TileTerror: A System for Procedurally Generating 2D Horror Maps

Arunpreet Sandhu[1]
Kyle Mitchell[2]
Joshua McCoy[3]
University of California, Davis
Davis CA 95616, USA
[1]asisandhu@ucdavis.edu
[2]kdmitch@ucdavis.edu
[3]jamccoy@ucdavis.edu

Abstract

Procedural content generation (PCG) algorithms pervade the world of game development. They are useful tools that allow creators to unshoulder a degree of authorial burden of non-trivial virtual worlds. Such a goal necessitates a well-defined m odel o f w hat t he a lgorithm i s p roducing. I n this work, a PCG system called TileTerror is presented that uses a model of horror constructed from a ludological analysis of well-known horror games to embed horror story features in a procedurally generated 2D tilemap. The horror features found in the ludology are separated into a distinct hierarchy: low-level features and high-level features. These features are used in conjunction with a procedurally generated tilemap to create an annotated map denoting candidate locations for strong horror moments. This system is evaluated using a series of pathfinding agents which score the annotated tilemap. TileTerror represents a first step into exploring what story elements can be embedded in procedurally generated constructs like 2D tilemaps.

Introduction

From fully-generated simulations with Dwarf Fortress (Bay12 Games 2006) to dungeon crawlers like Unexplored (Ludomotion 2017) to social worlds within Prom Week (McCoy et al. 2011), many video games today use procedural content generation (PCG) systems to create virtual worlds. The common thread among gameplay-pervasive PCG techniques is creating a cohesive experience for the player, such as generating content for gameplay or sound or narrative. Such PCG techniques include modeling physics to generate platformer levels (Summerville et al. 2020), modeling music to create themes for nonplayer character (NPC) music (Washburn and Khosmood 2020), and modeling of communication to generate dialogue (Horswill 2020).

PCG techniques are necessary for generating non-trivial virtual worlds. We say this to focus on how each game treats design differently; for example, a platformer is concerned with a different set of design goals than a shooter game. In turn, the virtual worlds generated by PCG algorithms are only as good as the model, and how well the model maps to the original design constraints and goals. In a similar vein, our work attempts to build cohesive worlds by modeling the horror genre for the procedural generation of annotated 2D tilemaps.

The horror genre has received academic attention from film and literary studies to interactive experiences (Perron 2009b; Therrien 2009). We chose horror because of the frequent use of environmental storytelling in levels to evoke feeling. We analyzed well-known horror games to build our model of low and high-level features. We restricted our model to two horror subgenres: slasher and psychological horror. We did this to keep the model concise and better understand the output of our PCG system, TileTerror. This should be considered the first step towards a more robust horror model. We define these features in the technical description of our system. TileTerror uses a design-oriented version of WaveFunctionCollapse (WFC), a PCG algorithm for tilemap generation (Sandhu, Chen, and McCoy 2019). Using a graveyard tileset as input to WFC, TileTerror generates a level, mapping low-level features to individual tiles and then annotates them for evaluation.

Thus, this paper contributes a PCG system that generates 2D tilemaps and evaluates their opportunities for horror. First, this paper describes a domain decomposition of horror aspects in video games, which features level design models that promote specific horror types a player experiences. Next, this paper describes the system architecture of TileTerror, which consists of two major parts: a preprocessing algorithm and a generator. Finally, five pathfinding agents analyze the tilemap for evaluation. Each pathfinding agent maps to a possible player type, inspired by the persona evaluations by Holmgård et al. (2019). Our system offers a unique bottom-up approach to this daunting task and an initial evaluation system for our generated artifacts.

Related Works

From the drama manager of Left 4 Dead (Valve 2008) to the character AI of Alien Isolation (Creative Assembly 2014) to the reactive agents within FEAR (Monolith Productions 2005), horror AI dips into different areas of design. Most horror AI focuses on character or pacing. But some games like Dead By Daylight (Behaviour Interactive 2016), an asymmetric player versus player horror game, place key game objects randomly on a 3D map. But the level generation is minimal at most, since objects are the only things changing, and not the parts of the 3D world. Inspired by this kind of level dynamic, TileTerror plans to expand to generating entire horror levels, not just randomizing objective locations like in Dead By Daylight.

