Received 17 June 2023, accepted 17 August 2023, date of publication 13 September 2023, date of current version 21 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3314754

APPLIED RESEARCH

Automatic Creation of Tiled Maps in Two-Dimensional Top-Down Digital Role Playing Games

KRZYSZTOF KACZMARZYK AND DARIUSZ FREJLICHOWSKI
Faculty of Computer Science and Information Systems, West Pomeranian University of Technology, Szczecin, 70-310 Szczecin, Poland

Corresponding author: Dariusz Frejlichowski (dfrejlichowski@zut.edu.pl)

This work was supported in part by the Zachodniopomorski Uniwersystet Technologiczny (ZUT) Highfliers School (Szkoła Orłów ZUT) Project coordinated by Assoc. Prof. Piotr Sulikowski within the framework of the Program of the Minister of Education and Science under Grant MNiSW/2019/391/DIR/KH and Grant POWR.03.01.00-00-P015/18, and in part by the European Social Fund under Grant PLN 2,634,975.00.

ABSTRACT Game level maps are fundamental for modern Digital Role Playing Computer Games. Creating such maps by hand is a time-consuming process. We propose an approach that can be applied to create such maps automatically. The approach is explained with clearly defined steps. The map creation process divides a map into four distinct layers. First, we explain the idea of converting input in the form of a rough sketch of a layer into the suitable form — a colour grid. Next, the issue of map creation from the initial stage of the colour grid is presented. We then introduce methods used to correct the inputs in order to create layers without defects. We apply specific methods to each layer, explaining the method in detail in subsequent sections. The layers are then merged to create the final map. As the next step, we discuss the improvements applied to the process that we developed. The complete approach is then tested in practice with a questionnaire, which — following a careful analysis — has showed that there are no significant differences between handcrafted maps and the maps created using our approach.

INDEX TERMS Automatic map generation, digital role playing games, tiled maps.

I. INTRODUCTION

Traditionally, Role Playing Games (RPGs) are games in which a player or a group of players assume the roles of characters and play (control) them in a fictional setting [7]. Digital Role Playing Games (DRPGs) are based on the same concept, but played using modern technology. On Steam [11], one of the biggest digital markets, Digital Role Playing Games are the fifth best-selling category, with one such game bought by every tenth of Steam's 150 million users [13]. The DRPGs are played on a computer, where characters as well as the setting are usually represented graphically. In most cases, the characters are able to traverse the fictional setting by moving on a randomly generated map or on one created by a game developer. Those maps can be further categorized, for example, based on their dimensionality. In this paper we focus on two-dimensional, top-down maps.

Our approach uses top-down maps which have a slightly oblique view and is based on tiled images, common in maps created using the RPG-Maker software [3]. The proposed solution is developed using the game Margonem [5], which has the same type of maps. Tiled images which compose those maps are called ''tiles'', and the group of tiles that a map is composed of is a ''tileset''. A tile is made up of an image and a set of constraints that dictate what tiles it connects to. It usually entails a specific type of background that is either on the side of a tile or at its corner. In the case of specifying corner backgrounds, both touching corners of adjacent tiles must have a matching background. The idea and related problems are described in more detail in Section III.

Using tiles to create maps greatly simplifies the process and makes it possible to create maps of the same quality

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

much faster, because after creating a tileset a developer needs only to place the tiles in the correct manner to create a map. Even then, creating a map may be time-consuming and is prone to error, as each tile needs to be placed individually and even one incorrectly placed tile can create a visual mismatch, ruining the seamless effect. That is why several solutions have been developed to aid developers in the map creation process, e.g. Tiled [12], Margonem Map Editor (in Polish ''Edytor Mapek Margonem'') [6] and MapForge [4]. What these solutions have in common is that they all still work on a tile-by-tile basis. They contain tools to expedite the map creation process; nevertheless, each tile needs to be placed individually by using brush-like tools or filled in randomly in a selected area.

We propose an algorithm that can automatically create complete tiled maps based on a rough sketch of a map along with a set of tiles. Generation is deterministic and based on developed algorithms, rather than created using machine learning. The deterministic method was chosen because of three key factors. The first is that with a deterministic method, it is easier to understand what each step in the method does, and it is easier to modify it if the need arises. Secondly, the tiled maps are, in our opinion, more suited to deterministic approaches since one can easily swap out the input tileset to create maps with vastly different looks without changing the map layout (in order to, for example, create a winter version of the same map). Lastly, the collection of appropriate training data required to train a machine learning solution would have been extremely difficult, especially for the problem that our method wants to solve. It comes from the fact that there are no benchmark datasets containing the required data, and the creation of such a large database would take months or more, and even in that case, there is no certainty that this data would be effective in the process of machine learning.

In order to evaluate the results, we investigate the generated maps against to maps created by hand. The results show that differences are either non-existent or insignificant and most people cannot pinpoint the origin of a given map, even when the person is familiar maps created using the selected tileset.

II. RELATED WORKS

In this section, we describe previous works related to the process of creating tiled maps, as well as the works that inspired the approach proposed in this article. Additionally, we provide an explanation of how our solution differs from them and where it fits in the map creation process. The problem described in the paper is technical in nature, and is therefore rarely discussed in the scientific works, most of which are focused on different aspects of maps; for example, [9] presents the creation of stylized maps, but it focuses on a different type of maps, not utilizing tilesets. However, map creation is a widespread problem in the game industry and led to the development of tools to make the process easier and faster. Some of them might not be available publicly, but others are provided to the public or developed as open software from the ground up. They will be briefly described in the following subsection.

