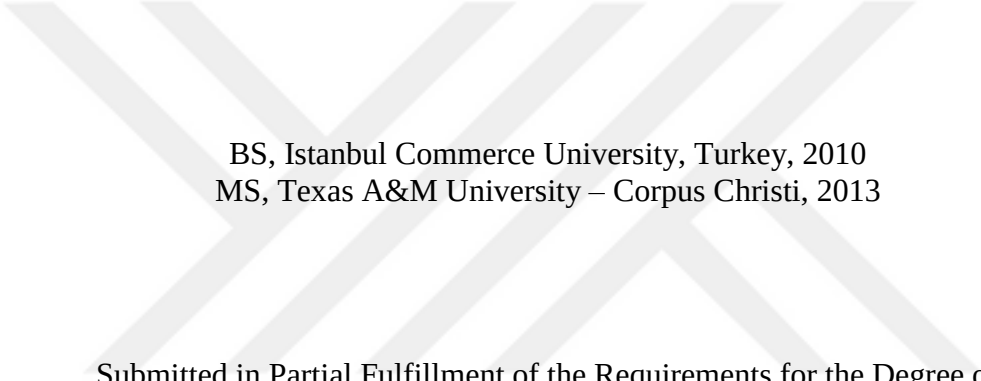


FLOOD SIMULATION AND VISUALIZATION FRAMEWORK USING POSITION-BASED
FLUIDS

A Dissertation

by

ISMAIL ALIHAN HADIMLIOGLU



BS, Istanbul Commerce University, Turkey, 2010
MS, Texas A&M University – Corpus Christi, 2013

Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in

GEOSPATIAL COMPUTING SCIENCES

Texas A&M University-Corpus Christi
Corpus Christi, Texas

August 2019



© Ismail Alihan Hadimlioglu

All Rights Reserved

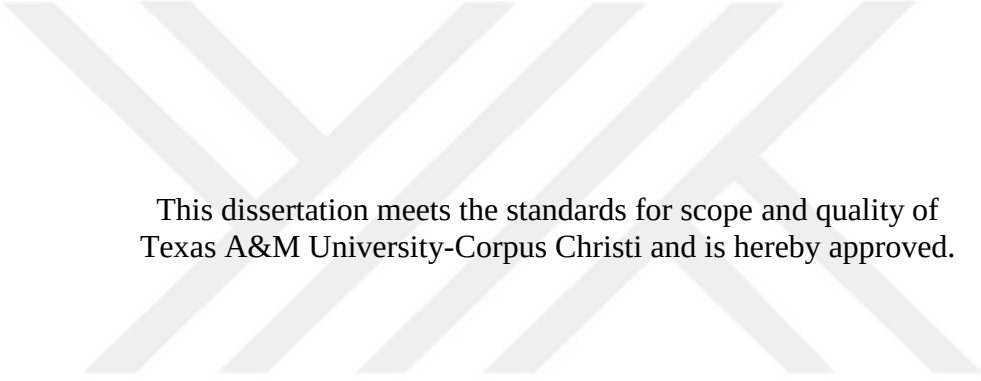
August 2019

FLOOD SIMULATION AND VISUALIZATION FRAMEWORK USING POSITION-BASED FLUIDS

A Dissertation

by

ISMAIL ALIHAN HADIMLIOGLU



This dissertation meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Scott A. King, PhD
Chair

Maryam Rahnemoonfar, PhD
Committee Member

Michael J. Starek, PhD
Committee Member

Chuleeporn Changchit, PhD
Graduate Faculty Representative

August 2019

ABSTRACT

Assessing flood resilience of urban environments has been an ongoing work as flood modeling techniques and various ways of visualizing flood help communities discover weaknesses and become more resilient against disasters. From 3D urban environment reconstruction to flood models that account for urban environments, tests are conducted to gain more insight regarding local dynamics that affect water flow. Recent work has been contributing to this field, by proposing methods to ease the preliminary evaluation yet efficiency is a problem, especially for real-time particle-based simulations. This work proposes a framework to generate the city model from OpenStreetMap (OSM) data, incorporate simulation parameters in the form of friction coefficients and storm drain information and then using mechanics such as precipitation and soil absorption to simulate and visualize flooding in 3D. As an efficient visual aid, a flexible depth mapping engine, which can transition between regular and adaptive grids is provided.

To process the city information, OSM data alongside Satellite Radar Topography Mission (SRTM) elevation information is extracted. Upon efficient construction of a simulation-ready virtual environment, position-based fluids are utilized to supply an efficient method of 3D simulation as several shortcomings of other prominent approaches are alleviated through position-based dynamics that this method introduces. Localized rainfall mechanics with spatial distribution differentiator and soil absorption mechanics are embedded to accommodate particle addition and deletion from the simulation environment.

Data collected from OpenStreetMap is utilized to generate digital city models of selected areas. Using 25km² of sections from each city from different regions of the world, the features are visualized, and city models are generated. Preliminary evaluation showed that all 6 areas are reconstructed and visualized properly for further evaluation. Moreover, it is concluded that, the

3D urban environment reconstruction module built for this work can accomplish parsing and rendering areas of varying extent, adding and removing features, extracting simulation parameters through feature-level friction parameters and storm drain information extraction. Upon generation of such a city model, it can be passed as an input to the flood visualization framework for viewing and simulating. A limitation is that larger sections of cities may not utilize real-time viewing experience due to performance issues caused by large number of features being rendered.

To supplement 3D particle-based flood simulations, a depth mapping engine is provided, which can efficiently track particle clusters. Considering several environmental parameters and simulation performance, the mapping engine has the power to choose a representation between regular and adaptive grids. Through utility of Jenks natural breaks for classification purposes, appropriate coloring is provided that adds context to the environments being investigated.

An experiment was conducted with the position-based flood visualization framework to assess its visual fidelity to the river flooding event of 2000, in Bewdley, UK. Utilizing SRTM with 1 arc-second resolution for preparation of the 3D digital model of the river and surroundings, the flooding was represented with approximately 1 million particles. The results were satisfactory as the flood extent was represented properly. Furthermore, comparison with an existing model investigating the same event showed that the visual fidelity of FloodSim was superior to the other method. Consequently, the framework can be a useful tool for practitioners who have sufficient of information about the study area and would like to visualize flooding using a particle-based approach for real-time particle tracking, flood path analysis and incorporating precipitation to their models.

DEDICATION

I dedicate this dissertation to the ones who fell defending their homeland against putschists on July 15, 2016 in Turkey. Thanks to their sacrifice, I was able to fly to the U.S. peacefully, continue my education and my career at Texas A&M University – Corpus Christi, and successfully completed this dissertation.



ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. King, and my committee members, Dr. Rahnemoonfar, and Dr. Starek, for their guidance and support throughout the course of this research. It has been a long adventure for me as I have been on campus since 2011, first as a student, then as a faculty and now as both. Dr. King has been advising me since Master's degree and I am glad that we have been working together. His genuine help and support surely benefited my work and helped me get to where I am at this point.

I would also like to acknowledge my family for their help and support throughout the entire process. I would not be able to succeed without their love and care.

TABLE OF CONTENTS

CONTENTS	PAGE
ABSTRACT	v
DEDICATION	vii
ACKNOWLEDGEMENTS	viii
TABLE OF CONTENTS	ix
LIST OF FIGURES	xi
LIST OF TABLES	xiv
CHAPTER I: INTRODUCTION	1
1.1. BACKGROUND AND MOTIVATION.....	1
1.2. CONTRIBUTIONS	5
1.2.1. City Maker: 3D Urban Environment Extraction and Visualization	6
1.2.2. 2D Visualization of Flooding Through Adaptive Spatial Resolution	7
1.2.3. FloodSim: Flood Simulation Framework Using Position-Based Fluids	8
1.3. STRUCTURE	10
CHAPTER II: REVIEW OF THE LITERATURE	11
2.1. 3D ENVIRONMENT REPRESENTATION AND VISUALIZATION	11
2.1.1. Background	12
2.1.2. CityGML and Hierarchical Representation of City Information	13
2.1.3. OpenStreetMap (OSM) and Crowdsourced Data	15
2.2. FLOOD MODELING AND VISUALIZATION	18
2.2.1. Background	19
2.2.1. One-Dimensional Hydrodynamic Models.....	20
2.2.2. Two and Three-Dimensional Hydrodynamic Models	21
2.2.3. Other Specialized Flood Models	24

CHAPTER III: METHODOLOGY AND SYSTEM DESIGN.....	26
3.1. CITY MAKER: 2D/3D URBAN ENVIRONMENT RECONSTRUCTION	27
3.1.1. Feature Extraction and OSM data Parsing	31
3.1.2. Drawing Pipeline and Visualization.....	34
3.1.3. Texture and 3D Model Mapping	37
3.1.4. Post Processing and Addition of Simulation Parameters	39
3.2. 2D VISUALIZATION OF FLOODING THROUGH ADAPTIVE SPATIAL RESOLUTION.....	41
3.2.1. System Design.....	43
3.2.2. Component Parsing and Scene Processing.....	44
3.2.3. Depth Mapping and Adaptive Resolution	48
3.2.4. Classification and Coloring of the Depth Information	54
3.3. FLOODSIM: FLOOD SIMULATION FRAMEWORK USING POSITION-BASED FLUIDS	56
3.3.1. Global and Local Friction Model	60
3.3.2. Soil Absorption Mechanics	63
3.3.3. Cloud and Rainfall Mechanics	66
3.3.4. Workflow	70
CHAPTER IV: RESULTS.....	74
CHAPTER V: SYSTEM EVALUATION	101
5.1. Evaluation Metrics	101
5.2. Case Studies	104
5.2.1. 3D Environment Reconstruction	104
5.2.2. Flood Simulation and Visualization	114
CHAPTER VI: CONCLUSION AND FUTURE WORK	122
6.1. Conclusion	122
6.2. Future Work	125
REFERENCES	128

LIST OF FIGURES

FIGURES	PAGE
Figure 1. System overview.	27
Figure 2. Parts of feature extraction and drawing hierarchy of City Maker.	29
Figure 3. Processes City Maker utilizes to produce a city model.	30
Figure 4. A Multipolygon construct defined by relations, ways, nodes and tags.	32
Figure 5. Features currently implemented for the City Maker.	36
Figure 6. A scenario that data stores are utilized to select an object to draw.	38
Figure 7. Overall system design for 2D visualization and mapping engine.	43
Figure 8. Parameters each component reader process	45
Figure 9. Digital model and the height map generated from the surface.	48
Figure 10. Area representation using a) regular grid, b) adaptive grid.	50
Figure 11. Computation of depth using a) relative height differences, b) fluid volume through number of particles per grid.	53
Figure 12. Height map generated using a) 4 classes, b) 6 classes and c) 9 classes.	55
Figure 13. Multi-hue and single-hue map coloring options.	56
Figure 14. Overview of the simulation mechanics.	57
Figure 15. Color coded representation of the application of Manning coefficients.	62
Figure 16. Process of generating absorption coefficients for the study area.	65
Figure 17. A cloud object and its significant parameters.	67
Figure 18. Representation of a cloud animation object timeline.	69
Figure 19. User interface of FloodSim.	71
Figure 20. FloodSim Workflow.	73
Figure 21. User Interface of the City Viewer.	74
Figure 22. Visualization of the TAMUCC Island and parts of Ocean Drive.	75
Figure 23. Visualization of parts of Corpus Christi in 3D wireframe.	76
Figure 24. Visualization of parts of Athens, Greece.	77
Figure 25. Selected area to extract OSM information.	79

Figure 26. City of Bursa and the mountain Uludag.	79
Figure 27. Mesh of the study area built from SRTM data.	80
Figure 28. Textured mesh of parts of Bursa, Turkey.	81
Figure 29. Representation of parts of Corpus Christi, TX through a) City Maker and b) FloodSim.	82
Figure 30. Representation of parts of Brooklyn, NY through a) City Maker and b) FloodSim using wireframes to render.	83
Figure 31. Visualization of the flooding in a) 2D regular grids, b) 3D aerial view.	85
Figure 32. Visualization of the neighborhood flooding using a) regular grids, b) adaptive grids.	86
Figure 33. Visualization of the canyon flooding using a) regular grids, b) adaptive grids.	87
Figure 34. Visualization of the mountain flooding using a) regular grids, b) adaptive grids.	88
Figure 35. Visualization of the study area in 2D using different color schemes.	89
Figure 36. Timeline of flooding using a) regular grids, b) adaptive grids.	90
Figure 37. Visualization of flooding without the local friction model for a hypothetical environment with hills, 400K particles.	92
Figure 38. Visualization of flooding with the local friction model for an hypothetical environment with hills, 400K particles.	93
Figure 39. Visualization of the depth in 2D for 50K particles a) with no local friction model and b) with local friction model.	94
Figure 40. Generated clouds across the study area.	95
Figure 41. A giant cloud, dropping 400K particles.	96
Figure 42. Animation of 400K particles dropped onto the floodplain in 3D.	96
Figure 43. Animation of 400K particles dropped onto the floodplain in 2D.	97
Figure 44. Storm sewer information loaded in City Maker.	99
Figure 45. Parts of the environment without soil absorption, 130K particles.	99
Figure 46. Parts of the environment with soil absorption, 130K particles.	100
Figure 47. Visualization of 6 different cities: a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.	105
Figure 48. Chart of the relationship between number of nodes and time it takes to parse an OSM file.	108
Figure 49. Chart of the relationship between number of features and frame rates.	109
Figure 50. Parts of the city of Corpus Christi with storm sewer inlets.	110
Figure 51. Visualization of storm drains in Corpus Christi	111

Figure 52. Comparison of generated digital city models: a) City Maker – Corpus Christi, b) CityEngine – Corpus Christi, c) City Maker – San Jose, d) CityEngine – San Jose	113
Figure 53. Photo showing Bewdley flooding.	115
Figure 54. Terrain generated from SRTM and passed to FloodSim	116
Figure 55. 1 million particles generated in the river bed with particle radii of: a) 0.08 units, b) 0.13 units, c) 0.18 units, d) 0.23 units, e) 0.28 units	117
Figure 56. Usage of memory within 5 minutes of simulation with constant particle generation, each particle having 0.18 units of radius.	118
Figure 57. Map of Bewdley, showing river flooding, extent of extreme flooding scenarios and flood defenses	119
Figure 58. Animation of river flooding with particle radius 0.18 units and a total of 1 million particles.....	120
Figure 59. Comparison of flooding sections a) Bewdley data, b) FloodSim, c) Flood Spreading Model [137].....	121

LIST OF TABLES

TABLES	PAGE
Table 1. Performance comparison of parsing and viewing three different cities and their embedded features	78
Table 2. Performance comparison of regular grids and adaptive grids under different environmental conditions	92
Table 3. Comparison of number of features, time it takes to parse OSM files and load CM files, to render the city models of a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.....	107
Table 4. Comparison of number of nodes, ways and relations found in OSM data files of a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.....	108
Table 5. Comparison of features between Esri CityEngine and City Maker.	112
Table 6. Comparison of inundation areas and differences of a) Bewdley data, b) FloodSim, c) Flooding Model [140].	121

CHAPTER I: INTRODUCTION

Flood modeling and analysis has been a vital research area to reduce damages caused by flooding and to make urban environments resilient against such occurrences. With the utility of 2D and 3D hydrodynamic flood models, visualization of flooding has helped practitioners and city planners recognize and assess certain patterns, dangers, and possible solutions to the issues in the environment. To accomplish a realistic simulation, environment must be defined accurately, and local dynamics must be realized to analyze such events. This section contains background, motivation and contributions of this work.

1.1. Background and Motivation

Throughout history, flood has been an integral part of our lives. Most of the time, flood is a major disaster that has been responsible for loss of millions of lives and collapse of cities. For centuries, we have been trying to fight against this natural disaster utilizing variety of strategies. Prolonged periods of rainfall is a dangerous condition that may cause environmental damage and loss of lives. It is noted that flash flood casualties and occurrences exceed that of the other types of floods [1]. An earlier example of such a disaster is known as St. Mary Magdalene's flood, which occurred around central Germany, caused thousands to die. The flood caused damage or destruction on many buildings, bridges and mills. St. Mary Magdalene's flood was significant due to the dry soil, which was caused due to heat and partial drought, was unable to absorb the waters, causing a devastating soil erosion [2]. With the introduction of modern road networks, additional parameters were introduced that would reshape emergency management efforts. Recent studies show that, although flood may not be totally prevented, proper coordination of entities in

emergency management can significantly relieve the damage and issues caused by it [3]. In order to be well-prepared for any unexpected disaster, city planners and emergency management officials need to know the strengths and weaknesses of the city. Considering that proper development of road networks and urban expansion go hand-to-hand [4], road networks are vital for estimating potential risks within modern cities.

It is an important approach to test the environmental hazards utilizing hydrodynamic simulation techniques to investigate the area in question considering variety of dynamic parameters for better understanding of the underlying mechanics. 2D simulations are preferred for their clear representation of the parameters being investigated and efficient tracking of changes in the study area. This approach provides a better overview of a larger area, yet it focuses on details less than a 3D model as designing the environment in 3D would allow for an additional dimension, or depth [6]. Consequently, utilizing a combined approach would be suitable to visualize the city in 2D and to focus on the details for some parts of the city by visualizing the features in 3D [6]. It is vital to underline that there are numerous methods to implement the simulation environment. Moreover, once the environment is generated, an accurate and efficient fluid simulation should be utilized to test flooding. For efficiency purposes, data structure optimization, reconstruction of the water model, using several approximations regarding environmental parameters and decreasing resolution might be required.

Fluid simulations has been used in a variety of industries such as automotive [7], aerospace [8], defense [9], life science [10] and machinery [11]. As the processing power and graphical representation have been improved over the years, advanced techniques have been proposed to accommodate the needs of these industries [12]. Fluid dynamics have been used to demonstrate ocean waves, tsunamis and flash floods to help improve emergency management efforts. Through

these simulations, otherwise challenging to detect aspects of these phenomena can be visualized and data can be used to make predictions or analysis about the conditions within the study area.

There have been numerous tools implemented to demonstrate storms and flash floods. Advanced Circulation Model (ADCIRC) [13] was developed to solve the equations of motion for a moving fluid on a rotating earth. With it, a 2D depth integrated model and a fully detailed 3D model can be used. Considering the use cases of ADCIRC, it is safe to say that it has a strong emphasis on the emergency management aspect of events. Furthermore, hurricane storm surge and flooding can be forecasted using this program. Delft3D [14] simulates 2D and 3D flow, sediment transport and waves and interrelationships between different mechanics of such phenomena. Its use includes flood risk estimation, adaptive delta planning, city planning and environmental applications. As it is quite expandable, with additional modules and tools, it is possible to demonstrate flooding overlaid in a 3D environment. HW3D [15] describes a tool and techniques to get a realistic looking, 3-dimensional, real-time visualization of flood information. It considers water level as a parameter and visualizes increases and decreases in water level as the flood progresses. Flow-3D [16] provides a complete computational fluid dynamics (CFD) simulation platform for engineers investigating the dynamic behavior of fluids. It utilizes TruVOF technique, which is based on volume of fluid (VOF) technique [17], to compute water simulation. Considering all these approaches, there is a clear indication that providing flexibility and efficiency are both concerns for a simulation. While some of these tools provide a platform for accurate representation of the water surface, the others focus more on creating a proper approximation to infer certain information from the environment.

An important consideration regarding 3D environment reconstruction approaches in the literature is the utility of CityGML structure to provide definitions of cities that can exchange

information [18, 19, 20]. Certain regions of Germany built on this idea and provided city definitions using CityGML notation [21]. Nevertheless, due to its syntax and semantics, it is restrictive and except the usage of ADEs [22], it is not open to customization. On the other hand, OSM provides an extremely flexible notation through custom tags [23]. Consequently, it has a huge potential if the data can be converted into a format that is similar to CityGML to describe buildings and features and provide custom tags to define additional properties regarding the environment. OSM is a crowdsourced map data source, which is open for improvements and therefore, exploiting its flexibility, environmental simulations can be designed and implemented. For OSM processing and parsing, current tools do not provide enough in terms of extent of data that they can parse and customizations they could utilize. Moreover, environmental simulations are not well-tested using OSM data.

For flood simulation and visualization, numerical approaches are utilized frequently for efficiency purposes. However, in the recent years, 3D flood simulation and visualization has been gaining a lot of interest due to advancements in graphics processing units (GPU) and innovation of efficient processing techniques of fluid flow. Among methods utilized to simulate flooding, position-based fluids (PBF) [24] is not utilized as it is a relatively new approach. Instead, many others utilize smoothed particle hydrodynamics (SPH) [25, 26], which has problems regarding simulation stability compared to PBF [24]. Furthermore, PBF is designed for real-time performance in mind [24] and consequently, if it can describe the flood extent properly, then the method can be utilized in practical scenarios.

To provide a flexible 3D environment reconstruction approach, City Maker is implemented. Through utilization of OSM data alongside SRTM with 1 arc-second resolution, the environment is reconstructed. Compared to industry-standard Esri CityEngine, City Maker provides greater

extent for processing. Furthermore, by providing a viewing engine alongside simulation parameters, the model can be viewed within a simulation, which is an advantage for urban environment reconstruction tools for visualization purposes. Default contextual coloring of City Maker makes it easier to recognize significant features. As City Maker can export the city model into other applications in OBJ format, our flood simulation and visualization framework, FloodSim, receives the model as input as a collision surface for fluid-solid interactions. An efficient depth mapping engine is provided for ease of understanding in complex situations to visualize the 3D environment as 2D depth map. Considering the depth mapping approach, a dynamic mapping engine that can transition between regular and adaptive grids depending on environmental features and performance is presented. Using natural breaks for classification and variety of commonly accepted single-hue and multi-hued color sets help recognize the depth patterns in the study area. Regarding the flood simulation framework, environmental dynamics are built on top of the position-based fluid method, which provide details such as feature-based friction coefficients, utilization of storm drain information alongside soil absorption coefficients and usage of cloud and rainfall mechanics to represent the precipitation information of the modeled study area. Using these different mechanics provide a dynamic representation of the environment and allow practitioners to input their environmental observations to the system for further calibration and more realistic representation.

1.2. Contributions

This work focuses on building a flood simulation and visualization framework through reconstruction of a 3D urban environment, visualization of the environment through a 2D adaptive depth map and a 3D representation of the study area. Hence, it is appropriate to underline these closely related fields that this work contributes to.

1.2.1. City Maker: 3D Urban Environment Extraction and Visualization

Firstly, 3D urban environment extraction and utilization is vital for a variety of tasks. From utility management to demographics analysis, 3D visualization is utilized to provide more context to otherwise challenging to understand domains. Such visualization approaches have been utilized recently for both static and dynamic visualization purposes. As simulations, and more specifically flooding is considered, some form of visualization would be helpful in determining flooded areas and help practitioners recognize significant patterns without the need to evaluate complex numerical models. Consequently, such visuals aid simulations and have been used recently for flood simulations.

This work provides 3D environment reconstruction mechanism, City Maker, using crowdsourced data from OpenStreetMap and Satellite Radar Topography Mission (SRTM) data for elevation. Significant features such as buildings, roads and farms are extracted from OSM data and then utilizing a 3D drawing pipeline, they are visualized. On top of the visualization aspect, each feature is provided simulation parameters to be utilized in dynamic simulations. Such an approach of embedding parameters into a 3D environment and passing it to the simulation engine would help visualization of complex simulations utilizing the extracted environment without the need for further processing.

More specifically, City Maker [27] contributes:

- 3D environment reconstruction from OSM data with no limitations on extent. Unlike available tools, extent is not a concern for City Maker as it can receive OSM data of any size.
- Through reading storm drain information and Manning coefficients [28], the information is embedded into the city model, which makes OSM map data usable

for environmental simulations. Custom formatted ‘stormsewer’ and ‘roughness’ tags are embedded into the OSM data for further exchange of information.

- Efficiency is provided through restructuring the environmental information acquired from variety of data sources. By preparing the absorption and friction coefficients at an earlier stage, namely the environment reconstruction stage, City Maker provides additional efficiency for the flood simulator.

1.2.2. 2D Visualization of Flooding Through Adaptive Spatial Resolution

Although 3D environments and 3D hydrodynamic simulations are very useful for visualization of different aspects in a study area, often this compression of information makes in unrecognizable due to the chaotic nature of such simulations. Frequently, 2D visualization is provided alongside 3D to aid understanding of certain dynamics occurring within the study area. Nevertheless, there are numerous ways of handling such mapping information and consequently, this area is suitable for further investigation to provide an efficient and a dynamic approach to display important changes within the environment. An efficient algorithm would alleviate performance issues caused by processing of fluid particles in particle-based simulations.

In this work, considering state-of-the-art approaches discussed, position-based fluids [24] was utilized for simulating flood, which consider a Lagrangian description and provide certain benefits over traditional smoothed-particle hydrodynamics (SPH) [25] methods by providing stability [29] and alleviating the neighborhood requirement [24] of traditional SPH, which has an expensive computation of neighborhood of particles [30]. On top of the simulation engine, a 2D mapping engine that can keep track of the flood, while generating a heat map of the study area, to visualize water depth in 2D for ease of understanding is implemented. As the simulation takes place, the dynamic mapping engine can decide on representing the flood using a regular grid or an

adaptive grid. The dynamic approach helps with accuracy on demand, while keeping the simulation efficient when the situation arises.

The dynamic approach of 2D visualization of flooding [31] provides:

- A flexible mapping engine, which can be used to visualize depth information through utilizing Jenks natural breaks [32, 33] for classification and coloring the generated depth map with predefined single-hue and multi-hue color schemes to enhance the readability of the depth map.
- An implementation of adaptive spatial resolution through utility of regular grids and quadtrees considering environmental parameters and computational expense to achieve efficiency, as well as detail on demand.
- A novel method of depth mapping for 2D and 3D Lagrangian fluid simulations to visualize particle clusters, or depth information, more precisely.

1.2.3. FloodSim: Flood Simulation Framework Using Position-Based Fluids

Although 1D and 2D simulations in the context of flood visualization have been used, with the advent of modern GPUs, 3D representation and especially particle-based 3D simulations have been used in various works. 3D simulations, according to numerous works, are useful in providing a more detailed description of the study area. Using 3D visualization alongside 2D has been mentioned to provide a clearer understanding of the underlying mechanics and to assess the state of flooding through visual cues.

By providing a 3D flood simulation and visualization framework utilizing position-based fluids, benefits of position-based dynamics in terms of simulation stability and neighborhood analysis are achieved. Through implementation of a local friction model, rainfall and cloud

mechanics and soil absorption mechanics, an approximate description of the study area showing effects of flooding is delivered. The framework mentioned in this section includes the approaches described in Sections 1.2.1 and 1.2.2 as well.

To be more specific, FloodSim framework presents:

- An efficient flood simulation framework using position-based fluids to provide real-time performance in a variety of scenarios.
- A local friction model through derivation of Manning coefficients [28] of several features in the virtual environment to supplement global friction model of position-based fluids alongside an approach of particle addition and deletion through the utility of cloud, rainfall and soil absorption mechanics to provide dynamic behavior to the simulation environment.
- Integration of real-world data through parsing storm drain information and precipitation data of the study area to provide practitioners better calibration and customization of the environment.
- A novel position-based fluids approach in urban environment flooding visualization. There is no other work currently utilizing position-based fluids method for such purposes. As position-based fluids is known for its efficiency and stability, this provides benefits over some other traditional SPH-based approaches.

1.3. Structure

This dissertation is structured as follows: After the first chapter, which contains a brief introduction, objectives, contributions and the dissertation structure, a literature review is provided. In Chapter 2, recent works regarding virtual 3D environment generation, flood modeling and flood simulation methods are discussed, and relevant aspects are underlined.

Chapter 3, which is divided into 3 major subsections, contains our methodology. The subsections discuss City Maker, 3D urban reconstruction tool, 2D flood visualization using dynamic adaptive spatial resolution and finally FloodSim, the 3D flood simulation framework using position-based fluids and variety of environmental parameters to visualize surface runoff. Chapter 4 contains results from the methodology presented and provides a discussion and comparison with recent similar work. Through underlining the output and the key advantages of our approaches, we emphasize the contributions provided with each result. Chapter 5 provides system evaluation by using the proposed framework with several real-world scenarios using real data. Findings are provided, underlining the advantages and disadvantages of utilizing the framework discussed in this dissertation. Lastly, Chapter 6 concludes with final remarks and a discussion about potential future work and areas that can be investigated.

CHAPTER II: REVIEW OF THE LITERATURE

Our focus in providing flood simulation and visualization framework using position-based fluids method, investigates several important areas of study. This work encompasses utility of crowdsourced and open GIS data, 2D and 3D environment reconstruction and visualization, depth mapping for 3D environments using adaptive spatial resolution and finally, 3D surface flow simulation and visualization to represent flooding scenarios. In the following sections, these topics are investigated more in detail and relevant studies to our work are underlined. The contents of this chapter can be listed as:

- Section 2.1. discusses about environment reconstruction and visualization with an emphasis on hierarchical formats and crowdsourced data.
- Section 2.2. evaluates flood modeling approaches and models used in the industry and research. This section also contains recent similar work of flood simulation and visualization.

2.1. 3D Environment Representation and Visualization

Visualization of virtual environments has been an important topic as many different areas of study utilize digital city models for visualization or simulation purposes [34, 35]. With recent advancements, larger corporations like Apple, Google and Microsoft provide their 3D visualization of city information as part of their map services. Moreover, such visualization is provided with varying levels of detail for optimization purposes, helping the services be utilized efficiently while providing detail when required [19]. Such detail might be required if a realistic visualization of the environment is needed. Moreover, 3D terrain, roads and building models are useful for augmenting the experience of the users and professionals alike and therefore, methods to provide 3D information has been an area of focus for years [36].

2.1.1. Background

Although storing the city information as shape files for 2-dimensional models may be considered appropriate, such large data providing 3D information requires an efficient way of storage and processing [19]. Especially, storage might be a major issue especially for larger study areas [18]. Furthermore, as the data is going to be utilized for other purposes and applications, a proper geometric and semantic structuring of the data for better accuracy is essential [34]. Most often, 3D information is provided through hierarchical and semantically consistent formats [20, 19]. Through such an arrangement, different applications, such as the ones for flooding [22, 37, 38] or urban development planning [39, 40] can utilize the data according to their requirements. In addition, providing multiple layers and storing the data in a hierarchical structure, more details can be incorporated into city description, which is essential for realistic and complex scenarios.

It is possible to model 3D environments using manual modeling through specialized software by creating models of features through software like SketchUp [41] and extracting the existing information of the study area by software such as CityEngine [42, 43], which incorporates importing 2D and 3D data from different map sources.. As urban environments are quite complicated in nature, most of the earlier studies of 3D environment reconstruction focus on automated generation, mapping and rendering of digital city models [44]. With the advent of various approaches to visualize environments, LiDAR, photogrammetry and aerial imagery have been used to build 3D models of urban environments as the terrain and variety of features can be extracted and defined properly [45, 46, 47, 48]. Through machine learning and pattern recognition techniques on 3D point clouds derived from LiDAR, automatic classification of buildings, ground and vegetation can be provided, and 3D reconstruction of building models can be accomplished

[49]. Furthermore, road extraction, lane detection and variety of other tasks related to 3D urban environment generation can be accomplished through LiDAR [50, 51].

Moreover, with the availability of open hierarchical map data sources, it is possible to use data from sources like OpenStreetMap (OSM) and convert the preexisting 2D data into 3D features [34, 52]. As OSM data contains point, line and polygon features [53], which can be exported on demand, it is a convenient data source to utilize. For hierarchical description of city models, CityGML is an accepted standard to represent large 3D urban environments and therefore, it has been utilized in many works [19, 18, 54].

2.1.2. CityGML and Hierarchical Representation of City Information

CityGML [55] is an open standardized data format based on geography markup language (GML) to exchange information about cities and environments. As it is a markup language, it has hierarchical qualities to define objects and parts in terms of relationships and dependencies [20]. Moreover, level-of-detail can be arranged for efficient description when required, and for additional details when needed. Upon arranging the environmental information in such a hierarchical way, the data can be used modularly for a variety of applications for the purpose of emergency management [56], urban planning [57] and transportation modeling [58]. Hence, CityGML is quite beneficial for representing and visualizing 3D city and landscape information.

Although classes are defined within CityGML, it is designed as an extensible format to be able to represent geospatial objects globally and therefore, it can create generic objects and generic attributes for classes and features that are not defined within the CityGML. Additionally, Application Domain Extensions (ADE) can be used to add more features [59]. ADEs provide more structured information through an extra XML schema definition file [60]. As the extension is formally defined, exchange is easier between different applications.

CityGML is currently being employed by numerous applications that perform variety of important tasks. In Germany, municipalities have been creating and exchanging 3D city information utilizing CityGML to perform and manage urban planning, business development and tourism [20]. Through making these models CityGML compliant, more information can be shared between cities and analysis can be performed on a national scale. Furthermore, CityGML could be utilized for homeland security scenarios and emergency management [21]. CityGML is also utilized in training simulators, which can utilize 3D cities for training purposes such as driving simulators, freight simulators, and similar applications [21].

CityGML provides different levels of detail (LOD), which describes the features more and more detailed as the level of detail is increased [61]. If less description is required, then lower levels of detail would be preferred. As different LODs can be provided for the same environment, it offers the possibility of a clearer visualization while enabling further analysis and processing when required [21].

LOD0 represents the digital terrain model alongside polygons drawn to represent features. With this level of detail, not much can be visualized in 3D. LOD1 allows for visualization of the features in 3D with low detail. In this level, buildings are represented using building blocks and other details are omitted. LOD2 provides additional information, such as roof structures, gardens, balconies and others. LOD3 incorporates architectural models and provides detailed roof structures alongside walls, doors and windows. LOD4 takes LOD3 a step further and describes interior details [21]. Beyond buildings, the LOD concept applies to the other object classes as well [20]. Considering data availability, higher levels of detail might not be available for every study area and therefore, extruding 3D information from LOD0 data has been investigated and methods to extrude buildings with no elevation information has been proposed [62] .

2.1.3. OpenStreetMap (OSM) and Crowdsourced Data

OpenStreetMap works in a similar fashion to Wikipedia in terms of community contribution to data and evaluation. Availability of cost-effective GPS receivers allow more and more users to contribute and import data into. Moreover, the users can import public-domain information found in other applications, such as public-domain Tiger information to better decorate and organize the OSM data. A user can then export a selected area from the map with all features embedded, which can then be used to parse and further use the data for specialized purposes [23].

OSM data is geographically represented data relative to WGS84 datum. OSM data has variety of point, line and polygon features, which can be selected as the study area is exported. These features can then be utilized in a meaningful way, for the purposes of the research. As there have been numerous different research topics utilizing OSM information, and there are multiple ways of mapping and describing data, it can be considered as a flexible map data source [53].

The point features describe places, points of interest, places of worship and several other important objects that a virtual environment may possess. Furthermore, line features define boundaries, roads and paths, railways, waterways and variety of other infrastructure related objects. It is vital to export both of these features to define road networks, important locations on the map and boundaries of objects. Lastly, polygon features define administrative areas, building outlines, land use, bodies of water and oceans [53]. Utilizing polygon features, residential buildings, farms, universities and several other vital objects can be extracted and visualized. Through the tags OSM data provides such as, “*amenity=police*” or “*landuse=cemetery*”, users can find detailed information about the study area and the data can be processed further to accommodate many of these features. Obviously, there may be certain efficiency issues trying to

visualize every single object type found in the dataset and hence, only a subset of features might be visualized at a time. Nevertheless, these are made available for the researches of varying fields of expertise to utilize and conduct research on different topics.

OSM data has been investigated for various purposes. Bright et al. [63] assesses OSM data for reliability for estimating spatial availability of alcohol. It is underlined that OSM data was sufficient enough to replicate findings in other works using alcohol licensing information. Huber and Rust [64] investigate possibility of using OSM alongside Open Source Routing Machine (OSRM) to find an optimal route utilizing different vehicles. In their work, it is mentioned that this approach outperformed a similar process handled through ArcGIS by a factor of 100. Li et al. [65] provide an implementation that incorporates emergency management efforts, OSM data and mobile applications. The implementation provides three layers of view, data interaction and logic control to supplement emergency logistics. Through these examples, it is clear that OSM can be utilized for various tasks and further specialized through incorporating other GIS concepts.

As applications and experiments based on OSM data are required to be well-structured and accurate, it has been a focus of research to investigate the completeness of OSM data. Such surveys are important to compare crowdsourced information with propriety commercial map data. As crowdsourcing utilizes people to provide information and context to features, it has a potential to provide more detailed information than commercially available map data [66]. Nevertheless, as data collection is not done primarily by professionals, it is expected to have certain inaccuracies or inconsistencies to occur [67, 68]. Furthermore, as the information organized for OSM is provided by locals, there may be several inconsistencies regarding localization. Therefore, it is important to provide country-specific training material for the users for OSM to be used in a serious

scenario, such as disaster management [69]. Nevertheless, many works indicate that OSM data, if complete, are suitable for use [63, 70, 71].

There are several issues to be considered in visualizing virtual environments. With the concerns of data availability and increased performance and usage of mobile devices, utilizing web and mobile applications to provide geospatial services became part of our lives. By utilizing web services and distributed geospatial information processing, more users can have access to the data, which in turn leads to more application being developed and utilized. Therefore, an efficient way of streaming the data must be considered for daily use of geospatial information [34]. Through utility of efficient streaming of geospatial data, users are able to incorporate their data sets into a collaborative environment to build more robust applications, such as OSM [23]. As web approaches have been recently investigated to provide GIS applications to a broader audience, multiple differing implementations provide variety in this field. Concerning practicality of such a web-based implementation, it has been proven that these applications are both portable as they are distributed through the web, and useful as web-based technologies offer efficient interaction with 3D information [72, 73].

