International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Terrain Generation from User Text using Cellular Automata

Anagha Zachariah C, Pankaj Kumar G, Umesh D R, Arun Kumar M N

Abstract: ***Last decade has been witnessed for the dramatic advancement in computer games. Latest studies shows that procedurally generated content makes the players remain engaged for a long time. Incorporating the user's preferences and creativity in the games is still a challenging task. In this paper, we focus on the diversity of terrain generation in terms of its components. Personalization of user experience via effective modeling, combined with real-time adjustment of the content helps for meaningful content generation. Our engine accepts input from the user in Terrain development language (TDL).The user has to specify the type and features of the terrain. The engine extracts features through Terrain Development Algorithm (TDA) and generates individual components according to the information. The end result is to provide a user skill-matched terrain, which can be rendered for various games.***

Keywords: Cellular Automata, Terrain Language, Terrain Development Algorithm.

I. INTRODUCTION

Games are inevitable for the modern society. Recent surveys show that, the total hours spent on games by the global population is 3 billion hours in a week. The static and repetitive contents create a rapid decrease in ratings of the game. So it's appealing to create something which maintains the diversity in gaming environment [1]. Our proposed system is an engine which accepts the description from the user and models accordingly. In our proposed model we are targeting the engagement of the user in the commercial game generation through skilled-matched contents [2]. In order to personalize, we propose a Terrain Description Language (TDL) and Terrain Generation Algorithm (TDA). User has to specify his needs and preferences in TDL. Each sample point of the track is fed to TDA. TDA unleashes the relevant information about the terrains from the user description and automatically creates the track with respect to the given details. Our terrain generator is capable of generating terrains with hills, trees and water bodies from TDL script. The engine assures rapid generated game contents and its diversity. Under the assumption that the interestingness of a track is a result of its structural features, the engine builds a model of the terrain from the details that the user specifies. This model is fed to entity generator for the generation of final results.

Creating an entire game requires significant effort and programming skills [3]. The proposed Engine helps the players to access the tools by themselves and to automate the contents of the game with simple TDL statements. Since this engine performs automated design, content generation shares the same challenges of actual designing. The modeling of the specified components of the game can be developed using 2-D cellular automata after feeding to the TDA. The effectiveness of the terrain generation depends on the quality of dynamically generated terrain which can be promised with the usage of cellular automata [4]. The engine provides personalized gaming experience and mechanisms to adjust elements of the game for the better experience of the player. User can specify anything as the components of the terrain which have a logical relation with each other. Even if the details given by the user is incomplete the engine manages itself to create complete information. User can evaluate the scenario described by them by rendering the environment to various games. Engine allows the user to choose a satisfied track and can export it into the game.

The paper is organized as follows. In section II, we review relevant related works. In section III, we present methodologies. In section IV we present the results of our research works. In section V the future works are depicted

II. RELATED WORKS

The work regarding skilled-matched content generation is also depicted in Text2Scene [5]. Text2Scene is an AI tool developed by Stanford. It automatically generates 3d scenes from natural text.Text2Scene first explicit the constraints in the user inputs and create a geometric scene from the details and renders a 3d scene [6]. The algorithm used in Text2Scene can adjust itself, if the people does not give logically related information .The system saves prior information and observations in a database. When the user gives various constraints, system uses this database to identify the most suitable spatial arrangements for placing objects in the scene [7]. The system is capable of self-adjustment and self-improvement.

Manuscript published on 30 September 2019
* Correspondence Author
Anagha Zachariah*, Dept of Computer Science and Engineering, Federal Institute of Science And Technology, Angamaly, India. Email: anaghazachariah@gmail.com
Pankaj Kumar G, Dept of Computer Science and Engineering, Federal Institute of Science And Technology, Angamaly, India. Email: pankaj.ekm@gmail.com
Dr Umesh D R, Dept of Computer Science and Engineering, PES College of Engineering, Mandya, India. Email: umesh.dr.pesce@gmail.com
Dr Arun Kumar M N, Dept of Computer Science and Engineering, Federal Institute of Science And Technology, Angamaly, India. Email: amrakmar.mn11@gmail.com

Text2Scene uses an automated metric which strongly correlates with human judgment. Text2Scene enables the user to generate a 3D scene from the user descriptions. The system accepts the description given by the user in natural language. The system parses the given input and applies constraints on what objects have to present and how they have to present.

Using the constraints and spatial knowledge, the engine creates a scene template. Then the objects are optimally placed [8].