Another inspiration for this work is Tanagra (Smith, Whitehead, and Mateas 2010) and how it generates levels through reactive planning via underlying beats, capturing 2D platformer pacing. Tanagra uses ABL, A Behavior Language (Mateas and Stern 2005), for beat generation, and Choco, a constraint solver, uses those beats to generate a satisfying configuration for the partially solved level, which inspired TileTerror. But we use WFC and focus on 2D top-down horror tilemaps.

Yet another inspiration for TileTerror is Subcutanean (Reed 2020), a procedurally generated psychological horror book, which focuses on two characters discovering more about themselves and their relationship. Written in a way where no two copies are the same, Subcutanean uses PCG techniques to keep the experience different for each reader. But the overall form of the story remains intact throughout the generation. Because of its horror roots and PCG techniques, Subcutanean is one of the closest siblings to TileTerror.

Games are the confluence of multiple creative fields, as Liapis et al. (2019) argue, and thus there is a heavy authorial burden that arises. PCG techniques lower this authorial burden by offloading work to an algorithm. Of all the creative outlets within games, world generation is one of the most fertile grounds for PCG, such as Perlin Noise to generate 3D worlds, as in Minecraft (Xbox Game Studios 2011), or mixing hand-authored content with the level generation, as in Dead Cells (Motion Twin 2017). Other games, such as Caves of Qud (Freehold Games 2019), use PCG techniques to generate the entire game world. But there hasn't been a technique, from our search, that focuses on how genre in particular impacts level generation.

The Horror Genre in TileTerror

This section discusses how TileTerror treats horror. First, we define horror from a games studies perspective in terms of two overlapping horror subgenres: slasher and psychological. Subgenres were chosen over the overall horror genre to avoid cluttering the high-level features with non-related genre tags when evaluating for subgenres. Then, an analysis of well-known horror games is provided which showcases how each game uses our identified set of low- and high-level features contained in the model.

A Word on Genre

Since the horror genre is the focus of TileTerror, this work defines the term genre as a loose definition, as described by Taylor (2009). By treating the genre as descriptive rather than prescriptive, TileTerror treats the horror genre as a collection of techniques and tropes. The jumpscare is one such technique that is commonly used across media, while the seemingly abandoned, isolated structure meant to trap individuals could be considered an example trope.

Slasher and Psychological Horror

Slasher horror features a character or cast of characters that face a threat that stalks them, eliminating characters throughout the story. The name has cinematic roots, with Psycho (Hitchcock 1960) being a foundational movie for the slasher genre. The setting of a slasher is often well-isolated from outside help or means of escape. Until Dawn (Supermassive Games 2015) exemplifies this by placing the main characters in a secluded location and eliminating characters based on the player's choices.

In contrast, psychological horror focuses on a character's mental, emotional, and psychological states to frighten and unsettle the player. Psychological horror often uses mystery, uncertainty, and unreliability surrounding the characters, plot, and setting to heighten tension and paranoia. A cinematic exemplar of this genre is The Thing (Carpenter 1982), throughout which the audience constantly questions who is the monster and who is still a human. A game exemplar is Pathologic 2 (Ice-Pick Lodge 2019), where the surrealistic, unreliable world leads the player to feel a sense of dread.

Ludological Analysis

Below are game examples that showcase both slasher and psychological aspects and the importance of environmental features, such as ambient creep, low visibility, jumpscare, and isolation—the four features used in our model.

Amnesia: The Dark Descent Amnesia: The Dark Descent (Frictional Games 2010) focuses on Daniel, who awakes in the Prussian Brennenburg Castle with no memory of his past. Daniel must navigate the decaying castle, escaping from surreal monstrosities while keeping himself sane. Amnesia draws on the balance between darkness and light, so much so that an entire mechanic revolves around it: If Daniel finds himself in a dark space without light, he will lose his sanity, visually warping his perception of the world. Ambient creep lives in every room within the castle, showing itself as maddening drops of water pouring in from broken stone ceilings, the oscillation between silence and startling creaks or screams, the constant dirge of halls once occupied. The immediate isolation of the castle–its thin corridors and crowded rooms–elevates the horrors by creating the necessary atmosphere in which monsters can successfully jumpscare the player.

