The Use of Digital Games to Teaching Computer Graphics: an Open Opportunity

José Saúl González-Campos
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
jsaulg@uoc.edu

Joan Arnedo-Moreno
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
jarnedo@uoc.edu

Jórdi Sánchez-Navarro
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
jsancheznav@uoc.edu

***Abstract*—The use of digital games for learning in the higher education is a current trend that is mainly motivated by their pedagogical attributes if they are designed with specific learning objectives in mind. In the specific case of computer graphics, the intrinsic interactive, visual, and addictive nature of games seems to be a valuable coincidence for learning a discipline which is also highly visual and interactive. In this work, it is presented an overview of the efforts of educators and practitioners to teach computer graphics at the undergraduate level with a special emphasis on the application of digital games as a learning strategy. Authors aim to increase awareness about the detected necessity of diversifying the current offering of digital games in this particular area, which in turn can lead to improvements on how computer graphics is taught in the classroom.**

Keywords—computer graphics; games; education

I. INTRODUCTION

By its own definition and purpose, Computer Graphics (CG) is visual, where the benefits and adoption of graphics algorithms are driven equally by their visual results and by their performance in time and memory management. For this reason, introductory courses in CG need to provide students with learning material that can be appealing and as much visual and interactive as possible, in order to clarify the principles and techniques utilized in the CG discipline. Moreover, CG courses usually require students to perform conceptual manipulations of 3D environments in addition to learning the theoretical knowledge involved in this field. These conceptual manipulations, which are known as visual-spatial abilities in psychology, are difficult to some students. This means that teaching CG should rely on effective methods of presenting learning materials to students that favor these cognitive abilities.

Courses in CG are, in general, well supported by visual representations, dynamic input-output demonstrations, interactivity, and engaging material, in opposition to other traditional media such as text, blackboard, printed material (even visual), and business-style presentations. Games could provide all these desirable characteristics, and this is why the incorporation of games in the CG syllabus is worth to consider. Games could provide a valuable learning environment to immerse students in the study of CG.

This work is focused on exploring diverse initiatives that have incorporated interactive and ludic strategies for teaching computing topics in general, and computer graphics in particular, either in the classroom or e-learning environments. In this sense, learning what proposals have been implemented and tested in the past years provides valuable insight into their historical trend. For example, knowing that digital games have not been fully explored in the case of CG education should raise interest in knowing why. There is evidence that games have been applied successfully in the higher education in a wide variety of domains, including computing, and it is precisely this fact that might lead to think that CG should not be the exception. By the end of this study, authors will provide evidence of the detected opportunity for new research needed on the development of games for learning to teach the common topics covered in introductory CG courses.

This work is structured as follows: In Section II, it is presented an overview of the CG discipline, its knowledge base, and the evolution of the CG syllabus. In Section III, some approaches that have been proposed to teaching CG are reviewed. Section IV covers the use of digital games in the higher education with a particular focus on their application for teaching computing topics. Section V addresses the specific case of using games as well as interactive simulators for teaching CG. Finally, Section VI provides a discussion of the study findings and Section VII the conclusions.

II. THE COMPUTER GRAPHICS CURRICULUM

Computer graphics is the broad body of knowledge regarding the computer generation and manipulation of images. In the following subsections, it will be provided an overview of its knowledge base, as proposed by the Association for Computing Machinery (ACM), and of the evolution of its syllabus, from a historical perspective. The last subsection describes three studies regarding different visions about the definition of the CG syllabus.

A. The Computer Graphics Knowledge Base

It is worth considering that, although CG as a field has existed since some decades, the rapid advances in graphics technology have made difficult to have a consensus about its knowledge base, and by this, a consensus in the contents of a modern CG syllabus. Some efforts to define a comprehensive knowledge base for the computer graphics discipline have been

978-1-5386-2957-4/18/$31.00 ©2018 IEEE 17-20 April, 2018, Santa Cruz de Tenerife, Canary Islands, Spain
2018 IEEE Global Engineering Education Conference (EDUCON)

attempted in the recent years. For example, the *Curriculum Knowledge Base Group*, created by the *ACM SIGGRAPH Education Committee*, aimed "to provide a curricular structure and supporting materials that will aid instructors and institutions working to develop or enhance academic programs in computer graphics" [1].

As a result of the work of this group since 2001, it was generated a report in 2006 [2, 3] that proposed a knowledge base composed of sixteen areas. Each area, in turn, defined a set of more specific topics. This guideline was jointly developed by educators and practitioners from the United States, Europe, South America, and Japan. The original areas have evolved and regrouped into the seventeen areas shown in the current online version of this report [1]. The proposed knowledge-base areas are depicted in Fig. 1.

