ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 3201
WSCG 2022 Proceedings

Feature based CAVE software factory

Jacek Lebiedź
Faculty of ETI
Gdańsk University of Technology
ul. Gabriela Narutowicza 11/12
80-233 Gdańsk, Poland
jacekl@eti.pg.edu.pl

Bogdan Wiszniewski
Faculty of ETI
Gdańsk University of Technology
ul. Gabriela Narutowicza 11/12
80-233 Gdańsk, Poland
bowisz@eti.pg.edu.pl

ABSTRACT

In the paper we convey the lessons learned along the path we have gone through several years since establishing a room-sized CAVE installation at our university, from craft manufacturing and ad-hoc software reuse of VR software products to the robust feature driven software product line (SPL) implementing the Product Line Engineering (PLE) factory paradigm. With that we can serve all our departments and other entities from the region by rapidly instantiating different VR products based on a standard set of core assets and driven by a set of common features of VR applications destined to be deployed in the same target CAVE system – with the minimal budget and time to market requirements. A comprehensive survey of the most representative CAVE applications created in Gdansk Tech Immersive 3D Visualization Lab (I3DVL) according to PLE paradigm presented in the paper provides evidence supporting this claim.

Keywords

CAVE product portfolio, VR feature tree, Software Product Line

1 INTRODUCTION

Since its premiere in 1992, the stereoscopic video theater known since then for its recursive acronym CAVE (Cave Automatic Virtual Environment) has found many followers at universities and engineering companies reapplying the concept in a variety of fields.

Costs of a CAVE can reach a few million dollar level, especially when rear-projection is used and a significant amount of additional space for the equipment is required. All this severely limits the deployment of such installations in everyday workspaces and usually goes beyond the financial resources of typical educational institutions and small or moderate size businesses. For this reason, the total number of room-sized CAVEs exploited in the world today is low and serving a rather narrow niche market. In consequence, development of software applications dedicated for the particular CAVE would involve a different set of schematics, not necessarily making the overall software process cost-effective and of sufficient quality to justify the high investment and maintenance costs of the installation.

The motivation for writing this paper was to summarize our experience in manufacturing custom-made CAVE applications for research, education and training, commissioned by our university departments or SMEs from the region. Since 2014, we have been systematically expanding our infrastructure to its present form, including three CAVEs of different sizes integrated with large-size haptic devices, a theatre for 100+ viewers and a supercomputer, as characterized in Section 2.

Although I3DVL is basically a non-profit entity operating on a very specific market of recipients of CAVE products, all its related activities must meet the basic criteria of the commercial software process, including product assurance, cost effectiveness, time to market, productivity, quality and agility. In the paper we describe the path we have gone through all these years, from craft manufacturing and ad-hoc software reuse to the robust feature driven *Software Product Line* (SPL) [Nor12a] and convey the lessons learned in the process. Today, by reusing core assets described in Section 3 of a common set we can develop new CAVE applications based on a proven concept involving optional and variable features within the tight schedule, staffing and budgetary constraints. Their overview is given in Section 4. The existing core assets that evolve out of product development into new assets provide in turn a strong feedback loop for further development of the former keeping the product line up-to-date and reactive to the evolution of the underlying feature model.

2 THE CAVE INFRASTRUCTURE

Below we present briefly basic technical characteristics of the principal components of the infrastructure of our CAVEs and classify immersion levels its users may experience. We will refer to this classification further in the paper, when presenting the VR software factory paradigm implemented on them.

Graphics workstation onlooker experience

Four workstations, each one equipped with a graphics card driving a 27" stereo 3D display of the 2560×1440 resolution, can provide viewers wearing shutter glasses with the sense of depth when viewing images of the dynamic scene generated in real time. No immersion can be sensed for the lack of tracking devices.

The miniCAVE low-relief experience

A miniature open CAVE consisting of four displays as above and arranged as the floor and three surrounding walls allows for a miniature scale immersion of the viewer's face only, as shown in Fig. 1. Displays are driven by separate computers interconnected with the 1 Gb/s Ethernet or 40 Gb/s InfiniBand. The audio system consists of four speakers arranged above the wall displays and a subwoofer below the floor display. Shutter glasses of the viewer are equipped with the IR smart tracking system, whereas navigation is enabled by the flystick. One computer is a master for the other three slaves. The master reads data from the flystick and the tracking glasses, generates images for the central screen and broadcasts the relevant data to the slaves. Their task is to generate images synchronized with a central view.

