[1.]Petar ČISAR, [2.]Sanja MARAVIĆ ČISAR

DEVELOPMENT CONCEPTS OF VIRTUAL REALITY SOFTWARE

[1.]University of Criminal Investigation and Police Studies, Cara Dušana 196, 11080 Zemun–Belgrade, SERBIA
[2.]Subotica Tech, Department of Informatics, Marka Oreškovića 16, 24000 Subotica, SERBIA

Abstract: The evolution of appropriate hardware and software platforms has resulted in intensive increase in such applications that have placed their action in the virtual world. This enables users a special experience – the experience of virtual reality (VR). The software for developing VR is in many ways specific and demanding. Having in mind its great diversity, this paper gives an overview of categorizations, general characteristics and evaluation methods. In addition, general principles of VR software development are especially elaborated, with an emphasis on creating and testing games.
Keywords: virtual reality, software, development, evaluation, game, testing

INTRODUCTION

Virtual reality (VR) is a completely three–dimensional (3D) environment created by a combination of appropriate software and hardware. This software immerses the user into the 3D environment, giving him the ability to interact with the virtual world in about a realistic way [1].

A few steps are required to create a good VR experience. The virtual world is created by software developers and then rendered in a way that users can interact with the objects. Headsets help provide users the illusion of being completely immersed in the 3D environment. These 3D objects tend to respond to changes in the user's movement and the interactions simulate those in the real world. Some additional devices, such as data gloves, can also simulate human senses (for instance, touch).

Speaking of computer reality types, it is needed to make a clear difference between augmented reality (AR) and VR. Both AR and VR shape the way we see the world around us. But they do it differently [2]. The main difference between virtual and augmented reality is that virtual reality creates a whole new space in which user becomes involved, while augmented reality adds artificial objects to the real world [3, 4].

Software for creating VR is specific, with a large number of features [5]:

- Content management – Numerous tools allow upload either raw 3D content, which will later be changed into a VR code or existing VR content onto the device with the ability to manage and store data.
- Editing content – The great part of VR software has editing capabilities. Users can edit raw 3D or existing VR content. Some editing features have drag–and–drop capability, which implies that users can edit their content with relatively modest previous programming experience.
- Hardware – software integration – Any VR application must integrate with adequate devices that supports planned virtual experiences. These devices are usually headsets, but can also be something else.
- Cooperation – VR tools allow multiple users to access applications remotely so they can cooperate in real time. While cooperating, users are able to interact on the same things simultaneously.
- Analytics – Some VR applications enable analytical opportunities. It allows better understanding the behaviour of audiences accessing the VR content.

VR SOFTWARE CATEGORIES

With regard to the purpose, there are various categories of VR software:

(a) Content management systems (CMS) – A CMS is used to collect, store and analyse all VR content in a centralized location. As VR continues to evolve, these tools will become increasingly important looking to manage and organize all virtual content. Users may upload 360–degree videos and images directly onto these platforms and edit them there. These tools allow publishing VR content directly from the platform. Many of these solutions also offer reporting and analytics, so users may better understand the behaviour of the audiences accessing the content. This software gives users the possibility to create consistency among VR experiences, ensuring brand specificity or actual regulations.

Software in this category must have the following characteristics: allowing 360–degree images and video to be uploaded onto the system, offering drag–and–drop editing capabilities, managing all created content within the platform and publishing created VR experiences.

(b) Development software – VR is about creating a virtual and immersive environment to replace the real world. Development software is used to create native applications, typically for computers with Windows operating system. VR software development kits (SDK) provide the necessary environment to design, create and test VR experiences.

(c) Game engine software – A game engine provides developers with the framework for creating a VR video game experience [6]. A VR game engine often contains a virtual reality SDK. These tools enable developers to create and edit 3D characters and fully immersive experiences. VR game engines help developers focus on creating a product for the end user instead of wasting efforts on tying all elements of a gaming system together. VR game engines are very similar to general game engines, but are unique in that they support VR operating systems and hardware either directly or through an API (Application Programming Interface) [7]. Using VR game engines, developers can create games for various devices (for example, game consoles and smart phones). Some VR game engines can also create augmented reality experiences [8].

