**System/Subsystem Design Description**

**for the**

**UMBC Virtual Tour 2.0 System**

**Document # CMSC447-05-FA2018-G03-SSDD-01A**

Revision A

20 November 2018

Authors:

Noah Johnson

Ronan Kaye

Tyler Little

Ryan Martin

Kristin McLaughlin

**Revision History**

**0.1 20 November 2018: CMSC447-05-FA2018-G03-SSDD-01A**

Release A contains the basic system/subsystem design description for the UMBC Virtual Tour 2.0 System.

Table of Contents

[**0.1 20 November 2018: CMSC447-05-FA2018-G03-SSDD-01A 2**](#_Toc530521335)

[1 Scope 4](#_Toc530521336)

[**1.1 Identification 4**](#_Toc530521337)

[**1.2 System Overview 4**](#_Toc530521338)

[**1.2.1 Purpose 4**](#_Toc530521339)

[**1.2.2. Development History 4**](#_Toc530521340)

[**1.2.3 Deployment Locations 4**](#_Toc530521341)

[**1.3 Document Overview 4**](#_Toc530521342)

[2 Referenced Documents 5](#_Toc530521343)

[3 System-Wide Design Decisions 5](#_Toc530521344)

[**3.1 3D Engine Selection 5**](#_Toc530521345)

[**3.2 User Access Mode Selection 5**](#_Toc530521346)

[**3.3 User Characteristics 6**](#_Toc530521347)

[**3.4 Constraints 6**](#_Toc530521348)

[**3.5 Assumptions and Dependencies 6**](#_Toc530521349)

[4 System Architectural Design 7](#_Toc530521350)

[**4.1 System Components 7**](#_Toc530521351)

[**4.1.1 Computer Resources 7**](#_Toc530521352)

[**4.1.2 Software Components 8**](#_Toc530521353)

[**4.2 Concept of Execution 25**](#_Toc530521354)

[**4.3 Interface Design 26**](#_Toc530521355)

[**4.3.1 Interface Identification and Diagrams 26**](#_Toc530521356)

[5 Requirements Traceability 28](#_Toc530521357)

[The following table provides tractability to SRS. This table is current as of Revision A. 28](#_Toc530521358)

[6 Notes 31](#_Toc530521359)

[**6.1 Background and Rationale 31**](#_Toc530521360)

[**6.2 Glossary 31**](#_Toc530521361)

[A Appendixes 32](#_Toc530521362)

**1 Scope**

This design description presents the designs used or intended to be used in implementing version 1.0 of a software application enabling virtual tours of the UMBC campus. The designs follow the the requirements identified in the Software Requirements Specification for this project (CMSC447-05-FA2018-G03-SSDD-02A).

**1.1 Identification**

Title: UMBC Virtual Tour 2.0

Abbreviation: VT2

Version Number: 1.0

**1.2 System Overview**

### 1.2.1 Purpose

The purpose of the VT2 system is to improve the existing basic UMBC virtual campus tour applications by importing the UMBC campus map and building information into the Unity game engine, enabling users to explore the campus freely in 3D. Additionally, the system offers other useful features, such the ability to highlight valid parking locations on campus based on user status. The intended users of the system are prospective students seeking to familiarize themselves with the campus environment and current students, faculty, and visitors trying to find their classes or event venues and seeking the best place to park.

### 1.2.2. Development History

Development of the system began in September 2018, with a prototype of version 1.0 of the system scheduled for completion in early December 2018. The project is sponsored by the UMBC Department of Computer Science and Electrical Engineering, and the development team consists of senior computer science majors at UMBC. If successful, the project will be acquired by UMBC and incorporated into university’s website in the future.

### 1.2.3 Deployment Locations

The only planned operating site for the software is the UMBC main campus located in Baltimore, Maryland. During the next phase of development, however, the software will be extended to include the UMBC campus at the Universities at Shady Grove, located in Rockville, MD.

**1.3 Document Overview**

This document is organized as follows: Section 1 identifies the scope of this document and lists the definitions, abbreviations, acronyms, and references used therein. Section 2 provides an overview of the system and a brief description of its architecture.

**2 Referenced Documents**

The following standards apply:

CMSC447-05-FA2018-G03-SRS-01A https://github.com/noahj1/UMBC-VT-2.0

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

MIL-STD-498 Military Standard Software Development and Documentation

UMBC Style Guide https://styleguide.umbc.edu/

Unity User Manual (2018.2) https://docs.unity3d.com/Manual/index.html

WebGL Manual https://docs.unity3d.com/Manual/webgl.html

**3 System-Wide Design Decisions**

This section details significant design decisions and other issues related to the design of the VT2 system.

**3.1 3D Engine Selection**

To meet the requirement of rendering the UMBC campus in 3D, the development team decided to select a commercially available 3D game engine. Game engines are designed to render realistic 3D worlds and are therefore suitable for the needs of the VT2 project. The development team selected Unity (developed by Unity Technologies) to create the 3D UMBC campus environment for several reasons. First, by some measures, it is the most widely used game engine in the world today[[1]](#footnote-1). The system’s current developers are familiar with it, and future developers working on the project will be more likely to have experience with Unity than with a less well-known engine. Second, Unity provides strong support for browser-based access, as will be dicussed further below. Finally, Unity offers free licenses for students and academic institutions.

**3.2 User Access Mode Selection**

In order to satisfy requirements for flexibility, availability, and maintainability, the design employs a browser-based client-server architecture. The code for the customized Unity engine will be stored on a web server that users access over the internet through a web browser. The server will take advantage of the WebGL framework to enable rendering of the Unity engine’s 3D graphics within the client’s web browser. This process is explained in greater detail in section 4.

The alternative to a browser-based clienter server architecture would be to make the customized Unity world available to users for download. Users would first have to download a copy of the Unity engine, then import the customized world. While this approach would make the system available offline and avoid potential network bandwidth and congestion issues, it would have a number of serious disadvantages. For example, users who otherwise would likely have no reason to download and install the Unity engine (a 9 Gigabyte) would have to do so just to run the software and would have to worry about updating it to the latest version. Additionally, users would have to download a new version of the VT2 software every time the design team makes an update to it. For these reasons, the browser-based client-server architecture is the best option to meet project requirements.

**3.3 User Characteristics**

Users of this system are assumed to possess basic familiarity with internet browsers and websites, but no other special knowledge or skills are required. All user interaction with the system will take place through browser-based menus, and the system will prompt the user to take action with clear and simple instructions when necessary.

