A Framework to Create Collaborative Games for Team Building using Procedural Content Generation

Umberto Picariello
DEIB - Politecnico di Milano
umberto.picariello@mail.polimi.it

Daniele Loiacono
DEIB - Politecnico di Milano
daniele.loiacono@polimi.it
OrcId: 0000-0002-5355-0634

Fabio Mosca
AnotheReality
fabio@anothereality.io

Pier Luca Lanzi
DEIB - Politecnico di Milano
pierluca.lanzi@polimi.it
OrcId: 0000-0002-1933-7717

***Abstract*—We present a framework to design collaborative games for team building that employs a search-based procedural content generation toolset to help designers creating levels. It comprises a multiplayer game with asymmetric interaction in which a player must reach the exit of a maze within a time limit, while wearing a head mounted display. The maze is complex and the player could not reach the exit on time without a team there to help her using a large number of printed maps describing candidate mazes. The player describes what she sees to the team who use such information to identify which one of the several available maps the player is currently navigating in virtual reality. Each game requires the generation of hundreds of very similar maps with several aliasing (confusing) situations but at the same time need to have a solution to guarantee that if the team members collaborate effectively they can guide the player to the exit. It would be infeasible for a human designer to provide such a massive number of maps for every game played. Accordingly, we developed automatic authoring tools to help designers generate such large sets of maps and also to optimize them based on design principles focusing on fun and pace. Our preliminary results show that the authoring pipeline we created can generate games (set of maps) adherent to such design principles.**

***Index Terms*—Search-based procedural content generation, collaborative puzzle games, team building.**

I. Introduction

Video games have seen a dramatic increase in their market share surpassing the film industry in 2018.[1] This widespread success fueled their application outside the realm of entertainment with videogames developed for specific game-based learning tasks such as military training [1], [2], acquisition of technical skills (e.g., effective fire extinguisher usage, forklift operation, commercial driving, and hazardous gas leak identification) [3], engineering education [4], [5], healthcare [6], [7], astronaut training [8], etc.

Commercial cooperative video games, usually with asymmetric multiplayer game modes, have recently become the go-to tools for team-buildings in companies [9], [10], [11], [12]. These commercial solutions have increased companies' interest toward applying video games for fostering creativity and team building but they are intrinsically limited in that they cannot be tuned to specific scenarios nor to a certain difficulty level. Accordingly, a number of startups started focusing on the creation of video games specifically designed for team building that provide highly customizable experiences like for instance [13], [14].

In this paper, we present a framework for team building that we designed and developed in collaboration with Anothereality[2], a startup specialized in immersive interactions for training and team-building. It implements a collaborative multiplayer gaming experience with an asymmetric interaction based on virtual reality. A player, wearing an head-mounted display (HMD), is placed on a start position inside a maze and must reach the exit within a certain time limit with the help of a team who have printed maps describing candidate mazes. The player describes what she currently sees to the team members who must work together to help the player reaching the exit by identifying which maze the player is navigating. A level comprises the environment that the player has to navigate in virtual reality and a large set of candidate printed maps. Levels are generated using methods of search-based procedural content generation. The difficulty level is determined by the amount perceptual aliasing in the maps available to the team members: the more the aliasing, the more difficult will be for the team to identify the map that the player is navigating. The game requires the creation of several (hundreds of) very similar maps that must present several aliasing (confusing) situations to make it difficult for the team to identify the map that the player is actually navigating. At the same time, the large set of map needs to provide the team with a unique solution to guarantee that if they collaborate effectively they can actually lead the player to the exit. This task would be infeasible to complete by a human designer without the help of some automatic authoring tools. Accordingly, our framework provides a toolset to help designer generate such set of maps and also to optimize it based on design principles focusing on fun and pace. We present experimental results showing how the pipeline we created can generate levels (set of maps) adherent to such design principles.

The paper is organized as follows. In Section II, we overview the relevant published literature while in Section III we overview the design of the collaborative game we designed for team building. In Section IV, we illustrate the pipeline we

[1] https://tcrn.ch/2HtFczS

[2] urlhttp://www.anothereality.io

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

created to generate the large set of maps that will be provided to the team. It comprises the generation of the player map (Section IV-A) and the aliased maps (Section IV-B) which includes a game pace analyzer and optimizer. We present the results of a series of experiments we conducted to show how the optimizer can actually produce maps that are coherent to the pace and design constraints that the designer can set. Finally, we draw some conclusions and delineate the future research directions in Section VI.

II. Related Work

In this section, we overview some of the literature that is more relevant for the work presented here. We refer the interested reader to [15] and to [16] for an in-depth discussion of procedural content generation and maze generation algorithms.