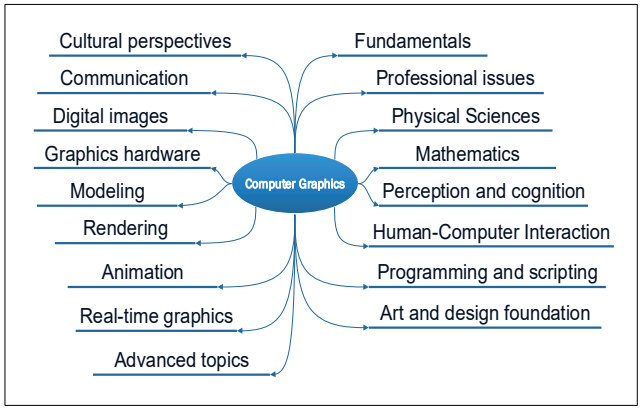

Fig. 1. *The seventeen areas proposed by the ACM SIGGRAPH as the computer graphics knowledge-base.*

B. The Modern Computer Graphics Curriculum

Computer Graphics is an undergraduate and graduate course mostly included in the Computer Science curriculum but also included in a diversity of academic programs around the world, such as Computer Engineering, Information Technology, Software Engineering, and Digital Media/Arts, just to name the most common. In the United States, over the last forty years, the ACM and the IEEE, among other organizations, have developed computing curricula guidelines for colleges and universities. The ACM has recommended CG as an undergraduate course for Computer Science majors since 1991.

From a historical perspective, according to [4], early CG courses from the 1970s through early 1980s were offered in just a few universities that could afford the expensive graphics hardware available at the time. CG curriculum in this era focused on low-level graphics hardware, basic rendering algorithms, introduction to 3D modeling, and interfacing graphics processors with mini-computers or mainframes.

In a second era, from the middle 1980s to early 1990s, with a much more affordable and somehow standardized graphics hardware, CG became a common course included in Computer Science degrees. Still, in that era, CG curriculum was focused on low-level algorithms, math, lighting and color principles, graphics hardware devices, and the use of primitive (and mostly unstandardized) APIs for graphics.

In the current era, since the late 1990s to present, CG syllabi have been designed around diverse APIs and have increased its presence in a lot more diverse academic programs than just Computer Science. This situation has happened due to the continuous development of faster, cheaper, and ubiquitous 3D graphics accelerators as well as the availability of robust and highly standardized graphics APIs, such as OpenGL [5], Direct3D [6], WebGL [7], or Java3D. Efforts in this era are focused on the selection of better software tools that are expected to ease the learning experience.

The earliest language selections to support a course in CG were based on using C/C++ to implement the core principles. Today, the most pervasive API to teach computer graphics is OpenGL. The utilization of this API includes its deprecated fixed-mode style [8-10], or the newer shader-based mode [11, 12], and WebGL [12] (the browser-interpreted version of OpenGL). Also, some non-OpenGL attempts have been applied in the classroom, such as Processing [13], Java3D [14], Microsoft XNA [15], or Maple [16].

Former core topics in the CG curriculum are less relevant today, or at least not relevant to all the academic programs offering CG. For example, the fact that today most of the low-level algorithms and the whole rendering process are fully integrated at the hardware level, including the support to the standard APIs, makes elective, from the CG curriculum point of view, the inclusion of certain topics or not. In modern times, CG syllabus is heavily influenced by the profile of the specific degrees and students where CG is offered. For example, Computer Science still can be heavily interested in math foundations, low-level rendering techniques, algorithms, and modern graphics accelerators architecture and optimizations. Software Engineering or Information Technology can be more interested in using higher-level programming frameworks for developing graphics application. Finally, Digital Media can be more interested in teaching CG principles through the use of state-of-the-art commercial graphics software for modeling, rendering, and animation.

C. Some studies on the CG syllabus

Some similarities and differences in the vision about the development of the CG syllabus can be found in the following three studies:

A first study [17], developed in the late 90's, was a survey among twenty-three universities in the United States regarding the current state of the CG syllabus at the time. It was found that topics such as viewing/camera transformations, hardware, lighting models, 3D transformations, user interaction, object representation, shading models, color models, curves, hidden-surface removal, and rasterization, were in the top of the most included. Also, as a result of that study, some insights were produced regarding the development of CG courses. These insights included the following, as stated by the authors: "Courses in CG should be inherently 3D", "the fundamental subject in CG is geometry (expressed in computational terms)", "CG needs to study light and surfaces", "algorithms in CG

must be considered not only for time and memory usage but also for their visual effects", "CG courses should be built upon a high-level graphics API", and finally, "CG courses should include interactive projects and cover event-driven programming".

A second study [18], developed in 2005, proposed a CG syllabus based on 2D and Image Processing (IP) topics instead of the traditional 3D-oriented CG course. In this study, there were analyzed more than seventy academic programs and courses in CG, IP, and HCI, as well as curricula in Computer Engineering, Computer Science, Information Technology, and Software Engineering. Although this proposal is somehow unconventional due its opposite point of view to the widely accepted 3D-centric CG syllabus, it intended to design courses that balanced the CG, IP, and HCI areas through the study of the many topics that these areas have in common and with an additional emphasis in 2D contents.

In a third study [19], developed in 2016, it was analyzed the CG curriculum from a new perspective that took jointly into account Computer Science (CS), Computer Technology (CT), and Computer Art (CA). In this study, there were selected, among nearly 400 registered programs, the most influential CG curricula in the United States to analyze their current trends. It was found that topics such as math, algorithms, color, rendering, lighting, illumination, pipelines, hardware, rasterization, curves, and programming, were among the primary themes included in the textbooks utilized in all the surveyed academic programs. This situation was not much different from the found one decade earlier. However, this study delineated a consensus among experts from industry and academia in the areas of CS, CT, and CA, that brought new insight into the current trends in the CG syllabus.