Software in this category must have the following characteristics: creation of custom VR video game experiences [9], creation and editing of immersive 3D experiences and integration with hardware that supports VR (mobile phones or headsets).

(d) Social platforms – These platforms allow cooperation in virtual reality from remote locations. They enable users to meet up in the same virtual space and communicate with each other (speech and text).

VR social platforms offer users the ability to choose and edit figures representing a particular person (avatars) as well as selected environments to host a virtual meet up. These meet ups can range from collaborating in a virtual conference room to viewing a presentation together. VR social platforms should allow users to develop virtual spaces to fit their specific needs [10], typically done so with a SDK. These solutions may also make easier recording and playback of virtual meetings.

Software in this category must have the following characteristics: allowing selection of a figure to represent them, enabling users to select and edit environments to host meet ups, providing the ability to communicate with other participants and allowing participation in a given activity together.

(e) Training simulator software – VR training simulators can be used in almost any area in an immersive virtual environment. These tools should not be confused with augmented reality training simulators, which provide training simulations through integrating digitally–created 3D images into the real world [11]. Giving users these experiences allows practicing and developing the skills that may be necessary in certain high–stress professions. However, the use of these tools can include industries, like aviation and transportation. Some of VR training simulators may also have SDK functionalities (developers can customize the simulator platform to fit their specific needs).

Software in this category must have the following characteristics: creation of specific virtual reality experiences, allowing upload of relevant content directly to headsets, supporting customized content (specific to business) and provide reporting on performance.

(f) Visualization software – VR data visualization software allows users to experience aggregated data. Data visualization enables to get analytics presented visually so they may fully understand what the data is communicating. VR data visualization is used across multiple industries, allowing working with real–time data. Also, it allows users to display large amounts of data. Bringing VR into data analytics enables simultaneous cooperation on work.

Software in this category must have the following characteristics: allowing multiple users to simultaneously work on data, enabling visualization in a completely virtual environment, data editing in real time and integration with devices.

VR SYSTEM SOFTWARE

VR system software is a collection of tools and software for designing, developing and maintaining virtual environments and appropriate databases where the information is stored. The tools can be generally classified into modelling and development tools.

Modelling tools – There are many modelling tools available for designing objects and environments. The most common ones are Unity, Google VR, 3ds Max, Maya etc.

Development tools – VR is a complex and integrative technology that borrows from many other technologies, such as real–time 3D graphics, tracking technology, sound processing and haptic technology, therefore software development flexibility and real time interaction is needed. Starting the development of a VR application from the basic codes (in C/C++/C#, Java, OpenGL etc.) requires a large amount of work and time. Such system reliability is usually low, therefore, development tools are used.

Detailed analysis is needed in choosing VR development tools due to the difference in flexibility of software packages as related to input model, interface compatibility, file format, animation characteristics, collision detection, supported I/O devices and support community.

Development tools used in VR content creation include virtual world authoring tools, VR toolkits / SDKs and APIs. But it is not uncommon to determine that some APIs are in fact toolkits (OpenGL Optimizer and Java 3D API).

EVALUATION OF VE (VIRTUAL ENVIRONMENT)

Evaluation of VR systems is mostly focused on testing its usability [12]. VR functionalities fall in two main groups:

a) navigation and exploration

b) interaction and manipulation of 3D virtual objects

It is very important to understand user needs and behaviours in order to adapt and improve VR products. To do so, systematic methods have been developed to evaluate and enhance usability issues:

Heuristic evaluation – includes methods and techniques of problem solving, learning and discovery based on experience.
Guerrilla testing – a fast and cheap method of capturing feedbacks that involves a user experience (UX) specialist asking questions about specific areas of application. What makes this testing unique is that participants are not engaged in advance.
In–person (test coordinator is physically in the same room as participants) and remote testing (via the internet)
VR analytics (numerical analytics that give data around particular actions – how many total views per scene, how many hotspots initiated, etc.) and heat maps (a visual representation of where users are looking)