Additionally, our previous works on the problem were described in [2]. However, whereas [2] focuses in particular on one aspect of the map creation process — specifically, the ground layer — this paper demonstrates the whole process of map creation, describing every step in detail, and presents more extensive tests for the obtained results.

A. SOFTWARE USED FOR MAP CREATION

The Margonem Map Editor [6] is a solution internally developed for the game Margonem as a part of its world editor, later released to the public. The software makes it possible to create top-down maps for the game using a predefined set of tiles. The map Editor offers eight layers to place tiles one by one, in rectangle batches, or by filling a space with one type of tile. It also provides a functionality that allows users to automatically connect selected tiles placed using a provided tool. Despite the tools it offers, the Margonem Map Editor is considered slow and not very practical for map creation. The user cannot quickly create a map according to their vision as each tile (or a group of tiles) needs to be placed individually. Complex map formations still require a significant time investment.

A similar issue can be seen in other publicly available software such as MapForge [4]. MapForge is map-making software, but it focuses on tactical battle-maps created for traditional tabletop RPGs (played at a table, either in person or virtually). It allows its users to create top-down or isometric maps and is mainly for creating maps where tiles can be placed disregarding the grid. Apart from the available tiles, the software can paint shapes filled with chosen textures, disregarding the grid entirely. Another advantage that distinguishes MapForge is that the user can simultaneously create day and night versions of a map. However, making complex maps requires putting the tiles together piece by piece, which is still a very time-consuming process.

The same problem can be found in Tiled [12], one of the popular solutions in this field. It focuses on two-dimensional tiled maps utilizing a side, top-down and isometric perspective. Its basic functionalities include placing tiles one by one, in groups or by filling a large area with one type of tile, all applied to an unlimited amount of layers. It also allows users to place previously prepared terrain automatically, choosing the correct tiles based on selected rules. Additionally, it supports many miscellaneous features such as tile animations or collisions. The software still lacks the ability to automatically create a full map which is not random and can be easily shaped.

B. A 'MISSING LINK' IN THE MAP DEVELOPMENT PROCESS

All the methods described above are useful for providing the game developers with tools to allow them to shape the game map however they want. However, their attention to details creates another problem, which we observed. Map creation often starts with a general idea of a map that the mapmaker later creates. Such an idea is usually not very detailed, consisting of key remarks about the map and objects placed on it. Large parts of the map are most often left to the mapmaker's discretion. While careful map detailing made by humans is a process that our solution cannot replicate, its strength lies in map prototyping and the creation of the map baseline. Our method was developed to enhance this key part of the map creation process by allowing the maps to be fully generated with minimal effort from a mapmaker. Using the previously described software, fast creation of the full, complex map is not possible. Creating multiple map prototypes becomes tiresome as each tile or group of tiles needs to be placed individually. With our solution, not only do we expedite the process of prototyping, but we also do not lose attention to the details of handmade maps, as the generated map can be used as a baseline for custom edits by the map creators if they decide that the generated map is missing something (for example, a specific object setup that is impossible to replicate with a simple sketch).

C. AUTOMATIC GENERATION OF A GRAPHIC BASED ON A SKETCH

Creating complex graphics based on a rough sketch is not a new idea. The technique has gained popularity with the rise of machine learning and image to image processing, as a way to influence the created image with minimal human work required. An example of such usage is shown in [8], where an input of a sketch created with simple colours marked as types of texture results in a detailed image of a terrain shaped like the sketch. The idea is very intuitive, easily applicable, straightforward for the graphic designer and constitutes a basis for our approach.

III. TILES

A single tile is composed of two elements: an image representing it and a set of restrictions limiting what other tiles it connects to. While there are several ways of introducing such restrictions, our approach uses a simple system of four corners. A particular terrain type is assigned to each corner of a tile, meaning that a full set of tiles for connecting two terrain types will contain sixteen tiles (one tile for each permutation). In order to correctly connect two tiles together, two corners that are adjoined need to have the same terrain type. Fig. 1 shows how the mechanism can be applied.

A. DEPTH PERCEPTION IN TILED MAPS USING LAYERS

Top-down oblique-view maps achieve their perceived depth by layering tiles and images. Objects that are higher on the Y axis of the map (two-dimensional height) are also lower on the Z axis (three-dimensional height). Thus, in order to create a map one should put tiles and objects from the top to the bottom of the map, so that objects at the bottom can cover

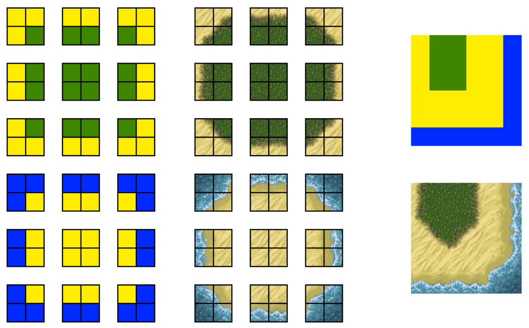

FIGURE 1. An example of of how the tile mechanism can be applied. On the left, tiles with terrain signified by colour are presented. In the middle, graphics for the tiles are presented. On the right, an example of a tile-to-image substitution is provided, with coloured terrain map at the top (without guidelines) and the final version of the map at the bottom.

objects and tiles at the top on the map. While creating maps by hand map designers usually achieve that by having multiple layers on which they place tiles. Layers are also used as a separation tool separating some parts of the map from others, if necessary. Our approach also uses layers but in a slightly different manner. There is no problem with placing tiles from the top to the bottom of a map, so we have distinguished four layers that an image can comprise: ground, cliffs, paths and objects. Such categorization makes it possible to create complex maps with only four different inputs. The ground layer is the only one which is always required for a map as it contains the terrain which is the basis of the map and other layers. The cliffs layer adds more verticality to the map. It contains multi-levelled cliffs that can be put on the final map. The paths layer adds details to created maps. It makes it possible to put paths and patches of terrain that are not necessarily ground terrain, or are impacted by cliffs. The last layer contains various objects that can be placed on a map to add visual flair and avoid it looking empty.