For representing physical objects a collection of 3d models is needed. A 3d model database is used for this purpose. These datasets contain the names and the tags. The engine itself resizes the objects according to the scenario. It also orients them so that they have a logical relation [9]. The static support relations of objects in existing scenes can be used to establish this relation. The parent objects which support the child objects are characterized by six directions: up, down, left, right, front and back. During parsing, objects involved in the description and their relations are identified [10]. A set of associated keywords is also defined for later querying in the 3d model database. All other adjectives and nouns in the noun phrase have to be extracted for identifying various properties of the object [11]. The system also supports 3d scene interaction. The user can navigate the viewpoint with mouse and keyboard. The modification of objects, like select and modify objects is also possible through textual commands. The system tracks the user interaction and can adjust its spatial knowledge with respect to it [12].

The system constructs a scene template after optimizing position of all components in the screen. The user can update the textual information and the system can integrate the component interactions. The final 3d scene can be rendered by a graphical engine. In text2Scene engine the scenes are represented as graphs and the components of the screen are represented as the nodes of the scene [13]. The relationships are represented as edges.

III. METHODOLOGY

The major objective of our work is to enable the user to specify a terrain using Terrain Description language. TDL provides a mechanism for a layman to describe the desired terrain without learning complex map editors or software's. The TDL script will be processed by TDA to generate the basic structure of the terrain. Cellular automata is used to generate the individual elements such as hills, trees etc [14].

The Fig. 1 shows the architecture of the terrain generator. The individual components are described in details below.

A. Terrain Description Generator

TDL is a simple and easy to use scripting language. The main components of TDL are entity and entity attributes. Entity refers to any valid entity type such as forest, water, hills etc. Entity attributes are used to specify the coordinates of the entity. Fig 2. shows a part of TDL script. TDL script does not make use any specific tags or indentation. The entity type is specified using the keyword 'Entity' followed by separator ':' and the entity name.

The entity attributes are specified in the next lines using attribute name separated with separator and the attribute. The attribute start and end denotes the starting and ending coordinates of the entity. Here the attribute values are specified in a tuple. The basic attributes of the terrain such terrain size, terrain type are specified at the beginning of the script. A list of entities and their attributes are shown in Table1.

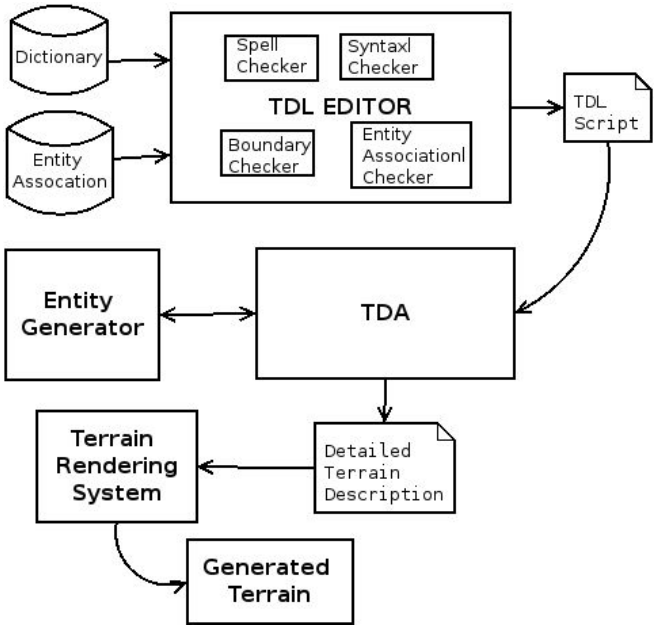

Fig. 1. Architecture of Terrain Generator

```
$screen_size:600,600
entity:forest
        start:(600,600)
        end:(260,270)
        type:"dense"
entity:forest
        start:(550,500)
        end:(550,270)
        type:"dense"
entity:occean
        start:(10,60)
        end:(10,270)
        type:"deep"
entity:hills
        start:(70,60)
        end:(100,200)
        type:"short"
entity:hills
        start:(260,100)
        end:(600,100)
        type:"high"
```

Fig 2 Sample Input Text

B. TDL Editor

TDL editor is designed to make it convenient for the user to write TDL script.

TDL offers all the features of a normal text editor. In addition to that TDL editor makes suggestions on entity types, attributes, entity bounds. It also generates warnings and errors. The development of the editor was forced by the following major reasons.

