Applying Data Mining to Extract Design Patterns from Unreal Tournament Levels

Luca Galli  
Dipartimento di Elettronica, Informazione e Bioingegneria  
Politecnico di Milano  
Email: luca.galli@polimi.it

Pier Luca Lanzi  
Dipartimento di Elettronica, Informazione e Bioingegneria  
Politecnico di Milano  
Email: pierluca.lanzi@polimi.it

Daniele Loiacono  
Dipartimento di Elettronica, Informazione e Bioingegneria  
Politecnico di Milano  
Email: daniele.loiacono@polimi.it

Abstract—We present a study on the application of data mining to extract design patterns from Unreal Tournament III levels used in the online gaming scene for *Duel* and *Team Deathmatch* games. The maps’ topological structure and their morphology was extracted using ad hoc bots we developed and several statistics have been computed using typical graph algorithms. The process resulted in datasets containing information about all the relevant positions in the maps (the nodes in the game navigation mesh used in the Unreal game engine) and their role in the game (i.e., whether they are navigation points, ammo pickups, weapon pickups, or powerup pickups). We have applied four data mining algorithms to the data to characterize both (i) the maps’ type (*Duel* and *Team Deathmatch*) based on the feature of the nodes they contain and (ii) the node types (ammo, weapon, powerup, or navigation) based on their features. Our results suggest that the maps’ type can be characterized in terms of the nodes they contain but it is difficult to characterize the role of nodes based solely on their features.

I. INTRODUCTION

First-person shooters (FPS) are combat-oriented games in which players navigate complex 3D worlds and engage opponents by exploiting a variety weapons, special powerups, and the knowledge of the territory. Levels in FPS games are usually designed for specific gameplay modes, e.g., single-player, multi-player, capture the flag, duel, team deathmatch, and many others. Single-player levels contain a series of challenges designed to obstacle the player ability to reach a specific goal like for instance reach the end of a level/maze, rescue a character or recover an item. Multi-player levels are designed to create suitable areas to foster competition among players. Although the level structure is the simplest aspect to consider when evaluating the level designers’ work, other aspects that influence the gameplay must be taken into account. For instance, level designers place objects in the world (weapons, ammunition, and powerups) to assure a smooth, balanced and engaging experience (with interesting but not overwhelming challenges) by taking into account the number of players involved and the target pacing; they also structure the environment to bias players exploration and to make the access to some level areas easier or more difficult.

There are several papers and books that discuss good principles of game and level design [1]–[4], however only quite recently researchers have tried to systematically identify and study game design patterns [5]–[10]. The first in-depth study of game design patterns is due to Björk et al. [5], who identified a set of patterns of players’ and game elements interaction. In the context of FPS games, Hullett and Whitehead [11] present a taxonomy of design patterns that appear in single-player first-person shooter (FPS) levels. Hullett [12] also performed a user study to examine common design patterns in single-player first person shooter (FPS) levels and analyzed cause-effect relationship between game design patterns and gameplay.

In this work, we applied data mining to extract design patterns from a set of Unreal Tournament III (UT3) levels selected from the most popular ones in the game scene for *Duel* and *Team Deathmatch* modes. UT3 has been developed on top of the Unreal Engine using the *Unreal Script* programming language. Most of the scripts developed for UT3 are publicly available and can be modified to modify the game. Thus, although the source code of the Unreal Engine is not available, the game itself can still be customized/extended using the proprietary script language. Using Unreal script we developed custom bots to extract information from UT3 maps (whose proprietary format makes them inaccessible unless the UT3 editor and manual inspection is used). In particular, our bots gathered information about the maps’ navigation mesh structure (the position of navigation points and the existing connections), the maps’ morphology using bots equipped with raytrace sensors (the shape of rooms and open spaces, the width of connecting corridors, the time needed to explore them, etc.), and about the position and characteristics of all the pickup points (weapons, ammunition, and powerups). We applied descriptive and predictive data mining to extract interesting patterns from the data (i) by building models that could determine the game mode associated to a map based on the features of the nodes it contains, and (ii) by building models to predict the type of a node (navigation, weapon pickup, ammo pickup, or powerup) based on its characteristics (e.g., its connections to surrounding nodes, its role inside the map, etc.). Our exploration of the collected map data provides a quantitative evaluation of the distribution of weapons in UT3 maps that confirm some of the community knowledge discussed in well-known forums [13]. The predictive models for the map type show that it is possible to characterize the type of map based on the nodes it contains and that some of the attributes (namely, the node pagerank score, authority, and
hub value) are the most relevant features to determine the map type. In contrast, the data does not allow to characterize the node type (whether it is a navigation point, an ammo pickup, a weapon pickup, or a powerup pickup) based on its features. In fact, the predictive models we derived tend to classify all the nodes as navigation and thus cannot distinguish among navigation nodes and the other three types of nodes. Our results suggest that it is possible to characterize 