B. PROBLEMS WITH TILED MAPS

Creating maps from tiles is really compelling for game designers since the style can be easily preserved and one does not necessarily need to be a graphic designer to make a map. However, there are some limitations. Firstly, maps made using the method look blocky. Repeated tile patterns can be easily seen in maps created from tiles. It is neither good nor bad and can be viewed as an art style choice. From the video game developer's perspective, the tile system itself and the number of tiles it requires can be seen as problematic. Sixteen different tiles are needed for just two types of terrain; adding only one more type of terrain puts the number at forty-eight (with tiles that can only have two terrain types at most). Going further, it is clear that the number of required tiles grows exponentially: for example four types of terrain require ninety-four tiles.

C. DEALING WITH TERRAIN PERMUTATION PROBLEM

In order to solve the problem of the exponentially growing number of tiles, it is possible to simply not create some combinations to avoid it. It leads to a different problem though: some combinations will not be available since tiles for them will not be created. However, it is still a much more manageable method. When manually creating maps, the designer can simply avoid those combinations. Things get more complicated when we want to create maps automatically from a sketch.

The person who did the sketch might not necessarily be aware of terrain combinations that have not yet been created or might create a transitional terrain part that is too small, which can get lost later in the translation step. In such cases, rather than showing a blank tile, we assigned a height index to each type of terrain. Then, we introduced a new restriction: types of terrain can only have tiles connecting to other types of terrains with a preceding or succeeding height index. As a result, the number of tiles will grow linearly instead of exponentially. For each newly added terrain type, one only needs to create either sixteen or thirty-two tiles (the latter is needed if the terrain is inserted between two existing terrains).

If our map creating algorithm detects terrains that do not contain preceding or succeeding height indices, it creates a minimal transition region between them. Starting from the largest height index difference, it changes the tiles with incompatible terrain types to ones with a terrain type whose height index is the average of the incompatible ones. The width of the transition region is proportional to the height index difference.

IV. PROPOSED APPROACH

We propose an approach for creating tiled maps which involves generating four distinct layers defined earlier in Section III-A and then combining them to create the final map.

A. REQUIRED INPUTS

The first three layers of the map require a similar input: one sketch per layer. Each sketch must be of equal size and must be composed of colours predefined for a particular layer. Each colour represents a different terrain type and is applied to help users create a sketch. Technically, it is not required to assign the colour blue to water or the colour yellow to sand; however, choosing a colour corresponding to a given terrain type (i.e. a colour typically associated with said terrain in real life) greatly facilitates creating and understanding the sketches.

The last layer requires a list of objects that could be placed on the map, along with settings for each individual object specifying how it should be placed. The settings include: an image or images of the object, its dimensions, terrain type it should be painted on and its density. The algorithm will then place objects on the map in accordance with their settings.

B. SKETCH CONVERSION

The first step of the approach is converting a sketch provided by the user into a form suitable for further work. It is done on the first three layers of the map. To convert a sketch into a workable terrain grid, one needs to segment the image into rectangles that are half the width and half the height of one single tile. Then, the most dominant colour of pixels in each rectangle results in the rectangle's terrain type. Combining all these rectangles produces a terrain grid twice the width and twice the height of the final tile map. It is a desirable outcome, because four terrain points on the grid will later create one tile. Fig. 2 illustrates an example of the sketch conversion.

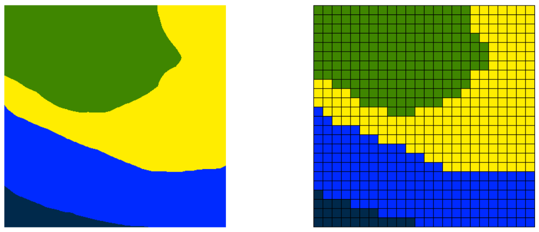
FIGURE 2. An example of a sketch-to-grid conversion. The input sketch is on the left. A grid made by means of sketch conversion is shown on the right.

C. INITIAL STAGE OF A TERRAIN GRID

Having converted the sketch into a terrain grid, we reach the initial stage. At this point, creating a layer from the terrain grid will result in sharp, mismatched edges (see Fig. 3) because tiles are structured in such a way that a smooth transition is possible only when it happens inside the tile. When a terrain grid is in the initial stage, the constructed tile map does not follow the rules established in Section III, as the terrain types on the corners of the tiles are mismatched.

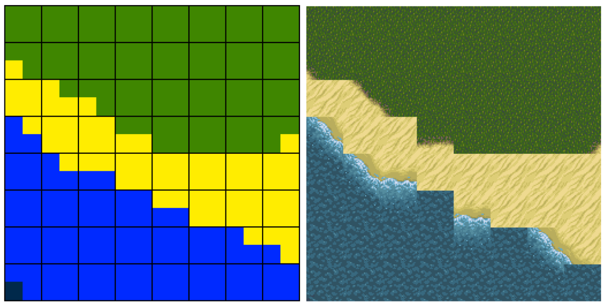
FIGURE 3. Visualized problem of the initial stage with the map tiles shown on the left and the map created directly from these tiles on the right.