It was suggested that from the seventeen areas initially proposed in [3], they could be probably reduced to nine in a contemporary CG curriculum. These areas were: Art and design, animation, digital imaging, physics, visual perception, visual communication, mathematics, cognitive sciences, and computer programming. This study suggested that "CG curricula must emphasize an interdisciplinary approach, and formulate outcome-based programs that connect scientific, technocratic, and artistic principles together to meet the growing needs of industry".

In addition to these studies, according to the most recent guidelines of the ACM and the IEEE for the Computer Science curricula in 2013 [20], the CG syllabus should encompass several interrelated topics such as fundamental concepts, basic and advanced rendering, geometry modeling, animation, and visualization. It is recognized that traditional graphics at the undergraduate level focuses on rendering, linear algebra, and phenomenological approaches, while more recent trends include physics, numerical integration, scalability, and special-purpose hardware. It is assumed that nearly every undergraduate course in CG will cover a basic rendering and that this topic is essential for any further study in this field. According to the ACM's analysis, some of the most common learning outcomes for students in CG are 2D/3D coordinate transformations and a basic knowledge of the 3D graphics pipeline.

III. General Approaches to Teaching Computer Graphics

General approaches to teaching CG have been envisioned through the evolving history of this discipline. These approaches have been the result of both, experiences in the classroom and contents structure addressed by textbooks in the field. Some of these approaches are described in the rest of this section, and they are summarized in Fig. 2.

One of the earliest classifications of these approaches was provided by E. Angel in 1997 [8]. Angel defined three fundamental approaches: The survey, the bottom-up, and the top-down, which are described next.

The survey approach usually just provides a general overview of the discipline with little or no programming involved. This approach was one of the first utilized at the beginning of the CG field, when graphics hardware and software were neither affordable nor available for most universities and topics in the course were covered mostly in a descriptive or theoretical point of view without the means to test or implement algorithms and graphics applications.

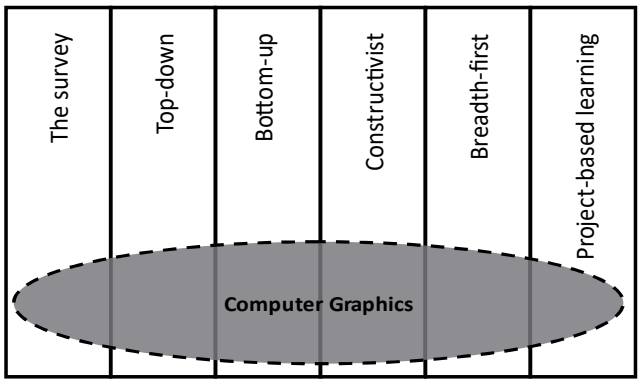

Fig. 2. *Different approaches proposed for teaching CG.*

The bottom-up approach focuses in studying in detail a collection of the most basic algorithms, mathematical methods, and other foundational elements, and then go gradually up into higher layers of integration, abstraction, and applications of the graphics technology. For example, in this approach, students could start by learning the Bresenham's algorithm to draw lines and other basic algorithms to draw simple two-dimensional curves, as well as algorithms for clipping and rasterization. This approach is well-aligned with the most classical textbooks in CG, such as Foley [21], Hearn [22], and Hill [23], and courses based purely in a bottom-up strategy were more frequent in the earlier stages of CG as an emerging discipline.

The top-down is another common approach to teach CG, which starts by analyzing and implementing graphics applications first and then moves gradually to lower levels of details [24], [25], and [12]. This approach intends to provide students with a more holistic understanding of the CG field. An analogy between automobiles and CG given by Angel [8] to describe the three mentioned approaches states that: "you don't need to know how a car works (the bottom-up approach) to drive it (the top-down approach), or even you don't need to

drive the car and hire a chauffeur to drive you (the survey approach)". Probably, the top-down approach is one of the most utilized in the recent years. When this approach was first proposed in the earliest CG years, it was mainly a recommendation of starting the course directly with 3D, avoiding a lengthy introduction of low-level 2D principles and algorithms. This recommendation was possible thanks to the availability of graphics APIs such as OpenGL.

Today, the top-down approach can be extended or diversified to cover a wider range of academic programs and student profiles. In fact, traditional CG courses found in Computer Science could still use the bottom-up approach as probably those students will be the tomorrow scientists and professionals that will advance the state-of-the-art in graphics technology. CG courses offered in Software Engineering or Information Technology could be more benefited with a top-down approach because students will learn how to program graphics applications at a higher abstraction layer and without knowing the hardware implementation details. Also, a Digital Media student could be benefited from a top-down approach. If the CG course starts using a high-end graphics application, this would allow students to overview, in a very attractive manner, the current state-of-the-art of graphics technology to model, animate, and render scenes. Then, gradually, the course could go into the details and principles of particular components.

The constructivist approach [26] promotes that instead of providing comprehensive lectures for each topic of the course, the teacher's role is to guide the students in constructing a conceptual image of how a component works given that it exhibits certain capabilities. This is favored by environments where students can visualize, test, and experiment with a diversity of didactic material according to the CG contents.

