Interactive Evolution of Levels for a Competitive Multiplayer FPS

Peter Thorup Ølsted
Center for Computer Games Research
IT University of Copenhagen
2300 Copenhagen, Denmark
Email: peter.oe@gmail.com

Benjamin Ma
Center for Computer Games Research
IT University of Copenhagen
2300 Copenhagen, Denmark
Email: benjamin@ma.dk

Sebastian Risi
Center for Computer Games Research
IT University of Copenhagen
2300 Copenhagen, Denmark
Email: sebr@itu.dk

Abstract—Traditionally a dedicated level designer has to create every aspect of a level by hand, distribute and gather people to test it. Especially for multiplayer maps, the cyclic process of developing and testing can be quite cumbersome and time consuming. This paper presents a novel approach to provide live generation of levels for such multiplayer maps through procedural content generation and interactive evolution. The approach is demonstrated in the FPSEvolver video game that aims at replicating the look and feel of popular games like Counter-Strike. Without leaving the game a group of players can generate, play and improve levels to fit their particular preferences by voting on a selection of evolving levels. This approach focuses on maintaining the level design principles for modern first-person shooters that encourages good engagements between players, in the popular game mode “bomb defusal”. The presented approach can generate enjoyable maps that adapt to the preferences of players across a range of skill levels. Several distinct types of levels were created during testing, which range from open to closed, and complex to simple, each fitting closely with what the different players consider a good map.

I. INTRODUCTION

The first-person shooter (FPS) genre has increased in popularity over the years, especially in the competitive scene that embraces the fast and precise gameplay. However poor level design can quickly turn off players. Developers often attempt to provide as much content as possible in multiplayer modes to keep players engaged, but designing even a single level can be very time consuming. After the level has been mocked-up, user feedback needs to be gathered before the level in its final state can be build. Targeted playing styles can vary greatly for different levels, and by designing for all styles at the same time, a designer might end up prioritizing differently than what the player prefers. Additionally, depending on their skill level, different players might favor maps with very different complexities or other particular qualities.

A promising technique for automatically creating new game content is procedural content generation (PCG; [1], [2], [3]). In PCG-based games the content can either be created before the game starts or when needed as the game progresses. Especially evolutionary computation and other search-based approaches can save on development costs when combined with PCG.

The approach presented in this paper aims at generating multiplayer levels that tie in with the state-of-the-art gameplay and level design of games like Counter-Strike1 and Call of Duty2. While PCG algorithms have been applied successfully in a variety of different domains, PCG in multiplayer FPS games is less explored and faces some unique challenges. In contrast to single-player games, multiplayer games allow for interesting social interactions as they provide a context for human interaction [4]. Multiplayer FPS gameplay is often acted out on the same maps repeatedly, while most single-player games feature a narrative that tends to be played only once. Thus players in multiplayer games can become highly familiar with the provided maps. Repeated play also encourages players to learn to predict the movement of the opponents, which in turn rewards strategic planning. However, while some players prefer to play the same level over and over again, others might rather play new maps or variations of known maps that offer the right balance between familiarity and novelty.

Creating variations on a theme is a property naturally supported by evolutionary systems. Thus, the combination of search-based PCG algorithms together with interactive evolution offers a new intriguing opportunity for FPS map creation: By allowing a group of players to interactively and collectively evolve multiplayer levels through voting, they can guide the evolutionary search towards map content they prefer. Importantly, the players can decide themselves how familiar or how novel the level to be played next should be.

The main result is that players are indeed able to create levels that often match the preference of the particular group of players, from open to closed levels or complex to simple ones. Additionally, players generally report that the quality of levels improved as the game progressed. The results show that it is possible to automatically create maps for a multiplayer FPS game that players rate as enjoyable, thereby opening up an exciting new direction for PCG-based video games.