A. Cooperative Puzzle Games

Puzzles are usually defined as *problem solving activities* or *games with a dominant strategy* [17] in which the player has to find a solution given a previous knowledge or by properly explore the space of possible solution for the given problem [18]. Examples of mainstream puzzle games include *Tetris*[3], *Puzzle Bobble*[4], *Rubik's Cube*[5], *Jigsaw*[6], *mazes*[7], *crossword*[8]. Allowing two or more players to cooperate towards the same goal, defines the subgenre of cooperative puzzle games. Sedano et al.[19] define cooperative games are activities in which individuals work together toward a common outcome that, in the context of puzzle games, means working collectively to find the dominant strategy of the game. Examples of mainstream cooperative puzzle games are: *Portal 2*[9], *Trine (videogame series)*[10], *Keep Talking and Nobody Explodes*[11]. An additional clarifying note must be done on the term *cooperative* used to define some sorts of puzzles, since the concepts of *cooperative games* and *dominant strategy* in the scientific literature most commonly refer to game theory, and some may confuse them with the definition of puzzles games we are referring to. Zaga et al. [20] distinguish two types of games in game theory: *non-cooperative/competitive* (every player wants to maximize its utility) and *cooperative* (players can form alliances, maximizing their respective utilities, but the game itself does not guarantee whether they will benefit or not from the cooperation). Accordingly to Zaga et al. a third category has been considered lately in game theory, called *collaborative games*: rewards and penalties are shared between the members that try to maximize the team's utility. It's clear that the difference between collaborative and cooperative games exists in game theory, but in the real of entertainment the two terms are used interchangeably [19], and the meaning of "cooperative games" is closer to the one given a while ago for collaborative games.

[3]https://tetris.com/
[4]https://en.wikipedia.org/wiki/Puzzle_Bobble
[5]https://www.rubiks.com
[6]https://en.wikipedia.org/wiki/Jigsaw_puzzle
[7]https://en.wikipedia.org/wiki/Maze
[8]https://en.wikipedia.org/wiki/Crossword
[9]https://www.thinkwithportals.com/
[10]https://www.frozenbyte.com/games/
[11]https://keeptalkinggame.com/

B. Procedural Content Generation for Cooperative Puzzles

Procedural content generation have been widely applied to single player puzzle games [21] but rarely their cooperative counterparts. The most known title is the above-mentioned *Keep Talking and Nobody Explodes* that sees two players in the challenge of defusing a bomb: one player has a manual explaining how to defuse the bomb while the second player must defuse it before the timer reaches zero. Defusing the bomb means solving puzzles and riddles on it generated procedurally by the game. In the scientific research a paper proposed by Arkel et al.[22] developed a cooperative puzzle-platform game using procedurally generative grammars to layout the levels and its puzzles exploiting game design patterns. Apart from these two examples, the usage of PCG in cooperative games genre is rather uncommon, mainly since generating levels for collaboration is more challenging because of the forced mutual benefits of the cooperation that puts an extra constraint on the design space[22]: so when considering a subset of games such as puzzles, the process can be only more difficult and challenging.

C. Team Building

While team training is used in many companies and organizations to improve team competences and processes, other team-affecting methods can be used in order to improve interpersonal relations and effectiveness of a work group [23]. Team building is an intervention focused on improving team functioning, mainly its operations and processes. Specifically, team building seeks to improve team processes, individual and team characteristics, and to change work structure or task[24]. Studies have underlined how team building activities focus on one (or more) of six components to achieve the "promised" improvement. These six components are: *goal setting*, *interpersonal relations*, *role clarification*, *managerial grid*, and *problem solving*. We have stated in the initial part of this section what a puzzle game is, and it is rather evident how the problem solving activity is inherently a part of it. The cooperative feature of games seems to invest more than one team-building component (interpersonal relations, goal clarification), making it clear that puzzle cooperative games are a valid and viable option for team building activities.

III. A Collaborative Video Game for Team Building

We developed a multiplayer video game with an asymmetric interaction pattern inspired to the well known *Keep Talking and Nobody Explodes*. The game has a sci-fi setting. A drone piloted by the player wearing a virtual reality headset must escape a factory before the exit closes (when the timer reaches zero). The player starts inside a grid-like maze in a starting position (facing north). The player has no vision of neighbor positions because of a *fog of war* or sliding doors that cut the vision in corridors taking to adjacent cells (Figure 1).

