Journal of Universal Computer Science, vol. 27, no. 2 (2021), 152-169
submitted: 6/4/2020, accepted: 1/2/2021, appeared: 28/2/2021 CC BY-ND 4.0

Videogame development training approach: A Virtual Reality and open-source perspective

David Bonilla Carranza
(Universidad de Guadalajara CUCEI, Guadalajara, Jalisco, México
https://orcid.org/0000-0002-8690-4865, jose.bcarranza@academicos.udg.mx)

Adriana Peña Pérez Negrón[1]
(Universidad de Guadalajara CUCEI, Guadalajara, Jalisco, México
https://orcid.org/0000-0001-6823-2367, adriana.pena@cucei.udg.mx)

Madeleine Contreras
(Universidad de Guadalajara, Sistema de Universidad Virtual, Guadalajara, Jalisco, México
https://orcid.org/0000-0003-0336-7003, mgabriela@suv.udg.mx)

Abstract: Videogame development is software development; though, videogame projects require unusual developers' technical and creative skills when compared with non-game projects. Based on the teaching experience of the Computer Simulation subject of the Computer Science Engineering program for undergraduate students, it was built a training strategy for videogame development projects oriented to software developers. As a result, it is presented the strategy described in terms of videogames for immersive virtual reality with open source platforms, but it can be adjusted to other technologies. This proposal links Project-Based Learning (PBL) with the SUM videogames development process, by including material, tools, and a creative perspective.

Keywords: Videogame design, Virtual environments simulation, Project-based learning, Human-computer interface
Categories: L.3.1, L.3.4, L.5.1
DOI: 10.3897/jucs.65164

1 Introduction

Videogames represent a growing billion dollars' industry [WePC, 20]. Although it constitutes an important part of the software industry, as Murphy-Hill, Zimmermann, and Nagappan [14] pointed out, the research community had not studied them enough. They conducted a study to understand how videogames development is different from software development; the study consisted of interviews and surveys. Results showed that there are significant differences between traditional software and videogame development. Namely, videogames developers use less clear requirements, tend more to use the agile process, better appreciate creativity and the ability to communicate with non-engineers, and require more diverse team members' abilities than non-game developers.

[1] Corresponding author.

As a result, even though videogames are software, their development differs from 'traditional' or non-game software development. Following Murphy-Hill, Zimmermann, and Nagappan [14], Pascarella et al. [18] conducted a study to understand these differences, but with a focus on open-source platforms for videogames, in three topics: 1) the developers' contribution to the project; 2) the malfunctions handle, and; 3) the development process perception. They analyzed the diversity of the file sources in public code repositories, results showed that in the non-game projects the files correspond mainly to the development category (80% in their study), while in the videogames projects the files correspond mainly to the multimedia category (70% in their study), which includes audio, images, and game scenarios type of files.

In the Pascarella et al. [18] study was found that videogame development teams create specialized sub-teams in charge of the evolution of different aspects of the game. Also, in videogame development teams were identified specialized owners that work on non-code related categories. Regarding malfunctions, the second topic, it seems that faults in videogames are spread in different categories with higher problems in graphics, while in non-game projects faults are mainly related to programming, which confirms the Murphy-Hill, Zimmermann, and Nagapann [14] results.

For the third topic, the videogame development process perceptions, a survey was conducted. Results showed that for videogames the reuse of code is more difficult to apply, it is less possible to overview the project requirements, and more difficulties were reported for automated tests when compared to traditional software. As a result, Pascarella et al. [18] concluded that for videogame development are required:

1) New methodologies to better support development;
2) Patterns design for better extensible and reusable source code;
3) Different approaches to support corrective actions; and,
4) New methodologies to identify malfunctions, and automatic testing.

It follows from this that videogame development requires different teaching/training strategies.

Although there is extensive literature on learning how to develop videogames, there are scarce proposals with approaches to teaching videogame development. Most teaching approaches in software development do not specialize in the development of videogames, which might lead to misunderstandings in their study causing the lack of an adequate training strategy.

In education, videogames have been found to motivate learning [Marín-Vega, 19]. Pivec and Dziabenko [04], recommends the use of videogames to support and facilitate the learning process, enhancing the collaborative social context of education. They presented a game concept for social skills and knowledge training. Bell and Gresalfi [17] explored how experiences of teaching with videogames impact classroom integration through a digital problem-solving game in mathematics. They identified the level of the teacher experience in the game as a key factor that directly affects the students learning.

More recently, Coleman and Money [20] presented a literature review to understand the student-centered learning techniques in which the use of videogames is the vehicle for learning. Student-centered learning follows the constructivist theories, where students are active participants of their own learning. They found a strong focus on the combination of student-centered technique and the use of videogames on elements such as active learning, deep learning and understanding, the development of senses of autonomy and increased responsibility, and accountability. And in social elements such as mutual respect, teaching and learner interdependences, and reflexive attitudes to learning and teaching.

Claypool [05] designed a teaching approach for the Software Engineering subject applying a videogame development process, generating an increase in the class enrolment, a significant improvement in grades, and comments from students suggesting a greater interest in Software Engineering. While there are merits to these approaches, they use videogames as a means to motivate learning and not as the main teaching subject.

For teaching videogame development, the Massachusetts Institute of Technology (MIT) has an Open Course Ware aimed to introduce students to working in multidisciplinary teams. The course comprehends four projects, emphasizing the group work to solve creative problems, encouraging interaction across different aspects of the videogame design (e.g., game design, audio design, visual aesthetics, fiction, and programming), and for the Project Management, they use Scrum practices. In this course, the students are motivated to work on projects aimed to help somehow society [Philip, 14]. The teaching strategy is explained in terms of what the student can expect from the course.