The breadth-first approach [27] is similar to the top-down but with a special emphasis in that the initial holistic view of the discipline be suitable, and the same, for students belonging to different academic programs, such as Computer Science, Software Engineering or Digital Media. It means that the "breadth" part of the course is highly interdisciplinary while the "depth" part (studying topics at a detailed level) is highly dependent on the student profile.

The Project-Based Learning (PBL) approach [28] is based on an educational strategy that promotes solving a problem or project in a student working group. In [28], this approach was tested in projects that focused on the development of a graphical environment based on OpenGL and C++ for visualization and handling of different scenarios. Students could start with a basic application skeleton and then completed the project by adding the missing functionalities according to a given task. As mentioned in that study, PBL promotes, in general, autonomy in the learning, teamwork, self-assessment, argumentation and critical reasoning, and integration of knowledge.

IV. Digital Games as a Learning Strategy in Higher Education

In the recent years, the application of digital games for learning has attracted the attention of researchers belonging to the educational and the computing fields. De Freitas [29] defined digital learning games as "applications using the characteristics of video and computer games to create engaging and immersive learning experiences for delivering specified learning goals, outcomes, and experiences". While this learning strategy has been present for a long time in the elementary and middle education, it is increasingly becoming more common in the higher education [30]. Although there is still a debate about the real effectiveness of games for learning [31], many studies endorse this practice as pedagogically well-founded [32] and useful. The digital games for learning, or serious games, have also been applied for a long time outside the educational field, for example in the military or public and private organizations. In these cases, their application is more commonly referred as "training" instead of "learning".

From the point of view of games-for-learning design and development, it is a huge interdisciplinary field, ranging from software engineering, artificial intelligence, psychology, pedagogy, physics, and art, just to name the more involved areas. Also, due to the great variety of game types (e.g., adventure, puzzle, strategy, sports, fighting, platform, role-play, shooter) and purposes, it is recognized that neither a universal game template nor a universal architecture does exist yet [33]. Architectures are also influenced by platforms (e.g., mobile, web, desktop, console, virtual reality) or schemes such as local or distributed, single or multiplayer, interoperability or integration requirements (e.g., embedded in learning management systems). According to [33], current research areas in games for learning involve at least the following: Efficient development processes, architecture blueprints, interoperability and data exchange, emerging user interfaces, arising gaming technologies, domain-specific game engines, and model-driven development.

Despite the current degree of maturity of the field of digital games for learning, it can still be seen as "in development". There are a variety of examples of successful applications in general domains as well as in highly-specialized ones. Two comprehensive surveys [34, 35] analyzed a total of 272 works. [34] comprised works from 2004 to 2009, and [35] from 2010 to 2016. Every surveyed work reported empirical evidence of the benefits of using computer games with respect to learning, among other effects. Diverse disciplines such as business, computing, engineering, health, history, language, mathematics, science, social issues, and military, were addressed by those games. This situation provides a clue about envisioning a future with an expanded use of games in the higher education.

The following studies are selected examples of applications of digital games in computing education, especially those designed to help students learn programming as well as software engineering.

In [36], a suite of four programs was developed to help learn Java for students with no programming experience. This suite was composed of the following applications: A first game, which helped the student learn typing Java source code. A second, multiple-choice game, based on the Xbox360 controller that helped the student to understand the Java programming structure. A third, two-players competition game that aimed to complete missing words in a Java source code.

Finally, a game that incorporated scripted commands to perform actions over game characters, in a style that reminded the utilized in visual programming.

In [37], it was developed a framework aimed to improve the development of computational thinking in students. This ability is a set of reasoning skills that allow people interact and think through the language of computation, which in turn is a key ability to be a programmer.

In [38], it was developed a set of games that consisted of a set of exercises arranged into categories freely chosen by students according to their particular preferences. These games were built on top of a free Java game development environment. The main use of this tool was to ease the learning of topics covered in an introductory programming course, such as basic programming constructs, structured instructions, subroutines, recursion, arrays, files, and complex data types.

In another study [39], it was implemented an Android-based game focused on help children, above eight years old, to learn the principles of object-oriented programming together with the principles of software engineering. Other work [40] built a pedagogical game, called Age of Computers, to be used in the course Computer Fundamentals as a replacement of weekly paper exercises. A mobile game was developed in [41] for learning the ActionScript language in a course using PBL as a learning strategy. In [42], a virtual game simulating a board game named SCRUMI was implemented for teaching the SCRUM framework for software projects management.

A comprehensive survey of games for learning in the area of software engineering was conducted in [43], where a total of 106 studies were found and classified as belonging to Game-Based Learning (GBL), Game-Development-Based Learning (GDBL), or Gamification.

With the initiatives described in this section, we are prompted to believe that serious games can be successfully applied in many areas of computing education, and probably we will continue seeing this tendency in the near future.

V. The Use of Digital Games and Interactive Simulators to Teaching Computer Graphics

In the particular case of using interactive simulators and games for teaching CG, some efforts have taken place in the recent years as games have been gaining popularity as a learning strategy in the higher education. In the following, a survey of these efforts is presented. This includes those works, classified as GDBL, which involve students in developing their own games as a way of learning CG topics. Also, those works that proposed interactive simulators, although not exactly games. Finally, those works, classified as GBL, which explicitly utilized digital games (not developed by the students themselves) for learning CG fundamentals. These three different learning strategies are represented in Fig. 3.