In the context of a virtual, and in order to evaluate the realized environment, the following heuristics can be identified [13]:

Natural engagement – User – VE interaction should be as close as possible to the real world. The efficacy of this heuristic will depend on the need for naturalness and the sense of presence and engagement.
Compatibility with the task and domain – The behaviour of objects should correlate as closely as possible to the expectation of real objects.
Natural expression of action – The presence in the VE should allow the user to move and act in a natural way and not restrict usual physical actions. This characteristic may be restricted by the used device.
Close coordination of user action and representation – Response time between user action and update of the displayed situation should be less than 100–150 ms to avoid motion sickness problem (very individual value) [14].
Realistic feedback – The result of actions should be immediately visible and in accordance with the laws of physics and the user's expectations.
Realistic view of scenes – The visual look of the virtual scenes should match to the user's perception, and the viewpoint change (by head movement) should be rendered without delay.
Navigation and orientation possibilities – The users need to be able to know where they are currently and return to starting positions.
Clear entry and exit points – Entering and exiting ways (points) from a virtual world should be clearly explained.
Design compromises – In situations of design compromises, they should be consistent and clearly marked (e.g. power adjustments in navigation).
Learning abilities – Active objects should be marked and if necessary contain additional explanations to provide learning of VEs.
Clear turn–taking – Where system action occurs it should be clearly signalled and rules defined for turn–taking.
Sense of presence – The perception of engagement and moving in a virtual world should be as natural as possible [15].

GENERAL DEVELOPMENT PRINCIPLES

During the development phase, certain guidelines must be followed for the final product to be satisfactory and to pass the specific tests. These guidelines can be divided into five categories: compatibility, functionality, graphics and performance, security and user interaction.

Compatibility ensures that the application remains compatible with libraries and that the developer takes care of recommended hardware specifications.
Functionality ensures that the application works within actual standards. For example, single player games must be paused when the player takes off the head–mounted display (HMD) or returns to the menu. Also, the application must not draw frames or accept input. In addition, the player must not remain stuck in the application. For example, the login screen must have an option to create an account. If the application requires the Internet and it is not available, the user must be informed of that. The application must not lose recorded user data.
Graphics and performance are very important in VR applications. If the graphics is poor or the application is slow, it can be uncomfortable for the user and can feel nauseous. The application has to run at 60 fps (frames per second) constantly. Otherwise the user will feel uneasy. Also, the application must work without freezing and cracking. After starting, the scene must appear within several seconds, otherwise a loading screen should be provided. CPU and GPU (Central & Graphics Processing Unit) must be used most efficiently.
Security relates to the protection of privacy and the integrity of software and user data. The application

must require a minimum number of permissions to function. If some information is sent to an external service, the user must be informed about it.

User interaction must be in accordance with known standards. For example, Back and Home buttons must always do the same. If the application requires a controller, the user must be informed about that. If the controller is supported, the user must have the option to choose between controller and touch pad. Also, the application must detect whether the controller is on the left or right side.

DEVELOPMENT ENVIRONMENT

There are several ways to develop VR applications, so they can be divided into the following groups: native development, game development tools and web browsers.

Native development includes drivers and software libraries that are used in conjunction with an operating system, such as Win32 libraries for C++ applications and Java libraries for Android. VR devices have an SDK for platform–dependent development and interface to access device elements. There is a need to develop low–level elements, like rendering of graphics or physics within the environment. Native application development is the most flexible and optimized, but takes a lot of time especially if we want to cover a larger number of platforms. Because of this, most developers use completed development environments.