Alien: Isolation Alien: Isolation places the player in the shoes of Amanda Ripley, who becomes trapped on a damaged space station, the Sevastopol. It is revealed to the player that Sevastopol has become the hunting ground for the Xenomorph, the antagonist from the Alien (Ridley 1979) film franchise. The player must complete objectives to get off Sevastopol alive, but the Xenomorph hunts the player throughout the ship, popping out of vents and shocking the player with jumpscares. The Xenomorph encounters become far more terrifying because of Sevastopol's design, like its use of low lighting to hinder sight or having the player move through grim settings, establishing a heavy atmosphere. These choices highlight the importance of low visibility and ambient creep to help deliver a horror atmosphere, making moments even more terrifying.

Misao In the 2D world of Misao (Sen 2011), the player follows a young woman named Aki, a classmate of the titular character Misao. A target of bullying, Misao goes missing three months before the start of the story. Aki begins hearing Misao's voice in class. Later, the school is torn from the world, trapping the students in a demonic dimension. Misao builds ambient creep and atmosphere by its horror tileset, which contains bones, blood, and tombstones. While Misao does not lean on light and darkness to block sight, the designers use objects to hinder vision. At one point, Aki finds herself in a dismal laboratory with large vats situated in the upper portion of the map, occluding the tiles around them. The player can guide Aki behind these vats to investigate what they hide, but a jumpscare awaits them: a shadowy fiend emerges and attacks Aki.

Transcending the Z-Dimension Alien Isolation's Sevastopol station, Amnesia's Brennenburg Castle, and Misao's supernatural school succeed in environmental storytelling—in that they provide enough set decoration to establish a horror atmosphere. Even though Sevastopol and Brennenburg Castle benefit from 3D environments and lighting systems, 2D horror manages to create a horror atmosphere by using appropriate tilesets and similar horror techniques, like jumpscares and isolation.

Technical Description

This section provides a walkthrough of our technical implementation, as shown in Figure 1. We begin by describing the various components involved in the generation process. First, there is a degree of preprocessing involved with the selection of a tileset that exhibits the low-level features identified in our ludological analysis; identifying the intrinsic local constraints of the chosen tileset; and designing any partial patterns of tiles we wish to be present in the output. This work uses a graveyard-themed tileset[1]. Second, both the original tileset with its intrinsic constraints and the added constraints of any partials are given as input to WFC, which generates the remaining unsolved sections of the output. Third, a feature detector analyzes the solved tilemap, creating an annotated map of low- and high-level features combined, which are visualized in Figure 3. Lastly, we provide a description of our evaluation process, which involves a series of pathfinding agents that solve the fully annotated tilemap and score it based on its potential for horror.

Preprocessing: Local Constraints

Critical components of WFC are hand-authored neighbor pairings that are made with design domain knowledge. In our work that uses the graveyard tileset, for example, it makes reasonable sense that tombstone tiles can appear next to each other. Thus, WFC uses these neighbor constraints and the tileset to generate graveyard tilemaps. The simple stress test for this work was generating a tilemap with a row of tombstones and one mausoleum; however, TileTerror wasn't always able to generate this particular construction. To both help WFC generate appropriate levels and to allow the designer to have more agency in the generation process, our system includes support for additional preprocessing in the way of designerly partial patterns, or more simply, partials.

[1]https://angrysnail.itch.io/pixel-art-graveyard-tileset

Preprocessing: Solving for Partials

To overcome multi-tile design constraints, TileTerror introduces partially solved tilemaps. Figure 2 shows an example of a tilemap partial; they are hand-authored designs, like the set of tiles that collectively illustrate a unified mausoleum. To achieve this effect, TileTerror takes the tilemap partials and places the partials within the map, ensuring no overlap. TileTerror generates a 2D matrix, representing a 2D tilemap, setting each cell to false, stating that the cell hasn't been used yet for the partial generation. The algorithm randomly chooses a location, checking if the tilemap partial can fit. If the partial is too big, or the space already has another partial, the algorithm will move to a new location, trying to find space for the partial. If there is no space, the solver will move on to another partial until there are no more. If there is space, the algorithm will resolve the undecided sub-matrix to the partial. After generating this partially solved tilemap, TileTerror inputs the results into WFC.