1Copyright 1999 Valve Corporation
2Copyright 2003 Activision
II. BACKGROUND

PCG allows game elements (e.g. maps, textures, items, quests, etc.) to be generated algorithmically rather than through direct human design [3], [5]. This approach can reduce design costs and can also benefit players by providing them unique experiences every time they play. For example, the popular Diablo series\(^3\) features procedurally generated dungeons that players explore as a central focus of the game. Like Diablo, many other PCG approaches similarly rely on a fixed set of parameters and randomness to generate content within a heavily constrained space of possibilities. However, a recent focus is to apply artificial intelligence approaches to enable more open-ended generation of PCG.

In particular, evolutionary computation and other search-based approaches [3] can limit the need for hand-designed rules, and may thus further save on PCG development costs. More interestingly, it also enables design of new content outside the scope of a fixed space of rules. One popular technique is interactive evolutionary computation (IEC) [6], in which the user in effect guides an evolutionary algorithm.

An example of IEC applied to games is provided by NeuroEvolving Robotic Operatives (NERO [7]), in which players guide the evolution of a team of fighting robots. In another example, Galactic Arms Race (GAR [8]), weapons are evolved automatically based on user behavior. Further examples include Avery et al. [9], who evolved several aspects of a tower defense game, Shaker et al. [10] who evolved levels for the platform game Super Mario Bros, Togelius et al. [11], who experimented with evolving the rules of the game itself, and the social game Petalz, in which players can interactively evolve PCG-encoded flowers [12], [13].

Only a few examples exist of PCG methods for FPS map generation. Cardamone et al. [14] tried to generate interesting multiplayer deathmatch levels with simulated agents. In their work the fitness of a map is determined based on how long the bot survived after it engaged the enemy. While the maps generated in Cardamone et al. [14] might be fitting for deathmatch play, they are less suitable for a Counter-Strike like gameplay. The levels contain dead ends and no specific areas are more dedicated to engaging opponents than others. On the other hand, the approach presented in this paper tries to create maps that funnel players towards likely engagement locations. More importantly, while Cardamone et al. evolved levels guided by a fitness function, the levels in this paper are created through IEC by a group of players. The promise of this approach is that it should allow the system to generate high-quality levels that adapt to the players’ preferences across a wide range of skills, guided by their personal preferences.

III. APPROACH: INTERACTIVELY EVOLVING MULTIPLAYER FPS MAPS

In this paper maps for a novel FPS game, called FPSEvolver, are interactively evolved by a group of players. The basic game loop proceeds as follows: (1) Players select a map from a list of PCG generated maps by voting on them (the map with the highest number of votes is selected), (2) players play the map, and (3) variations of the played map are evolved and presented to the players. The idea behind the presented approach is that players can decide together in which direction the evolutionary search should proceed. In this way, the system should be able to evolve levels that match the particular skill level and preferences of the group. Interesting questions in this context are if the group as a whole can guide the search to maps that most players prefer and if reaching consensus through a voting process is possible. In addition to creating maps tailored to a group of players, PCG also offer a constant stream of novel maps that should help in keeping the players engaged longer.

In multiplayer FPS games, many different game modes exist that differ in how the game is played. A game mode made popular by the video game Counter-Strike is “bomb defusal”, in which an attacking team of players tries to plant a bomb that the defending team has to defuse. This particular game mode was chosen for the experiments in this paper because it encourages more tactical planning and teamwork than the basic deathmatch mode. It is advantageous for either team to quickly familiarize themselves with the level to gain an upper hand in confrontations, by taking alternative routes for flanking or simply the shortest path. When killed, players do not spawn before one team wins. At the beginning of the round the bomb is given to a randomly chosen player of the attacking team who then has to plant it in one of the two predefined plant locations. If the attacking team successfully plants the bomb, they have to prevent the defending team from defusing it for a certain amount of time until the bomb explodes. If the bomb is not placed and all players on a team are killed, the surviving team wins. After a few rounds, the two teams are swapped, making the attackers the new defenders and vice versa.

A. Game Development

FPSEvolver’s basic game mechanics are built on a pre-made and easily modifiable system for the Unity Game Engine, which provides modern shooting- and movement mechanics. The *Realistic FPS Prefab* by Azuline Studios [15] has most of the features needed for a FPS game and resembles Counter-Strike and Call of Duty in both visual elements and gameplay. Local networking is handled using the standard solutions in the Unity Game Engine. Figure 1 shows an example of the player in the game.