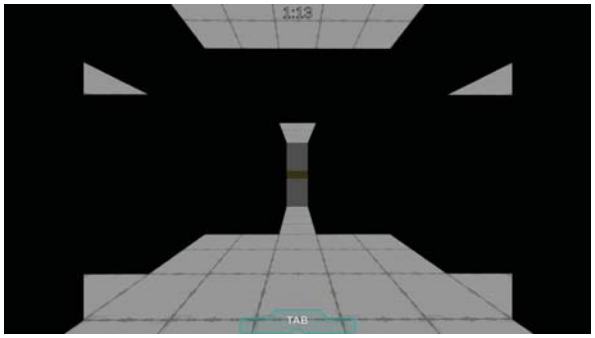


Fig. 1: The player view: sliding doors cut the view of the neighboring rooms/cells.

Furthermore, the maze is quite complex and it would be impossible for the player to reach the exit by herself within the time limit. Accordingly, the player needs a team to help her navigate the maze by interacting with her. The player describes what she sees in virtual reality to the team members who have several *candidate* maps describing very similar mazes. The team has several printed maps available and must use what the player describes to identify which one of the candidate mazes the player is actually navigating in order to be able to lead her to the exit. The solution is unique in that the is only one map that corresponds to what the player sees. All the maps are very similar to each other and contain several aliased positions so that it is very difficult for the team to identify which map the player is navigating. The team members use the information they receive for the players and what they can induce from the available maps to suggest which direction the player should go next according to the information they have. Meanwhile, the player with the virtual reality headset has to constantly describe the new rooms she enters, so that team can rule out some of the maps they have.

The goal of the cooperation is to rule out all the fake maps as soon as possible, in order to identify the actual map the player is navigating and so as to lead the player to the exit.

IV. Level Generation

Our game also includes two components that the designers use to generate levels by first creating the map that the player will navigate and next by generating the aliased maps that will challenge the team helping the player.

A. Player Map Generator

This component provides the designer with two algorithms to procedurally generate the player map based on Randomized Prim's algorithm[12] and Cellular Automata [13]. The tool lets designers select the general parameters (e.g., size, start position, end position), the generation algorithm and its parameters (Figure 2). Next, the map is generated and presented to the designer with some useful statistics to evaluate the complexity of the generated map (Figure 3).

[12]https://en.wikipedia.org/wiki/Prim\%27s_algorithm
[13]https://en.wikipedia.org/wiki/Maze_generation_algorithm

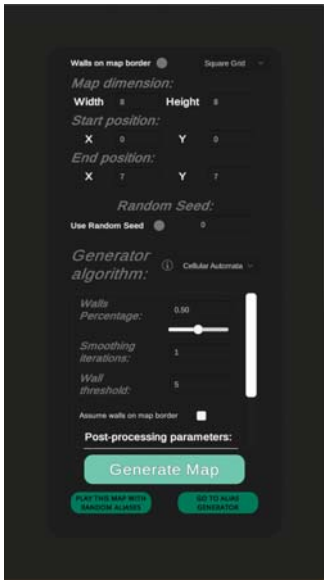


Fig. 2: The parameters panel: on top there are the parameters that are common to both algorithms; on the right a drop-down menu to select the desired generator; at the bottom the algorithm-specific parameters.

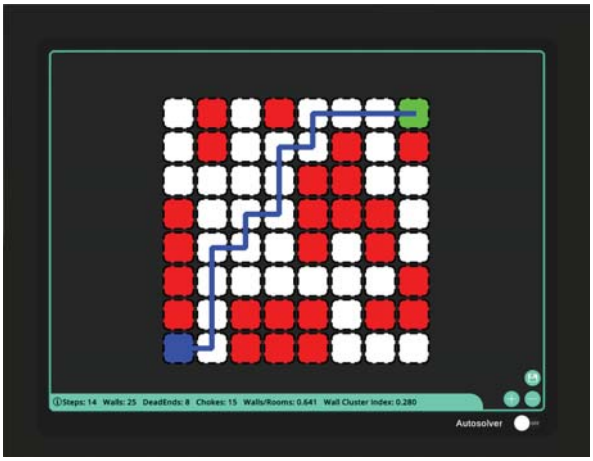


Fig. 3: The map generator: on the bottom-left map statistics; the bottom-right the buttons save, zoom-in and zoom-out the map.

B. Aliased Maps Generator

This component is responsible of generating the maps that will be provided to the team from the player map produced at the previous step. Initially, designers specify the parameters for the generation. Next, the aliased maps are generated through a sequence of three components: the *Alias Generator*, the *Pace Chart*, and the *Optimizer*.