Game development tools serve as a middleware, which means they have implemented lower development elements. It is an integrated development environment (IDE), primarily for video game development, allowing creation of application using higher programming languages. Due to their characteristics, video game making tools have become the basis not only for video games but also for other types of 3D applications. The tool provides export to multiple platforms, has a built–in visualizer for 2D and 3D graphics, a drive to simulate basic physical laws, sound processing, embedded animation, artificial intelligence (AI), networking and more. The most famous commercial game engine tools for rapid application development are: Unity 3D (C#, Boo, UnityScript), Unreal Engine 4 (C++), CryEngine 3 (Lua), Blender (Windows, Mac OS, Linux), Unity 4 and 5 (C# Mono and JavaScript), Amazon Lumberyard (C++, Lua), CopperCube (JavaScript) and others. Most commercial tools can be downloaded for free for educational purposes or for the development of start–up projects. The companies behind specific tools give access to the source code to their users for a fee or for free. Regarding the open source solutions, there are: libGDX (Java), Torque 3D, Urho3D (C++), OGRE (Open Source 3D Graphics Engine) and others. The source code for these tools can be downloaded from the GitHub platform.

Web browsers involve the development of multi–platform applications using HTML5, WebGL and JavaScript technologies [16]. An example is developing applications using the WebGL (web graphics libraries) based JavaScript 3D libraries Three.js or Babylon.js. VR–enabled web browsers are Google Chrome VR and Mozilla Firefox Nightly. Google and Mozilla joined resources on developing a new standard for VR called WebVR, whose JavaScript API allows access to a web browser through a VR device. In addition, there is a domain–specific language and an A–Frame tool for WebGL–based applications. It was written by Mozilla and is open source project.

GAME DEVELOPMENT

Due to the popularity of games among younger population, game development gradually takes up more space in the VR software domain. In order to achieve the best player effect, several criteria must be met during game development.

(1) *Optimization* – There are various suggestions that can help optimize games. One of them is static batching. All objects in a scene that do not move can be marked as static and will then be plotted as a single object. It's even better if these objects use the same material, as each draw on the scene makes one draw call.

Good level design can significantly optimize the game. For example, if a player moves from room to room, it is not good to load all the rooms at the beginning, but to load them when the player enters. The downside is that more memory would then be used when loading each object. Therefore, it is best to separate the rooms by levels and load them asynchronously as needed from the code. Asynchronous level loading is very useful if the game has multiple levels. If the level is loaded classically (without asynchronous loading), the game will freeze until the next level is loaded.

Pre–rendering of light is very important not only in VR games. All static objects can be turned off real–time shadows and then pre–render the light. This creates a fixed texture with shadows that is rendered static (as an image) instead of "live" shadows that are drawn in each frame.

(2) *Moving* – Moving is a very important factor that determines player's satisfaction. If there is some moving in the game, it must be as close to natural as possible. Otherwise, player feels nausea (VR disease). It occurs because the body of the player is static in the real world, while moving in the virtual world. It is best to avoid moving, unless it imitates the true moving of the player, as it may affect the vestibular system (balance system).

In classic one–person games, nausea often occurs, so it's best to avoid them. However, if the developer chooses this type of game, it is best to test the moving with as many users as possible. One solution to FPS (First–Person Shooter) play is to make constant static visual references such as the cockpit, cabin or interior of a car.

A popular method of moving is the fade / blinking eye transition. The player moves from one position to another by darkening the camera quickly to give the user only dark in front of their eyes. Then the camera moves to a new position and the dark colour disappears. This is one of the pleasant and easiest ways to move players.

(3) *Player interaction* – Depending on the VR model, the game itself, and the player's choice, controls can be made from the HMD or from the controller. Both the device and the controller have a touch pad that can be clicked or dragged. The drag is detected in four directions up (forward), down (back), left and right.

Some games also have gaze control that works as follows: when a player looks at an object that is predefined for interaction, a circle appears and fills up for a second or two (depends on the developer) and then registers as if it were a player clicked. If the player takes his eyes off the object while filling the circle, the circle is removed and the timer is reset.

(4) *User interface and experience (UI/UX)* – When creating a user interface for VR games, there are approaches to consider that are not used in classic games. There are four types of UIs commonly used: non–diegetic, diegetic, spatial and meta.

For ordinary non–VR games, the interface is all over the screen and is called HUD (Heads Up Display). It's a non–diegetic interface, an interface that is not within the world but makes sense for the player as a spectator of the game [17]. These are usually player energy, health, score, etc. The term diegetic is taken from the film industry where non–diegetic sound is background music in a movie or TV program and a diegetic sound would be some conversation. An example of a non–diegetic interface can be seen in the figure below.