Simple visualization of the environment may not be enough for several scenarios, such as time-critical tasks and emergency management. Real-time visualization is a vital approach to many applications that require immediate response to the changes in the environment and therefore, it has been investigated for decades. Performance has always been a concern for visualizing virtual environments with complex city structures and textures to use. Nevertheless, it is possible to render and display such complex scenes in real-time [74]. Visualization of such environments in real-time allow the researches to focus on the changing parameters as simulations proceed. There are various ways of utilizing simulations alongside a 3D environment extracted from map information.

For emergency management scenarios, flood modeling, population estimation, and other areas of interest, urban environments are helpful in viewing the information as it provides more context than mere numbers, or values.

Another major concern is the heterogeneity of the data sources [75]. As there are numerous ways of describing map data, there may be inconsistencies if multiple data sources are combined into one to produce a larger data set. As mentioned earlier, OSM is affected with this issue due to its access for the general public. As the professionals are not evaluating the data for consistencies, several issues may arise. Nevertheless, it is mentioned in the literature that there are ways to alleviate this problem by detecting the problems automatically [67], focusing on localized data more effectively and converting user tags into proper GIS features. Once these problems are resolved, multiple data sources can be utilized to visualize a complex environment for viewing, editing and simulating important parameters.

2.2. Flood Modeling and Visualization

Modeling a complex environmental phenomenon is always a challenging task due to difficulty of representation and accounting for numerous parameters involved in the process. Nevertheless, as our lives are constantly being affected by certain environmental conditions, scientists and researchers developed a variety of ways to account for these issues. Now, numerous models have been utilized in the industry and research, and innovative approaches have been proposed to provide both accuracy and efficiency in the progress. Alongside some advancements on the graphical developments, numerous flood models, both commercial and non-commercial are provided to help practitioners analyze the parameters and investigate the patterns that are existent in study areas.

2.2.1. Background

Considering the flood modeling and analysis, historical information is the firsthand source to be utilized. Although limited in terms of data, if historical information about a region is found, then flood risk analysis can be performed, and several assumptions can be made. Geologic evidence left behind due to occurrence of large floods enables researchers to collect and process information to analyze flood data [76]. However, many floods are not included in the written history and consequently, it is challenging and expensive to search for historical evidence to assess flooding [12]. For this reason, there are other models to provide insight about the study area and calculate the probabilities of certain conditions in the environment. Although not based on the historical flood data of the region, empirical and hydrological simulations of varying dimensions help determine possible risks and potential flooding of a region.

Empirical methods have been investigated to model flooding and how the local dynamics direct or divert flooding [77, 78]. These observations can then be utilized in emergency management scenarios to support decision making [79, 80]. Moreover, such models can be utilized to serve as a method of calibration for other hydrodynamic models. As empirical flood modeling focus on formulation of certain behaviors of water, it is important to collect sufficient information regarding the study area to be able to provide values to the system. Maximum flood discharge, rainfall intensity, catchment area and certain coefficients specific to the area being investigated are significant parameters for such formulas. Especially with the prominence of 1D and 2D hydrodynamic simulations to simulate flooding, empirical models serve as ground truth to provide a level of correctness, or utilized in calibration of other models [81].

Hydrodynamic models utilize a mathematical approach to solve variety of equation depending on the domain to represent the physical behavior of flooding or waters. Depending on

the dimensions utilized, commonly 1D, 1D/2D coupled, 2D and 3D models have been utilized. Traditionally, the focus has been with 1D and 2D simulations due to the complexity and performance concerns of 3D simulations. 1D and 2D models have been used in analyzing and modeling flash floods [82, 83], decision support systems [79] and hydrodynamic modeling [84]. Recently, with the improvements of modern GPUs and availability of data, 3D simulations and especially particle based Lagrangian simulations have been investigated in the literature [12]. In the next subsections, we will discuss about these hydrodynamic approaches more in detail. Furthermore, we will mention several significant flood models that have been used in the industry for specific tasks.

2.2.1. One-Dimensional Hydrodynamic Models

One-dimensional models can be used for variety of scenarios depending on the environment conditions and the domain 1D simulation is applied to. As the need for an additional dimension is not required, representation of flood water flow through storm drains and pipe network is possible with this approach [85]. These simulations focus on a single aspect of the study area and therefore, they have less power in terms of expressivity. However, they provide an efficient simulation in most cases and therefore, they may be preferred over more complicated 2D and 3D simulations [12]. Commonly, 1D flood analysis and waterflow simulations solve Saint-Venant continuity and momentum equations [86]. Although it can be considered as a simplified model compared to 2D and 3D hydrodynamic models, it still has high computational cost for representing flooding across a long river and wide floodplains [87].

Furthermore, there are hybrid approaches such that 1D simulations can transition to 2D simulations on demand [88, 89, 90, 91]. In such approaches, 1D features such as description of the water flow within pipes can be mapped to 2D grids to visualize the expansion, to describe the flood

plain. Consequently, the flood plains, which require expansion to be described utilizing more than a single dimension, can possibly be represented suitably with 1D/2D coupling [92]. It has been mentioned that the coupled model is useful for easily identifying the vulnerabilities of the regions in the study area [93].

2.2.2. Two and Three-Dimensional Hydrodynamic Models

2D models have been used in the research community for its accuracy in solving full shallow water equations [12]. By describing an additional dimension, these models are better in describing the flood progression and diffusion compared to 1D simulations. Consequently, when more information is needed and an additional dimension is useful, these models are utilized. Through the advancement of modern graphical processing units (GPUs), 2D and 3D visualization have been utilized in flood modeling and analysis. Teng et al. [12] introduce state-of-the-art empirical, hydrodynamic and simplified methods used in flood modeling and underline that hydrodynamic models utilizing 2D and 3D simulations may prove to be better in representing detailed flow dynamics. Furthermore, simplified models can be used best for probabilistic flood risk assessment. Leskens et al. [94] present a system for analyzing flooding scenarios through 3D visualization. Upon representing the virtual environment, numerical methods that solve 2D hydraulic equations for water flow are used to simulate the flood. It is noted that 3D representation of the flood helps practitioners to better realize and estimate the impact of a flood. Such surveys point out the power of 2D and 3D visualization in describing flooding.

Due to this power of visualization, many works proceed with various ways, simulation methods, and dynamic parameters to display the properties, which would be difficult to see otherwise. 2D and 3D models have been used for storm surge flood routing visualization [95], risk estimation [96], flood modeling and analysis [13, 15, 94], rainfall forecasting [97], landslide

simulation [98] and to visualize many other scenarios. Recent works show a move from static visualization to dynamic visualization [6, 99, 100], to provide more interactive and accessible simulation tools to incorporate virtual environments and dynamic simulation parameters.

There have been numerous works regarding flood modeling and analysis as discussed. As flooding is and always will be a risk factor wherever the humans live, it is worthwhile to investigate different approaches that would benefit analyzing and predicting flooding events beforehand. For such reasons, many of the numerical methods discussed earlier may not be acceptable for practitioners as visual information carry more context and possibly provide an understandable interface [94]. Therefore, it is suitable to consider 3D visualization approaches for flooding scenarios to enhance the study and possibly evaluate the parameters, which may not be immediately visible otherwise [12].

To represent flooding, proper physical representation of water is needed. Fluid modeling, in general, requires a detailed representation of fluids. Various approaches have been proposed to increase efficiency and to reduce redundancies while attaining an accurate simulation [24, 29, 30]. When visual representation is important, solvers mainly focus on solving Eulerian equations [101] or Navier-Stokes equations [102], which describe the motion of fluid particles. Several other methods including Lattice-Boltzmann methods [103] also work well for fluid simulation. Consequently, it is acceptable to consider different methods for flood simulation. Some prominent approaches in this category are SPH [25], position-based fluids (PBF) [24] and mesh-free vortex particles (vortons) [104].

Since the introduction of SPH by Gingold and Monaghan [25] and Lucy [26], it has been used in a variety of industries, including computer graphics. It is a mesh-free particle-based method based on Lagrangian formulation. As SPH is Lagrangian in nature, it is not constrained by a mesh,

which is its main strength. This sets it apart from grid-based approaches, which suffer from loss of mass and inflexibility due to discretizing the study area into grids. However, neighborhood is a vital part of SPH, and it is generally associated with its inefficiency for real-time simulation of large environments [30]. As the value of a variable at a particle of interest can be found by checking its neighbors, efficient algorithms of reducing neighborhood dependencies is vital. There are other approaches like position-based fluids [24] that improve some of the shortcomings of SPH. Nevertheless, SPH and its variants have been commonly used in 3D flood visualization [105, 95].

Utilizing open data to implement 3D flood visualization has been an ongoing topic among the community and variety of approaches have been implemented to demonstrate the power of open data. Herman et al. [106] provide a survey on open data sources and provide an implementation of 3D visualization framework using OSM data. In this work, it is shown that with no additional plug-ins or specialized software required, open data can be used to create a 3D environment for utility in emergency management and flood analysis. Although there are limitations to open data sources as underlined earlier, there is a big potential for open data if errors and inconsistencies are removed.

As depth, or water level is a significant parameter to estimate risk and damage, it is vital to be able to track the water flow while computing and visualizing water levels [107, 108]. Nevertheless, 3D representation may distort or complicate human-readability of the study area and therefore, it might be significant to provide 2D representation of flooding alongside 3D information [6]. 3D simulation methods can still be utilized in 2D visualization of the environment to describe flooding [109]. Although 3D representation may carry more in terms of data and offer better compression [6], 2D representation can provide better performance [12] as less data is considered.

2.2.3. Other Specialized Flood Models

There are numerous software packages, each offer a collection of models, to represent a specialized aspect of flooding. These packages provide up-to-date modeling approaches in their categories and provide various tools to calibrate the flood data. In this section, some of these models are briefly described, important features and their usage in the research community are mentioned. Although there are numerous models, only three of these models, Hydrologic Engineering Center's River Analysis System (HEC-RAS), MIKE FLOOD and LISFLOOD-FP are briefly introduced.

Hydrologic Engineering Center's River Analysis System (HEC-RAS), is a numerical model featuring an integrated 1D model to represent flow and provides an interactive environmental management scenario [110]. It has been used commonly for evaluating and visualizing river dynamics through performing 1D steady and 2D unsteady river flow calculations to estimate flooding [111]. Utilizing 1D aspect of HEC-RAS is useful as it can make sufficient predictions of flood extent when a high-resolution DEM is utilized to simulate the water [112]. Thol et al. [113] provide an analysis of this model to predict flood extent of Mekong River in Cambodia and report that the results are in agreement with the observations they gathered for the study area. Yakti et al. [114] reports similar results, utilizing 2D HEC-RAS model to simulate the overland flow for the case of Way Ela Natural Dam failing. The resulting flooding, depth and arrival time are verified through observations. Due to its efficient and accurate modeling of the environment, HEC-RAS can be utilized to represent and examine large floodplains [115].

Similar to HEC-RAS, MIKE FLOOD is a unique software providing a coupled 1D-2D hydrodynamic model [116]. It is underlined that this toolbox is suitable for modeling and accomplishing flood forecasting, flood management, flood risk analysis and hazard mapping,

visualization and evaluation of dam break scenarios and river and coastal flooding. Furthermore, this model is expandable through the modules available. By integrating certain specialized modules on the base software, more detailed representation with added features can be acquired. [116]. Patro et al. [117] utilize MIKE FLOOD for simulation of flood inundation extent and flood depth. Upon description of the environmental conditions of the study area and preparation of the bathymetry of the study area with a resolution of 90m using SRTM model, flood data is simulated. Coupling 1D and 2D aspects of MIKE FLOOD, and calibrating several parameters to fit the study area, it is noted that the results were satisfactory to represent river flooding investigated in their work. This model has been used to accurately represent the depths and velocities of flooding [118].

LISFLOOD-FP [119] is a 2D hydrodynamic model, which excels in visualizing and processing flood inundation efficiently for complex topographies. As it is a grid-based approach, it computes flooding depth per cell in each time step to visualize the floodplain, embedding depth information throughout. This method is a non-commercial research tool that has been utilized in a variety of institutions [120]. There have been numerous researches conducted with this model to investigate a variety of topics. Works of Amaranth et al. [121] and Andreadis et al. [122] provide implementations to assess risk of flooding in the study areas. Their results show that LISFLOOD-FP can be used for such scenarios to estimate risk. Neal et al. [123] provide an optimization for 2D hydraulic model LISFLOOD-FP, which boosts the performance of the method significantly. Such results show the usability of the model for complex scenarios given the performance improvements are achieved to provide an efficient simulation experience.

CHAPTER III: METHODOLOGY AND SYSTEM DESIGN

This chapter describes the three major aspects and contributions of this study. First, a 3D urban environment extraction and visualization approach, City Maker is discussed. This implementation provides a simulation-ready, 3D visualized data extracted mainly from crowdsourced OpenStreetMap (OSM) information. Through utility of OSM data and an elevation model, the environment with important features such as roads, buildings and landmarks are generated. Next, a 2D depth visualization approach through adaptive spatial resolution is described. 2D visualization allows the water levels of the study area to be tracked effectively and efficiently. As a supplement to 3D visualization, 2D information helps certain aspects of the area to be recognized. An intelligent transition between regular and adaptive grids provide faster visualization performance. Lastly, a 3D position-based flood framework is described with its components to visualize 3D flooding information of the study area. The flood framework provides flexibility to design and model a study area for the purpose of visualization of surface flow. With several important modules such as a local friction model, soil absorption and rainfall mechanics, our framework provides practitioners a tool for visualization of flood data.

These three different implementations can be utilized together to provide a smooth environment reconstruction, depth mapping, real-time flood simulation and visualization. Initially, City Maker helps extract the data describing the environment, including significant features. Furthermore, at this stage, simulation parameters are baked into the 3D model of the urban environment. Through utility of City Maker, a 3D model of the study area is generated to be further used in the simulation. 2D visualization of depth is used alongside the 3D position-based flood framework that is built to provide supplementary depth visualization. Figure 1 shows the intended workflow.

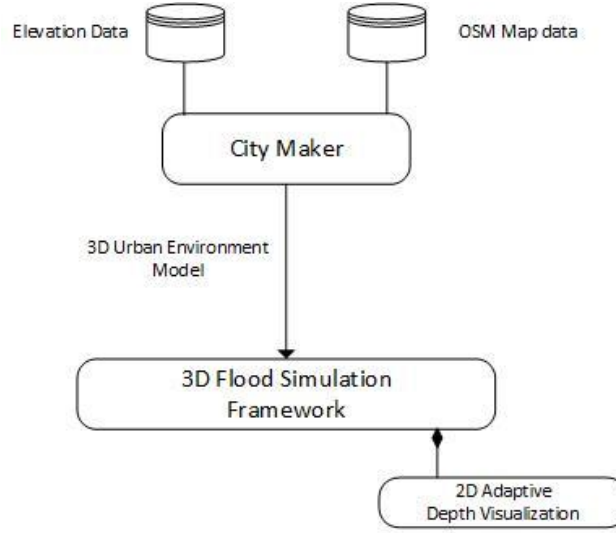


Figure 1. System overview.

3.1. City Maker: 2D/3D Urban Environment Reconstruction

The initial issue with this domain is to find an appropriate data structure to store geographical information. Once such a data structure is utilized, the information can then be parsed or extracted for further use. There are geographical data sources that can be found online and be utilized for a variety of purposes. These data sets, often specialized in a certain topic, allow us to visualize different aspects of the study area. Among these data sets, there are freely available and crowdsourced approaches for ease of access and manipulation. OpenStreetMap (OSM) [53] offers a crowdsourcing option to build a set of map data which can be altered by users and accessed freely.

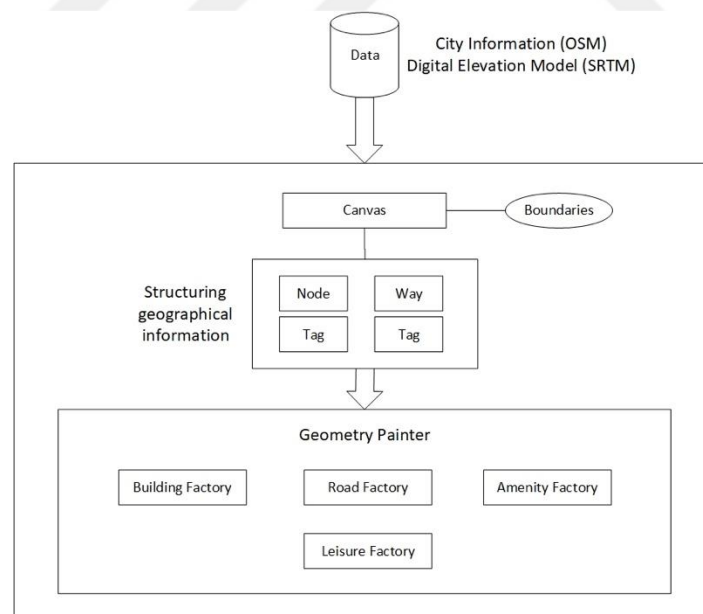
City Maker considers OSM data to provide an efficient way to visualize city information both in 2D and 3D. Through the utility of significant features, several basic geometric primitives can be used to represent the 3D environment for added efficiency. Moreover, color mapping is implemented to provide a more context in visualization of the study area. By repeating contextual colors organized in semantic categories, the features are better recognized, and further

customizations about visualization can be achieved. Lastly, by adding simulation parameters to each feature that the engine generates, the environment is exported for use in simulations, which provides an advantage over merely providing the objects for collision detection purposes.

The city building framework contains several layers of abstraction to accomplish boundary generation, feature extraction and 3D environment reconstruction.. To utilize the geographical information, the base map is provided by OpenStreetMap (OSM) and the preliminary geographical information is parsed. The study area can be exported from OSM website and any urban environment can be selected for use in the environment reconstruction process. Once the base map is processed appropriately, the data is enhanced with elevation information acquired from the digital elevation model (DEM) from Satellite Radar Topography Mission (SRTM), which has been a common elevation model for variety of uses [124]. The version utilized for this work has 1 arc-second, or 30 meters resolution, which is sufficient for broader description of elevation. As the data availability is a concern for certain parts of the world, a global elevation model such as SRTM is useful to account for remote locations as well as urban environments even though some precision is sacrificed. Detailed models for environments, if found, can be incorporated into the City Maker for precise representation of the study area. **Error! Reference source not found.** **Error! Reference source not found.** shows the overview of the framework and how its components relate.

Initially, a canvas is created to accommodate the 3D drawing components to build the environment. The canvas is the surface, or the empty area, that GIS features can be placed on. Once the canvas with its appropriate boundary parameters, which are longitude and latitude information that define the study area, are generated through reading the header of the OSM file, geographical information extracted from the OSM data is reorganized for further processing. This

structuring phase provides vital information to the 3D drawing component, which is responsible for drawing a variety of objects and surfaces to represent the study area. OSM data provides tags, which are map features, that are attached to data structures called nodes, ways and relations. As the OSM follows this hierarchical data format, the data structures can easily be organized by parsing the exported OSM file and reading through nodes, ways, relations and then, individual map features through the tags. Upon organization of the data structures for the map features, a drawing pipeline, which has the power to draw basic geometric primitives and create relationships between separate components, is utilized to draw them to generate the 3D environment. At this stage, vertices, normals, colors and any applicable textures are passed to the drawing pipeline, which is implemented as a series of factories to draw the geometry and dress the objects up. Triangulation is utilized to accommodate concave polygons and extrusion is applied to provide



sides of the features with a base and a roof, such as buildings.

Figure 2. Parts of feature extraction and drawing hierarchy of City Maker.

Once the drawing phase is completed, the simulation parameters are then added for use in simulations. Manning coefficients define surface roughness through coefficients, which can be used in describing free surface flow in flood simulations. These coefficients are acquired from Arcement and Schneider's paper [28] and calibrated for use within the flood simulator used in testing. As OSM data contains keys and values to define the type of features, our system uses this information to assign appropriate coefficients to features. Moreover, by utilizing storm sewer information, flood water absorption is defined for the 3D environment. For City Maker to properly generate sewer information, inlets, manholes and other relevant objects must be provided in a file alongside their geographic coordinates and absorption rates. Practitioners can further calibrate this data by providing a well-detailed description of each object, which would improve results obtained by the simulation. The steps in 3D environment reconstruction that City Maker considers is shown in Figure 3.

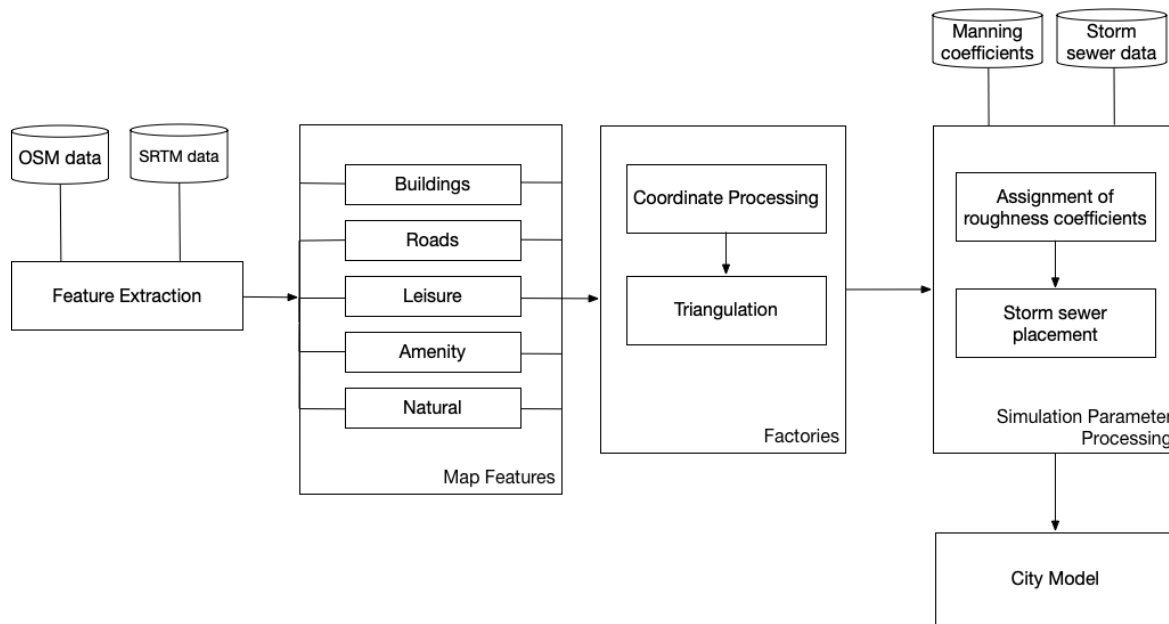


Figure 3. Processes City Maker utilizes to produce a city model.

3.1.1. Feature Extraction and OSM data Parsing

Feature extraction is the stage in which features from the OSM data are extracted and processed for the drawing pipeline. This process encompasses parsing the OSM data, restructuring the nodes data by converting them from longitude, latitude pairs into relative 3D environment coordinates and then passing the structured information to the drawing pipeline. As OSM data is structured hierarchically, the format is quite predictable. While nodes contain longitude and latitude information alongside some significant map features as tags, ways define a collection of nodes that describe certain constructs such as buildings. Moreover, relations define even bigger constructs by providing a list of ways to utilize. Nevertheless, with the inclusion of user-defined tags, some features may be inconsistent across different environments. Hence, it is an important task to manually read the exported OSM file of the selected region to remove any inconsistencies.

There are four major categories of components that the feature extractor considers: relations, ways, nodes and tags, as OSM data is usually structured. Firstly, a canvas is utilized to describe the city model and its boundaries. OSM contains this information as the data is exported and therefore, it is straightforward to incorporate this information into the 3D environment. After minimum and maximum longitudes and latitudes are stored as the boundary information, the feature extractor processes nodes and ways from the map data. Initially a vector is utilized to store all the nodes for further manipulation once the ways are determined. Ways are ordered sets of nodes that define entities on the map. At this point, tag information is read from these 'ways' to determine the type of object the drawing framework is going to receive. For example, a tag with a key 'building' and a value 'residential' defines a building utilized for residential purposes. Some of the buildings contain undefined functionality, which are indicated by the value 'yes'. Figure \ref{structure} shows how a multipolygon relation construct defining an area and its relevant

features is organized through ways, which contain multiple nodes that carry information alongside significant tags that define attributes. Furthermore, height information is associated with every entity the feature extractor determines. Height values are provided in meters unless indicated through notation otherwise. The extracted information is later sent to the drawing pipeline to visualize the study area.

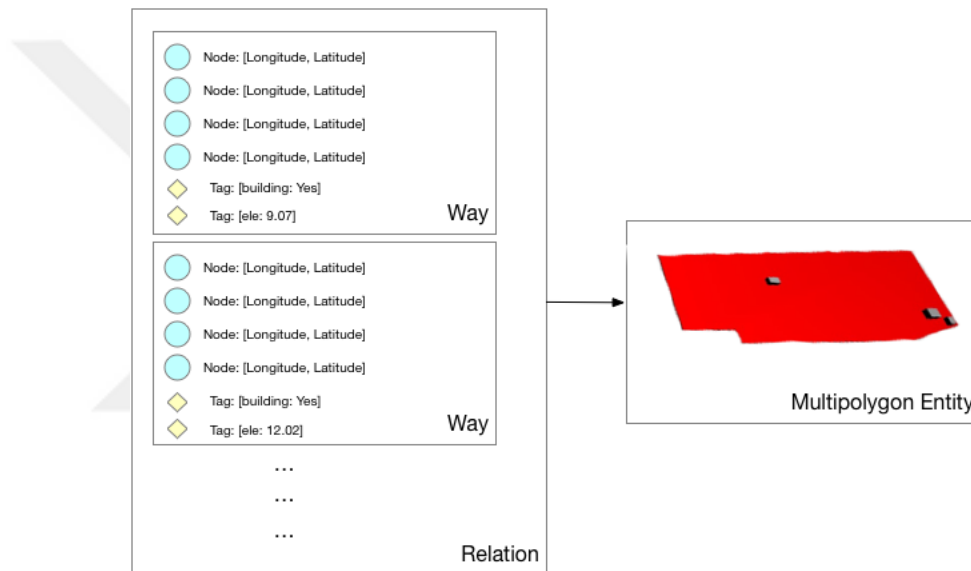


Figure 4. A Multipolygon construct defined by relations, ways, nodes and tags.

To convert the data to the appropriate format, features are evaluated using their geographical coordinates and then parsed to generate vertex coordinates as they are required to place the map features in the 3D city model. Normally, vertex positions alongside normal and texture coordinates are specified to format the objects appropriately. Furthermore, these object files can contain a variety of attributes to specify a particular rendering attribute, such as level of detail, material name, and ray tracing information. In addition, grouping can be specified to

combine relevant features together for a more contextual semantic organization. Currently, basic shading and color coding is available within the City Maker.

$$hav(\theta) = hav(\varphi_1 - \varphi_2) + \cos(\varphi_1) \cos(\varphi_2) hav(\lambda_2 - \lambda_1)$$

(1)

To compute distances between two points in the city model, haversine formula [125] is used. In Equation 1, ϕ_1 and ϕ_2 indicate latitudes while λ_1 and λ_2 indicate longitudes. Therefore, to compute the great circle distance between two points, which is dependent on locations of two points and the radius of earth, latitudes and longitudes which are extracted from the nodes within OSM data are used. The distances are stored to provide proper georeferencing as well as a user-readable context to the features; instead of showing geographic coordinates, knowing the distance of a building from the sea in meters could provide better insight in many scenarios.

Once the distances are realized, the component translation matrix is formed with relative distances based on the object's location from the boundaries of the study area. Moreover, this is the stage to organize the object data in a format that the drawing pipeline would recognize and process them. Buildings footprints are extruded with the elevation information, while roads are organized by structuring the nodes as flat planes and lines. These properties are stored in meaningful data structures to be passed later to the drawing factories. Certain features like roads require connection of multiple features and therefore, required rotation matrices are derived from the processed coordinates and then placed accordingly in the environment. Furthermore, SRTM information is exported and passed to the next stage to carry elevation information for the study

area. As the focus is on description of LOD0 and LOD1 details as specified by CityGML [56], basic geometric primitives are sufficient to generate both the 2D and 3D digital city models.

Of course, an important limitation in this approach is, as underlined earlier, that we considered a variety of common features and tags. Consequently, it would not yield the most customizable of results as most of the user customized tags that appear in OSM data are left out. Moreover, for practical purposes several tags are merged into one if the entities are similar in identity and appearance. A benefit of this approach is to have limited number of coloring, textures and relatively small number of ways to draw the objects utilizing geometric primitives, which contributes to the efficiency of our approach. It is possible to parse the OSM data further and provide additional features if required.

3.1.2. Drawing Pipeline and Visualization

Visualization of the city features is handled through a unique hierarchical drawing pipeline implemented for City Maker to accomplish efficient component construction. Geometric primitives are limited in number and size to proceed with efficient visualization while retaining most of the original structure of the study area. The lower lever pipeline is formed of component builders for significant renderable mesh to be drawn in the scene. These component builders are defined for shapes such as triangles, cubes, rectangular prisms, cylinders and several other geometric primitives to create likenesses of features within the study area. The components are created in a hierarchical fashion to properly relate parts of the features. For example, the roof of a particular building must be associated with the main body and therefore, body is considered as the root object while any other features like garden or roof are considered as neighbors of this root node.

As the structured data from feature extraction is passed to the pipeline, the drawing pipeline organizes them according to a specific order. Initially, certain areas, such as waterways and parks are visualized to provide the base layer of the environment. At this level, elevation information is also incorporated into the system to provide more details for the environment. Next, smaller and concrete features, such as buildings and landmarks are visualized. Considering basic geometric primitives and colors, these features are placed in the 3D environment. Lastly, roads are drawn to provide additional depth to the visualization. As the roads are placed directly on the canvas or other map features, they are drawn at a later stage. Upon drawing and coloring these objects, simulation parameters in the form of storm sewer data and roughness coefficients for the features placed in the 3D environment are carried over for use in simulations.

Areas are, in implementation, textured planes that represent different types of materials such as grass or water. OSM data contains tags for such areas and thus, it is easy to differentiate between different kinds of surfaces. The default surface is set as concrete as this work focuses on visualizing urban areas and consequently, roads, residential and commercial areas are expected. To draw areas, extreme points of the surface is analyzed and required tessellation parameters are set. Appropriate tessellation is needed for incorporating detail into the plane. As elevation information is extracted in the earlier stages, this data must be incorporated into these planes so that height information can be visualized. Upon checking elevation and analyzing the required level of detail in a given area, tessellation is performed, nodes are placed on the plane and height is visualized.

Roads are built directly on the canvas. Depending on the outline of the environment, which can be directly recognized through OSM tags like waterways, highways and so on, different types of planes are generated. Roads are drawn by considering start and endpoints of each individual

road and utilizing lines to draw lanes, junctions and other relevant information. Although our study does not focus on determining the number of lanes each road may have, it is possible if the information is available. This could possibly be improved through extracting such features and implementing additional steps in the drawing pipeline. Sometimes nodes carry such information in OSM data and therefore, further processing can be done to extract such information.

Buildings are drawn based on their dimensions and using the height information derived from OSM data. Height information is already embedded in OSM data as meters and therefore, it can be utilized after appropriate conversions from meters to vertex coordinates. Length and width information is acquired through computing the distances between extreme sides of the building. These distances are gathered by checking through the list of nodes each building contains. These nodes contain longitude and latitude information. With appropriate conversion to simulation environment units, the features can be placed in the environment with no issues.

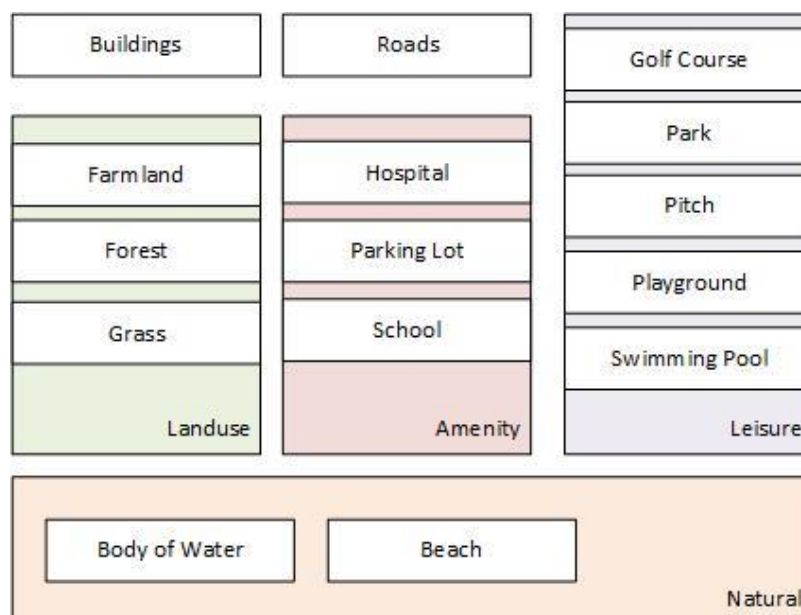


Figure 5. Features currently implemented for the City Maker.

The features shown in Figure 5 have their unique ways of drawing, texturing and 3D model selection. The way they are triangulated and placed on the surface of the model is defined within each factory in the drawing pipeline. For example, waterways are defined to be level with the base elevation data and provide a blue polygon in the area indicated by the coordinates, while hospitals define areas that contain hospital features such as buildings and parks. Depending on the extracted features and availability of data, more information can be incorporated in the drawing phase. For example, some buildings contain a height information in terms of meters. This is helpful in drawing the buildings yet not all buildings have this height information embedded. Consequently, some simplifications and assumptions may need to be made in the drawing phase. One major assumption is that the study area may not provide sufficient height information for all the features to be drawn in 3D. For this purpose, our system considers only the available height information in the surrounding area, averages the height, and then assigns a slightly randomized height parameter to the features with no height data. If none of the features provide height information, then the user can select an arbitrary height value for the features to use as default height for rendering purposes.

3.1.3. Texture and 3D Model Mapping

Textures are organized according to the entity type as the skinning process requires knowledge of the entity for appropriate properties to be utilized. Diffuse textures are arranged through creation of required surface textures depending on the study area, such as water, grass or concrete. Moreover, as a mechanism to avoid repetitive visuals, variants of the same textures are produced and consequently, a single texture may have more than a dozen variants. As mentioned earlier, buildings, roads and other entities have parameters that indicate if they should be drawn with more than a single geometric primitive and thus, multiple textures may need to be utilized to accommodate the drawing requirements of the entities on the scene. Furthermore, this

implementation defines a randomization factor to provide a variety of objects to draw while avoiding repetitive object placement. Through localization, which allows for unique types of objects depending on the geographical location of the study area, and randomization factors implemented for this purpose, an approximate environment with significant features is generated. Illustration of this selection process is shown in Figure 6.

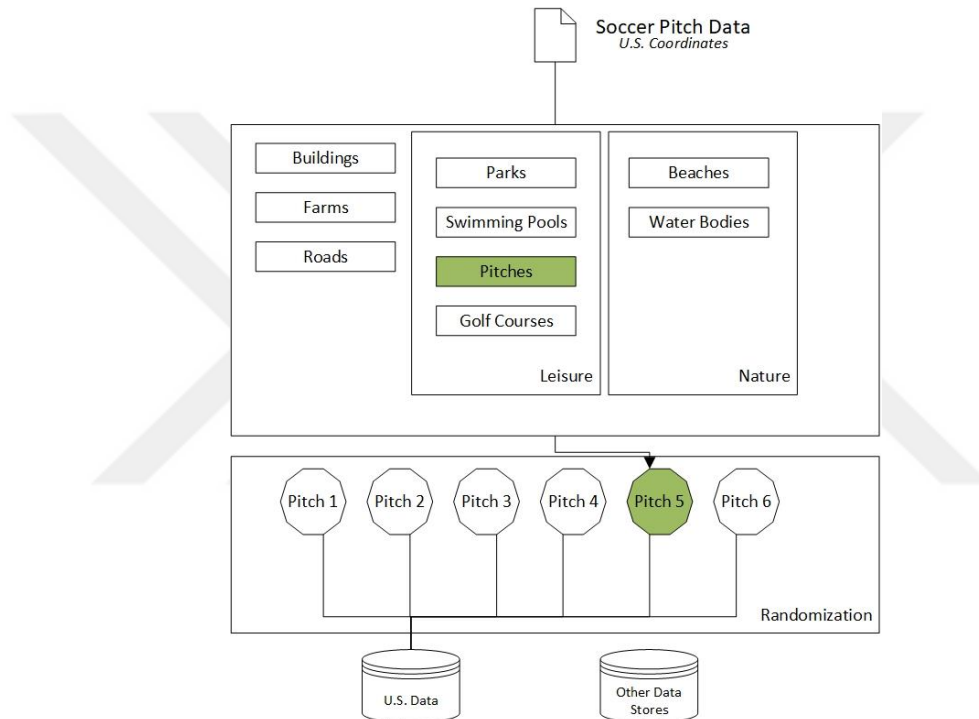


Figure 6. A scenario that data stores are utilized to select an object to draw.

Figure 6, shows the selection of colors and textures work for a soccer pitch. At an earlier stage, when the OSM data is initialized, the coordinates are used to recognize the localization information and therefore, in this example the U.S. data store is used to select features from. After the U.S. data store fetches available pitch models to place to the environment, a randomization is applied, as indicated earlier, to prevent repetition and an object is selected. This object is then placed on the terrain using the OSM data coordinates, feature width, elevation information and

height. As it can be seen in Figure 6, there are numerous object categories, each corresponding with features that can be found in the OSM data specification. Although several categories and features are designed to be used in the City Maker, OSM data support can be improved through provision of additional relations, types of features and custom tags. Object placement is handled by evaluating the terrain information and determining the terrain type. If the terrain is suitable for placement, such as the case of land, concrete, and similar solid surfaces, then the object is placed on the terrain.

