Pangeo SRS Documentation

Egermier, Isaac; Gaddi, Chezka; Theesen, Michael; Tong, Garfield

April 1, 2020

CONTENTS:

Chapter ONE

INTRODUCTION

1.1 Purpose

The purpose of this document is to present a detailed description of Project Pangeo. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, and the constraints under which it must operate. This document is intended for both the stakeholders and the developers of the system.

1.2 Scope

The SDSMT VRtualize Team is working together with L3Harris on Project Pangeo. Pangeo is a platform aiming to render the real world with 3D technology in a dynamically generated virtual reality environment. This platform will cover two major needs of L3-Harris. The first is a method to quickly and easily survey an area in the real world. The second is a cost-effective method to train their computer vision algorithms.
For the first method, VRtualize approximates these savings:

- 60% time saved per 5 square miles
- 196 times more area covered per square degree than distance surveyed in a given workday
- $1540 saved from room and board to support surveying one square degree

According to our source, within a given workday, data is gathered on latitude and longitude, elevation, geographical features, soil composition, and in some cases, civilian features, for 5 square miles. Pangeo provides latitude and longitude, elevation, and geographical features.
VRtualize also estimated saved costs for training computer vision algorithms below:

- Depending on the aircraft, L3-Harris will save $1,500-$4,000 per flight hour
- The salary of a cargo pilot, which is around $60,000 a year

1.3 Audience

Pangeo is designed to be an internal research tool for the company L3Harris. Despite that, the aim of the project is to be usable by anyone with access to virtual reality equipment and an interest in touring geographical points of interest from the comfort of their home.

1.4 Glossary

Term	Definition
Application	The interface between the Explorer and the Local Database. It handles interpretting Terrain Data for Generated Terrain.
Bing Maps Database	The Cooperating System that the Local System retrieves Terrain Data from. Owned by Microsoft and distributed for free use.
Cooperating System	Any system external to the Local System necessary to the overall System Environment, primarily elevation information databases.
Explorer	The user viewing Generated Terrain. Can be a researcher, home user, or any other Stakeholder.

Familiar VR Movements	Common methods adopted by software for users to move about and look around in a virtual reality environment. Incorporates use of trackpads for vertical and horizontal movement through the generated world.
Generated Terrain	Interactive 3-Dimensional Virtual Reality imagery rendered from interpreted Terrain Data.
Local Database	A database instance within the Local System. Handles caching logic and serving Terrain Data to the Application.
Local System	The system environment installed on the Explorer's machine.
Software Requirements Specification	A document that describes all functions of a proposed system and the constraints under which it must operate. For example, this document.
Stakeholder	Any person with an interest in the project who is not a developer.
System Environment	The overall environment consisting of the Local System, Cooperating Systems, and users.
Terrain Data	Data necessary to create Generated Terrain.
VR Equipment	Virtual Reality headset and associated accessories necessary to interact with Interactive 3-Dimensional Virtual Reality imagery.

1.5 Overview

This document is comprised of two parts, the high-level description and the requirements specification.

The high-level description includes the overall system environment, and assumptions, risks, and dependencies VRtualize has considered for this project.

The requirements specification outlines the functional and non-functional requirements. Functional requirements can be broken down into use case requirements, local system requirements, and external interface requirements. Use case requirements are requirements of supported usages between the application and the explorer. local system requirements are requirements of supported usages of the local system. local system requirements are further broken down into the application functions and the local database functions. Last, external interface requirements are the requirements of the cooperating systems necessary for the local database to fetch terrain data.