Parameters. Designers can parametrize the Aliased Maps Generator by specifying: (i) the number of alias maps k that will be generated; (ii) the minimum number of steps of the solution; (iii) the amount of aliasing generated that dictates

how much the alias map resembles the real map; finally, (iv) the type of aliasing to generate based on the strategy that the player might follow to reach the exit which could be purely *explorative* or combine *exploration and backtracking*. The former assumes that a player is more interested in exploring new rooms and subsequently avoids elements in the maze that might result in penalties. The latter assumes a more rational strategy and that the player might interact with objects that might result in penalties is these can help to rule out some maps. Thus, it provides the designer with the real minimum and maximum number of steps that the team need to cooperate with the player in order to rule out the aliases. After all the parameters have been specified, the alias generation can start.

Alias Generator. This component applies Alias a stochastic generate-and-test approach to produce a list of aliased maps. More precisely, the generation of one aliased map is done by taking a map with dimensions, start cell and end cell identical to the player map, using the two generation algorithms used for the player map. The process is repeated several times to produce a large number of candidate maps that are then checked for validity. An invalid map is either discarded or fixed using the autosolver (if enabled in the parameter list). The list of aliased maps is sorted according to a similarity metric computed using Mahalanobis distance computed as,

$$D^2_{\boldsymbol{G}}(\boldsymbol{I_x}, \boldsymbol{I_y}) = \sum_{i,j=1}^{d} g_{ij}(x_i - y_i)(x_j - y_j) = (\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{G}(\boldsymbol{x}-\boldsymbol{y}) \tag{1}$$

Where $\boldsymbol{I_x}$ and $\boldsymbol{I_y}$ are two given samples/instances (in our case the mazes) for which we want to compute the distance D. Given a generic coordinate k in the maze, x_k and y_k are the values (in our case 1 if there is a wall, 0 otherwise) of a cell of the maze at position k. Our function choice for G is a Gaussian function so that the conditions for a valid pseudo-metric are directly entailed [25]. From the Gaussian function we want that the value halves from the maximum at $|\boldsymbol{C}_i - \boldsymbol{C}_j| = 1$ (orthogonal cell) and that at $|\boldsymbol{C}_i - \boldsymbol{C}_j| = \sqrt{2}$ (direct diagonal neighbour cell) the value of the Gaussian is less than or equal to $\frac{1}{4}$ the maximum value of the Gaussian[14]. A Gaussian function that satisfies all our design constraints is $\mathcal{N}(0, \frac{3}{4})$:

$$g_{ij} = f(|\boldsymbol{P}_i - \boldsymbol{P}_j|) = \frac{1}{\sqrt{2\pi\frac{3}{4}}} e^{-\frac{|\boldsymbol{P}_i - \boldsymbol{P}_j|^2}{2\frac{3}{4}}} \tag{2}$$

where $\boldsymbol{G} = (g_{ij})_{dxd}$; and finally our distance metric squared:

$$D^2_{\mathcal{N}(0,\frac{3}{4})}(\boldsymbol{I_x}, \boldsymbol{I_y}) = \frac{1}{\sqrt{2\pi\frac{3}{4}}} \sum_{i,j=1}^{MN} e^{-\frac{|\boldsymbol{P}_i - \boldsymbol{P}_j|^2}{2\frac{3}{4}}} (x_i - y_i)(x_j - y_j) \tag{3}$$

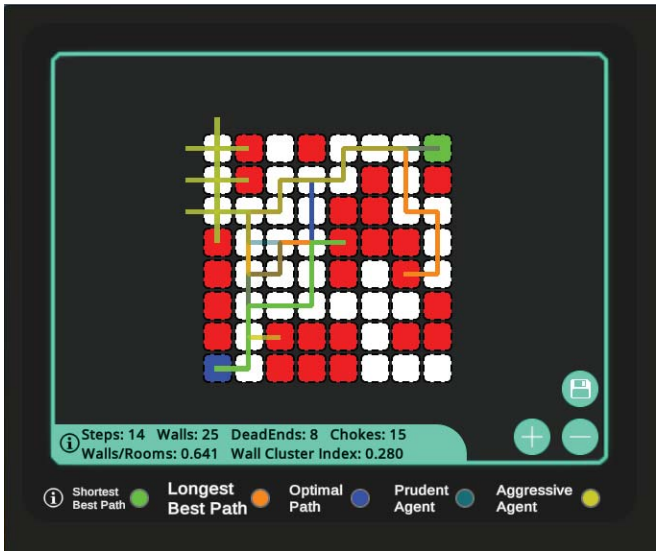
Since g_{ij} is a Gaussian function, all the conditions for the distance metric are satisfied (see the work done by Wang et al. [25], [26] for more details). Given our formal definition, some implementation changes where taken to improve computation and give a better understanding of the distance $D_{\mathcal{N}(0,\frac{3}{4})}$ metric to the designers. We avoid the computation of matrix G, especially when the map dimension changes. This is implicitly done by directly assigning the right g_{ij} value in the dot product of the distance in equation 1. To have a better understanding of the "distance" between two maps, we increase the scale of g_{ij} by a factor that makes its maximum value equal to one. Values of $g_{ij} \leq 0.1$ (i.e. according to Equation 2, all the cells are outside Moore neighborhood of range one) are considered irrelevant for single cell similarity and so counted as zero. The most similar or dissimilar aliased maps are finally presented to the designer.