Coleman, Roebke, and Grayson [05] developed the Gedi (Game Engine Design and Interface), an open-source game engine to teach videogame design and programming in C++ in project-oriented elective courses for undergraduate Computer Science students. They argued that a game engine abstracts the APIs details, allowing programmers to focus on implementing the game design. The first half of the course initiates by analyzing and documenting a 2D game design and then programming the game using a minimal set of features of the Gedi. By the end of the first-half course, the students built a prototype game with design features such as play rules, real-time control, animations, collision detection, among others. In the second-half course, the students work in reviewing and debugging code, and discussing implementation issues to end up with a complete game development.

For a teaching approach for videogame development, it is important to highlight that videogames require two main different areas of development. 1) The technological area, which involves the elements of the development process, the specific definition of hardware and software; and 2) the creative area, which is the conception of the idea [Chamillard, 06]. The creative area includes planning, art design, and defining the narrative integrated into the game elements [Tschang, 03]. The whole process requires a collaborative effort between design and programming.

On the other hand, the Virtual Reality (VR) industry has become a fashionable trend. Although it is not yet a reality for regular computer users, several companies are trying to get on board with this constantly growing branch of technology [Cuenca, 18]. Nowadays, VR's main challenge is getting inside our daily lives [Mora, 14]. Although not an easy task, the continuous decline in prices for special VR devices, particularly the head-mounted display (HMD) for immersive virtual reality, along with better computer performance is bridging the gap [Contreras, 18].

VR can be defined as the technology that joins three-dimensional (3D) graphics and users' interaction. Therefore, VR is intrinsically related to videogames since 3D-graphics videogames are VR [Peña, 12]. Specialized institutes that focus on the development of videogames are incorporating VR as a formal subject for undergraduate students [Cuenca, 18]. Currently, the demand for specialized professionals in VR, along with videogame developers, is increasing [Peña, 12, Cuenta18]. Therefore, it seems beneficial to start preparing software developers in these areas.

In this paper, a proposal for training software developers in videogame immersive virtual reality-based projects is formally presented. This approach has been applied and improved during eight-semester courses of the Computer Simulation subject for undergraduate students in Computer Science Engineering [Bonilla, 20]. The approach links the videogame development methodology of Acerenza, et al. [09], and the Project-based learning (PBL) instructional methodology.

The paper is structured as follows: After this introduction, the next section 2 presents a background in which the basic areas of the approach are described. Section 3 presents the proposed approach for videogame training. And section 4 addresses conclusions and future work.

2 Background

One of the key features of the game design is the environment in which it will be played. Videogames' main platforms are personal computers, game consoles, and mobile devices [Aleem, 16].

In this case, immersive VR was selected as the computer technology for the proposal, which can be adapted for the different platforms aforementioned [Bonilla, 20]. Virtual reality is a 3D-graphics computer-generated scenario, with which the user can interact; VR is considered immersive when the user cannot interact but with the virtual environment [Peña, 12]. In recent years, the use of the HMD device for immersive VR is increasing mainly because of the drop in its price. HMDs have a head tracker to display the image in the correct perspective, calculated from the position of the user's head, direction, and orientation. The virtual scenario is displayed for each eye in a stereo view to create the 3D perception [Yu, 08]. The most popular HMD branches and their platforms for videogames are next listed:

- HTC Vive^{TM} for PC
- Oculus ^{TM} for PC and Xbox
- PlayStation VR ^{TM} for PS4
- Google daydream ^{TM} for Android devices
- Google cardboard ^{TM} for Android devices
- Nintendo labo vr kit, a cartoon device for Nintendo console
- Windows ^{TM} Mixed Reality for PC
- Lenovo ^{TM} Mirage Solo with Daydream for Android devices

Most computer games can be adapted for VR taking advantage of immersion devices; also, new games based on VR are continuously emerging [Koss, 20]. However, this is not a straight forward transition with particular challenges to overcome.

2.1 Virtual reality development challenges

Among the main challenges in the process of developing VR-based videogames is the conceptualization of the human-computer interaction (HCI). Studies in this area search for new designs on hardware and software that interpret the human behavior characteristics to adapt them to the digital environment. The emergence of new HCI technologies aims to improve the user's experiences more naturally and efficiently.

HCI in VR looks to create sensory information for the user. In VR the senses mainly involved are sight and sound, and in some cases touch, a key consideration for HCI solutions. In this context, Sutcliffe et al. [19] suggested including a cognitive psychology approach for HCI in the VR design process, with a user-centered design by considering the psychological constraints such as attention, working memory limitations, visual dominance, and sensory integration.

In any case, it has to be kept in mind that even though VR is founded on both hardware and software that are constantly improving, its workflow development remains, that is, concept, preproduction, production, testing, polishing, and delivery [Jiménez, 16].

Typical videogames consumers are used to certain usability interaction formulas, where VR represents a new dimension. In immersive VR the fourth wall is broken; the fourth wall metaphor refers to an imaginary wall in front of the videogame through which the audience sees the story of the game [Rodríguez, 16]. This is accomplished by placing the player inside the scenario and allowing the user to interact with it.

Rodríguez [16] highlighted three significant challenges for the development of videogames based on VR:

1. *The movement system.* In a VR-based videogame, the movement of the player cannot follow the typical game rules. It is important to be cautious of not producing "dizziness", caused mainly due to the displacement of frames that do not match a real-world situation, also known as cyber annoyances [Cuevas, 13].
2. *The player's camera.* It has to be more natural and flexible. Also, because it is possible to play with the game assets position and the close exploration of environments, the designer should take care of the nearby textures.
3. *Interaction with the environment.* It has to be taken into account that the interaction is different; the gamepad might not be the only control system. For example, the new hands tracking systems radically change the playing interaction [Rodríguez, 16].