Generation: WaveFunctionCollapse

WaveFunctionCollapse then takes in both the partially solved map and the local neighbor constraints. Using the partially solved map, WFC generates a new 2D matrix, where each cell contains an array of true booleans, effectively creating a 3D tensor. Each boolean represents a tile from the tileset.

WFC generates a choice heuristic by using an entropy calculation for each cell. The value represents how stable a cell is, and the lower the entropy value is, the closer a cell is to stability. Stability in WFC refers to how close a boolean array is to containing only one true value. WFC uses the entropy value when choosing matrix cells, prioritizing cells with the lowest entropy score.

Upon generating the entropy value, WFC generates a set of neighbors for each tile. This set contains the four cardinal directions for each tile. Each tile must have a neighbor for all four directions, and if a tile does not have a neighbor for one of the four, it will be incompatible with whatever neighbors it borders, at least with the WFC variant TileTerror uses.

WFC chooses a random cell and chooses a random boolean index to keep true, switching all other booleans to false. WFC updates the adjacent neighbors based on the neighbor constraints, validating each neighbor and ensuring it is still compatible with the newly chosen tile. If there are any incompatible tiles, WFC sets their boolean flags to false within the boolean array, propagating the removed tiles to the adjacent tile's neighbors, beginning the validation process once more. This removal propagation stops when all matrix cells are compatible with their adjacent neighbors; or

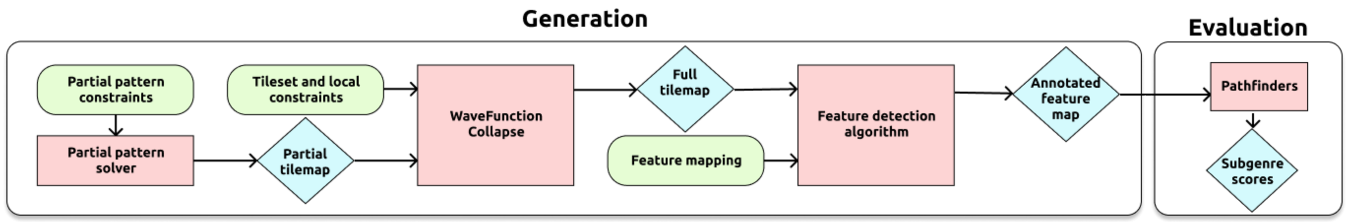


Figure 1: A system diagram for TileTerror. Each box represents a different phase of the pipeline. Phases highlighted in green, with rounded edges, are those that require hand-authoring. Phases in red, in square boxes, are those that are performed algorithmically. Phases in blue, in diamonds, represent output.

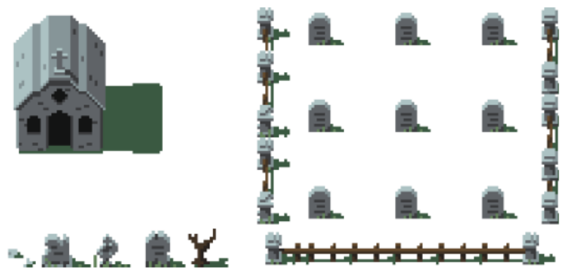


Figure 2: Top left: A mausoleum partial constructed with 16 separate tiles. Right: A graveyard partial with 42 tiles. Bottom left: A series of tiles illustrating low-level features.

a cell reaches an incompatibility point, where every boolean flag within a cell's array is false. If this happens, then the generation process has failed, and WFC will return nothing.

If there are no incompatibilities, WFC chooses another cell, using the entropy calculation as a guide. This choice-propagation pattern continues until every cell's array has only one true value or an incompatibility occurs. After generation, feature annotations are added to the map, searching for high-level features and appending them as they are found.

