Desire Path-Inspired Procedural Placement of Coins in a Platformer Game

Anurag Sarkar, Varun Sriram, Riddhi Padte, Jeffrey Cao, Seth Cooper
Northeastern University, Boston, Massachusetts, USA
{sarkar.an, sriram.v, padte.r}@husky.neu.edu, jhccao@gmail.com, scooper@ccs.neu.edu

Abstract

Many games feature collectible items that are manually placed by a designer. In this work, we developed an algorithm, inspired by *desire paths*, for automatically placing collectible coins in a platformer game. Desire paths are paths naturally formed where people walk, rather than those laid down artificially, and are often the shortest or easiest route between an origin and destination. Our algorithm uses player trajectories to find paths along which to place the coins for each level. We ran an experiment comparing path-based placement to other placement methods. Although we did not find a difference in total time spent playing or likelihood of finishing the game, our results suggest that path-based placement leads to players collecting more coins in less time than with designer or randomly placed coins. Further, we found that players played similarly when coins were either path-based or there were no coins, and similarly when coins were either placed by a designer or randomly.

Introduction

Many games have levels containing collectible items (such as coins, bonuses, or power-ups) that may serve different purposes within the game. Often, collecting these items is a secondary objective, either related or unrelated to some primary objective (such as completing the level). Thus, automating the placement of such level elements via procedural methods could save the designer's time in authoring the level by allowing the designer to focus on the primary goals of the level, and then automatically adding collectibles after the level has been designed.

However, previous work by Andersen et al. (2011b) has shown that placing such secondary objectives within levels may *decrease* engagement if the placing is not done in a manner that serves the primary objective of the levels. That is, to be effective and not cause any side effects detrimental to player experience, such procedural placement algorithms should take into account the level's primary objective and how the player achieves that goal.

Platformer games usually consist of levels where the primary objective is to traverse through a sequence of platforms and hazards to reach some goal at the end of the level. For levels where there are multiple paths from start to finish, absent other objectives, some types of players may prefer to quickly achieve progress by following the path that is shortest or easiest to traverse, i.e. a path that allows them to complete the level quickly while having to contend with relatively few hazards (Bartle 1996; Yee 2007). Note that such paths may not necessarily be enforced by the level's design but may be discovered after multiple playthroughs of a level.

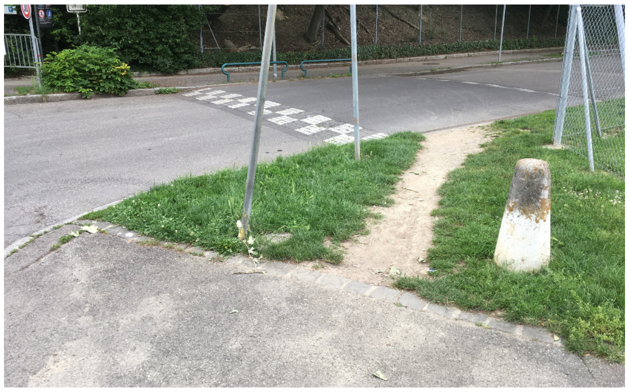


Figure 1: A real-world desire path: the small dirt path leading to the crosswalk.

In nature, paths that are created organically by the footfall of people where they naturally tend to walk, rather than by artificial construction, are called *desire paths*. These paths usually represent a shortcut or a route that is less circuitous than a constructed sidewalk or other walkway. An example is shown in Figure 1. Borrowing this concept for platformers, we use desire paths to refer to the paths through a level that players are likely to traverse on account of being simple, direct routes from start to goal. Thus, placing secondary level elements, such as collectibles, along these paths may help circumvent the issues associated with their placement.