The methodology consisted of reviewing the literature for a broad span going from the late 90s to the present. This span was chosen in order to encompass all the considered modern era of CG, seen as a discipline and as a curriculum. The databases searched were the following: IEEE Xplore, ACM Digital Library, Elsevier's ScienceDirect, SpringerLink, Web of Science, and Google Scholar. The keywords searched were a combination of *computer graphics*, *education*, *video games*, *serious games*, *digital games*, *artifacts*, and *learning*.

Although diverse studies arose when searching, only those belonging to the CG topics domain and under the categories of "development of games", "interactive didactic artifacts", and "playing video games for learning", were included. Many studies that proposed the use of libraries such as OpenGL, Direct 3D, Java3D, and others, to be used in class as programming assignments or to teach CG topics with prototypes or frameworks provided by lecturer were not included. This decision was made to effectively separate all works that didn't propose solutions based on games or solutions without an important degree of interactivity.

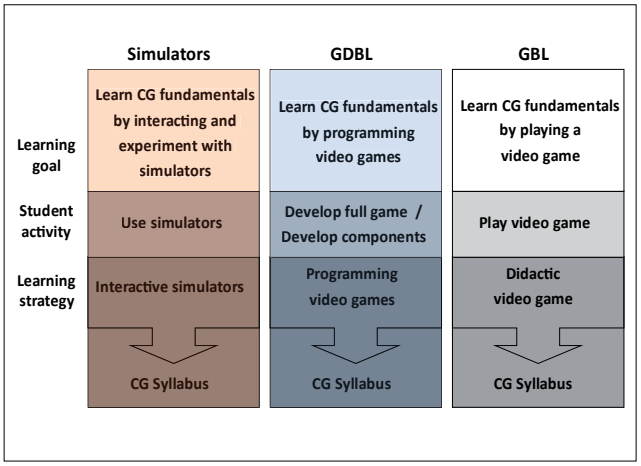

Fig. 3. *Three surveyed strategies for teaching CG topics*.

A. GDBL approaches

A top-down approach to structure an introductory CG course and the use of Java3D as the programming API to develop a game is the strategy described in [14]. Researchers implemented an interactive tool to learn and practice Java3D (named Interlab3D) as well as a didactic multiuser game engine written in Java3D (named enJine) to ease the development of a 3D game as a semester project. While the course contents were not oriented to games development, authors found attractive this alternative for teaching introductory CG. In this case, the whole syllabus was designed to match the theory with the practice. A video game was incrementally developed at the same pace as the topics were studied in order to have a game completion at the end of the course. Three assignments corresponded to the practice part of the course, starting with the modeling of the static components of the game, then building the dynamic components and finally working on the game completion.

Another research where game design is seen as a useful strategy for teaching CG is found in [44]. In this study, researchers implemented a platform in C++, named GameX, built on top of OpenGL and DirectX. A novel approach, in this case, was that the platform architecture allowed a varied degree of program abstraction. For example, students could learn by

calling low-level functions to draw primitives in the basic level of abstraction or could apply 3D modeling concepts in the next level. At the highest level, students could develop a game in a collaborative and interdisciplinary manner. Moreover, advanced students were able to learn through the platform source code itself. The platform architecture supported introductory CG courses requiring simple projects and assignments by using its basic interface. Also, it supported intermediate courses that required team projects by using its intermediate interface. Finally, it supported advanced courses, like those oriented to games development that required collaborative and highly interdisciplinary projects, by using its high-level interface.

A similar approach is found in [45], where a collaborative and competitive development of a multiplayer racing game was implemented in an introductory CG course with the help of a proprietary framework named eNVyMyCar. Authors organized all the practice exercises around this framework, which has a single-server and multiple-client architecture. The world represented consisted in a static part that included all the fixed elements of the scene, and a dynamic part that essentially represented the state of the race, like cars position, orientation, and speed. The framework was designed in such a way that students did not have to know networking or the game physics details. Students just were concentrated on interactively describing and rendering scenes using simple C++ classes. When commands from the clients to the server were sent, the system broadcast them, together with the system state, to all the clients. Additionally, clients could send snapshots of the rendering provided by the player in order to other students could appreciate the visual results and added with this a competition factor regarding who achieved the more refined scene or effects.

In [46], some elements of games programming were incorporated in an introductory CG course (designed with a top-down approach) to help students to understand the core learning objectives. In this case, no full-game was developed by students, but game design principles were incorporated into regular programming assignments in order to engage and motivate students to pursue an active learning.

Authors in that study classified courses that involve games and programming in three categories: Games development classes, games programming classes, and games development clients. The first category included those courses designed specifically to develop new games as an end product, so they were concerned with all aspects of real games production. The second category included classes that study technical aspects of games programming, such as loops, path planning, and terrains, just to name a few, and usually, it was not required to build an end product but individual components. Authors positioned their work in the third category, the games development client, which are courses that creatively integrated games into their syllabus, just as programming assignments, or to teach abstract concepts, or as an example of application areas.