Generation: Features and Detection

This paper identifies four features an environment can use to generate horror: ambient creep, low-visibility, jumpscare potential, and isolation. As a caveat, these are not the only features a horror environment can contain, but the four we believe are the basis for other features. All features we describe here are added to an annotation matrix which directly maps to the generated tilemap, cell by cell.

Low-Level Features For this paper, low-level features are intrinsic qualities, like a tombstone on the tile. Tiles can also have multiple low-level features, such as a tree being creepy and blocking the line of sight, i.e. low-visibility. The first low-level feature is ambient creep which describes the tile's atmospheric horror quality. This work assigns ambient creep to tombstones, dead trees, and bone tiles. Figure 2 gives an example of ambient creep with the first four tiles on the left: bones and tombstones. The next low-level feature is low-visibility, which describes how a tile obscures a player's line of sight. Given that the horror genre makes heavy use of obscuring sight, low-visibility becomes the more important of the two low-level features for high-level feature detection. Figure 2 contains an example of low-visibility with the far right tile, a dead tree. Finally, the low-level feature set includes a traversability feature. It should be noted that traversability is not a horror-specific domain feature, but a feature needed for evaluation.

High-Level Features While low-level features focus on single tiles, high-level features focus on the relationship between tiles. To find these relationships, detectors are used on each tile within the tilemap, inspecting both the tile and its neighbors. If the detector discovers a high-level feature, then the detector appends the feature to the tile's feature set. This work uses the following high-level features: jumpscare and isolation. For this work, the combination of low-visibility and traversability determines the jumpscare potential and isolation potential of a tile. Figure 3 shows the high-level features through heatmaps of a generated tilemap.

To determine a tile's jumpscare value, a tile is checked for its low-visibility. If there is no low-visibility property, no changes occur. If the tile has a low-visibility feature, traversability becomes the next checked feature. If the tile is also traversable, then it is marked as a location for a potential jumpscare. If the tile is not traversable, the tile's immediate neighbors are checked, seeing if any of them are traversable. If those neighbors are traversable, the detector marks the neighbor as a potential jumpscare location.

The next high-level feature is isolation, which can be elicited in two ways: one, if a tile is surrounded by a number of low-visibility tiles, it is marked as isolated; and two, if a tile is not neighboring any tiles with features other than traversability, it is marked as isolated. This duality comes from the idea that isolation is not only linked to claustrophobia, but also to a general sense of vastness. To detect isolation, the detector moves through the matrix, checking if a tile is traversable. If the tile is not traversable, the detector moves on to the next tile. If the tile is traversable, the immediate neighbors are gathered, evaluating how many of them have a low-visibility feature. If the number of low-visibility neighbor tiles reaches a threshold, the detector annotates the chosen tile with an isolation feature. This work sets the threshold to 3. If a tile is next to another isolation tile, then it will be marked with a temporary lonely tag. This tag is used for increasing the tile's isolation score.

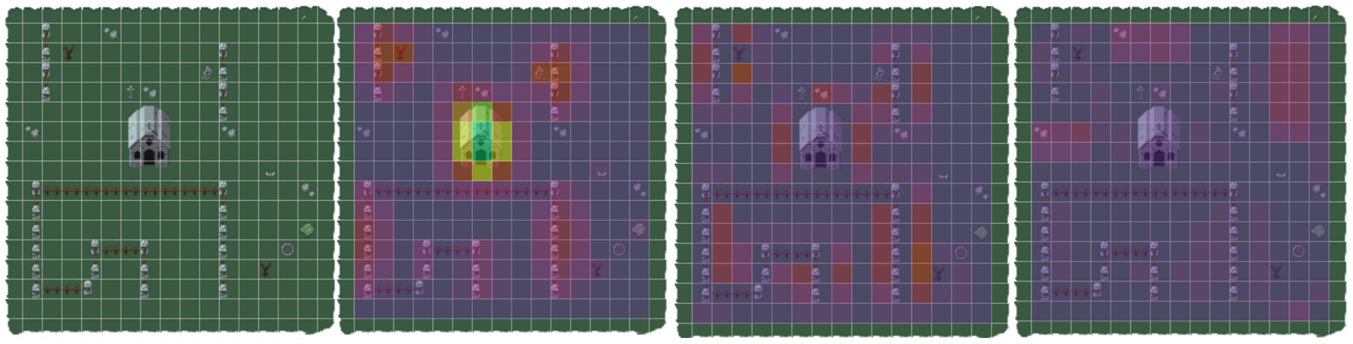


Figure 3: A "horrorscope" of high-level features being detected in a generated tilemap. Far left: The generated tilemap. Middle left: A heatmap representation of low-level features (ambient creep and low-visibility). Middle right: A heatmap representation of the jumpscare potential for tiles. Far right: A heatmap representation of the isolation potential for tiles.