In this work, we extracted such desire paths for each level in the platformer game *Iowa James: Treasure Hunter* (screenshots shown in Figure 2) using player trajectories and placed collectible coins along these paths. We compared the path-based method of coin placement with coins placed by a designer, with coins placed randomly, and with levels having no collectible coins. Our findings indicate that placing

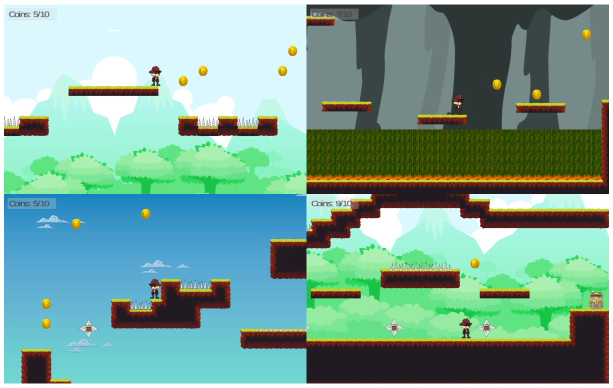


Figure 2: Examples of Iowa James on his adventures.

coins along paths leads to players collecting more coins in less time than with designer or randomly placed coins. When compared to having no coins, designer placed coins led to fewer levels being completed with more time spent in each level, while we did not find such differences for path-placed coins. Overall, we found no differences in player behavior when coins were placed along paths or when there were no coins. Similarly, we found no differences in player behavior when coins were placed manually by a designer or placed randomly, indicating that in some cases it may be possible to save designer time simply by placing collectibles randomly.

Related Work

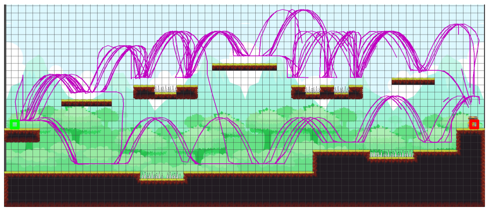
Desire paths in design. While most often directly used in the context of urban planning, desire paths have received attention in more general design as representative of what people actually do rather than what designers might prefer or expect (Kohlstedt 2016). Desire paths have been used as a metaphor in areas like product design (Myhill 2004), software interaction design (Zhang, Padman, and Levin 2014), and more general social behaviors (Nichols 2014).

Secondary Game Objectives. Secondary objectives form a fundamental component in the design of many games. Initial work by Andersen et al. (2011a) found that the inclusion of secondary objectives, such as coins, could negatively impact engagement. However, further work (Andersen et al. 2011b) found that secondary objectives that supported the game's main objective could positively impact player behavior. Specifically, placing coins along the main path through levels in a platform game led to players completing more levels and spending more time playing than having off-path coins in the levels. This informed our use of player trajectories through levels to procedurally determine desirable paths for levels and place coins along these paths.

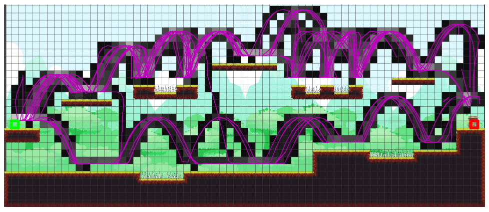
Procedural Content Generation. Procedural content generation (PCG) (Shaker, Togelius, and Nelson 2016) may be defined as the process of automatically creating content using procedural or algorithmic techniques. Different types of such methods—e.g. search-based techniques (Togelius et al. 2011) or constraint satisfaction (Smith and Mateas 2011)—have found widespread use in the generation of a variety of game content, such as platformer levels (Snodgrass and Ontañón 2014), action-adventure missions (Dormans 2010), strategy game maps (Liapis, Yannakakis, and Togelius 2013a), and space shooter weapons (Hastings, Guha, and Stanley 2009). One of the primary motivations, among many, for such procedural techniques is to assist or save the designer's time in authoring game content (Smith 2014b). While much PCG work focuses on generating entire levels, in our work, we focus on placing collectibles within existing levels. By automating the design or creation of secondary or tertiary game artifacts, the designer can then focus more thoroughly on the principal components of a game's design.