D. TWO RULES ALGORITHM

In order to repair the terrain grid, we propose a Two Rules Algorithm. It is based on the basic principles formulated in Section III, but it quantifies them in a more meaningful and usable manner. The Two Rules Algorithm comprises the Double Edges Rule and the Middle Rule, which were first presented in [2], but are once again explained in this article for the sake of clarity.

Algorithm 1 The Two Rules Algorithm

```
Data: $T$: Pixel grid to be fixed
Result: Fixed pixel grid
Expand $T$ by 2 pixels in every direction by duplicating the edges;
for $i$ in$[0, 2, 4, \dots, length(T) - 3]$ do
    for $j$ in$[0, 2, 4, \dots, length(T[i]) - 3]$ do
        DoubleEdgesRule($T[i, \dots, i+3,\ j, \dots, j+3]$);
        MiddleRule($T[i, \dots, i+3,\ j, \dots, j+3]$);
    end
end
Shrink $T$ by 2 pixels for every direction by removing an edge on each side;
```

1) DOUBLE EDGES RULE

If the pixels on the edge of a tile are of the same colour pairwise, and the pixels of the adjacent edge are also of the same colour pairwise, these four pixels must be of the same colour. To this end, the Double Edges Rule must be applied. Since a smooth transition from one colour to another only occurs when the colour change is present in the middle of a tile and tile images are placed independently of one another, the rule prevents sharp edges from appearing in the final image. Fig. 4 illustrates the Double Edges Rules.

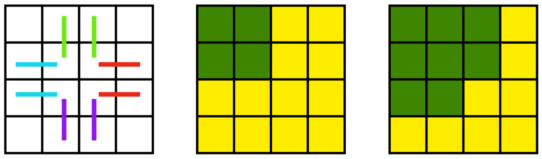

FIGURE 4. Graphic representation of the Double Edges Rule. Pixels handled by the rule are highlighted on the left. An example input is shown in the middle. The output produced by the algorithm based on the given input is on the right.

2) MIDDLE RULE

The pixels in the middle of a given 2×2 tile set must be of the same colour. The Middle Rule must be applied to this end. If there are three pixels of the same colour, the fourth pixel is changed to that colour. Otherwise, all four middle pixels change their colour to the one that is the lowest amongst them, i.e. the one that would be overwritten by all the other ones. Fig. 5 presents how the algorithm works.

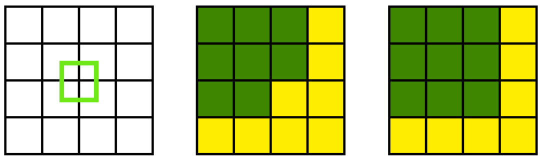

FIGURE 5. Graphic representation of the Middle Rule. Pixels handled by the rule are highlighted on the left. An example input is shown in the middle. The output produced by the algorithm based on the given input is on the right.

E. CLIFFS LAYER

In order to properly create multilevel cliffs, it is necessary to record their height, as it marks the level at which cliffs can end. The transition from level zero (ground) to the second level should take two single cliff heights, whereas the transition from level zero to the fourth level should take four single cliff heights. However, the transition from level two to level four should take two cliff heights. Since input sketches can mix and match different cliff heights, we need to take that into account. To that end, we start from the bottom of the map and work our way up to the top, counting and recording at which level a given tile should be. That way, we can keep the cliff height consistent. Fig. 6 demonstrates a map with one cliff that has two different transitions. The red rulers count how many tiles of height cliffs have. The left side of the cliffs goes from level zero to level two and then from level two to level four, while the right side goes from level zero to level four. We exclude the top tile of a cliff from calculations, since a cliff always requires at least two tiles to go up one level.

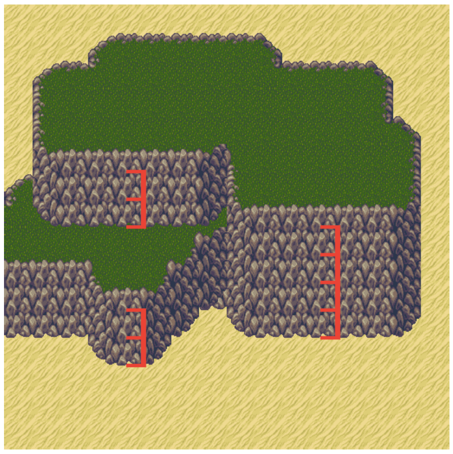

FIGURE 6. An example of a cliff with two different transitions. On the left side, a transition from level zero to level two and then from level two to level four. On the right side, a transition from level zero to level four. The red rulers count the height of the cliff.

In order to additionally make the final product closer to what the user might expect, heights that are too short (for example, a fourth-level cliff that only spans one tile in height) are extended so that they cover at least the minimum height they should take. After that, we place the calculated tiles from the top to the bottom of the map, from the shortest to the tallest.

F. PATHS LAYER

Ground and cliffs have to already be created in order to make a paths layer. Parsing input is identical to parsing the ground layer. However, the process of transforming a terrain grid into tile images is slightly different. Instead of simply choosing a tile that is assigned to the combination of four terrain corners, the tiles from the ground and cliffs layers have to be taken into account. The combination of corner terrains constitutes the tile combination (for example top left end of path when the corner terrain is on the bottom right), whereas the underlying background establishes the tile type (for example a ladder on a cliff or a muddy path on the grass). After calculating both the combination and the file type, the correct tile is chosen and placed. When creating a tileset, it is worth noting that such tiles do not need to strictly follow the combinations like the ground tiles and can be used with less restrictions, for example, by having a full ladder tile regardless of the underlying combination.