**3.4 Constraints**

According to the Unity manual (docs.unity3d.com/Manual/webgl-browsercompatibility.html), the WebGL framework that allows the Unity engine to be run within a web browser is not currently supported on mobile devices because most mobile devices are not powerful enough and lack sufficient memory to adequately support Unity WebGL content. The content may work on high-end devices, but by default, Unity WebGL displays a warning message when a user tries to load content on a mobile device. Therefore, version 1.0 of the VT2 system will not support mobile devices, and users should acess the system via desktop or laptop computers instead. Future versions of the system will provide support for mobile computing using a different framework. Additionally, users must access the system through a compatible browser as described in the Unity WebGL manual. Most modern browsers (Firefox, Chrome, Safari, Microsoft Edge) are supported.

The system relies on UMBC’s Imaging Research Center (IRC) to provide the object files necessary for creating three-dimensional renderings of the campus buildings. As of late November 2018, the IRC did not have renderings and textures for some of the buildings. As a result, version 1.0 of the VT2 system will only contain a subset of the 43 buildings that comprise the UMBC main campus.

Finally, version 1.0 of the software will not provide the user with directions to or from parking lots or buildings. It is intended that the system will provide this capability in a future release.

## 3.5 Assumptions and Dependencies

It is assumed that the CSCIs described in this version of the SSDD are the basic CSCIs necessary to meet customer requirements. Once the design team has successfully implemented these CSCIs, the team may proceed with implementation of “reach goals” such as direction-finding algorithms and mobile support upon agreement with the customer.

The VT2 system will also contain links to the main UMBC website (www.umbc.com). If the UMBC website were to become unavailable, some features of the software would cease to work, but the system’s core functionality would be unaffected.

**4 System Architectural Design**

## 4.1 System Components

The following subsections address the computer resources and hardware and software components of the UMBC VT2 system. These components include both modules developed by the design team and commercially available components.

The hardware resources used by the VT2 system are described in subsection 4.1.1. All of the system's software components (including CSCIs) are installed within the hardware resources and are described in the remaining subsections.

The hardware resources used by the VT2 system are as follows:

* *Server*: The webserver that hosts the system’s CSCIs.

The system includes the following CSCIs:

* *Virtual Unity Engine (VUE) CSCI*: Customized version of the Unity engine that includes accurate 3D renderings and textures of the UMBC campus buildings and allows natural movement around the campus with motion and camera effects similar to first and third-person point of view video games. It also provides the ability to identify parking lots where the user is allowed to park based on the user’s status (faculty member, commuter student, residential student, visitor, etc.).
* *Virtual Tour Interface (VTI) CSCI*: Provides a menu-based web interface for the VT2 system based on the WebGL framework. It manages the user’s interaction with the VUE CSCI.

There are no Hardware Configuration Items (HWCIs) associated with the system.

Aside from user interactions, there are no manual operations associated with the VT2 system.

### 4.1.1 Computer Resources

#### 4.1.1.1 Server

The VT2 system uses a LiteSpeed web server to host the customized Unity engine and other CSCIs. However, any web server that meets or exceeds the following requirements could be used as the system’s server:

* Operating System (OS) Requirements
  + Windows 7 SP1+
  + macOS 10.11+
  + Ubuntu 12.04+
  + Steam OS+
* Hardware Requirements
  + Graphics card with DX10 (shader model 4.0) capabilities
  + CPU with SSE2 instruction set support
  + Minimum 2.4 GHz processor with 8 Gigabytes of memory

#### 4.1.1.2 Client Hardware Requirements

The client computer must be a PC or a laptop. It should have at least a 2.4 GHz processor and 4 Gigabytes of memory.

Figure 1 depicts the computer hardware resources and requirements for both the clients and server for the VT2 system.

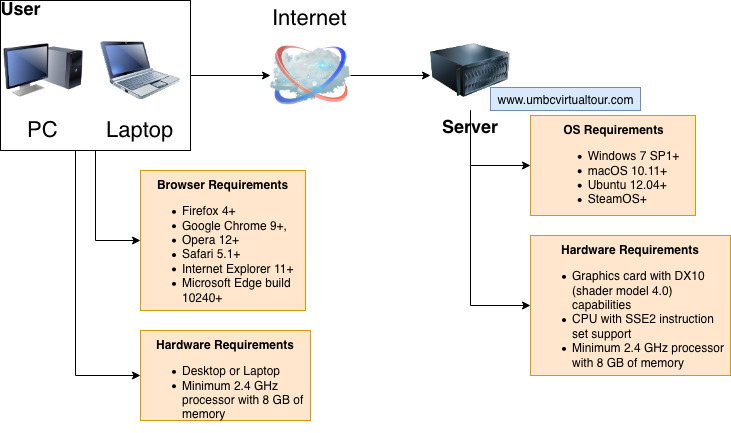


Figure 1 System Hardware Resources and Requirement

**4.1.2 Software Components**

The VT2 system consists of a customized Unity engine with a web interface. The customized Unity engine—the VUE CSCI—incorporates several Computer Software Components (CSCs). The web interface is provided through the VTI CSCI. Figure 2 shows the relationship between the system CSCIs. The subsections that follow explain each CSCI in detail.

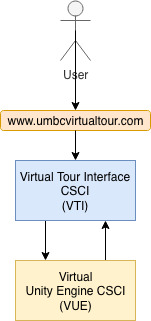


Figure 2 Relationship Between VT2 CSCIs

#### 4.1.2.1 Virtual Unity Engine (VUE) CSCI

The Virtual Unity Engine (VUE) CSCI is a customized version of the Unity game engine that provides a 3D rendering of the UMBC campus. The VUE in the VT2 1.0 system is based on Unity Personal version 2018.2.13 (available at https://store.unity.com/download).

To simplify development, the VT2 system uses two different versions of the VUE: VUE\_CampusExplorer and VUE\_ParkingFinder. Each of these versions is a CSC under the VUE CSCI. This design decision allows for easy switching between the VCE and VPF modes from the VTI and avoids having to shift between two different modes while in the same game, which is more complicated to implement. These two CSCs are described in subsections 4.1.2.1.1 and 4.1.2.1.2.

The following steps were taken to enhance the basic version of Unity to create the 3D map of the UMBC campus. The VUE\_CampusExplorer CSC completes step 9a (camera and player controller modules), while the VUE\_ParkingFinder CSC completes step 9b (parking lot highlighting):