B. Key Principles in Good Multiplayer Map Design

FPS maps follow some relatively simple but effective guidelines. In order to create engaging maps through a PCG-based approach, it is vital to understand what these guidelines are. In this context, we introduce the term *the good engagement* (TGE), a level experience in which the intended strategic and skill-based gameplay shines through. TGE does not mean that the game is fun to play, but rather that the level design...
supports and encourages interesting choices, from which a fun gameplay experience should emerge naturally. In this paper, the map representation (explained in Section III-D) and the maps shown to the player to pick from are influenced by TGE.

In good “bomb defusal” maps the bomb positions, spawn points, paths and choke points should be placed in a balanced way. A level should offer different types of engagements to suit a variety of playing styles. Because some players prefer sniping from long distances, while others prefer quick reflexes in close encounters, a map should have short, medium and long engagements.

For example in Counter-Strike interesting choices often occur around the choke points (i.e. areas where the two opposing teams will likely encounter each other). At the start of each round the two teams have to decide at which choke point they want to engage the enemy. The defenders can normally reach the bomb positions before the attackers arrive and it is usually possible for the attackers to get to another choke point without confronting the defenders. The exact locations of engagements change slightly depending on the playstyle of the teams. In terms of the playing experience, the choke points are map locations with a high probability of encounters; this is where the fun in the game lies for most players. Since most levels only have two to four choke points, the number of sensible strategies for each team are limited. When building competitive levels, determining the location of choke points is one of the keys to good engagements between players. Playing a map repeatedly gives players a familiarity with both the map and how it is usually played. With this knowledge the player can make “interesting” choices about how, when and where to engage the enemy to create an advantageous situation. In the “bomb defusal” mode it includes when to plant and defuse the bomb in safety from the opposing team.

The levels in Counter-Strike are often constructed using very angular designs in the overall layout, thereby highlighting the paths of the map. An interesting observation is that levels created for “bomb defusal” do not feature four-way intersections, since they would provide players with too many choices when moving towards their target destination. Call of Duty has both levels that are very open as well as corridor-based; Counter-Strike is almost exclusively corridor based.

Our analysis of Counter-Strike and Call of Duty maps and their multiplayer gameplay revealed five key aspects when designing levels for “bomb defusal” modes. These rules are what we determine important to maintaining TGE in procedurally generated levels:

- Maps should always have a few choke points, which the teams can reach at about the same time.
- Bomb locations should always be closer to the defending team, giving the defenders a small advantage.
- Bomb points, as well as spawn points, should never be visible from one another.
- Levels should not contain too many choices, especially in the form of four-way connections.
- Spawn points should be in opposite ends of the level to make sure that encounters do not start too early.

C. Selecting and Evolving Levels

In the approach presented in this paper players can vote on a selection of six maps presented to them (Figure 2). Inspired by Cardamone et al [15], players can rate levels by either a thumbs up or down indicating their preference. After the players are done voting, a score is calculated for each map based on the votes of all the players in the current group; a positive vote increases the score of a map by one, and a negative reduces it by one. The map with the highest score is chosen as the next one to be played. A video showing the voting process together with footage from a two-player match can be found here: http://youtu.be/V6eMktY15so.

In order to not present the players with particular bad levels, new levels are generated until six levels are found that adhere to the rules of TGE. In particular, the filter removes levels with (1) too long corridors without any choices (longer than 30 game units), (2) spawn points that are immediately visible...
from one another, (3) bomb position too close to one another (less than 20 units), (4) imbalanced maps where one team has a clear advantage (more than 85% of the level is closer to one team), (5) spawn points that are not reachable from one another, and (6) levels with either too high or too low map complexity $C$, which is calculated as:

$$C = n + r,$$

where $n$ is the number of three-way connections and $r$ is the number of room entrances/exits.