Evaluation

Our evaluation focuses on the expressive range of TileTerror through pathfinding agents that solve the map, scoring for both slasher and psychological horror potential. Five agents start at the same point: the origin, or the upper left corner of the map. Each agent navigates the map, reaching the goal of the furthest corner from the start point: the bottom right corner for these experiments. The pathfinding algorithm TileTerror uses is the A* algorithm (Hart, Nilsson, and Raphael 1968), favoring the lowest score for the next step.

Pathfinding Agents and Scoring Mechanism

The first agent employed is a speed-running pathfinder, which takes the shortest path to the goal. This agent takes the Euclidean distance function as its score. The second agent, the completionist, takes the longest route by scoring the reciprocal of the Euclidean distance. These two agents showcase what parts of the level might need a designer's attention. The shortest path helps show developers where speedrunners might go, and the longest path gives the developers a chance to place rewards for those who complete the entire level or add more opportunities for horror.

The three other agents solve both for the goal and prioritizing either slasher or psychological horror features. The first of the horror-focused agents is a horror-avoidance agent or the scaredy-cat agent. The scaredy-cat agent's scoring function sums up all the feature scores across ambient creep, low-visibility, jumpscare, and isolation, using the sum as the score. If a tile has more horror features, its score is higher, and the agent tries to avoid the tile. If there is no better option, then the agent must traverse over the scary tile. The distance from the goal is also a part of the score, making the agent find the shortest path if there is no nearby horror. This score is given as distance times the total horror score.

The fourth agent is a slasher horror-prioritized agent, which searches for all horror, but it prizes jumpscares over all else. Unlike the scaredy-cat agent, the horror-prioritized agents use the reciprocal of the total horror score, prioritizing horror potential rather than avoiding it. By not removing the total horror score (THS), TileTerror co-opts all of the horror values together, summing them up for each tile. But, each feature is weighted differently. The total horror score for the slasher agent is given as: $THS = (1 * AC) + (4 * JS) + (2 * LV) + (0.1 * I)$. For this formula, we denote ambient creep with the variable AC, jumpscare with the variable JS, low-visibility with the variable LV, and isolation with the variable I. Finally, The THS is multiplied by the distance from the goal for the total score.

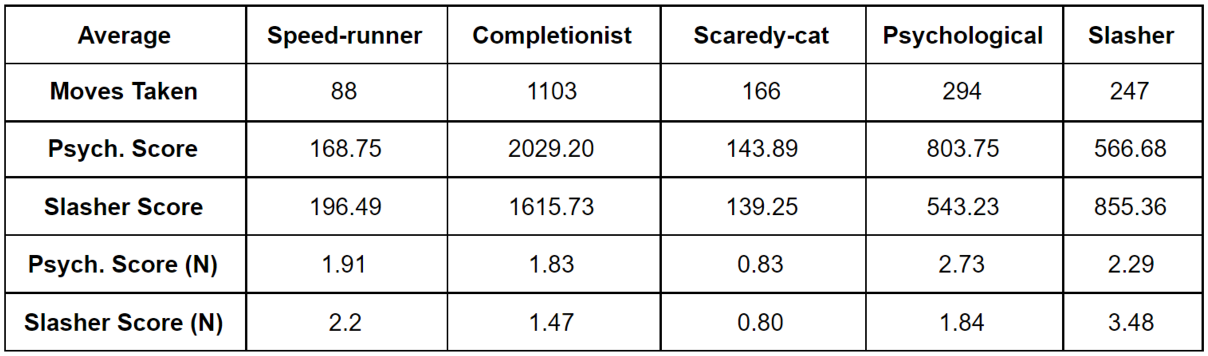
The final agent is a psychological horror-prioritized agent that seeks out isolation and ambient creep, devaluing jumpscares. Below is a formula for this agent's valuation of total horror, altering the weights to reflect those features we think are salient to psychological horror: $THS = (2 * AC) + (0.1 * JS) + (1 * LV) + (4 * I)$. Its final score is given by the distance times the reciprocal of the above total horror score.