These differences for VR videogames imply different design concerns. In the next section, some key issues for videogame design are briefly discussed.

2.2 Videogame Design

There is no consensus on a definition of game design [Kultima, 15]; though, through its design, the videogame structure is settled. Ernest Adams [13] defines game design as the combination of the functional elements of the game and its aesthetics, aimed to create leisure.

The main functional elements in the game are the mechanics, the rules, processes, and data that bring the game to life. As a first design element are the core mechanics, the most influential ones, followed by the identification of other mechanics. The game mechanics can be classified as follows [Adams, 12]:

- *Physics*. Motion and force of the elements in the game, including timing and rhythm, which can be different from the real world.

- *Internal economy*. The transactions involve game elements that can be collected, consumed, and traded; including abstractions such as health, popularity, and magical powers.
- *Progression mechanisms*. Such as levels, indicating the progress of the player that might manage control mechanisms that lock or unlock access to certain areas.
- *Tactical maneuvering*. Like the placement of game units on a map for offensive or defensive advantages, critical in strategy games, some role-playing, and simulation games.
- *Social interaction*. The mechanics that reward interaction with others such as giving gifts or inviting new friends to join the game.

The game mechanisms are implemented to create an engaging user experience, desires, and motivations [Mitre-Hernandez, 16]. The mechanics' relations in the game generate the dynamics [Adams, 12]. While the dynamics are expected to evoke a physiological emotion in the gamer [Chow, 17]; though, there is no agreement on a direct relation of the mechanisms with the dynamics game element [Muñoz, 18].

Once the basic elements of the game are established, they are developed using different techniques and tools. The requirements of the game are formulated in a document, the game design document (GDD) [Salazar, 12], which presents the game structure.

For game design, as for traditional software, visualization communications are essential to identify and construct cases, charting the flow, modeling sequences of events and showing relations [Araújo, 09]. The most common diagrams for non-game software are UML and flowcharts. According to Araújo [09], for game design, UML diagrams at a conceptual level lack formal semantics, and in large and complex models can present a problem for validation. And flowcharts are limited to model sequential, non-concurrent systems, and then they are not proper for game design.

Alternatives in the industry are Token-based models, where tokens are the basic elements of the game such as characters, player avatar, or mechanics [Rollings, 04]. Diagram-based Machinations, also dividing the game into elementary pieces that can be implemented using Petri nets [Araújo, 09], or UML adaptations [Dormans, 08, Tenzer, 07]. Other interesting alternatives for game design visualization can be found in [Djaouti, 13].

Finally, it is important to underline that a narrative has to guide the story in the game [Neil, 12], central mechanics and the narrative are the main components of the game. The players must get a clear understanding of the story and be aware of the characters they are not interpreting. The narrative can be understood as the background, while the mechanics are the foreground. Central mechanics and narrative will make an impact on the audience. Take for example 'The Super Mario Bros^{TM}' videogame, which is based on a character with a main mechanism, the 'jumps' that help the character to solve different problems, such as jumping from platform to platform. The mechanics accompanied the story or narrative, the character has to save a princess, a context of why the player does everything.

The difference between traditional and non-game development represents important implications not only in the design phase but in the development process as well.

2.3 Videogame Development Process

The complexity in the development of videogames involves diverse activities in the creative arts disciplines such as storyboarding design, animation, artificial intelligence, video production, music, and sound, which has lead to the guidelines for the game development process life cycle [Aleem, 16].

Aleem et al. [16], based on a systematic literature review, established that the process life cycle for videogame development can be merged in three main phases, as shown in Figure 1. According to them, in the videogame life cycle, the development process takes place in the Production phase, in contrast with traditional software in which the production process is when the software actually runs and is ready for use. In the Pre-production stage, the game scenarios are tested for feasibility, and the Post-production stage is for testing, marketing, and advertising. The framework Scrum for videogame development follows these same phases but named as Pre-game, Game, and Post-game phases [Sutherland, 15].

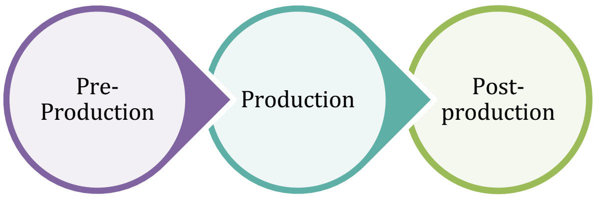

Figure: 1 Stages for videogame development life cycle [Aleem, 16]

Acerenza et al. [09], based on the Scrum framework, created what they call the SUM methodology. In SUM are integrated two more phases; on this approach, the Pre-production or Pre-game phases correspond to the Planning and Elaboration phases. The third phase, Post-production or Post-game corresponds to the Beta and Closing phases. Also, a first phase, Concepts, is included; Figure 2 depicts this structure. The SUM phases are next briefly described:

Phase 1: Concepts. This phase's objective is to specify the required basic elements for the design of the videogame; such as target audience, business model, game elements, main features, gameplay, characters, and history. In this phase is also defined the technical languages and tools. Here, takes place the Development of the game concept.

Phase 2: Planning. In this phase, the project schedule is built along with its main milestones. Each of the videogame's functional and non-functional characteristics is estimated and prioritized. Here, the teams are defined according to technical requirements. This phase is flexible and adaptable according to the evolution of the project. An administrative plan and the videogames specifications are settled.