- Users may incorrectly enter the keywords.
- Users may not have used the correct syntax.
- Users may fail to specify the mandatory attributes.
- Users may incorrectly specify the coordinates of entities outside the map.
- Multiple entities may share some region due to incorrect coordinate specification.
- Users may incorrectly specify the coordinates of an entity inside the area assigned for another entity

Entity	Mandatory Attributes	Optional Attributes
hills	start,end	height_range, base_range, type, density
hill	start,end	height,base,type
forest	start,end	type,density
tree	start,end	type,height,age
terrain	start,end	type

Table 1. Examples of entities and their attributes

.

Algorithm 1: Terrain Boundary checker takes an entity and checks whether it is inside the terrain.

```
TerrainBoundaryChecker (terrain, entity)
  if entity.startx < terrain.startx or
entity.starty<        terrain.starty  then
        return True
  end if
  if entity.endx > terrain.endx or
entity.endy >         terrain.endy  then
        return True
  end if
  return False
```

Algorithm 2: Entity Boundary checker takes two entities and checks whether they share boundary

```
EntityBoundaryChecker (entity_1, entity_2)
  if (entity_1.startx >  entity_2.startx and entity_1.startx    <
entity_2.endx) and (entity_1.starty>entity_2.starty    and
entity_1.starty<entity_2.endy  ) then
        return True
  end if
  if (entity_1.endx >  entity_2.startx and entity_1.endx    <
entity_2.endx) and (entity_1.endy>entity_2.starty    and
entity_1.endy<entity_2.endy  ) then
        return True
  end if

  return False
```

Algorithm 3: Entity association checker takes two entities and entity association database as arguments

```
EntityAssociationChecker (entity_1,entity_2,associations)
  if entity_1 inside entity_2 then
      if  association(entity_1,entity_2)  in
association          then
        return True
    end if
  else
      if  association(entity_2,entity_1)  in
association          then
        return True
    end if
  end if
  return False
```

TDL editor has been designed to deal with the above mentioned issues. TDL editor deals with problem of incorrect entries by the usage of a spell checker. The spell checker uses a tiny dictionary containing the keywords. A syntax checker is used to issue warnings when the user input does not obey the syntax. The checking of mandatory attributes is also done by the syntax checker. To deal with remaining issues a boundary checker and entity association checker was designed. The boundary checker performs two iterations over the input script. In the first iteration, the validity of start and end attributes of all the entities are performed. For incorrect entries an error message is displayed on the bottom of the editor. In the second iteration the entities that share common regions are identified and warning messages are displayed in the bottom. Entity association checker performs a single iteration over the script and analyses various entity associations present in the script. If the user has placed a non-compatible entity inside another entity an error message is generated. For eg: User may specify incorrect association among entities such as a tree inside region specified for water or vice versa. Land inside water and water inside land are correct associations. The set of valid entity relationships are specified in an entity relationship database.

The TDL editor converts the script into XML format while saving.

C. Terrain Development Algorithm

TDA takes the TDL script as input and generates a Detailed Terrain Description. This can be viewed as an expansion phase.in which the finer details are generated. In the TDL script, user is only assigning an area for an entity. It does not specify the finer details. For example, when the entity specified by user is forest, user would not be specifying what kind of trees is combined to generate the forest. When the user specified entity is hills, the user would not be specifying the heights and base of individual hills in the group. If the generated entity is homogeneous in nature, it will look less natural.

TDA addresses the following challenges

- How individual entities should be rendered.
- How individual entities of a composite entity are picked.
- How the entities should change over time.

The individual entities are rendered as it is if all the attributes of the entity is specified. A user might not specify optional attributes such as height, age etc. For terrain dependent attributes, an intelligent choice has to be made. If the type is not specified for a tree, and the basic terrain is desert, it will be assigned 'palm'. or if the terrain is ice, it will be assigned 'birch'. For composite entity such as forest, hills the user would not be specifying the details of individual entities. If we are considering a forest TDA will have to determine the density of trees based on the forest, the distribution of height among the trees, age distribution etc. Our engine is capable of generating forest containing homogeneous vegetation only. However the Detailed Terrain Description still lacks originality, because the entities are constructed in perfect shapes such as rectangle, square, triangle etc. For imparting realistic shapes for entity a cellular automata is used.

D. Overview of Cellular Automata

One of the famous methods for the procedural content generation is cellular automata. It offers a self-organizing structure for a complex system. Cellular automata divide the entire problem space into cells. Each cell has a finite dimension and they can communicate with each other [15]. The shape of the cells can be achieved through the set of rules based on the state of neighboring cells. After multiple generations of the cells, all these patterns may together form a grid [16]. The final grid is dependable to the cells and the rules.

