Enhancing player experience using procedural level generation

Bogdan Maxim
Technical University of Cluj-Napoca
Str. Memorandumului, Nr. 28
bogdan.maxim@cs.utcluj.ro

Daniel Ciugurean
Technical University of Cluj-Napoca
Str. Memorandumului, Nr. 28
danielciugurean@outlook.com

Dorian Gorgan
Computer Science Departament, Technical University of Cluj-Napoca
Str. Memorandumului, Nr. 28
dorian.gorgan@cs.utcluj.ro

ABSTRACT
In this paper we explore different types of procedurally generated content to see if and how it can affect the gameplay experience of the player. We conduct a survey with different types of procedurally generated content in order to see if and how it affects the player when playing a game. The paper presents a simple turn based tactical strategy game where the objective of the player is to eliminate the enemy team. The player must achieve this goal in different circumstances: from a simple grid to a procedurally random generated levels. The experiments conducted will show the way procedurally generation will enhance the gameplay experience of the player and where it should be used.

Author Keywords
Video games; procedurally generated content; player experience; gameplay; replayability

INTRODUCTION
Procedural level generation is a great way to add new content for a game on the fly. The content that can be created varies from maps, characters, levels, weapons, powerups, loot boxes etc. The possibilities for procedurally generated content are endless. They can generate entire levels for games, they can generate the position of the enemies, they can generate lootboxes and they can even generate characters in order to immerse the player into the world.

Player experience is given by different factors from the game: graphics, sound, gameplay, artificial intelligence, immersion and content. All of these elements affects the way the player perceives the game, the time he enjoys spending with it and how addictive the game is. Using procedural generated content, the designers of the game are able to generate a lot of content on the fly without having to hand craft everything. While some games which have employed procedural content generation are successful, some of them are not.

The objective of this paper is to see how procedural generated levels affects player experience, when it shall be employed and when it is enough. This paper is divided into six sections: Introduction, Related Works, Concepts, Methods, Tests and Results and Conclusion. The related works section presents games who had an impact over the industry and employ procedural content generation from 1980 to present day. The Concepts section presents some concepts that need to be understood in order to understand the next sections. The Methods section presents the game implemented and the iteration made over it in order to be able to conduct the experiments. Also, this section discusses the technology and concepts used to implement it and the way they were constructed. Tests and Results section presents the extensive way we conducted the experiments over the subjects in order to find out more about the gameplay and procedural level generation. It shows an elaborate way to conduct a research over a group of people in order to get some conclusions. The conclusion section reviews the results and presents further work.

RELATED WORKS
In the right formula, procedural content generation is magical. It elevates the design and highlights the elegance of the core system loops. The computer games industry is well known for enhancing games with procedural level generation. These games have been available even before the Video Game Crash of 1983. The first game which used procedural generation is Beneath Apple Manor. This game is one of the earliest examples of roguelike games, where the player must navigate a procedural generated dungeon. But the major breakthrough of rogue like games was achieved by a game called Rogue in 1980. This was soon followed by Elite which introduced 3D gameplay mechanics and procedural generated galaxies. Until 1991, no major titles made breakthrough in this field. In 1991, a game called Civilization showed the world that procedural level generation can be used in order to build maps for turn based strategies, achieving the possibility to change tactics for each game you've played. The Elder Scrolls: Arena released in 1994 used procedural content generation for the map, thus introducing content on the run for the first person role playing games. Diablo introduced the procedural content generation for dungeons in action role playing games, adding new content for the role playing gamers. This meant that no two playthroughs would be the same. The next iterations of Civilization and Diablo would improve and perfect their traditional formulas in order to

enhance the replayability and the variety of the gameplay. By 2008, roguelike games were almost non-existent, until a game called Spelunky revived the genre. Each dungeon in Spelunky is procedurally generated given different game experience each time one level is being played. Also, in 2008 two major breakthroughs used procedurally generated content to enhance the content: Spore and Left 4 Dead. Spore used procedural generated content for the characters so that they occupy as few bytes as possible. Each character has a seed which acts like a DNA and from there the character is generated in combination with pre-made content by the developers. In Left 4 Dead the game's artificial intelligence would observe the players' effects on the world and based on that, would adjust the difficulty accordingly. This is a very intelligent example of using procedural content generation in combination with artificial intelligence in order to enhance the player's experience. Left 4 Dead II refined this idea and improved the AI. In 2011 a game called Minecraft took the world by surprise with its gameplay elements and procedural generated content. The player can build anything in a voxel-based space which is procedural generated and also could bring friends in order to join the fun. No Man's Sky shocked the world in 2016 by generating over 18 quintillion planets to explore in the game. Very little game data would be stored as everything is generated on the fly from a specific seed. XCOM II(2016) generates level portions on the run in order to add new content each time it is played. Being a turn based tile based strategy game, different tactics would be employed each time a battle is fought.