Figure 1: MiniCAVE for low-relief experience

The midiCAVE high-relief experience

A full scale immersion of the viewer in the straight-up (standing) position is provided by an open CAVE consisting of the square 2.12m×2.12m front screen wall, two side 2.12m×1.34m screen walls and the floor screen of the same size, as shown in Fig. 2. The stereoscopic image is displayed on each wall by a pair of projectors (1920×1200 for each eye) in passive technology with spectrum selection. Rear projection is used for the vertical screens, and front projection for the floor screen. Rectangular screens are operated by individual pairs of projectors, while the square front screen by two such pairs. Each pair of projectors is controlled by a single computer, interconnected alternatively with the 1 Gb/s Ethernet or 40 Gb/s InfiniBand. The audio system consists of four speakers in four upper corners of vertical screens and a subwoofer outside of the screens. The viewer is tracked in midiCAVE by eight IR cameras placed in all its corners and is enriched with a full body motion tracking system. For navigation a flystick is used.

Figure 2: MidiCAVE for high-relief experience

The bigCAVE monument experience

Complete immersion is provided by the bigCAVE's six square 3.4m×3.4m walls surrounding the person inside the cube, creating impression of an unlimited space, beyond the reach of his/her arms, as shown in Fig. 3. Stereoscopic images are displayed on each screen by two 1920×1200 projectors in a rear projection mode, which gives after using the blending technique the 1920×1920 resolution of one screen in total. Stereoscopy is obtained by two alternative technologies: active Nvidia 3D Vision Pro or passive with spectrum selection. Each projector is driven by a separate server providing a rate of 60 fps for each eye. This performance level is well above the minimum requirements for a single viewer controlling the dynamic scene view reacting to his/her head movements. A fully successful attempt was therefore made to duplicate the number of images displayed on each of the six bigCAVE walls to allow two observers to control the scene view simultaneously without changing the number of projectors [Leb21a]. As a result, the 60 fps rate for each eye of each observer was reduced to 30 fps to allow for generating four frames instead of two; for most of the implemented application scenarios outlined in Section 4, these values turned out to be perfectly sufficient. Twelve computers plus two more for synchronizing the former, tracking the viewer inside and generating full dimensional sound constitute a server farm of 14 units, interconnected with 1Gb/s Ethernet and 40Gb/s InfiniBand. The audio system consists of eight speakers, two

of them in each upper corner of the bigCAVE's cube and a subwoofer outside of it. The viewer is tracked by four IR cameras placed in the upper corners of the cube. One of the servers in the farm can also generate a stereo image perceived by the user inside to be transmitted (if needed) to the projector in the external large (100+ seats) 3D theatre.

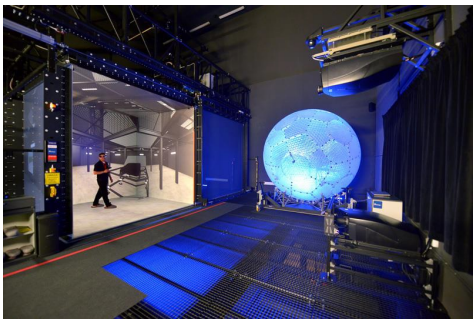

Figure 3: BigCAVE and the VirtuSphere

Large scale haptic devices

In addition to immersion perceived by the sense of sight and hearing, two haptic devices are currently being introduced in I3DVL to broaden the experience of immersion by affecting the participant's sense of balance:

- *CyberSphere* with force-feedback, founded on eight drive and measurement (DME) rolls with linear actuators driving the actual VirtuSphere with the user placed inside. Whereas the VirtuSphere is mounted on eight passive rollers evenly spaced on a round base and driven only by the body movements of user walking inside it, the actuators can measure angular velocity and adjust downforce (pressing the roller against the sphere) to provide additional feedback from the generated virtual worlds [Kow18a]. The sphere in either configuration fit in the bigCAVE cube and can be used to further extend the monument-like immersion perceived by the user, as shown in Fig. 4.

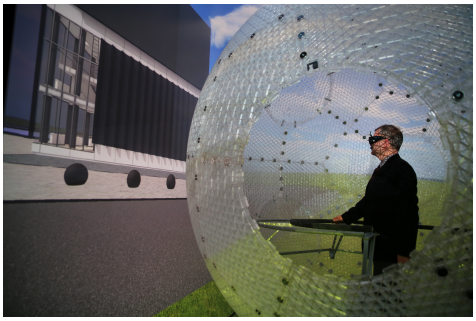

Figure 4: Virtu/CyberSphere in the bigCAVE

- *Stewart (4 DoF) platform* enabling simultaneous simulation of yaw, pitch, roll and lift with four actuators setting the platform in a sliding and rotating motion. One or more persons standing on its top can feel realistic movements of a deck of a simulated vessel (open sea experience) or the undercarriage of a land vehicle (off-road experience) [Laz94a].

HPC support

A cluster-based supercomputer composed of over 1600 computing server nodes provides nearly 1.5 PFLOPS of computational power of jointly 38500 cores and 48 GPUs. All nodes of the cluster are connected with InfiniBand FDR 56 Gb/s. Three 40Gb/s links between the supercomputer and the farm of computers driving our bigCAVE allow for sending data from it for processing as well as sending back to it voluminous data streams in real time. So far the HPC support was experimentally used by us for simulating complex weather phenomena (precipitation) and internal fires (flashover and backdraft). However, it is still the experimental facility planned for our future products.