1. Export a map selection containing the UMBC campus from OpenStreetMap (www.openstreetmap.org/) as a .osm file.
2. Convert the OpenStreetMap .osm file to a 3D object model (.obj file) using OSM2World (osm2world.org/).
3. Import the 3D object model of the UMBC map into Unity, creating a basic campus map.
4. Obtain models (.obj and .mtl files) and textures (.png and .tga files) for 25 UMBC campus buildings from the UMBC IRC.
5. Import the models and textures into Unity and add them to the basic campus map.
6. Enhance the map with additional features such as grass, water, and trees to increase realism.
7. Add object height mapping features to make movement and terrain more realistic.
8. Add name labels for the campus buildings
9. a) Add camera and player control modules

b) Add parking lot highlighting modules

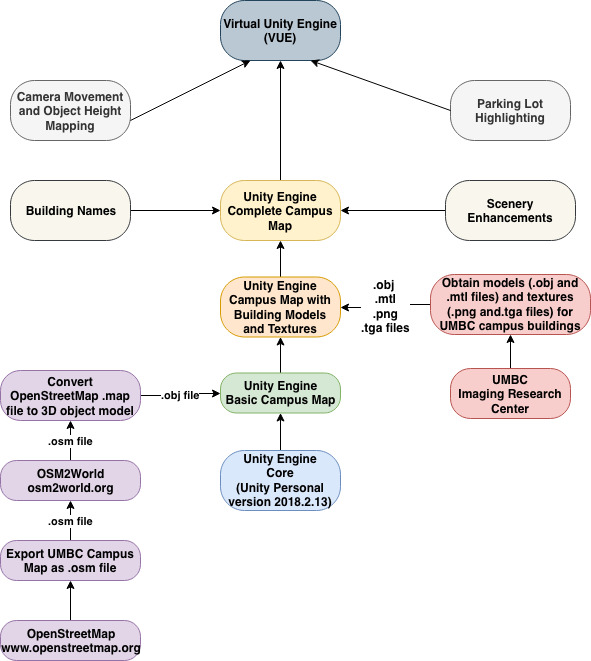


Figure 3 Development of Virtual Unity Engine (VUE) CSCI

The VUE CSCI is composed of the following CSCs:

* VUE\_CampusExplorer
* VUE\_ParkingFinder
* VUE\_BasicMap
* VUE\_BuildingModels
* VUE\_HeightMapping
* VUE\_Scenery
* VUE\_Camera
* VUE\_Highlighting

Figure 4 shows the relationship between the CSCs in the VUE CSCI.

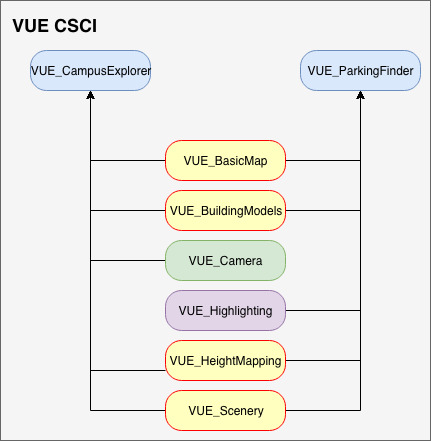


Figure 4 Relationship Between the CSCs in the VUE CSCI

##### 4.1.2.1.1 VUE\_CampusExplorer

The VUE\_CampusExplorer CSC replicates all the functionality of the Virtual Campus Explorer (VCE) CSCI specified in the SRS for this project and replaces it. The VUE\_CampusExplorer enables the user to explore the campus freely in first or third-person camera modes. The CSC is launched by a menu option in the VTI CSCI, as described in section 4.1.2.2.

##### 4.1.2.1.2 VUE\_ParkingFinder

The VUE\_FindParking CSC, combined with the VTI CSCI described in section 4.1.2.2, replicates all the functionality of the Virtual Parking Finder (VPF) CSCI specified in the SRS for this project and replaces it. The VUE\_FindParking CSC allows the user to identify the closest parking lots to a specified building, but does not include the ability to explore the campus map. The user will still have the ability to zoom in and out of map locations, however. Additional functionality associated with the VUE\_ParkingFinder CSC is found in the discussion of the VPF CSCI in section 4.1.2.3.

##### 4.1.2.1.3 VUE\_BasicMap

The VUE\_BasicMap CSC includes two components: the UMBC campus map acquired from Open Street Map in .osm file format, and the 3D .obj file generated by OSM2World. The .obj file can be imported directly into Unity to serve as the backbone for the customized 3D world. Figure 4 belows shows a screenshot of the basic 3D UMBC campus map object.

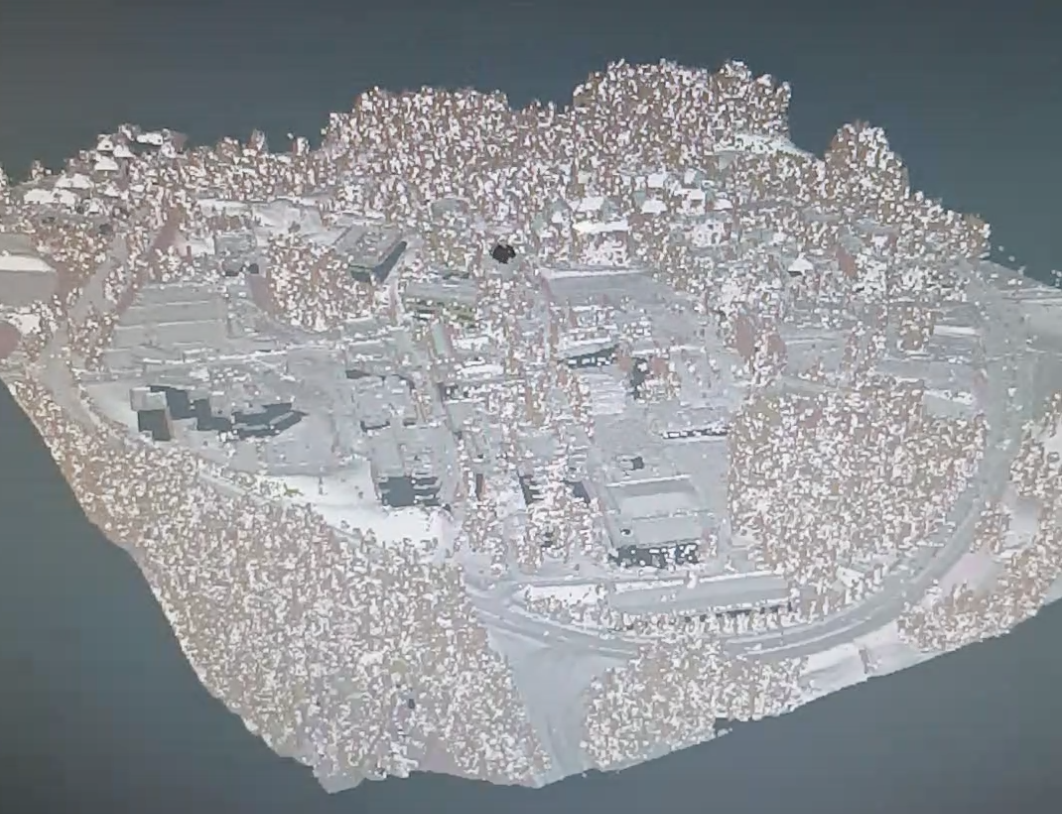


Figure 5 Basic 3D Object Model of UMBC Campus

##### 4.1.2.1.4 VUE\_BuildingModels

The VUE\_BuildingModels CSC includes model and texture files for 25 UMBC campus buildings acquired from the UMBC IRC. The buildings and associated files and textures are identified in Tables 1 and 2 below.