A refresh button is also part of the interface, allowing players to generate a new set of levels if they are not satisfied with the current selection. The button follows the same voting rules as the levels (i.e. it needs the most votes in order to be activated). To ensure that levels always improve in a lineage, players are asked if the current level is better than the parent level after each match. If the new level is deemed better the lineage continues with the current level. However, if the level receives more negative than positive votes, the lineage reverts back to the parent, and only progresses once a better offspring has been found. A summary of the game flow is shown in Figure 3.

### D. Map Representation and Generation

In search-based PCG the representation should allow a variety of different maps to be expressed, while also generating maps that appeal to the user. The general aim of the developed map representation is to resemble the layout and perpendicularity of Counter-Strike levels. The basic building block of the map genotype is a list of pairs of two-dimensional points (which can have variable length). Additionally, the genotype specifies if these pairs of points should be connected first via a horizontal or a vertical line (Figure 4a). The resulting L-shaped corridors are snapped to a grid to ensure that nearby shapes are grouped together (Figure 4b). The size of the map and the grid resolution are also encoded in the genotype.

Once all the pairs are connected into L-shapes, the lines begin to resemble the structure of a level. To conform with the TGE rules established in Section III-B, the algorithm modifies these genetically determined shapes. In the first step all dead ends are eliminated (Figure 4c) and four-way connections are reduced to three-way connections (Figure 4d). Similarly to the setup of Cardamone et al. [14], the layout is then carved out from an all-black and impassable space (Figure 4f). Corridors have a variable width that is part of the genes encoding the L-shapes. Areas, which are also encoded in the genotype by their position, width and height are carved out as rooms (Figure 4g), and props (e.g. crates to hide behind) and doorways (encoded by their position, the relative width of the opening and the direction that the doorway is facing) are placed in the corridors. Spawn points are positioned as close as possible to their encoded position (Figure 4h). The generated level layouts are then analyzed, to determine which team is closest to each position in the level. This analysis also determines the location of possible bomb positions, which should be reachable faster for the defending team than the attacking team (Figure 4i). From this list, two bomb locations are chosen, based on an integer value encoded in the genotype.

Mutations on the genotype can create new L-shapes, remove existing ones, create new props or modify the values of existing items. Figure 2 demonstrates that the encoding can generate levels with modest and more significant phenotypic changes that resemble the parent level to varying degrees. Importantly, small changes in the map’s genotype typically produce small changes in the resulting phenotype. Additionally, while less polished, the maps resemble the layout and perpendicularity of Counter-Strike levels (Figure 5). It is important to note that the encoding is deterministic, since all the values necessary to create the map are encoded in the genotype itself.
IV. EXPERIMENTS

A total of three playtests were conducted with players who had at least some prior experience with FPS games (Table I). For the second (pt2) and third playtest (pt3), the testers filled out a questionnaire after they played the game, while the first playtest (pt1) served only as a verification of the technology and no data was recorded. The quantitative answers were either yes-no questions or a rating between 1 and 6, where 1 was the lowest and 6 is the highest score. The maps played were recorded for pt2 while every map, vote, rating, kill and death was recorded for pt3.

The second playtest had session with three separate groups of four to five people (pt2.1, pt2.2 and pt2.3), each lasting 90 minutes. For the first hour the groups played the game and evolved levels. Every sixth or seventh generation the game was reset to start a new lineage of maps. After an hour of playing the game, the participants filled out a questionnaire.

For pt3 the mutation probabilities were increased slightly, and the voting system was altered, allowing players to vote on the map first and then on the bomb position. Additionally, while players in pt1 and pt2 were only allowed to vote for a single map, the final voting system allowed them to rate multiple levels. This change in the voting system aimed at minimizing dissatisfaction among voters [16]. The third test focused on gathering relevant gameplay and player data, to analyze if there are any correlations to the questionnaire data. Votes and ratings were recorded to follow the progress of maps and how they evolved. After an hour of playing the game, the participants filled out a questionnaire.

The questionnaire in pt2 asked the players whether or not the maps’ quality increased over the sessions, whereas in pt3 we also asked if the maps got worse. Not a single participant felt the levels got worse and the majority of players stated that the quality actually improved over the course of evolution. The high number of positive ratings suggests that progressively likeable levels were created (Figure 8), even with many people voting at the same time.