3 THE CAVE PRODUCT PORTFOLIO

The requirements baseline for CAVE applications is typically focused on three basic attributes:

- *immersion* – high fidelity of images in terms of resolution as well as the sense of depth combined with the spatial sound should give viewers the impression of the fullest possible immersion in the generated dynamic scene of the virtual world;
- *interaction* – the generated dynamic scene views must exhibit an appropriate level of realism when reacting to the movement of viewers navigating in the related 3D space and respond promptly to any changes in their position;
- *performance* – calculation of the outcomes of viewers' actions affecting the dynamic scene content must be performed by the computing system in real-time, adequately to the dynamics of viewers.

Given the above a multitude of similar but separate VR software products could be considered – with the variety of optional features to be implemented to satisfy the end user needs. With the traditional approach to handling that developers would reuse pieces of code (small-grained reuse) [Mor02a] or larger parts of another product to build a new one (clone and own approach) [Gha18a]. However, it may be considered ineffective for CAVE products, as two applications built from the same base would be deployed and maintained in the same CAVE separately, with high duplication of effort to fix possible errors in multiple products multiple times and often implementing independently the same enhancement in different ways in similar products. If no systematic reuse of artifacts from previously implemented products is considered, or worse, no

such artifacts have been collected yet, one extreme alternative would be craft manufacturing [McB02a]. This cut-and-try approach involving writing and modifying the application code by skilled programmers to make it fit in the CAVE system would dangerously reduce the latter to the role of an expensive toy for a small group of enthusiasts – unacceptable alternative to entities obliged to strictly follow the rules of financial discipline, such as our university.

When setting up the infrastructure described in Section 2 we took a closer look at common features that constitute requirements baseline for our CAVE products and the assets that constitute the core of its technical specification. Based on that our SPL adopts the underlying *Product Line Engineering* (PLE) factory paradigm and allows for optimizing effort, time and quality in product assurance of VR applications intended for use in our CAVEs. Further below we present this approach, which may provide a template solution to other universities struggling with the problem of maintaining such an expensive installation as room-sized CAVEs and its cost-effective use in everyday educational activities.

Requirements baseline – features, options

The basic tool for describing the desired product line functionality is the feature model [Fer02a], which can define the relationship of features in a family of products. It may be represented by a tree, which each vertex corresponds to one of many possible features, whereas different edges represent specific relationships of these features. In Fig. 5 we specify a tree of features of CAVE applications which can be derived from the three basic attributes listed before. There are four types of relationships between its nodes:

- *mandatory features* define the minimum functionality of a target CAVE application, i.e. each vertex, whose edge from the parent is marked with the $\oplus$ symbol must be selected. It may be seen that such an application should be able to provide simultaneously *immersion fidelity*, *interaction realism* and *real-time performance*, in terms of the dynamic scene *graphics* as well as *simulation* of its relevant phenomena. Moreover, *graphics* performance should concern both *static* and *dynamic* objects.

- *optional features* represent at least one option to be selected. It is assumed that from among all sibling vertices of a given parent node with the respective edges marked with the $\circledcirc$ symbol, the leftmost one must be selected first, and next the other ones in the order of diminishing preference from left to right.

- *alternative features* concern exclusive choice from a set of equally preferred options. In other words, from among all sibling vertices of a given parent node with the respective edges marked with the $\otimes$ symbol, exactly one vertex must be selected.

- *prospective features* are planned for future applications, as new hardware and software assets will be available. Their respective edges are marked with the $\circ\!\!\!\sim$ symbol. In the future they will be merged with the $\oplus$ symbols marking the respective edges of their sibling vertices into the $\circledcirc$ symbol, since their vertices will introduce new optional features.

Each our CAVE application allows viewers to experience immersion by attracting their senses with the spatial audio system *and* stereoscopic video solutions suitable for either a single viewer *or* multiple viewers. They enable simple navigation with a flystick (or alternatively with the inserted VirtuSphere) *and* tracking of a viewer's head based on IR with the additional *option* for tracing other body movements involving body markers. Real-time performance of the graphics is provided for *both* static objects (handling dynamic changes of each static object appearance due to the observer's head movements) and dynamic objects (moving independently across the wall screens). Simulation of the latter is supported by one computer of the CAVE farm with the *option* of additional support provided by the locally available HPC platform. Interesting extensions planned by us for future CAVE applications indicated in Fig. 5 concern the aspects of immersion which are not directly related to sensual impressions. Several applications in our product line already concentrate on *emotions* evoked by the content and meaning of the dynamic scene rather than its appearance only. Based on the research in affective computing and automatic monitoring of human emotions [Lan16a] we plan to introduce the *feedback* feature, to make applications more responsive and further deepen the immersion experience of viewers. Various combinations of the features in Fig. 5 specify the requirements baseline for our CAVE product portfolio and drives development of their respective code and technical specification. The latter constitutes a set of core assets that capture the underlying feature model and each new product is supposed to be built from that set in a prescribed way.