Table 1 List of UMBC Building Model Files

|  |  |
| --- | --- |
| **Building Name** | **Files** |
| Administration Building | Admin.mtl  Admin.obj |
| Biological Sciences Building | BiologicalSciences.mtl  BiologicalSciences.obj |
| Campus Police and Central Plant | CampusPoliceCntrlPlant.mtl  CampusPoliceCntrlPlant.obj |
| Chesapeake Hall | ChesapeakeHall.mtl  ChesapeakeHall.obj |
| The Commons | Commons.mtl  Commons.obj |
| Engineering Building | Engineering.mtl  Engineering.obj |
| Erickson Hall | EricksonHall.mtl  EricksonHall.obj |
| Event Center | EventCenter.mtl  EventCenter.obj |
| Fine Arts Building | FineArts.mtl  FineArts.obj |
| Harbor Hall | HarborHall.mtl  HarborHall.obj |
| Information Technology/Engineer Building | Ite.mtl  Ite.obj |
| Mathematics & Psychology Building | Math.mtl  Math.obj |
| Meyerhoff Chemistry Building | Meyerhoff.mtl  Meyerhoff.obj |
| Performing Arts and Humanities Building | Pahb.mtl  Pahb.obj |
| Patapsco Hall | PatapscoHall.mtl  PatapscoHall.obj |
| Physics Building | Physics.mtl  Physics.obj |
| Potomac Hall | PotomacHall.mtl  PotomacHall.obj |
| Public Policy Building | PublicPolicy.mtl  PublicPolicy.obj |
| Retriever Activities Center (RAC) | Rac.mtl  Rac.obj |
| Sherman Hall | Sherman.mtl  Sherman.obj |
| Sondheim Hall | Sondheim.mtl  Sondheim.obj |
| Student Development & Success Center (SDSC) | StudentSuccessCntr.mtl  StudentSuccessCntr.obj |
| Susquehanna Hall | SusquehannaHall.mtl  SusquehannaHall.obj |
| True Grit’s | TrueGrits.mtl  TrueGrits.obj |
| University Center | UniversityCenter.mtl  UniversityCenter.obj |

Table 2 List of UMBC Building Texture Files

|  |  |
| --- | --- |
| **Texture Type** | **Files** |
| Brick01\_Tex | T\_Brick01\_D.tga |
| Brick02\_Tex | T\_Brick02\_D.png  T\_Brick02\_N.png  T\_Brick02\_ORM.png |
| Brick03\_Tex | T\_Brick03\_D.png  T\_Brick03\_N.png  T\_Brick03\_ORM.png  T\_SkchupEricksonBrick\_D.png  T\_SkchupEricksonBrick\_N.png  T\_SkchupEricksonBrick\_ORM.png |
| Brick04\_Tex | T\_Brick04\_D.png  T\_Brick04\_N.png  T\_Brick04\_ORM.png |
| BrickLmstn01\_Tex | T\_BrickLmstn01\_D.png  T\_BrickLmstn01\_N.png  T\_BrickLmstn01\_ORM.png |
| Library\_Tex | Lib\_MainSect01\_Col\_Tex.png  Lib\_MainSect02\_Col\_Tex.png  Lib\_Shtrs\_Col\_Tex.png |
| Lmstn01\_Tex | T\_Lmstn01\_D.png  T\_Lmstn01\_N.png  T\_Lmstn01\_ORM.png |

##### 4.1.2.1.5 VUE\_Camera

The VUE\_camera CSC consists of two components: a camera controller package and a script that enables switching between first- and third-person (1P and 3P) camera views.

The camera controller component uses the Third Person Controller Basic Locomotion Template developed by Invector, available for free at the Unity Asset Store (https://assetstore.unity.com/packages/templates/systems/third-person-controller-basic-locomotion-free-82048). This package allows the creation of player models that have a 3P controller and camera input/output.

The package is used to create two separate player models. Initially, both models have a 3P camera view. The camera view of one of the models is then modified by reducing the default camera distance (from 2.4 to 0) so that it is directly over the head of the player, creating the appearance of a 1P camera view.

When the game is initiated, the camera change script takes control of both player models. One of the player models is deactivated while the other player model is set as active. The user can switch between the player models (between 3P and 1P views) by pressing the *c* button. The script then activates the deactivated model and deactivates the active one. The positions of the player and camera are copied to the activated player model to save the location and current view of the player while in the same mode.

Figure 5 depicts the camera change script controlling switching between the two player models.

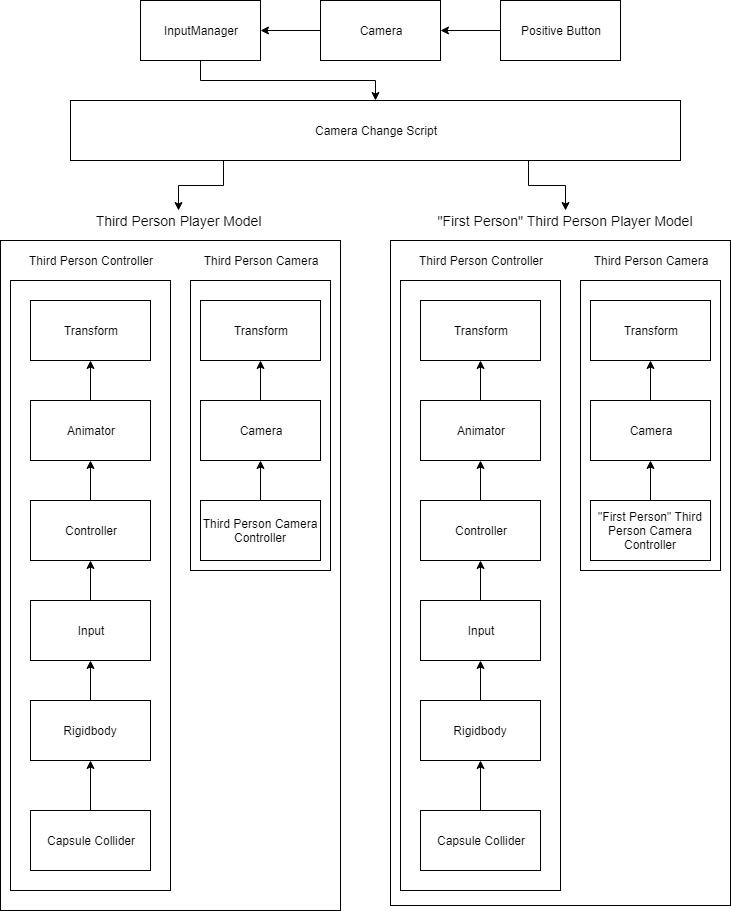


Figure 6 Camera Change Script Controlling 3P and 1P Player Models in the VUE\_Camera CSC

The definitions of each module in Figure 5 are as follows:

* *Transform*: Controls the position, rotation, and scale of an object (such as a camera or player model)
* *Animator*: Interface to control the animation system; animates the player model
* *Controller*: Determines values and mechanisms for player model movement (such as strafing, sprinting, and jumping)
* *Input*: Handles input to the player model and the camera attached to the player model
* *Rigidbody*: Adds physics to the Player Model (such as gravity effects)
* *Capsule Collider*: A capsule object to attach to the player model that makes the character "solid" (without this module, the model would fall through the floor in a game)
* *Camera*: Component that showcases the world to the player
* *Third Person Camera Controller*: Calculates input to simulate a third person camera point of view with relation to a target player model

##### 4.1.2.1.6 VUE\_HeightMapping

The VUE\_HeightMapping CSC includes scripts imported into the VUE to provide realistic rendering of the differing heights of buildings.