\textit{Duel} and \textit{Team Deathmatch} maps based on the features of the pickup points they contain but it is difficult (impossible for us) to distinguish the role of a node in a map solely based on its features.

The paper is organized as follows. Section II briefly overviews the relevant works on game design patterns. Section III describes Unreal Tournament III and the Unreal Engine while Section IV introduces the \textit{Duel} and \textit{Team Deathmatch} game modes used in this work. Section V discusses how the data have been collected and Section VI reports on the experimental results.

## II. LEVEL DESIGN PATTERNS

The study of game design patterns is a rather recent but rapidly growing hot topic in the game development community. Indeed, there are several papers and books that discuss good principles of game and level design, however only quite recently researchers have tried to systematically identify and study game design patterns \cite{5}–\cite{10}.

Early works related to the study of patterns for level design include \cite{1}–\cite{4}, \cite{14}. In particular, Chen et al. \cite{14} compared level design to the architectural design that is used in real world buildings. In \cite{14}, the authors identify some architectural principles that level designers may apply to create \textit{spaces} that enable engaging gameplay, e.g., having a clear path through the level, how to use different spatial organizations such as linear or hub-and-spoke, or including unique elements to break up the design. Clayton \cite{1}, Bryne \cite{2}, and Feil et al. \cite{3} present in-depth analyses of how level design enables gameplay but they do not identify nor discuss design patterns. Co \cite{4} takes the reader through the process of designing a FPS level, from brainstorming initial ideas, to the actual creation using the Unreal Editor, to the testing and the improving of the level. Güttler et al. \cite{15} examined how space is used in team-based multiplayer FPS levels. In \cite{15}, the authors focused on collision points and tactical choices: they identified common spatial configurations and how these contribute to gameplay.

There are empirical studies that evaluate the effects of level design on gameplay. For instance, Tahhan \cite{16} presents an empirical study on directional choices in FPS levels. In \cite{16}, the author identified different techniques for presenting alternate routes and performed user studies on a set of representative levels. Survey responses and subject observations contributed to the conclusion that choice improves player immersion, as the lack of choice in a linear level can break the illusion of being in large, dynamic world. Gee \cite{17} studied the use of dead-ends in FPS levels and identified ways in which dead ends are used; the experiments presented in \cite{17} suggest that dead ends do not have a negative impact on FPS levels.

Björk et al. \cite{5} have presented the first in-depth study of game design patterns in which the authors identified a set of patterns, analyzed how players interact with game elements, and how such elements interact with each other in a game. Although, Björk et al. \cite{5} do not specifically focus on with level design, the patterns they identify are still relevant to tackle the major level design issues (e.g., balancing, goals, locations, and objects). Hullett and Whitehead \cite{11} present a taxonomy of design patterns that appear in single-player first-person shooter (FPS) levels to define characteristics of a pattern language for the domain of level design. Recently, Hullett \cite{12} performed a user study to examine common design patterns in single-player first-person shooter (FPS) levels and analyzed cause-effect relationship between game design patterns and gameplay. Antonios Liapis et al. \cite{18} identified three game design patterns in maps (area control, exploration and balance) providing formulas for measuring the extent to which a level includes these concepts along with evaluation functions for levels in two different game genres: multiplayer strategy game maps and single-player roguelike dungeons. Giusti et al. \cite{19} present preliminary results on the classification of weapons based on the gameplay behaviors they elicit using a language of common weapon design patterns. Dahlskog and Togelius \cite{20}, \cite{21} discussed how procedural content generation and design patterns could potentially be combined in game design and later how to use design patterns in procedural level generation, with particular reference to the classic console game Super Mario Bros.