Game elements were incorporated in such a way that they did not compromise the syllabus schedule, for example not dedicating lecture time to cover topics specific to games programming. In order to achieve this, authors first identified the contents shared by both, the introductory CG and the Games Programming domains. Also, they identified the contents that are seen in-depth in CG courses only (e.g., transformations, modeling, viewing, projections, illumination, rendering buffers, textures). Finally, they identified the essential contents suitable for games programming only (e.g., Newtonian physics, scripting, sprites animation, resources management, artificial intelligence, audio programming, file formats).

A fundamental strategy followed in that study was the concept of a CDA (Concept Demonstration Application), which were custom-built, event-driven interactive programs that demonstrate specific foundational CG concepts. As a result of that project, more than one hundred of CDAs were implemented to support topics included in the CG syllabus as well as in the programming assignments involving games programming elements.

B. Interactive simulators

Some of the earliest works proposing the use of interactive simulators to teach CG were based on Java *applets* and emerged in the late 1990s. This is the case of the *applets* developed by Patrick Min in 1996 [47], which covered CG topics such as Bézier curves, 2D transformations, 3D viewing, clipping, and lighting. According to [48], these *applets* were used successfully in the classroom, mainly to complement class explanations as students could visually review the results after lecturer interacted with these *applets* when covering specific topics related to the textbook content.

A similar study, in 1998, implementing Java *applets*, is documented in [49], where a web-based course was supported by lessons, examples, programming exercises, and documentation. The role of the Java *applets*, in this case, was both, to be an interactive tool to learn CG concepts and to be a programming framework where students extended the functionality of existent *applets* or develop new ones as class assignments.

Still, another study in 1998 using Java applets and Virtual Reality Modeling Language (VRML) is presented in [50], where a framework for the development of educational applications was proposed as a means to create interactive 3D models to teach and clarify the working of certain graphics algorithms.

A similar approach was followed in [51], in 2002, where more than fifty interactive applications written as Java *applets* were developed to help students learn specific topics in an online CG course. These applications covered CG contents such as 2D/3D geometrical transformations, digital image processing, z-buffer algorithm, smooth shading, lighting models, materials, texture-mapping, rendering, animation, as well as others. All these contents were grouped in eight areas or chapters, according to the online course organization, enumerated as follows: Rendering, image processing, digital image representations, 2D image generation, 3D coordinate transformations, 3D modeling, computer animation, and digital image and modeling. These Java *applets* were not games in the sense of having fun with them but had interactive elements to engage students in learning. Also, a feedback mechanism to

encourage students to study harder and keep records of their learning status was also implemented. This feature was important considering that this proposal was aimed to support a distance-learning course. Students were going to study and learn through the course online lessons and materials, as well as interacting with the Java applets to learn all the CG contents. An important benefit of the Java applets approach is that students needed only an Internet browser to use them, no other software installation was needed.

Another study that aimed to develop an environment that supported interactive applications to teach CG, as well as virtual reality concepts, is found in [52]. In this work, it was developed a didactic framework, named Mental Vision that could be used as a graphics engine in practice sessions or student projects. The framework was developed in C++ and OpenGL. A set of applications featuring real-time and dynamic demonstrations of the course contents was also developed. Every application presented a single topic through an intuitive interface that allowed a dynamic modification of the parameters of the exposed algorithms. The same framework could be utilized to rendering and handling the 3D content in projects that involved advanced virtual reality interfaces, such as data-gloves, head-mounted displays, motion capture, and haptic workstations. A later update of this same work [53] reported an expanded multiplatform support for heterogeneous devices, such as mobile devices as well as Cave Automatic Virtual Environments (CAVEs).

C. GBL approaches

In this category, the development of a set of web-based games was proposed in [54] as an aid for students enrolled in Engineering Graphics. It is an introductory engineering course, included in many engineering curricula, that is related to visualizing 3D environments, constructing auxiliary views, and dimensioning and tolerancing. In that work, each game was constructed as an individual web page using JavaScript code to create the logic of a variety of puzzles. The main concepts that puzzles helped students learn were: An introduction to graphics, multi-view drawings and pictorials, constructing auxiliary views, manipulation of parts and a reference coordinate system in a 3D space, and dimensioning and tolerancing. While this approach was not specially intended to teach CG but Engineering Graphics, the improving on visualization skills, through the use of the games, might have benefits in the learning and performance of students in CG as well.

Work [55], presented in 2016, proposed an environment named: Gamified Training Environment for Affine Transformations (GEtiT), that used a 3D space to visualize the effects of translations, rotations, scaling, reflections, and shearing. The game required mastering 3D transformations to modify objects to achieve a given goal position or shape. The learning tasks were designed according to the level design, the selection of a type of transformation, and the victory or goal conditions. Transformations were represented as cards, and immediate visual feedback was generated by the game engine. Authors of this work explain the game goal as follows: "GEtiT players start trapped in a sealed room from which they can only escape when they open the portal. In order to do so, they need to transform the object using their transformation types in such a way that it matches the victory conditions. However, they have to pay attention to the environment as the object cannot translate through obstacles that are placed at a particular level. Once the object matches the victory conditions, the portal gets activated, and the players can proceed to the next level". This game was tested in the classroom and obtained a similar learning outcome than traditional methods. However, students reported a higher enjoyment during the learning process.