G. ADDING OBJECTS

The initial three layers need to be created first in order to add objects to the final map. It is then possible to iterate trough a list of objects and put them on the map when the conditions are met. The conditions can include being able to be added to the map only on certain top tile, only on certain ground tile, or with a set density. When an object has been put down, the tiles it occupies are marked as unusable for other objects. After all objects have their designated places, the algorithm adds their image (that can be randomly selected from a predefined list) from the top to the bottom of the map. Even if an object's size is bigger than one tile, it is added to the map in full on the row where its base is. This is how the final map is made to appear three-dimensional, creating the illusion that some objects are closer to the viewer than others.

H. COMBINING LAYERS TOGETHER

After all the layers are ready, the final image can be created by simply combining them, with the new layers overwriting the pixels of the previous ones. Since the ground layer does not contain any transparency and other layers will almost certainly contain some transparency, the result is sophisticated maps with layers building atop the previous ones. Fig. 7 illustrates the process of combining the layers to create a map.

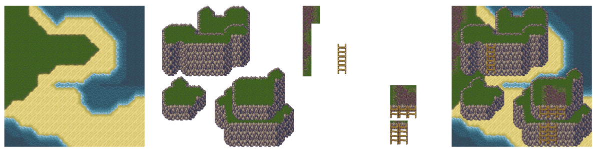
FIGURE 7. An example of combining layers together. From left to right: the ground layer, the cliff layer, the paths layer, and the map created by combining these three layers.

V. APPLIED IMPROVEMENTS

While the approach proposed in Section IV produces satisfactory results, we can further enhance the images by applying some techniques that do not directly modify the core approach but insert an additional step into the process of generating particular layers or automatically add a specific configuration to them.

A. WAVES

The tileset we use contains a water-to-sand transition that has waves expanding to the second water tile. Creating a map without adding them would result in slightly sharp edges in the water. In order to prevent this, we developed an additional step to be executed after the ground map is generated. It iterates through the whole map and verifies if the currently analysed tile is a water-transition tile and if there is an empty water space on its side. If both conditions are fulfilled, a half-transparent water edge is inserted onto the water. The fact that it is half-transparent is important here as it makes it possible to combine several edges on one tile. Fig. 8 features an example of a map before and after the technique is applied.

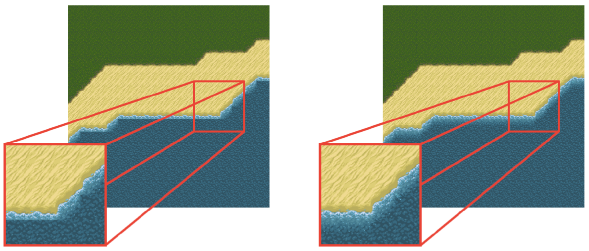
FIGURE 8. An example of applying wave improvement. A map before the proposed technique is applied is shown on the left and the same map after waves have been added is on the right. Note the unnatural, sharp edge in the place where the sand meets the water on the left.

B. CLIFFS IN WATER

The process of creating cliffs produces cliff bases which are unattractive when placed on deep water. In order to solve this issue, we add objects of cliffs going into the water to the object list while creating the last layer. When the ground terrain is deep water and the top tile is a cliff base, a proper image is inserted resulting in a smooth transition. Fig. 9 shows a map before and after the watered cliff base objects are applied.

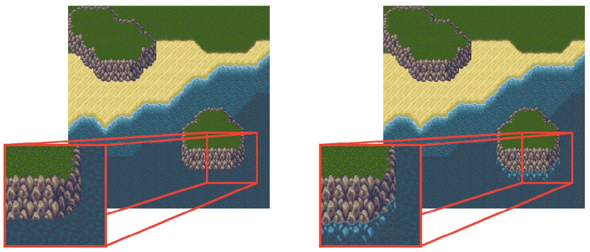
FIGURE 9. An example of applying the improvement of cliffs in water. A map before adding cliff transitions is shown on the left. On the right, the same map after cliff transitions have been added. Note the smooth cliff transition to the water on the right.

VI. EXPERIMENT

We have evaluated the similarity between handcrafted maps and maps created using our approach. To this end, we developed an experiment utilizing a questionnaire. Interviewees were shown maps of different origins and were tasked determining their respective origin. The experiment we carried out does not compare performance metrics with

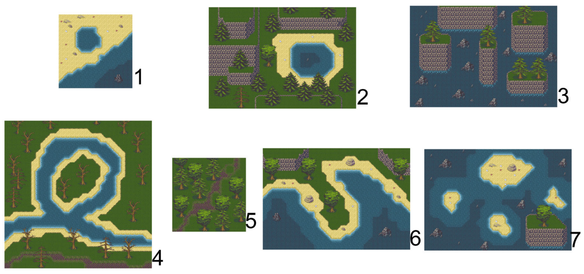

FIGURE 10. Seven maps created by hand, using only GIMP, labelled 1 to 7.

similar methods because methods that achieve similar final results automatically given the equivalent input data do not exist yet. The structure of the experiment comes from the lack of baseline methods that could be used for comparison.

A. DATASET

The dataset used in the experiment contains seven maps created utilizing the approach proposed in this paper as well as a control group of seven maps created by hand with the image editor GIMP [1]. All of the maps were created using the same tileset. Fig. 10 features the maps created by hand whereas the maps created using our approach are presented on Fig. 11. It is noteworthy that it took significantly more time to create maps by hand than using the proposes approach: four to five hours and thirty minutes respectively.