Technical specification – core assets

As mentioned before the primary capability of the software product line is instantiation of a final product derived from the common set of shared assets rather than crafting its code manually from scratch or reusing components of other products. The term "development" in this context means that not only the product code is being built, but many other assets that further support the overall engineering process toward the final product are also instantiated. They constitute a common platform

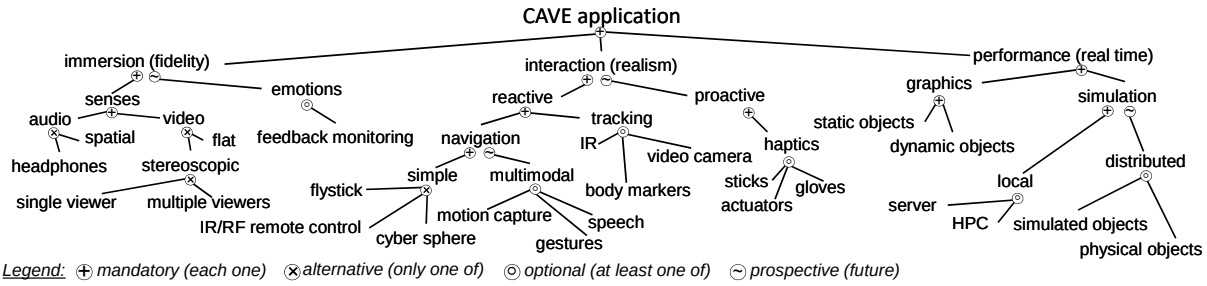

Figure 5: Feature model of CAVE applications

upon which many different products may be developed. The core assets that are shared across our SPL implementation for CAVE applications include the following categories [Nor12a]:

Requirements; incremental refinements of the basic feature model addressing technical details of its individual features (parameters and their variability, characteristics and attributes appropriate for the given software product instance, its target audience).

Architecture; a basic scheme for assembling the product from software components in the core asset base instantiated for each individual product.

Software components; basic building blocks for each individual product, reused without alteration or altered using inheritance or parametrized templates.

Performance models; hard and soft real time constraints, qualitative characteristics and methods for determining them, including image rendering, synchronization and switching.

Cost/schedule; generic framework for work schedule development and time and effort estimates for the entire product line.

Tools/processes; support for software product development and making changes, appropriate for the entire product line and production stages.

Test cases/data; generic testing artifacts suitable for all products in the product line and extensible to accommodate variation among them.

People/skills; evolving from programming toward relevant domain expertise and technology forecasting.

The key assets in the architecture and tools and processes categories of our SPL are the CAVE infrastructure specified in Section 2 and any modern game engine (GE) satisfying the criteria defined in [Pet12a]. Each such GE provides an IDE and resources enabling rapid development of high-fidelity virtual worlds by handling in the uniform way complex computations related to rendering of the dynamic scene, simulating physics of dynamic objects, detecting collisions, etc. Thanks to their modular structure, they can be repeatedly reused to instantiate a variety of VR products, very similar to what SPL provides. Currently our principal GE asset is Unity and Unreal Engine [Pet12a]. The initial content of the assets listed above are the following:

Requirements; Features selected from the baseline set (limited to the single 3D flat-screen display view, keyboard and mouse). Additional target domain-related requirements (external asset) added.

Architecture; Reference architecture selected (Unity or Unreal Engine). Development framework set up (non-stereoscopic 3D graphics display, keyboard and mouse).

Software components; The candidate application code and documentation (external asset) added.

Performance models; Scene rendering complexity (graphics performance, navigation reactivity) and dynamics of simulation objects assessed.

Cost/schedule; Additional candidate application cost and schedule (external asset) added.

Tools/processes; Toolset specific to the selected platform acquired (Unity, Unreal Engine or other).

Test cases/data; Preliminary acceptance criteria defined. The candidate application formally qualified as the product line asset (regular graphics workstation, single flat-screen display, keyboard and mouse).

People/skills; Candidate application domain specific skills (external asset) added.

Note that the core set is extended by external assets brought to the project by a candidate application that will be finally instantiated and deployed in the bigCAVE system as the stereoscopic 3D one. Besides this application's code and documentation other external assets include such domain-specific artifacts as the additional target domain requirements, specific performance characteristics and staff skills. The baseline requirements at this stage are defined by the feature model

limited to the standard 2D user interface (display, keyboard and mouse). In the following production stages they are gradually refined, respective to the capabilities of the target CAVE system. The internal technical staff that can be assigned to any production activity in I3DVL varies from one to three persons, one more team leader person and a couple of external target domain experts. The internal staff roles and responsibilities require the skills of a graphics designer, programmer, tester and instructor – one person may simultaneously play several roles at various stages of the production process.