Data-Driven PCG. Smith (2014a) defines a *data-driven PCG system* as one that "uses external data to inform the generation of content." Data-driven approaches have found wide use for player modeling. Zook et al. (2012) use data-driven player models for generating missions while Zook and Riedl (2012) do so for dynamic difficulty adjustment. Closer to our work is that of Pedersen, Togelius, and Yannakakis (2010) who build data-driven player models, which can then be utilized to inform level design. Similarly, Jennings-Teats, Smith, and Wardrip-Fruin (2010) used players playing through platformer level chunks to train a model of player experience rather than craft one by hand. The model was then used to generate platformer levels of desired difficulty. Such work falls more squarely in the field of PCGML (Summerville et al. 2017), where generative models trained on existing game data are used for creating new content. Our data-driven model differs from most traditional models in that rather than having a model of each individual player, we utilize a model of player movement with all players considered in aggregate. More recently, data-driven approaches have been applied not just to inform level design, but to generate entire adventure games (Barros, Liapis, and Togelius 2016).

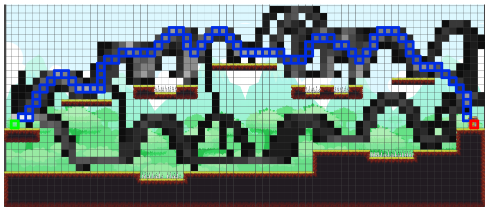
Mixed-Initiative PCG. While purely automated content generation systems are useful, we often require procedural systems that afford authorial control during generation. These systems enable a human designer to guide the PCG system towards content with certain desired properties. Such systems are categorized as *mixed-initiative PCG* since they combine the creativity and judgment of a human designer with the ability of a generative system to quickly produce a large amount of output. Smith, Whitehead, and Mateas (2011) developed the mixed-initiative generator *Tanagra*, which uses reactive planning and constraint satisfaction to create 2D platformer levels that a human designer can then edit. Smith et al. (2011) also created the *Launchpad* level generator, which uses a grammar-based method and the notion of rhythm groups to generate playable platformer levels that the designer can then tune to define player paths through the level as well as frequency of level components. Similarly, Shaker, Shaker, and Togelius (2013) created *Ropossum*, a level generator for *Cut the Rope* that takes into account designer input. Another such system, *Sentient Sketchbook* (Liapis, Yannakakis, and Togelius 2013b), generates playable levels from simple maps sketched out by a human designer. Baldwin et al. (2017) also described a mixed-initiative system that uses game design patterns to create dungeons using genetic algorithms. Our coin placement algorithm can con-



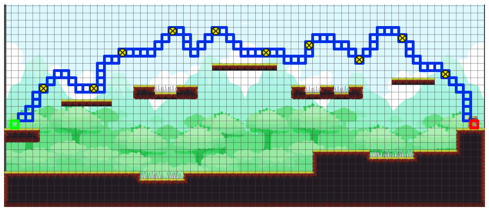
Inputs: The inputs to the coin placement algorithm are the number of coins to place, grid cell definition, starting and ending locations, and list of player trajectories from players who won the level.



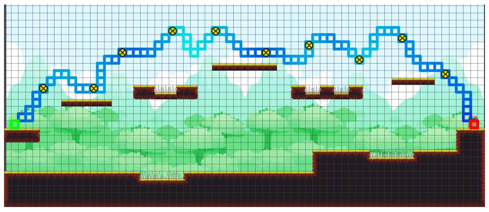
Step 1: For each grid cell $c = (x_c, y_c)$, count the proportion w_c of winning trajectories that pass through that cell. Thus, if no winning trajectory went through a cell, $w_c = 0$ and if all winning trajectories went through a cell $w_c = 1$.



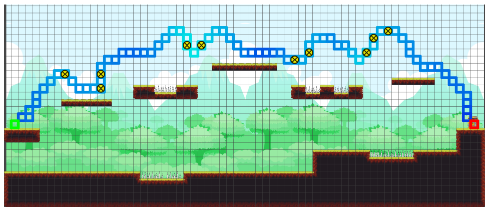
Step 2: Find the lowest cost path through the grid (via A*) from the starting grid cell to the ending grid cell. It is possible to move to any 8-connected neighbor, and the cost to move from grid cell s to t is $\frac{|t-s|}{max(w_t,\epsilon)^2}$, where ϵ is some small number.