VI. DISCUSSION

After reviewing different efforts found in the literature that aimed to improve the teaching of CG through interactive simulators and games, it is worth summarizing some findings.

There were eleven studies found in the literature in total, spanning from the late 90's to 2016. Four studies belonged to the category of GDBL, five were in the category of interactive simulators, and two were about GBL, that is, playing video games as the learning strategy. These proportions are shown in Fig. 4.

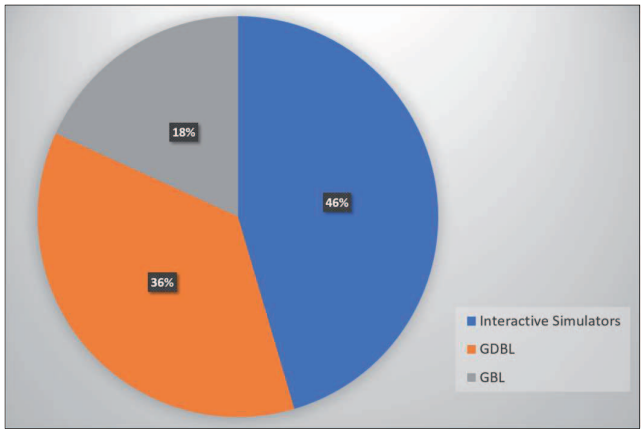

Fig. 4. *The proportion of studies belonging to each category.*

The oldest efforts were those that implemented interactive simulators. All of them, except one, were presented in the late 90's and proposed Java *applets* as the technology chosen to create the simulators. It seems that, at the time, the newly extended capabilities to run stand-alone applications powered by Java *applets* and generate 3D graphics directly through the web browsers, without any installation of libraries or local applications, greatly favored the emergence of this kind of proposals. Also, they were well suited to support the first e-learning courses in CG. Only one of the surveyed studies in this category was proposed in the first decade of the 2000s, and in this case, a lot of new tools and technology, including virtual reality and supporting heterogeneous devices was envisioned.

The four studies in the category of GDBL were proposed in the first decade of the 2000s. They utilized the prevalent graphics libraries such as OpenGL, Direct3D, and Java3D together with programming languages such as C++ or Java to create frameworks that simplified the student coding for games development. All the surveyed studies in this category were chosen to fit the case of proposing the development of games in

the context of a CG course, not in a Game Development course.

Considering the broad span of the last twenty-two years, which is almost the same span that the considered modern CG field (and curriculum) has, this survey accounted for a non-homogeneous distribution of the three categories of analyzed studies. For example, as Fig. 5 depicts, in the first interval, from 1995 to 2000, there were only proposals of simulators present. In the second interval, from 2001 to 2006, there was a presence of the three categories. It was the only interval where this situation happened. The third interval, from 2007 to 2012, only had proposals of GDBL. It can be seen that almost the same number of proposals regarding simulators is found in the two consecutive intervals from 1995 to 2006. A similar situation with the GDBL category occurred in the two consecutive intervals from 2001 to 2012. Finally, in the fourth interval, only the category of GBL is present, although it consists of just one study.

Only two studies were found in the category of GBL for learning CG. One of them proposed in 2001, and the other in 2016. It is worth noting that the study of 2001 is not directly oriented to CG but to an Engineering Graphics course, that although having some topics in common, was not entirely focused on the CG syllabus. Also, the second study, although directly oriented to CG, it was only designed to cover the "affine transformations" topic, and nothing else.

From data shown in Fig. 5 is evident that the category where fewer proposals exist is the GBL. Just two studies in the last twenty-two years propose that students in introductory courses of CG play video games to learn the course contents. This can be seen as strange if it is considered the high popularity that didactic games have in other domains.

Fig. 5. *Distribution of studies in the last twenty-two years.*

VII. CONCLUSIONS

While CG teaching has some decades of refinements, syllabi evolution, and adaptations to the continuous advances in graphics technology, there is always a need of creative proposals to present learning materials to students and engage them in learning.

There is evidence that the use of digital games for learning in higher education has been attempted in diverse domains such as science, engineering, and computing courses, as an effort to motivate students to improve their learning. However, the full potential benefit of GBL seems unaccomplished in the specific case of teaching computer graphics, with very few efforts found in the literature aimed to incorporate this approach. Studies closer to this vision were those that implemented a collection of simulators to practice CG concepts. Nevertheless, simulators have just a didactic part but miss a fun part that only a game can provide. The GBL category, as classified in this study, accounted for just two proposals in a huge span of twenty-two years. With this study, authors aim to increase awareness about the detected necessity of diversifying the current offering of digital games in this particular area, which in turn can lead to improvements on how CG is taught in the classroom.

REFERENCES

[1] T. Alley.,"Computer Graphics Knowledge Base Report", 2006. [Online]. Available:https://education.siggraph.org/resources/knowledge-base/report. [Accessed: 10-Nov-2017]

[2] T. Alley *et al.*, "knowledge base for the emerging discipline of computer graphics," presented at the ACM SIGGRAPH 2006 Educators program, Boston, Massachusetts, 2006.

[3] C. Laxer and J. Orr, "A Knowledge Base for the Emerging Discipline of Computer Graphics": Report of the SIGGRAPH Education Committee Curriculum Working Group," in Workshop on Computer Graphics Education (CGE'06), 2006.