A limitation of the City Maker in its current state is the lack of 3D models to be used in the randomization process. Although the underlying mechanism is implemented and the objects are dynamically selected, there are only a few distinct models per feature that could be allocated for the City Maker. Additional objects should improve the state of randomization and unique environment generation. Furthermore, the building structures, except 2D/3D wireframe modes are not retained due to limitations of the simulation environment. As the simulation environment requires a simulation-ready mesh to interact with particles, formatted object files and polygon files are utilized and loaded to the scene for interaction. Hence, the simulation engine utilizes parts of the City Maker and then implements its own workflow through particle generation and fluid interaction.

3.1.4. Post Processing and Addition of Simulation Parameters

Once the components are drawn and textures are mapped, the object hierarchy is provided for the users of the environment for quick access to features of interest and to provide a structured description of the study area. This feature potentially allows for addition and subtraction of environment features and therefore, it is possible to provide a fully dynamic description of the study area. For testing purposes, additional structures can be placed in any location and effects of

such changes, whether it is for traffic congestion or flooding can be evaluated. Although, if the purpose of the application is to simply generate a static description of the study area to be utilized as a static mesh for the simulation, then this feature simply provides ease of access to already existing features. As mentioned earlier, Figure 5 shows a list of available features that can be utilized in the City Maker.

Aside from 3D environment reconstruction capabilities, City Maker has the ability to embed simulation parameters into 3D city model it generates. At this stage, the system analyzes the city model built from the OSM data and incorporates simulation-related information. This stage allows for further customization of the simulation environment through incorporation of approximate simulation parameters for realistic simulation behavior. If the data is available, then it is possible to incorporate any simulation parameter to use in real-time simulations, yet it would not be necessary for the purpose of this work. Therefore, calibration has not been performed for different soil types, different surface textures and different road types. Nevertheless, global default simulation parameters are defined and configured for each entity and if needed, could be changed by the designer and therefore, it can accommodate different types of simulations.

Initial simulation mechanic considered is the roughness coefficients, or Manning coefficients. These values define the behavior in fluid simulations to restrict the movement of particles by defining certain friction values for the surface. City Maker, being aware of the Manning coefficients [28] for each feature placed in the environment, embeds these friction coefficients into the city model as a separate layer, allowing the model to be directly inputted for environmental simulations. This approach later benefits efficiency as well since no analysis is required to determine terrain roughness once these values are initialized at this stage.

Moreover, City Maker contains an implementation for storm sewer data. Through a storm sewer data file, the information is parsed, and the environment is assigned certain absorption coefficients to simulate water leaving the environment. To provide storm sewer data, the file must be prepared in a certain way to contain latitudes, longitudes, flow line elevations and similar relevant parameters. Once the data is processed, appropriate coefficients are provided about sections of the study area and then embedded into the city model as a separate layer. These coefficients define the absorption power of different sections within the city. For example, part of the city with many storm sewer inlets would absorb flood waters faster than some other part with no inlet. Consequently, it is vital to define this parameter if simulation of flooding would occur in urban environments.

Furthermore, whenever the city information is processed once, it is exported into an application-specific format for reuse. An extension of .cm (City Maker File) is considered for this purpose and provided details about object placement, texture types, coloring schemes and simulation parameters for easy display and manipulation. A sample of this file type is provided in Appendix B. Using the City Viewer implemented to work alongside City Maker, previously embedded information can be viewed, certain parameters can be modified, and different coloring schemes can be selected to refine the viewing experience. This approach provides a performance improvement over repeatedly parsing the OSM information whenever the city information is to be viewed. The performance results are shared in Chapter 4 and 5.

3.2. 2D Visualization of Flooding Through Adaptive Spatial Resolution

Considering different approaches of describing and modeling flood, it has been a very important task to create numerous models to visualize flooding. Such techniques can be used either to optimize processing for performance, or to increase detail to see more information about the

study area. As depth, or water level, is a significant parameter to estimate risk and damage in various works such as [107, 108, 126, 127], it is vital to be able to track the water flow while computing and visualizing water levels. Such visualization provides insight as to how damaging flooding might be. Here, since position-based fluids are utilized for simulating flood, particle clustering provides a straightforward computation of flooding and depth. On top of the simulation engine, a mapping engine is implemented that can keep track of the flood, while generating a heat map of the study area. As the simulation takes place, the dynamic mapping engine can decide on representing the flood using a regular grid or an adaptive grid. The dynamic approach helps with accuracy on demand, while keeping the simulation efficient when the situation arises. As quadrees are used to represent the adaptive grid, the engine also tracks the number of cells and depth of each subtree. Initial parameters regarding maximum number of cells, rows, columns and maximum depth are derived from the boundaries of the study area. Depending on the size and the shape of the environment, appropriate values are determined.

Our mapping engine, which has the power to decide the method of representation considering environmental parameters and computational expense, provides an efficient and optimized approach to visualize water flow for flood analysis. Furthermore, our visualization approach provides a framework for mapping the depth of any 2D or 3D Lagrangian, particle-based water flow simulation, which is important for flooding scenarios. The engine provides a flexible mapping experience by utilizing Jenks natural breaks for classification, and coloring the generated depth map with predefined single-hue and multi-hue color schemes to enhance readability of the depth map [128]. As rainbow maps have been discredited in a number of articles [129, 130] and better ways of visualization for the human eye are already discussed, such approaches are used to visualize the water depth.

3.2.1. System Design

Our system consists of multiple layers for better modularity. Initially our scene descriptions are read by the scene builder, which is responsible for initializing parameters to start the simulation. After the terrain data is analyzed and a height map is produced through relative height differences from the vertex coordinates and the surface baseline, simulation begins. As the fluid travels on the terrain, the depth analyzer is executed every second to evaluate the particles and draw the map of the environment. Depending on the performance requirements, this value can be changed parametrically to refresh the computation at a higher or a lower rate. Figure 7 **Error! Reference source not found.** shows an overview of the system layers.

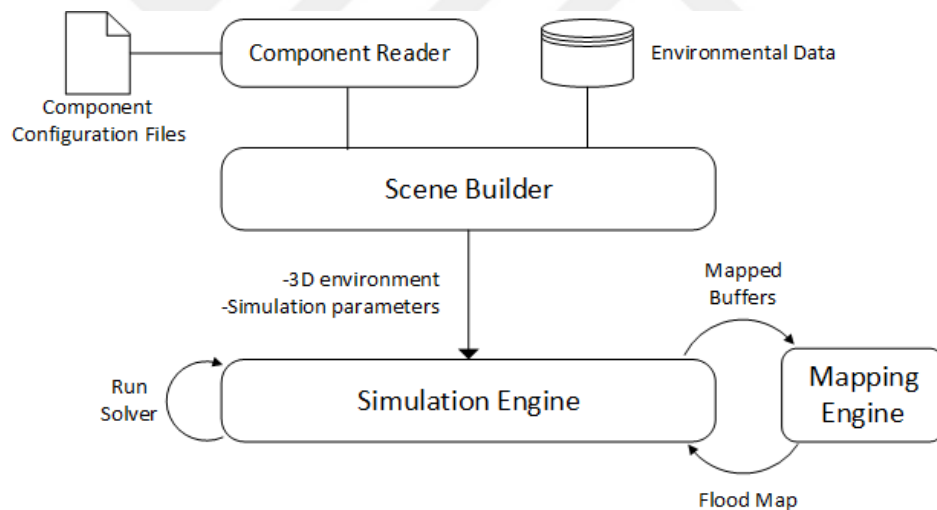


Figure 7. Overall system design for 2D visualization and mapping engine.

As the system focuses on optimizing performance, it executes particle mapping and depth analysis at certain intervals. The depth map of the environment can be produced using both fixed-size cells and quadtrees and therefore, depending on the environmental parameters and the study area, the system is able to choose an appropriate method of representation. Once the depth map is generated, the mapping engine draws a heat map using Jenks natural breaks classification method.

This map can then be sent as an output for further evaluation or it can be viewed as the simulation is running for the purpose of evaluating the fluid flow. The latter approach is useful for producing a series of still images for later evaluation of the simulation environment, which may be beneficial if real-time visualization is not required.

3.2.2. Component Parsing and Scene Processing

There are several components that are required for the mapping engine to properly analyze the environment. The component reader contains a set of tools to read necessary data files and place them into a container so that they can be passed to the next phase, which is the simulation itself. There are submodules that parse different types of files, such as cloud animation data, environmental configuration data, and object configuration data. Each of these files provide necessary information to simulate the environment properly and to pass configuration information to utilize in depth computation.

In the earliest stage, a scene description is provided to the component reader. The component reader then executes several commands step-by-step to load the data and arrange the definitions into a meaningful description. Scene description initially sends messages to load the environment mesh and texture. These files are stored in the data directory to be utilized throughout the application. Depending on the scenario and the utilization of the environmental objects, the environment can be scaled. This is especially useful for making evaluations regarding simulation by increasing the terrain and particle sizes. Moreover, as different modeling tools exported objects, point cloud data and similar formats may yield different orientations, a rotation matrix can be applied to transform the environment. Furthermore, a simulation speed coefficient is determined to vary the speed of the simulation. This is useful for speeding up the simulation for testing purposes. Nevertheless, as the rate is increased, physics computations must be taken care of as it

might sometimes cause the simulation to lose its stability by redrawing the environment too fast. Therefore, balancing speed and processing is suggested for stable and smooth water flow simulations. At this point, the scene description also provides commands to load cloud configuration and object configuration files. Figure 8 shows various components and parameters they initialize.

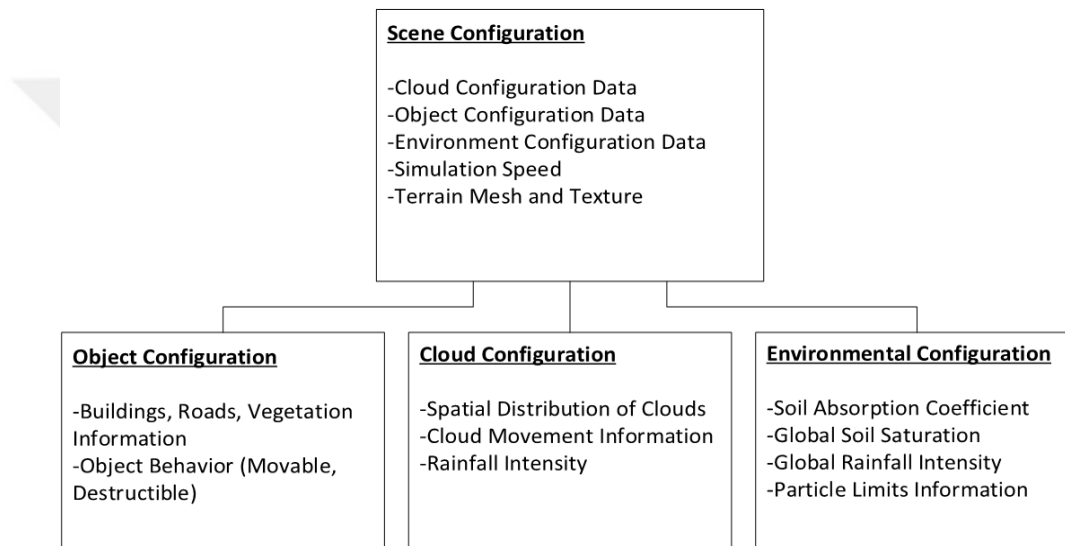


Figure 8. Parameters each component reader process

Environmental configuration data includes several vital constructs, such as soil absorption coefficients, rainfall intensity and particle limits. These parameters are utilized in the simulation process and therefore, are packed into a structure to be passed to the simulation engine. Preparing the environmental conditions for our work, a soil saturation of 100 is considered, providing the value in percentages, indicating that the soil is fully saturated and thus, surface runoff is computed accordingly. Nonetheless, providing this parameter can set a global soil saturation percentage lower than 50%, requiring the soil to be saturated first. As the topic of soil is a large topic by itself, for practical purposes full saturation is considered. Global rainfall intensity is defined if the scene

description defines a 'background rain', which indicates rainfall throughout the entire scene with a constant rate. Otherwise, it can be considered void, as clouds define differing rainfall intensities at different locations in the scene. The environmental configuration also provides a hard limit on particle count if required. This hard limit is utilized for performance purposes and therefore, is important if real-time performance is demanded.

Next, object configuration data is read and if needed, certain changes are done to the scene. In this phase, external entities, such as buildings, cars or trees with their simulation parameters can be imported into the scene. These objects, for consistency purposes are derived using OpenStreetMap (OSM) format. Hence, it is not possible to create user-defined types of objects. Bearing this in mind, the user can define buildings, roads, intersections, vegetation, landmarks and many other objects as required. These objects have their own unique properties, such as latitude, longitude, height, building type, and many other attributes. Normally these are exported directly from OSM into the simulation, yet if a more precise definition of the surroundings is required, or a user-defined area is represented, then this method yields appropriate results. Upon importing these entities, the simulation terrain can be merged with these objects. If that is the case, water does not take into account local friction coefficients, movability or destructibility of the objects; the objects are simply treated as part of the terrain. On the other hand, if the objects are set to be treated independent from the terrain, then simulation considers this and computes flooding accordingly. If not requested, the objects stay on their own with no connection to the terrain, simply used in diverting the water.

Lastly, cloud configuration data has information about cloud positions, rainfall intensities and their potential travel path. Cloud positions can both be provided in a relative format, utilizing vectors, or latitude, longitude and height format. In either case, the clouds are initialized to leave

rain in the amount that is defined as their rainfall amount and intensity parameters. A dynamic description of the environment allows for more interactivity and therefore, the cloud configuration files have an animation section, in which keyframes can be placed and matrices can be provided to define a dynamic scene. This way, by altering latitude, longitude and height tuples, and providing the time parameter to simulate clouds moving, the environment is considered dynamically. This is especially useful if the purpose is to track large sections of clouds which may leave heavy rain. If such animation keyframes are not provided, then the environmental description is considered as complete and the environment is set to process rainfall considering static locations for the clouds. This approach is used to test resilience of certain sections within the city by simply 'making it rain' and seeing the outcome as the simulation progresses.

Once the component reader processes the files, these structures are then sent to the simulation engine. Before any simulation begins, a height map for the environment is generated to be used for computations at a later time, either for visualization of the terrain information or for computation of the submerged area under water. A height map is utilized alongside a depth map for computations. Consequently, a limitation of this approach is that the terrain map is not regenerated with environmental alterations; the terrain is considered as an immovable object and thus, it is not subject to change. This prevents dynamic translation of small objects like rocks, debris and other possibly useful terrain objects to be considered as static objects. Naturally, this comes with an advantage of processing efficiency as the height map is generated before the simulation begins and is never recreated. Throughout the course of this work, erosion and surface deformation were not considered and thus, they are not integrated into the simulation. Figure 9 shows a sample height map generated using 40x40 regular grids and the environment it was extracted from.

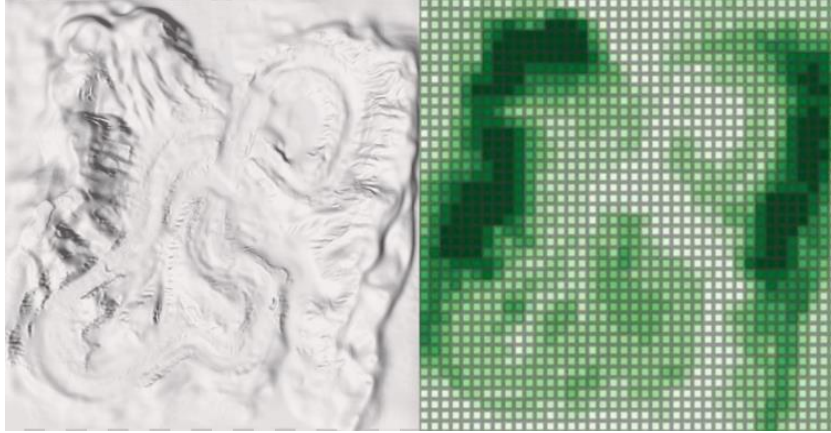


Figure 9. Digital model and the height map generated from the surface.

At this point, it is worth underlining that the height map information is not yet passed to the mapping engine and therefore, it is just stored as a structure within memory. When the mapping engine requests the data, the structure is passed and utilized by the engine, whether it is for visualizing the height of the environment, as shown in Figure 9, or just to utilize the data for calculating water depth. Visualization of both the depth information and terrain might be requested. Then, the height map structure is passed to the mapping engine to be drawn with the mapping parameters provided. On top of this, height map can be used to calculate depth information by considering water particles and estimating submerged land. Nevertheless, if this is the case, visualization is not required, and height map is simply used as a data structure in calculation. Moreover, as height map is passed to different components within the simulation environment, it can be reused as the simulation is running, making the process more efficient. If requested, analysis can be performed to extract the updated height map, which might be useful when the movable objects are included in the virtual environment.

3.2.3. Depth Mapping and Adaptive Resolution

The implementation of the mapping engine and algorithms utilized to generate the maps are considered with an emphasis on performance and accurate visualization. The engine is able to

recognize and reason about the environment that it is mapping through analysis of several environmental parameters. Such parameters like the size of the environment, clustering of water bodies, rainfall parameters and particle limit help the mapping engine select the most suitable way of visualizing the environment. Upon evaluation of these parameters, an appropriate representation is selected, and map information is passed back to the simulation engine for viewing purposes. Additionally, the engine can process and extract the map data if requested for later viewing. Hence, it is possible to run for a certain amount of time and extract screenshots per predefined interval of time.

The mapping engine considers two major representations of discretization: regular and adaptive grids. Depending on the environmental parameters and the way the fluid behaves in the simulation, the engine is able to switch from one representation to the other. For performance purposes the engine is, if instructed, able to make changes to the maximum number of cells it is utilizing to make the simulation run faster. Nevertheless, if performance is not a concern, the mapping engine can use as many subdivisions or cells as needed to represent the environment with more detail. As quadtrees are used to implement the adaptive grid, the engine also tracks the number of cells and depth of each subtree. Such tracking is also useful to color the map according to the depth of subtrees to recognize where the particles are clustered. The initial parameters regarding maximum number of cells, rows, columns and maximum depth are derived from the boundaries of the study area. Depending on the size and the shape of the environment, appropriate values are determined. These two approaches are shown in Figure 10.

Regular grids are commonly used where the fluid particles are stretched across the study area. These grids help the system focus on each region equally, which may visualize the bodies of water more accurately and precisely. Although it is wasteful in terms of storage and in some cases

not as precise as a limit is applied to the maximum number of cells, it is valuable when the study area receives fluid particles in its entirety. Recent works show that adaptive grids are beneficial in terms of storage for flood visualization purposes, regular grids are still easier to represent equally or maps with unified particle distribution. if detail is sought, then cell limitations might be lifted, and a full detailed regular grid can be drawn. Although this would be costly in terms of memory utilization, it might be negligible if enough memory is provided. Moreover, under certain conditions, computing the depth of each node within a quadtree might not be the best option computationally.

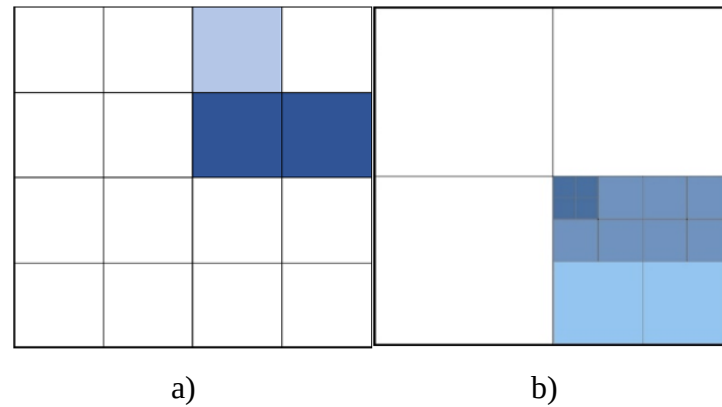


Figure 10. Area representation using a) regular grid, b) adaptive grid.

The engine prefers this algorithm when the study area is small, fluid gets distributed across the entire map, or a body of water, such as a lake or an ocean, is defined as the main terrain of the scene. Once a regular grid is selected, the engine tries to minimize the number of stuck particles from being included in the computation and therefore, it removes particles located below the reference point, which is provided as an input for the simulation engine. As the simulation can be performed on a large area, it is worthwhile to determine a reference point either to indicate water levels or at least a point of no importance so that the simulation efficiency is maintained.

Adaptive grids have their own advantages in terms of representation. Firstly, their nature enables the engine to optimize storage and increase the speed of fluid visualization on demand. As areas that are not flooded are not considered in visualization, it might provide some performance benefits. This approach is best utilized when there are specific regions with significant fluid interaction in the study area, while the rest remains relatively stable. When such clusters occur, the engine uses adaptive cells to describe the areas with fluid interaction similar to regular grids with little or no loss of precision in representation. Therefore, it provides better performance due to stable regions not being discretized. Moreover, as indicated earlier, the engine is able to make a transition between the two approaches and change the level of precision depending on the number of particles and simulation performance, to make the mapping process execute faster. Utilizing these environmental parameters, the mapping engine can continuously transition between these two representations unless it is instructed to remain fixed.

Mapping engine transitioning between two representations is handled through a rule base designed to recognize environmental conditions. Size of the study area, environmental parameters such as main terrain object being solid land, simulation performance in terms of frames-per-second, water particle distribution and cover ratio considering the overall study area, are all significant parameters that the engine considers. For example, if the mapping engine recognizes a significant drop of performance from 55Hz to 32Hz, it makes an attempt to help alleviate the situation by switching to adaptive grids. With this, an initial frame rate spike is possible due to release of some resources, yet, it provides performance benefits in the next frames as discussed in Chapter 4. Upon evaluation of the performance metrics and environmental parameters, such as terrain type, distribution of flood particles and total number of particles on the scene, at each simulation second, a preference value is generated. With this fuzzy approach, the engine derives

the suitability value for regular and adaptive grids and decides which representation to utilize. If the suitability values are too close, for example, 100 and 110 for regular and adaptive grids, the engine skips making the decision for 60 frames. Once it recognizes a significant difference, which is 25 in its current implementation, a transition occurs. Once the decision is made, the next simulation second is visualized using the selected approach and the engine further evaluates the parameters for the next second. This process continues unless the mapping engine is turned off, which can be done dynamically at the time of simulation, or the engine is instructed not to make transitions.

Depth values are estimated through two different approaches. A relative difference method is used on demand, to calculate the difference between the height of water and the height of the terrain submerged in water. The height values are computed considering a reference point, which is passed as a scene configuration parameter. The engine calculates a depth value for each grid cell and then passes the depth map as the input for the next stage. Furthermore, the engine can compute the depth map using particle count as the main indicator of flooding. With this approach, the engine counts the number of particles clustered in each cell and then produce a volume-based depth value. Upon computation of the values for each grid, the depth map is passed to the next stage. These two implementations are vital for the mapping engine as multiple representations are utilized. Quadtrees benefit from volume-based approach as the depth value is closely related to the depth of each subtree. As a performance optimization, the system considers all objects except fluid particles as immobile and consequently, it cannot track soil erosion or object translation from one point to another. Figure 11 shows the visualization of these different approaches.

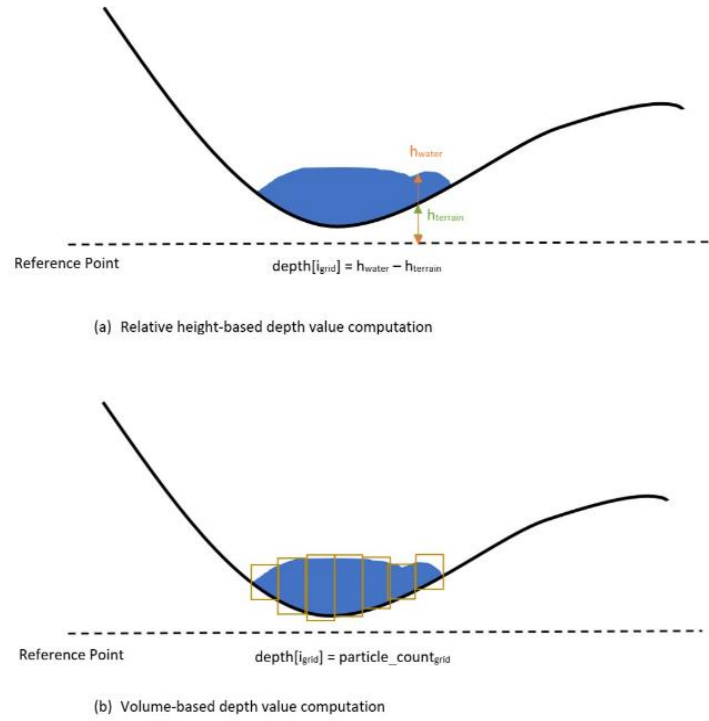


Figure 11. Computation of depth using a) relative height differences, b) fluid volume through number of particles per grid.

For proper depth computation, several conditions must be kept track of. The fluid particles, as they are dynamic, update their positions, velocities, and other parameters depending on the environmental conditions. Before any attempt is made to compute depth, these parameters must be updated, or mapped so that the current values are taken into account. Therefore, with a specified interval as parameter, the particles are mapped to their current state and then the data is passed to compute depth. This process can be executed per second or several milliseconds. For performance purposes, it is best to make rare updates but for real-time tracking, this may be impossible. If the demand is to map once in every few seconds, snapshots of the study area can be extracted for later use. This is useful if real-time tracking is not a concern. If tracking is required, then the system finds an optimum refresh rate to keep the simulation running smoothly.

For more complex scenarios and environments, if real-time refresh rates cannot be achieved, then screenshots of a given environment can be extracted providing the interval. In this approach, it is possible to extract state changes of the environment for longer periods and as the mapping is not done in real-time, performance issues do not arise. Extracted data structure can then be loaded into the environment viewer, and visualized for further analysis of the flooding, or flow. This is also important for testing and debugging purposes as it is possible to capture every simulation second using this approach. Catching these visual anomalies and inconsistencies might not be possible using real-time simulation.

3.2.4. Classification and Coloring of the Depth Information

The stage after computation of depth maps is the classification of depth data. An approach like equal interval could possibly work if the system is sure to have plenty of fluid particles all around the map. This, however, is not the case for many applications that require tracking fluid translation. Thus, implementation of Jenks natural breaks method is helpful to provide a meaningful representation of the study area. As the spatial distribution of rain differ greatly for larger study areas, using natural breaks would allow us to visualize the data in a more appropriate way. Same is correct for visualizing terrain height as we cannot expect to have equally distributed height values within the map. The mapping engine considers 3 to 9 different classes and therefore, by passing a parameter, various classification schemes can be evaluated and an appropriate one for the study area can be picked and utilized. Figure 12 shows varying classes of natural breaks used in representing the test area.

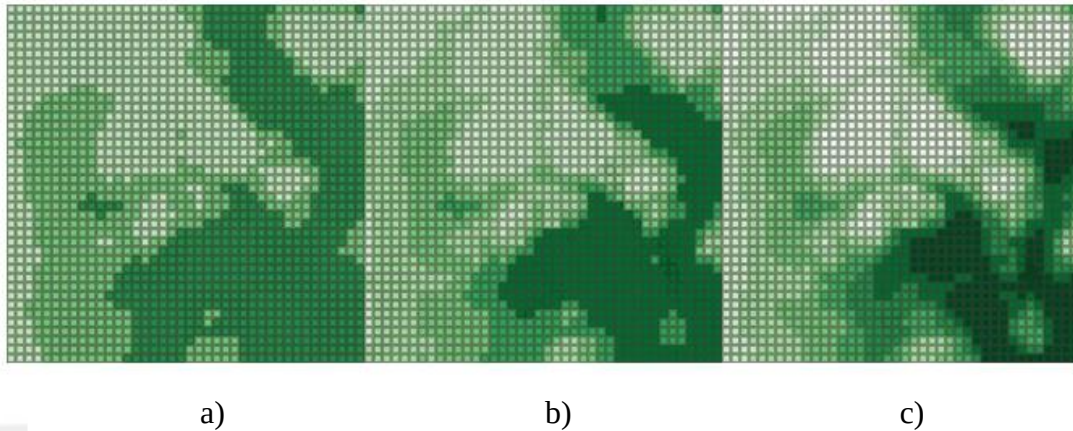


Figure 12. Height map generated using a) 4 classes, b) 6 classes and c) 9 classes.

Once the classes are organized through natural breaks method, the depth map is visualized. There are certain predefined color gradients that the system can receive as input. ColorBrewer [131] is used to export the color information and various schemes are predefined within the mapping engine. Since it is indicated as a bad method of representation [128], the rainbow color map is not included in the available color sets. On the other hand, numerous other coloring schemes are included in the system and users can select the color scheme they prefer. These coloring schemes can be used to describe the terrain or the flood. Figure 13 shows the multi-hue and single-hue coloring options the mapping engine has. These coloring schemes can be used for 3 to 9 classes of classification and therefore, various color schemes are predefined for each option. Depth map is initialized with the single-hue blues coloring option as it makes sense to provide varying shades of blue for depth information. Nevertheless, the color may be altered, or a multi-hue pattern may be selected to represent the same area. These multi-hue schemes are also built for 10, 11 and 12 classes yet, practically, these classes do not provide much sense as the mapping engine is predominantly used for depth mapping; varying colors from purple to yellow does not give much context in terms of depth.

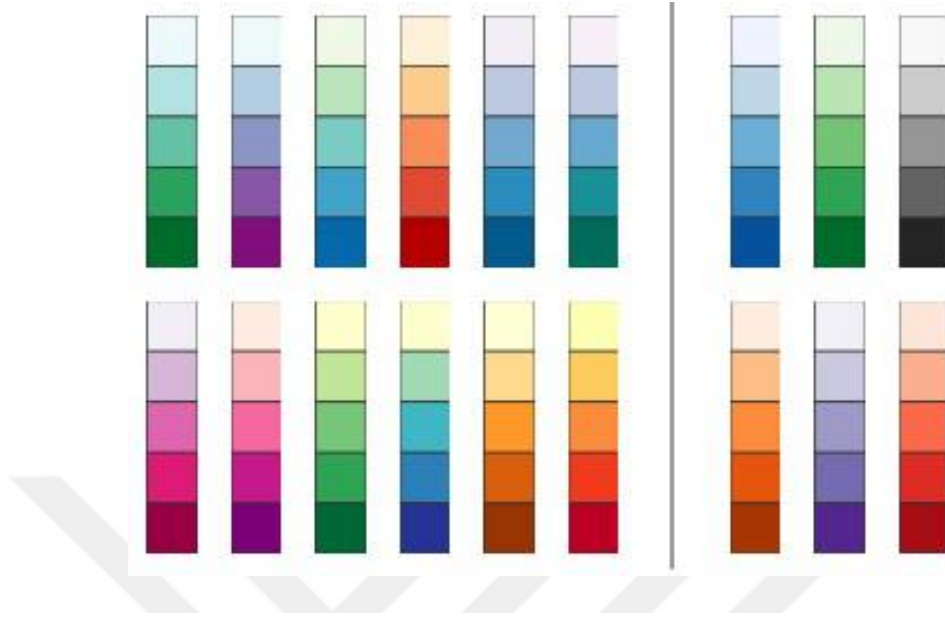


Figure 13. Multi-hue and single-hue map coloring options.

The visualization can be viewed in real time, for real-time rendering and tracking or stored for later evaluation, especially for large and complex environments. Additionally, depending on the parameter provided to the environment, the engine can capture snapshots of the study area in certain time intervals. For example, the user can request to take snapshots every 5 simulation seconds for a total of 100 simulation seconds. This scenario provides us 20 snapshots stored in a file and, as mentioned earlier, later can be loaded to view the details of each snapshot. These mementos can then be used to replicate similar conditions as they carry all environmental parameters at a given time. Such an approach alleviates certain impediments regarding efficiency. As the mapping engine can be used to map larger areas, real-time performance might not be possible. If real-time tracking is not required in a certain use case, exporting the environment for later evaluation helps the simulation run efficiently.

3.3. FloodSim: Flood Simulation Framework Using Position-Based Fluids

Position-based fluids has been used successfully in a variety of real-time applications. Nevertheless, as it is a relatively recent method compared to other more traditional methods such

as SPH, it is not utilized frequently in the literature. Moreover, FleX library [132], which is a unified particle-based simulation library, includes an implementation of position based-fluids. The version 1.0 of this library was released in March 2016 as a standalone development tool and the plugin for the Unity game engine was released in July 2018. As it is a recent implementation, not many works are found utilizing this approach for surface runoff simulations.

FloodSim, the simulation framework built for this study, is layered into multiple levels of abstraction. There are multiple modules that are initialized to prepare the simulation engine for further processing as several environmental parameters and components are required to simulate flood. These environmental parameters alongside objects to interact with define the simulation environment. Moreover, the features and surface properties need to be initialized to define meaningful interaction between simulation objects. Figure 14 shows an overview of the simulation mechanics implemented over the environment.

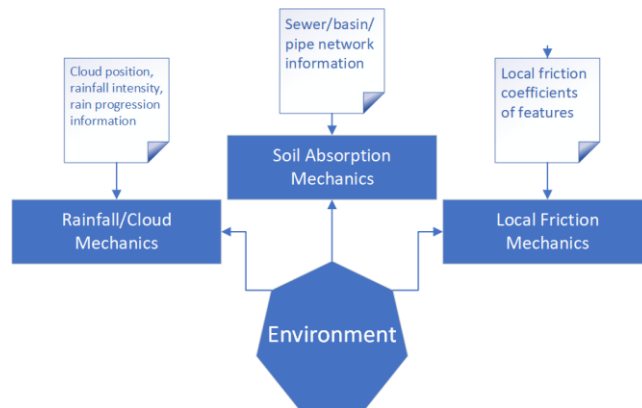


Figure 14. Overview of the simulation mechanics.

Position-based fluids is used as the method of simulation. As mentioned earlier, among computational method to render and visualize fluids, position-based fluids is a recent but a well-established method. Its implementation, FleX resides within NVIDIA's libraries for game

development and consequently, it is well optimized for real-time soft interactive simulations. Utilizing position-based fluids alongside Compute Unified Device Architecture (CUDA) [133], NVIDIA hardware can be fully utilized for various purposes. Furthermore, for non-NVIDIA hardware DirectX can be utilized for processing. Currently the available platforms are Microsoft Windows, Linux and Android.

With the use of position-based fluids as the main method of simulation, local friction, soil absorption and rainfall mechanics are implemented, to supplement the simulation with added realism through enhanced representation of the study area. As global friction would not be sufficient to distinguish between different types of objects that interact with water, a local friction coefficient is implemented. Through the local friction coefficients model, every feature drawn on the scene is reserved a friction coefficient, which is later sent to the system for utilization in the fluid-solid interaction. Various works underline that some utility of local friction coefficients are used to simulate flood flow [134, 135, 136] and consequently, the implementation is provided for the users to provide friction values for the features. Although it is not investigated in detail, through proper refinement and adaptation, these coefficients should be able to describe a detailed interaction between different types of objects. For this work, an urban environment with 3 major categories of objects, buildings, road networks and landcover are considered and provided different friction coefficients. These friction coefficients are derived from Manning coefficients and calibrated to be suitable with global static and dynamic friction coefficients for the position-based fluids framework.

Soil absorption is an important mechanic regarding proper representation of the environment concerning fluid flow and arrangement of storm sewers, basins and pipes. 3D implementation and design of these features would be possible given the availability of data.

Nevertheless, it is challenging to design the features in 3D for this purpose as for many applications, pipes, sewers and such flow information is modeled in 1D [12, 88, 137]. Furthermore, it is extremely expensive in terms of visualizing an underground pipe network of a large city and consequently, soil absorption mechanic is not tracking the pipe network; it is implemented as an approximation of the available storm drains of given sections within a city. This information is then utilized to drain the waters as they pass through the environment.

Lastly, we implement a rainfall mechanism utilizing fluid emitting clouds to represent rainfall values of the study area. Multiple clouds are placed in the simulation environment to account for spatial distribution of rainfall and differing rates in differing areas within an environment. The dynamic implementation of these clouds allows us to animate them and therefore, rainfall can have varying speed and location. Each of these clouds are provided their unique keyframes for animation over a specified timeline. These keyframes are supposed to be programmed according to the data availability of the study area. It is possible to provide made up values just to test the environmental parameters and fluid interaction. Nevertheless, their purpose is to receive real-world input as rainfall parameters and show the changes within the environment by altering the position of clouds, scaling them, or changing the rainfall intensity.

Utilizing these modules implemented for the FloodSim, surface runoff is simulated. Updating environmental parameters occur every specific interval, as defined by the user for efficiency purposes. Under normal circumstances, water flow is a continuous phenomenon and therefore, position-based fluid solver never stops computation. Nevertheless, to reserve the processing power for computation, rendering may be delayed or turned off. For example, instead of rendering the updated environmental parameters every simulation second, a user can request a render every 10 seconds, reducing the demand from the GPU for better performance. Moreover,

as described in Section 3.2, 2D adaptive visualization of the environment can be used instead of 3D visualization. 2D visualization is provided to our flood simulation framework as a module and therefore, it can be utilized if selected.

3.3.1. Global and Local Friction Model

Simulations require several important environmental parameters for the computations to be done accurately. Among these parameters, friction is vital to provide the visuals about the roughness of the surface and more importantly, to solve the fluid equations. The viscous forces, which are considered in the computations, exist as friction is considered between fluid particles. As friction is related to the viscosity coefficient in the equations, it directly affects the computation of these equations and therefore, to provide a more realistic visualization of the environment, appropriate friction coefficients must be provided.