CONCEPTS

Games are famously hard to define[4]. By games we refer to the videogames, computer games, board games, puzzles, card games etc. Games have the role to leisure the user and to entertain him. Entertaining is the main objective of the game. The more the user is entertained and wants to spend his time within the game's world, the better the game is. There are different types of video games. The most popular are: first person shooters, third person shooters, real-time strategies, turn-based strategies, role playing games, adventure games, puzzle games, action games and the list goes on. From these genres, we are going to focus on the turn-based strategies.

A **turn-based strategy game** is a genre in which each player or team has a limited number of units on the board. Each unit have a limited number of action points that can take each turn. This action points consist in movement, attack or calling special abilities in order to achieve victory. After all the action points are gone for the turn or the player decides that he won't do anything for that turn, he will end his turn, letting the enemy make his moves. Turn-based strategies are characterized by the fact that the action takes place in sequential turns. Usually, these kinds of games take place on a board represented by a grid.

A **tactical turn-based** strategy is a computer and video game genre that through stop-action simulates the consideration and circumstances of operational warfare and military tactics in generally small-scale confrontations as opposed to more strategic considerations of turn-based strategy games.

The **grid** represents the gameboard with a specific number of squares arranged in a specific pattern. The grid may have a rectangular shape or a square shape. By controlling the dimension of a tile and the number of tiles, different maps can be achieved.

Procedural content generation is the algorithmic creation of game content with limited or indirect user input [1]. In other words, procedural content generation refers to computer software that can create game content on its own or togheter with one or many human players or designers. A key term here is content. In the context of procedurally generators, content may refer to: characters, maps, levels, weapons, loot boxes etc. Procedurally generated characters add detail and depth to characters and multiple characters can be created in a very short period of time. These can vary from family dynasties to individual characters with whom the player can build different relationships.
Procedural generated levels means that each level of the game will be generated on the fly based on a set of rules and different assets. In this way, combining the same mechanics and the same assets in different modes, players will achieve a different experience each time they play a level. In the same way, huge universes can be generated in a short amount of time giving players a huge world in which they can play. Procedurally generating maps are used in turn based strategy games. Each battle will be held on different layouts with different units generated by the computer so that no two matches will be the same. This adds to the replayability and life duration of the game. [2,7]

Playability is the ease by which the game can be played or the quantity or duration that a game can be played and is a common measure of quality of gameplay[5]. By using procedural level generation in the right situation, the playability of a computer game may be boosted which will give the player extra playtime experience and quality time.

Replay value represents a video game's potential for continued play after its first completion. Different techniques can be used in order to maximize the replay value: different endings, different party combination or different levels which can be achivied by building them at runtime [3].

Extending content represents the action through which a game's lifespan may be extended by adding different levels, maps, mechanics or characters into the game. By using procedural content generation, a game content can be easily extended by generating that content instead of letting a team of game developers to hand craft that content.

Fun represents the enjoyment of pleasure, particularly in leisure activities. This is the main purpose of video games in general. By having a high fun factor, the game induces the addiction factor which leads to the case in which the player thrives for more time spent playing the game. Procedural content generation helps adding a fun factor in form of helping the designers add more content in a convenable time period. The objective of this study is to find out the right amount of generated content in order to enhance the fun factor.

METHODS

This section presents the tools, environments and an incremental approach constructed in order to see how procedural generated levels affect the content and player experience. For the testing, we have implemented a simple tactical turn-based strategy game based on XCOM: Enemy Unknown. The game consists of 9 different type of units, each having unique properties, assets and moves. They fight in a dungeon which, at first it is handcrafted(two rooms) and then they become procedurally generated by a given algorithm. The main objective is to destroy the enemy team by using each unit's attack. Each unit has a different number of action points. Playing with different kind of settings, we observe players interest time, the time until they get bored and leave the game.

For constructing the simple game, we chose Unreal Engine for multiple reasons: first, Unreal Engine is free, which means that we could use all its features. This leads to the fact that we could focus on implementing the game and measuring the results and having a high quality game. We wanted to measure the way procedural content generation affects users' experience and we needed a way in order to focus on that. This meant that things like control or graphics should be by default good so that the users' experience isn't affected by these. Second, XCOM: Enemy Unknown was built in Unreal Engine so we wanted to keep the same engine for that kind of game.