Pace Chart. This component was inspired by the work of Sorenson et al.[27], [28], [29] on automatic generation of "fun" levels. The component presents the designers five different and interesting paths on the real map with their respective pace:

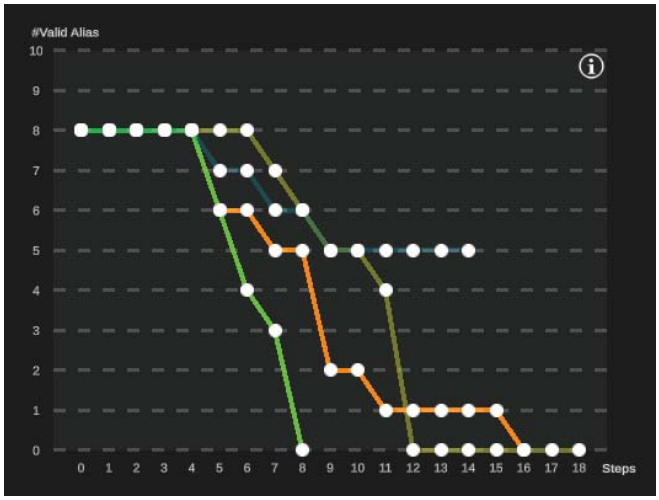
1) *Shortest Best Path*, the *path* that would rule out all the aliased maps with the lowest number of steps;
2) *Longest Best Path*, the *path* that would rule out all the aliased maps with the the highest number of steps.
3) *Optimal Path*, the path computed using the well-known A* algorithm.
4) *Prudent Agent Path*, the path computed by simulating a maze-traversing agent that tries to reach the end considering the most probable next cell (based on the aliasing);
5) *Aggressive Agent Path*, the path computed by simulating a maze-traversing agent that tries to reach the end considering the less probable cell as the next cell (based on the aliasing).

For each path, the number of valid alias at each step is computed and shown to the designer (Figure 4a) with an analysis of the pace for each path (Figure 4b). The pace chart shows the decreasing number of the challenging position in one path on one or more steps is a valid tool for a *what-if* analysis and for further optimizations of designers' interest. The pace chart shows curves that reflect the challenge level for each path and so to the level of fun that the game intended to reach.

[14]The quantity $|\boldsymbol{C}_i - \boldsymbol{C}_j|$ computes the euclidean distance between cell coordinates at i and j positions.



(a)



(b)

Fig. 4: Alias generator: (a) the five interesting paths reported on the player map; (b) the pace chart reporting the number of aliasing positions from the set of the generated map.

Optimizer. Finally, designers can run hill-climbing to modify the list of aliased maps so as to optimize the curves reported in the pace chart using of the following metrics,

- *First Zero Point (Min)* minimizes the first point that goes to zero on the Pace Chart considering all the displayed lines.
- *Best Path Pitchfork (Max)* maximizes the first and the last point that go to zero in the Pace Chart considering only the Best Path curves.
- *Agents Pitchfork (Max):* maximizes the first and the last point that go to zero in the Pace Chart considering only the agents paths curves.
- *Overall Pitchfork (Max):* maximizes the first and the last point that go to zero in the Pace Chart considering all the displayed lines.
- *Reliability Best Path (Min):* minimizes how much is likely that the path of the player will incur in one of the Shortest Best Paths that could be displayed on the map (simply by removing and re-inserting the same map in the alias set).
- *Reliability Best Path (Max):* maximizes how much is likely that the path of the player will incur in one of the Shortest Best Paths that could be displayed on the map (simply by removing and re-inserting the same map in the alias set).

V. Experimental Results

We performed a set of experiments to evaluate the optimization process that is responsible to modify set of maps to be adherent to one of the metrics that designers can select (Section IV-B) that is, First Zero Point (Min), Best Path Pitchfork (Max), Agents Pitchfork (Max), Overall Pitchfork (Max), Reliability Best Path (Min), and Reliability Best Path (Max). Our goal is to determine whether, given one of two map generation algorithm, the level found through optimization using a random-restart hill-climber (RR) is better than the one found doing basic random search (RS) by simply randomizing all the parameters in the generator using the same number of trials/iterations. In the case of RR, the number of trials are the number of perturbation and evaluation performed; in the case of RS, the number of trials simply corresponds to the number of alias set randomly generated. For each metric, we performed 10 runs with an 8x8 target map in which the start cell is located at the bottom left corner, position $(0, 0)$, and the exit cell is located at the upper right corner, position $(7, 7)$.