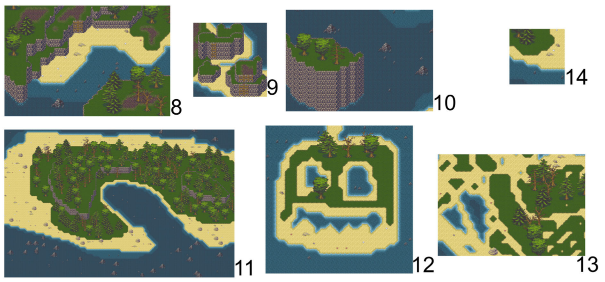

FIGURE 11. Seven maps created using the proposed approach, labelled 8 to 14.

Each of the fourteen prepared maps is based on a different sketch. The direct comparison of the same map made by a human and the equivalent made by the proposed approach was intentional. That is because, due to the use of tiles, both maps will look almost identical if the human mapmaker accurately depicts the sketch. The only difference between them would be in object placement. A human would most certainly place objects in more specific patterns than the simple algorithm used in our approach. If we would then use those almost similar maps in the experiment, the results would be inaccurate since the participants would directly compare two maps and decide which one is human-made based on the randomness of the object placement and not by taking the map as a whole at face value. That is why we choose maps based on different sketches while trying to keep them similar in form and complexity.

B. PARTICIPANTS

The participants were informed that the results of the questionnaire will be used as a part of the research and agreed to participate in it based on this assumption. The experiment participants divided themselves into two groups by answering a question about their previous contact with the Margonem game maps. This distinction was made in order to test whether knowing the tileset helps to correctly determine the origin of a map.

C. PROCEDURE

The order the maps were shown to the interviewees was established using random.org [10]. We used its list randomizer tool to arrange the maps in a truly random order. Each map was accompanied by a scale which the interviewees used to determine their certainty about the origin of the map. The scale was as follows:

1) "I am certain this map was created by the algorithm"
2) "I think this map was created by the algorithm"
3) "I cannot determine the map's origin"
4) "I think this map was created by hand"
5) "I am certain this map was created by hand"

VII. RESULTS

The questionnaires were completed by 31 interviewees. 25 of them stated that they had previously seen the Margonem game maps, while six (6) of them stated they did not. We used two metrics that were calculated from the answers, both related to the similarity between the handcrafted maps and the maps created using our approach. The first metric is the percentage of correct answers (details are given in the next subsection) and the second one is the percentage of answers which a particular interviewee was certain about.

A. ACCURACY OF THE APPROACH

In our experiment we assumed the accuracy of the approach as a degree to which interviewees chose the correct answer. It was considered a correct answer is an answer when the interviewee properly identified the origin of the map and selected "I think", or "I am certain" for that origin. The accuracy is presented on Fig. 12. The average accuracy of the approach was 48% for all of the interviewees, with 52% for the interviewees who had seen the game maps previously, and 35% for the interviewees who had not.

B. CERTAINTY OF THE ANSWERS

In order to determine the certainty of the answers based on the results, we first calculated the number of the "I am certain" answers and then calculated how many of them were correct. Fig. 13 presents the results. The average certainty was at 38%, i.e. the interviewees answered "I am certain" to 38% of the questions. 59% of those answers were correct, which means 22% of the answers were both certain and correct. If we divide the interviewees into two groups, one that had seen the game maps and a second one that had not, the first group's certainty

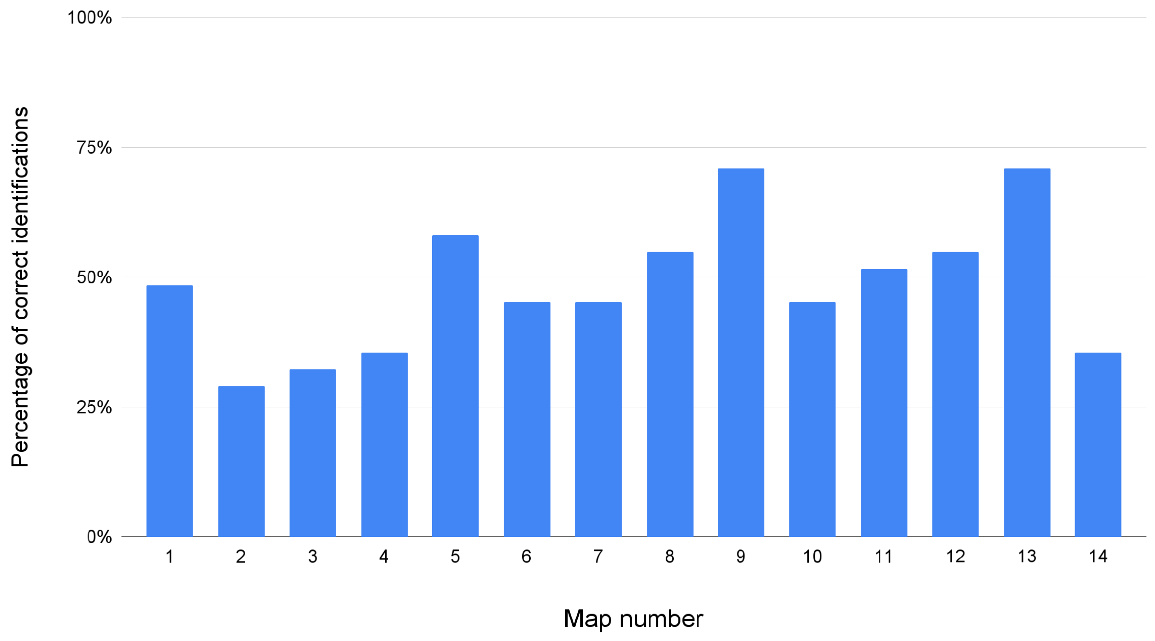

FIGURE 12. The percentage of users who correctly identified the origin of the map. Maps from 1 to 7 were handcrafted, whereas maps from 8 to 14 were created using our approach.

was at 39% with 63% correct answers, which means that 25% of the answers were both certain and correct. The second group's certainty was at 36% with 28% of their answers correct, which means that 10% of the answers were both certain and correct.