## III. UNREAL TOURNAMENT III

Unreal Tournament III (UT3) is a best-selling commercial \textit{first person shooter} (FPS) based on the Unreal Engine, a game engine with impressive rendering, accurate game world physics\textsuperscript{1} which has been used in several commercial titles.\textsuperscript{2} UT3 was developed using Unreal Script, a java-like scripting language interpreted by the underlying Unreal Engine. This two-tier architecture separates the development of the gameplay from that of the engine; accordingly, any modification to the engine does not require a change to the scripts implementing the gameplay and vice versa. Therefore, although the source code of the Unreal Engine is not available, the game itself can be customized using Unreal Script. In fact, most of the scripts developed for UT3 are publicly available and can be modified to alter the original gameplay.

There are two major approaches to modify UT3 using the Unreal Script \cite{22}, namely \textit{Mutators} and \textit{Game Types}. \textit{Mutators} are the easiest way to modify almost everything in a game (e.g., the game rules, its goals, the available weapons, the available items and power-ups). \textit{Mutators} are designed to be applied in a chain to combine their effects. Accordingly, there are limitations on game elements that can be modified using a mutator in order to guarantee the compatibility with other mutators. \textit{Game Types} are typically used to change the game completely or when it is necessary to perform operations that are not available with mutators (for instance, to access game

\textsuperscript{1}http://en.wikipedia.org/wiki/PhysX

\textsuperscript{2}http://www.unrealtournament.com
A. UT3 Game Modes

UT3 has several game modes available to players: Deathmatch, Team Deathmatch, Capture the flag, Duel, Warfare, Vehicle Capture the Flag, Betrayal, and Greed. In this study, we focused on the two game modes that are more popular in the competitive scene, Duel and Team Deathmatch.

A. Duel Maps

Duel involves two players that have to either reach a target number of kills (or frags) or the highest number of frags by the end of the match. It is a demanding game mode that requires deep knowledge of the map and of the mechanics of the game to succeed. Duel maps are typically compact and they are restricted to only 4 to 6 weapon types, with 2 spawn beacons each; spawnpoints are spread widely, yet one will normally spawn in front of a weapon. Duel maps usually do not contain super weapons (e.g., the Redeemer which fires a small nuclear warhead), but have one or two hitscan weapons (which when fired instantly hit whatever they are pointing at) and one or two projectile weapons (e.g., a rocket launcher). An important aspect of Duel games is the control and use of pickups (which can be either categorized as health, armor/shield, and powerups) with limited time or number of uses (e.g., the Jump Boots or UDamage). Pickups often come in two sizes, big and small. For example, Thigh Pads provides 50 armor points while the Body Armor provides 100 armor points. As discussed in the UT3 community forums, well-designed Duel maps balance ease of access to pickups against the value of the pickup. In Duel maps, ammo, health, smaller armor and the Jump Boots are usually easy to find; in contrast, powerups like the Shield Belt, SuperHealth, UDamage and Invisibility are usually hard to get and expose the player to risks in the process of obtaining them. There is no requirement for any of the pickups to be implemented in a map.

B. Team Deathmatch

Team Deathmatch (TDM) involves two teams (usually Blue vs. Red) that have to amass more kills (or frags) than the opposing team. The team score is the sum of the frags of each player in the team (computed as the difference between the number of kills and the suicides); the team with the higher frag count wins. Note that, in TDM a single strong player can often lead a team to victory, or conversely if teams are fairly even, a single weak player can drag the rest down.