By having these tools at our disposal, we incorporated an incremental approach: we started our tests with the same level and the same unit for both the player and the enemy. Then we added procedural map generation in order to change the layout and do the tests again. After this, we added different characters for both the player and the enemies. These are the same characters each time. Only the layout is changed. After the tests were run on this setting, we started to add random characters at different positions on the grid and observe the result. Finally, we experimented a little bit with the grid's dimension and the number of rooms.

Each of the next subsections will describe each iteration of the algorithm and give details about specific aspects of the changed part.

Same room and characters

We first started by constructing the grid. The grid is formed by tiles which are squares specified by a certain dimension. We chose 100 units in Unreal. To each square we attached a static mesh. A static mesh is a piece of geometry that consists of a set of polygons that can be cached in video memory and rendered by the video card. By using the caching principle, this means that a static mesh is rendered efficiently, allowing to efficiently render bigger grids. Static meshes can be translated, rotated, scaled but can not be animated. By encoding two parameters, one for the tile dimension and one for the grid dimension, we will be able to control the dimension of the grid by adjusting these two parameters. This will be useful in a future experiment. Once we generated the grid, we added one giant room and we placed 5 soldiers for the player's team and 5 soldier for the enemy team. We used a basic mesh for the unit with idle stance, attack stance and death stance so that the player can visually see what is going on. Each unit has health represented by a points(health points). The default unit has 500 health points. When a unit attacks, it deals a specific type of damage, represented by damage points. The default unit has a 100 damage points. To compute the health of the attacked unit, we just subtract from the total health points of that unit the damage points dealt by the other unit. The units can not regenerate health. This allows for a very basic strategy turn based game in order to see how fast the fun factor runs out. Each unit has an exact number of 7 action points. After 7 actions have taken place, the unit can not do anything for this turn.

Generated rooms and same characters

While maintaining the same characters as before, we added procedural map generation for the rooms. In this way, each map will be different so each player will have a different experience each time. No two games will be the same. For this, we employed a space partitioning algorithm. We used a quadtree [6] in order to partition the grid. In the quadtree, each quadrant will contain a single room(placed stochastically) as well as an empty space. Corridors will be added after the entire generation process is completed. We hold an array for the cells in memory. Each tile has, besides the size and the world position, an integer value which specifies where does that tile belong to. A value of 1 indicates that the tile belongs to a room, a value of 2 indicates that the tile belongs to a corridor and a tile of 0 indicates that the tile belongs to an inaccessible space.

After generating the grid(with the mention that each tile starts with value 0), we create the quadtree and assign to each tile in the grid a place in the quadtree. After this, we randomly assign some tiles value for a room and start to build the room around it. The value of the size of the room is chosen randomly. Then we iterate over and stochastically drop some rooms in order to keep them in a playable fashion.

For corridors, we find the nearest room from the room that we are in, find the closest edges and find the line from A to

B. Then we walk that line and add the specific numbers to the tile.

After each tile has a number assigned to it, we have a component which takes the grid as the input and builds the mesh for that grid. It will start by building the room and then the walls and corridors and then the inaccessible part. Figure 1 shows the generated rooms from above with the status bar of all the units.

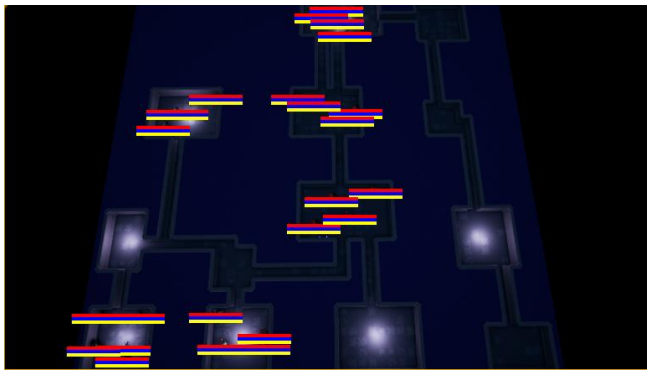

Figure 1. Procedurally generated rooms in the game

Generated rooms and multiple characters at the same locations

After having different dungeons for each level, the next iteration of our method added characters instead of the same default unit. For each game, different characters will be used, but in this iteration, they will be the same from one game to another. We added the following classes: Assault(500 health points, 150 damage points), Heavy(1000 health points, 100 damage points), Medic(500 health points, 50 damage points), Specialist(500 health points, 100 damage points).