Chapter TWO

HIGH LEVEL DESCRIPTION

2.1 System Environment

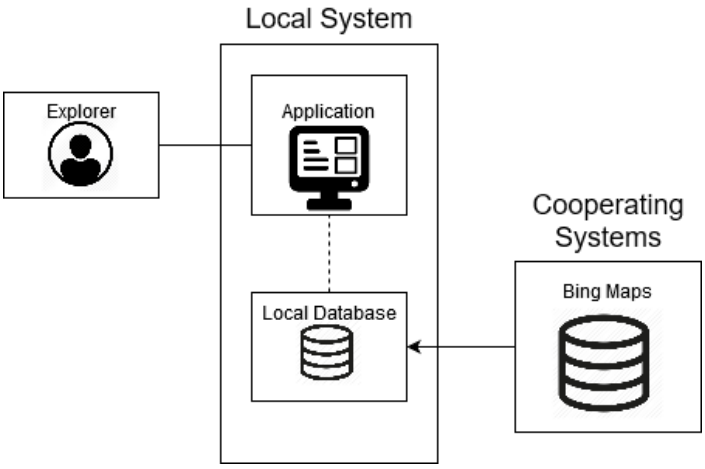

Fig. 1: Diagram depicting application component interactions

The overall system has one active user (the Explorer), one Local System, and Cooperating Systems. The Explorer interacts with the Local System via VR Equipment. The Local System is comprised of two major parts, the Application, and the Local Database. The Local System retrieves Terrain Data from a Cooperating System and stores it within the Local Database for caching logic. The Application requests Terrain Data from the Local Database through an abstraction layer to loosely couple the Terrain Data to the Application. The Application will use the Terrain Data to create Generated Terrain for the Explorer.

2.2 Assumptions, Risks, and Dependencies

A number of factors may affect the requirements specified by the Software Requirements Specifications

- The physical machine may not meet requirements specified by System Characteristics in chapter 3.2.1
- The local system may be unable to communicate with cooperating systems.
- Cooperating systems update APIs with breaking changes for our abstraction layer.
- The explorer may not be familiar with VR equipment.
- The explorer is prone to motion sickness with VR equipment.

Chapter THREE

REQUIREMENTS SPECIFICATION

3.1 Functional Requirements with Use Cases

This section outlines the use cases of the Explorer.

3.1.1 Use case: View Generated Terrain

Brief Description

The Explorer can see and explore Generated Terrain.

Initial Step-By-Step Description

1. The Explorer Triggers *View Generated Terrain* via:
    - Accessing the Application via VR Equipment.
    - Warping to a location via *Warping*.
    - Moving to the edge of the Generated Terrain via *Interact with Generated Terrain*.
2. The Application will *Create Generated Terrain*.
3. The Application provides Generated Terrain to the Explorer.

Strictly speaking, this function describes what the Explorer should see through his/her VR Equipment when accessing the Local System. Interaction between the Explorer and the Generated Terrain is detailed in *Interaction with Generated Terrain*.

3.1.2 Use case: Interacting with Generated Terrain

Brief Description

The Explorer can move and look around the Generated Terrain with Familiar VR Movements.

Initial Step-By-Step Description

Before this use case can be initiated, the Explorer has already accessed the Application.

1. The Application creates Generated Terrain.
2. The Explorer may use Familiar VR Movements to move about and look around the Generated Terrain.
3. If the Explorer moves to the edge of the Generated Terrain, *View Generated Terrain* is triggered.

3.1.3 Use case: Warping

Brief Description

The Application provides controls to warp to a specified longitude and latitude provided by the Explorer.

Initial Step-By-Step Description

Before this use case can be initiated, the Explorer has already accessed the Application.

1. The Application provides a menu the Explorer may access anytime inside the Application.

2. The Explorer presses the main menu button and navigates to the teleport page.
3. The Application displays dialog to receive longitude and latitude from the Explorer.
4. The Explorer inputs longitude and latitude and submits to the Application.
5. *View Generated Terrain* is triggered centered around received longitude and latitude.

3.2 Local System Requirements

This section outlines the functions of the Application and the Local Database

3.2.1 Application: Create Generated Terrain

Brief Description

The Application Renders Interactive 3-Dimensional Virtual Reality Imagery.

Initial Step-By-Step Description

1. The *Request Terrain Data* is triggered.
2. Terrain Data is interpreted into Generated Terrain.
3. The Application returns Generated Terrain to *View Generated Terrain.*

3.2.2 Application: Request Terrain Data

Brief Description

The Application contacts the Local Database for Terrain Data

Initial Step-By-Step Description

1. The Application requests Terrain Data through an abstraction layer.
2. The Local Database performs *Prepare Terrain Data*.
3. The Application receives a response from the Local Database through the abstraction layer.
4. Request Terrain Data returns this response to *Create Generated Terrain*.