C. DISCUSSION

The extracted average results are presented in Table 1. The experiment results show a similarity between the handcrafted maps and the maps created using our approach. The accuracy of answers is close to 50%, meaning that the interviewees were as likely to correctly identify a map's origin as they were to identify it incorrectly or to not be able to identify it at all. Knowing Margonem helped, since the interviewees who were not familiar with the game had a significantly lower chance of identifying the map's origin correctly.

The interviewees who had seen maps from the game before showed a similar level of certainty in their answers as the ones who had never been acquainted with them. However, the former were correct in their assumptions twice as often. Nonetheless, only 25% of the answers given by the group familiar with the maps from Margonem were both certain and correct, which is low number, statistically. Therefore, we conclude that it is within reason to state that the approach proposed in this paper is successful in creating maps that do not differ from handcrafted maps to any significant degree.

When analysing the obtained results, we noted that map 2 is an outlier in the percentage of correct identifications and has the highest incorrect "I am certain" response ratio. While looking at the map in question, it indeed can look blocky and geometric — attributes usually associated with

TABLE 1. Average combined results of the questionnaires.

Type of result	Previous contact with the Margonem game maps		Total
	Yes	No	
Accuracy	52%	35%	48%
Certainty	39%	36%	38%
Certain and correct	25%	10%	22%

automatically generated maps, which might explain the incorrect identification. The map 5 was another outlier, which had a high correct identifications rate and was the only map that had no incorrect "I am certain" responses. This map contains a really thin path that looks like it might be hard to replicate with an algorithm, which would explain no "I am certain this map was created by the algorithm" responses.

VIII. TIME ASPECT

One of the key aspects of the approach presented in this paper that was not tested by the questionnaire is how much time can be saved when creating maps using the approach as compared to creating maps manually. This factor is hard to measure, since it is significantly more time-consuming to create a more detailed map by hand than a simpler one. In addition, the time spent depends on what software is used, as map-making software can provide various means to expedite map creation. However, it is noteworthy that it took substantially less time to make the maps for the experiment employing the proposed approach than to make them by hand in an image editor. It took four minutes to create the most complex map for the questionnaire using our approach (see Fig. 11, map 8).

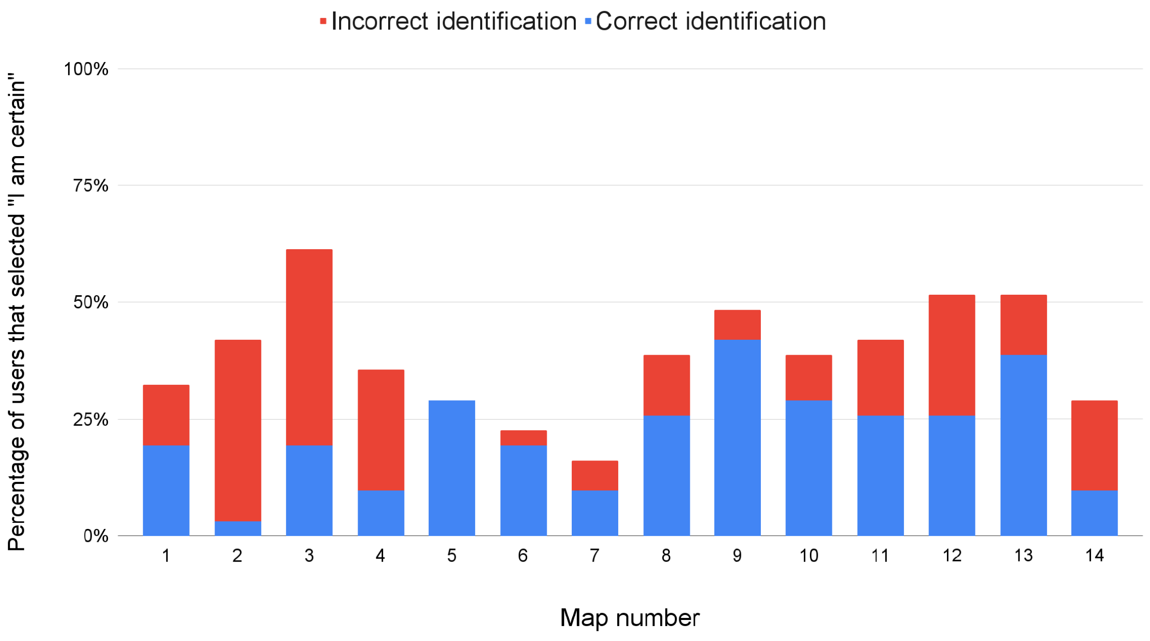

FIGURE 13. The percentage of users who responded that they were sure of the map's origin. Maps from 1 to 7 were handcrafted, whereas maps from 8 to 14 were created using our approach.

On the other hand, the most complex hand-made map (see Fig. 10, map 6) took more than thirty minutes to complete. For the least complex maps, the least complex hand-made map (see Fig. 10, map 1) took five minutes to create, while the least complex map created using the presented approach (see Fig. 11, map 14) took about thirty seconds.