Phase 3: Elaboration. In this phase the videogame is implemented, the process follows an iterative and incremental approach to get an executable version. This phase is divided into three threads; objectives planning, execution of the tasks, and the evaluation of the state of the videogame.

Phase 4: Beta. This phase aims to evaluate and adjust different aspects of the game such as gameplay, fun, learning curve, and its difficulty curve, as well as eliminating detected errors.

Phase 5: Closing. To conclude the process, in this phase a final version of the videogame is developed based on the requirements. Also, feedback is gotten through an evaluation of the problems that might occur, the achieved successes, the presented solutions, the fulfillment of objectives, and the estimates' accuracy.

Finally, throughout the whole project development, risk management has to be present, seeking to minimize the consequences and impact of possible detected problems. Risks must be analyzed to establish the probability of occurrence and impact, mechanisms of monitoring, and mitigation strategies within a contingency plan.

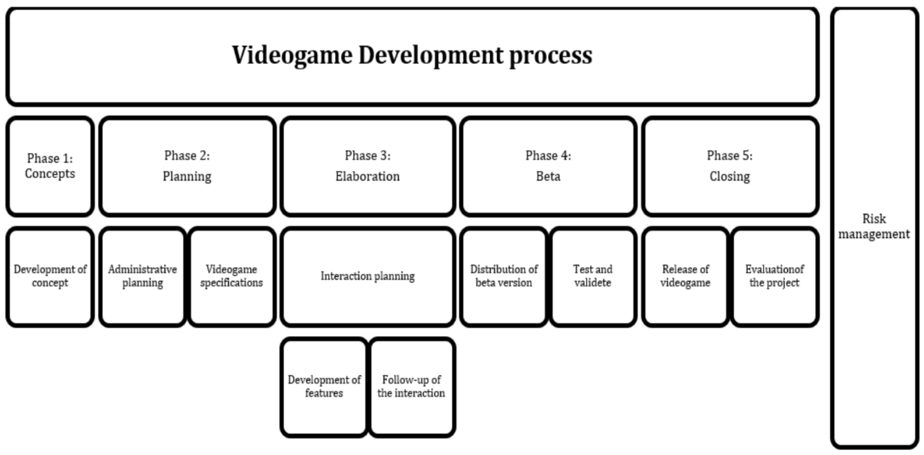

Figure 2: Development phases for videogames [Acerenza, 09].

The SUM methodology process also presents four development roles in the creation of a videogame: developer, internal producer, client, and beta tester. The developer has the Scrum team characteristics, but with industry typical sub-roles within the team: developer, graphic artist, sound artist, and game designer [León, 07]. The internal producer and the client correspond to the Scrum Master and the Product Owner Scrum roles respectively. And the beta tester, not present in Scrum, is responsible for the functional verification of the videogame. Special attention requires Phase 1: Concept, not included in the Scrum for games approach because the game design is key for its development and it is based on creative skills.

The SUM methodology is flexible and adaptable, proper to be linked to a training approach for videogame development; the complete detail of SUM is available at [SUM, 09].

As the videogame development training approach should include an actual game development [Coleman, 05; Philip, 14], Project-based learning (PBL) was selected as a suitable instructional methodology [Remijan, 17].

2.4 Project-based learning (PBL)

Videogame development, as well as non-game software development, should be taught not only as a tool of interest for individualized learning but as a support for teamwork learning and the creation of information. In the application of PBL, the team is the foundation to carry out learning [Pérez, 2015].

Larmer et al. [15] presented their PBL gold standard with 8 elements to complete a project. At the center is the element of the *key knowledge and understanding* applied to the real world with the key success skills, which include critical thinking and solving problems, working well with others, and managing themselves effectively. In order to achieve such knowledge and skills, the project design requires to include the next elements: *challenging problem or question*, this will guide students to acquire knowledge to solve the problem or answer the question; *sustained inquiry*, to provoke investigation, exploration, looking for alternative solutions using several sources; *authenticity*, a problem from the real world that includes the students' personal concerns; *voice and choice*, a proposed solution from teamwork, creating ownership; *reflection*, examination of what has been learned; *critique and review*, receiving constant constructive feedback; and *a public product*, a tangible product to solve a problem. These elements support the learning experience through a suitable project.

Likewise, the students must identify the difficulties and mistakes during the development process, so they can reach learning. This aspect is accompanied by the basic definition that reinforces the conceptualization of the workflow in the development of projects when the student can overcome difficulties; this is an intentional exercise [Marti, 10]. According to PBL, this self-regulation learning is a self-directed process through which learners transform their mental abilities into academic skills.

PBL supports students during the learning process through two fundamental elements: evaluation strategies, and classroom management in collaborative workgroups [Pérez, 15], which is proper for a teamwork process for videogame development.

PBL for videogame development training is an integral methodology where teaching expectations end on a functional product. Using as a framework the SUM methodology [Acerenza, 09], our teaching approach for videogame development was linked to the PBL methodology.

3 The training process for videogames

For the training process for videogames are taken as a basis the Acerenza et al. [09] phases adapted to the PBL learning process, using a VR design perspective, and including suggestions regarding open source tools. During project execution, the student should acquire the required skills and knowledge through the development of practices that reinforce the basic concepts to create VR-based videogames.

This teaching approach provides predictable results. The students must manage time resources efficiently, consider probable risks during the development of the project, and manage constant interaction with teammates [Zarraonandia, 12]. When using SUM [Acerenza, 09], they band together to adjust small multidisciplinary teams from three to seven members working in the same physical location or distributed, and in a short project with a duration of no more than six months, looking for a high degree of participation. The next sections describe the complete approach in phases.