Generated rooms and multiple characters at random locations

The next iteration consists in choosing a random place for start. All the units of the player will be situated in a random room which was selected at the beginning and all the units which belong to the enemy will be situated in another room, different from the room of the player. For this, when we generate the room, we keep their middle position in a separate array and we mark if we have something in that room. For random placement, we choose a random index in the range of that array and make the necessary computations. In the room, we spawn random positions for the characters. We also employ an array for all the characters classes we have and we random generate a number in the range of that array in order to spawn a random character.

Generated rooms, multiple characters at random locations and different grid dimensions

The last iteration for our tests consist in having everything we had up until this point, but playing with the map dimensions. Randomly choosing the size of the map by adjusting the dimension of the tiles and different tile numbers for the length and width, we procedurally generate levels in this way in order to see its impact over the players.

TESTS AND RESULTS

This section presents the tests and results which we employed over the iterations presented above.

By having this incremental approach at our disposal, we wanted to measure the flow[8] of the player. By this we asked students from our University in order to playtests all the iterations presented above and after that we employed two playtests methods in order to see what iteration made an impact over them. The two playtest methods are traditional methods and technical approaches. Traditional methods consist of direct observations and verbal reports, while technical approaches consist of design experiments and surveys[9].

We are going through all the iterations and present the results. For each iteration we applied the direct observation and then we conducted the technical approaches in order to emphasize the results.

Same room and characters

We start by presenting the players the basic level we constructed. Naturally, we expected the fact that the fun factor should go away extremely fast. This was confirmed by direct observation where we watched the people play the game and observing their behavior. After a few minutes they would stand up and leave the game as it is. Having the same characters and same level repeated drains the fun factor pretty fast. The verbal reports confirmed. Player complained about the repetitive nature of the level and the fact that the enemy is making the same moves over and over again.

Generated rooms and same characters

When presented with the procedural generated rooms, players had a positive reaction and the direct observations confirmed this. They were motivated to clear every room and attack enemy soldiers in order to achieve victory. This was confirmed by the time they spent with the game. They didn't want to leave the game as soon as the possibility above. By having a different layout every time, they needed to rethink their strategy in order to achieve victory which opened various possibilities in gameplay terms. Their verbal communication gave us positive results, with the only complain that fully different tactical possibilities will be achieved when different characters will be presented on the screen. The verbal reports were positive

Generated rooms and multiple characters at the same locations

From a tactical point of view, multiple characters add to the possibilities that can be employed, together with random generated rooms. The main complaint we had here is the fact that players needed a way to change the characters in order to open up the variety of tactical possibilities that these kind of games offer. The verbal reports continued to be positive

Generated rooms and multiple characters at random locations

Even though the players loved this combination, they stated that the impact given by random characters at random locations was not as big as the procedurally generated rooms, in terms of their perception over the game world. They knew what to expect from the game and they stated that this part didn't add too much to the fun factor and flow.

Generated rooms, multiple characters at random locations and different grid dimensions

By employing different grid dimension, the length of the games varied but still, players knew what to expect. This meant that procedural generation content did not had too much impact over the players and they stated this verbally. This led us to the next phase where we compared each of the iterations and drawing a line after where the procedural generated content will not affect too much the experience of the player.

Survey

After each iteration, the player was asked five questions for the survey. The questions are:

- How interesting did you find this level?(1 not interesting at all to 5 extremely interesting)
- Rank the following features: procedural generated rooms, multiple characters, different map size, random locations(1 most liked, 4 least liked)
- How much did you changed tactics according to the level?(1 not at all, 5 changed my way of playing)
- How distorted was your experience of time?(1 not at all, 5 fully lost the notion of time)
- Rank the levels from a replayability point of view(1 has the most replayability value, 5 has the least replayability value)

The questions refer to the iterations presented above, unless notified.

The most interesting level was the one with random generated positions, multiple characters and procedurally generated levels. This had all the variety that a gamer would expect.

The procedurally generated rooms were the favorite feature of the gamer and the reason for that is that by having a different map every time, each player must think the strategy different in order to win. This lead to an intense thought process which kept the players entertained.

Surprisingly, the biggest change in tactics came from the procedural generated rooms, not from the characters which lead us to believe that this is the core procedural generated content element that makes the difference.

The gamers related that they mostly lost the notion of time on every level, except the first one which was extremely boring. We specifically design that level in order to approach incrementally the fun factor through different iterations.

The biggest replayability value was the one with different characters simply because the game had different start positions with different characters and different rooms generated each time. In this way, the tactical approaches that the players would embrace will be different for each game, leading to different playtime experience at each run.