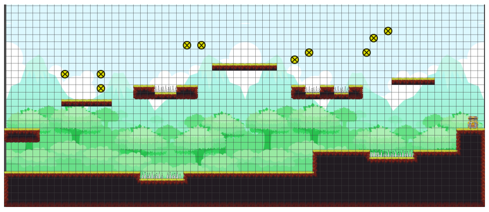
Step 3: Place the required number of coins evenly spaced along the path.



Step 4: For each grid cell p_i along the path, compute a priority value r_i, using an estimate of the local deviation from a straight line as $r_i = |p_i - 0.5 * (p_{i-1} + p_{i+1})|^2 + |p_i - 0.5 * (p_{i-2} + p_{i+2})|$.



Step 5: Given a coin at p_i, if $r_{i+1} > r_i$ and there is no coin at p_{i+1} or p_{i+2}, move the coin to p_{i+1}; or if $r_{i-1} > r_i$ and there is no coin at p_{i-1} or p_{i-2}, move the coin to p_{i-1}. Iterate through all coins, moving each if needed, until no coin moves.



Outputs: The algorithm outputs a list of coin locations.

Figure 3: Steps in the coin placement algorithm. $|v|$ denotes vector magnitude.

stitute a part of a mixed-initiative PCG system such as the ones above since the coin placement is automated while the remainder of each level is hand-authored.

Game and Recruitment

For this work, we used a platformer game we developed in Unity called *Iowa James: Treasure Hunter* (Figure 2). The player controls an avatar that can move and jump. The goal is to navigate a level to reach a treasure chest at the end. Upon reaching the chest, the player wins the level and moves on to the next one. There are several types of hazards that can be encountered along the way, including pits, spikes, timed retracting spikes, and moving spike balls. The player dies upon touching a hazard and starts the current level over from the beginning. The player has unlimited lives, so the only way to lose the game is to stop playing.

The game has 14 levels that cover a variety of layouts and usually contain several ways to navigate from the start to the goal. The first level is meant to be a simple introductory level. If the player completes all the levels, they are taken to a game over screen. The game also features collectible coins, which the player can collect by touching. Each level contains 10 coins. A UI element displays how many coins out of the total possible the player has collected so far in the level. If the player restarts a level due to encountering a hazard, the coins they had collected so far remain collected. The coins do not have any effect on gameplay.

To recruit players, we posted Human Intelligence Tasks (HITs) on Amazon Mechanical Turk (MTurk). Prior work has shown that players recruited through MTurk respond similarly to game design experiments as volunteer players do (Sarkar and Cooper 2018). The HIT was titled *Platformer Game* with the description *Play a platformer game! Run and jump to collect treasure at the end of each level!* and paid $0.50. The HIT provided the payment code upfront, thus not requiring the game to be played at all for payment. Hence, any time spent playing the game on the HIT could be considered voluntary. We found that three-quarters of people who accepted the HIT, went on to play the game, with the rest simply taking the payment and not playing. Before playing, players were given a short set of instructions on how to play. To ensure that the same information could be provided across different versions of the game, there was no mention of coins in the HIT details or instructions.

Path Coin Placement

The coin placement algorithm we developed generates coin locations in a level based on winning player trajectories. The algorithm places coins based on grid cells; in this work, we used a tile-based game and each tile was a grid cell. The algorithm takes as inputs the number of coins to place, the grid cell definition, starting and ending locations, and a list of player trajectories from players who won the level. Note that the algorithm does not know which grid cells are impassible to the player (although this could be incorporated), instead relying on player paths to determine where the player can go. The output is a list of coin locations.

An illustrated overview of the algorithm is shown in Figure 3. We used several heuristics to guide this coin placement. The main heuristic we used was:

- Coins should be on a single path through the level; the path should have been traversed by successful players (*Steps 1 and 2*). This is motivated by the work of Andersen et al. (2011b).

We used additional heuristics that were based on our subjective experience from playing games with collectible coins. It is possible other heuristics could work just as well or better (for example, simply placing coins randomly along the path). We consider further study of such additional heuristics as future work.