3.1 Phase 1 Concept

In this phase, basic concepts are transmitted where workflow can be known in the development of a VR videogame for HMD devices. It includes teaching the integration of user interface design techniques [León, 17], low polygon modeling, and videogame mechanics based on graphic engines. The way to evaluate this knowledge is through relevant activities, such as summaries or conceptual maps.

A good way to evaluate the game design is the construction of a prototype that can be made of paper [Adams, 12], a quick and economic way that will help to understand the player's interaction with the game mechanics. It is important to highlight the reward element in videogames, a motivational factor for the player, very successful and frequently used in mobile devices. This mechanic is based on the Japanize gacha, mechanic machines that give a surprise prize for players.

The game design is a creative activity that can be demonstrated through successful cases. Next are listed some videogames recognized for their great design, that can help as a support in this process, for example:

Boktai: The Sun is in Your Hand^{TM}, released in 2003, is an example in which technology has a key impact on the design. Through photosensors, the light in the real world is detected to modify the digital game.

Challenge as conflict can be found in Dark Souls^{TM}, released in 2011, a third-person action role-playing game in which the player learns from past mistakes. This is considered a highly difficult game.

The Joruney^{TM} developed in 2012 by Thatgamecompany emphasizes the aesthetics in a videogame, which in this case tends to provoke astonishment for the user. This emotion is reinforced by matching soundtracks for each scenario, transmitting an immediate response to the player's actions.

Super Mario Maker^{TM} released in 2015 for Wii U^{TM} console, shows how physics mechanics are used for levels in which the character stops avoiding obstacles to be the game developer constructor.

Celeste^{TM}, released in January 2018, is a kinesthetic game (i.e. requires a psychomotor effort to be played). It has mechanics' interaction, jumping, climbing and a mid-air dash (sustained jump) to be improved during the game according to the affronted challenges.

The creative areas of knowledge acquisition are mainly based on observation. The recommendation is to study successful cases, to reinforce the development of new ideas.

To implement this phase, the practices can be focused on the topics of videogame design, main components of VR, principles of 3D modeling, and the creation of elements for the design of user interfaces. The students can be required to make a concept map workflow at the end of this phase.

Phase 1 correspond*s to Phase 1: Concepts* of *SUM*, and it is linked to *PBL* by the golden elements *challenging problem* and *sustained consultation*.

3.2 Phase 2 Techniques

The techniques phase is also related to the *SUM Phase 1: Concept* phase reinforced with practices such as definition, but mainly by learning specific software tools. Once basic concepts are comprehended, developers acquire technical skills using applications. These practices are focused on training different concepts of the software development process for VR-based videogames.

At this point, each practice must have an incremental relationship so they can be later integrated as a whole into a final project. Through this phase, the developers will acquire the skills and knowledge to develop the game concept, which has to be reached at the end of the phase, also technical languages and tools are chosen.

For the development of the concept, it is important to consider a real problem to solve, looking to have an impact on the community giving rise to the *PBL* golden element of *authenticity.* Also, the developers will establish the whole concept of the project, what they intend to solve through it, which correspond to the PBL golden element *voice and choice.*

Applications and tools. The suggested support applications, open-source or free for use software for the technical phase are:

For the videogame design StarUML software (http://staruml.io/), and Dia Diagram Editor (http://dia-installer.de/).

For the interface elaboration vector graphics manipulation software is needed. Inkscape™ (https://inkscape.org/) is suggested because it allows fluid game resource generation.

For the sound effects, the Audacity™ (https://www.audacityteam.org/) software is suggested, also it facilities musical composition.

And for the 3D object modeling, Blender™ (https://www.blender.org/) is recommended. In Blender™ can be generated scenarios, characters, and all the 3D elements for the environment. This application supports textures and material for 3D objects to get a realistic scene.

3.3 Phase 3 Integration

It is sought that the practices developed in the previous stage will serve to acquire fundamental techniques knowledge in the development of VR videogames. That is why during their practice the students have to be focused on a final project. The integration phase consists of how the learning process is directed to the collaborative learning project, based on the previously acquired knowledge.

The *PBL* golden element reached here is *reflection* because the skills and knowledge previously acquired support videogame planning.

This phase corresponds to the *SUM Phase 2: Planning* where the videogame characteristics are estimated, the teams are built, and the administrative plan is developed.

The team members discuss the game design. At this point, they can define the basic resources needed for the videogame such as characters, 3D models, sounds, treasures, and scripts to create the VR application.

Application and tools. For the administrative plan, based on Scrum can be used WeKAN (https://wekan.github.io/) to monitor the development of the project through cards in columns with the different tasks; this platform is linked to the free software repository GitHub (https://github.com/)

3.4 Phase 4 Implementation

In this phase, the different resources are developed and integrated into a game engine. This is the *Phase 3: Elaboration* in *SUM.* Following SUM, the suggested approach is iterative and incremental, and with constant evaluation on the released incremental developments.

The recommended game engine is Godot™ (https://godotengine.org), a videogame engine with a series of libraries for the design, creation, and representation of a videogame. Its most outstanding aspects are its graphic capabilities responsible for displaying 2D and 3D images on the screen, as well as calculating some aspects as polygons, lighting, and textures. This game engine supports exporting the game into different platforms, including VR environments. Godot™ also handles physics, calculating attributes such as weight, volume, acceleration, or gravity. It has a sound engine, for loading tracks, modifying their bit rate, removing them from playback, and synchronizing, for example. Videogame engines are based on a programming language to implement functioning characters and objects, Godot™ uses C# as a programming language to write scripts.

In order to implement the HMD device can be used the Android, HMD's library.

Here it is important to consider the VR interface for detecting anomalies involving the interaction of the virtual interface. It can be detected, for example, if the displacement is correct or if there are dizziness problems that need to be fixed. In this phase, it is also important to run validation and verification tests.

