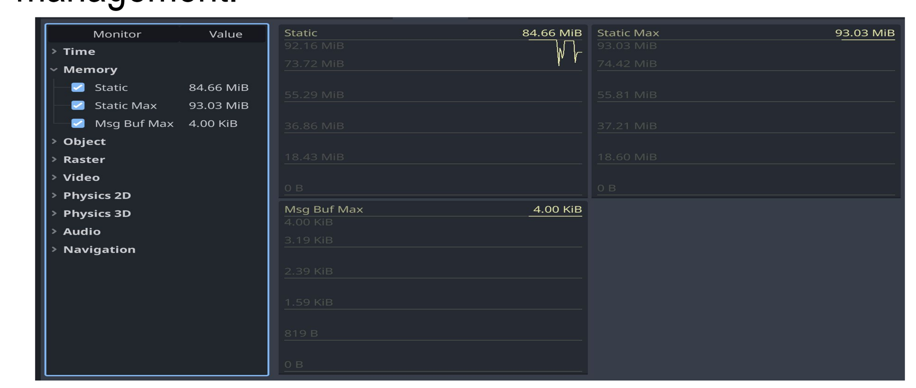

Fig 7. Memory Usage Graph

Summary/Conclusions

- The Procedural Dungeon Generation (PDG) system showcases significant progress in creating dynamic and varied dungeon environments through the use of advanced algorithms and AI.
- The system's iterative development, combined with user feedback, provides an effective solution that meets its design objectives, adapts to user needs, and maintains strong performance.

Key References

[1] Donjon. Random Dungeon Generator [Web tool]. Retrieved from https://donjon.bin.sh/fantasy/dungeon/

[2] Torres, P., and Gustavsson, P. (2017). L-system Application to Procedural Generation of Room Shapes for 3D Dungeon Creation in Computer Games. Chalmers University of Technology, Department of Computer Science and Engineering.

[3] Pereira, L. T., Prado, P. V., Lopes, R. M., and Toledo, C. F. M. (2021). "Procedural generation of dungeons' maps and locked-door missions through an evolutionary algorithm validated with players," Expert Systems with Applications, vol. 180, p. 115009.

Acknowledgements

- We express our sincere gratitude to Prof. Richard Sinn, our advisor, for believing in us and giving his invaluable guidance and support throughout this project.
- We also like to express our heartfelt appreciation to the distinguished engineering faculty at San Jose State University,