EG UK Computer Graphics & Visual Computing (2025)
Y. Sheng and A. Slingsby (Editors)

Peril: A Level Design Tool for Games Education

David James[1], Kieran Hicks[1] and Chris Headleand[1]

[1]Staffordshire Games Institute
School of Digital, Technologies, Innovation and Business
University of Staffordshire, UK

Abstract

This technical report presents Peril; a complete, playable video game framework containing a streamlined level creation system. Created using Unreal Engine 5, Peril provides students with a fully-featured, accessible framework for level design, incorporating a procedural assistance system which automates non-gameplay critical aspects of level layouts, such as walls and ceilings. Peril is used as a valuable pedagogic tool in games design education, supporting experiential learning and rapid protoyping - a task that is challenging to teach using base game engine tools. This paper describes the technical implementation of this procedural assistance approach, allowing practioners and academics the means of incorporating these techniques into their own teaching.

1. Background and Motivation

This technical report presents Peril, a game framework [JH24] containing a level creation systems designed to support rapid prototyping and experiential learning. Level design is a unique challenge within Games Design education, as (unlike many of the design subdisciplines) there are limited technical requirements with many low or no-code tools (such as level editors) to facilitate development. However, level designers have significant impact on player experience, and understanding how players interact with the environment requires a mix of strong conceptual understanding and practical experience.

To facilitate this experience there are commercial game titles that include level editors (such as Team Fortress 2), which has made them a popular choice across Higher Education games curricula. However, these games may have an associated purchase cost, or not be available on the different hardware platforms that students own. They also often lack access to the game's underlying code base, this means that any new feature or mechanic that the student may wish to implement is often impractical or impossible. Furthermore, level editors are often "over fitted" to a specific game, making the skills developed less transferable.

To address these limitations, a custom game framework containing a full set of mechanics and streamlined level-editing tools was created and provided to students freely. This was implemented in Unreal Engine, providing full access to the Game's underlying code and structure, while also supporting the students development of technical skills in an industry standard engine.

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

2. Design

Peril is a first-person shooter made using Unreal Engine 5. It is predominantly sprite-based, in style of classic 90's games such as 'Wolfenstein' and 'Doom'. Peril is a "2.5D" game; it appears 3D from the player's perspective, but the player and enemy movement is confined to a single flat plane (the floor). This means that first year students can focus on making compelling experiences, focusing on theoretical concepts such as beats, pacing, combat encounters and critical paths without the added complexity of 3D levels that contain verticality.

Figure 1: *Peril Gameplay.*

This approach has the further benefit of making the game very efficient, which was useful when students had the need to run the software on their own PC hardware. Peril contains a range of mechanics, including multiple different weapons and enemy characters, as well as environmental objects such as hazards, checkpoints

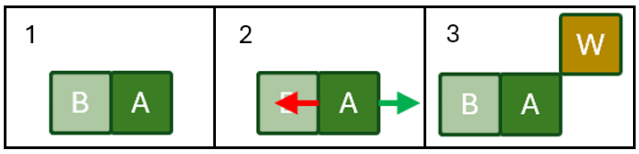

Figure 2: *Diagram showing Procedural Generation. Frame 1 shows two Floor tiles (A and B) from a side-on perspective. In Frame 2, Floor A is performing a trace on each side to see if it is adjacent to another Floor Tile. The red arrow denotes it has found a neighbouring floor, whilst the green arrow denotes a free space. In Frame 3, Floor A has spawned a Wall in the free space.*

and level goals. It is a full video game experience of commercial quality. Aside from having a "flat" design, Peril levels are also made of equally-sized floor and wall cubes.

This cube-based (voxel) approach to the level structure means that map design can be represented easily and accurately using "pixel maps"; where each pixel represents a single cube (floor or wall) of the level.

3. Procedural-Assisted Design

Peril features a "Procedural Assistance" system that supports rapid level design. The designer will place floor cubes, along with gameplay entities such as enemies, items, and the level goal. All of these entities snap into position (see figure 2). At run-time, Peril will procedurally convert these blocks into the necessary walls and ceilings to encapsulate the level. This is an agent-based system with each block being an individual agent that is responsible for generating specific content [DP10].