Figure 3 shows a VR game project in a stereo view. This is the first test with external control, where can be seen the displacement of the character within the game level. A deliverable function.

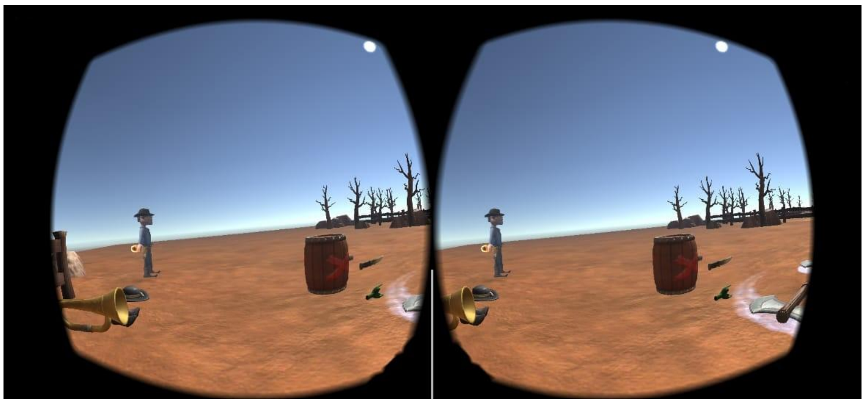

Figure 3: The user interface of a working project.

This Implementation phase also includes the *SUM Phase 4: Beta*, to evaluate and adjust aspects such as gameplay. The *PBL* gold element achieved in this phase is *critique and revision,* to reinforce such gold element it is recommended for the developers to present their progress to their peers and the instructors.

3.5 Phase 5 Closing

This corresponds to the *SUM Phase 5: Closing*, where the product is released and the project evaluation is conducted. In the closing phase, feedback on the finished product is received, and it is presented to an audience. The final presentation is a functional prototype, where the students receive constructive comments on their projects.

This Phase is related to the golden element of *PBL* of *public product.*

The SUM and PBL integration are depicted in Figure 4. On the left side are the SUM process phases, where risk management is a transversal feature. At the center is the proposed training approach, separated by phases with its main elements. Here can be observed which SUM phases correspond to the proposal. Then, on the right side, the PBL gold standard elements are also associated with each phase of the proposal, and as transversal golden element are the key knowledge and understanding, and success skills.

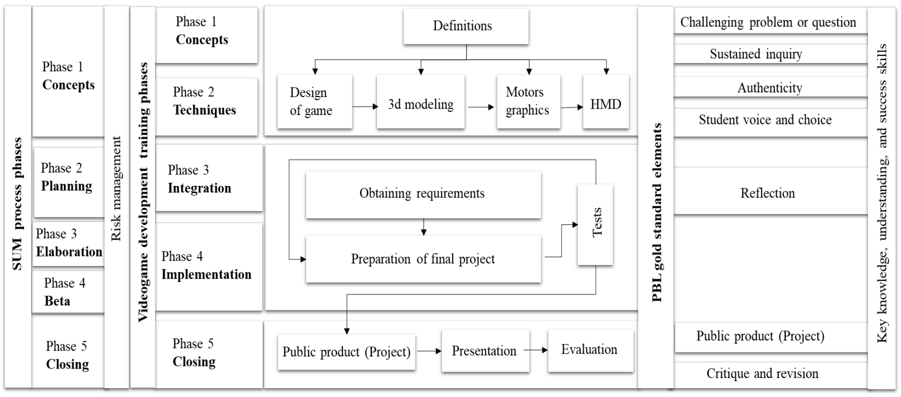

Figure 4: Training process for VR-based videogame (based on PBL and SUM)

3.6 Limitations

This approach arises from five years of teaching experience in the Computer Simulation subject of a Computer Science Engineering program for undergraduate students. While this helped to create the strategy, it also supported some limitations detection. The main weaknesses founded for the success of the training course are those related to the videogame technology orientation, in this case, free software platforms and VR, individual deficiencies, and teamwork soft skills.

Free software is usually distributed over the world, with potential peer review, attracting excellent programmers, but those developers are unpaid volunteers making it impossible to fully rely on the project participants [Michlmayr, 03], resulting frequently in a lack of proper support. For developers not used to free software platforms this extra work might at first be frustrating.

VR as mentioned requires a different HCI that might create confusion, especially if the student is not familiarized with this technology. Also, VR represents a high-cost computer resource, that might not be available, although this had been overcome using mobile devices and free software platforms.

At the individual level, the parse of the training sometimes is not suitable for everyone. This has been observed mainly in basic videogame development knowledge, required for the next steps. Also, computer simulation concepts are not an easy topic for everyone. Besides, formalizing the technical acquired knowledge to apply it to the videogame project is not a straight forward task for some students.

The artistic creative thinking might also represent a barrier for very technical people. Because of the lack of experience in creativity tasks, time estimation for non-game software developers can become a complication.

As for the teamwork, the instructor very frequently will have to intervene for team formation. A number of software developers do not exhibit proper teamwork soft skills; in these cases, it is not easy to make them fill as part of a common project. Finally, students might have a hard time assigning individual activities and responsibilities to aim for the shared goal.

4 Conclusions and future work

The construction of virtual environments is a complex task, so it is necessary to use a training approach that allows software developers to focus on different skills to understand the complete process.

In this paper, a training approach for software developers to create a VR-based videogame for HMD technology was presented, the approach is based on the videogame development process SUM [Acerenza, 09], following the PBL gold standard elements [Larmer, 15] and oriented to VR, using open source tools. It is worth mentioning that the software proposed for this approach is based on free software, which allows granted access to the tools.