##### 4.1.2.1.7 VUE\_Scenery

The VUE\_Scenery CSC includes textures and objects imported into the VUE to provide additional scenery features, such as grass, trees, and water.

#### 4.1.2.2 Virtual Tour Interface (VTI) CSCI

The Virtual Tour Interface (VTI) CSCI provides a web interface to the VUE, VCE, and VPF CSCIs, as well as other basic website functionality. The website for the VT2 system is www.umbcvirtualtour.com. The website has no login features and stores no data about the user.

*WebGL Framework and Webserver File Structure*

The VTI uses the WebGL framework to enable rendering of the Unity engine’s 3D graphics within the client’s web browser. The WebGL process works as follows:

* The developer creates code in the Unity engine using C# scripts and the .NET framework and chooses to build and run the code as a WebGL build target.
* The .NET bytecode is converted into corresponding C and C++ source files using Intermediate Language to C++ (IL2CPP).
* The emscripten compiler toolchain is used to cross-compile the Unity runtime code (written in C and C++) into asm.js JavaScript, an optimized subset of JavaScript that allows JavaScript engines to Ahead-of-Time (AOT) comple asm.js code into efficient native code.
* After building the project, Unity creates a *Build* folder containing project files an an *index.html* file to which browsers can navigate to load content.

Figure 7 depicts the WebGL process. Figure 8 shows the files created after building the WebGL target in the context of the overall webserver file structure. These files include code to generate a progress bar showing how fast the Unity build is loading (not shown here).

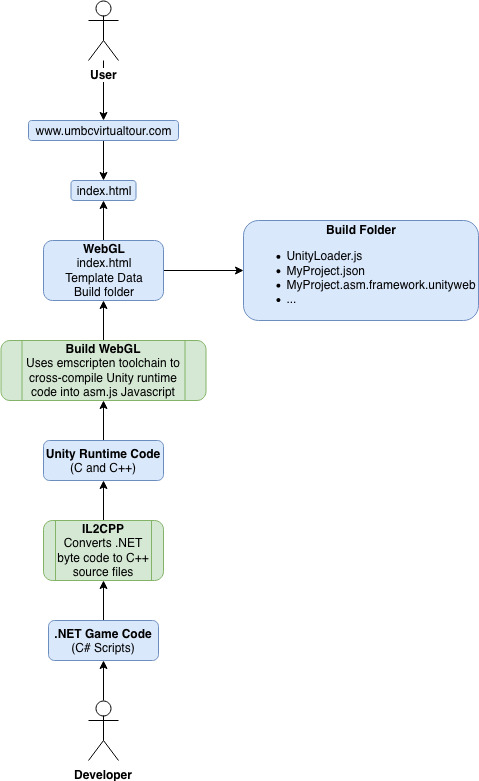


Figure 7 WebGL Framework

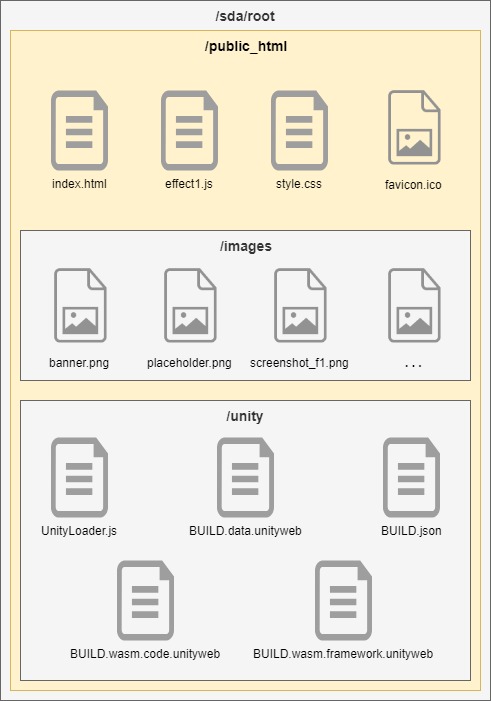


Figure 8 VT2 Web Server File Structure

*Website Layout and Menu System*

Figure 9 depicts the layout of the VT2 website. The website presents the user with the following options, each of which links to a new page on the website and, in some cases, launches a functional module. Within each page, the user has the option to hit a button to return to the main page.

* *About this Website*: Links the user to a page of the site that explains the VT2 system’s purpose
* *Help*: Links the user to a page of the website that provides tutorials on the Explore Campus and Find Parking functions
* *Explore Campus*: Links the user to a page that launches the VUE\_CampusExplorer CSC
* *Find Parking*: Links the user to a page that launches the VUE\_ParkingFInder CSC