In pt3, the average player placed over 100 votes in total, with roughly 58 positive and 46 negative votes. The new voting system introduced in pt3 provided the user with multiple votes and increased the perceived impact for voting from 4.23 on average (sd=1.09) in pt2, to 4.8 (sd=1.34) in pt3. While not statistically significant (according to the Student’s t-Test), this increase could be explained by a minimization of dissatisfaction among voters [16].

<table>
<thead>
<tr>
<th>Playtest</th>
<th>Shorthand</th>
<th>Skill level</th>
<th>Player count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second test</td>
<td>pt2.1</td>
<td>Skilled</td>
<td>4 players</td>
</tr>
<tr>
<td>Second test</td>
<td>pt2.2</td>
<td>Medium skilled</td>
<td>5 players</td>
</tr>
<tr>
<td>Second test</td>
<td>pt2.3</td>
<td>Low-medium skilled</td>
<td>5 players</td>
</tr>
<tr>
<td>Third test</td>
<td>pt3.1 preliminary</td>
<td>All skill levels</td>
<td>11 players</td>
</tr>
<tr>
<td>Third test</td>
<td>pt3.1</td>
<td>All skill levels</td>
<td>13 players</td>
</tr>
<tr>
<td>Third test</td>
<td>pt3.2</td>
<td>Skilled</td>
<td>6 players</td>
</tr>
<tr>
<td>Third test</td>
<td>pt3.3</td>
<td>Unskilled</td>
<td>7 players</td>
</tr>
</tbody>
</table>

TABLE I: An overview of all playtests, their names, skill ranges and player count.

V. RESULTS

An interesting observation is that, despite a wide range in FPS experience, the groups were able to agree on a direction that the level design should take. Players with skills ranging from 2-6 (where 6 is highest) in pt2.2 and 1-4 in pt2.3 still managed to reach consensus, without the players explicitly agreeing on any design.

Group pt2.2 reported a clear preference for open spaces in their questionnaire responses (Figure 7). Figure 6b, which shows the final map of pt2.2 suggests that the system is able to capture their preferences; the map has more open space than corridors. In fact, in the discussions following the test a player mentioned that the last level was particularly fitting with his preferences. In group pt2.3 (Figure 6d), the level did not evolve to something complex but instead to a smaller and simpler levels. This again was in line with what the players preferred, according to their questionnaire answers on what constitutes a good level. These results suggest that the evolutionary approach is often able to generate levels that fit with the preferences of the particular group of players.

While the final maps of pt2.2 and pt2.3 matched the preferences of the players closely, the final maps of pt3 also matched the descriptions but not as accurately. This discrepancy is likely due to the limited number of generations that players were able to evolve maps for in pt3 (because of network issues only three generations in pt3 compared to six in pt2).

The questionnaire in pt2 asked the players whether or not the maps’ quality increased over the sessions, whereas in pt3 we also asked if the maps got worse. Not a single participant felt the levels got worse and the majority of players stated that the quality actually improved over the course of evolution. The high number of positive ratings suggests that progressively likeable levels were created (Figure 8), even with many people voting at the same time.

In pt3, the average player placed over 100 votes in total, with roughly 58 positive and 46 negative votes. The new voting system introduced in pt3 provided the user with multiple votes and increased the perceived impact for voting from 4.23 on average (sd=1.09) in pt2, to 4.8 (sd=1.34) in pt3. While not statistically significant (according to the Student’s t-Test), this increase could be explained by a minimization of dissatisfaction among voters [16].
Fig. 6: The maps of the first and last generation for pt2.2 and pt2.3. The main result is that the IEC system can create maps that suit the preferences of the particular group. Players of medium skill (pt2.2) preferred open spaces, and disliked long corridors, and where able to guide the evolutionary search towards such a map (b). Players of low skill (pt2.3), on the other hand, preferred less complex levels and shorter corridors, which is also reflected in their final map (d).

What seemed to constitute a good map?