A spatial or spatial user interface would be an interface located within the game world itself. This may be the main menu or some auxiliary menu in the game. With this interface it is very important to pay attention to the distance from the user. If it is too close, the user will strain their eyes, and if it is too far it will seem as if the focus is on the horizon. It is best to position the interface a few meters away from the user, and scale the images and text to be visible and easy to read.

Figure 1. Non–diegetic interface (insert from Horizon Zero Dawn)

Figure 2. Spatial interface (insert from Splinter Cell: Blacklist)

An alternative to a spatial interface is to place the interface elements in the environment itself – diegetic interface [18]. It can be a real clock on the wall, a TV, a computer, a mobile phone or even a screen on a futuristic rifle. An example of a diegetic user interface can be seen in the following figure – elements that follow an object in the scene. A good example is the energy of the enemy on stage.

Figure 3. Diegetic interface (insert from Steel Battalion)

Meta interfaces are similar to non–diegetic interfaces. Meta interface is the effect of displaying a scene that is not part of the VR world (e.g. change of colours if the user loses energy). A good example is Call of Duty

– there is no health bar, but when the player is low on health, the screen would be overlay with blood to show that he is wounded. Meta UIs are usually represented two dimensionally.
All these types of UIs can also be combined in same game.

USABILITY TESTING OF VR APPLICATIONS

Usability is a quality attribute that indicates the ease of use of a certain application. Usability tests consist of short sessions where participants interact with the application on a specific environment, while an expert team observes, records and measures the course of that interaction. In VR domain, the essence of this is session–based exploratory testing, i.e. exploring through the application in order to find and achieve what was planned, in a fixed time period [19].
Usability tests should be performed at the right time – when the prototypes and the application are finished. It is highly recommended that the testers perform a heuristic test before the users are given the devices. The reason is that there are many issues that can be detected in this phase and fixed before the usability test [20].
For testing purposes, two factors can be identified:

- Participants – A reasonable number of test participants is at least five. This enables to get a collection of different opinions. The profile of the user has to match the target audience of the application. It is important to know the skills, VR experience and the interests of the potential participants. Different levels of expertise can be required for making the session more interesting and effective.
- Technical assumptions – It is necessary to have technical equipments like headsets and adequate devices to generate virtual reality of satisfactory quality.

Minimal hardware assumptions:

Accessories – controllers, earphones
Computer (recommended requirements): Intel i5–4590 (equivalent or greater), NVIDIA GTX 970 or AMD R9 2900 (or greater), 8 GB RAM (or more), HDMI 1.3 and 3x USB 3.0 plus 1x USB 2.0, Windows 7 64–bit (or greater)
Headset (supported by computer) Oculus Rift, HTC Vive (PC/Mac), console (PlayStation VR (PS4)) or smartphone (Google (Android/iOS), Samsung Gear VR)

As for the application, there are some general aspects to pay attention when testing (test criteria):

User interaction with the VR interface – Moving across different spaces inside the application.
Execution of flows – Test the application for what is not supposed to do (analysis of border cases in order to maximize coverage), ensure every requirement is covered.
Functionality works expectedly – Isolation from the real world, real world does not interrupt the experience, immersion is achieved completely (360–degree view, user feels sensations).

CONCLUSIONS

In the future, it can be expected that VR will continue to evolve primarily because of the great interest of companies and the user community. Virtual and augmented reality devices are popular among VR developers. Big companies are currently controlling the VR technology market. Applications are most accessible on desktop and mobile devices. The WebVR and UI/UX standards are being actively developed. New interdisciplinary fields have been created that link VR with education, medicine, film, construction, sales and the automotive industry. VR applications can be developed using popular programming languages and tools that do not require significant financial investment from the users. The ability to develop different types of VR applications can lead to major changes in the implementation of information technology. Applying best practices and standards can lead to a quality VR experience.
VR games are very popular form of software. In the course of their development, in addition to the general ones, more criteria need to be met in order to achieve fully immersive experiences. For this it is also necessary to fulfil demanding hardware and software assumptions, as well as specific testing methodology.