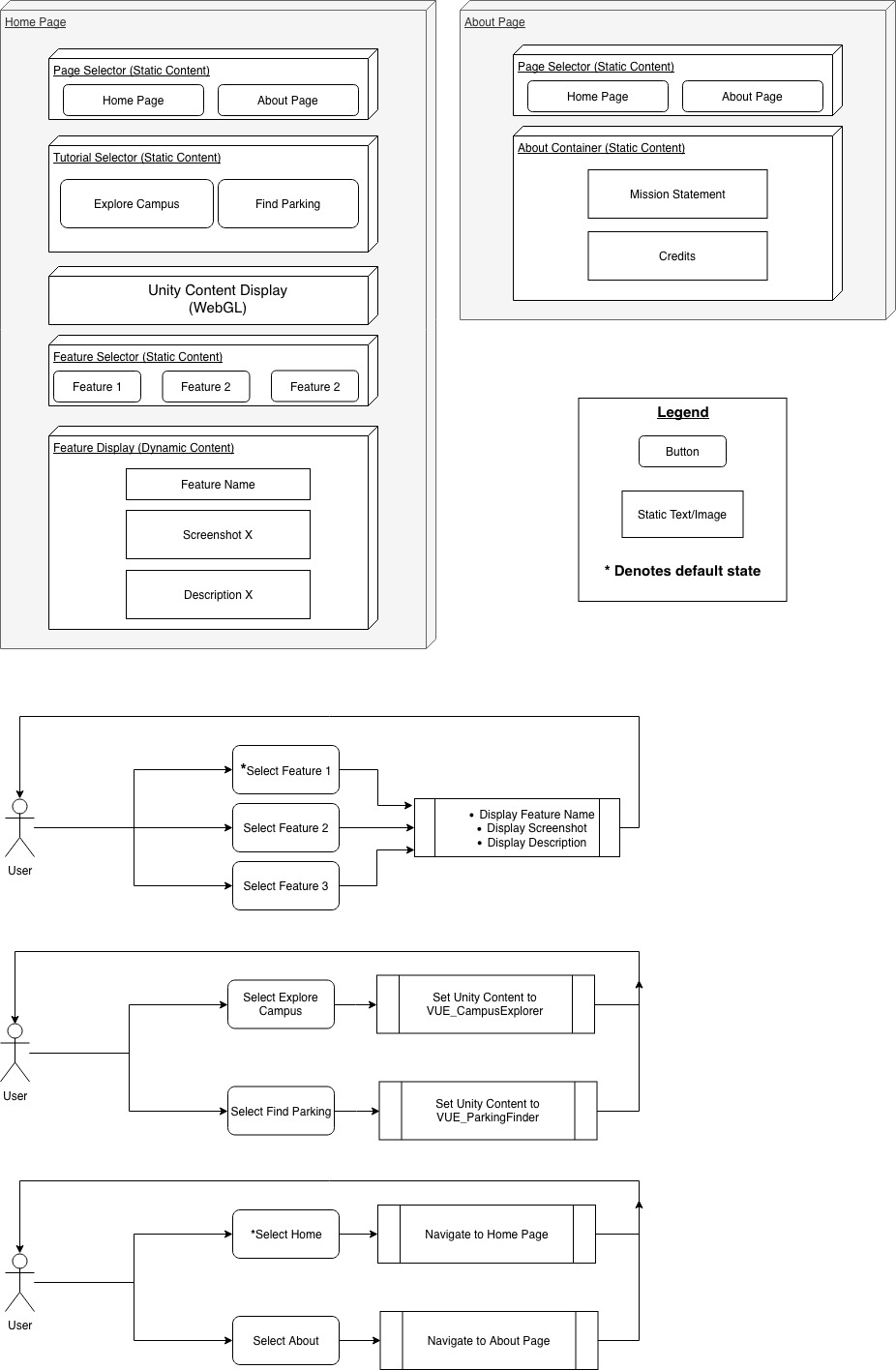


Figure 9 VT2 Website Layout

4.1.2.2.1 About this Website

The “About this Website” page provides a brief introduction to UMBC and identifies the purpose of the website as providing virtual tours of the university.

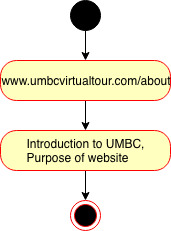


Figure 10 Activity Diagram for About this Website

4.1.2.2.2 Help

The “Help” provides tutorials on how to use the Explore Campus and Find Parking options.

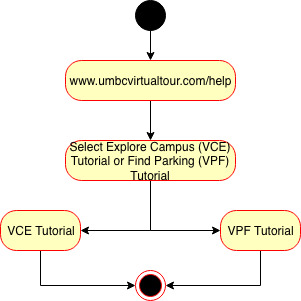


Figure 11 Activity Diagram for Help

4.1.2.2.3 Explore Campus

Selecting “Explore Campus” takes the user to a new page on the website that launches the VUE\_CampusExplorer CSC. The below diagram shows the activity path for the Explore Campus option.

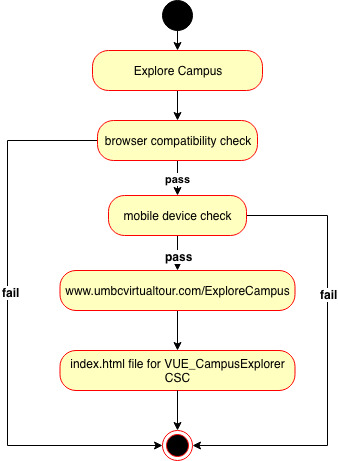


Figure 12 Activity Diagram for Explore Campus

4.1.2.2.4 Find Parking

Selecting “Find Parking” opens a new page on the website that launches the VUE\_ParkingFinder CSC and takes the user into the following activity path:

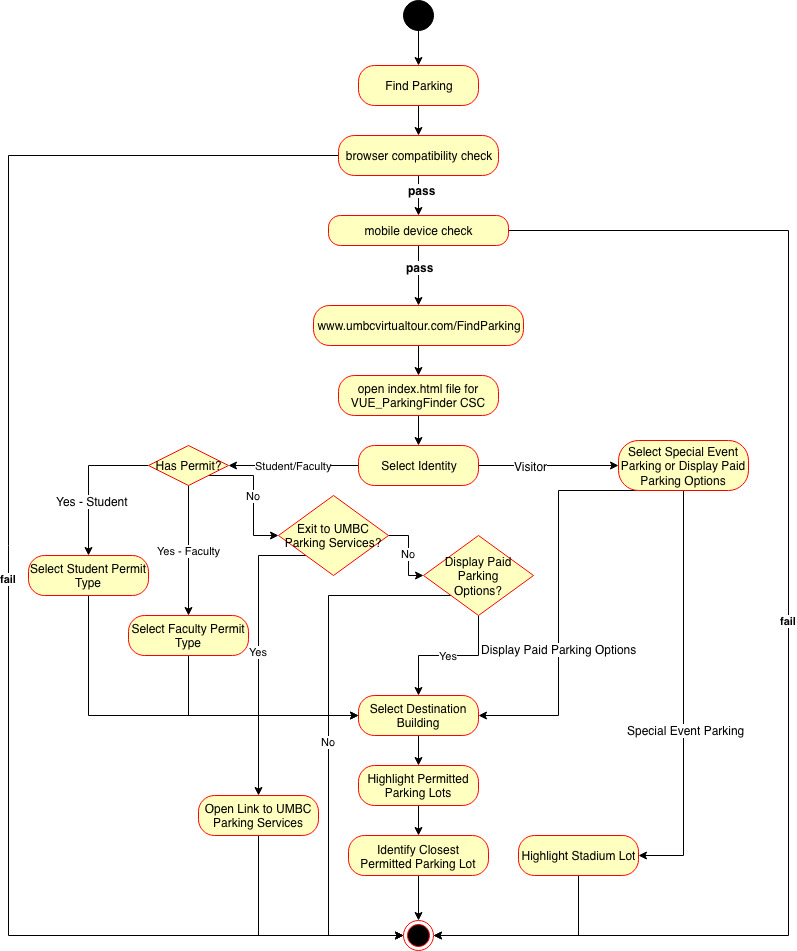


Figure 13 Activity Diagram for Find Parking

## 4.2 Concept of Execution

The following diagram shows the finite state machine and flow of execution for the VT2 system.