- Coins should be well distributed throughout the level (*Step 3*). This is motivated by making it more likely for players to have coins to collect all along the path through the level.
- Coins should be clustered near curves and arcs in the path (*Steps 4 and 5*). This is motived by the groupings of coins in games such as *Super Mario Bros.*

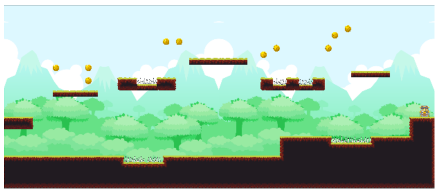
To gather player trajectories, we ran a HIT that recruited 200 participants through MTurk, of which 160 played the game. The version of the game used to gather trajectories had no coins or associated UI. In order to distribute plays across the levels, after the introductory level, the remaining levels were served to players in a random order. In this HIT, we gathered trajectory data from players at high temporal resolution (10 Hz) and filtered out winning trajectories with missing events, as well as gaps within individual winning trajectories. For the introductory level, which was served to all players, we used 77 winning trajectories for coin placement. For the remaining levels, we used from 8 to 22 winning trajectories for coin placement. The data from this HIT was only used for coin placement, and was not used for the evaluation described below.

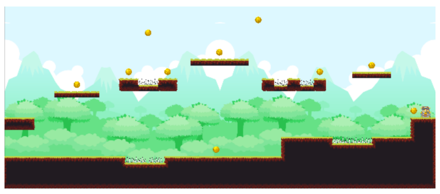
Experiment

To evaluate different coin placement methods, we ran a second HIT that recruited 1600 participants through MTurk, of which 1226 played the game. In the versions of the game used in this experiment, levels were served in a fixed order. To place levels in rough order of increasing difficulty, the ordering was based on the player success rate from the previous HIT, with levels having lower success rates placed later in the order.

Players were randomly assigned into one of four versions of the game with different coin placement, based on the following conditions:

- `NONE`: No coins were in the game. The coin count was hidden from the UI.
- `PATH`: Coins were placed using the path coin placement algorithm described above and depicted in Figure 3.
- `DSGN`: Coins were placed manually by a game designer working with our research group. We asked the designer to place the coins wherever he wanted, as long as the coins were reachable by the player. The designer spent around 4 hours placing coins.





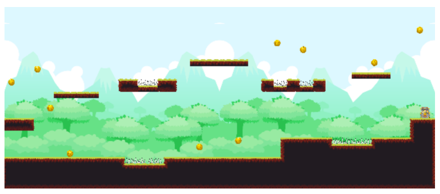


Figure 4: Examples of coin placement for each method used in the experiment.

	NONE	PATH	DSGN	RAND
Levels Won[†]	5^a	4^{ab}	4^{bc}	3^c
Finish Rate (%)	8	10	6	6
Total Time (s)	224	216	226	174
Per-Level Time (s)[†]	38^a	37^a	47^b	41^{ab}
Total Coins[†]		38^a	25^b	26^b
Per-Level Coins[†]		8^a	6^b	6^b

Table 1: Summary of data and statistical analysis. Medians are given, except for *Finish Rate*. A significance level of $\alpha = .05$ was used. Daggers[†] show significant omnibus tests. Superscripts^{abc} show significance groups in post-hoc tests; i.e, conditions that share a superscript were not significantly different. Significant comparisons had $p < .001$, except for the post-hoc comparisons in *Levels Won*: NONE–DSGN, $p = .003$, and RAND–PATH, $p = .031$, and *Per-Level Time*: NONE–DSGN, $p = .009$.