Literature

[1] T. Parisi: Learning Virtual Reality, O'Reilly Media, 2015
[2] P. Tiefenbacher, N.H. Lehment, G. Rigoll: Augmented Reality Evaluation: A Concept Utilizing Virtual Reality, International Conference on Virtual, Augmented and Mixed Reality VAMR 2014, pp. 226–236.
[3] A. Amer, P. Peralez: Affordable Altered Perspectives Making Augmented and Virtual Reality Technology Accessible, Global Humanitarian Technology Conference (GHTC) IEEE, 2014
[4] R. Azuma: A survey of augmented reality, Presence 6(4), 1997, pp. 355–385.
[5] T. Mazuryk, M. Gervautz: Virtual reality history, applications, technology and future, Technical Report TR–186–2–96–06, 1996
[6] B. Cowan, B. Kapralos: An Overview of Serious Game Engines and Frameworks, Recent Advances in Technologies for Inclusive Well–Being, Volume 119, Series Intelligent Systems Reference Library, February 2017, pp. 15–38.
[7] C. Cruz–Neira, M. Fernandez, C. Portales: Virtual Reality and Games, Multimodal Technologies and Interaction, 2018
[8] M. Lanham: Augmented Reality Game Development, Packt Publishing, 2017
[9] P. Bouvier, F. De Sorbier, P. Chaudeyrac, V. Biri: Cross–benefits between virtual reality and games,

Computer Games Multimedia and Allied Technology Conference, 2008
[10] K.M. Stanney, R.R. Mourant, R.S. Kennedy: Human Factors Issues in Virtual Environments: A Review of the Literature, Presence,Vol. 7, No. 4, August 1998, pp. 327–351.
[11] E. Ragan, C. Wilkes, D.A. Bowman, T. Hollerer: Simulation of augmented reality systems in purely virtual environments, In: Proc. VR, IEEE, 2009, pp. 287–288.
[12] T. Marsh: Evaluation of virtual reality systems for usability, Proceeding CHI '99, Extended Abstracts on Human Factors in Computing Systems, pp. 61–62.
[13] A. Sutcliffe, B. Gault: Heuristic evaluation of virtual reality applications, Interacting with Computers 16, 2004, Elsevier, pp. 831–849.
[14] K. Norman: Evaluation of Virtual Reality Games: Simulator Sickness and Human Factors, GHItaly@AVI, 2018
[15] A. Sutcliffe, K.D. Kaur: Evaluating the usability of virtual reality user interfaces, Behaviour and Information Technology, 19(6), November 2000, pp. 415 – 426.
[16] T. Pant, S. Neelakantam: Learning Web–based Virtual Reality: Build and Deploy Web–based Virtual Reality Technology, Apress, 2017
[17] I. Iacovides, A.L. Cox, R. Kennedy, P. Cairns: Removing the HUD: The impact of non–diegetic game elements and expertise on player involvement, Conference: 2015 Annual Symposium on Computer–Human Interaction in Play (CHI PLAY '15), 2015
[18] P. Salomoni, C. Prandi, M. Roccetti, L. Casanova, L. Marchetti, G. Marfia: Diegetic user interfaces for virtual environments with HMDs: a user experience study with Oculus Rift, Journal on Multimodal User Interfaces, January 2017, pp. 1 – 12.
[19] BlazeMeter, https://www.blazemeter.com/blog/user–testing–of–virtual–reality–applications/
[20] P. Dias, A. Pimentel, C. Ferreira, F. van Huussen, J.–W. Baggerman, P. van der Horst, J. Madeira, R. Bidarra, B.S. Santos: Usability in virtual and augmented environments: A qualitative and quantitative study, Proceedings of SPIE – The International Society for Optical Engineering, 2007

ACTA TECHNICA CORVINIENSIS – Bulletin of Engineering
ISSN: 2067-3809