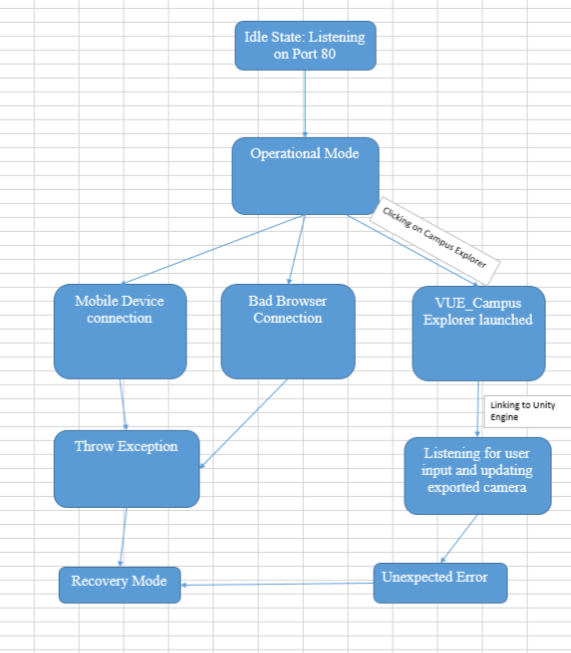
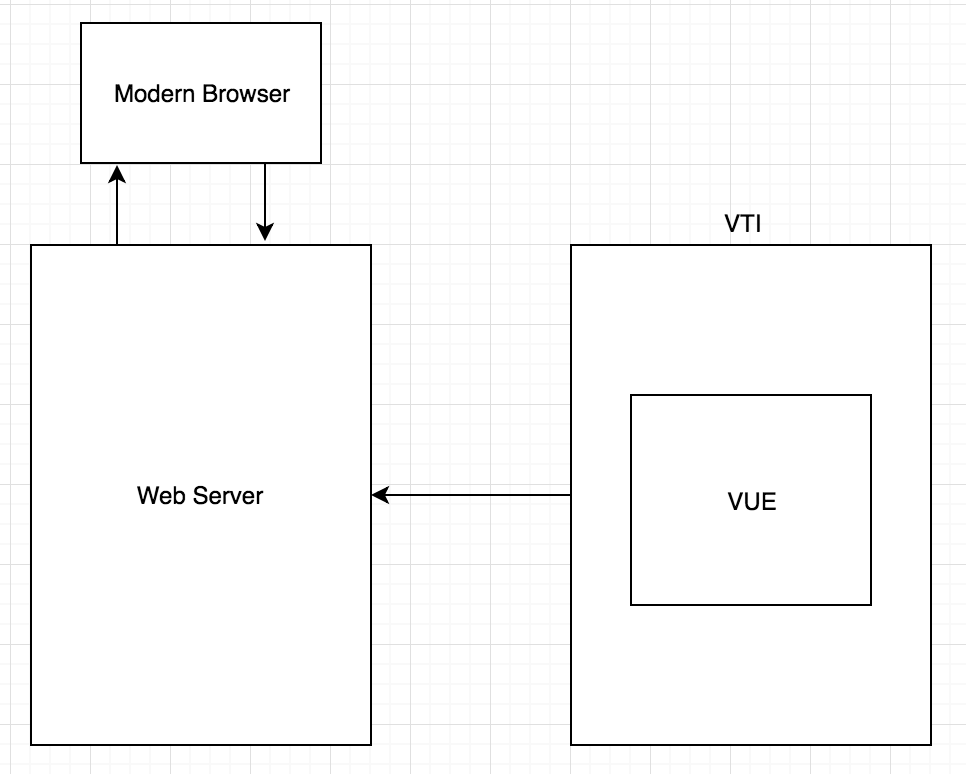


Figure 14 Finite State Machine and Flow of Control for VT2 System

## 4.3 Interface Design

The goal of this section is to elaborate the interface characteristics of the UMBC Virtual Tour 2.0 system components previously analyzed in subsection 4.1.2.

As shown in Figure 15, the Virtual Tour Interface CSCI is a website utilizing UMBC web hosting services which encapsulates the VUE CSCI (for simplicity, the VUE represents both the VUE\_CampusExplorer and the VUE\_ParkingFinder). The VUE is comprised of two separate HTML embedded 3D tours of the UMBC campus developed in Unity.



*Figure 15 System Component of VTI*

### 4.3.1 Interface Identification and Diagrams

As the VTI is a hosted website, all communication will take place between the user’s web browser and the web server. As such, the VTI is the sole interface which facilitates the virtual tour. This method of web hosting is implemented with the notion of ease of access for users who wish to explore the UMBC campus in a more convenient manner.

The VUE is encapsulated within the VTI as a static webpage. On the VTI homepage, the user must click the “Virtual Tour” or “Free Roam” buttons to access the VUE\_ParkingFinder and VUE\_CampusExplorer respectively (see Figure 16).



**5 Requirements Traceability**

The following table provides tractability to SRS. This table is current as of Revision A.