There are few maps specifically designed for TDM and most of the games are played using maps designed for Deathmatch (DM), the extension of Duel to more than two players. However, the actual TDM maps are often much larger than DM and Duel maps and usually fit 8-16 players, or occasionally 4-24 players. TDM maps for more than 24 players are very rare because of the time required to design them, of their prohibitive download sizes, and for the daunting challenge they pose to balance the gameplay. Because of the size difference, TDM maps tend to be wide open so that hitscan weapons are often dominant. TDM maps seldom have superweapons typically positioned in a difficult-to-reach spot. Depending on the size of the map, there are usually about 6-9 weapon types. Spawn positions are typically near a weapon but fairly far from each other and in somewhat protected areas (for instance, a player will seldom spawn in the middle of an open floor); note that spawn points are not team specific. TDM maps have all standard deathmatch weapons and the number of hitscan weapons is approximately the same as the number of projectile weapons. Low-level powerups are common and usually easy to find, but more powerful ones like the UDamage, SuperHealth, and Belt are either difficult to find or not implemented at all (usually smaller maps will not include them, but larger maps will).

V. Collecting Gameplay Data

In this study, we applied data mining to characterize UT3 maps based on their topology, on the type and position of the pickups and weapons, and the gameplay that the combination generates. For this purpose, we analyzed both static information about the maps’ logical structure and their morphology.

A. UT3 Maps

UT3 maps are created using a special editor that stores them in a proprietary (and inaccessible) binary format containing all the map assets and its description as a set of nodes (the PathNodes), their type, and the edges that connect them. Each map object is described by (i) an ID (a string), (ii) a string that identifies its type (NavigationPoints for normal nodes, PickupFactory for the other items), and (iii) its position (as x,y,z float coordinates). Each object has also specific attributes for the PickupFactory nodes subtypes; Table I and Table II describe all the relevant attributes for the weapons and pickups which can be found in UT3 maps.

B. Maps Logical Structure

UT3 maps can be analyzed manually using the proprietary editor but this approach is clearly unfeasible when several
TABLE I: UT3 Weapons

<table>
<thead>
<tr>
<th>Name</th>
<th>Primary Type</th>
<th>Primary Damage</th>
<th>Primary Rate</th>
<th>Secondary Type</th>
<th>Secondary Damage</th>
<th>Secondary Rate</th>
<th>Weapon Pickup Ammo</th>
<th>Ammo Pickup</th>
<th>Max Ammo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bio Rifle</td>
<td>Projectile</td>
<td>31</td>
<td>1.1/s</td>
<td>Projectile</td>
<td>210</td>
<td>5.8/s</td>
<td>25</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Shark Rifle</td>
<td>Hitman</td>
<td>45</td>
<td>0.7/s</td>
<td>Projectile</td>
<td>25</td>
<td>175/s</td>
<td>20</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>Link Gun</td>
<td>Projectile</td>
<td>20</td>
<td>0.9/s</td>
<td>Hitman</td>
<td>110</td>
<td>5.8/s</td>
<td>20</td>
<td>30</td>
<td>120</td>
</tr>
<tr>
<td>Sniper</td>
<td>Hitman</td>
<td>14</td>
<td>1.0/s</td>
<td>Projectile</td>
<td>36</td>
<td>5.8/s</td>
<td>100</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>Rail Cannon</td>
<td>Projectile</td>
<td>100</td>
<td>1.0/s</td>
<td>Projectile</td>
<td>100</td>
<td>1.0/s</td>
<td>190</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>Rocket Launcher</td>
<td>Projectile</td>
<td>100</td>
<td>1.1/s</td>
<td>Projectile</td>
<td>100</td>
<td>1.0/s</td>
<td>190</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>sniper rifle</td>
<td>Hitman</td>
<td>1.0/s</td>
<td>200</td>
<td>Assim</td>
<td>N/A</td>
<td>N/A</td>
<td>10</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>Redemptor</td>
<td>Projectile</td>
<td>1500</td>
<td>1</td>
<td>Projectile</td>
<td>1500</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

maps are involved. Accordingly, for this study we modified the controller of a custom bot to automatically collect the maps data by exploring all the map PathNodes in-game.