First Zero Value. This metric minimizes the first zero value, that will be the one of the Shortest Best Path (green line in the Pace Chart, Figure 4b) in practically all the cases. So by optimizing this metrics, we aim at having at least one of the Shortest Best Paths. Figure 5 shows that the optimization process converges almost immediately to the optimum. This is not surprising since this metrics is very easy to optimize and in fact another set of experiments showed that even one iteration of random search is sufficient to find the optimal value for this metric.

Best Path Pitchfork. This metrics maximizes the difference between the zero of the Shortest Best Path and the zero of the Longest Best Path found. Thus, this optimization actually maximizes the first zero value of the Shortest Best Path (green line in the Pace Chart, Figure 4b) giving to the designer the possibility to maximize it and see how much further she can increase the minimum steps required to solve the alias challenge. Figure 6 shows that optimization reaches the value obtained using basic random search.

Agents Pitchfork. This metrics aims at maximizing the difference between the first zero value of the aggressive agent and the prudent one (evaluating the respective curves on the Pace Chart). Figure 7 shows that random search provides better

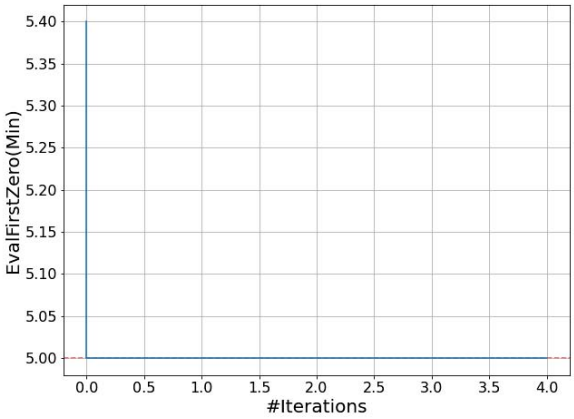


Fig. 5: The mean optimal values (averaged over 10 experiments) found by random-restart hill-climbing algorithm in 5 5 iterations. The red-dotted line shows the best value found by random search.

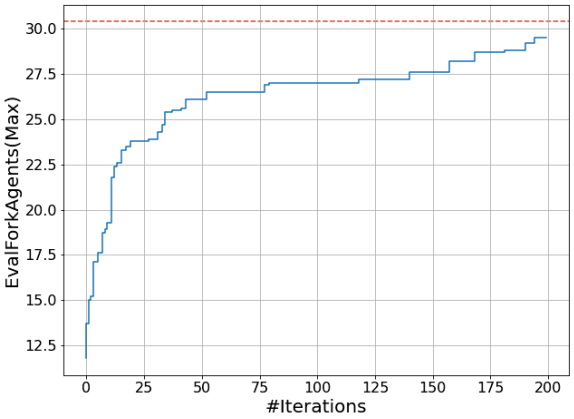


Fig. 7: The mean optimal values (averaged on 10 experiments) found by random-restart hill-climbing algorithm in $K = 200$ iterations. The red-dotted line shows the best value found by random search.

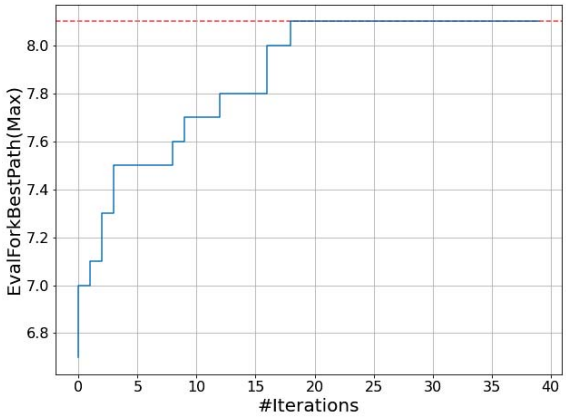


Fig. 6: The mean optimal values (averaged on 10 experiments) found by random-restart hill-climbing algorithm in $K = 40$ iterations. The red-dotted line shows the best value found by random search.