TileTerror reconstructs each agent's path and evaluates each tile for its slasher and psychological score. The first is the slasher score, which only takes into account the two low-level features and jumpscare. The psychological score takes into account the two low-level features and isolation. These scores are the basis of TileTerror's expressive range, further elaborated in the results section.

Results

We ran TileTerror for 10,000 iterations, producing 9,613 maps, which the pathfinding agents then evaluate. Figure 4 contains a summary of our results. Because the starting and the ending locations may not have a reachable path and WFC can fail, there is a probability of failure within TileTerror—which in our experimentation was approximately 0.04. In case of failure, TileTerror can produce a new result since we envision this tool for offline generation.

The completionist path results seem strange at first, with psychological being the greater of the two. Yet, TileTerror's output often has large swatches of empty tiles that increase the psychological horror, which can be seen as a more diffused experience. And slasher horror requires tiles with objects on them, like gravestones or buildings or fences, to appear within a level. Thus, an agent with more moves taken will, on average, experience more psychological horror than a slasher because TileTerror seems to generate more open levels. But, these parameters are mutable, letting the devel-



Average	Speed-runner	Completionist	Scaredy-cat	Psychological	Slasher
Moves Taken	88	1103	166	294	247
Psych. Score	168.75	2029.20	143.89	803.75	566.68
Slasher Score	196.49	1615.73	139.25	543.23	855.36
Psych. Score (N)	1.91	1.83	0.83	2.73	2.29
Slasher Score (N)	2.2	1.47	0.80	1.84	3.48

Figure 4: Results of 10,000 iterations of TileTerror for a 40x40 tilemap. Values denoted by a paranthetical N are normalized by distance.

oper change how often a tile appears within a level. Thus, the completionist path scoring gives developers a warning sign if the generation parameters need tuning for either psychological or slasher horror.

Surprisingly, the shortest path agent was only a few points away from the scaredy-cat agent, but the scaredy-cat agent took twice as long to complete the maps. This increase in moves could indicate that when the scaredy-cat agent sensed horror, they ran away, increasing steps in the map but not increasing their overall score. And thus, it seems there is often a less terrifying path within the level that developers can discover. With this path, developers can alter the level's flow by adding more scares in that section or blocking the path entirely.

Between slasher and psychological, it was unexpected to see a slight bump for the slasher score. However, when comparing the moves taken, the same argument from the longest path can be used here: The slasher agent roams the map longer, hitting more psychological horror, while the psychological agent spends less time on the map overall. Figure 4 illustrates each agent's slasher and psychological scores normalized by distance taken.

As shown by the normalized scores, each of the agents does what is expected. What was interesting seems to be the tradeoff between moves taken and the overall slasher score. It seems the longer an agent is in the level, the more psychological horror they gain. This result corroborates the shortest path agent having a lower overall psychological score while having a higher slasher score and the longest path agent having a higher psychological score than the slasher.

Future Work

We view this work as a stepping stone to more robust bottom-up generation and feature detection algorithms. The first step is to create a denser set of features for our model, looking towards game studies literature such as Perron's study of videoludic horror games (2009a) and integrating temporal features such as tension into generation and evaluation. By introducing temporal features, we hope to eke out a sense of flow (Csikszentmihalyi 1990) within the map and introduce flow-like generation within this system. By introducing a flow-focused generation procedure, we hope to decrease the number of maps with long tension segments, ensuring the player doesn't get worn out from a constant state of agitation.