- Good mix of open and closed spaces, distance between bomb points, multiple paths so gameplay doesn’t become monotone.
- A good mixture of open areas (with some stuff to hide behind), choke points and long corridors. Also multiple ways to get around the map is good for keeping the match from becoming a stalemate.
- No long corridors. Bomb places placed in opposite sides of the map. Lots of small chunks of block in open areas that would allow you to take cover easily.
- Not too long pathways to arrive at a bomb site. Only at one certain point, did arriving to bombsite B take way too long path which was difficult to arrive at. Open spots in the middle of the map were fun because it created a battlefield rather than just corridors.
- Different paths that lead to big areas, so you can plan from where to attack the enemy and surprise them. Big areas with more covers, to strategically hide and take cover. . . .

Fig. 7: The answers of players in pt2.2 on what constitutes a good map. The evolved map of the last generation can be seen in Figure 6b. Notice that every tester mentions open areas and how the final map contains many open areas.

What seemed to constitute a good map?

- A good mix of open and closed spaces, distance between bomb points, multiple paths so gameplay doesn’t become monotone.
- A good mixture of open areas (with some stuff to hide behind), choke points and long corridors. Also multiple ways to get around the map is good for keeping the match from becoming a stalemate.
- No long corridors. Bomb places placed in opposite sides of the map. Lots of small chunks of block in open areas that would allow you to take cover easily.
- Not too long pathways to arrive at a bomb site. Only at one certain point, did arriving to bombsite B take way too long path which was difficult to arrive at. Open spots in the middle of the map were fun because it created a battlefield rather than just corridors.
- Different paths that lead to big areas, so you can plan from where to attack the enemy and surprise them. Big areas with more covers, to strategically hide and take cover. . . .

Fig. 8: Players for pt2 and pt3 were asked if they thought the quality of levels improved over time. The main result is that the majority of players agreed that the levels got better as evolution progressed.

A. Level Complexity

Not surprisingly, players of higher skill level tended to prefer more complex levels. The average complexity of a level voted on had a positive Pearson’s correlation coefficient of $r=0.3$ to the skill level of the player, indicating a weak positive linear relationship. Figure 9, which shows the complexities of levels as they evolve, suggests that there are different target complexities depending on the testers involved. While most maps saw some change in complexity as they evolved to fit with the overall preference of the players, the map complexity for pt3.1 with 13 players remained steady throughout the test. In pt2.2 a short increase in complexity can be noticed after which it settles at 17. What is not apparent from these numbers, is the increase of open areas in this particular playtest; the complexity metric only takes into account the amount of three-way connections and room entrances (Equation 1). Therefore, the addition of open areas does not necessarily change complexity significantly. For pt3.3 and pt2.3 a continuous change in complexity throughout the sessions is observed, with upwards and downwards trends.

B. Player Satisfaction

The overall player satisfaction appears to be quite positive. Almost all players agreed that the game was representative of an FPS, and the enjoyment of the game was high in pt3 and pt2, with an average score of 5.15 (sd=0.66) and 4.54 (sd=1.39) on a scale of 1–6. Correlations in the data show that there is a link between the player’s performance (measured in kill-death ratio), and the enjoyment of the game ($r=0.58$). Interestingly, there is also a clear correlation between reported player enjoyment and increase in reported map quality as the game progresses ($r=0.87$).

VI. DISCUSSION AND FUTURE WORK

The results from pt2 and pt3 suggest that the players’ choices actually allowed them to evolve increasingly better suited levels. The overall quality had an average score of 4.07 (sd=1.07) and 4.31 (sd=0.91) in the two tests on a scale of one to six. 61% and 77% agreed that the levels improved in quality during the test sessions. In pt3 we also asked whether or not the quality of maps decreased and not a single testers
stated it did. The best levels received an average score of 4.8 (sd=0.97) and 5 (sd=0.78). The qualitative answers and the evolved maps suggest that FPSEvolver offers an interesting new alternative in the creation of engaging multiplayer maps.