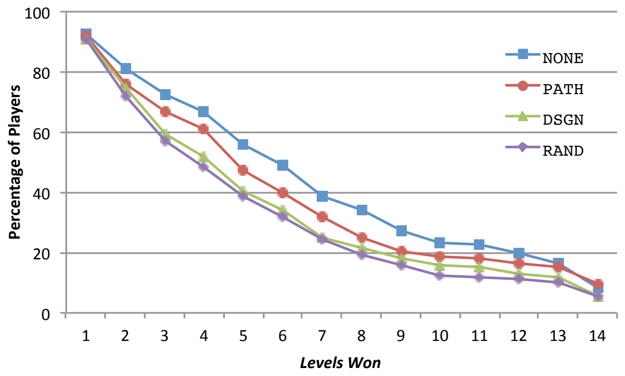


Figure 5: Survival plot showing the percentage of players who won a given number of levels. Note that although PATH and DSGN have the same median value for *Levels Won*, the distributions are different.

- RAND: Coins were placed randomly in any cell that had at least one winning trajectory pass through it. In this condition, each player had the coins placed in different random locations.

Examples of coins placed by each method for one level of the game are shown in Figure 4. Of the 1226 participants who played the game, 209 were randomly assigned to NONE, 391 to PATH, 318 to DSGN and 308 to RAND. For each player, we measured the following variables:

- *Levels Won*: The number of levels won by the player.
- *Finish Rate*: 1 if the player finished the game (completed all 14 levels), else 0. Presented in Table 1 as percentage of players per condition who finished with value 1.
- *Total Time*: The total time the player spent playing the game, in seconds.
- *Per-Level Time*: The mean time the player spent playing each level attempted, in seconds.
- *Total Coins*: The total number of coins collected by the player.
- *Per-Level Coins*: The mean number of coins collected by the player in each level attempted.

For statistical analysis, data were not normally distributed as determined by a Shapiro-Wilk normality test. We thus ran an omnibus Kruskal-Wallis test for each variable, and if significant, ran post-hoc pairwise Wilcoxon rank sum tests with the Holm correction. For the variables looking at coins (*Per-Level Coins* and *Total Coins*), we did not include the condition with no coins (NONE) in the analysis. A summary of the results of our analyses is given in Table 1. A survival plot for *Levels Won* is shown in Figure 5.

Discussion

In terms of coin collection, we found that players collected more coins in PATH than in DSGN and RAND. This makes sense, as to collect all coins in this condition, players simply had to follow the desire path. Since the desire path is meant to be a natural route in the level from the start to the goal, by following this path, players could collect all coins while also progressing naturally toward the primary objective (i.e. the treasure chest) of the levels. In other words, players may have naturally collected more coins on their way to the goal, or the coins along the desire path acted as a guide that players could follow to reach the goal. In the other conditions, players had to go off the desire paths to collect the coins. That is, DSGN and RAND required them to temporarily ignore the primary objective of moving towards the chest, to achieve the secondary objective of collecting all coins.

Additionally, players spent more time playing each level in DSGN, than in NONE and PATH. This is likely because DSGN required players to backtrack or take multiple paths through a level to collect all coins, unlike PATH where the coins were placed along the same desire path towards the goal or NONE where there were no coins to collect. Similarly, we did not find a difference in time spent playing each level between DSGN and RAND, possibly because randomly placed coins might also require players to backtrack.

Our findings in relation to the platformer game studied by Andersen et al. (2011b) are particularly interesting. If we consider DSGN and RAND to have off-path coins and look at levels won and time played, we only found that PATH outperformed RAND in terms of levels won. A possible explanation for this may be that the coin placement in DSGN and RAND was not far enough off-path to have as large an impact on gameplay. Similar to Andersen et al. (2011b), we did not find significant differences between no coins (NONE) and on-path coins (PATH). However, while they saw a general trend upward for time played with on-path coins, we saw a trend downward for time played and levels won. It is possible that our game was too short, or the levels in our game were simply not challenging enough for the players to benefit from having a path of coins as a guide. This is partly supported by the survival plot in Figure 5. Though for the first 12 levels, more players keep playing in NONE than in PATH, for the last 2 levels, PATH's survival rate catches up with that of NONE. It is possible that if there were harder levels beyond level 14, PATH would further overtake NONE in terms of survival rate. If true, this implies that a path of coins helps the player only if the levels are sufficiently hard. Exploring this is fertile ground for future work.