It is important to highlight that training methods need to be adaptable for new technologies to come, allowing new generations to enjoy dynamic and interactive content development learning.

We consider that a learning system based on projects is an excellent way to transmit knowledge and getting feedback on a specific topic. Developers' general comments support the merits of the approach; they show a great interest in the VR development process as a whole.

Through several semesters, we have gotten good comments from the students. Many of them refer to the subject as one in which they improved their technical and project development skills. With this approach, developers naturally discover the importance of knowing the workflow of a VR development on time, understanding the differences with traditional software development. It is very favorable for developers to learn during the development of a project and to interact in teams, where the social part helps them to solve doubts, enhance their social skills, promoting collaborative learning. The structure of the course is presented to reinforce the development of the project by addressing both the theoretical and practical aspects.

References

[Acerenza, 09] Acerenza, N., Coppes, A., Mesa, G., Viera, A., Fernandez, E., Laurenzo, T., Vallespir, D.: Una Metodología para Desarrollo de Videojuegos. In: 38o JAIIO - Simposio Argentino de Ingeniería de Software, Argentina, 171-176 2009

[Adams, 13] Adams, E.: Fundamentals of game design, 30-31 52 2013 https://doi.org/10.1017/CBO9781107415324.004.

[Adams, 12] Adams, E., Dormans, J.: Game mechanics: advanced game design. New Riders, 6, 17-19 2012

[Aleem, 16] Aleem, S., Capretz, L.F., Ahmed, F.: Game development software engineering process life cycle: a systematic review. Journal of Software Engineering Research and Development. Brazilian, 4, 2016 https://doi.org/10.1186/s40411-016-0032-7

[Araújo, 09] Araújo, M., Roque, L.: Modeling games with petri nets. Breaking New Ground: Innovation in Games, Play, Practice and Theory - Proceedings of DiGRA, London 2009

[Bonilla, 20] Bonilla, D., Peña Pérez Negrón, A., Contreras, M.: Teaching Approach for the Development of Virtual Reality Videogames. In: Advances in Intelligent Systems and Computing. 276–288 January 2020 https://doi.org/10.1007/978-3-030-33547-2_21.

[Chamillard, 06] Chamillard, A.T.: Introductory game creation: no programming required. In: SIGCSE 2006, Houston, Texas USA, vol. 39, 515–519 2006

[Chow, 17] Chow, I., Huang, L.G.: A software gamification model for cross-cultural software development teams. ACM International Conference Proceeding Series. 2017, 1–8 https://doi.org/10.1145/3034950.3034955

[Claypool, 05] Claypool, K., Claypool, M.: Teaching software engineering through game design. In: Proceedings of the 10th annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2005, Monte De Caparica, Portugal, 123-127 2005

[Contreras, 19] Contreras, M., Bonilla, D., Peña Pérez Negrón, A.: Proposal of a model for the development of labor competencies based on serious games in the context of industry 4.0. In: Advances in Intelligent Systems and Computing. 80–87 2019 https://doi.org/10.1007/978-3-030-01171-0_7.

[Coleman, 05] Coleman, R., Roebke, S., Grayson, L.: Gedi: a game engine for teaching videogame design and programming. Journal of Computing Sciences in Colleges, vol. 21, no. 2, 72-82 2005

[Coleman, 20] Coleman, T. E., Money, A. G.: Student-centred digital game–based learning: a conceptual framework and survey of the state of the art. Higher Education, vol. 79, no. 3, 415-457 2020

[Cuenca, 18] Cuenca, D.: Game Studies: Estado del arte de los estudios sobre video juegos. Luciérnaga Revista Virtual, 2018, Trois-Rivières, Colombia, 13-24, 2018.

[Cuevas, 13] Cuevas, B.G. and Valero, L.: Efectos secundarios tras el uso de realidad virtual inmersiva en un videojuego. Int. J. Psychol., 63-178 2013

[Djaouti, 13] Djaouti, D.: Game Design Tools for Collaboration, 2013, http://www.gamasutra.com/view/feature/187777/game_design_tools_for_collaboration.php/ accessed March 2020

[Dormans, 08] Dormans, J.: Visualizing Game Dynamics 1 Running head : VIZUALIZING GAME DYNAMICS Visualizing Game Dynamics and Emergent Gameplay, 2008

[Jiménez, 16] Jiménez, E.M., Oktaba, H., Díaz-Barriga, F., Piattini, M., Revillagigedo-Tulais, A.M., Flores-Zarco, S.: Methodology to construct educational video games in software engineering. In: 2016 4th International Conference in Software Engineering Research and Innovation, IEEE, Puebla, 110-114 2016

[Kasurinen, 16] Kasurinen, J.: Games as Software: Similarities and Differences between the Implementation Projects. In: Proceedings of the 17th International Conference on Computer Systems and Technologies 2016 - CompSysTech '16, ACM Press, Palermo, Italy, 33-40 2016

[Koss, 20] Koss, H. What does the future of the gaming industry look like?, 2020, https://builtin.com/media-gaming/future-of-gaming

[Kultima, 15] Kultima, A.: Game Design Research. In: Proceedings of the 19th International Academic Mindtrek Conference, Association for Computing Machinery, New York, NY, USA, 18–25 2015 https://doi.org/10.1145/2818187.2818300.