As part of the position-based fluids framework, a global friction coefficient is provided to dictate the overall simulation and visualization of the environment. Most of the particle-based simulation software provide a global coefficient to simplify the computation. Furthermore, two global friction coefficients as static and dynamic friction are provided to account for different scenarios. A static friction coefficient is considered for stable entities in the environment while dynamic friction coefficients are utilized to define the behavior of moving fluid particles. With static friction, the stability of water particles can be achieved as the coefficient defines the behavior of the particles when they are not in motion. On the other hand, dynamic friction defines the interaction between the surface and the fluid as the particles are moving. Nevertheless, both coefficients are global friction coefficients and do not define the behavior of different materials in the environment.

As our domain is urban environments, differentiating between grasslands, farmlands, road networks, concrete and any other significant material is imperative. Water flow and soil absorption works differently on different surfaces and thus, local friction coefficients are implemented to define this behavior. To enhance the global friction model with the local friction model, Manning coefficients [28] are utilized. Utilizing these values with the surface information coming from the flood visualization framework, appropriate behavior of water is defined. As we focused on extracting the environment from OSM data using the City Maker that we implemented, environmental information comes automatically. As the OSM tags carry the context through tags and values, a feature being farm, road, waterway or any other significant entity is predefined. Nevertheless, if a user chose to import a manually defined environment, then an appropriate file carrying contextual information must be provided. For this, our system reads environmental information alongside the Manning coefficients file to define areas in the environment with distinct friction parameters. Matching feature types with the values found in the coefficients file, roughness parameters are embedded into the digital model generated. This process is shown in Figure 15. In this figure, darker reds show higher friction coefficients while lighter ones indicate lower coefficients. By using this approach, overall behavior of water is defined for the study area. Road networks are more slippery, and they tend to allow flood waters to flow freely while surface like grass and farmlands provide more friction due to roughness of these surfaces.

An advantage of using the City Maker as described in section 3.1, is that these coefficients are baked into the simulation environment and therefore, no further processing is required for the simulation. If the simulation parameters are not embedded using the City Maker, then the users must define these simulation parameters for the study area and pass the required documents to the simulation engine. Furthermore, City Maker contains more context than most other formats

through key and value pairs for tags. Extracting this information might be important for the users. However, one disadvantage of using this approach is that the OSM data is not defined for many urban environments and therefore, it might not extract detailed information if it does not exist in the OSM map data.

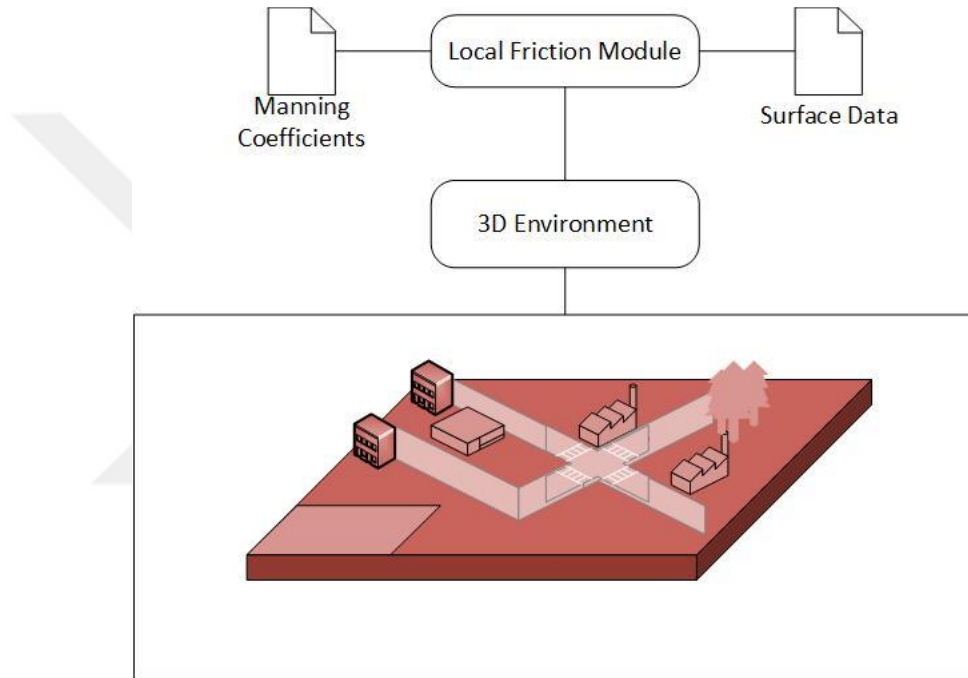


Figure 15. Color coded representation of the application of Manning coefficients.

Upon dividing the environment into several parts to utilize different local friction coefficients, simulation proceeds normally. Not only the friction coefficients extracted from the environment information, but custom coefficients can be utilized if required by the study. The surface definition format allows for extension and thus, new areas can be defined to use different friction coefficients. As mentioned earlier, this approach is needed for definition of the study areas that have missing OSM data. By defining local friction coefficients using this approach, missing context regarding surface properties may be embedded into these environments. For example,

Corpus Christi has nice coverage on OSM but cities like Houston and San Antonio lack certain environmental details. Although for our test scenarios this feature is not utilized, it could be useful in the hands of professionals who have sufficient data about the study area to enhance realism and increase resolution of the simulation environment.

3.3.2. Soil Absorption Mechanics

An important component embedded into the simulation framework is the mechanics that define how soil absorption works within the environment. Considering other frameworks developed to visualize water translation, this feature is either left out intentionally or implemented with certain approximations. Although empirical models may utilize the real values for the soil type of the environment after a thorough survey of the region, experimental models such as SPH-based 3D fluid mechanics utilize alternative ways to compensate slower performance due to particle interaction. As underlined earlier, another approach is to leave this feature out if the experiment is about surface runoff simulations and environmental effects over some period is not being tracked. For such simple applications, soil absorption does not yield any further benefit.

In the context of this study, soil absorption does not only refer to the terrain, soil type and how saturated the soil is. It also refers to overall absorption of waters, through citywide stormwater drains. The side inlets and grated inlets are modeled according to the specification provided by the city, if available. Upon modeling these inlets and placing them properly around the 3D environment, the soil absorption works by removing a certain number of particles per simulation second to represent inlets carrying water away. Consequently, it is vital to find the drain information of the study area. Furthermore, features placed in the environment and the terrain may have different local soil absorption coefficients for realistic behavior of flood waters. Therefore, for each feature drawn on the environment, a local absorption coefficient is provided for further

accounting for the different types of soils environment might have. For example, a park may have a higher intrinsic absorption value than a road. By utilizing storm sewer data, this behavior cannot be provided and therefore, feature level absorption coefficient can be used for representing this absorption mechanic. Although not calibrated for this work, it provides a potential use case and with the availability of real-world soil information, appropriate parameters can be provided to customize the environment for more realistic simulations.

Flood modeling often utilizes 1D or 1D/2D hybrid models for pipe networks, to define water flow and determine the amount of flow the pipe network may carry. For many purposes, this approach is suitable to represent accurate values and hence, it is the preferred approach in numerous flood models. Nevertheless, with the advent of 3D technologies and utility of 3D visualization, different approaches can account for soil absorption and pipe networks. Full representation of the storm drains and pipe network, although possible, may not be the best idea in terms of efficiency and simulation stability. As particle-based simulations focus on particles' interactions with the surrounding environment, too much interaction may cause slow performance, simulation instability and consequently, visual anomalies. To prevent this, our work proposes global and local soil absorption coefficients for the study area and particle removal is handled according to these coefficients. Furthermore, storm drain information, if available, is incorporated into this model, and can be visualized in any resolution that the study area requires. Figure 16 shows an overview of soil absorption mechanics and how it places absorption coefficients across the virtual environment.

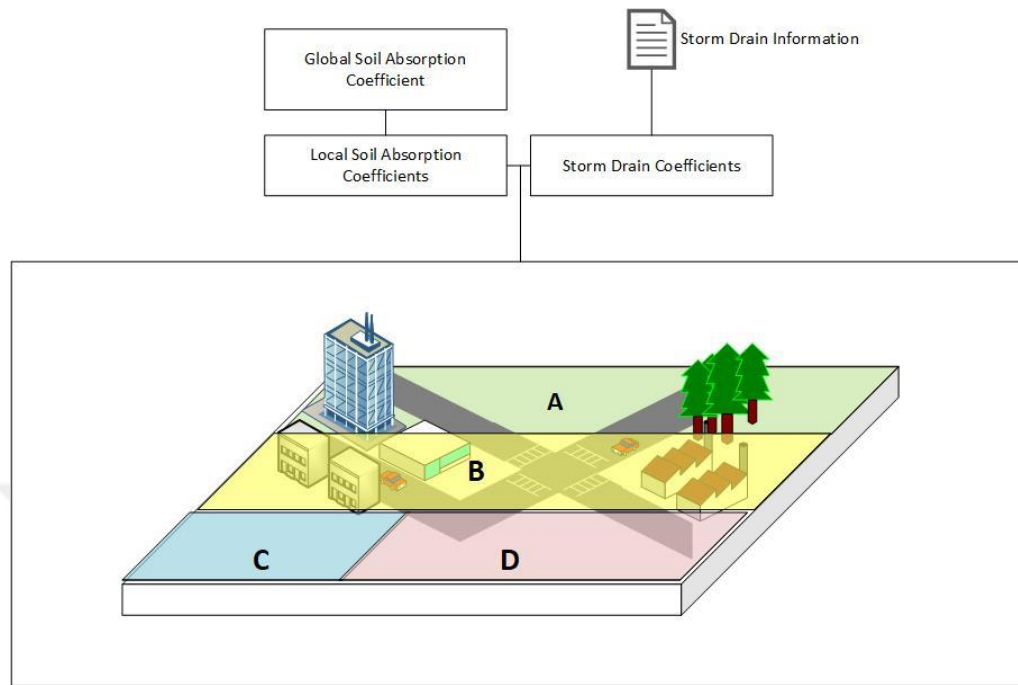


Figure 16. Process of generating absorption coefficients for the study area.

In this figure, reading the information from storm drain data and utilizing other global and local soil absorption coefficients, regions A, B, C and D are defined. These regions have different absorption values and thus, water particles are removed at different rates. It is possible to provide a very high resolution by providing unique locations of every single side inlet, sacrificing performance in the process. Nevertheless, for real-time simulation, it might not be the most appropriate solution. An adaptive approach benefits both the demand of performance and detail in description. Consequently, our work provides an adaptive approach in which, grid sizes may be altered on demand to provide both performance and detail.

The storm drain document is a vital part of this implementation as it contains detailed representation of the environment and where the inlets are placed. This document provides longitudes, latitudes and flow line elevation information for the absorption computation to run appropriately. As such information might be incomplete or unavailable and therefore, resolution

can be decreased to represent an overall absorption for given areas, instead of focusing on individual inlets. This is done through utilization of an adaptive grid as discussed in section 2.2, by using a quadtree to divide the study area into grids. By predefining a proper level of depth as required for the study, absorption is computed through averaging the inlets' properties in generated cells. This behavior is then visualized in 3D, through particle removal per simulation second, as part of the flood simulation framework. Overall, this process is efficient because predefined cells contain fixed number of inlets and therefore, computation of absorption coefficients are done prior to fluid simulation. Consequently, the simulation engine proceeds with no interference as it does not need to reassess the parameters per simulation second. A portion of the storm drain information document is provided in Appendix C.

3.3.3. Cloud and Rainfall Mechanics

Rainfall must be described in some meaningful way that its inherent behavior can be properly represented in 2D and 3D simulations. This behavior mostly comes from the observations that we can find for the given study area. Nevertheless, it might also be preferable to conduct experiments instead of following and engineering the existing data for the purpose of visualization. Cloud and rainfall mechanics include instances of clouds and a rainfall intensity document to read from to represent the rainfall data in 3D environment. Furthermore, custom clouds can be defined and placed around the map with differing rainfall intensities, sizes and speed. Hence, cloud and rainfall mechanics offers full customization of how the particles are added to the simulated environment. Together with soil absorption mechanics, they define addition and removal of particles from environment and thus, they are vital components of the simulation for the purpose of efficiency.

Cloud objects are formed with several significant parameters to define their position and direction to describe the rainfall. These objects are read from a cloud configuration file, which holds references to the clouds with their identifiers, position, rainfall and rainfall intensity. At this point, it is possible to activate the clouds to make the rainfall occur but, this is normally done dynamically using cloud animation objects to dictate cloud behavior in the simulation. A vector keeps track of the cloud references and the objects get processed at every time step. There is no limit for the maximum number of clouds that can be generated in the study area. Nonetheless, there is a hard particle number limit for real-time simulation purposes, which might stop clouds producing additional rainfall. Although this limit can be lifted, real-time behavior may not be possible without a particle limit. Figure 17 shows a representation of a cloud object.

Cloud position is derived from the geographic coordinates found in the cloud configuration file. This information is, in most cases, implemented manually and therefore, initial evaluation of the rainfall data may be required. Upon checking available rainfall data for the city, the practitioners can divide the city into several subsections, define clouds, or areas where rainfall occur, and then input the expected rainfall amount for the given area.

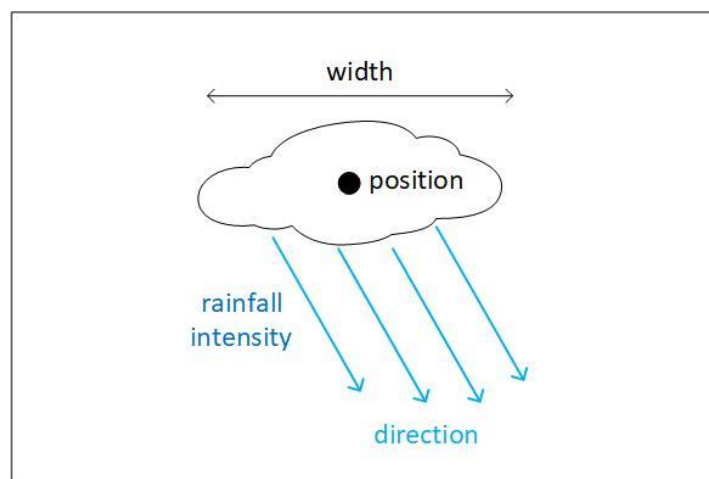


Figure 17. A cloud object and its significant parameters.

Cloud animation objects are, in truth, extensions of cloud objects. Each cloud object comes with an additional cloud animation object associated with itself. These cloud animation objects carry keyframe information to specify important changes over time such as cloud movement and rainfall intensity changes. Moreover, an active cloud can be deactivated after a certain amount of time. The keyframes contain start and end timestamps to define an animation. When the cloud animation objects are loaded into the scene, the vector containing these objects are passed to the visualization engine. At this stage, the visualization engine restructures the data and processes keyframe information as the time passes. Multiple timers are utilized to account for the CPU time, simulation time and animation times and therefore, slowing down or speeding up these animations are possible.

Available animations provided for cloud animation objects are creation, translation and deletion of clouds, altering rainfall intensity, changing the direction of rainfall and altering the width of clouds. Some of these actions, such as altering rainfall intensity, can be initialized as gradually changing over time. If gradual change is not set up, then an immediate change occurs at a given timestamp. The timestamp feature utilizes 24-hour notation to determine the time of events written in cloud animation configuration file.

If rainfall data can be acquired, then cloud animation objects can be helpful in designing a real-world scenario and evaluate the results for calibrating the local dynamics of the system. Utilizing timestamps acquired, the cloud configuration files can carry real world precipitation information and visualize the event in the system through appropriate calibration of the local environment. Otherwise, if no data is available or no calibration is attempted, clouds are customizable and consequently, any rainfall related scenario can be tested on the study area. Figure

18 shows a representation of how cloud animation data files work and how they help the simulation engine to visualize cloud information over time.

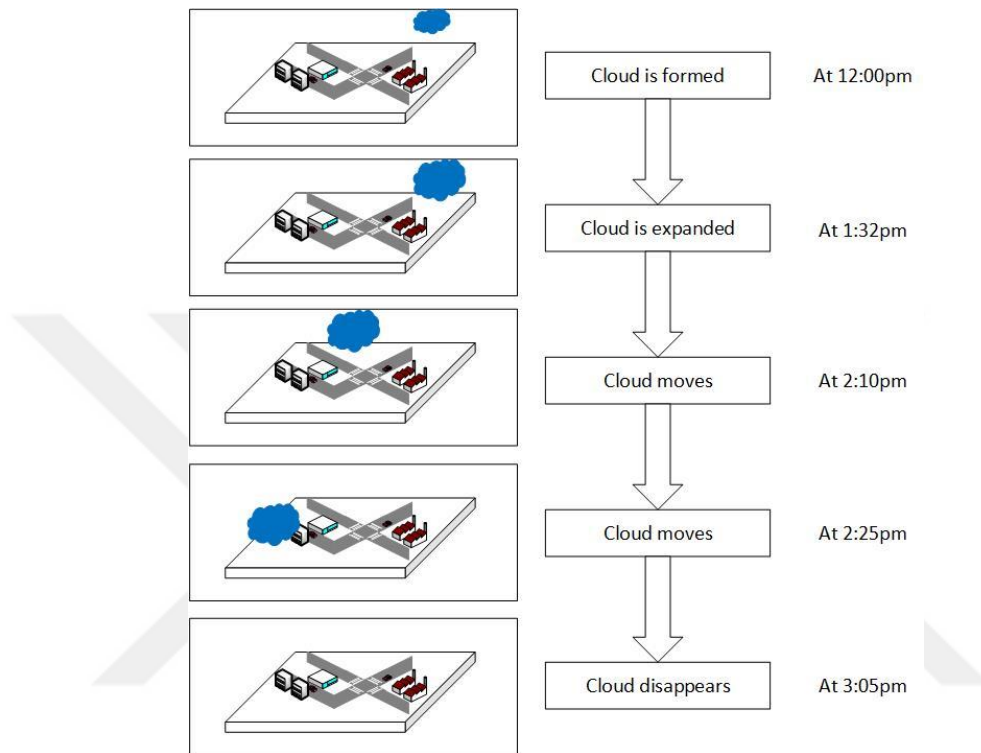


Figure 18. Representation of a cloud animation object timeline.

An advantage of utilization of the cloud animation objects is that the simulation environment becomes quite extensible and allows users to manipulate information in real-time. Furthermore, such conditions can be recorded and logged into a data file for rebuilding the same environment. Although the randomization in the wind factor, rainfall splatter and GPU processing would alter the visualization to some extent, it would be possible to recreate similar conditions and evaluate the parameters of the environment. Another advantage is provided through the provision of dynamic parameters for the cloud. Not only the position and rainfall intensity, but any other parameter can be given values and be animated over time. This allows for interpolation of the unknown data, as the user can define a command like ‘increase the intensity by 5 mm over an

hour'. Therefore, rather than utilizing abrupt changes of environmental parameters, a smooth transition can occur, which would potentially help practitioners incorporate their data into the simulation environment.

3.3.4. Workflow

Upon initialization of the simulation engine, the scene descriptions are read. As described in section 3.2, the scene description contains the environment to be utilized for the simulation, and relevant translations, initialization of certain simulation parameters and reading flood simulation framework module information. Initially the environment is loaded as terrain for the scene. Terrain objects can be a singular unified entity or can consist of hundreds of parts. The framework allows both approaches to be utilized. The advantage of utilizing a unified body is that the simulation runs more efficiently. As the object does not contain subparts to be evaluated further once particle interaction occurs, the computation is handled in a straightforward fashion. The disadvantage is, obviously, the lack of feature differentiation and therefore, it might be detrimental if local friction coefficients are to be utilized. Conversely, the advantage of utilizing multi-part terrain is the inclusion of local friction model with the cost of additional computation per feature placed in the terrain. This might be a big loss of performance, especially for larger areas, considering one of our main focuses is to provide an efficient flood simulation framework.

After the terrain and some simulation parameters, such as maximum number of particles are initialized, environmental configuration files are read. These configuration files are, cloud configuration files, storm drain information, building information, if any individual building configuration data is provided, and lastly, local friction coefficients. By reading these files and initializing certain data structures to be utilized in the computation, environment is organized for the simulation. After this, a height map is generated from the terrain to be used in depth

computation once the simulation begins. With the creation and the storage of the height map, the simulation starts executing and producing results as requested.

Initially, a choice is made to utilize 2D depth tracking alongside 3D flood simulation. If selected, then the depth map and related storage, such as a regular grid and a quadtree with provided maximum depth, are created. With this utility, simulation can be visualized in a mini map placed at the right-bottom part of the simulation environment, alongside 3D information in the main simulation screen. This mini map provides depth information as the water traverses the environment. By providing a 2D view of the environment while getting the benefits of 3D simulation, as underlined in [6], we provide better clarity of the depth information of flooding across the study area. An outline of the user interface developed for FloodSim is provided in Figure 19.

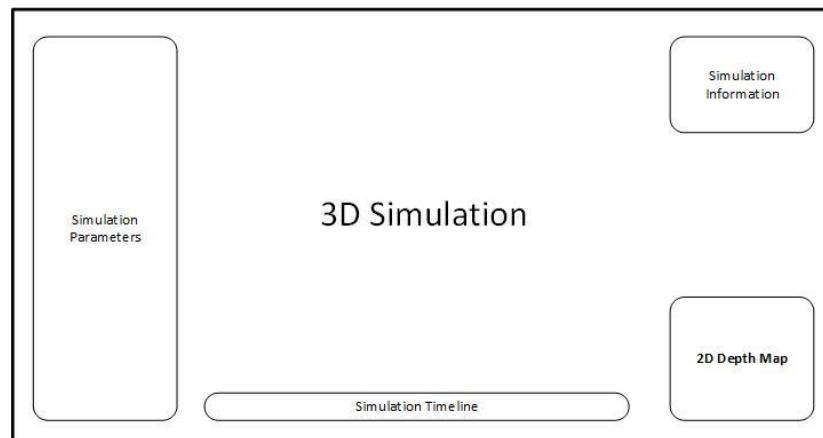


Figure 19. User interface of FloodSim.

Upon initialization of the 2D mapping module, the information on the environment is updated per frame, starting with the cloud information being processed, rainfall and hence, the water particles being produced and positioned. The clouds are set to emit the water particles with

given rainfall intensity and direction. Upon creation of these particles, the buffers are utilized to track different aspects of the particles, such as positions, velocities, densities and several other parameters. These parameters are vital for position-based fluid simulation environment to work properly.

Every simulation second, several important conditions are checked and processed by the system. The engine sends an ‘absorb’ message to the soil absorption module, which is responsible for checking environmental soil absorption coefficients and processing the requests coming from the simulation engine. As the scene initialization already covered reading soil absorption coefficients and restructuring the data into proper data structures, soil absorption module simply checks if the particles are touching the soil. If the module recognizes the particles touching the soil, it utilizes the absorption rates defined for the sections of the simulation environment to remove these particles from the environment. Once particle removal is complete, the simulation engine processes cloud animation objects. The cloud animation data file is read during scene initialization and therefore, the simulation engine has the knowledge of all future events at this stage. The engine considers the current simulation time and examines the timestamp information embedded into cloud animation objects to estimate if new events occur at the current timestamp. Furthermore, if cloud parameters are currently being animated, the simulation engine calculates the parametric changes that will occur in the next frame and passes this information to the drawing component. In the next frame, the drawing component flushes the updated parameters.

These operations continue until the simulation is halted. Module refresh rate can be modified if requested. For the purpose of this work, we considered refreshing the parameters and processing the depth information every second, setting the required frame rate to 60 frames per

second. If efficiency is not a concern, this parameter could possibly be initialized using milliseconds instead of seconds. A typical workflow in FloodSim is represented in Figure 20.

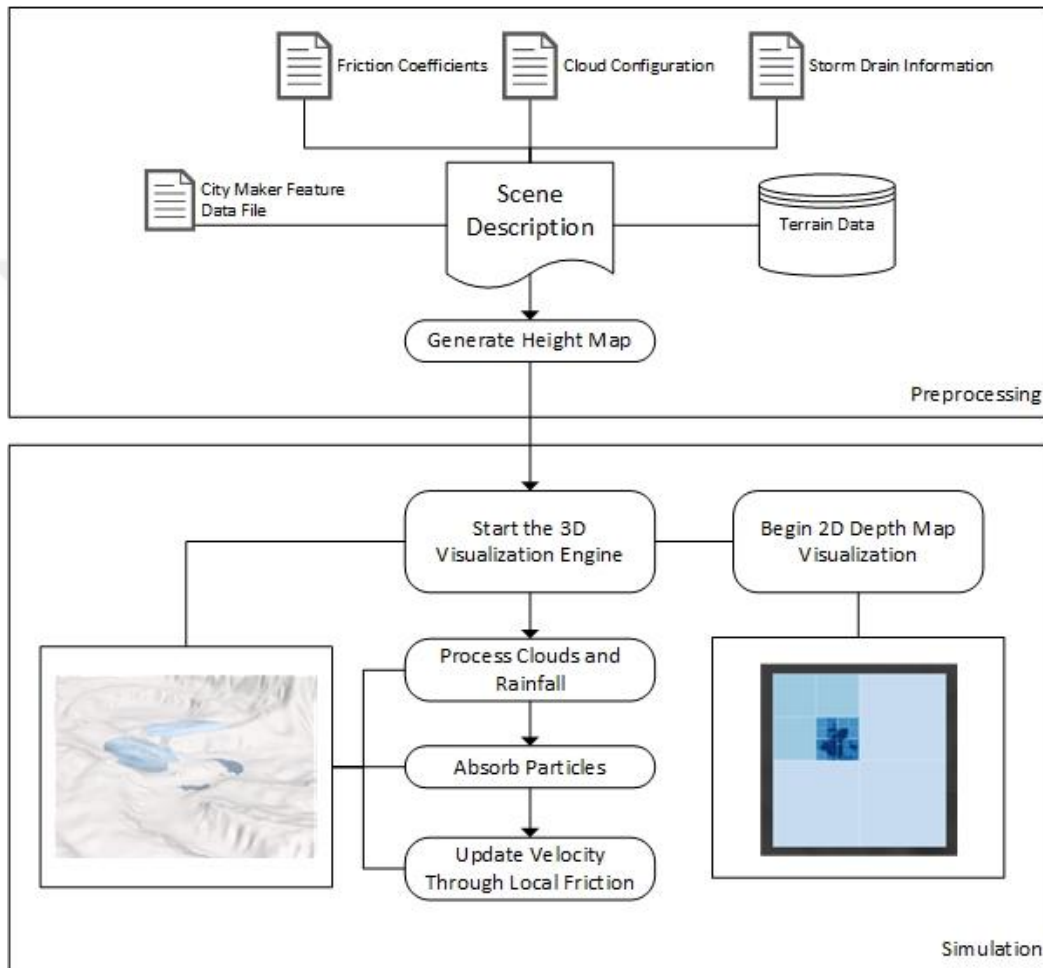


Figure 20. FloodSim Workflow.

CHAPTER IV: RESULTS

Urban environment generation is applied to different sections of cities and tested to see if data extraction and 2D/3D visualization is proceeded with properly. Initially, a 2D wireframe map is implemented to test data extraction capabilities of City Maker. In this test, several flags can be utilized to display the features we would like to see on the map. For example, using the implementation, we can turn on or off roads, buildings, farms, and various other features that can be extracted from the OSM data. A convenient tool to load and save city information is implemented for this purpose. Figure 21 shows the user interface developed for visualization of city information.

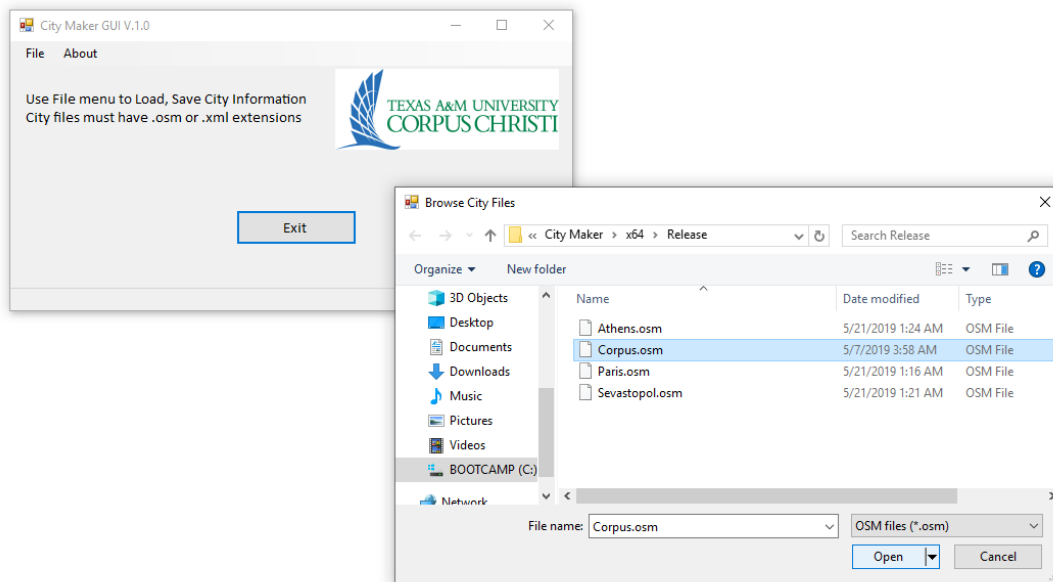


Figure 21. User Interface of the City Viewer.

Upon selecting a city, the file is parsed, and relevant boundary information is processed. Initially, the process captures the boundaries of the study area and considers the visualization of the environmental properties, features and other components relative to the boundaries it extracts. With the city information parsed, factories process each type of GIS feature implemented. In the

end, a new window, utilizing OpenGL, displays the 2D information of the environment with the selected feature layers. At this point, various color schemes are designed to visualize the features. Depending on the user preferences colors may be inverted to provide a more comfortable display of features. Figure 22 and Figure 23 shows different parts of the city using the default color scheme.



Figure 22. Visualization of the TAMUCC Island and parts of Ocean Drive.

As it can be seen in the figures, features can be visualized in 2D by appropriately parsing OSM data. As mentioned in Section 3.1, some features are not directly implemented due to either missing information or lack of implementation for user-defined tags. Nevertheless, most of the vital information including roads, buildings, farms, parks, bodies of water, pools and many others are visualized with this approach.

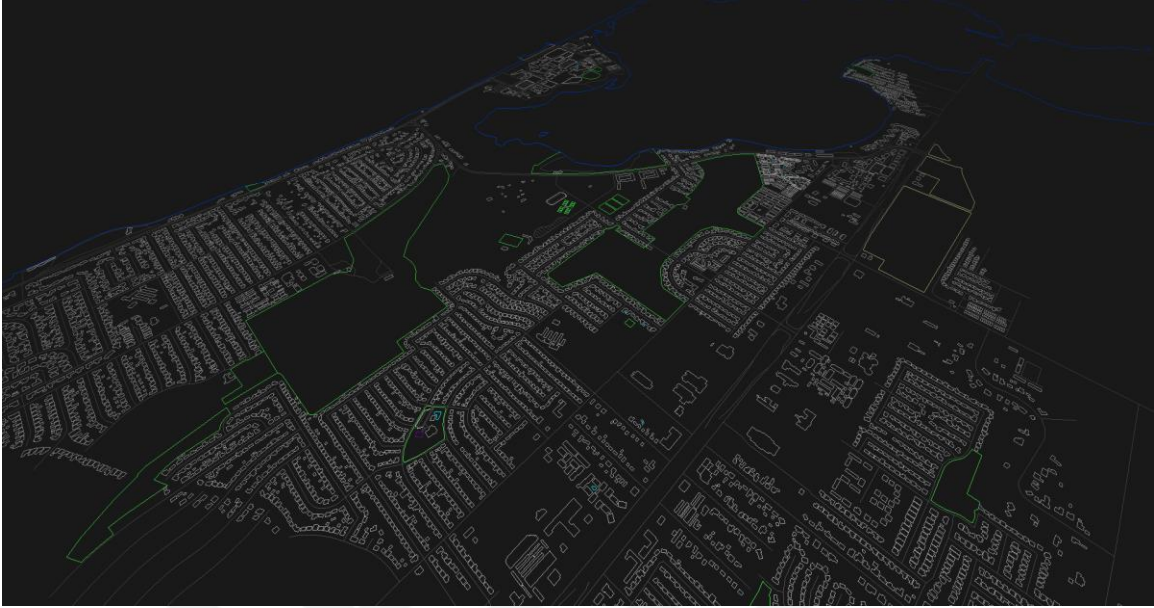


Figure 23. Visualization of parts of Corpus Christi in 3D wireframe.

As it can be seen in the figures, features can be visualized in 2D and 3D wireframe modes by appropriately parsing OSM data. As mentioned in Section 3.1, some features are not directly implemented due to either missing information or lack of implementation for user-defined tags. Nevertheless, most of the vital information including roads, buildings, farms, parks, bodies of water, pools and many others are visualized with this approach.

Furthermore, the environment can utilize the building height value information it derives from the OSM data if available. As it varies greatly from region to region, default height values are also provided as can be seen in Figure 23. This is the case for numerous countries, which provide local coordinates for the features, yet fail to incorporate height information into the data set. Nevertheless, Corpus Christi contains height information of most of the buildings in meters and thus, it is suitable to be rendered with building heights embedded. The visualization of the buildings with the height information and colored polygons is shown in Figure 24.

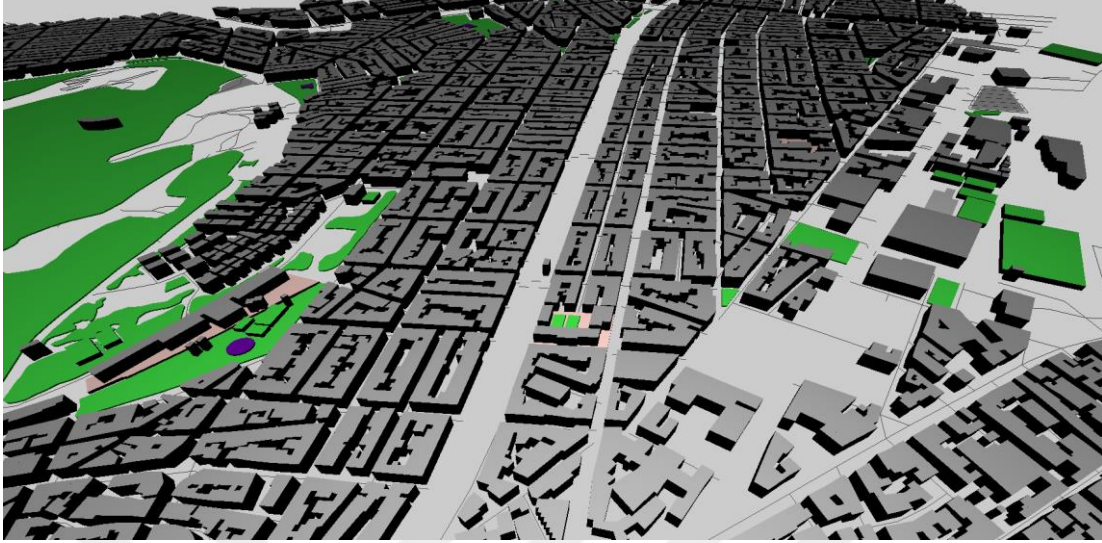


Figure 24. Visualization of parts of Athens, Greece.

Considering the size of standard OSM data files, processing sections of a city may take some time. Table 1 is provided to show the advantages of the processed city, in City Maker format, for viewing the environment. For the cities, only parts of these cities are selected and compared. For recognition of their size, significant features are provided in the table. As it can be seen, although parsing time of the OSM data varies depending on the size of the area and especially the significant features like buildings and roads, once parsing is done, the City Viewer, loads the city information in less than a second for the three examples provided. Parsing and viewing times are averages after 100 executions and it can be concluded that it provides a reliable and fast viewing experience. For the cities with a larger number of features, such as Brooklyn in our case study, it took on average around 7.5 minutes of parsing yet less than a second of loading. This is due to the node structure of OSM data files as they are relationally attached to components and require further processing. However, this hierarchical format helps compile the city information properly and therefore, it may be required the way it is. Trimming can be utilized if certain features are not going to be visualized. This option is provided in the City Maker.

Table 1. Performance comparison of parsing and viewing three different cities and their embedded features

	Corpus Christi, TX	Athens, Greece	Brooklyn, NY
Buildings	5686 features	9393 features	31800 features
Roads	671 features	415 features	1742 features
Parks	8 features	4 features	18 features
Hospitals	1 features	2 features	5 features
Schools	4 features	5 features	15 features
Parsing OSM	17.3 seconds	37.2 seconds	458.4 seconds
Viewing CM	0.58 seconds	0.6 seconds	0.80 seconds

With the features visualized in 2D, 3D urban environment generation can be proceeded with to visualize the city in a more detailed fashion. Although the same features and properties are implemented for the 3D scenario, there are significant differences in the workflow as described in Section 3.1. For example, the 2D wireframe visualization does not contain any SRTM elevation information, yet for 3D visualization purposes, this is required. Before the study area is parsed for feature extraction and 3D visualization, SRTM data of the study area is acquired. Initially, the study area is selected from openstreetmap.com and the coordinates are stored, as shown in Figure 25, for SRTM processing. These coordinates are then used to query SRTM data from NASA.

With the features visualized in 2D, 3D urban environment generation can be proceeded with to visualize the city in a more detailed fashion. Although the same features and properties are implemented for the 3D scenario, there are significant differences in the workflow as described in Section 3.1. For example, the 2D wireframe visualization does not contain a SRTM elevation

information, yet for 3D visualization purposes, this is required. Before the study area is parsed for feature extraction and 3D visualization, SRTM data of the study area is acquired.