Design experiments

Our design experiments method consisted in establishing the hypothesis that procedural generation content can enhance player experience up until a specific point, where hand crafted content starts to take over. We used the iterations we have built for this experiment together with the testing methods presented above in order to confirm the hypothesis. The immediate confirmation that we had was in the verbal communication series where, through our questions and the feedback given by the users, we could assert that this hypothesis was confirmed. When the users started to play the second iteration and they lost the sense of time, we draw the conclusion that procedural generated content helps with a lot with the content addition. When they said that the impact on different characters was not as big as the impact of procedural generated rooms, the hypothesis was confirmed.

By comparing the iterations, one after another, we could see where the line is being drawn. After procedural content was added for the first time(in the second iteration), players have a totally different notion of gameplay and they wanted to play more. After characters were added, their gameplay was enhanced from their tactical point of view. After random positions were generated, they knew what to expect so procedural content generation didn't helped too much in this case(as it was almost the same as before). At this point, a different hand-crafted technique should be employed in order to enhance the gameplay(like unique characters, abilities or missions). By trying different dungeon maps, the gameplay experience was enhanced but not with an impact as before in the sense that the users knew what to expect and this variation didn't come up with a new gameplay factor.

We gathered the results in the surveys and we observed that for them, the procedural generated rooms with the same characters and procedural generated rooms with different

characters at random positions almost got the same score. This means that the huge gap was made from the first iteration to the second iteration where the procedural generated content helps building a better player experience. This third fact confirms the hypothesis.

Another major observation that should be drawn from the rest of the survey answers is the way procedural generation content should be used. The main difference between procedural generation content and hand crafted content is that hand crafted content has more soul and unicity, while the procedural generated content can be repetitive sometimes. By analyzing the rest of the survey answers we can conclude that procedural generated maps have a very big impact over the players . After we tried to add different ways to generate data(like the position of the player or different map dimension), the impact over the player was almost the same. Games who implement very good a single element which consist of procedural generated data have a much higher rate of success than these who add multiple techniques in order to add more content. This observation is drawn from the first, third and fifth question of the survey, in combination with the observations given to us by the players.

CONCLUSION

Our research proved the fact that procedurally generated content can enhance the player experience when used right. There is a thin line between effectively using procedural content generation in order to maximize gameplay replayability and using procedural content generation and not achieve any impact.

Different types of procedural content generation affect different parts of gameplay: while procedural character generation can emphasize the relationship of the player with the world around him, procedural level generation can open up different tactical possibilities and let the user learn the game mechanics, not the levels. This will determine the user to think more and to expand its way of thinking.

The experiments carried out support this fact and are a strong basis of proving that procedural content generation should be much more employed in games and used as a tool by different developers, rather than taking their jobs.

As further work, we plan to experiment with different objectives and gameplay mechanics and to see a way the procedural generated content can enhance these capabilities. Also, we can employ an artificial intelligence in order to generate a level to see its impact over the player.

ACKNOWLEDGMENTS

This research has been carried out in the Computer Graphics and Interactive Systems Laboratory (CGIS) of the Computer Science Department, in The Technical University of Cluj-Napoca.

REFERENCES

1. Togelius, Julian, et al. "What is procedural content generation?: Mario on the borderline." *Proceedings of the 2nd international workshop on procedural content generation in games*. ACM, 2011.
2. Shaker, Noor, Julian Togelius, and Mark J. Nelson. *Procedural content generation in games*. Springer International Publishing, 2016.
3. Fyn, Amy F. "Sources: Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming." *Reference & User Services Quarterly* 52.4 (2013): 353-353.
4. Wittgenstein, Ludwig. "Philosophische Untersuchungen I Philosophical investigations (GEM. Anscombe & R. Reesh, Eds.,)." (1953).
5. Usability First: Usability Glossary: playability Archived 2009-10-18 at the Wayback Machine. - http://www.usabilityfirst.com/glossary/term_657.txl
6. Shaker, Noor, et al. "Constructive generation methods for dungeons and levels." *Procedural Content Generation in Games*. Springer, Cham, 2016. 31-55.
7. 7 uses of procedural generation that all developers should study(Available: May 2018) - https://www.gamasutra.com/view/news/262869/7_uses_of_procedural_generation_that_all_developers_should_study.php
8. Cognitive Flow: The Psychology of Great Game Design(Available: May 2018) - https://www.gamasutra.com/view/feature/166972/cognitive_flow_the_psychology_of_.php
9. Mike Ambinder March 2009. "**Valve's Approach to Playtesting: The Application of Empiricism**." Game Developer Conference