Production stages and product decisions

Although PLE can capitalize on commonality of final product instances destined for the same target system, the preliminary set of core assets listed before may exhibit variation due to inherent variability of the underlying feature model specified in Fig. 5. Note that depending on the particular application purpose and behavior individual products may be instantiated in different ways. Managing this variation in the production process requires a clear definition of internal variation points where appropriate product decisions are made. The point in the production timeline at which the decisions for a variation point are bound is referred to as the *binding time* [Ros11a]. Our SPL utilizes multiple binding times determining four production stages outlined in Fig. 6.

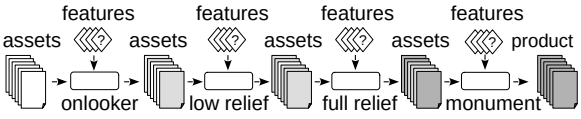

Figure 6: CAVE application SPL production stages

The respective production stages are marked with rounded rectangles and their relevant binding decisions are marked with diamonds. By starting with the preliminary set of assets listed before the product outputs from binding decisions at one production stage become partially instantiated asset inputs for binding decisions at the next production stage. The progress of instantiation of assets toward the final product is symbolically marked by darkening gray.

Workstations of the *onlooker* stage mimic configuration of the computers controlling projection on a single wall of bigCAVE. Owing to that developers can assess the quality of the depth and the overall design and logic of the content of stereoscopic images that the prospective application is supposed to deliver to each screen at its target installation. *Low-relief* stage utilizes the miniCAVE system for testing synchronization of images in side and floor screens with the central (front screen) view in the master-slave fashion, in particular simultaneous changes of views on all displays, joining adjacent screen views and verifying correctness of virtual cameras definitions on the master and slaves. *High-relief* stage utilizes midiCAVE to test synchronization of images in the application under development and verifying correctness of all virtual cameras definitions and assessment of blending of images on the central (front) screen generated by two slaves. Moreover, scenarios involving body motion capture may be implemented and multiple viewers may be involved if needed. *Monument* stage is the final one, as the bigCAVE system is the target environment for our VR applications. Blending of images on each screen wall is tested to qualify the application to the final acceptance testing of image synchronization.

For brevity we skip detailed description of the evolution of each core asset of the set into the final one, but remarks on some of them are in place (see Fig. 6): the application *architecture* asset evolves from a standard flat screen through a single stereoscopic 3D form up to the master-slave form of multiple screens, the *software components* asset expands from the initial application code to the form augmented with synchronization and interaction objects from our own CAVE library developed for that purpose, and the *performance models* asset expands from the basic quality (graphics performance, navigation reactivity and dynamics of simulation objects) characteristics to multiple screens synchronization quality characteristics, such as geometric continuity, 3D image stitching and consistency of adjacent images. Besides a GE platform for going from flat screen display to the 3D stereoscopy we use popular tools for unwrapping mono or stereo audio sources to surround formats [Tho07a].

The reason why our primary preliminary software component asset is mostly a regular non-stereoscopic 3D graphics flat-screen application with the mono or stereo sound is quite simple: most of the interesting domain applications are created in various departments of our university on ordinary flat-screen workstations and not necessarily with the intention of their subsequent implementation in CAVEs. The current ease of developing them thanks to the widely available and popular GE platforms (Unity in particular) means that several dozen of them are created at our university every year. They are systemically delivered by students as part of class projects or built by our faculty as 'proof of concept' in various R&D projects. Some the best of them are deployed over time in the university's didactics and become qualified candidates to convert them to the fully immersive, interactive and high-performance CAVE applications. There are also individual cases of building dedicated applications for research in the field of computer graphics, such as ray tracing in heterogeneous environments for example [Kacz21a]. I3DVL also receives ocassionally a small number of candidate applications developed by the third parties (mostly co-

operating SMEs) with a request to deploy them in the bigCAVE.

Such a clear separation of the two phases of the bigCAVE application life cycle promotes agility and significantly reduces costs: for the initial flat-screen version phase the primary focus is on the semantic correctness of the scene content, its logic and realism, whereas for the phase described in the paper the proper development of the CAVE application takes place.

Assets of a bigCAVE product instantiated in the monument stage finally take the following form:

Requirements; synchronization of six screens and flystick functionality.

Architecture; multiplication of a single screen application to the master-slave (front vs. side, floor and ceiling screens) configuration.

Software components; master-slave synchronization and interaction objects (the own CAVE library).

Performance models; navigation/synchronization quality characteristics (geometric continuity, 3D image stitching, consistency of graphical effects in adjacent images).

Cost/schedule; fixed cost model (master-slave parallelization of six screens, two person-days, linking the single screen application to the bigCAVE interface one person-day).