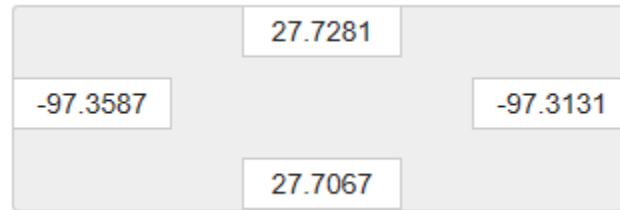


Figure 25. Selected area to extract OSM information.



Figure 26. City of Bursa and the mountain Uludag.

Initially, the study area is selected in the openstreetmap.com and the coordinates are stored, as shown in Figure 25, for SRTM querying. SRTM data queried from NASA is horizontally referenced to WGS84 and vertically referenced to the EGM96 geoid. Upon acquisition of void filled SRTM data with 1 arc-second resolution, the elevation mesh is generated to be incorporated into the 3D simulation framework. This mesh is converted into .obj format with its texture, which is the satellite image of the study area, attached. Nevertheless, for performance and testing, the textures are not rendered. As the data structure contains the appropriate UV map and the texture

file, this should not be an issue and is considered for future work. Figure 26 shows the satellite image of another study area, Bursa, Turkey, with its parts close to Uludag mountain.

Part of the city and the mountain are selected to be converted into a 3D environment. As indicated, initially the SRTM data of the area is acquired through querying the SRTM data of the selected area. Upon acquisition, a mesh is generated using BlenderGIS plugin. This mesh, with no texture is shown in Figure 27. After creation of the mesh, an appropriate texture is arranged and the object file is created with its vertices, normals and texture coordinates attached. If requested, BlenderGIS is able to incorporate texture information into the models, yet there may be higher resolution imagery available elsewhere and therefore, other sources can be used for this purpose. Figure 28 shows the terrain after a texture is attached. The object can then be passed to the urban environment extractor to be used as the base, or the canvas, as implemented for this work. Then, other objects are incorporated into the environment by getting added to the canvas.

The canvas, therefore, is the foundational layer of the environment, which can be used to embed other features to the environment. Environmental information is acquired through City Maker files as discussed earlier in Section 3.1.

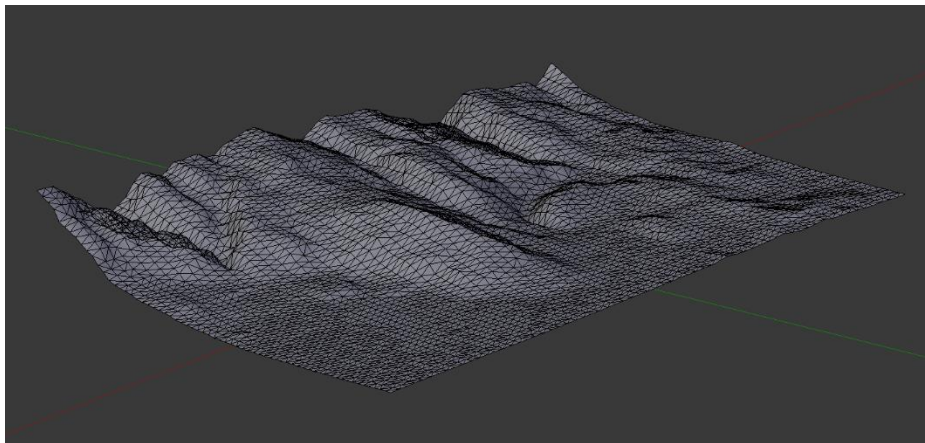


Figure 27. Mesh of the study area built from SRTM data.

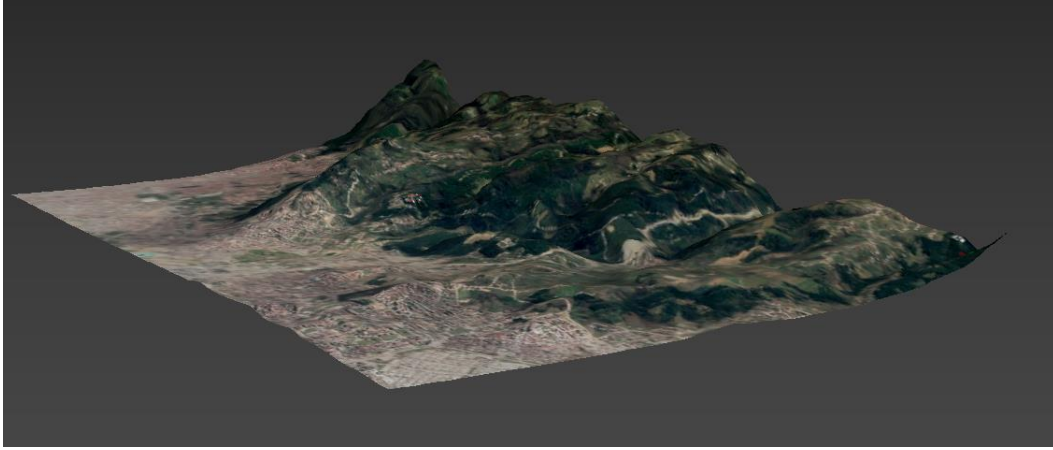


Figure 28. Textured mesh of parts of Bursa, Turkey.

Upon generating the mesh, similar to the previous stage of wireframe feature visualization, OSM data is organized to visualize the environment in FloodSim. For object representation, various models are utilized. As for large environments in which the mesh generation cost is too high, rectangular prisms are used to represent residential buildings. Other features are assigned unique objects from the model data set. With availability of additional models, the environment generation can be done uniquely. Furthermore, as simulation performance is significant, textures are useful only for post simulation visualization of the environment and not when real-time simulation takes place.

Upon generation of this environment, it is transferred to the simulation engine. For variety in the results, we utilized many different environments for representation purposes and provided figures for different environments. Some of these environments are test environments; they are virtual, and they do not have any real significance. They are used for testing and calibration and they represent some of the features and mechanics implemented for this work. Some other environments are real world environments extracted by the City Maker and they are used for integrating real city data into the model. Sections of Corpus Christi are utilized for demonstrating

usage of storm drain information alongside precipitation information to demonstrate these features of the simulation framework. Figure 29 shows the output of the City Maker for parts of Ocean Drive and the surrounding area. The other visual is captured from the FloodSim showing a simplified representation of the environment for flood simulation purposes. Real world geometry is not required to process FloodSim data as the bounding boxes are extracted for different mechanics the simulation framework requires. As representing real geometry through tessellation requires additional processing, a decision was made to left out such geometry and utilize basic primitives, such as rectangular prisms for visualization of the environment. If available, textures can be mapped, yet for demonstration purposes, no textures were utilized.

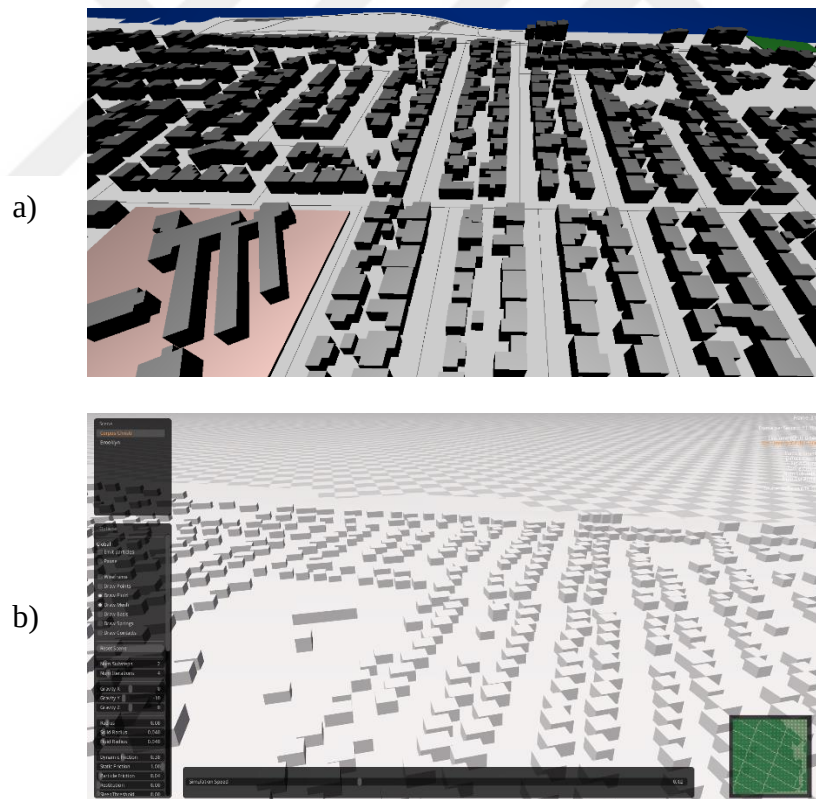


Figure 29. Representation of parts of Corpus Christi, TX through a) City Maker and b) FloodSim.

Several other features, such as parks and golf courses are removed from visualization in the FloodSim, yet they are recognized and provide vital information. As the bounding boxes, or coordinates, are carried along with the City Maker output, the areas of friction and soil absorption can easily be initialized with this approach. So, a *'landuse'* key with a *'farmland'* value would correspond to a bounding box with added mechanics for soil absorption and surface friction that is calibrated for earth. Figure 30 shows another environment, parts of Brooklyn, NY, visualizing the environment. This example clearly shows FloodSim drawing bounding boxes as it utilizes wireframes to display the information on the scene.

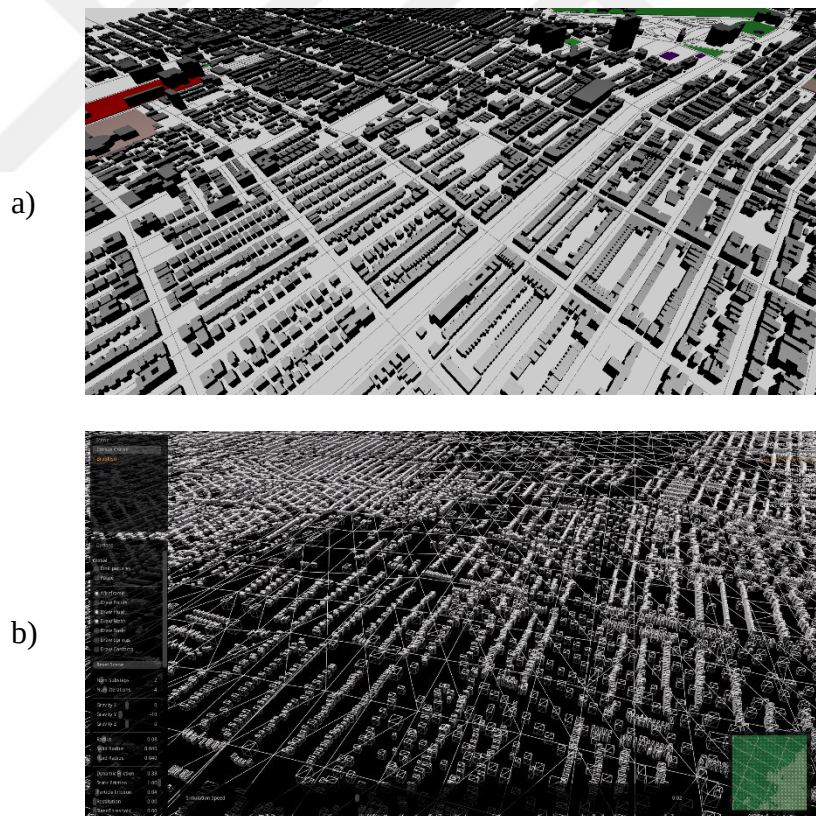


Figure 30. Representation of parts of Brooklyn, NY through a) City Maker and b) FloodSim using wireframes to render.

Once the simulation is initialized, testing is performed for 2D visualization of flooding through an adaptive spatial resolution approach. A 2D mini map for the 3D flood visualization environment is provided in the user interface allowing the results to be directly seen as the simulation progresses. In the next few pages, the results of depth mapping extracted through this approach will be provided and discussed. Upon discussion of significance of adaptive depth mapping approach and showing visuals of different scenarios using regular grids and adaptive grids together, we then proceed with discussion of 3D mechanics and how they are implemented on the environment in FloodSim.

The mapping engine was tested utilizing 10 different environments. Various terrain models were picked to utilize environments with differing characteristics so that performance advantages and disadvantages of unique structures can be determined. For each environment, height maps were generated and then a flood was simulated to keep track of water particles. As the water particles interact with the different collision mesh, performance and visual fidelity were tracked. Moreover, as previously discussed, cloud mechanics were randomly arranged to make the environment more dynamic. For each environment, a different number of clouds were generated, different animations were predefined, and different rainfall intensities were selected. As the FloodSim is utilized as the foundational layer of the simulation, mechanics relevant to the framework are utilized. At this point, as the data is randomized, visual accuracy and appropriate representation is the only metric for realism. When real world data is incorporated, other metrics are used to evaluate performance and accuracy.

Figure 31 shows a 2D regular grid and a 3D aerial view of the environment. As the simulation is occurring in 3D environment and camera can be initialized any way the user wishes, it is suitable to provide a 2D map for clearer viewing of the depth information. As 3D environment

utilizes lighting, it generates numerous shades of blue for describing water and thus, depth information may not be immediately apparent to the user.

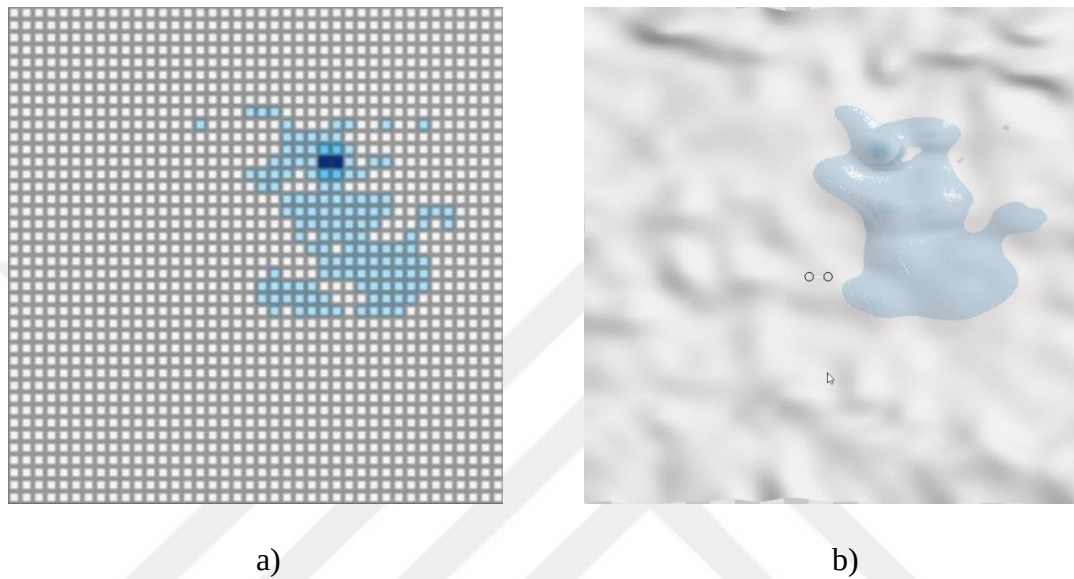


Figure 31. Visualization of the flooding in a) 2D regular grids, b) 3D aerial view.

Considering about the approach in general, it is apparent that regular grids with Jenks natural breaks for classification provide more in terms of depth. Quadrees represent the area differently and therefore they may not be preferred in some scenarios. However, they also have advantages in terms of clearer representation of flood flow as cells get subdivided once particles move into different regions in the study area. This phenomenon is more difficult to recognize in regular grids as the mini map is already subdivided. Regular grids, as they have more in terms of number of cells they are utilizing initially, provide clearer shades to represent different levels of depth in the environment. Therefore, utilizing natural breaks method with regular grids provide clearer description of depth for many scenarios. However, as the mapping engine processes more

information with this approach, adaptive grids provide better performance. Figure 32, Figure 33 and Figure 34 show the differences in visualization of some test environments.

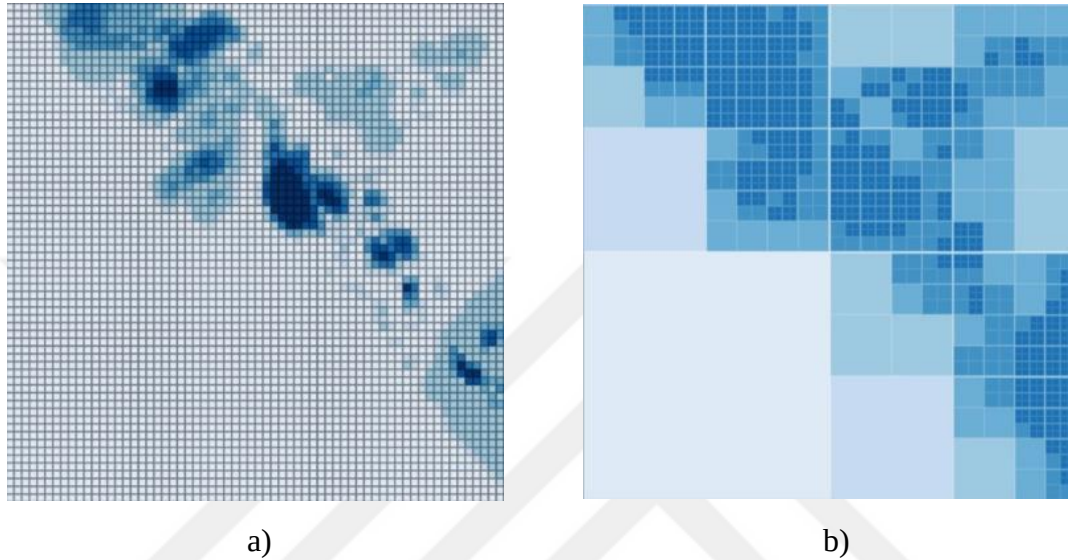


Figure 32. Visualization of the neighborhood flooding using a) regular grids, b) adaptive grids.

According to these results, adaptive maps are not able to represent the study area as detailed as the regular approach if a depth limit is defined for the quadtrees. If the limit is lifted, it can describe the environment with better detail, yet the performance benefits are also cancelled and thus, it is not preferred. Moreover, it is challenging to set a hard particle limit on each node of the quadtree as the dynamic conditions of the environment are initially unknown. Another important issue with lifting the depth limits of the quadtrees is that, with a scenario with 15 levels of depth, it is challenging to recognize 15 different shades with a single-hue coloring scenario and therefore, limiting the depth to a maximum of 9 classes makes visual sense. This approach also prevents performance issues due to creating the subtree hierarchy too deep. The mapping engine tries its best to estimate the depth limit and the particle limit using environmental parameters, such as

maximum number of particles on the scene, terrain type, and terrain height variations and provides a limit for the quadtree for performance and clarity reasons.

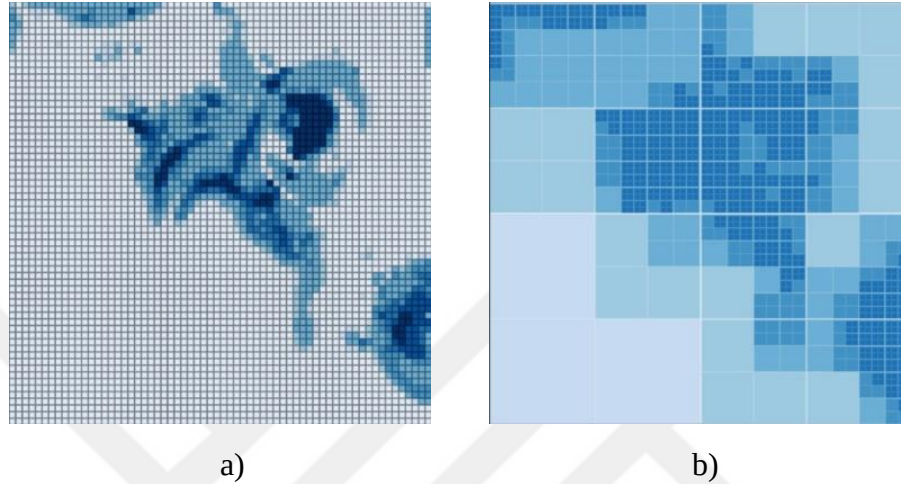


Figure 33. Visualization of the canyon flooding using a) regular grids, b) adaptive grids.

Figure 33 illustrates one important issue regarding adaptive grids with limited depth. When a certain section of the environment has high depth values, the adaptive grid approach shows similar color in the surrounding cells due to compression. However, when we consider the regular grid approach with natural breaks classification, the local maxima are clearly represented. If an overall sense of depth is sufficient for the study, especially for larger areas like cities, then an adaptive approach may be preferred. For such a scenario, the focus of the study is to estimate larger sections of flooding within a city or neighborhood, rather than detecting pools of water formed around a single rock. As the details may not be necessary, performance benefits of adaptive grids can be utilized by the mapping engine. Initially, the requirements of the study must be clearly determined before proceeding with the adaptive approach. If it is a large-scale map, which shows a small area in the simulation, then the adaptive approach may be sufficient to describe the depth information. For small scale maps, as the depth limit of the quadtree may distort results, it should

be used with care. Our system can pick the representation by considering the size of the area. Nevertheless, it is possible that it can choose to transition into adaptive grids to limit performance issues caused by 3D simulation.

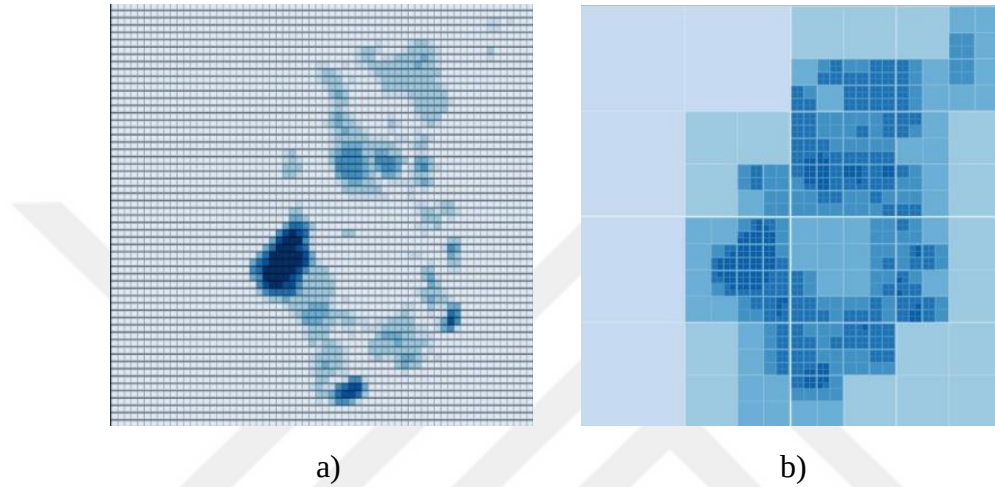


Figure 34. Visualization of the mountain flooding using a) regular grids, b) adaptive grids.

On the other hand, Figure 34 shows that when local maxima are dispersed, unlike the previous scenario, then an adaptive approach does a better job describing classes of depth. Consequently, this is one of the factors that the mapping engine considers as it switches between different methods of representation. If there are similar values distributed over a large area, then the mapping engine prefers regular grids unless performance is a concern. On the other hand, when the depth values differ and there are few local maxima defined over the map, adaptive approach is selected for representation unless more details are demanded.

As more rules are incorporated into the mapping engine, it yields better results in terms of choices it is making. Several failures that occurred during the visualization tests were due to the lack of knowledge about the surroundings. In some cases, considering the affected area of the environment, the engine decided to increase the maximum depth of the quadtree, which causes

frame-rate spikes. This is due to noisy terrains having more particle interaction. In such environments the computation takes longer and if the quadtree is too deep, it may perform worse than a regular grid. To fix such issues, additional rules are defined to guide the dynamic mapping engine to pick correct approaches. However, if more rules are incorporated, then the mapping engine cannot make decisions with a few basic items of environmental parameters and thus, it is not preferred.

As different color schemes are provided, the same environment can be visualized in different ways. The color schemes are derived from ColorBrewer, a service that provides appropriate color schemes for mapping purposes [138]. Same study area, visualized in different single-hue color schemes is shown in Figure 35.

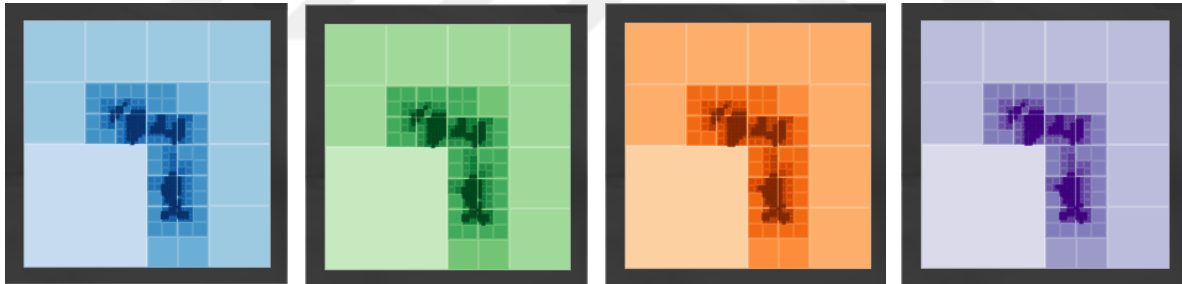


Figure 35. Visualization of the study area in 2D using different color schemes.

Figure 36 shows flood progression over 13 simulation seconds. Through calibration, each second of simulation, corresponds to 20 hours and therefore, it shows a little less than 11 days in real world time. The environment generated for this test contains a small local cloud carrying rainfall on a particular path. When regular cells and adaptive cells are compared, several interesting results can be extracted. Firstly, as the flood is in its early stages, quadtree cannot represent the event properly due to lack of information. As the tree gets divided to 4 when number of particles

pass beyond a certain limit, initial subdivision gives an incorrect sense of flooding for the upper right quarter of the study area. This might have undesirable effects considering the topic of study and precision required for such a scenario. Then, as the particles spread around and the total number of particles on the scene increase, representation becomes more accurate. This is due to increased resolution due to additional subdivisions in the surrounding area. For this particular scenario, the dynamic mapping engine initially draws the map using the regular grid and then after 21 simulation seconds pass, it switches the representation to the adaptive grid, due to performance concerns. Furthermore, the mapping engine utilizes three classes at the beginning of the mapping process. It then gradually increases the number of classes until it reaches 9, which is the maximum number of classes for classification considering single-hue coloring schemes. If multi-hue schemes were picked, then the engine would continue increasing the number of classes until it reaches 12. However, in this particular scenario, utilizing multi-hue does not provide meaningful results as the depth information being investigated and varying colors of purple, red and blue does not clearly describe the study area. Regular cells and quadtrees were utilized side-by-side to evaluate potential performance benefits of using adaptive cells.

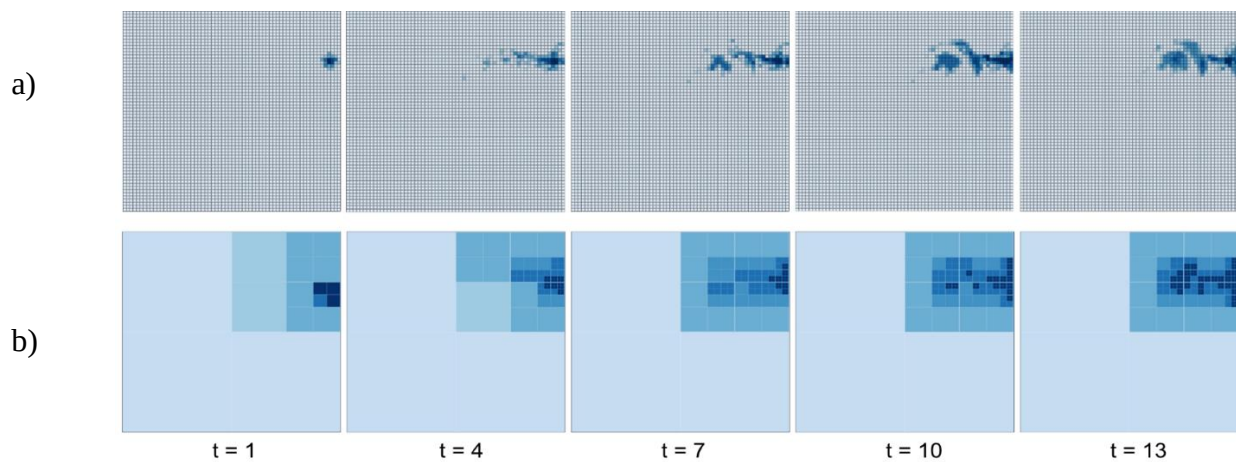


Figure 36. Timeline of flooding using a) regular grids, b) adaptive grids.

Table 2 shows several test environments and average refresh rate for each approach. After 50 runs on the same environments, capped at 300K and 400K particles, adaptive cells improve performance. Consequently, the mapping engine optimizes the performance selecting adaptive cells as required. Nevertheless, if the performance is not a concern, the mapping engine can be instructed to optimize quality.

As the performance comparison provided in Table 2 is showing frame rate per environment, it is possible to estimate how the gridding mechanism affects the overall performance. If the number of cells utilized by both approaches are the same, there should be trivial differences in performance. It is used for realizing different frame rates per environment and grid type. According to our tests, adaptive grid performs worse if depth limit is removed. Nevertheless, the dynamic mapping engine recognizes possible performance issues and therefore, it limits the depth of the quadtree. Consequently, in our tests, no performance issues were realized that would hinder the stability of the simulation.

Upon initialization of the FloodSim, several tests are performed to demonstrate the capabilities of different mechanics utilized in the simulation. As discussed in Section 3.3, there are three major engine components built into the FloodSim: global and local friction model, soil absorption mechanics and cloud/rainfall mechanics. These mechanics are intended to work together and therefore, required files are provided for each component individually. For the hypothetical environment several random parameters are defined. For Corpus Christi, a randomized precipitation data is utilized alongside the storm sewer data file to define absorption areas appropriately. Initially, the local friction model is tested using a hypothetical test environment with hills, and then Corpus Christi. The test was performed with and without local

friction model to demonstrate the purpose of provision of such a model compared to the global friction model utilized by the solvers provided through position-based fluids.

Table 2. Performance comparison of regular grids and adaptive grids under different environmental conditions

Environment	Regular Grids	Adaptive Grids	Difference
Flat neighborhood with embedded buildings 300K particles	30.83 Hz	31.67 Hz	2.72 %
Flat neighborhood with embedded buildings 400K particles	18.47 Hz	19.35 Hz	4.76 %
Canyons with tall hills 300K particles	26.35 Hz	27.41 Hz	4.02 %
Canyons with tall hills 400K particles	17.35 Hz	19.02 Hz	9.62 %
Mountains with noisy terrain 300K particles	22.48 Hz	23.68 Hz	5.33 %
Mountains with noisy terrain 400K particles	15.97 Hz	16.62 Hz	4.07 %

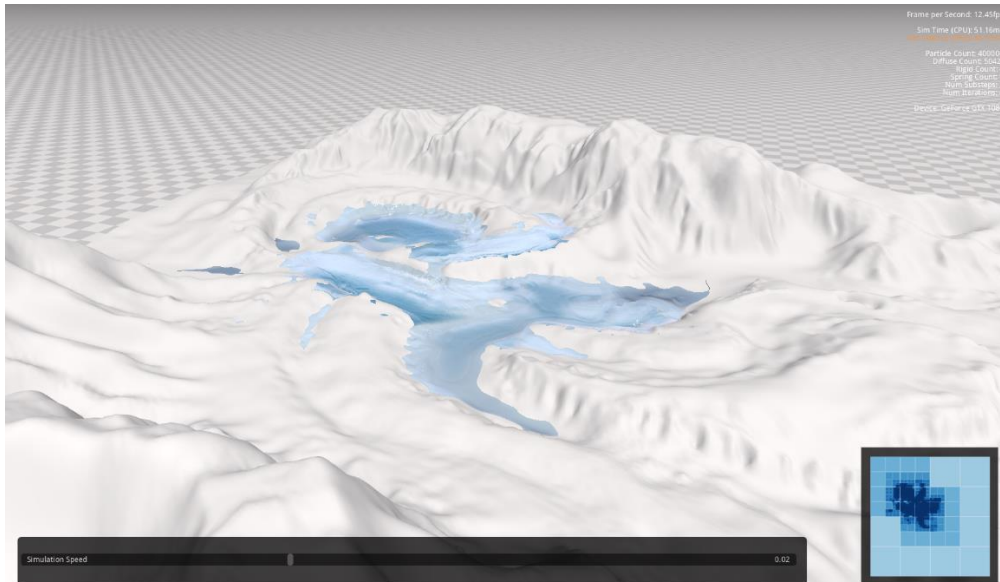


Figure 37. Visualization of flooding without the local friction model for a hypothetical environment with hills, 400K particles.

In Figure 37, with no local friction coefficients, the system must be utilizing the default static and dynamic friction coefficients that the position-based fluids solver utilizes. Consequently, depending on the calibration of these parameters, the water behaves static throughout the simulation. Nevertheless, in real world scenarios, friction must be considered as surface roughness is directly related to how flooding proceeds. As it can be seen in Figure 38, with the added local friction model, the system is able to reason between different types of soils, features and objects. Through realization of the entities, velocity is decreased according to Manning formula. Although the test environment devised is represented with a flat surface, terrain with high vegetation is selected as the friction coefficient. It can be clearly seen that the water is more contained than the test environment shown in Figure 37.

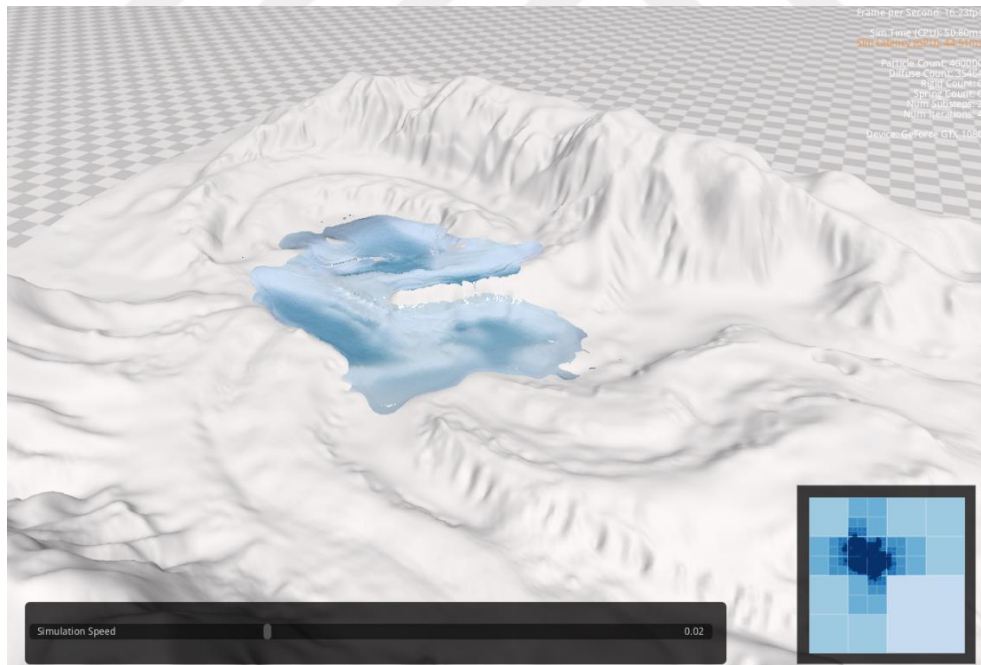


Figure 38. Visualization of flooding with the local friction model for an hypothetical environment with hills, 400K particles.

Another test with 50K particles just focuses on the adaptive 2D depth map to see the differences on the flood spread. Confirming the point about clarity of 2D depth visualization in

contrast to 3D depth visualization, the Figure 39 makes it immediately apparent that friction helps contain particles. This also shows the advantage of quadtrees over regular grids in description of flood spread.

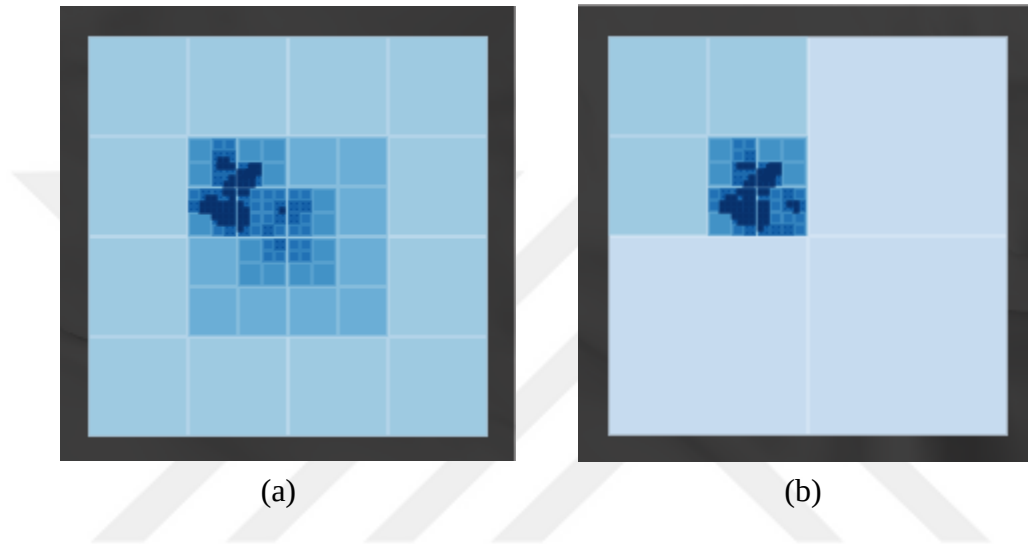


Figure 39. Visualization of the depth in 2D for 50K particles a) with no local friction model and b) with local friction model.