A cellular automaton is a model of a system of "cell" objects with the following characteristics

- Cells live in a grid. It can be single dimension, two dimensional or multidimensional [17].
- Each cell has a state. The number of cell states is finite.
- Each cell has a neighborhood. Typically the neighborhood consists of adjacent neighbors.
- Cell values are updated based on function defined using its neighbor [18].

Fig 3 Group of trees created using Cellular Automata

E. Entity Generator

The entity generator takes TDL as input and generates natural borders for the entities. Typically the user assigns perfect shapes such as rectangle, triangle etc in the TDL script. The input to the cellular automaton is a 2D matrix. Each element of the matrix represents a cell and their states are randomly initialized. Different sets of rules and neighborhood are assigned for different entities. Edges are also assigned different neighborhood and rules [21].

Example rules, a tree may die with 60% if it is surrounded by 4 neighbors. A tree may grow with probability to 70% if its neighborhood contains zero trees [22]. The resultant 2D matrix is returned to the TDA and the DTD is generated. Cellular automata are widely used because of its easiness to represent the system based on their natural relations and its ability to represent the relations with other cells [19]. In the Proposed engine cellular automata are used for the generation of components in the terrain. The self-replicating individual components together form the whole terrain. Simple discrete components can be modified to exhibit the same behavior as described by user [20]. A detailed explanation of cellular automaton is beyond the scope of this paper.

Algorithm 4: Entity Generator takes a composite entity and generates the details of the individual entities

```
EntityGenerator ( entity,n)
rules=GetRules(entity)
entity_new=entity
for i = 1 to n do
 for  j= 1 to entity.height do
  for k=1 to entity.width do
     neighbours=GetNeighbours(entity.content[j][k]
     entity_new.content[j][k]=ApplyRules(neighbours)
  done
 done
 entity=entity_new
done
return entity
```

IV. RESULT AND DISCUSSION

The development of terrain generator was done in Python and libraries used are PyQT, PyGame. PyQT was used in the development of the TDL editor . PyGame was used for the development for rendering engine. The Fig. 3 shows the output.

V. CONCLUSION

In this paper a cellular automata based terrain generator was proposed. The paper also introduces a Terrain Description Language (TDL) using which the user can specify custom terrains. The processing of TDL script using TDA was also explained. The terrain generator takes each composite entity and generates the detailed views by using cellular automaton. In the experiments the terrain generator was successful in generating terrains from the TDL script. The work can be further extended to generate 3d terrains.

ACKNOWLEDGMENT

Anagha Zachariah thanks Cognitive Computing Research Centre (CCRC), FISAT where the works has been carried out.

REFERENCES

1. G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player Modeling," in Artificial and Computational Intelligence in Games.
2. Daniele Loiacono,Luigi Cardamone and Pier Luca Lanzir, "Automatic Track Generation for high-end racing games using evolutionary computation", IEEE transactions on computational intelligence and AI