[Larmer, 15] Larmer, J., Mergendoller, J., Suzie, B.: Setting the Standard for Project Based Learning - Based Learning, Assn, Virginia, USA 2015

[León, 17] León, N., Eyzaguirre, S., Campos, R.: Proceso de diseño de software y proceso de diseño sonoro para un videojuego educativo. Campus 22, 27-34 2017

[Marín-Vega, 19] Marín-Vega, H., Alor-Hernández, G., Colombo-Mendoza, L. O., Sánchez-Ramírez, C., García-Alcaraz, J. L., & Avelar-Sosa, L.: Zeus–a tool for generating rule-based serious games with gamification techniques. IET Software, vol. 14, no. 2, 88-97 2019

[Marti, 10] Marti, J.A., Heydrich, M., Rojas, M., Hernández, A.: Aprendizaje basado en proyectos: una experiencia de innovación docente. Revista Universidad EAFIT, Colombia, vol. 46, 11- 21 2010

[Michlmayr, 03] Michlmayr, M. and Hill, B. M.: Quality and the reliance on individuals in free software projects. Proceedings of the 3rd Workshop on Open Source Software Engineering. ICSE: Portland, USA 105–109 2003

[Mitre-Hernandez 16] Mitre-Hernandez, H., Lara-Alvarez, C., Gonzalez-Salazar, M., Mejia-Miranda, J., & Martin, D.: User experience management from early stages of computer game development. International Journal of Software Engineering and Knowledge Engineering, vol. 26 no. 08, 1203-1220 2016

[Mora, 14] Mora, M.A., Lopez, I., Meza, C.A., Portilla, A., Sánchez, N., Sanchez, C.R.: Metodología para el Desarrollo de Mundos Virtuales, con un Caso de Estudio: Independencia de México. vol. 10, 6-8 2014

[Muñoz, 18] Muñoz, M., Peña Pérez Negrón, A., Mejia, J., Gasca-Hurtado, G. P., Gómez-Alvarez, M. C., & Hernández, L. Applying gamification elements to build teams for software development. IET Software, vol. 13 (2), 99-105 2018

[Murphy-Hill, 14] Murphy-Hill, E., Zimmermann, T. & Nagappan, N. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? Proceedings - International Conference on Software Engineering 1–11 2014

[Neil, 12] Neil, K.: Game Design tools: Time to evaluate. DiGRA, 2012

[Pascarella, 18] Pascarella, L., Palomba, F., Di Penta, M., Bacchelli, A.: How is video game development different from software development in open source? In: Proceedings of the 15th International Conference on Mining Software Repositories - MSR 2018. 392-402, ACM Press, Gothenburg 2018

[Peña, 12] Peña Pérez Negrón, A. Collaborative Nonverbal Interaction within Virtual Environments. *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica*, 1-1, 2012

[Pérez, 15] Pérez, M.: Aprendizaje basado en proyectos colaborativos. Rev. Educ. 14, 158-180 2008

[Philip, 14] Philip Tan, Richard Eberhardt, Sara Verrilli, and Andrew Grant. CMS.611J Creating Video Games. Fall 2014. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

[Pivec, 04] Pivec, M., Dziabenko, O.: Game-based learning in universities and lifelong learning: "UniGame: Social skills and knowledge training" game concept1. Journal of Universal Computer Science. 10, 14–26 2004

[Remijan, 17] Remijan, K.W.: Project-based learning and design-focused projects to motivate secondary mathematics students. Interdiscip. J. Probl.-Based Learn. 11, 1 2017

[Rodríguez, 16] Rodríguez, T.: Cómo se hace un juego de realidad virtual, Xataka, 2016 https://www.xataka.com/ realidad-virtual-aumentada/como-se-hace-un-juego-de-realidad-virtual Accessed 3 May 2019

[Rollings, 04] Rollings, A. and Morris, D.: Game Architecture and Design : A New Edition. 483-486, 2004

[Salazar, 12] Salazar, M.G., Mitre, H.A., Olalde, C.L., Sánchez, J.L.G.: Proposal of game design document from software engineering requirements perspective. Proceedings of CGAMES'2012 USA - 17th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational and Serious Games. 81–85 2012 https://doi.org/10.1109/CGames.2012.6314556.

[SUM, 20] SUM Eclipse Process Framework Version 1.5 http://www.gemserk.com/sum 2020

[Sutcliffe,19] Sutcliffe, A. G., C. Poullis, A. Gregoriades, I. Katsouri, A. Tzanavari, and K. Herakleous. 2019. Reflecting on the Design Process for Virtual Reality Applications. International Journal of Human-Computer Interaction 35, 2, 168–179 2019

[Sutherland, 15] Sutherland, J. Sutherland, J.J.: Scrum: El revolucionario método para trabajar el doble en la mitad de tiempo. Grupo Planeta, Spain 2015.

[Tenzer, 07] Tenzer, J., Stevens, P.: GUIDE: Games with UML for interactive design exploration. Knowledge-Based Systems, vol. 20, no. 7, 652-670 2007

[Tschang, 03] Tschang, T.: When does an idea become an innovation? The role of individual and group creativity in videogame design. Presented at the Summer Conference, Copenhagen, Denmark, 1–26 2003

[WePC, 20] WePC, Video Game Industry Statistics, Trends & Data, 2020, https://www.wepc.com/news/video-game-statistics/#video-gaming-industry-overview accessed January 2020

[Yu, 18] Yu, D., Liang, H.N., Lu, F., Nanjappan, V., Papangelis, K., Wang, W.: Target selection in head-mounted display virtual reality environments. Journal of Universal Computer Science. 24, 1217–1243 2018

[Zarraonandia, 12] Zarraonandia, T., Ruíz, M.R., Díaz, P., Aedo, I.: Combining game designs for creating adaptive and personalized educational games. In: Presented at the the 6th European Conference on Games Based Learning, Ireland, October, 559-567 2012