Rainfall and cloud mechanics form the basis of how particles are created for the environment and consequently, they are vital for the system. Cloud configuration file is provided to make the system recognize the placement required and animations, if any, that need to be performed on the scene over some time. Figure 40 shows 4 different clouds created and placed in the simulation environment for particle generation. The parameters of these clouds are also supplied through the cloud configuration file. Creating variety of clouds with differing positions, rainfall intensities and paths, the study area can be further customized instead of overlaying the same precipitation information across the environment.

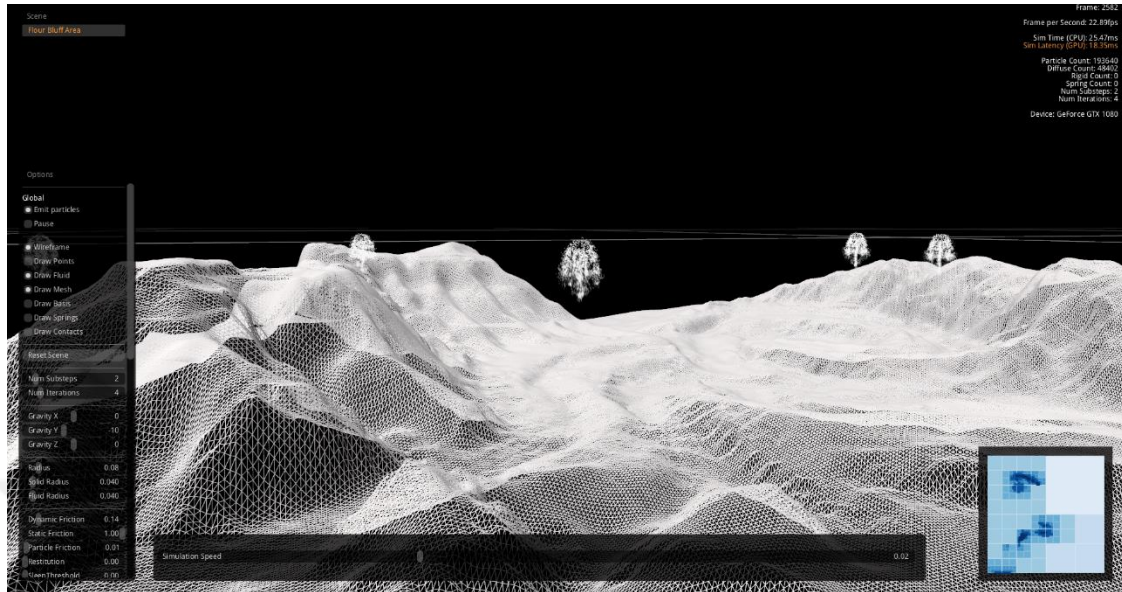


Figure 40. Generated clouds across the study area.

Through embedding cloud width, an overall rainfall amount for the area specified can be provided. Although the simulation would not look as realistic, as a huge cluster of particles are dropped at certain intervals, it still represents the amounts properly and therefore, it is suitable for use when information about local precipitation amounts is not immediately available. An example of such a large cluster is shown in Figure 41. In this test, an extreme rainfall amount that corresponds to 400K particles in the FloodSim are dropped on a selected location through a cloud. Such large areas can be selected with more reasonable amounts of precipitation for interpolation of rainfall across an area. This test uses an extreme case just to demonstrate the capabilities of clouds.

Furthermore, Figure 42 and Figure 43 show the flood animation in 3D and 2D depth map. Once again, the depth map helps visualize flood spread clearly, while 3D provides an additional perspective of depth to visualize the flooding. Consequently, it can be concluded that both approaches are useful in presenting the environment and how flooding progresses over time.

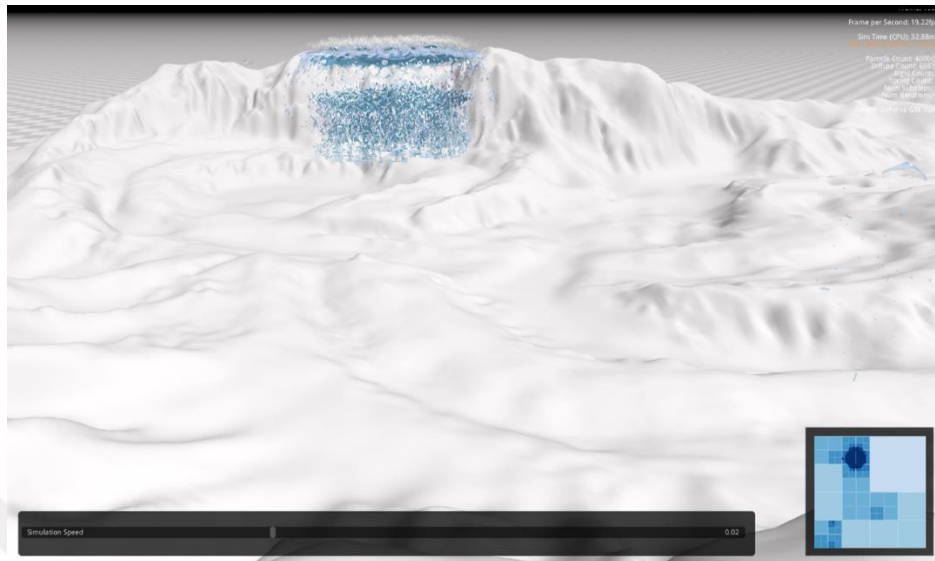


Figure 41. A giant cloud, dropping 400K particles.

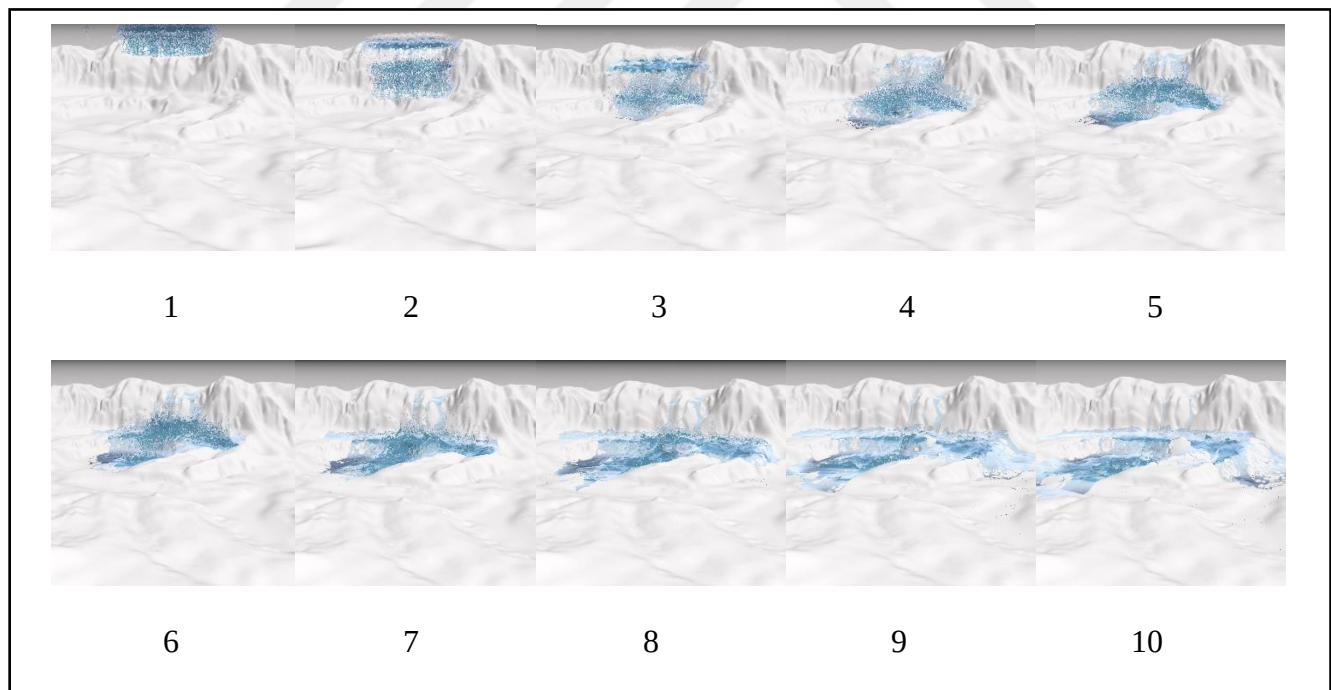


Figure 42. Animation of 400K particles dropped onto the floodplain in 3D.

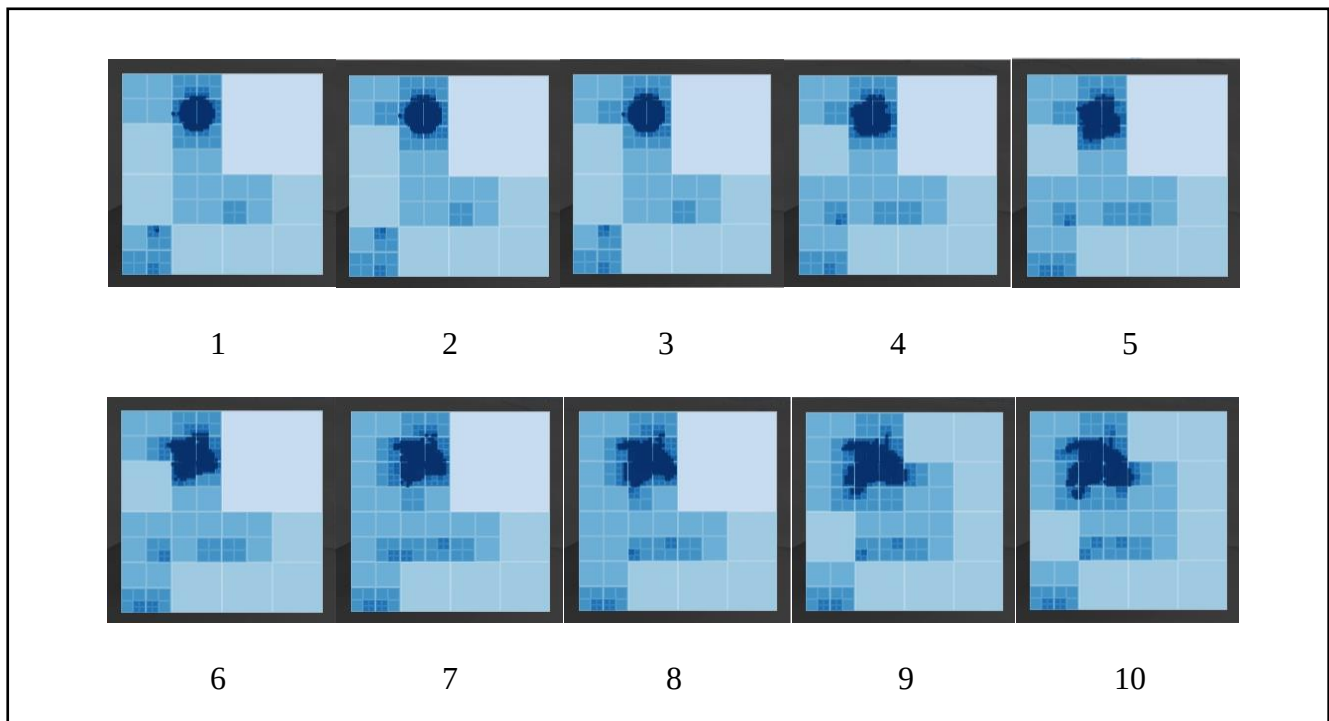


Figure 43. Animation of 400K particles dropped onto the floodplain in 2D.

Although it is apparent that the mechanics implemented for FloodSim are all required to represent the fluid dynamics properly, the system could not proceed with the simulation without absorption mechanics. As the particle removal is a vital factor for efficiency purposes, leaving this mechanic out is not an option to run the simulation for longer times. Consequently, an absorption component is implemented to incorporate storm drain information of the environment. For this purpose, the drainage map of Corpus Christi is acquired, and data is evaluated. This information is then used to create Sewer components, first in the City Maker and then in the FloodSim to represent the areas where absorption occurs. The storm sewer information can be formatted so that City Maker does not need to parse. However, for demonstration purposes both components are utilized together to process the storm drain information. This data file can be converted to OSM format and be placed directly within OSM file for potential use in other applications. Consequently, it is important to follow the hierarchical structure of OSM data files.

Upon evaluation of this information, the City Maker is utilized again to parse storm sewer data as an additional feature to be represented in FloodSim. Although the simulation environment does not directly show the inlets as objects on the scene, they are utilized to absorb waters according to the rates specified. This specification can be provided alongside storm sewer information for every single inlet, if information is available. Otherwise default absorption parameters are utilized as discussed in Section 3.3. Figure 44 shows the City Maker, visualizing inlets as small red squares. Checking the information available at hand about Corpus Christi, it is concluded that the data is parsed and visualized correctly. Although with the current implementation, visualization is not accounted for different flow line elevations and different types of inlets, it could be incorporated into the City Maker through color coding these features. With additional data regarding the inlets, manholes and their placements, further refinements can be done to accommodate a more accurate simulation.

Once the City Maker processes this information the storm sewer data is parsed into the .cm format, as mentioned in Section 3.1. This file is then used to load the absorption coefficients to the FloodSim. Figure 45 and Figure 46 show the differences of flooding concerning the utility of absorption mechanism. Utilizing light rainfall in 3 different spots on the map, particle clustering is tracked. As the rain is generated by rainfall and cloud mechanics, the particles are dropped to the scene using gravity as the force. The scene with the absorption, using the coefficients derived from the previous stage considers the particles generated by rainfall and applying a rate of absorption, removes these particles as they touch the surface. If there is no absorption coefficient set up, then the particles are never removed; water simply flows over the surface until the simulation is terminated.

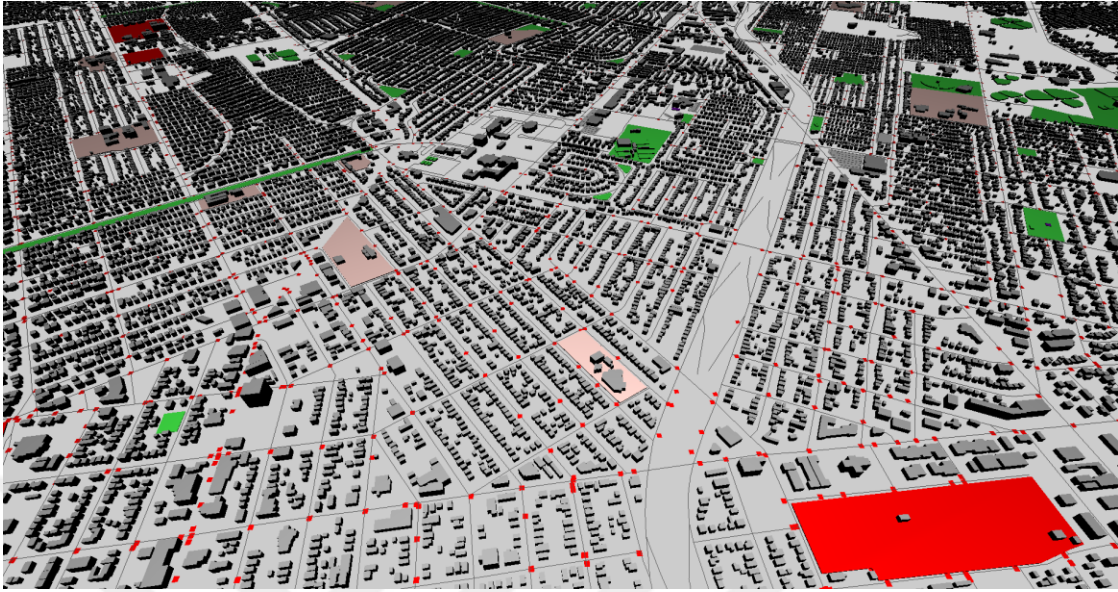


Figure 44. Storm sewer information loaded in City Maker.

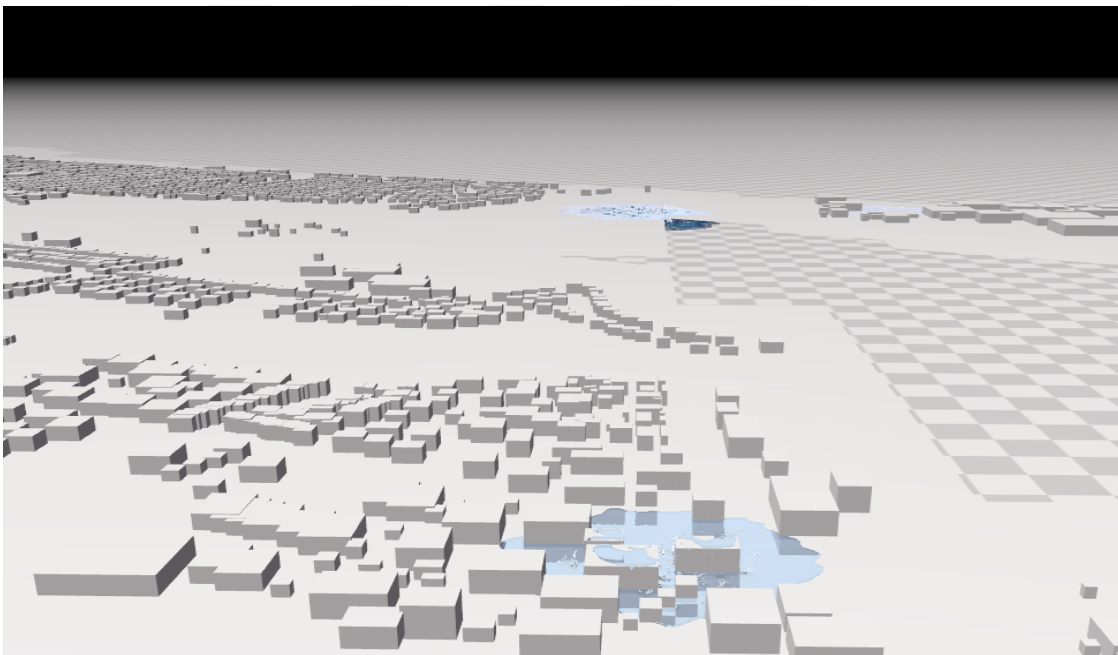


Figure 44. Parts of the environment without soil absorption, 130K particles.

As it can be seen, with soil absorption, the inlets and the passive absorption coefficient, which accounts for evaporation and other types of fluid translation across the environment is made more consistent with a light rain scenario. With no soil absorption, the particle movement can be

achieved through elevation and with an environment like Corpus Christi, it is impossible to only rely on elevation information to move the particles.

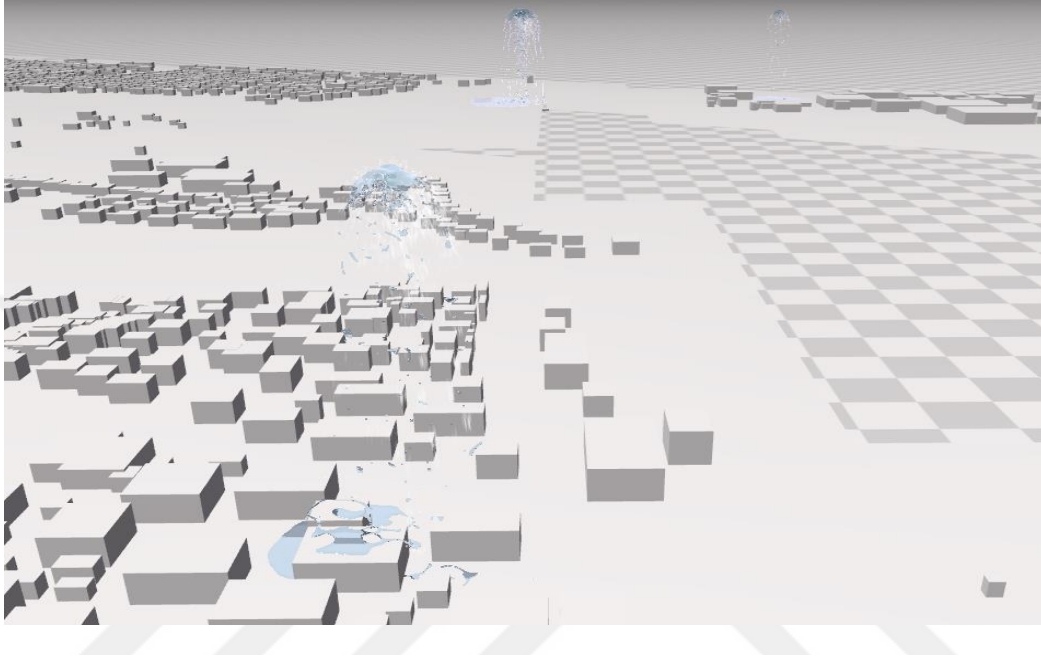


Figure 45. Parts of the environment with soil absorption, 130K particles.

With this approach, it is clear that inlets must be given realistic parameters to absorb water. Further refinements can be implemented if accurate information regarding inlets are available. Moreover, concerning some of the inlets, which might not work to their full capacity in an event of flooding, some randomization factor is utilized to reduce the maximum absorption rate. This implementation can be useful if the city has the information regarding the failure rates of these inlets and for testing different scenarios incorporating these failure rates as the randomization factors.

CHAPTER V: SYSTEM EVALUATION

To evaluate different components developed for the framework discussed in this work, several case studies to produce results similar to the results obtained in the previous section are designed and tested. This section introduces the evaluation metrics utilized for testing the system. Metrics used in most of the relevant work are used alongside additional considerations it was required to evaluate the results properly. Upon introduction of the metrics, two case studies utilizing real data for 3D environment reconstruction and flood simulation are designed and implemented. With the results acquired, other relevant works are compared to this work underlining major differences.

5.1. Evaluation Metrics

There have been various works that investigate how to qualitatively and quantitatively evaluate the products of a work. From a visualization perspective, as certain preferences of the users can be subjective, this has been a very challenging task. Nevertheless, an overview of visualization metrics was provided by Marrero [139] and a detailed survey was introduced by Behrisch et al. [140] to describe visualization approaches that are most prominent in research.

$$\arg (\min, \max) q(\phi|D, U, T) \quad (2)$$

Equation 2 is the quality metric discussed in the work, which is useful for utilization in measuring quality of visualization. In the formula below, to quantify a particular visualization V , an visualization definition ϕ , which is sufficiently efficient, must be computed from a universe of potential instances that would maximize or minimize the quality criterion. In this equation D , U and T stands for data, user and task respectively. Consequently, it is a measure of capturing how effective the instance is bearing the purpose in mind. Furthermore, these metrics are expanded by practical visualizations that are suitable for the industry and the aspect this work is approaching

the problem of visualization. As the focus is on visualization and representation of 3D environment, qualitative results, considering the visuals, components, and organization of the virtual environment are evaluated. As the primary purpose of City Maker is to visualize clearly different features, differing contextual colors are provided alongside a legend for visualization purposes. Similarly, for FloodSim, single-hue and multi-hue color sets are used to visualize the depth of flooding for better recognition of the underlying phenomenon.

According to [139], metrics used for visualization deal with certain important concepts. Firstly, mathematical metrics are utilized to account for data intensity, number of dimensions, occlusion percentage and reference context alongside number of identifiable points. For this purpose, number of data points are tracked as well as providing a reference context. In the context of 3D environment reconstruction, this corresponds to the features illustrated on the map, alongside satellite imagery to provide a reference. Moreover, mapping an environment requires a proper legend to be provided for proper description. For the flood visualization framework, knowing maximum number of particles that can be used for environments of varying size may provide insight about limitations of the proposed method.

$$Frame\ Rate = n_{frame} / s \quad (3)$$

Scalability is another concern for visualizations as time to process and visualize large amounts of data affect the effectiveness of the approach. In the context of City Maker, this would directly correspond time to parse and load OpenStreetMap (OSM) information and to export environmental information as objects. Another significant performance indicator for the visualization aspect is the frame rate using City Viewer. Number of frames per second, as shown in Equation 3, directly corresponds to rendering speed of the environment and consequently, it is used for testing how optimal the visualization of the digital city model is.

These metrics would provide a better understanding about the performance of handling real-world data acquired from OSM. FloodSim also considers the same frame rate metric for performance analysis. Furthermore, as it loads environmental objects produced by City Maker, a metric to estimate loading times for these objects is vital.

Accuracy, expressiveness and similarity are all difficult to quantify metrics commonly found in research utilizing geospatial data [140]. For the analysis of accuracy, evaluating the source data and comparing it with the results would be sufficient. This would correspond to evaluating OSM data and the satellite image for City Maker and evaluating the flood progression or river overflow map as the stability is acquired in simulation for FloodSim. Comparing to real imagery would provide both a sense of accuracy and in our case, as it is mostly a visualization approach, visual similarity. For the flooding purposes, flood extent is computed and then utilizing satellite imagery, inundation area is computed. Utilizing the inundation areas acquired from different models, a comparison is provided. Such an approach provides us quantifiable data to make inference regarding the expressivity and accuracy of these models.

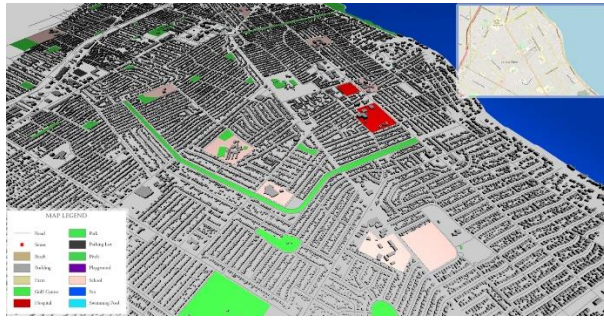
A metric not considered for this dissertation is a user survey, which is significant for visualization research. Many visualization applications, once they are ready to be deployed, provide user survey to account for quantitative aspect of proper visualization. User adoption indicates success in visualization and representation of the environment. And consequently, it aids the results by giving an idea of acceptability about the method that is implemented. As the flood simulation framework is currently not adapted for industry use, user survey is left out as future work.

5.2. Case Studies

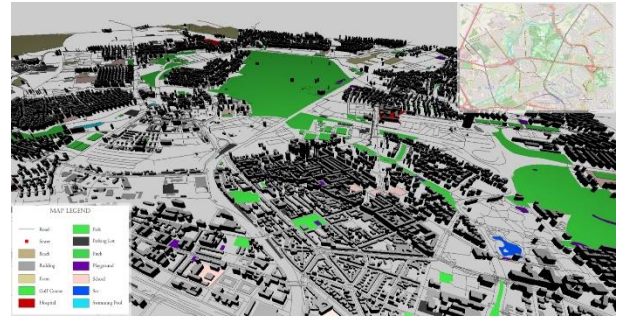
In this subsection, several case studies will be presented to evaluate the capabilities of the proposed implementation. Initially, several cities are exported from OpenStreetMap and the files are evaluated to provide comparison with the results. For each result provided, a legend is attached alongside satellite imagery to allow for evaluation of the data. Finally, performance data about parsing, loading, viewing and exporting these virtual environments are provided. Case study about flood visualization includes Bewdley river overflow that occurred in 2000. We utilize public information about this event and then compare it with the framework described in this dissertation.

5.2.1. 3D Environment Reconstruction

Figure 47 shows visualization of 6 different cities using the City Maker. Comparing the reconstructed environments with the satellite imagery, it can be concluded that these study areas can be visualized with their baseline features using the information derived from OSM information. Nevertheless, it is worth noting that data availability might be a concern for certain cities. For example, according to our tests, Corpus Christi contained sufficient information in its exported OSM file for visualization purposes. With height values provided for buildings and multipolygon areas defined to visually separate feature usage and clear definition of boundaries, Corpus Christi could be properly visualized in 3D. On the other hand, visualization of Izmit was incomplete as most of the building blueprints were missing. Considering San Jose, building information was lacking and therefore, buildings in certain parts of the city could not be visualized. Marseille, Frankfurt and Zagreb contained sufficient information, similar to Corpus Christi, for visualization purposes. Clearly, OSM data must be improved and a lack of data must be accounted for to generalize the visualization approaches.



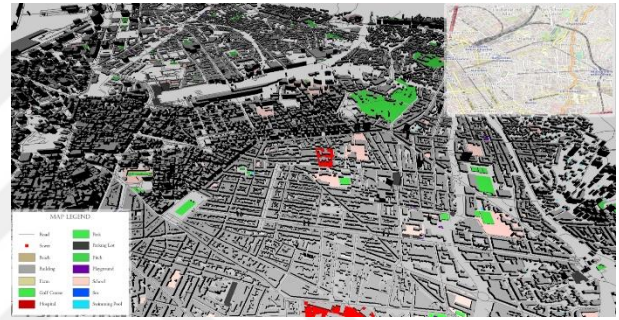
(a)



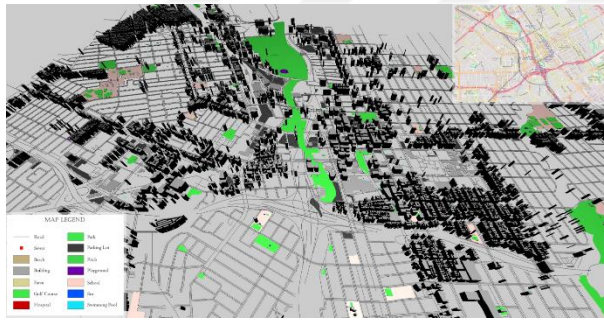
(b)



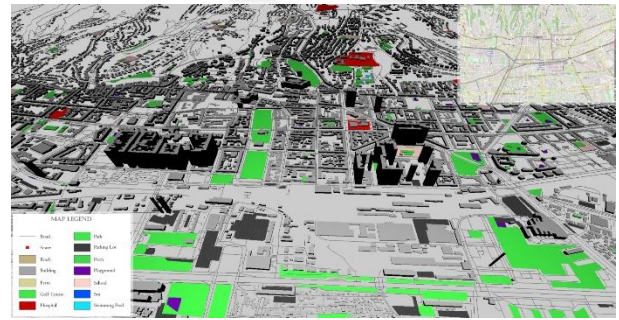
(c)



(d)



(e)



(f)

Figure 46. Visualization of 6 different cities: a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.

Considering the hierarchical structure of standard OSM data files, processing sections of a city takes longer with an increase in size of the study area and number of nodes it is required to process for restructuring feature information. Table 3 is provided to show the advantages of the processed city, in City Maker format, for viewing the environment. For this experiment, only parts of these cities that cover a 25 km² area are selected and compared. For recognition of their size,

significant features are provided in the table. To evaluate performance, a Mac Mini with Intel Core i7-8700B CPU, 32GB of DDR4 -2666 SODIMM memory is used and an eGPU box with Radeon RX Vega 56 installed is connected through Thunderbolt 3 ports. As it can be seen, for the same area, number of features affect parsing time of the OSM data. This directly corresponds to the nodes the OSM data contains and therefore, to alleviate performance issues, potential preprocessing can be performed to remove nodes that are not required for visualization. Once parsing is done, the City Viewer, loads the city information in less than a second for the three examples provided. Parsing and viewing times are averages after 100 executions and it can be concluded that it provides a reliable and fast loading experience. For the cities with a larger number of features, such as Marseille, France in our case study, it took on average around 42 minutes of parsing yet less than a second of loading. This is due to the node structure of OSM data files as they are relationally attached to components and require further processing. However, this hierarchical format helps compile the city information properly and therefore, it may be required the way it currently is. Trimming can be utilized if certain features are not going to be visualized. By removing some features that may not be required for a particular study, performance can be improved as the City Maker processes less information. Average frames-per-second is captured for the study areas. It is understandable that larger the number of features, worse the performance becomes and consequently, it would be wise to optimize rendering for larger areas.

For each one of these cities, the OSM file structure is investigated and number of nodes, ways and relations are tracked to reason about performance. As OSM has a hierarchical structure, nodes, ways and relations affect the performance of parsing a city. Although ways are more complicated than nodes, and relations are more complicated than ways due to the way they are nesting the latter, it is important to recognize that they are semantically associated constructs

focusing on structuring the nodes appropriately. Table 4 shows the number of components for each of these cities. Considering the number of nodes, we have for these cities, it is expected that Marseille should be the most demanding city in terms of processing, while Izmit should be the easiest to handle. According to the Table 3 and comparing it with Table 4, it is clear that the expected result is acquired. The exception was Frankfurt as it had lesser nodes than Corpus Christi, yet it took on average around 17 seconds longer to parse the OSM file. This is potentially due to the number of Ways and Relations the city has. As ways contain nodes and relations contain ways to be processed, it takes longer to parse semantically related constructs. Consequently, it is a vital optimization to remove unwanted relations and ways to reduce the impact of the nodes on processing time.

Table 3. Comparison of number of features, time it takes to parse OSM files and load CM files, to render the city models of a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>Buildings (features)</i>	22327	17915	689	71694	5716	17648
<i>Roads (features)</i>	971	5705	3682	7318	11184	9282
<i>Parks (features)</i>	31	61	34	27	41	139
<i>Hospitals (features)</i>	2	2	6	7	0	13
<i>Schools (features)</i>	15	29	28	131	26	48
<i>Parsing OSM (s)</i>	166.21	183.91	8.17	2532.42	120.54	338.52
<i>Loading CM (s)</i>	0.31	0.33	0.24	0.67	0.27	0.35
<i>Average FPS (Hz)</i>	21.36	24.82	214.69	5.29	45.32	21.80

Table 4. Comparison of number of nodes, ways and relations found in OSM data files of a) Corpus Christi, TX, USA, b) Frankfurt, Germany, c) Izmit, Turkey, d) Marseille, France, e) San Jose, CA, USA, f) Zagreb, Croatia.

	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
<i>Nodes (count)</i>	128876	123336	31641	388177	115011	171686
<i>Ways (count)</i>	23646	27265	5477	85794	24066	33238
<i>Relations (count)</i>	57	672	53	1071	631	811

Figure 48 shows the relationship between number of nodes OSM file contains and time it takes for City Maker to parse the file. As discussed earlier, a linear relationship is found except for the case of Frankfurt.

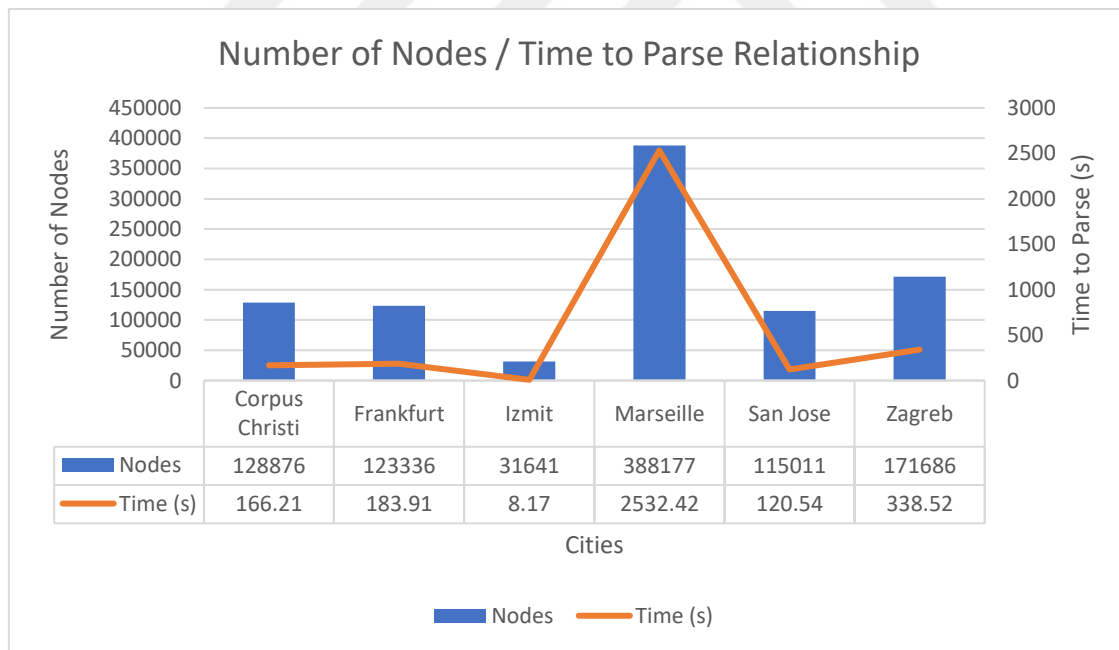


Figure 47. Chart of the relationship between number of nodes and time it takes to parse an OSM file.