When UT3 loads a map it performs a sequence of calls to the engine to load the map logical description. At this point, our modified bot can collect all the static information about the map which basically consists of the navigation mesh, the position/type of the navigation nodes, and the edges connecting them. For each node, our bot records the node ID, its type (a label), its position, and its SpecialAction (a Boolean flag which is true if the node requires a special action to be reached, e.g., a rocket or hammerjump). For each edge connecting a pair of nodes, our bot records the ID of the starting and ending nodes, the distance computed in Unreal units (an integer), and the type of the edge (an integer), which describes whether a path requires special actions to be followed.

C. Map Morphology

Morphology is a major component of a FPS 3D environment; the presence of open spaces, hallways, small rooms or narrow corridors have an huge impact on the fighting strategies in a first person shooter. Unfortunately, UT3 does not provide access to these information from the map files nor from the description loaded at init time. Accordingly, we devised a bot to explore a map exhaustively to collect the morphology information using methods similar to the algorithms for simultaneous localization and mapping (SLAM) [23] used in robotics. For this purpose, our bot integrates the static map representation (obtained at init time) with the information gathered using a belt of five raytrace sensors positioned at 0 (facing the direction of movement), 45, -45, 90, and -90 degrees, which retrieve the distance from the first object encountered (mesh or opponent). The bot exhaustively explores all the navigation graph using well-known visit algorithms and at each time tick, for each ray, it collects the mean, the min and max values over the edge and the quartiles. In addition, the bot stores information about the traveling time to explore an edge, the type of edge, the current health level (some paths may pass through dangerous areas like lava) and the internal weight assigned to the path in UT3. Thus, for each edge, the bot collects,

- ID of the path (string)
- Starting node id (string), label (string), position (a vector of float values)
- End node id (string), label (string), position (a vector of float values)
- Type of the edge (string), its cost (integer), the traveling time (float), the health of the pawn at the end of the edge (float)
- Acquired mean ray distance to the first obstacle (float)
- Acquired ray min/max distance acquired at each time tick by traveling the edge (float)
- Acquired ray distance in quartiles (float)

Although the use of a bot is the only way to gather morphology information about levels in UT3, it posed several issues which we had to solve using ad hoc heuristics and hacks. For instance, some maps have overlapping path nodes placed by the designer which generated traversal loops in the graph, accordingly, our bot had to include a specific heuristic to identify these situations. In addition, there are nearby nodes that are not connected in the static map loaded during initialization because of a limitation of the engine, but the bot could move between them anyway. Accordingly we had to enforce a policy to allow the exploration of such statically nonexistent but dynamically possible connections. Furthermore, some nodes were placed inside dangerous zones (voluntarily or by mistake) that could kill our bot thus enforcing a respawn and making traversal impossible; accordingly, we had to monitor the health status of the bot in each edge and apply corrective measures to ensure that the bot could reach the target node safely or, in case of void areas or traps, avoid deadly consequences.

VI. EXPERIMENTAL RESULTS

We used the custom bots we developed to collect data from twelve UT3 maps selected from the most popular ones in the competitive scene [24]; in particular, we chose four Duel maps (BioHazard, Fearless, Hypoxia, and RisingSun), five Team Deathmatch maps (CampGrounds, CBP3-Salvation, Diesel, Sanctuary, Subterrane), and three maps used both in Duel and Team Deathmatch games (Deck, Liandri, Sentinel). For each map, we collected the static topology (the nodes and the edges of the Navigation Graph, the position of all the pickups within the map) and the morphology data retrieved from the exploration with a bot equipped with raytrace sensors (Section V).