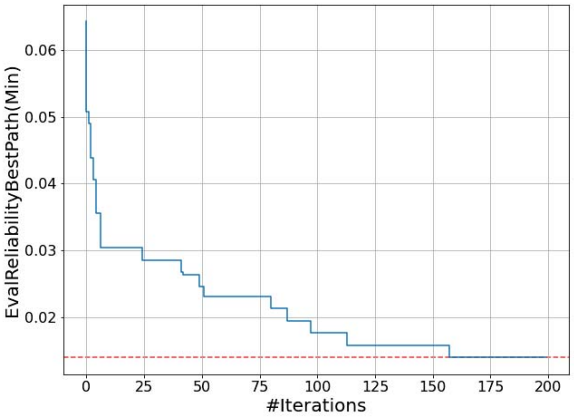


Fig. 8: The mean optimal values (averaged on 10 experiments) found by random-restart hill-climbing algorithm in 200 iterations. The red-dotted line shows the best value found by random search.

results that hill climbing suggesting that we need a better search algorithms to get better results that plain random search.

Reliability Best Path (Min). This metrics computes how many paths of length equal to the Shortest Best Paths are actually best paths and must be minimized. Also in this case, plain random search can generate better solutions than the one obtained through hill climbing (Figure 8). However, the optimization requires half trials than random search to obtain the same performance. Accordingly, the results suggest that minimizing the reliability best path function is a valid option.

Reliability Best Path (Max). This metrics computes how many paths of length equal to the Shortest Best Paths are actually best paths and must be maximized. Figure 9 shows that random restart hill climbing can find better solutions than plain random search. This is probably due to the fact that the metrics favors the incremental construction of solution step by step.

Overall, we noted that for most of the metrics plain random search finds better solutions than hill climbing suggesting that for very simple metrics (e.g., First Zero Value) plain random search could be more than enough, while for other metrics we should investigate more powerful search methods and also consider multi-objective optimization to evolve maps

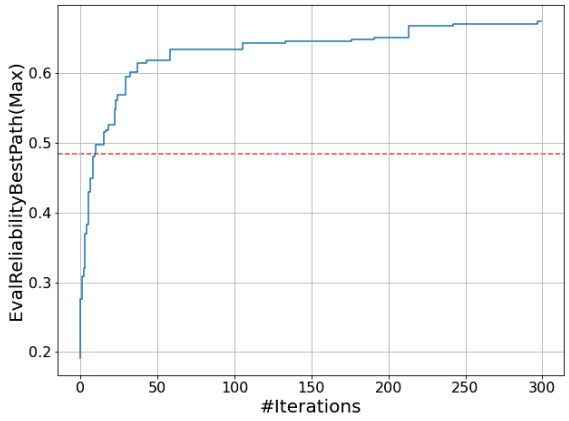


Fig. 9: The mean optimal values (averaged on 10 experiments) found by random-restart hill-climbing algorithm in 300 iterations. The red-dotted line shows the best value found by random search.

that optimize more criteria at once.

VI. Conclusions and Future Developments

We presented our framework for designing collaborative gaming experiences for team building. The framework consists of a multiplayer game with asymmetric interaction and a toolset to help designers create levels using a search-based procedural content generation pipeline. In the game, a player wearing a head mounted display is inside a maze and has to reach the exit within a rather short time limit. The maze is complex enough to make it almost impossible for the player to find the exit alone. Accordingly, a team with a large set of candidate maps work together to understand which one of the several available maps is the one that the player is currently navigating. Once they identified it, they can easily guide the player to the exit. Each game played require the creation of several (hundreds of) very similar maps that must present several aliasing (confusing) situations to make it difficult for the team to identify the map that the player is actually navigating. At the same time, a level should have a unique solution to guarantee that if the team collaborates effectively it can actually lead the player to the exit. Such large number of maps would be infeasible to generate without some automatic authoring tools. Accordingly, our framework includes a toolset to help designers generate such set of maps and also to optimize it based on design principles focusing on fun and pace. We presented experimental results showing that some metrics can be easily optimized using plain random search over the space of feasible solutions while hill climbing is needed for a more complex metrics like Reliability Best Path (Max). Overall, the results suggests that we should provide designers with more alternative optimization strategies depending on the target metrics (simple random search for basic metrics and more complex ones for advanced metrics).

Future research directions include additional generation algorithms for the player map (which is currently done using randomized Prim's algorithm and cellular automata) and the use of more powerful search algorithms for pace optimization considering also multi-objective search algorithms to produce levels that would optimize more criteria at once. We also plan to perform an extended experimentation with human subjects that was planned but we did not performed due to the social distancing restrictions that did not allow us to perform studies with human subjects.

References

[1] C. Frederick-Recascino, D. Liu, S. Doherty, J. Kring, and D. Liskey, "Articulating an experimental model for the study of game-based learning," in *Human Interface and the Management of Information. Information and Interaction for Learning, Culture, Collaboration and Business,*, S. Yamamoto, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 25–32.

[2] N. B. Huntemann and M. T. Payne, *Joystick Soldiers: The Politics of Play in Military Video Games*. Routledge - Taylor & Francis Group, 2010.