The voting system for pt2 and pt3 changed between playtest. Instead of a single voting screen determining the layout of bomb and spawn positions, the bomb positions were now on a second voting screen. For pt2 and pt3 most players agreed on the question “Did you feel voting could impact the bomb positions?”. In pt2 23% of players did not feel they could impact the bomb positions, while only 15% in pt3 felt the same way. The fact that not all players agreed could indicate that the six variations presented where too similar to one another. As the bomb positions had to be placed closer to the defenders, it did only leave about half the map for the bomb positions, and thus not a lot of possibilities for variations.

Correlations exist between the skill levels and enjoyment of the game (ranging from 0.46 to 0.61) and the personal perceived impact (ranging from 0.45 to 0.71). It is possible that a less enjoyable game experience is due to a lack in perceived impact when voting. Playing in the bigger group with 13 players in pt3.1, might have also negatively influenced the playing experience for some players. It is possible that the inclusion of the long ranged weapon gave new purpose to these long corridors that had none in pt2, thereby changing the preferences of the players. In fact, the sniper rifle was included due to a number of players in pt2 complaining about long corridors, which is something that is not necessarily considered negative in Counter-Strike or Call of Duty.

We sometimes saw widely different results between pt2 and pt3 in regards to the player’s reported enjoyment of the game, and in some of the correlations. It is difficult to conclude why, but we suspect that the treatments of the different playtest groups could have had an impact. For example in pt2 the three groups were all separated, unlike pt3 where all players would first play as a group. During pt3.1, which in part was used to determine the players’ skill level, an imbalance appeared early on in the team arrangements. These imbalanced teams likely decreased the enjoyment for weaker players that now lost more often. Additionally, the round duration was longer in pt3.1 and pt3.3, which means that players that died early had to wait longer to rejoin the game.

Another difference in the two playtests was the delivery of instructions. For pt2 we had printed instructions, and placed them at each computer. In pt3 we had instead visual instructions in the menus of the game, which some players missed when entering the game. The instructions for the voting screens were delivered orally in both tests, but having a larger crowd of people was more difficult than anticipated, resulting in some players missing important information.

The generated levels for pt3 have noticeably fewer generations per session than pt2. This makes it harder to evaluate the evolutionary progress for this playtest. Had it not been for network overload issues during the playtests, it is possible that the maps in pt3 would have been more in line with the results from pt2. Adaptation to group preferences seem to increase in general as the lineages progress. Additionally, a higher sample size would have been preferable to further verify the results with more than the 19 unique participating players.

For the three playtests we invited people with different backgrounds and skill levels but all participants had an interest in games. A few players had very little interest in FPS games in particular, but everyone was familiar with how to play them. During the play test we only distinguished between skilled and low-skilled players. Some players had a background in game development and some with level design experience. Some testers were skilled FPS players but did not have any game development background, while others had a level design background with poor FPS skills. Some testers were also returning from pt2, and others had only tested the game once. Despite the mix in player skills, the results indicate that players can eventually reach an agreement in which direction evolution should proceed. However, tests with very specific
skill levels or preferences might have yielded a much more rapid evolutionary adaptation and perhaps very different maps.

In the future it could also be interesting to evolve levels based on multiple parents, instead of simply selecting one. Additionally, all votes could have an influence on the evolutionary progress instead of a simple winner-takes-all solution.

An important future challenge is to extend the current local networking approach to a larger user base through online play. Not only could this garner different results due to the lack of physical presence with other people, but could also provide a substantially larger data set for further analysis. The lineages of maps evolved in this paper were quite shallow, with the longest lineage encompassing six generations. Playing online could allow for more generations than were possible during local play. Here evolution also has to deal with a potentially changing group of players with very diverse skills and changing preferences from round to round.

VII. CONCLUSION

We presented a novel approach to create multiplayer FPS maps using interactive evolution and procedural content generation, which adapts quickly to the preferences of a group of players. By rating the levels using simple votes, a group of players can rank the levels, and collaboratively evolve maps based on qualities they prefer. Across a range of skill levels, almost all players reported the levels were of higher than average quality and increased in quality as they played the game. The evolved levels matched the players’ preferences closely and most players expressed enjoyment with the game in general. In the future, an online version of the presented approach could produce a continuous stream of novel and distinct maps for a wide range of different players.

REFERENCES