City Viewer is utilized to view these cities and track frame rates as the environment is investigated by zooming in and out, rotating the camera and viewing complex sections with these

cities. An average frame rate is then computed and provided to evaluate the performance of rendering large cities. Similar to the relationship chart, shown previously for number of nodes and time to parse OSM files, Figure 49 shows the relationship between total number of features and average frame rate for each city

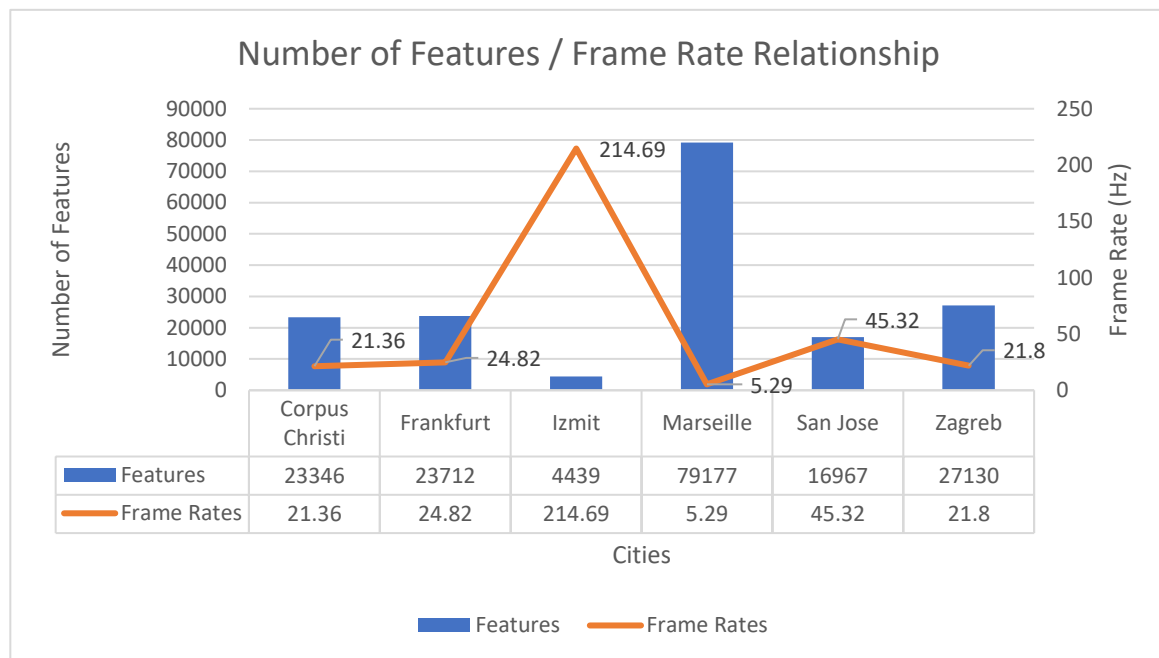


Figure 48. Chart of the relationship between number of features and frame rates.

Number of features and frame rates have an inverse relationship, which is expected. Marseille having an average frame rate of 5.29 Hz is not satisfactory for real-time visualization and simulations. Nevertheless, fully detailed 3D visualization is not usually proceeded with to display around 72,000 buildings at the same time and therefore, a rendering approach that would visualize smaller sections should be sufficient for a variety of applications.

City Maker can embed simulation parameters into the digital city model it generates. To incorporate utility information, the drainage map of Corpus Christi is acquired from the city of Corpus Christi, and data is evaluated. Figure 50 shows parts of the city and red dots shows inlets.

This information is then used to create Sewer components in the City Maker to represent the areas where drainage occurs. Initially, the data that the city provided in a spreadsheet was manually converted into OSM node format to allow the City Maker to immediately parse the information. Therefore, it can read longitude and latitudes alongside each sewer inlet identifier to place them in the digital city model. This model is then exported to be used for flood simulations. Consequently, no further processing of data is required at the simulation level; the required simulation parameters are parsed by the City Maker in the 3D city reconstruction phase.

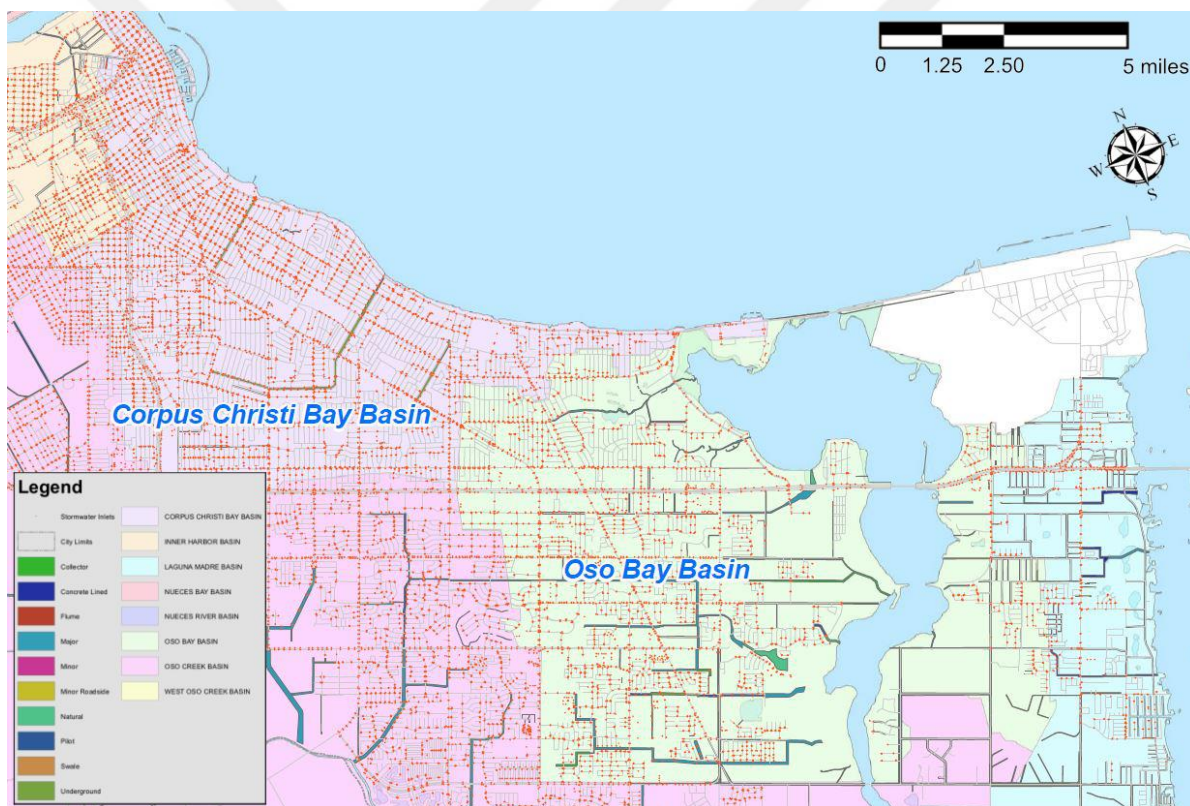


Figure 49. Parts of the city of Corpus Christi with storm sewer inlets.

Figure 51 shows parts of Corpus Christi with inlets placed according to the data provided to the City Maker. Comparing it to the inlet data acquired from the city of Corpus Christi, the placement is accurate. Further improvements can possibly be done by incorporating more

information into every single storm drain, their elevation, rate of absorption and similar parameters. If the drain information is available in any given city, it can be converted to the OSM format as utilized in City Maker and be incorporated into the 3D urban environment reconstruction process for added details.



Figure 50. Visualization of storm drains in Corpus Christi

Upon placement of these storm inlets, FloodSim receives the digital city model, considers the placement of inlets for generation of suitable absorption coefficients, and then performs the simulation. For City Maker, inlets are placed simply for passing the information to the next phase, that is simulation, and for visual representation purposes. These inlets, using their geographic coordinates, can be rearranged, manipulated and deleted. Furthermore, new inlets can be placed to demonstrate the effects of additional inlet placement in a given area.

For a comparison with a similar 3D environment reconstruction tool, a popular software called CityEngine 2019 by Esri is utilized. This software has been used in the industry and the academia for a variety of purposes including urban planning and analysis, urban studies and other work that would incorporate a 3D urban environment reconstruction.

Table 5. Comparison of features between Esri CityEngine and City Maker.

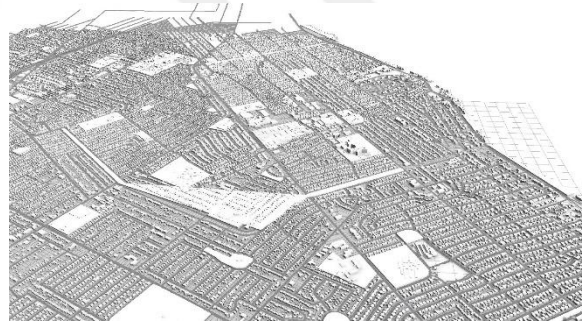
	<i>Esri CityEngine</i>	<i>City Maker</i>
<i>OSM feature parsing</i>	✓	✓
<i>Parsing any large area</i>	✗	✓
<i>Default coloring scheme</i>	✗	✓
<i>Ability to change color</i>	✓	✗
<i>Add/delete features</i>	✓	✓
<i>Edit features</i>	✓	✗
<i>Simulation</i>	✗	✓

Table 5 shows feature comparison between Esri CityEngine and City Maker. Although CityEngine is able to individually create, change and delete individual features after the data is loaded, City Maker provides default mass feature coloring and ability to parse any large area, even though performance is sacrificed. Furthermore, simulation capabilities of City Maker would make it preferred option if needed. CityEngine provides a sandbox that the user can create, edit and experiment with the data with no simulation considered. It is a useful software for 3D urban environment reconstruction. City Maker, providing similar features, can parse larger areas and utilize simulation parameters making it a suitable choice for relevant studies.

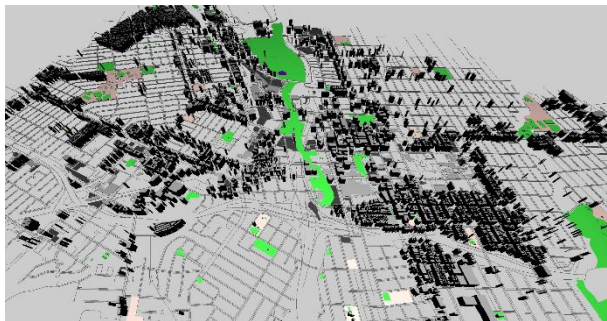
Figure 52 shows 6 cities that City Maker generated with comparison to the models generated by Esri City Engine. For generation purposes same OSM data file is utilized. The reason that comparison could be performed with only Corpus Christi, TX and San Jose, CA is because the other cities could not be parsed by the CityEngine as their extents were too large to be parsed. According to these visuals, similar models are generated by both software. City Maker utilizes default coloring, making features more apparent. Moreover, certain visual anomalies can be seen on the CityEngine side due to triangulation. An advantage of City Maker is that it can average building heights in a given OSM file from the available building heights and extrude the remaining building blueprints with similar heights. That's the reason in (d), San Jose looks empty. As the building heights are not provided in the OSM file, CityEngine ignores them, while City Maker averages the known data and generates buildings with similar height.



(a)



(b)



(c)



(d)

Figure 51. Comparison of generated digital city models: a) City Maker – Corpus Christi, b) CityEngine – Corpus Christi, c) City Maker – San Jose, d) CityEngine – San Jose

A similar test of performance is conducted to compare differences between these two implementations. Table 6 shows the comparison of performance between City Maker and CityBuilder. Considering the values acquired from these tests, it is clear that CityBuilder has an efficient OSM parsing approach. Nevertheless, saved files were loaded much faster using City Maker. Consequently, it is fair to say that while City Maker excels in loading parsed environments, it requires additional optimizations for parsing OSM data. Furthermore, average frame rate was not computed for CityBuilder as it does not offer a simulation feature accessible through the software. Another interesting difference is that the environment of Corpus Christi took longer to parse and load for the City Maker compared to San Jose. Nevertheless, CityBuilder parsed and loaded San Jose slower than Corpus Christi. Considering the number of features each environment contains, City Maker parses the buildings slower, while CityBuilder parses and loads roads slower.

Table 6. Comparison of performance between City Maker and CityBuilder for cities of Corpus Christi, TX and San Jose, CA.

	<i>CM – Corpus C</i>	<i>CB – Corpus C</i>	<i>CM – San Jose</i>	<i>CB – San Jose</i>
<i>Parsing OSM (s)</i>	166.21	27.57	120.54	78.14
<i>Loading CM (s)</i>	0.31	4.42	0.27	13.27
<i>Average FPS (Hz)</i>	21.36	N/A	45.32	N/A

5.2.2. Flood Simulation and Visualization

For testing flooding scenarios with FloodSim, an environment generated from City Maker is required. This is correct for urban environments as buildings, roads and features need to be extruded from it before FloodSim can utilize the terrain. For environments that have negligible

number of housing, or if performance is a concern, then SRTM data with 1 arc-second resolution can be utilized to provide the elevation information for the FloodSim.

For a river flood scenario, Bewdley flood of 2000 is selected for the case study as the data was available, and results from other work were already published to provide a baseline for comparison. It is noted that Bewdley was one of the towns that got the effect of flooding which occurred in the Fall 2000, along the Severn Valley. The town was flooded three times within six weeks and therefore, some measures needed to be taken to improve flood defenses, emergency management efforts and flood warning systems. Figure 53 shows the town flooded.



Figure 52. Photo showing Bewdley flooding.

For flooding visualization, only a section of Bewdley was selected and SRTM with 1 arc-second resolution is utilized for providing the elevation information. As it is a river flooding scenario, threshold is arranged in such a way that a proper depth is acquired for visualizing the river. Upon calibrating the height values to represent flooding, arrange particle sizes for the

simulation and constrain the river water, the environment is placed in the scene for further arrangements. Figure shows the exported OBJ of the terrain that will be used in FloodSim.

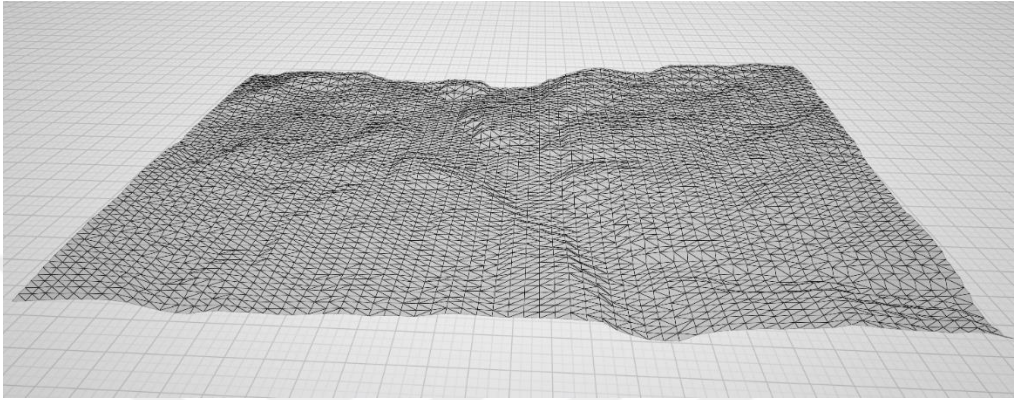
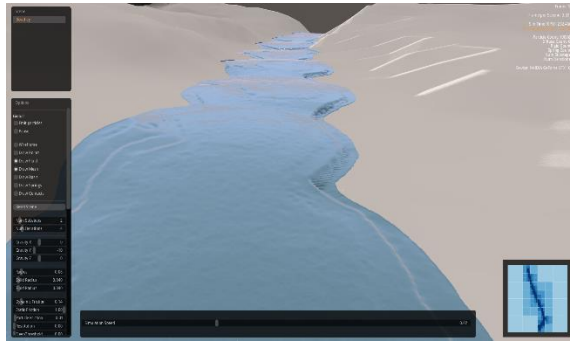
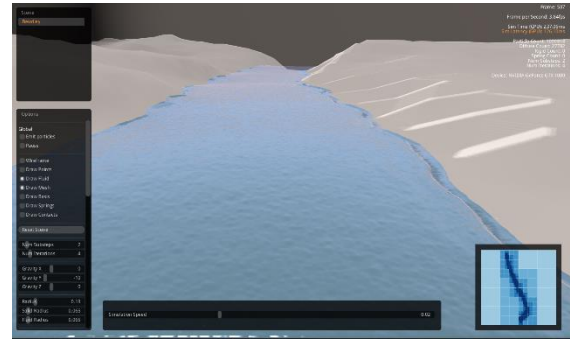


Figure 53. Terrain generated from SRTM and passed to FloodSim

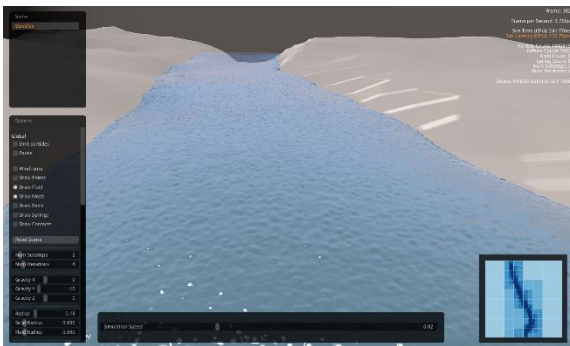
Once the terrain is imported into FloodSim, fluid properties must be calibrated to match with the scenario being implemented. For the extent of the area selected for Bewdley river flooding scenario, it is determined that particle radius to be set to 0.18 units. Smaller particle radius helps with higher resolution, yet memory limitations may require the environment to be calibrated through particles with larger radius. For testing the radius values considering expressivity in the study area, radius of the particles was altered from 0.08 up to 0.28, incrementing by 0.05 units each turn. Each time 1 million particles are generated and visualized. Figure shows the differences between these 5 cases with differing particle radii. This experiment shows that depending on the extent of the area and scenario to be visualized, particle radius must be calibrated to accommodate proper visualization of flooding. As it can be seen, using 1 million particles with 0.08 units of radius as in the case of a) was not sufficient to fill the river. Cases of b) and c) look appropriate and suitable to represent water with further calibration. d) and e) provided too large particles for achieving visual realism.



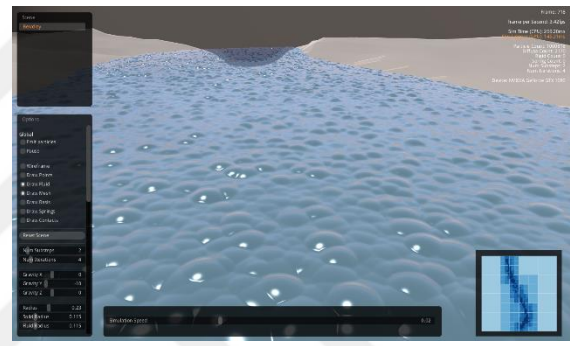
(a)



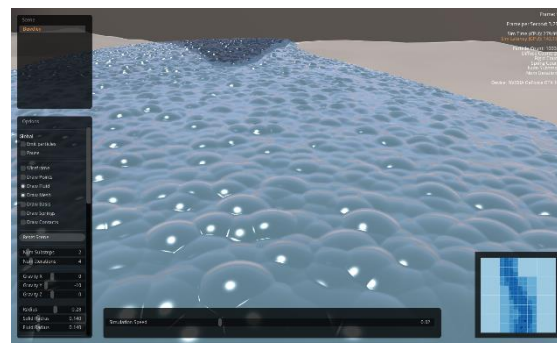
(b)



(c)



(d)



(e)

Figure 54. 1 million particles generated in the river bed with particle radii of: a) 0.08 units, b) 0.13 units, c) 0.18 units, d) 0.23 units, e) 0.28 units

Considering the performance of particle generation and limitation FloodSim currently has, Figure shows that with constant particle generation for 5 minutes, 3.5 million particles are generated and 23.8GB of RAM is used up. As memory would be required to continue the processes

within the solver and compute interaction between particles, it would not be feasible to work with such a setup. Therefore, scalability is an important concern for simulation of larger areas. Less particles mean less details in description of flooding and therefore, errors are expected. Such errors must be accounted for through provision of precise details about the study area and utilization of more capable CPUs and GPUs.

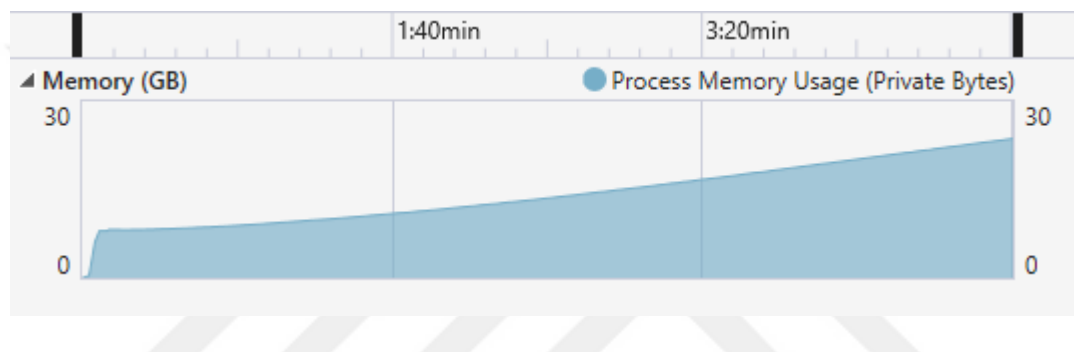


Figure 55. Usage of memory within 5 minutes of simulation with constant particle generation, each particle having 0.18 units of radius.

For testing visual realism, Bewdley flooding case study is acquired and data is utilized to generate similar conditions. Furthermore, the work described in [141] is considered for comparison as they directly provide results for this particular flooding event. With the choices of simulation parameters, around 1 million particles are required to get the flood to similar levels as in the flooding. Furthermore, 300,000 particles are required to simply fill the riverbed for proper representation of the river. Upon stabilization of the particles, rainfall is provided through clouds that are interpolated over the river. 14 clouds distributed at equal distances above the riverbed produce sufficient rainfall for the event to be recreated. Figure 57 shows the map found for Bewdley showing river flooding event, possible flood extent and flood defenses. A comparison

with the expected flooding extent of the area and the results produced by FloodSim can show the feasibility of the visualization framework it provides.

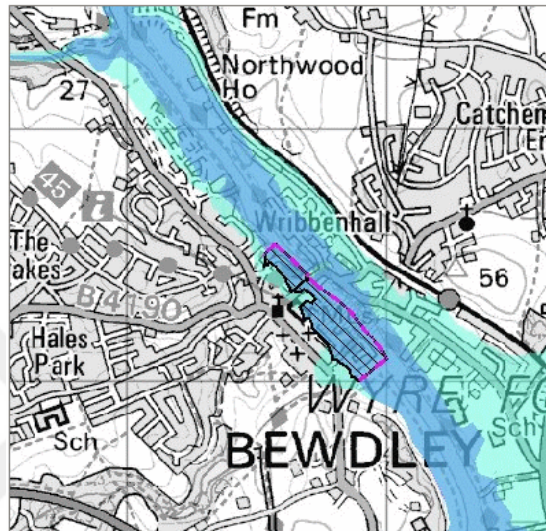


Figure 56. Map of Bewdley, showing river flooding, extent of extreme flooding scenarios and flood defenses

An animation of the river flooding is provided in Figure 58. Flooding is captured around every 100,000 particles of increase in water amount and lasted a total of 220 seconds. Once simulation reaches approximately 1 million particles, the amount of water required to demonstrate the flooding is placed into the riverbed. Additional time is reserved for waiting as the simulation reaches to a stable state. Upon visual investigation of Figure 57 and the result generated by FloodSim, it can be seen that the flooding area is visualized similarly. Nevertheless, there are some parts of the flooding that were not visualized accurately. This is partly due to flood defenses, which our method does not account for, and due to resolution of SRTM, which FloodSim works with. Improving the DEM model and providing an accurate river bathymetry would improve results. Nevertheless, performance is still a concern if the current configuration of CPU and GPU are to be utilized.

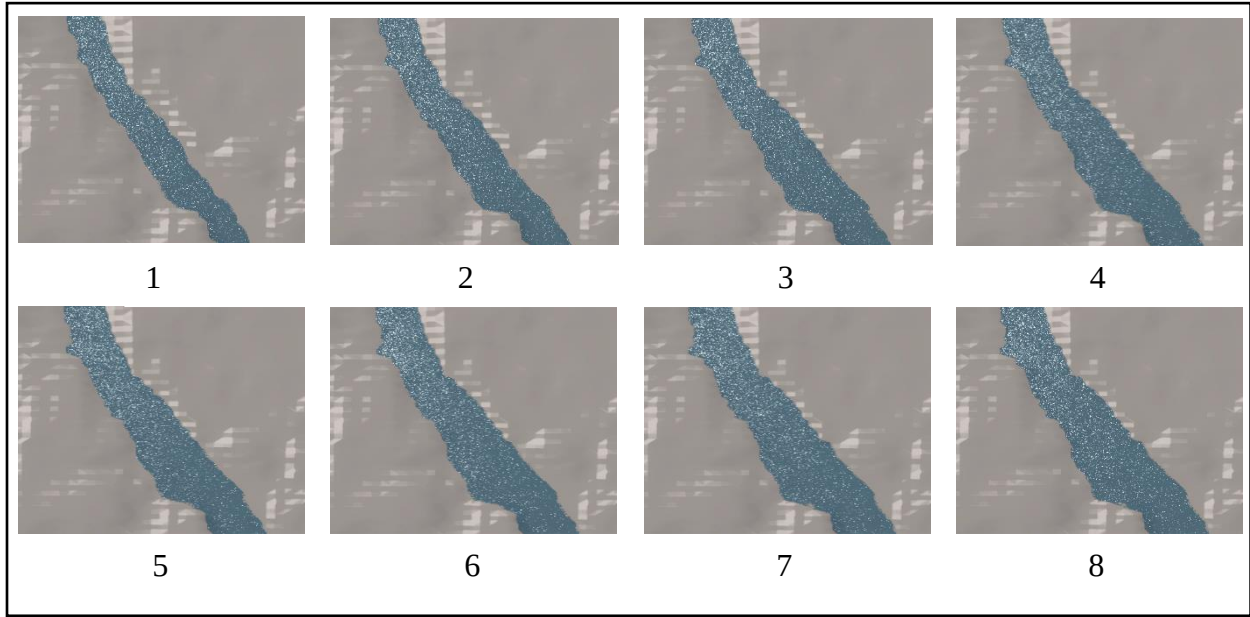


Figure 57. Animation of river flooding with particle radius 0.18 units and a total of 1 million particles

Figure 59 shows a comparison of river sections upon flooding. Visually, we can reiterate that FloodSim performs well to represent the extent of flooding. The flood model in [141], which utilizes Bewdley flooding as a case study, also performs similarly. An advantage of FloodSim is that it is utilizing Lagrangian description of fluid flow, which allows the user to investigate individual particles and still performing fast enough for viewing the occurrence as it progresses.

For these three outputs, same sections of the river flooding from the bridge at Load St. to Kidderminster Rd. are extracted and inundation areas are calculated. These areas are then compared side-by-side to clarify any potential differences. Table 6 shows the comparison between three calculated inundation areas. Our model computes and visualizes the inundation area closer to the real data acquired for Bewdley. The difference computed is around 7%, while the flood model considered for comparison had an 18% difference. They have similar visual realism, yet the extent of flooding was captured better through FloodSim. This shows the feasibility of position-based fluids in description of surface flow and determination of flooding extent. Considering that

SRTM data with a resolution of 30m was used to generate the terrain, it is realistic enough to be considered as a basis of a developing Lagrangian-based flood simulator. Further experiments can be conducted to calibrate the model further and comparisons can be made with more recent river flooding models to compare visual fidelity.

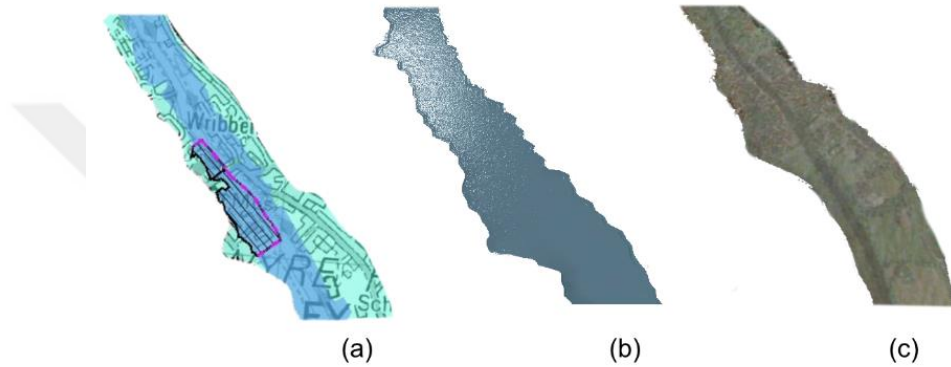


Figure 58. Comparison of flooding sections a) Bewdley data, b) FloodSim, c) Flood Spreading Model [140]

Table 6. Comparison of inundation areas and differences of a) Bewdley data, b) FloodSim, c) Flooding Model [141].

	Inundation Area (m ²)	Difference (m ²)	Error (%)
<i>Bewdley Flood Data</i>	12,224		
<i>FloodSim</i>	11,395	-829	-6.78%
<i>Flooding Model [141]</i>	10,157	-2067	-16.90%

CHAPTER VI: CONCLUSION AND FUTURE WORK

Availability of GIS data sources and advanced techniques implemented for simulation and visualization of physical phenomena has been driving the researchers to investigate alternative ways to describe and represent the data. Through provision of efficient and visually pleasing models to evaluate and simulate the information, more context can be provided to practitioners for possible use in research and industry. The framework described in this work provides a multi-procedural workflow to demonstrate the usability of position-based fluids to simulate and visualize flooding events. Through implementation of a 3D city information extractor, a 2D variable spatial resolution depth mapping engine and a 3D flood simulation framework, an efficient and customizable toolset is provided for visualizing environments and simulating flooding events.

6.1. Conclusion

City Maker utilizes OpenStreetMap (OSM) data to provide a virtual environment to visualize and evaluate the study area. By providing a drawing method utilizing a few basic geometric primitives and contextual colors or textures, a data extractor and virtual environment generator is implemented. This information can then be parsed and sent to the simulation engine to be used in flood simulation scenarios. Although there are several prominent desktop, cloud-based and WebGL solutions to handle urban environment reconstruction, City Maker provides some unique advantages about extent of visualization and OSM data restructuring. Similar to the results obtained by [6, 94, 142] it can be concluded that 3D visualization of virtual cities is very important for recognizing valuable information and patterns regarding the study area. City Maker [27] offers the following advantages:

- 3D urban environment is generation is automated using OSM data, which is readily available for numerous cities. (Figure 47)

- Compared to similar industrial software, City Maker can visualize larger extents and provide clear visuals through default feature coloring. (Figure 52)
- The environment, once analyzed and parsed, is converted into the .cm format, which carries simulation environment information of significant features. Through this format, even complex environments were demonstrated to load in less than a second. This provides efficient preprocessing of city information for the simulation environment. (Table 1, Table 3)
- Simulation-specific features, such as storm drain information, Manning's coefficients and other simulation parameters, which cannot be found in OSM data can be processed and visualized using City Maker. This offers a complete solution of generation of the environment, implementation of simulation parameters and visualization of the features. (Figure 50, Figure 51)

In this work, a flexible depth mapping engine, which can be used to visualize depth information for 3D Lagrangian simulations is presented to provide better insight for flooding scenarios. The engine having an adaptive resolution provides better efficiency than standard grid-based implementations. Making the choice to select the representation dynamically, shortcomings of both regular grids and adaptive grids are alleviated. Moreover, by utilizing position-based fluids as the underlying simulation method [24], advantages of position-based dynamics is acquired, which provides better simulation stability and efficiency over traditional SPH [25] approaches.

2D visualization of flooding utilizing adaptive spatial resolution [31] is useful for multiple reasons. Firstly, earlier works of [6] and [109] provide insight about the benefits of utilizing 2D visualization alongside 3D as they underline that 2D representation is more maintainable and human-readable and therefore, it helps alleviate some of the complexities of 3D simulations.

Moreover, through utilization of quadtrees and implementing the mechanism for the system to decide on which representation to choose, efficiency, as well as detail on demand are provided. Providing a depth mapping mechanism helps 3D Lagrangian simulations to describe particle clusters and hence, water levels in the case of flooding, more precisely. To summarize:

- A flexible mapping engine, which can be used to visualize depth information using adaptive spatial resolution is provided. Through some examples it is demonstrated that 2D visualization of the environment helps recognize certain patterns clearer than 3D representation. (Figure 36, Figure 39)
- The underlying mechanism of transitioning between regular grids and adaptive grids help alleviate performance issues in complex simulations. Furthermore, transitioning back to regular grids when the situation arises, more detail can be provided, which provides clearer contours on the 2D map. (Table 2)
- Through the depth representation approach described, particles in Lagrangian simulations can be tracked and depth can be mapped effectively. (Figure 42, Figure 43)

FloodSim was designed, bearing in mind that several parameters, such as rainfall, soil absorption and water drainage to simulate flooding scenarios. This work provides a new flood visualization and simulation framework with position-based fluids by providing the benefits of position-based dynamics [24] to contain particles and prevent usual neighborhood problems [30] that occur in SPH [25]. Such a framework, with further calibration to incorporate a quantitative flood model, can be utilized to test different scenarios regarding flooding to visualize a city's or an area's resilience against flooding events. Considering that empirical and many hydrological models provide a gridded approach to describe the fluid flow, our model utilizes a particle based

Lagrangian approach, which is useful for experimental results. Although it can be underlined that Eulerian models visualize continuum properly, Lagrangian models, with more advanced GPUs help experiment with the study area utilizing particle interaction. Consequently, this work can be used alongside other frameworks to gain a different perspective regarding the study area. Further refinements in the hands of practitioners can be performed to describe local dynamics of an environment more precisely. FloodSim provides the following advantages:

- As position-based fluids are more suited for real-time simulations than some traditional SPH models, provision of this framework allows for real-time interaction through efficient particle representation. (Figure 40, Figure 41, Figure 42, Figure 43)
- Manning's coefficients provide additional local friction model to position-based fluids' already existing friction coefficients, helping with further customization and enhancement of environmental representation. (Figure 37, Figure 38)
- Cloud and rainfall mechanics along with the soil absorption provide dynamic representation of the study area. (Figure 40, Figure 45, Figure 46)
- Utilizing environmental data through SRTM elevation with 1 arc-second resolution, river flooding simulations can be performed with realistic representation of the study area. (Figure 59, Table 6)

6.2. Future Work

As there are variety of assumptions utilized throughout this work, these areas offer opportunities to continue with future work. For the city visualization aspect, considering the coloring schemes utilized, more contextual texture provision would yield more realistic results. Currently, as the environment is demonstrated using a limited set of model files and coloring for

the City Maker, visual realism is not directly incorporated into the implementation. It is also worth underlining that although the underlying mechanism of localization with regional features is implemented, lack of artistic content and an appropriate texture visualization engine prevents the framework from utilizing this feature. Future work should incorporate implementation of such a visualization engine to better represent the study area.

Another consideration for the City Maker is to integrate the environmental description to a web-based implementation for portability. Although it is mentioned that performance is a concern, an efficient implementation should be attempted, which could provide collaboration and automated generation of cities online. Through utilizing desktop version of the City Maker and uploading and storing the .cm files to the online viewer, users can work on their virtual environments with ease.

Although the 2D visualization system is tested using several different environments, urban environments such as towns and cities present an additional challenge as the concept of water depth and terrain height must be considered in a different context. Even though the current implementation of the system considers the building, car, tree and similar entities separately, more testing needs to be performed to eliminate any potential inconsistencies regarding computation of depth and height. An interesting challenge would be to consider population to determine the risk factors of such flooding. As too much flooding in an empty area does not cause risks such as flooding in the main parts of the city, it is worth investigating additional layers to incorporate more context and data into the simulation framework.

As underlined earlier, many of the environmental parameters are loaded through the component reader. These parameters are not altered after the initial processing. Simulating a fully dynamic environment with shifting wind direction over time, intensifying the rain, considering

soil erosion and soil absorption could yield interesting results. Such scenarios should be implemented and experimented with in the future to provide a more precise simulation framework for utilization in real-world scenarios such as risk assessment and emergency management. Upon description of a more detailed flood model for specialized purposes, recent flood models in the industry can be compared with FloodSim to recognize advantages and disadvantages of utilizing position-based approach.

Further investigation must be done to clarify scalability issues of the flood simulation framework. Although the data can be parsed and large city models can be reconstructed with the current approach, SRTM data provides low resolution elevation information that prevents us from proceeding with a higher resolution simulation. Moreover, providing the framework utilizes lower resolution elevation map, instead of visualizing and simulation sections of cities, city-wide or region-wide simulations will be incorporated in the future work.

The framework must be tested by the experts in the field for usability, expressivity and feasibility. Through provision of a completely modular version of the framework, experts can test a variety of scenarios and provide input regarding the advantages and disadvantages of using the framework. Furthermore, by conducting a user survey about clarity and visual representation, user preferences can be accommodated to improve the visual aspect of the framework.

REFERENCES

- [1] S. T. Ashley and W. S. Ashley, "Flood Fatalities in the United States," *Journal of Applied Meteorology and Climatology*, vol. 47, no. 3, pp. 805-818, 2008.
- [2] J. Herget, A. Kapala, M. Krell, E. Rustemeier, C. Simmer and A. Wyss, "The millennium flood of July 1342 revisited," *CATENA*, vol. 130, pp. 82-94, 2015.
- [3] R. Giordano, A. Pagano, I. Pluchinotta, R. O. del Amo, S. M. Hernandez and E. S. Lafuente, "Modelling the complexity of the network of interactions in flood emergency management: The Lorca flash flood case," *Environmental Modelling & Software*, vol. 95, pp. 180-195, 2017.
- [4] G. Zhao, X. Zheng, Z. Yuan and L. Zhang, "Spatial and Temporal Characteristics of Road Networks and Urban Expansion," *Land*, vol. 6, no. 2, 2017.
- [5] R. J. Dawson, "An agent-based model for risk-based flood incident management," *Natural Hazards*, vol. 59, no. 1, pp. 167-189, 2011.
- [6] S. Van Ackere, H. Glas, J. Beullens, G. Deruyter, A. Wulf and P. De Maeyer, "Development of a 3D dynamic flood WEB GIS visualisation tool," *International Journal of Safety and Security Engineering*, vol. 6, pp. 560-569, 2016.
- [7] H. Chang-Kyu, Y. Seon-Woong, L. Kyung-Ho and J. Hun, "A study on thermal behaviour prediction for automotive electric relay based on CFD," in *2018 24rd International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, Stockholm, Sweden, 2018.