Based on our overall statistical results of player behavior, the coin placement strategies appear to fall into two groupings, for which we did not find any significant differences. NONE and PATH did not show differences from each other, and DSGN and RAND did not show differences from each other. That is, discounting the fact that players could not collect coins when there were none, players in PATH behaved similarly to those in NONE, while players in DSGN behaved similarly to those in RAND. We note, however, that significance might have been found with a larger sample size.

Overall, our findings imply that each of NONE, PATH, DSGN and RAND has its own benefits. Absent of other utilities, secondary objectives like collectibles do not necessarily incentivize players to complete more levels and thus, in such cases, not having any collectibles in the level (i.e. NONE), if possible given a game's design constraints, is a reasonable choice. However, if such items are desired, then placement strategy depends on the designer's goals. If the primary purpose of collectibles is to help players in completing the levels, then PATH is preferable whereas if the goal is to make the player explore as much of the level as possible and spend more time in each level, at the risk of completing fewer levels, then DSGN or RAND are suitable alternatives.

Conclusion

In this paper, we introduced a desire path-inspired algorithm that uses player trajectories to place coins in platformer levels, and studied how different placement methods affected player behavior. We consider many avenues for future work.

In this work, we only looked at one game and game genre. Future work could study other games and genres, including 2D top-down or puzzle and 3D games. We also examined a narrow design space: there were only 10 coins to place, and only one type of coin, which had no impact on gameplay. Designers may want to place more coins, other types of coins (such as "challenge" coins that are intentionally hard to collect), or other collectibles that impact gameplay (such as powerups or extra lives). We also only looked at player behavior and not subjective experience.

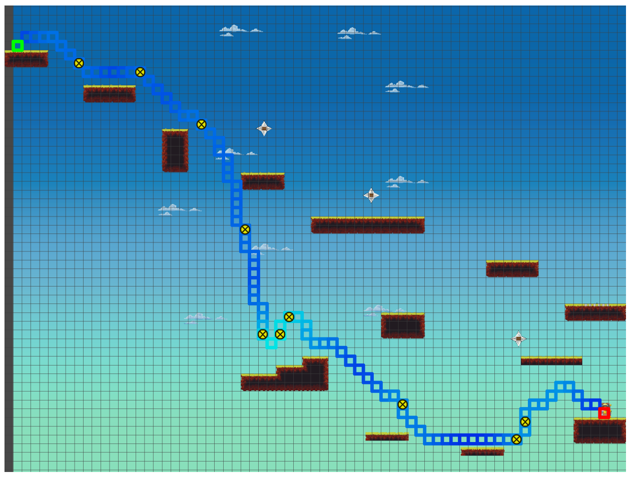


Figure 6: Example of a shortcut found by players used in coin placement. Many players dropped in the gap between the third and fourth platforms to bypass the designer's intended path through the level. Note that the two platforms near the bottom move vertically and horizontally.

The algorithm itself was guided by our own heuristics, but other heuristics may do better, possibly improving the impact of coin placement. Our algorithm requires some players to win the level (although used as few as 8), which may be a problem for especially difficult games, though in such cases, it could be possible to use trajectories of players who make the most progress through the level in place of winning trajectories. The algorithm does not incorporate losing trajectories or direction of player movement between cells, which may be useful (for example, to prevent a path from being found between two neighboring cells that is actually impossible to move between).

We noticed that in some cases our algorithm used extreme "shortcuts" that players had discovered through the level (e.g. the early drop in Figure 6). This is a natural result of using desire paths, but if the designer does not prefer coins in such shortcuts, they could be moved afterwards, using the path placement as an initial suggestion of locations for further manual refinement.

Finally, in this work we used player trajectories to generate a single path for coin placement. Future work can explore sampling multiple paths, based on player trajectories, to generate many possible coin paths through a level.

Acknowledgements

The authors would like to thank Liam Fratturo for the designer coin placement, and the MTurk workers for playing.

References

Andersen, E.; Liu, Y.-E.; Snider, R.; Szeto, R.; and Popović, Z. 2011a. Placing a value on aesthetics in online casual games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, 1275–1278. New York, NY, USA: ACM.