Tools/processes; configuration and deployment of master-slave synchronization objects and interaction objects from the CAVE library with .tools from the relevant development framework.

Test cases/data; acceptance testing of the final application wrt. the feature model baseline requirements (immersion, interactivity, real-time performance).

People/skills; one person (tester and programmer).

4 CAVE APPLICATIONS OVERVIEW

Each CAVE application in our SPL has been instantiated in the production stages outlined in Fig. 6 out of some initial flat-screen application and the preliminary set of core assets mentioned before. For the sake of brevity, we will not place a complete bibliographic record for each of the applications discussed below, and only highlight their basic effort and time characteristics. Their purpose varied from teaching or training, through prototyping (devices, buildings, structures) or treatment of patients, up to pure or applied research.

For example, the Old Town application (Fig. 7) was aimed to synthesize visually new planned buildings in the existing architecture under variable day/night lighting and weather conditions (prototyping), but also involved teaching (objects can be designed by students). Viewers interact with the dynamic scene by selecting time of the day and weather conditions and navigate with a flystick along a predefined path or teleport to selected places [Leb16b]. Total instantiation time of this application was three months and involved two programmers or testers, supported by three teams of architects (nine persons in total).

Figure 7: Old Town interactive visualization (drizzle)

The purpose of the interactive small UAV simulation application (Fig. 8) was also prototyping – a virtual drone may be reconfigured and reequipped with various simulated on-board devices like cameras or sensors. It can be controlled with a specialized controller in three different views: from the ground, from the cockpit, and from behind the drone's tail. Additionally, to enable monitoring of the UAVs in-flight state a standard instrument panel may be displayed on demand on the CAVE's wall currently pointed by the controller. Total development time of this application was three months and involved just one programmer/tester, supported by a single domain expert (pilot).

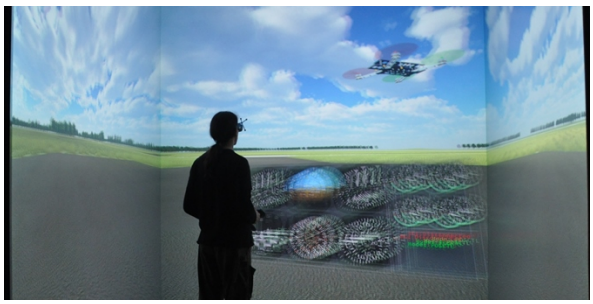

Figure 8: A small drone platform tester and trainer

These two applications are similar in the aspect of simulating weather phenomena, but totally different with regard to the dynamics of the simulation objects involved. Nevertheless, they both were instantiated out of the same set of core assets and, as indicated before, within the similar staffing and time-to-market limits.

Another applications in our SPL involving simulation of basically the same physical phenomena (gas flows), are the small fire simulator (Fig. 9) and a sail-boat in

the virtual wind-tunnel (Fig. 10). The former may be used to train people how to operate a jet from a hand-held fire extinguisher to quickly suppress a small fire in an office room, whereas the latter to make them understand settings of the sail for different courses of the boat in relation to the wind direction; individual bands and their colors show the direction and speed of the airflow [Leb16a]. Owing to the more complex simulation of the hot (flame) gases the instantiation time of the first application was four months, whereas for the latter it was just one month. In both cases only one programmer/tester was needed, supported by the relevant domain expert (respectively a fireman and a sailor).

Figure 9: A small fire and extinguisher simulator

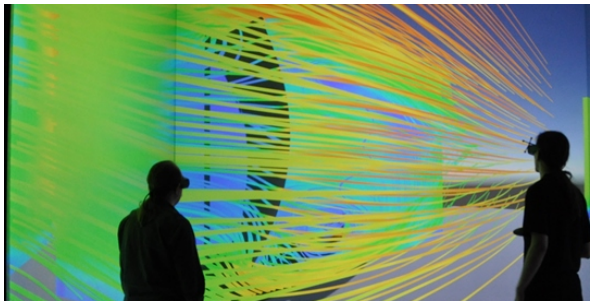

Figure 10: Simulation of an airflow around the sailboat

Another segment of our products are CAVE applications developed by our students in cooperation with the Polish Space Agency. The average instantiation time of each application in this segment was three months, with one programmer/tester and one domain expert (astronomer) involved. These applications address various issues related to understanding the Solar System [Mar22a], exploration of celestial bodies, detection of black holes with gravitational lensing, and visualization of constellations from various places on Earth, among others. They all take advantage of the possibility to directly immerse viewers in the visualized phenomena to help them understand its essence and course and are intended for primary and secondary school students. These applications use our bigCAVE with a few individuals inside, and a larger audience in the connected video theater outside. For example the application shown in Fig. 11 helps viewers to understand distances between planets of the Solar System, types of its celestial bodies (the Sun, planets, asteroids) and mechanics of their orbits. Users can navigate using teleportation and observe them from various places of the Solar System.