3.2.3 Local Database: Prepare Terrain Data

Brief Description

The Local Database prepares Terrain Data necessary to *Create Generated Terrain*

Initial Step-By-Step Description

1. The Local Database checks for Terrain Data within itself.
2. If the Local Database contains the necessary Terrain Data, proceed to step 6.
3. If the Local Database does not contain the necessary Terrain Data, *Fetch the Data*.
4. If the Local Database is not “full”, store the necessary Terrain Data and proceed to step 6.
5. If the Local Database is “full”, use caching logic (CPU inspired, FIFO, LRU), decide which cached Terrain Data to replace with necessary Terrain Data and proceed to step 6.
6. Return necessary Terrain Data to *Request Terrain Data*.

Depending on caching logic, there may be different definitions of “full”. If VRtualize mimics CPU cache implementations, “full” would be when a “hit” is encountered for the index calculated with relevant information of the Terrain Data, and the Local Database will replace the existing Terrain Data with the necessary Terrain Data. Not “full” would be when a “miss” is encountered for the index calculated with relevant information of the Terrain Data, and the Local Database will store the necessary Terrain Data in the index.

If VRtualize implements a First in First Out (FIFO) implementation, the “full” amount of Terrain Data is predetermined, and when “full”, the Local Database will remove the oldest fetched Terrain Data, and the new necessary Terrain Data will be inserted.

If VRtualize implements a Least Recently Used (LRU) implementation, the “full” amount of Terrain Data is predetermined, and each fetched Terrain Data will have a unique time stamp. Every time necessary Terrain Data is already in the Local Database, the time stamp is updated. When the Local Database is “full”, the Local Database will remove the fetched Terrain Data that has the oldest time stamp.

3.2.4 Local Database: Fetch the Data

Brief Description

The Local Database fetches data from a Cooperating System.

Initial Step-By-Step Description

1. Determine the “best” Cooperating System for the fetch.
2. Request necessary Terrain Data from the “best” Cooperating System.
3. Return the Terrain Data to *Prepare Terrain Data*.

An algorithm to determine the “best” Cooperating System per fetch is left to be determined by the development team.

3.3 External Interface Requirements

This section outlines the requirements of the Cooperating Systems necessary for the Local Database to fetch Terrain Data.

The Bing Maps Database does not have any interface requirements necessary for the Local Database to retrieve Terrain Data. However, in order to retrieve data correctly, the application is first required to calculate a specific quad-key based on the desired location’s longitude and latitude coordinates.

The Application assumes the Explorer will access it via VR Equipment

3.4 Non-Functional Requirements

3.4.1 System Characteristics

The Explorer

The Explorer is expected to be familiar with VR Equipment. As well, the Explorer is assumed to be knowledgeable about valid geological data (longitude and latitude) and terminology.

The Local System

The physical machine must install the Local System. The machine hosting the Local System is expected to have these characteristics:

- External Interfaces
  - VR Equipment
  - 1GB/s or better network connection
- Minimum System Specifications – Hardware
  - Quad Core Processor
  - 8GB of RAM
  - 1TB available storage space
  - NVIDIA GTX 970 / AMD Radeon RX 480
  - 1GBps NIC
- Minimum System Specifications – Software
  - Microsoft Windows 10

The Application part of the Local System will provide interactions familiar to users who have used virtual reality before. Buttons, menu options, etc. will behave similar to other virtual reality software, and will be placed in locations similar to other virtual reality software. In places where the Application receives input from the Explorer, the Application assumes the input is valid.

The Local Database part of the Local System will have encapsulated caching logic, and the Application should not need to know how the caching works when requesting Terrain Data. The Local Database must also handle fetching Terrain Data from Cooperating Systems and the Application should not need to interact with Cooperating Systems.

Cooperating Systems

The Local Database expects at least one Cooperative System to be active and connectable.