[3] S. Bloom, "Game-based learning," *Professional Safety*, vol. 54, p. 18, 2009.

[4] C. Colombo, N. D. Blas, I. Gkolias, P. L. Lanzi, D. Loiacono, and E. Stella, "An educational experience to raise awareness about space debris," *IEEE Access*, vol. 8, pp. 85 162–85 178, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2992327

[5] A. P. Markopoulos, A. Fragkou, P. D. Kasidiaris, and J. P. Davim, "Gamification in engineering education and professional training," *International Journal of Mechanical Engineering Education*, vol. 43, no. 2, pp. 118–131, 2015. [Online]. Available: https://doi.org/10.1177/0306419015591324

[6] L. E. Nacke and S. Deterding, "The maturing of gamification research," *Comput. Hum. Behav.*, vol. 71, pp. 450–454, 2017. [Online]. Available: https://doi.org/10.1016/j.chb.2016.11.062

[7] M. Pirovano, R. Mainetti, G. Baud-Bovy, P. L. Lanzi, and N. A. Borghese, "Intelligent game engine for rehabilitation (IGER)," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 1, pp. 43–55, 2016. [Online]. Available: https://doi.org/10.1109/TCIAIG.2014.2368392

[8] F. Cornelissen, *Gamification for Astronaut Training*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2012-1275405

[9] M. J. Keith, G. Anderson, J. E. Gaskin, and D. L. Dean, "Team video gaming for team building: Effects on team performance," *AIS Trans. Hum. Comput. Interact.*, vol. 10, no. 4, p. 2, 2018. [Online]. Available: https://aisel.aisnet.org/thci/vol10/iss4/2

[10] Last visited on August 2020. [Online]. Available: https://www.businessnewsdaily.com/15032-video-games-for-team-building.html

[11] Last visited on August 2020. [Online]. Available: https://watsonadventures.com/blog/team-building/5-great-video-games-corporate-team-building/

[12] Last visited on August 2020. [Online]. Available: https://fireflyteamevents.com/team-building-office-video-games/

[13] "Watson adventures," last visited on August 2020. [Online]. Available: https://watsonadventures.com/

[14] "Anothereality," last visited on August 2020. [Online]. Available: https://www.anothereality.io/

[15] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Spriger-Verlag, 2016.

[16] J. Buck, *Mazes for Programmers: Code Your Own Twisty Little Passages*. Pragmatic Bookshelf, 2015.

[17] J. Schell, *The Art of Game Design A Book od Lenses*, 3rd ed. CRC Press Taylor & Francis Group, 2019.

[18] M. Hendrikx, S. Meijer, J. Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 9, 02 2013.

[19] C. Islas Sedano, M. Carvalho, N. Secco, and C. Longstreet, "Collaborative and cooperative games: Facts and assumptions," 05 2013, pp. 370–376.

[20] J. Zagal and J. Rick, "Collaborative games: Lessons learned from board games," *Simulation & Gaming - Simulat Gaming*, vol. 37, pp. 24–40, 03 2006.
[21] B. De Kegel and M. Haahr, "Procedural puzzle generation: A survey," *IEEE Transactions on Games*, vol. 12, no. 1, pp. 21–40, 2020.
[22] B. van Arkel, D. Karavolos, A. J. Bouwer, S. Bakkes, and F. Nack, "Procedural generation of collaborative puzzle-platform game levels," 2015.
[23] N. Stanton, A. Hedge, K. Brookhuis, E. Salas, and H. Hendrick, *Handbook of Human Factors and Ergonomics Methods*, 2004.
[24] S. I. Tannenbaum, R. L. Beard, and E. Salas, "Chapter 5 team building and its influence on team effectiveness: an examination of conceptual and empirical developments," in *Issues, Theory, and Research in Industrial/Organizational Psychology*, ser. Advances in Psychology, K. Kelley, Ed. North-Holland, 1992, vol. 82, pp. 117 – 153. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166411508626011
[25] Liwei Wang, Yan Zhang, and Jufu Feng, "On the euclidean distance of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1334–1339, 2005.
[26] B. Sun, J. Feng, and L. Wang, "Learning imed via shift-invariant transformation," 06 2009, pp. 1398–1405.
[27] N. Sorenson and P. Pasquier, "The evolution of fun: Automatic level design through challenge modeling," 01 2010.
[28] ——, "Towards a generic framework for automated video game level creation," vol. 6024, 04 2010, pp. 131–140.
[29] N. Sorenson, P. Pasquier, and S. Dipaola, "A generic approach to challenge modeling for the procedural creation of video game levels," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 3, pp. 229–244, 09 2011.