Figure 11: Interplanetary travel in the Solar System

Another application of the space segment in our portfolio is the Moon/Mars rover simulator (Fig. 12). The rovers are simulated according to the gravity of the respective celestial body and real hypsometric data of the explored terrain (Apollo LEM and Curiosity rovers).

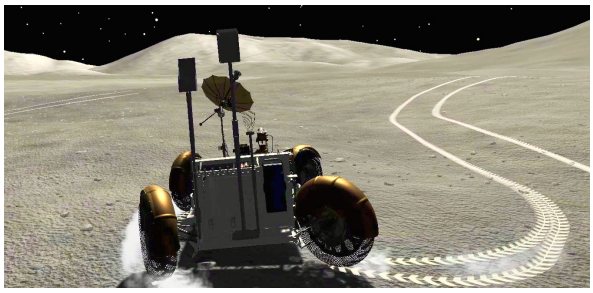

Figure 12: LEM lunar rover simulator

An important segment of our SPL are biomedical applications developed for research into therapies of various kinds based on immersion. One example is shown in Fig. 13; this application is intended to treat acrophobia (height anxiety) with the implosive therapy augmented with gamification. The task of the patient is to move from a safe room to various levels of a tall building and navigate along narrow gangways, suspended high between various buildings and collecting coins of different value along the way. A psychologist supervising the exercise can give advice to the patient inside, comment on decisions of the latter and encourage him/her to choose specific optional segments of the path. The instantiation time of this application was three months and involved three programmers/testers, including development of its initial version, as the preliminary software component asset. The supporting team of domain experts included five psychologists.

A biomedical application shown in Fig. 14 was developed for research into the diagnosis of dizziness. A patient is immersed in an unstable space of a tunnel built of rotating rings and with accompanying unpleasant sounds. Reactions of the patient can be measured in

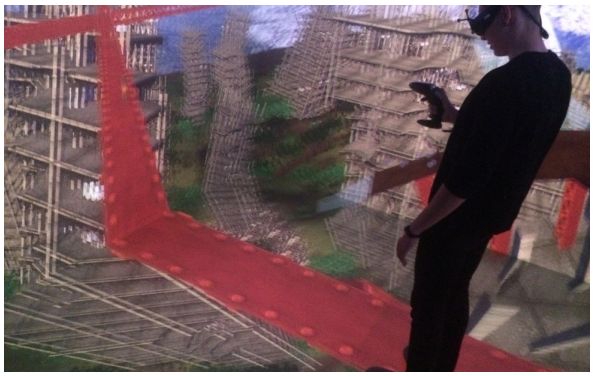

Figure 13: Acrophobia treatment game

a couple of ways: by recording the pressure exerted by each patient's foot on the ground, or recording electrical activity of the brain with a portable EEG, or both. The instantiation time of this product was three months, involved one programmer/tester and one domain expert (a biomedical engineer).

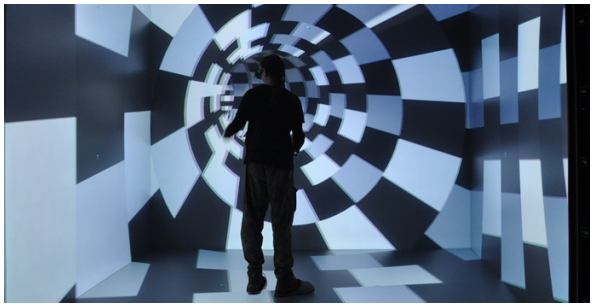

Figure 14: Stimulation for dizziness measurements

An interesting use of the bigCAVE in I3DVL is a virtual escape room in which puzzles play an educational role. The player's task is to guess the codes that open the door to leave the virtual room shown in Fig. 15. The codes to open individual locks can be obtained by solving chemical puzzles, assembling molecule models or performing virtual chemical reactions. The application was prepared by a team of three in two months, supported by a specialist chemist.