We used the collected data to build the graphs representing the static structure and morphology of the maps; then, we applied graph/network mining algorithms to compute several statistics to evaluate the maps in terms of node connectivity and flow (using the morphology information collected using the raytrace sensors); we also computed several statistics to
characterize the different types of nodes (that is, navigation nodes, ammo pickups, weapon pickups, and powerups). As an example, Figure 1 shows an example of the graph built for the Rising Sun map using the Gephi tool [25].

A. Maps Statistics

The statistics we collected for each map include (i) Close-ness Centrality (the inverse of the farness), computed as the sum of distances to all other nodes, thus the more central a node is the lower its total distance to all other nodes. (ii) Betweenness Centrality computed as the number of shortest paths from all vertexes to all others that pass through that node and can be used to identify whether a node is fundamental in the traversal of the map and, on the contrary, if the node is isolated (e.g. if a pickup is a special one it may be isolated). (iii) Clustering Coefficient, which quantifies how close the node neighbors are to be a clique; clustering coefficients can be used to identify if clusters of nodes are present in the map (e.g. pickups grouped together). Other statistics include the node (iv) degree, (v) in-degree, (vi) out-degree and their weighted counterparts; we also computed the node (vii) authority, (viii) eigenvector centrality, and (ix) its role as a hub.

B. Weapon and Ammo Distribution on Maps

At first, we analyzed the distribution of weapons and ammo pickups in Duel and Team Deathmatch maps to check whether the distribution of the weapons and ammunition is dependent on the map genre as discussed in several UT3 community blogs and to provide a quantitative evaluation of such community knowledge. Figure 2 shows the distribution of weapon pickups on maps used for Duel, Team Deathmatch and for the three maps used in both game types. As it can be noted, Bio Rifle and Rocket Launcher are more frequent in Duel maps than in Team Deathmatch. In contrast, Flak Cannon and Sniper Rifle are more frequent in Team Deathmatch maps. Redeemer is never present in maps used exclusively for Duel, this probably due to the smaller size of Duel maps. In fact, Duel maps are often smaller and Duel matches generally have a rather high pace, accordingly, the Redeemer, the Flak Cannon and the Sniper Rifle are less frequent than in Team Deathmatch where they enable more interesting gameplay. Finally, as it should be expected, the three maps used for both game modes share similarities with both the other map types.

Figure 3 shows the distribution of ammo pickups on the maps used for Duel, Team Deathmatch, and for both game modes; there is no ammo pickup for Redeemer because this weapon does not have them. As it can be noted, Team Deathmatch maps and the maps used for both game modes have a rather uniform distribution of ammo pickups whereas in Duel maps the distribution is skewed so that some weapons have much less pickups. The results in Figure 3 can be easily explained considering that Duel is a fast paced mode involving only two players in a small area so that collecting ammunition is less important; in addition, in Duel map, limited ammunition also does not let one player to take easily over by controlling the more strong weapons. Team Deathmatch maps are usually quite large and fit many players thus is very important for the gameplay to guarantee that players cannot run out of ammunition easily otherwise pace might slow down dramatically.

C. Analysis of Nodes

Figure 4 shows degree, betweenness centrality and pagerank authority for the weapon, ammo and powerup nodes. The plots suggest that weapon nodes (red line in Figure 4) have slightly higher degree and pageRank authority, suggesting that they are more connected and play an important role in the navigation route of the maps. Interestingly, ammo nodes appear to have a slightly smaller betweenness centrality (green line in Figure 4b), suggesting that they are often located in less central area of the maps with respect to the other two types of node.

D. Characterizing Maps

Finally, we applied different classification algorithms to extract patterns that might characterize Duel and Team Deathmatch maps both in terms of maps structure and in terms of weapon, ammo, and powerup pickups. In particular, we focused on Naïve Bayes, Decision Trees, Decision Rules, and Logistic Regression and used the Weka tool to perform all the analysis [26].