Walls and ceilings, and other visual obfuscations are important in a first-person game; they prevent the player from seeing the entire level from the offset, thus providing the need for exploration; they are used as cover from enemy attacks; and they limit the render distance within the engine (improving efficiency). This allows the designer to focus on the "blocking out" of the level, establishing the flow between areas and the subsequent encounters therein. Being able to quickly layout these playable block-out floor-plans allows for rapid implementation, testing and iteration [Bea24].

A "Level Manager" object stores all of the floor blocks into an array. The Manager then processes the array, accessing each floor block in turn, and asking each floor block to call its own wall generation function. This function involves the Floor Block performing a trace on each of its 4 sides in order to ascertain as to whether there exists a neighboring floor block. If a neighbor is present, the floor block will not spawn a wall block on that side. If there is no neighbor detected, then a wall block is spawned on that side. Similar approaches have been applied in other procedural environment generators in the literature [HHACT14]. Once a floor block has check all 4 directions, it informs the Manager it has completed. The Manager then proceeds to the next floor block in its array. Once the Manager has processed every floor block, it then processes the array a second time, instructing each floor block to perform their ceiling spawning function which encapsulated the generated room.

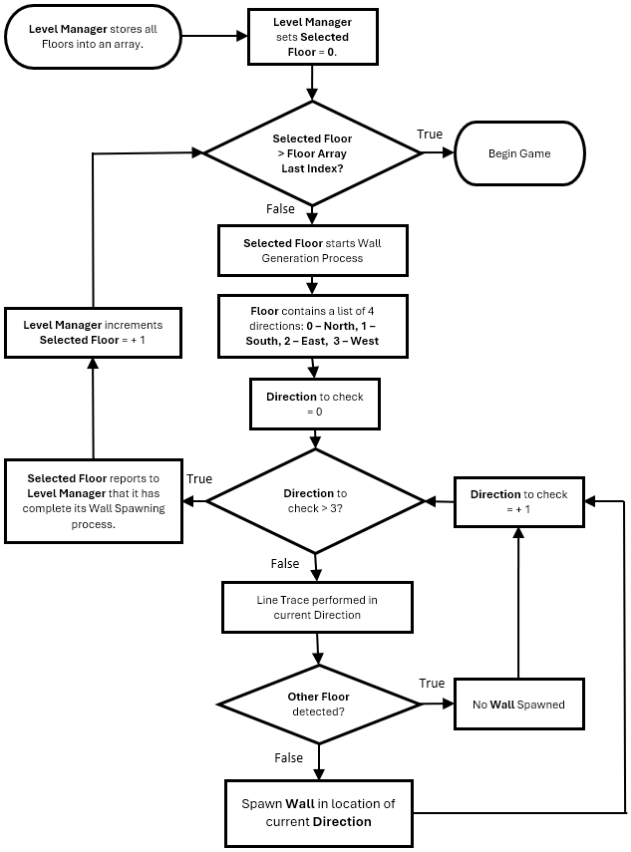

Figure 3: *Wall generation process algorithm as a flow chart.*

4. Conclusion

Peril offers an accessible, in-house framework that supports rapid level design and prototyping. Its streamlined procedural assistance tools remove extraneous level design tasks (such as wall placement) and by providing a full set of functioning game mechanics, students can focus on core design concepts such as encounter design, flow and pacing. As confidence grows, students can explore more advanced engine features, making Peril a strong foundation for both practical learning and progression in games education.

References

[Bea24] BEARDWOOD M.: *Fundamental Level Design and Analysis: A to B*. CRC Press, 2024. 2

[DP10] DORAN J., PARBERRY I.: Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games 2*, 2 (2010), 111–119. 2

[HHACT14] HEADLEAND C. J., HENSHALL G., AP CENYDD L., TEAHAN W. J.: *Randomised multiconnected environment generator*. Tech. rep., Bangor University, 2014. 2

[JH24] JAMES D., HEADLEAND C. J.: Games design frameworks: A novel approach for games design pedagogy. In *Proceedings of the 18th European Conference on Games Based Learning* (2024), Academic Conferences and publishing limited. 1

© 2025 The Author(s).
Proceedings published by Eurographics - The European Association for Computer Graphics.