games,vol.3,No3,pp.245-260,september 2011.
3. Roland van der Linden,Ricardo Lopes, and Rafael Bidaraa,"procedural generation of dungeons",IEEE transactions on computational intelligence and AI in games,vol.6,no.1,pp.78-80,march 2014.
4. Sneha N.Dessai and Rachel Dhanaraj,"CReation of 3d scene from row text",IEEE international conference on recent trends in electronics information technology,pp.14661469-May 20-21,2016
5. Georgios N. Yannakakis and Julian Togelius,"Experience-DRiven Procedural Content Generation",IEEE transaction on affective computing,vol.2,no.3pp.147-158,july-september 2011
6. Adam Summerville,Sam Snodgrass,Matthew Guzdial,Christoffer Holmgard,Amy K.Hoover,Aaron Isaksen,Andy Nealen and Julian Togelius,"Procedural content generation via machine learning",IEEE transactions on games,vol.10,no.3,pp.257-258,september 2018
7. Levi H.S.Lelis,Willian M.P.REis, and Ya'akov Gal,"procedural generation of game maps with human-in-the-loop algorithms",IEEE transactions on games,vol.10,no.3,pp.271-281,september 2018
8. Theodosis Georgiou,Yiannis Demiris,"Personalised TRack design in car racing games",In2016 IEEE Conference on Computational Intelligence and Games (CIG) 2016 Sep 20 (pp. 1-8). IEEE.
9. Jiao JIan Wang,OLana Missura,"Racing tracks Improvisation",In 2014 IEEE conference on computational intelligence and games.
10. Angel X.Chang,Manolis Savva and Christopher D.Manning,"Interactive Learning of Spatial knowledge for text to 3D scene generation",in proceedings of the ACL 2014 Workshop on Interactive Learning,visualization and Interfaces(ACL-ILLVI)
11. Angel X. Chang, Manolis Savva, and Christopher D. Manning,"Learning Spatial knowledge for text to 3d scene generation",in EMNLP 2014.
12. Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D. Manning,"Text to 3D scene generation with rich lexial grounding",in ACL2015.
13. Fuping Yang,Yun Zhou,Xiaobo Luo,"Scene Layout in Text-to-Scene Conversion",in 2014 2nd international conference on systems and informatics(ICSAI 2014).
14. Young-Seol Lee and SUng-Bae Cho,"Context-Aware petri Net for Dynamic procedural Content generation in Role-playing game" in IEEE computational intelligence magazine.
15. M.Bevilacqua and A.Tsourdos,A.Starr"Particle swarm for path planning in a racing circuit simulation",IEEE.
16. T. Taylor, Play Between Worlds: Exploring Online Game Culture. MIT Press, Mar. 2006.
17. C. Bateman and R. Boon, 21st Century Game Design. Charles River Media, 2005.
18. N. Shaker, G.N. Yannakakis, and J. Togelius, "Towards Automatic Personalized Content Generation for Platform Games,"Proc. Artificial Intelligence and Interactive Digital Entertainment,pp. 63-68, Oct. 2010.
19. C. Conati, "Probabilistic Assessment of User's Emotions in Educational Games," J. Applied Artificial Intelligence, special issue on merging vognition and affect in HCI, vol. 16, pp. 555-575,2002.
20. G.N. Yannakakis, J. Hallam, and H.H. Lund, "Entertainment Capture through Heart Rate Activity in Physical Interactive Playgrounds," User Modeling and User-Adapted Interaction, special issue: affective mModeling and adaptation, vol. 18, nos. 1/2, pp. 207-243, Feb. 2008.
21. S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci,"Modeling Enjoyment Preference from Physiological Responses in a Car Racing Game," Proc. IEEE Conf. Computational Intelligence and Games, pp. 321-328, Aug. 2010.
22. R. van der Linden, R. Lopes, and R. Bidarra, "Designing procedurally generated levels," in Proc. 2nd Workshop Artif. Intell. Game Design Process, 2013, pp. 41–47.

AUTHORS PROFILE

Anagha Zachariah C, pursuing her Bachelor of Technology (B.Tech) in Computer Science from Federal Institute Of Science And Technology , Angamaly, Kerala. Her research interest pertains to areas of Cognitive Computing and Game Theory

Pankaj Kumar G completed his Bachelor of Technology (B.Tech) from Mahatma Gandhi University,Kottayam in 2010. He completed his Master of Technology (M. Tech) in Computer Science and Engineering from Mahatma Gandhi University,Kottayam in 2013. His research interest pertains to areas of Cognitive Computing, Machine Learning, Data Analytics and Game Theory. He has published more than 10 research papers. He has 9 years of teaching experience

Dr. Umesh D R completed his Bachelor of Engineering (BE) in 2002 and Master of Technology (M. Tech) in 2009 from VTU, Belgaum, Karnataka. He completed his Ph, D from University of Mysore in 2017. His research interest pertains to areas of Data Mining & Analytics. He has published more than 10 research papers. He has 15 years of teaching experience.

.Arun Kumar M N completed his Bachelor of Engineering (BE) from University of Mysore in 1996. He completed his Masters in Computer Science and Engineering from VTU, Belgaum, Karnataka and Ph.D from University of Mysore in 2013. His research interest pertains to the areas of Image Processing and Data Mining. Dr. Arun kumar M N has published more than 35 research papers in various international journals; most of them are indexed by SCOPUS and SCIE. He has 22 years of teaching experience. He is a life member of CSI and ISTE and has attended more than 25 Seminars/Workshops/Conferences. Dr.Arun Kumar M N is recognized as a Ph.D Research Supervisor by Kerala Technological University (KTU), Kerala and Visvesvaraya Technological University (VTU), Karnataka and is guiding 3 Ph.D research scholars. He is a reviewer of some international journals and was session chair for some reputed international conferences. He has organized 3 international conferences and 2 Faculty Development Programmes.