First, we applied the selected classification algorithms to a dataset containing all the data about the ammo pickup points labeled according to the type of maps they belonged (Duel or TDM); the goal was to extract models that could identify the map type based on the characteristic of the ammo pickups. Table III reports the accuracy obtained by all the algorithms using a ten-fold crossvalidation. As it can be noted Naïve Bayes obtain the least performance probably because of
their underlying working assumption that all the attributes are independent, which clearly does not hold in this case. All the other algorithms generate models with similar performances that the paired t-test reports to be not statistically significant at the 95% confidence level. We repeated the same procedure with weapons and powerup pickups. Table III shows similar results for ammo and powerups. In all the three cases, Logistic Regression provides the most accurate prediction and in the case of weapons pickups the difference in performance is statistically significant at 95% confidence level.

The analysis of the Logistic Regression’s coefficients show that the most relevant variables in categorizing the types of pickups (ammo, weapons, and powerups) are the node pagerank, its authority and hub values. This finding was also confirmed by an analysis we performed using feature selection algorithms which reported the same variables as being the most relevant in the dataset. Overall, these results suggest that Duel and Team Deathmatch are well-characterized by some properties of their pickups (the authority, the pagerank and hub values).

Next, we applied the same classification algorithms to extract models that could distinguish among the three types of nodes (ammo, weapons, and powerups). Our goal was to determine what characterizes the types of pickup nodes (in this case we did not include navigation nodes). In this problem, the dataset is slightly unbalanced as powerups and ammo pickups are more frequent than weapons pickups. Accordingly, the majority guess has an accuracy respectively of 42.09%, when all the maps type are considered, of 44.97%, when only Duel maps are considered, and of 45.00% when only Team Deathmatch maps are considered. All the methods achieved a rather low classification accuracy, the highest classification accuracy being reached by 57.46% accuracy when all the nodes were considered (both for the Duel and Team Deathmatch maps) and when maps were considered separately. These results suggest that it is difficult to classify the node type based on the node features. At the same time, the results on the map type classification suggest that it is possible to characterize maps based on their node features. Thus, Duel and Team Deathmatch can be characterized by the combination of weapons, ammo, and powerups nodes; however the nodes alone do not provide enough information.
TABLE V: Prediction of the node’s type (navigation node, ammo pickup, weapon pickup, or powerup pickup). An example of the resulting confusion matrix (where row is the true class, and column is the predicted class).

<table>
<thead>
<tr>
<th></th>
<th>powerup</th>
<th>navigation</th>
<th>ammo</th>
<th>weapon</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickup</td>
<td>23</td>
<td>309</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>navigation</td>
<td>16</td>
<td>2080</td>
<td>41</td>
<td>3</td>
</tr>
<tr>
<td>ammo</td>
<td>9</td>
<td>307</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>weapon</td>
<td>5</td>
<td>130</td>
<td>13</td>
<td>0</td>
</tr>
</tbody>
</table>

Finally, we applied the same four algorithms to all the map nodes, i.e., pickups (ammo, weapons, and powerups) and navigation nodes. Table IV compares the performance of the four algorithms on a dataset containing all the maps nodes, one containing only Duel nodes, and one containing only Team Deathmatch nodes. As it can be noted all the algorithms except Naïve Bayes reach a similar accuracy (around the 70%) and, according to a t-test, differences are not statistically significant at a 95% confidence level. The 70% rate roughly corresponds to the actual percentage of navigation node in the dataset and the analysis of confusion matrices (see an example in Table V) show that basically all the models tend to classify everything as navigation points. These results suggest that there might not be specific features that can distinguish among different types of nodes. We performed another set of experiments using a cost matrix that penalized classification of pickup points (ammo, weapons, and powerups) as navigation points. The generated models had the same average performance (around 70%) and they were able to distinguish among navigation nodes, ammo and powerups just slightly better; however, the models still could not classify weapon nodes anyhow since most of the weapon nodes were classified as navigation, ammo or powerups. Overall these results suggest that the ammo, weapons, powerups and navigation nodes have not highly distinguishing features that can characterize the specific type.