The other interesting observation made when creating the maps for the questionnaire was that in the process of creating maps by hand, the layout of the map was often visualized with a sketch before the tiles were actually put on the map. A similar sketch could be used to create a map using the presented approach, saving the time that would otherwise be spent placing the tiles by hand. What is more, our approach makes it possible to quickly iterate on ideas as the actual time needed to generate a map is negligible and consistently below five seconds for reasonably sized maps, like the ones used for the questionnaire.

IX. CONCLUSION

This paper presented an approach for creating ground level maps in two-dimensional, top-down digital role playing games. Maps are divided into four layers, each with its own input. Each layer is treated as separate before all of them are merged to create the final map. The first three layers use rough colour sketches provided by the user as input. They are then converted into terrain grids which are modified using the developed algorithms. The last layer uses a simple list of objects and conditions for adding them. All of the layers are merged to create the final map. We determined that the resulting maps do not differ from handcrafted maps to any significant degree.

The presented novel approach is based on a sketch given by a user and works in real time, which allows game developers to save time by expediting the process of map creation in a way that was not possible before. It is especially useful for map prototyping and quickly creating a base map to work on, features that we found lacking in the existing solutions. There are no similar methods for tiled map creation that have been developed so far.

In the future, more extensive tests of the proposed method using objective evaluations to measure its robustness, speed, and performance of generated maps are assumed. The tests will include cooperation with both players and map developers in order to confirm the effectiveness of the method at different stages of the typical RPG map lifecycle (its creation and then usage by players). Additionally, we are planning to develop a solution that would make it possible to create a non-linear colour height map. Currently, terrain colours can have borders with no more than two other colours. Hypothetically, an ideal solution would remove this constraint without having to deal with the terrain permutation problem, explained earlier in Section III-C.

REFERENCES

[1] (Dec. 30, 2022). *GIMP—GNU Image Manipulation Program*. [Online]. Available: https://www.gimp.org/

[2] K. Kaczmarzyk and D. Frejlichowski, "An algorithm for automatic creation of ground level maps in two-dimensional top-down digital role playing games," in *Computer Vision and Graphics*, L. J. Chmielewski and A. Orłowski, Eds. Cham, Switzerland: Springer, 2023, pp. 213–222.

[3] (Feb. 14, 2023). *Make Your Own Game With RPG Maker*. [Online]. Available: https://www.rpgmakerweb.com/

[4] (Dec. 12, 2022). *MapForge—Battlemap Creation Software for Windows and MAC*. [Online]. Available: https://www.mapforge-software.com/

[5] (Dec. 30, 2022). *Margonem MMORPG*. [Online]. Available: https://www.margonem.pl/
[6] (Dec. 30, 2022). *Nasze Mapki 2—Forum—Margonem MMORPG*. [Online]. Available: https://forum.margonem.pl/?task=forum&show=posts&id=493129
[7] (Feb. 14, 2023). *Oxford Advanced Learner's Dictionary*. [Online]. Available: https://www.oxfordlearnersdictionaries.com/definition/english/role-playing-game?q=role+playing+games
[8] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," 2019, *arXiv:1903.07291*.
[9] M. Prachyabrued, T. E. Roden, and R. G. Benton, "Procedural generation of stylized 2D maps," in *Proc. Int. Conf. Adv. Comput. Entertainment Technol.*, New York, NY, USA, 2007, pp. 147–150, doi: 10.1145/1255047.1255077.
[10] (Dec. 30, 2022). *Random. org—List Randomizer*. [Online]. Available: https://www.random.org/lists/
[11] (Dec. 30, 2022). *Steam Store*. [Online]. Available: https://store.steampowered.com/
[12] (Dec. 30, 2022). *Tiled | Flexible Level Editor*. [Online]. Available: https://www.mapeditor.org/
[13] E. J. Toy, J. V. H. H. Kummaragunta, and J. S. Yoo, "Large-scale cross-country analysis of steam popularity," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, 2018, pp. 1054–1058, doi: 10.1109/CSCI46756.2018.00205.

KRZYSZTOF KACZMARZYK received the B.Eng. degree in computer science from the West Pomeranian University of Technology, Szczecin, in 2023. His current research interests include tiled maps and image generation.

DARIUSZ FREJLICHOWSKI received the M.Sc. degree in engineering and the first Ph.D. degree in computer science from the Szczecin University of Technology, Poland, in 2001 and 2005, respectively, and the second Ph.D. degree in computer science from the West Pomeranian University of Technology, Szczecin, Poland, in 2012. He was a Ph.D. Student (2001–2005), a Research Assistant (2005–2006), and an Assistant Professor (2006–2008) with the Faculty of Computer Science and Information Technology, Szczecin University of Technology. From 2009 to 2019, he was an Assistant Professor with the Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, where he has been an Associate Professor, since 2019. From 2008 to 2012, he was the Head of the Department of Internet Technology. From 2012 to 2016, he was the Deputy Dean for Science and Development. From 2016 to 2019, he was the Head of the Department of Medical Informatics. He is the author of three books and more than 120 articles. His research interests include image analysis and processing and recognition in many topics and applications, e.g., shape description and recognition, fusion of various features representing an object of interest, content based image retrieval, computer games, applications of image extraction and recognition methods in erythrocyte recognition, trademark recognition and retrieval, airplane silhouettes recognition, ear biometrics, binary images compression, 3-D shapes, localization of vehicles, license plates recognition, and color and shape fusion for CBIR. Since 2016, he has been a member of the Governing Board of the Association for Image Processing (Polish Member Society of the IAPR—International Association for Pattern Recognition). In 2018, he received the Bronze Cross of Merit for his achievements in scientific works. He was the Editor-in-Chief of the *Journal of Theoretical and Applied Computer Science* (Polish Academy of Sciences).