Andersen, E.; Liu, Y.-E.; Snider, R.; Szeto, R.; Cooper, S.; and Popović, Z. 2011b. On the harmfulness of secondary game objectives. In *Proceedings of the 6th International Conference on the Foundations of Digital Games*.

Baldwin, A.; Dahlskog, S.; Font, J. M.; and Holmberg, J. 2017. Towards pattern-based mixed-initiative dungeon generation. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*.

Barros, G. A.; Liapis, A.; and Togelius, J. 2016. Murder mystery generation from open data. In *Proceedings of the International Conference on Computational Creativity*.

Bartle, R. 1996. Hearts, clubs, diamonds, spades: Players who suit MUDs. *The Journal of Virtual Environments* 1(1).

Dormans, J. 2010. Adventures in level design: Generating missions and spaces for action adventure games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*.

Hastings, E.; Guha, R.; and Stanley, K. 2009. Evolving content in the Galactic Arms Race video game. In *2009 IEEE Symposium on Computational Intelligence and Games*.

Jennings-Teats, M.; Smith, G.; and Wardrip-Fruin, N. 2010. Polymorph: A model for dynamic level generation. In *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Kohlstedt, K. 2016. Least resistance: How desire paths can lead to better design. 99% Invisible. https://99percentinvisible.org/article/least-resistance-desire-paths-can-lead-better-design/.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013a. Generating map sketches for strategy games. *Proceedings of Applications of Evolutionary Computation* 7835.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013b. Sentient Sketchbook: Computer-aided game level authoring. In *Proceedings of the 8th International Conference on the Foundations of Digital Games*.

Myhill, C. 2004. Commercial success by looking for desire lines. In Masoodian, M.; Jones, S.; and Rogers, B., eds., *Computer Human Interaction*, 293–304. Berlin, Heidelberg: Springer Berlin Heidelberg.

Nichols, L. 2014. Social desire paths: A new theoretical concept to increase the usability of social science research in society. *Theory and Society* 43(6):647–665.

Pedersen, C.; Togelius, J.; and Yannakakis, G. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1).

Sarkar, A., and Cooper, S. 2018. Comparing paid and volunteer recruitment in human computation games. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*.

Shaker, N.; Shaker, M.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving Cut the Rope levels. In *Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Shaker, N.; Togelius, J.; and Nelson, M. 2016. *Procedural Content Generation in Games*. Springer International Publishing.

Smith, A., and Mateas, M. 2011. Answer set programming for procedural content generation: A design space approach. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):187–200.

Smith, G.; Whitehead, J.; Mateas, M.; Treanor, M.; March, J.; and Cha, M. 2011. Launchpad: A rhythm-based level generator for 2-D platformers. *IEEE Transactions on Computational Intelligence and AI in Games* 3(1).

Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):201–215.

Smith, G. 2014a. The future of procedural content generation in games. In *Proceedings of the Experimental AI in Games Workshop*.

Smith, G. 2014b. Understanding procedural content generation: A design-centric analysis of the role of PCG in games. In *Proceedings of the 32nd annual ACM Conference on Human Factors in Computing Systems*.

Snodgrass, S., and Ontañón, S. 2014. Experiments in map generation using Markov chains. In *Proceedings of the 9th International Conference on the Foundations of Digital Games*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A.; Isaksen, A.; Nealen, A.; and Togelius, J. 2017. Procedural content generation via machine learning (PCGML). In *Proceedings of the 12th International Conference on the Foundations of Digital Games*.

Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.

Yee, N. 2007. Motivations for play in online games. *CyberPsychology and Behavior* 9(6):772–775.

Zhang, Y.; Padman, R.; and Levin, J. E. 2014. Paving the COWpath: Data-driven design of pediatric order sets. *Journal of the American Medical Informatics Association* 21(e2):e304–e311.

Zook, A., and Riedl, M. 2012. A temporal data-driven player model for dynamic difficulty adjustment. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Zook, A.; Lee-Urban, S.; Drinkwater, M.; and Riedl, M. 2012. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proceedings of the 7th International Conference on the Foundations of Digital Games*.