VII. CONCLUSIONS

We applied data mining to extract design patterns from UT3 levels. We selected twelve popular UT3 maps from the online gaming scene. Then, we developed two ad hoc bots to collect data about the topological structure of the maps and their morphology. We applied graph algorithms to extract several maps/nodes statistics and then applied four classification algorithms to extract models that (i) could characterize the map type based on the features of the pickup nodes they contain and that (ii) could identify the node type based on its topological/morphological features. The exploration of the data we collected provided a quantitative measure of the distribution of weapons in UT3 maps (Section VI-B) confirming also some of the community knowledge discussed in well-known blogs [13]. We applied four classification algorithms to extract models to characterize map types based on the features of nodes they contain. Naïve Bayes generates the least accurate models suggesting that their underlying working hypothesis (i.e., that variables are independent [26]) does not hold in this case. All the other models resulted in models with similar accuracy. The analysis of variable scoring (produced by Logistic Regression and feature selection methods) shows that pagerank, authority and hub are the variables that better characterize the types of maps. Finally, we applied the same approach to characterize the type of nodes (navigation, powerups, weapons and ammo) based on their features. The results for the characterization of nodes show that no algorithm could find a model that would accurately identify the node type and all of them tend to classify all the nodes as the majority class, that is navigation nodes. Overall our results suggest that Duel and Team Deathmatch maps are actually well characterized by the features of the pickup nodes they contain however nodes per-se are not highly characterized and thus it is difficult to identify the node role by its features. Future research direction include the integration of the data analyzed here with gameplay data extracted from simulated matches using bots. The long term goal is to characterize UT3 maps also using the dynamics element that the maps elicit.

REFERENCES

TABLE III: Prediction of the map’s type (Duel or Team Deathmatch) based on the features of a single node. Accuracy of different algorithms applied to classify ammo pickups, weapons pickups and powerup pickups. The Majority Guess always predicts the most frequent class in the dataset and is reported here as a baseline.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>classification of ammo pickups</th>
<th>classification of weapons pickups</th>
<th>classification of powerup pickups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Guess</td>
<td>62.90</td>
<td>55.03</td>
<td>52.70</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>42.32 ± 4.15</td>
<td>48.76 ± 10.84</td>
<td>51.32 ± 4.99</td>
</tr>
<tr>
<td>Decision Trees (J4.8)</td>
<td>67.89 ± 5.45</td>
<td>56.66 ± 9.37</td>
<td>63.77 ± 7.06</td>
</tr>
<tr>
<td>Classification Rules (JRIp)</td>
<td>69.75 ± 5.58</td>
<td>53.98 ± 11.34</td>
<td>59.19 ± 7.88</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>69.54 ± 6.31</td>
<td>63.92 ± 12.78</td>
<td>65.76 ± 7.39</td>
</tr>
</tbody>
</table>

TABLE IV: Prediction of the node’s type (navigation node, ammo pickup, weapon pickup, or powerup pickup). Accuracy of different algorithms applied to classify the nodes of all the maps, of Duel maps and of Team Deathmatch maps. The Majority Guess always predicts the most frequent class in the dataset and is reported here as a baseline.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>All Maps</th>
<th>Duel Maps</th>
<th>Team Deathmatch Maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Guess</td>
<td>70.44</td>
<td>69.24</td>
<td>71.25</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>46.34 ± 4.26</td>
<td>46.34 ± 4.26</td>
<td>39.98 ± 3.85</td>
</tr>
<tr>
<td>Decision Trees (J4.8)</td>
<td>70.18 ± 2.75</td>
<td>70.18 ± 2.75</td>
<td>73.63 ± 2.04</td>
</tr>
<tr>
<td>Classification Rules (JRIp)</td>
<td>71.59 ± 1.71</td>
<td>71.59 ± 1.71</td>
<td>72.71 ± 1.48</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>69.85 ± 1.07</td>
<td>69.85 ± 1.07</td>
<td>71.61 ± 0.93</td>
</tr>
</tbody>
</table>