- [8] M. H. Siemann and S. A. Ritt, "Novel particle distributions for SPH bird-strike simulations," *Computer Methods in Applied Mechanics and Engineering*, vol. 343, pp. 746 - 766, 2019.
- [9] J. Chen, C. Peng and F. Lien, "Simulations for three-dimensional landmine detonation using the SPH method," *International Journal of Impact Engineering*, vol. 126, pp. 40-49, 2019.
- [10] J. Xing, H. Wang, K. Luo, S. Wang, Y. Bai and J. Fan, "Predictive single-step kinetic model of biomass devolatilization for CFD applications: A comparison study of empirical correlations (EC), artificial neural networks (ANN) and random forest (RF)," *Renewable Energy*, vol. 136, pp. 104-114, 2019.
- [11] G. J. Peng, G. X. Zhou, S. S. Fu, J. F. Ma, X. Huang and Q. J. Zhu, "Preliminary study on internal flow simulation of centrifugal dredge pump by SPH algorithm," *IOP Conference Series: Earth and Environmental Science*, vol. 163, p. 012118, 2018.
- [12] J. Teng, A. J. Jakeman, J. Vaze, B. F. W. Croke, D. Dutta and S. Kim, "Flood inundation modelling: A review of methods, recent advances and uncertainty analysis," *Environmental Modelling & Software*, vol. 90, pp. 201-216, 2017.
- [13] R. A. Luettich, J. J. Westerink and N. W. Scheffner, "ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts, and Estuaries. Report 1. Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL," *Dredging Research Program Tech. Rep. DRP-92-6*, p. 143, 1992.
- [14] Deltares, "Delft3D," Deltares, 2019. [Online]. Available: <https://oss.deltares.nl/web/delft3d>. [Accessed 17 6 2019].

- [15] J. Bender, D. Finkenzeller and P. Oel, "HW3D: A tool for interactive real-time 3D visualization in GIS supported flood modelling," in *17th international conference on computer animation and social agents*, Geneva, Switzerland, 2004.
- [16] Flow Science, "Why FLOW-3D?," 2018. [Online]. Available: <https://www.flow3d.com/products/flow-3d/>. [Accessed 12 5 2018].
- [17] C. W. Hirt and B. D. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *Journal of Computational Physics*, vol. 39, no. 1, pp. 201-225, 1981.
- [18] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi and T. H. Kolbe, "3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML," *Open Geospatial Data, Software and Standards*, vol. 3, no. 1, p. 5, 2018.
- [19] F. Prandi, M. Soave, F. Devigili and R. Amicis, "3D web visualization of huge CityGML models," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vols. XL-3/W3, pp. 601-605, 2015.
- [20] G. Gröger and L. Plümer, "CityGML - Interoperable semantic 3D city models," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 71, pp. 12 - 33, 2013.
- [21] T. Kolbe, "Representing and Exchanging 3D City Models with CityGML," in *3D Geo-Information Sciences*, Seoul, South Korea, Springer, Berlin, Heidelberg, 2009, pp. 15-31.
- [22] C. Schulte and V. Coors, "Development of a CityGML ADE for dynamic 3D flood information," in *Joint ISCRAM-CHINA and GI4DM Conference on Information Systems for Crisis Management*, Harbin, China, 2008.

- [23] P. Mooney and M. Minghini, "A review of OpenStreetMap data," in *Mapping and the citizen sensor*, London, UK, Ubiquity Press, 2017, pp. 37-59.
- [24] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics*, vol. 32, no. 4, 2013.
- [25] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics - Theory and application to non-spherical stars," *Monthly Notices of the Royal Astronomical Society*, vol. 181, pp. 375-389, 1977.
- [26] L. B. Lucy, "A numerical approach to the testing of the fission hypothesis," *Astronomical Journal*, vol. 82, pp. 1013-1024, 1977.
- [27] I. A. Hadimlioglu and S. A. King, "City Maker: Reconstruction of Cities from OpenStreetMap Data for Environmental Visualization and Simulations," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 7, 2019.
- [28] G. J. Arcement and V. R. Schneider, "Guide for selecting Manning's roughness coefficients for natural channels and floodplains," *United States Geological Survey Water-Supply Paper*, vol. 2339, pp. 1-38, 1989.
- [29] M. Müller, B. Heidelberger, M. Hennix and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109-118, 2007.
- [30] R. Winchenbach, H. Hochstetter and A. Kolb, "Constrained Neighbor Lists for SPH-based Fluid Simulations," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Goslar, Germany, 2016.

- [31] I. A. Hadimlioglu and S. A. King, "Visualization of Flooding Using Adaptive Spatial Resolution," *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 5, 2019.
- [32] G. F. Jenks, "Generalization in statistical mapping," *Annals of the Association of American Geographers*, vol. 53, pp. 15-26, 1963.
- [33] G. F. Jenks, "The Data Model Concept in Statistical Mapping," *International Yearbook of Cartography*, vol. 7, pp. 186-190, 1967.
- [34] M. Over, A. Schilling, S. Neubauer and A. Zipf, "Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany," *Computers, Environment and Urban Systems*, vol. 34, no. 6, pp. 496 - 507, 2010.
- [35] K. Schröter, S. Lüdtke, R. Redweik, J. Meier, M. Bochow, L. Ross, C. Nagel and H. Kreibich, "Flood loss estimation using 3D city models and remote sensing data," *Environmental Modelling & Software*, vol. 105, pp. 118-131, 2018.
- [36] N. Shiode, "3D urban models: recent developments in the digital modelling of urban environment in three-dimensions," *GeoJournal*, vol. 52, no. 3, pp. 263-269, 2001.
- [37] V. Varduhn, R. Mundani and E. Rank, "Multi-resolution models: Recent progress in coupling 3D geometry to environmental numerical simulation," in *3D Geoinformation Science*, Cham, Switzerland, Springer International Publishing, 2015, pp. 55-69.
- [38] S. Amirebrahimi, A. Rajabifard, P. Mendis and T. Ngo, "A framework for a microscale flood damage assessment and visualization for a building using BIM—GIS integration," *International Journal of Digital Earth*, vol. 9, no. 4, pp. 363-386, 2016.
- [39] U. Isikdag and S. Zlatanova, "Interactive modelling of buildings in Google Earth: A 3D tool for Urban Planning", bookTitle="Developments in 3D Geo-Information Sciences," in

Developments in 3D Geo-Information Sciences, Berlin, Heidelberg, Springer Berlin Heidelberg, 2010, pp. 52-70.

- [40] R. Neuville, J. Pouliot, F. Poux and R. Billen, "3D Viewpoint Management and Navigation in Urban Planning: Application to the Exploratory Phase," *Remote Sensing*, vol. 11, no. 3, 2019.
- [41] H. Xu, R. Badawi, X. Fan, J. Ren and Z. Zhang, "Research for 3D visualization of Digital City based on SketchUp and ArcGIS," in *Proc. SPIE 7492, International Symposium on Spatial Analysis, Spatial-Temporal Data Modeling, and Data Mining*, Wuhan, China, 2009.
- [42] A. A. A. Abdullah, N. Mohd Noor and A. Abdullah, "Constructing and Modeling 3D GIS Model in City Engine for Traditional Malay City," in *Proceedings of the Second International Conference on the Future of ASEAN (ICoFA) 2017 -- Volume 2*, Singapore, 2018.
- [43] T. Agius, S. Sabri and M. Kalantari, "Three-Dimensional Rule-Based City Modelling to Support Urban Redevelopment Process," *ISPRS International Journal of Geo-Information*, vol. 7, no. 10, 2018.
- [44] T. Glander and J. Döllner, "Abstract representations for interactive visualization of virtual 3D city models," *Computers, Environment and Urban Systems*, vol. 33, no. 5, pp. 375-387, 2009.
- [45] Y. Zhang, Z. Zhang, J. Zhang and J. Wu, "3D Building Modelling with Digital Map, Lidar Data and Video Image Sequences," *The Photogrammetric Record*, vol. 20, pp. 285 - 302, 2005.

- [46] F. Deng, Z. Zhang and J. and Zhang, "Construct 3D City Model by Multi-Sensor Data," in *ISPRS Workshop on Service and Application of Spatial Data Infrastructure*, Hangzhou, China, 2005.
- [47] J. Xiao, T. Fang, P. Zhao, M. Lhuillier and L. Quan, "Image-based Street-side City Modeling," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1-12, 2009.
- [48] K. Aringer and R. Roschlaub, "Bavarian 3D Building Model and Update Concept Based on LiDAR, Image Matching and Cadastre Information," in *Innovations in 3D Geo-Information Sciences*, Cham, Switzerland, Springer International Publishing, 2014, pp. 143--157.
- [49] G. Forlani, C. Nardinocchi, M. Scaioni and P. Zingaretti, "Complete classification of raw LIDAR data and 3D reconstruction of buildings," *Pattern Analysis and Applications*, vol. 8, no. 4, pp. 357-374, 2006.
- [50] L. Caltagirone, S. Scheidegger, L. Svensson and M. Wahde, "Fast LIDAR-based road detection using fully convolutional neural networks," in *Fast LIDAR-based road detection using fully convolutional neural networks*, Los Angeles, CA, USA, 2017.
- [51] Y. Wang, L. Cheng, Y. Chen and Y. L. M. Wu, "Building Point Detection from Vehicle-Borne LiDAR Data Based on Voxel Group and Horizontal Hollow Analysis," *Remote Sensing*, vol. 8, no. 5, p. 419, 2016.
- [52] Z. Wang and A. Zipf, "Using OpenStreetMap data to generate building models with their inner structures for 3d maps," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vols. IV-2/W4, pp. 411--416, 2017.

- [53] F. Ramm, "OpenStreetMap Data in Layered GIS Format. Version 0.6.7.," 2015. [Online]. Available: <https://www.geofabrik.de/data/geofabrik-osm-gis-standard-0.6.pdf>. [Accessed 12 5 2019].
- [54] R. Braun, V. Weiler, M. Zirak, L. Dobisch, V. Coors and U. Eicker, "Using 3D CityGML Models for Building Simulation Applications at District Level," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, Stuttgart, Germany, 2018.
- [55] OGC, "CityGML," 2019. [Online]. Available: <https://www.opengeospatial.org/standards/citygml>. [Accessed 15 5 2019].
- [56] T. H. Kolbe, G. Gröger and L. Plümer, "CityGML – 3D city models and their potential for emergency response," in *Geospatial Information Technology for Emergency Response*, London, UK, Taylor & Francis, 2008, pp. 257-274.
- [57] M. Kalantari, "Future City Pilot-1: Using IFC/CityGML in Urban Planning Engineering Report," Public Engineering Report, Open Geospatial Consortium, 2017.
- [58] A. Labetski, S. van Gerwen, G. Tamminga, H. Ledoux and J. and Stoter, "A proposal for an improved transportation model in citygml," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, Vols. XLII-4/W10, pp. 89-96, 2018.
- [59] F. Biljecki, K. Kumar and C. Nagel, "CityGML Application Domain Extension (ADE): overview of developments," *Open Geospatial Data, Software and Standards*, vol. 3, no. 13, 2018.

- [60] H. Ledoux, K. A. Ohori, K. Kumar, B. Dukai, A. Labetski and S. Vitalis, "CityJSON: A compact and easy-to-use encoding of the CityGML data model," *arXiv*, vol. 1902.09155, 2019.
- [61] I. Toschi, E. Nocerino, F. Remondino, A. Revolti, G. Soria and S. Piffer, "Geospatial data processing for 3d city model generation, management and visualization," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vols. XLII-1/W1, pp. 527-534, 2017.
- [62] F. Biljecki, H. Ledoux and J. Stoter, "Generating 3D city models without elevation data," *Computers, Environment and Urban Systems*, vol. 64, pp. 1-18, 2017.
- [63] J. Bright, S. De Sabbata, S. Lee, B. Ganesh and D. K. Humphreys, "OpenStreetMap data for alcohol research: Reliability assessment and quality indicators," *Health & Place*, vol. 50, pp. 130-136, 2018.
- [64] S. Huber and C. Rust, "Calculate Travel Time and Distance with Openstreetmap Data Using the Open Source Routing Machine (OSRM)," *The Stata Journal*, vol. 16, no. 2, pp. 416-423, 2016.
- [65] L. Li, R. Pi and H. Zhou, "The Design and Implementation of Openstreetmap Application System Aimed at Emergency Logistics," in *Information Technology and Intelligent Transportation Systems*, Cham, Switzerland, 2017.
- [66] J. P. Albuquerque and B. a. E. M. Herfort, "The Tasks of the Crowd: A Typology of Tasks in Geographic Information Crowdsourcing and a Case Study in Humanitarian Mapping," *Remote Sensing*, vol. 8, no. 10, 2016.

- [67] A. Basiri, M. Jackson, P. Amirian, A. Pourabdollah, M. Sester, A. Winstanley, T. Moore and L. Zhang, "Quality assessment of OpenStreetMap data using trajectory mining," *Geospatial Information Science*, vol. 19, no. 1, pp. 56-68, 2016.
- [68] G. Medeiros, M. Holanda, A. Araújo and M. Victorino, "OpenStreetMap: Quality assessment of Brazil's collaborative geographic data over ten years," in *Geoinfo*, Salvador, BA, Brazil, 2017.
- [69] M. Eckle and J. P. Albuquerque, "Quality Assessment of Remote Mapping in OpenStreetMap for Disaster Management Purposes," in *ISCRAM 2015 Conference*, Kristiansand, Norway, 2015.
- [70] A. Graser, M. Straub and M. Dragaschnig, "Is OSM Good Enough for Vehicle Routing? A Study Comparing Street Networks in Vienna," in *LBS2014 11th International Symposium on Location-Based Services*, Vienna, Austria, 2015.
- [71] J. Arsanjani, P. Mooney, A. Zipf and A. Schauss, "Quality Assessment of the Contributed Land Use Information from OpenStreetMap Versus Authoritative Datasets," in *OpenStreetMap in GIScience: Experiences, Research, and Applications*, Cham, Switzerland, Springer International Publishing, 2015, pp. 37-58.
- [72] G. Gesquière and A. Manin, "3D visualization of urban data based on CityGML with WebGL," *International Journal of 3-D Information Modeling*, vol. 1, pp. 1-15, 2014.
- [73] J. Gaillard, A. Vienne, R. Baume, F. Pedrinis, A. and Peytavie and G. Gesquière, "Urban Data Visualisation in a Web Browser," in *Proceedings of the 20th International Conference on 3D Web Technology*, Heraklion, Crete, Greece, 2015.

- [74] M. Beck, "Real-Time Visualization of Big 3D City Models," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vols. XXXIV-5/W10, 2003.
- [75] D. Ma, M. Sandberg and B. Jiang, "Characterizing the Heterogeneity of the OpenStreetMap Data and Community," *ISPRS Int. J. Geo-Inf.*, vol. 4, no. 2, 2015.
- [76] M. J. Machado, B. A. Botero, J. Lopez, F. Frances, A. Diez-Herrero and G. Benito, "Flood frequency analysis of historical flood data under stationary and non-stationary modelling," *Hydrology and Earth System Sciences*, vol. 19, no. 6, pp. 2561-2576, 2015.
- [77] D. Didier, P. Bernatchez, G. Boucher-Brossard, A. Lambert, C. Fraser, R. L. Barnett and S. Van-Wierts, "Coastal Flood Assessment Based on Field Debris Measurements and Wave Runup Empirical Model," *Journal of Marine Science and Engineering*, vol. 3, no. 3, pp. 560-590, 2015.
- [78] J. Leandro, A. S. Chen, S. Djordjevic and D. A. Savic, "Comparison of 1D/1D and 1D/2D Coupled (Sewer/Surface) Hydraulic Models for Urban Flood Simulation," *Journal of Hydraulic Engineering*, vol. 135, no. 6, pp. 495-504, 2009.
- [79] Z. Liao, Z. Xu, D. Wang, S. Lu and P. M. Hannam, "River environmental decision support system development for Suzhou Creek in Shanghai," *Journal of Environmental Management*, vol. 92, no. 9, pp. 2211 - 2221, 2011.
- [80] Y. Wang, R. Liu, L. Guo, J. Tian, X. Zhang, L. Ding, C. Wang and Y. Shang, "Forecasting and Providing Warnings of Flash Floods for Ungauged Mountainous Areas Based on a Distributed Hydrological Model," *Water*, vol. 9, no. 10, 2017.

- [81] S. Gabellani, F. Silvestro, R. Rudari and G. Boni, "General calibration methodology for a combined Horton-SCS infiltration scheme in flash flood modeling," *Natural Hazards and Earth System Sciences*, vol. 8, no. 6, pp. 1317--1327, 2008.
- [82] Y. Chen, J. Li, S. Huang and Y. Dong, "Study of Beijiang catchment flash-flood forecasting model," *Proceedings of the International Association of Hydrological Sciences*, vol. 368, pp. 150-155, 2015.
- [83] H. Roux, D. Labat, P. Garambois, M. Maubourguet, J. Chorda and D. Dartus, "A physically-based parsimonious hydrological model for flash floods in Mediterranean catchments," *Natural Hazards and Earth System Sciences*, vol. 11, pp. 2567-2582, 2011.
- [84] S. J. Wester, R. Grimson, P. G. Minotti, M. J. Booij and M. Brugnach, "Hydrodynamic modelling of a tidal delta wetland using an enhanced quasi-2D model," *Journal of Hydrology*, vol. 559, pp. 315-326, 2018.
- [85] O. Mark, S. Weesakul, C. Apirumanekul, S. B. Aroonnet and S. Djordjevic, "Potential and limitations of 1D modelling of urban flooding," *Journal of Hydrology*, vol. 299, no. 3, pp. 284-299, 2004.
- [86] V. T. Chow, D. R. Maidment and L. W. Mays, *Applied Hydrology*, New York: McGraw-Hill, 1988.
- [87] P. Dimitriadis, A. Tegos, A. Oikonomou, V. Pagana, A. Koukouvinos, N. Mamassis, D. Koutsoyiannis and A. Efstratiadis, "Comparative evaluation of 1D and quasi-2D hydraulic models based on benchmark and real-world applications for uncertainty assessment in flood mapping," *Journal of Hydrology*, vol. 534, pp. 478-492, 2016.

- [88] S. J. Noh, J. Lee, S. Lee, K. Kawaike and D. Seo, "Hyper-resolution 1D-2D urban flood modelling using LiDAR data and hybrid parallelization," *Environmental Modelling & Software*, vol. 103, pp. 131 - 145, 2018.
- [89] J. Jang, T. Chang and W. Chen, "Effect of inlet modelling on surface drainage in coupled urban flood simulation," *Journal of Hydrology*, vol. 562, pp. 168 - 180, 2018.
- [90] Q. Liu, Y. Qin, Y. Zhang and Z. Li, "A coupled 1D--2D hydrodynamic model for flood simulation in flood detention basin," *Natural Hazards*, vol. 75, no. 2, pp. 1303-1325, 2015.
- [91] P. Nguyen, A. Thorstensen, S. Sorooshian, K. Hsu, A. AghaKouchak, B. Sanders, V. Koren, Z. Cui and M. Smith, "A high resolution coupled hydrologic-hydraulic model (HiResFlood-UCI) for flash flood modeling," *Journal of Hydrology*, vol. 541, pp. 401-420, 2016.
- [92] U. Pasquier, Y. He, S. Hooton, M. Goulden and K. M. Hiscock, "An integrated 1D--2D hydraulic modelling approach to assess the sensitivity of a coastal region to compound flooding hazard under climate change," *Natural Hazards*, pp. 1-23, 2018.
- [93] T. Webster, K. McGuigan, K. Collins and C. MacDonald, "Integrated River and Coastal Hydrodynamic Flood Risk Mapping of the LaHave River Estuary and Town of Bridgewater, Nova Scotia, Canada," *Water*, vol. 6, no. 3, pp. 517-546, 2014.
- [94] J. G. Leskens, C. Kehl, T. Tutenel, T. Kol, G. d. Haan, G. Stelling and E. Eisemann, "An interactive simulation and visualization tool for flood analysis usable for practitioners," *Mitigation and Adaptation Strategies for Global Change*, vol. 22, no. 2, pp. 307-324, 2017.

- [95] X. Liu, D. Zhong, D. Tong, Z. Zhou, X. Ao and W. Li, "Dynamic visualization of storm surge flood routing based on three-dimensional numerical simulation," *Journal of Flood Risk Management*, vol. 11, 2016.
- [96] J. Liu, J. Gong, J. Liang, Y. Li, L. Kang, L. Song and S. Shi, "A quantitative method for storm surge vulnerability assessment - a case study of Weihai city," *International Journal of Digital Earth*, vol. 10, no. 5, pp. 539-559, 2017.
- [97] H. Toros, A. Kahraman, S. Tilev-Tanrioer, G. Geertsema and G. Cats, "Simulating heavy precipitation with HARMONIE, HIRLAM and WRF-ARW: a flash flood case study in Istanbul, Turkey," *European Journal of Science and Technology*, vol. 13, pp. 1-12, 2018.
- [98] Y. Bao, Y. Huang, G. R. Liu and W. Zeng, "SPH simulation of high-volume rapid landslides triggered by earthquakes based on a unified constitutive model. part ii: solid-liquid-like phase transition and flow-like landslides," *International Journal of Computational Methods*, 2018.
- [99] S. Zhang, Z. Xia and T. Wang, "A real-time interactive simulation framework for watershed decision making using numerical models and virtual environment," *Journal of Hydrology*, vol. 493, pp. 95-104, 2013.
- [100] F. Macchione, P. Costabile, C. Costanzo and R. De Santis, "Moving to 3-D flood hazard maps for enhancing risk communication," *Environmental Modelling & Software*, vol. 111, pp. 510-522, 2019.
- [101] D. Christodoulou and S. Miao, *Compressible flow and Euler's equations*, vol. 9, Somerville: International Press,, 2014.

- [102] O. A. Ladyzhenskaya, "Mathematical analysis of Navier-Stokes equations for incompressible liquids," *Annual Review of Fluid Mechanics*, vol. 7, no. 1, pp. 249-272, 1975.
- [103] S. Succi and J. M. Yeomans, "The Lattice Boltzmann equation for fluid dynamics and beyond," *Physics Today*, vol. 55, no. 12, pp. 58-60, 2002.
- [104] A. J. Q. Alkemade, F. T. M. Nieuwstadt and E. Groesen, "The vorton method. Theory and applications to fluid mechanics," *Applied Scientific Research*, vol. 51, pp. 153-158, 1993.
- [105] Q. Liang, X. Xia and J. Hou, "Efficient urban flood simulation using a GPU-accelerated SPH model," *Environmental Earth Sciences*, vol. 74, no. 11, pp. 7285-7294, 2015.
- [106] L. Herman, J. Russnák and T. Řezník, "Flood Modelling and Visualizations of Floods Through 3D Open Data," in *Environmental Software Systems. Computer Science for Environmental Protection*, Cham, Switzerland, Springer International Publishing, 2017, pp. 139-149.
- [107] J. N. Ghazali and A. Kamsin, "A real time simulation of flood hazard," in *2008 Fifth International Conference on Computer Graphics, Imaging and Visualisation*, Penang, Malaysia, 2008.
- [108] P. Adda, D. Mioc, F. Anton, E. McGillivray, A. Morton and D. Fraser, "3D flood-risk models of government infrastructure," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 38, 2010.
- [109] S. Alahmed and Q. Zou, "Modelling urban coastal flooding through 2-D array of buildings using smoothed particle hydrodynamics," in *35th International Conference on Coastal Engineering*, Antalya, Turkey, 2016.

- [110] G. W. Brunner, "HEC-RAS River Analysis System. Hydraulic Reference Manual. Version 5.0," US Army Corps of Engineers, Davis, CA, 2016.
- [111] J. Yang, R. D. Townsend and B. Daneshfar, "Applying the HEC-RAS model and GIS techniques in river network floodplain delineation," *Canadian Journal of Civil Engineering*, vol. 33, pp. 19-28, 2006.
- [112] M. S. Horritt and P. D. Bates, "Evaluation of 1D and 2D numerical models for predicting river flood inundation," *Journal of Hydrology*, vol. 268, no. 1, pp. 87-99, 2002.
- [113] T. Thol, L. Kim, S. Ly, S. Heng and S. Sreykeo, "Application of HEC-RAS for a flood study of a river reach in Cambodia," in *The 4th International Young Researchers' Workshop on River Basin Environment and Management*, Ho Chi Minh City, Vietnam, 2016.
- [114] B. P. Yakti, M. B. Adityawan, M. Farid, Y. Suryadi, J. Nugroho and I. K. Hadihardaja, "2D Modeling of Flood Propagation due to the Failure of Way Ela Natural Dam," *MATEC Web of Conferences*, vol. 147, p. 03009, 2018.
- [115] F. E. Hicks and P. T, "Suitability of HEC-RAS for Flood Forecasting," *Canadian Water Resources Journal*, vol. 30, no. 2, pp. 159-174, 2005.
- [116] MIKE Powered by DHI, "MIKE FLOOD," 2019. [Online]. Available: <https://www.mikepoweredbydhi.com/products/mike-flood>. [Accessed 30 4 2019].
- [117] S. Patro, C. Chatterjee, S. Mohanty, R. Singh and N. S. Raghuwanshi, "Flood Inundation Modeling using MIKE FLOOD and Remote Sensing Data," *Journal of the Indian Society of Remote Sensing*, vol. 37, no. 1, pp. 107-118, 2009.

- [118] P. Delaney, Y. Qiao, T. Mereu and N. Lorrain, "Using detailed 2D urban floodplain modeling to inform development planning in Mississauga, ON," in *22nd Canadian Hydrotechnical Conference*, Montreal, Quebec, Canada, 2015.
- [119] P. D. Bates and A. P. J. De Roo, "A simple raster-based model for flood inundation simulation," *Journal of Hydrology*, vol. 236, pp. 54-77, 2000.
- [120] University of Bristol, School of Geographical Sciences, "LISFLOOD-FP," 2019.
[Online]. Available:
<http://www.bristol.ac.uk/geography/research/hydrology/models/lisflood/>. [Accessed 10 5 2019].
- [121] G. Amarnath, Y. M. Umer, N. Alahacoon and Y. Inada, "Modelling the flood-risk extent using LISFLOOD-FP in a complex watershed: case study of Mundeni Aru River Basin, Sri Lanka," in *Proceedings of the International Association of Hydrological Science (IAHS)*, Prague, Czech Republic, 2015.
- [122] K. Andreadis, G. Schumann, D. Stampoulis, A. Smith, J. Neal, P. Bates, C. Sampson, R. Brakenridge and A. Kettner, "Building a flood climatology and rethinking flood risk at continental scales," in *EGU General Assembly 2016*, Vienna, Austria, 2016.
- [123] J. Neal and T. a. C. S. a. A. S. a. P. B. Dunne, "Optimisation of the two-dimensional hydraulic model LISFOOD-FP for CPU architecture," *Environmental Modelling & Software*, vol. 107, pp. 148-157, 2018.
- [124] P. A. M. Berry, J. D. Garlick and R. G. Smith, "Near-global validation of the SRTM DEM using satellite radar altimetry," *Remote Sensing of Environment*, vol. 106, pp. 17-27, 2007.

- [125] J. Inman, *Navigation and Nautical Astronomy: For the Use of British Seamen*, London, UK: W. Woodward, C. & J. Rivington, 1835.
- [126] S. Duvvuri, "GIS Based Management System for Flood Forecast Applications," in *Proceedings of International Conference on Remote Sensing for Disaster Management*, 2019.
- [127] Y. Xing, Q. Liang, G. Wang, X. Ming and X. Xia, "City-scale hydrodynamic modelling of urban flash floods: the issues of scale and resolution," *Natural Hazards*, vol. 96, no. 1, pp. 473--496, 2019.
- [128] Y. Liu and J. Heer, "Somewhere over the rainbow: an empirical assessment of quantitative colormaps," in *ACM Human Factors in Computing Systems (CHI)*, Montréal, Canada, 2018.
- [129] D. Borland and R. M. Taylor, "Rainbow Color Map (Still) Considered Harmful," *IEEE Computer Graphics and Applications*, vol. 27, no. 2, pp. 14 - 17, 2007.
- [130] K. Moreland, "Diverging Color Maps for Scientific Visualization," in *ISVC 2009: Advances in Visual Computing*, Las Vegas, NV, 2009.
- [131] M. Harrower and C. A. Brewer, "ColorBrewer.org: an online tool for selecting colour schemes for maps," *The Cartographic Journal*, vol. 40, no. 1, pp. 27-37, 2003.
- [132] NVIDIA, "NVIDIA FleX," 13 8 2015. [Online]. Available: <https://developer.nvidia.com/flex>. [Accessed 12 5 2019].
- [133] NVIDIA, "CUDA Toolkit Documentation v10.1.168," 7 5 2019. [Online]. Available: <https://docs.nvidia.com/cuda/index.html>. [Accessed 20 5 2019].

- [134] D. C. Mason, D. M. Cobby, M. S. Horritt and P. D. Bates, "Floodplain friction parameterization in two-dimensional river flood models using vegetation heights derived from airborne scanning laser altimetry," *Hydrological Processes*, vol. 17, no. 9, pp. 1711-1732, 2003.
- [135] V. Bellos, I. Nalbantis and G. Tsakiris, "Friction Modeling of Flood Flow Simulations," *Journal of Hydraulic Engineering*, vol. 144, no. 12, p. 04018073, 2018.
- [136] Z. Liu, V. Merwade and K. Jafarzadegan, "Investigating the role of model structure and surface roughness in generating flood inundation extents using one- and two-dimensional hydraulic models," *Journal of Flood Risk Management*, vol. 12, no. 1, p. e12347, 2019.
- [137] Y. Fan, T. Ao, H. Yu, G. Huang and X. Li, "A Coupled 1D-2D Hydrodynamic Model for Urban Flood Inundation," *Advances in Meteorology*, vol. 2017, 2017.
- [138] C. A. Brewer, G. W. Hatchard and M. A. Harrower, "ColorBrewer in Print: A Catalog of Color Schemes for Maps," *Cartography and Geographic Information Science*, vol. 30, no. 1, pp. 5-32, 2003.
- [139] N. Marrero, "Visualization Metrics: An Overview," 2007. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.7892&rep=rep1&type=pdf>. [Accessed 25 6 2019].
- [140] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, T. Schreck, D. Weiskopf and D. A. Keim, "Quality Metrics for Information Visualization," *Computer Graphics Forum*, vol. 37, no. 3, pp. 625-662, 2018.

- [141] C. Wang, T. R. Wan and I. J. Palmer, "Urban flood risk analysis for determining optimal flood protection levels based on digital terrain model and flood spreading model," *The Visual Computer*, vol. 26, pp. 1369-1381, 2010.
- [142] D. Cornel, A. Buttinger-Kreuzhuber, A. Konev, Z. Horváth, M. Wimmer, R. Heidrich and J. Waser, "Interactive Visualization of Flood and Heavy Rain Simulations," in *Proceedings EuroVis 2019*, Porto, Portugal, 2019.



Appendix A – Sample OSM data, which is sent as an input to the City Maker.

Nodes

```
<node id="4528274624" lat="37.9774838" lon="23.7111736" version="1"
timestamp="2016-11-29T12:39:08Z" changeset="44033454" uid="411246"
user="makmar"/>
  <node id="4528274625" lat="37.9774111" lon="23.7111533" version="1"
timestamp="2016-11-29T12:39:08Z" changeset="44033454" uid="411246"
user="makmar"/>
```

Building

```
<way id="486554389" version="1" timestamp="2017-04-13T10:37:33Z"
changeset="47738167" uid="411246" user="makmar">
  <nd ref="4791486109"/>
  <nd ref="4791486107"/>
  <nd ref="4791486106"/>
  <nd ref="4791486105"/>
  <nd ref="4791486104"/>
  <nd ref="4791486108"/>
  <nd ref="4791486109"/>
  <tag k="building" v="yes"/>
</way>
```

Hospital

```
<way id="32297119" version="11" timestamp="2018-03-16T12:55:20Z"
changeset="57240482" uid="7789636" user="DepieCh">
  <nd ref="363404628"/>
  <nd ref="363404629"/>
  <nd ref="363404630"/>
  <nd ref="363404631"/>
  <nd ref="1458966507"/>
  <nd ref="363404634"/>
  <nd ref="363404635"/>
  <nd ref="1459106690"/>
  <nd ref="1459106689"/>
  <nd ref="363404642"/>
  <nd ref="363404643"/>
  <nd ref="363404644"/>
  <nd ref="363404645"/>
  <nd ref="363404646"/>
  <nd ref="363404628"/>
  <tag k="addr:city" v="Île de France"/>
  <tag k="addr:country" v="GR"/>
  <tag k="addr:housenumber" v="10-12"/>
  <tag k="addr:postcode" v="11521"/>
```


Appendix B – Sample City Maker file.

```
#
MAP_PROPERTIES 0.0537 27.7323 -97.3053
FEATURE_BUILDINGS
OBJECT
H 14.63
V: 62.8462 84.0796
V: 63.0961 83.4317
V: 63.5881 83.6691
V: 63.6707 83.4689
V: 64.0776 83.668
V: 63.9725 83.9237
V: 64.1101 83.9934
V: 63.8448 84.6591
V: 63.7222 84.5976
V: 63.5681 84.9738
V: 63.2563 84.8157
V: 63.355 84.5765
V: 63.2617 84.5301
V: 63.3153 84.3992
V: 63.1777 84.3298
V: 63.2066 84.2598
V: 62.8462 84.0796
FEATURE_PLAYGROUNDS
OBJECT
V: 85.475 178.365
V: 86.2874 178.202
V: 86.2248 177.763
V: 86.0042 177.518
V: 85.9713 177.525
V: 85.3499 177.652
V: 85.475 178.365
FEATURE_SEWERS
OBJECT
V: 155.399 101.238
OBJECT
V: 136.64 105.415
OBJECT
V: 138.896 182.033
OBJECT
V: 171.619 167.586
OBJECT
V: 151.606 202.505
OBJECT
V: 141.731 183.82
OBJECT
V: 162.722 185.565
#
```

Appendix C – Sample storm drain information passed to the City Maker.

X	Y	OBJECTID	FLOWLINEEL	ASSET_CODE
-97.43051707	27.78956941	2001	32.54	STMMNHL1161266
-97.43179442	27.79048687	2002	0	STMMNHL1161722
-97.34827803	27.67010699	2003	0	STMMNHL1161724
-97.39224592	27.6565539	2004	10.8	STMMNHL1161270
-97.38922432	27.65711872	2005	12.99	STMMNHL1161726
-97.38990245	27.65931939	2006	0	STMMNHL1161271
-97.2982124	27.65137574	2007	9.14	STMMNHL1161728
-97.30164012	27.64726367	2008	0	STMMNHL1161273
-97.29307309	27.65124734	2009	6.4	STMMNHL1161274
-97.22300323	27.61590107	2010	3.25	STMMNHL1161731
-97.53933682	27.76801096	2011	0	STMMNHL1161277
-97.54232377	27.76710708	2012	0	STMMNHL1161279
-97.53933691	27.7665766	2013	0	STMMNHL1161732
-97.65187814	27.86649999	2014	17.26	STMMNHL1161733
-97.65297759	27.8654673	2015	26.47	STMMNHL1161558
-97.39211931	27.7562976	2016	15.6	STMMNHL15885292
-97.39159804	27.75551865	2017	16.43	STMMNHL15885316
-97.38940405	27.753949	2018	23.5	STMMNHL15885355
-97.39449627	27.76304147	2019	24	STMMNHL15885844
-97.39739831	27.67550766	2020	0	STMMNHL15890986
-97.63139529	27.8604881	2021	75.75	STMMNHL15917600
-97.4252896	27.79882889	2022	18.85	STMMNHL15917607
-97.37830381	27.66437747	2023	10.2	STMMNHL15928271
-97.37724218	27.66606941	2024	11.59	STMMNHL15928300
-97.37587149	27.6682688	2025	12.94	STMMNHL15928310

BIOGRAPHICAL SKETCH

Ismail Alihan Hadimlioglu earned B.S. Computer Engineering degree at Istanbul Commerce University, Turkey in 2010. During his degree and two internships, his major focus was on graphics programming, musical compositions and sound engineering. Upon graduation and a short period of work, he arrived at Texas A&M University – Corpus Christi to earn a Master’s degree in Computer Science. With a focus on visualization and simulations, Hadimlioglu completed his degree in 2013. Hadimlioglu has been serving the Texas A&M University – Corpus Christi as an instructor since 2013.

Education:

- B.S. Computer Engineering – Istanbul Commerce University, Turkey – 2010
Thesis: Art in Video Games
- M.S. Computer Science – Texas A&M University – Corpus Christi – 2013
Thesis: Automated musical transitions through rule-based synthesis using musical properties

Professional Employment:

- Instructor – Texas A&M University – Corpus Christi – 2013 - 2019

Journal Articles:

- Hadimlioglu, I. A. and King, S. A. (2018). "Automated musical transitions through rule-based synthesis using musical properties." Entertainment Computing **28**: 59 - 67.
- Hadimlioglu, I. A. and King, S. A. (2019). "Visualization of Flooding Using Adaptive Spatial Resolution." ISPRS Int. J. Geo-Inf. **8(5)**: 204.
- Hadimlioglu, I. A. and King, S. A. (2019). "City Maker: Reconstruction of Cities from OpenStreetMap Data for Environmental Visualization and Simulations" ISPRS Int. J. Geo-Inf. **8(7)**: 298.