Given that horror often employs adversaries that characters encounter, encoding features like enemy placement would enhance the tilemap and be another reasonable next step. To this end, we envision including a set of adversarial pathfinders in our collection of navigational agents. By doing so, the evaluation agents can attempt to move through the map given a starting and ending location, while the adversaries give chase or lie in wait in certain areas. By discovering where the adversaries and evaluation pathfinders meet, a possible "confrontation" module can be used for the generation, nudging the map towards generating a more deliberate setting for the confrontation.

Conclusion

This work describes a bottom-up architecture for evaluating horror story devices in procedurally generated tilemaps. First, TileTerror examines two subgenres of horror—slasher and psychological—to acquire a baseline understanding of low- and high-level features for tilemap evaluation. TileTerror defines these low-level features as ambient creep, traversability, and low-visibility—horror qualities discovered from a preliminary examination of the horror genre. Combining the low-level features, TileTerror creates a set of high-level features for evaluation.

These high-level features map directly to commonly used horror devices: jumpscares and isolation. Our technical implementation uses WFC to generate a tilemap, the template for the low-level matrix, and detectors run through the feature matrix, finding high-level features. Finally, TileTerror uses a system of evaluation that scores the resulting tilemaps on each map's horror potential through a set of navigational agents.

References

Bay12 Games. 2006. Dwarf Fortress.

Behaviour Interactive. 2016. Dead By Daylight.

Carpenter, J. 1982. The Thing.

Creative Assembly. 2014. Alien Isolation.

Csikszentmihalyi, M. 1990. *Flow: The Psychology of Optimal Experience.* Harper and Row.

Freehold Games. 2019. Caves of Qud.

Frictional Games. 2010. Amnesia: The Dark Descent.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Hitchcock, A. 1960. Psycho.

Holmgård, C.; Green, M. C.; Liapis, A.; and Togelius, J. 2019. Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics. *IEEE Transactions on Games* 11(4):352–362. Conference Name: IEEE Transactions on Games.

Horswill, I. 2020. Generative Text using Classical Nondeterminism.

Ice-Pick Lodge. 2019. Pathologic 2.

Liapis, A.; Yannakakis, G. N.; Nelson, M. J.; Preuss, M.; and Bidarra, R. 2019. Orchestrating Game Generation. *IEEE Transactions on Games* 11(1):48–68. Conference Name: IEEE Transactions on Games.

Ludomotion. 2017. Unexplored.

Mateas, M., and Stern, A. 2005. Structuring Content in the Façade Interactive Drama Architecture. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2005)*, volume 3.

McCoy, J.; Treanor, M.; Samuel, B.; Reed, A. A.; Mateas, M.; and Wardrip-Fruin, N. 2011. Prom Week: Designing past the game/story dilemma.

Monolith Productions. 2005. F.E.A.R.

Motion Twin. 2017. Dead Cells.

Perron, B. 2009a. *Horror Video Games: Essays on the Fusion of Fear and Play*. McFarland, 1st edition.

Perron, B. 2009b. Introduction: Gaming After Dark. In *Horror Video Games: Essays on the Fusion of Fear and Play*. McFarland, 1st edition edition.

Reed, A. 2020. *Subcutanean*. Independently published, 1st edition.

Ridley, S. 1979. Alien.

Sandhu, A.; Chen, Z.; and McCoy, J. 2019. Enhancing wave function collapse with design-level constraints. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 1–9. San Luis Obispo California USA: ACM.

Sen. 2011. Misao.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: a mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10*, 209–216. Monterey, California: ACM Press.

Summerville, A.; Sarkar, A.; Snodgrass, S.; and Osborn, J. 2020. Extracting Physics from Blended Platformer Game Levels.

Supermassive Games. 2015. Until Dawn.

Taylor, L. 2009. Gothic Bloodlines in Survival Horror Gaming. 46–61.

Therrien, C. 2009. Games of Fear: A Multi-Faceted Historical Account of the Horror Genre in Video Games. In *Horror Video Games: Essays on the Fusion of Fear and Play*. McFarland, 1st edition edition.

Valve. 2008. Left 4 Dead.

Washburn, M., and Khosmood, F. 2020. Dynamic Procedural Music Generation from NPC Attributes. In *International Conference on the Foundations of Digital Games*, 1–4. Bugibba Malta: ACM.

Xbox Game Studios. 2011. Minecraft.