Table Requirements Traceability

|  |  |  |  |
| --- | --- | --- | --- |
| **SRS (Version 01A) Paragraph Number** | **Capability** | **SSDD (Version 01A) Paragraph Number** | **CSCI/CSC** |
| 3.2.1.1 | Displaying the VTI through website | 4.1.1.1, 4.1.2.2 | VTI |
| 3.2.1.2 | VTI built with HTML5, ECMAScript 2019 | 4.1.2.2 | VTI |
| 3.2.1.3 | Use of WebGL Javascript API | 4.1.2.2 | VTI |
| 3.2.1.4 | Display VTI Main Page | 4.1.2.2 | VTI |
| 3.2.1.5 | VTI Home Page Details | 4.1.2.2 | VTI |
| 3.2.1.6 | VTI Menu Display | 4.1.2.2 | VTI |
| 3.2.1.7 | Display “About this Website” page | 4.1.2.2, 4.1.2.2.1 | VTI |
| 3.2.1.8 | Campus Explorer | 4.1.2.2.3, 4.1.2.1.1 | VTI, VUE\_CampusExplorer |
| 3.2.1.9 | Parking Finder | 4.1.2.2.4, 4.1.2.1.2 | VTI, VUE\_ParkingFinder |
| 3.2.1.10 | Help Tutorial | 4.1.2.2.2 | VTI |
| 3.2.1.11 | Allows User to Return to VTI Main Page | 4.1.2.2 | VTI |
| 3.2.1.12 | Prevents Storage of User Data | 4.1.2.2 | VTI |
| 3.2.1.13 | Functionality for Specified Web Browsers | 4.1.2.2.3, 4.1.2.2.4 | VTI, VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.2.1.14 | Functionality When Not Using a Specified Web Browser from 3.2.1.13 | 4.1.2.2.3, 4.1.2.2.4 | VTI, VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.2.1.15 | Error Message When on Mobile Devices | 4.1.2.2.3, 4.1.2.2.4 | VTI, VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.2.2.1 | Quickly Load VUE CSCs | 4.1.1.1., 4.1.2.2 | VTI, VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.2.2.2 | Initial View of VUE\_CampusExplorer | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.2.3 | Double-click on location to start exploring | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.2.4 | Zoom to 3P view after double-clicking | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.2.5 | Error when clicking on non-walkable areas | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.2.6 | Avatar is True Grit | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.2.7 | Shifting camera views | 4.1.2.1.5 | VUE\_Camera |
| 3.2.2.8 | 3P Player and camera movement | 4.1.2.1.5 | VUE\_Camera |
| 3.2.2.9 | 1P Player and camera movement | 4.1.2.1.5 | VUE\_Camera |
| 3.2.2.10 | Player movement | 4.1.2.1.5 | VUE\_Camera |
| 3.2.2.11 | Display overhead map in 1P and 3P views | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.3.1 | Find Parking – Get identity | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.2 | Find Parking – Get type of student permit | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.3 | Find Parking – Get type of faculty permit | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.4 | Find Parking – No permit option | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.5 | Find Parking – Select building | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.6 | Find Parking – Highlight valid lots | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.7 | Find Parking – Visitor menu options | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.8 | Find Parking – Highlight stadium lot for special event | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.9 | Find Parking – Paid parking options select building | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.3.10 | Find Parking – Highlight valid paid parking lots | 4.1.2.2.4 | VTI, VUE\_ParkingFinder |
| 3.2.4.1 | Unity version | 4.1.2.1 | VUE |
| 3.2.4.2 | Use OSM for base map | 4.1.2.1.3 | VUE\_BasicMap |
| 3.2.4.3 | Add building models and textures | 4.1.2.1.4 | VUE\_BuildingModels |
| 3.2.4.4 | Include 25 required campus buildings | 4.1.2.1.4 | VUE\_BuildingModels |
| 3.2.4.5 | Appearance of building names in overhead view of map | 4.1.2.1.1, 4.1.2.1.2 | VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.2.4.6 | Appearance of building names in 1P and 3P view of map | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.4.7 | Building heights | 4.1.2.1.6 | VUE\_HeightMapping |
| 3.2.4.8 | Camera location | 4.1.2.1.5 | VUE\_Camera |
| 3.2.4.9 | Movement only allowed on walkable areas | 4.1.2.1.1 | VUE\_CampusExplorer |
| 3.2.4.10 | Scenery | 4.1.2.1.7 | VUE\_Scenery |
| 3.2.4.11 | Display lot numbers | 4.1.2.1.1, 4.1.2.1.2 | VUE\_CampusExplorer, VUE\_ParkingFinder |
| 3.9.1 | Server hardware requirements | 4.1.1.1 |  |
| 3.9.2 | Browser access requirements | 4.1.1.1, 4.1.2.2.3, 4.1.2.2.4 | VTI |
| 3.10.1 | Server hardware requirements | 4.1.1.1 |  |
| 3.10.2 | Server hardware requirements | 4.1.1.1 |  |
| 3.10.3 | Server hardware requirements | 4.1.1.1 |  |

**6 Notes**

## 6.1 Background and Rationale

A virtual campus tour is an important component of a university’s strategy for recruiting students. In an environment of intense competition for students of all types—domestic and international, in-state and out-of-state, and undergraduate and graduate—a strong virtual tour application can convince a prospective student to apply or visit the campus in person. Additionally, virtual campus tours can help current students and visitors navigate their way to their classes or special events.

UMBC currently has several websites that nominally offer virtual tours of the campus. The Undergraduate Admissions UMBC Virtual Tour (located at undergraduate.umbc.edu/visit/virtual-tour.php) provides 9 panoramic views of the campus, though it claims to offer 25 views. A virtual tour site for the graduate school (gradschool.umbc.edu/discover/vtour/) simply provides a link to the same site that hosts the panoramic campus views noted above. Additionally, a UMBC undergraduate student created a basic virtual tour mobile application for Android devices in 2014 titled, “Introducing UMBC Tours - A Virtual Campus Tour Experience for Android” (www.youtube.com/watch?v=zRI61jkUDT4). However, this implementation had extremely limited functionality and did not represent a significant improvement on the applications offered on the UMBC website.

The purpose of the UMBC VT2 software described in this design document is to dramatically improve the currently available UMBC virtual tour applications by importing the UMBC campus map and building information into the Unity game engine. It will allow users to select any location on a three-dimensional map of the campus and allow them to explore it freely. It will provide browser-based access to this system through a web application. Moreover, the new system will provide several other useful features, including the ability to highlight valid parking locations based on user status. The system will primarily benefit prospective students seeking to familiarize themselves with the campus environment and current students, faculty, and visitors trying to find their classes or event venues and seeking the best place to park.

## 6.2 Glossary

API Application Programming Interface

CSC Computer Software Component

CSCI Computer Software Configuration Item

GUI Graphical User Interface

HWCI Hardware Configuration Item

HTML Hyper Text Markup Language

IL2CPP Intermediate Language to C++

IRC Imaging Research Center

MTL File A Material Library (.mtl) file contains one or more material definitions, each of which includes the color, texture, and reflection map of individual materials. These are applied to the surfaces and vertices of objects and are stored in ASCII format.

OBJ File An object (.obj) file is a standard 3D image format that can be exported and opened by various 3D image editing programs. It contains a three-dimensional object including 3D coordinates, texture maps, polygonal faces, and other object information.

OSM Open Street Map

SIMD Single Instruction, Multiple Data

SRS Software Requirements Specification

SSE2 Streaming SIMD Extensions 2

TCP Transmission Control Protocol

TGA File A Truevision Graphics Adapter (.tga) file is a raster graphics file format that can store raw or compressed images.

UMBC University of Maryland, Baltimore County

Unity The Unity cross-platform game engine

VCE Virtual Campus Explorer CSCI

VPF Virtual Parking Finder CSCI

VTI Virtual Tour Interface CSCI

VT2 UMBC Virtual Tour 2.0

VUE Virtual Customized Unity Engine CSCI

WebGL The Web Graphics Library, a cross platform JavaScript API for rendering 2D and 3D graphics in a web browser

XML Extensible Markup Language

UDP User Datagram Protocol

**A Appendixes**

1. https://venturebeat.com/2017/03/01/game-engine-ceos-talk-past-each-other-when-it-comes-to-statistics/ [↑](#footnote-ref-1)