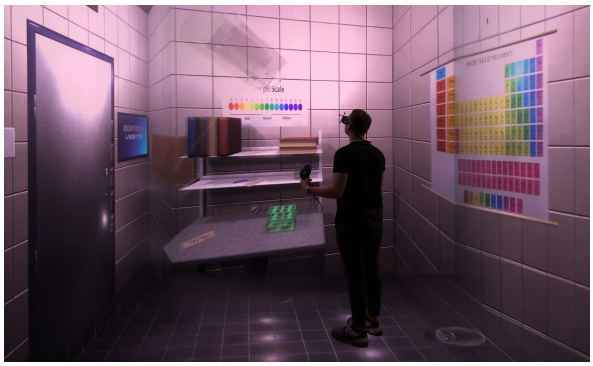

Figure 15: Chemical escape room

5 CONCLUSION

The survey of just a small fraction of our SPL portfolio of well over one hundred CAVE applications instantiated so far was intended to prove the point on PLE, which when based on a well defined set of core assets can make it possible to optimize development of CAVE applications in terms of both time and staffing requirements. The product delivery time (or so to speak time-to-market, although still within the university) for any of our applications rarely exceeded three months with at most three programmers/testers required to instantiate the product for the target bigCAVE system. We have demonstrated that the SPL paradigm can provide a cost-effective solution for rapid instantiation of CAVE applications. Thanks to the fact that it can exploit a common, managed set of features of software products destined to be deployed in the same target CAVE system, the otherwise different products can be developed as one. By defining the feature model for our product line, generating core assets from our first products (at that time crafted for bigCAVE) and setting up next the appropriate production stages supported by the specialized hardware, we were able to keep the delivery time within the limits imposed by the academic schedule and optimize development time and effort while keeping the size of the development team small. Adoption of the SPL approach also contributed greatly to improving quality of each new final product instance released, as functional and performance testing of the latter in various production stages helped a lot in fixing potential problems of the shared assets (adding a new variation point, refining the performance model, adding new test cases, etc.), thus improving the entire family of products and not just that one product instance.

6 REFERENCES

[Fer02a] Ferber, S. et al. Feature interaction and dependencies: Modeling features for reengineering a legacy product line, in Proc. 2nd Int. Conf. Software Product Lines SPLC 2, pp.235–256, San Diego, CA, USA, 2002.

[Gha18a] Ghabach, E. et al. Clone-and-Own software product derivation based on developer preferences and cost estimation, in Proc. 12th Int. Conf. on Research Challenges in Information Science RCIS 2018, pp.1–6, Nantes, France, 2018.

[Kacz21a] Kaczmarek, A., Lebiedź, J., Jaroszewicz, L., and Święszkowski, W. (2021). 3D scanning of semitransparent amber with and without inclusions. In *Proc. 29th Int. Conf. on Computer Graphics, Visualization and Computer Vision (WSCG 2021)*, pages 145–154, Plžen, Czech Republic, May 17-20 (on-line).

[Kow18a] Kowalczuk, Z. and Tatara, M. Sphere drive and control system for haptic interaction with phys-

ical, virtual, and augmented reality, IEEE Trans. Control Syst. Technol. 27, No.2, pp.588–602, 2018.

[Lan16a] Landowska, A., Szwoch, M., and Szwoch, W. Methodology of affective intervention design for intelligent systems, Interacting with Computers 28, No.6, pp.737–759, 2016.

[Laz94a] Lazard, D. and Merlet, J. The (true) Stewart platform has 12 configurations, in Proc. 1994 IEEE Int. Conf. on Robotics and Automation, pp.2160–2165, San Diego, CA, USA, 1994.

[Leb21a] Lebiedź, J. and Mazikowski, A. (2021). Multiuser stereoscopic projection techniques for CAVE-type virtual reality systems. *IEEE Trans. Human-Mach. Syst.*, 51(5):535–543.

[Leb16a] Lebiedź, J. and Redlarski, J. (2016). Applications of Immersive 3D Visualization Lab. In *Proc. 24th Int. Conf. on Computer Graphics, Visualization and Computer Vision (WSCG 2016)*, pages 69–74, Plžen, Czech Republic, May 30-June 3.

[Leb16b] Lebiedź, J. and Szwoch, M. (2016). Virtual sightseeing in Immersive 3D Visualization Lab. In *Proc. Fed. Conf. on Computer Science and Information Systems (FedCSIS 2016)*, pages 1641–1645, Gdańsk, Poland, Sept. 11-14.

[Mar22a] Martinez, K., Lebiedź, J., and Bustillo, A. (2022). Graphical interface adaption for children to explain astronomy proportions and distances. In *Proc. 30th Int. Conf. on Computer Graphics, Visualization and Computer Vision (WSCG 2022)*, pages 1–8, Plžen, Czech Republic, May 17-19 (on-line).

[McB02a] McBreen, P. Software craftsmanship: the new imperative, Addison-Wesley, 2002.

[Mor02a] Morisio, M., Ezran, M., and Tully, C. Success and failure factors in software reuse, IEEE Trans. on Soft. Eng. 28, No.4, pp.340–357, 2002.

[Nor12a] Northrop, L. and Clements, P. A framework for software product line practice, version 5.0, white paper REV-03.18.2016.0, Software Engineering Institute, Carnegie Mellon University, 2012.

[Pet12a] Petridis, P. et al. (2012). Game engines selection framework for high-fidelity serious applications, Int. J. of Interactive Worlds, art. ID 418638.

[Ros11a] Rosenmüller, M. et al. Flexible feature binding in software product lines, Automated Software Engineering 18, No.2, pp.163–197, 2011.

[Tho07a] Thorton, M. (2007). Surround sound from stereo. Upmixing plug-ins for Pro Tools. *Sound on Sound*, https://www.soundonsound.com/reviews/surround-sound-stereo.