[4] L. E. Hitchner and H. A. Sowizral, "Adapting computer graphics curricula to changes in graphics," *Computers & Graphics,* vol. 24, no. 2, pp. 283-288, 2000.

[5] Khronos Group, "OpenGL", 2017. [Online]. Available: https://www.opengl.org/. [Accessed: 10-Nov-2017]

[6] Microsoft, "Direct3D", 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/hh309466. [Accessed: 10-Nov-2017]

[7] Khronos Group, "WebGL", 2017. [Online]. Available: https://www.khronos.org/webgl/. [Accessed: 10-Nov-2017]

[8] E. Angel, "Teaching a three-dimensional computer graphics class using OpenGL," *ACM SIGGRAPH Computer Graphics,* vol. 31, no. 3, pp. 54-55, 1997.

[9] S. Cunningham, "Re-inventing the introductory computer graphics course: providing tools for a wider audience," *Computers & Graphics,* vol. 24, no. 2, pp. 293-296, 2000.

[10] S. Cunningham, "Powers of 10: the case for changing the first course in computer graphics," *ACM SIGCSE Bulletin,* vol. 32, no. 1, pp. 46-49, 2000.

[11] J. O. Talton and D. Fitzpatrick, "Teaching graphics with the OpenGL shading language," *ACM SIGCSE Bulletin,* vol. 39, no. 1, pp. 259-263, 2007.

[12] E. Angel and D. Shreiner, Interactive computer graphics : a top-down approach with WebGL, 7th edition. ed. Boston: Pearson, 2015.

[13] J. Linares-Pellicer, P. Micó, J. Esparza-Peidro, and E. Carrasquer-Moya, "Computer Graphics: From Desktop to Mobile and Web," *IEEE Computer Graphics and Applications,* vol. 31, no. 4, pp. 94-96, c3, 2011.

[14] R. Tori, J. L. Bernardes Jr, and R. Nakamura, "Teaching introductory computer graphics using java 3D, games and customized software: a Brazilian experience," in ACM SIGGRAPH 2006 Educators program, 2006, p. 12: ACM.

[15] R. P. Mihail, J. Goldsmith, N. Jacobs, and J. W. Jaromczyk, "Teaching graphics for games using Microsoft XNA," in *Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious*

Games (CGAMES), 2013 18th International Conference on, 2013, pp. 36-40: IEEE.

[16] K. Sridharan, "Teaching computer graphics and robotics using symbolic computation software," *Computer Applications in Engineering Education,* vol. 8, no. 1, pp. 18-30, 2000.

[17] R. Wolfe, "Bringing the introductory computer graphics course into the 21st century," Computers & Graphics, vol. 24, no. 1, pp. 151-155, 2000.

[18] E. Paquette, "Computer Graphics education in different curricula: analysis and proposal for courses," *Computers & Graphics,* vol. 29, no. 2, pp. 245-255, 2005/04/01/ 2005.

[19] M. A. Roller, "A Consensus on the Definition and Knowledge Base for Computer Graphics," Ph. D. Dissertation, Purdue University, USA, 2016.

[20] ACM/IEEE, ACM, Ed. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. New York, NY, USA: ACM, 2013, p. 518.

[21] J. D. Foley, Introduction to computer graphics. Reading, Mass.: Addison-Wesley, 1994.

[22] D. Hearn and M. P. Baker, Computer graphics, C version, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1997.

[23] F. S. Hill, Computer graphics : using OpenGL, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2001.

[24] E. Angel, Interactive computer graphics : a top-down approach with OpenGL, 2nd ed. Reading, Mass.: Addison-Wesley, 2000.

[25] K. Sung and P. Shirley, "A top-down approach to teaching introductory computer graphics," Computers & Graphics, vol. 28, no. 3, pp. 383-391, 6// 2004.

[26] G. Taxén, "Teaching computer graphics constructively," *Computers & Graphics,* vol. 28, no. 3, pp. 393-399, 2004.

[27] G. Domik and F. Goetz, "A breadth-first approach for teaching computer graphics," in EG 2006, 2006.

[28] E. Martí, D. Gil, and C. Julià, "A PBL Experience in the Teaching of Computer Graphics," Computer Graphics Forum, vol. 25, no. 1, pp. 95-103, 2006.

[29] S. de Freitas and M. Oliver, "How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?," *Computers & Education,* vol. 46, no. 3, pp. 249-264, 4// 2006.

[30] N. Whitton, Learning with digital games: a practical guide to engaging students in higher education, 1 ed. (Open and Flexible Learning Series). New York: Routledge, 2009.

[31] P. Backlund and M. Hendrix, "Educational games - Are they worth the effort? A literature survey of the effectiveness of serious games," in Games and Virtual Worlds for Serious Applications (VS-GAMES), 2013 5th International Conference on, 2013, pp. 1-8.

[32] M. Kebritchi and A. Hirumi, "Examining the pedagogical foundations of modern educational computer games," Computers & Education, vol. 51, no. 4, pp. 1729-1743, 2008.

[33] H. Söbke and A. Streicher, "Serious Games Architectures and Engines," in *Entertainment Computing and Serious Games*: Springer, 2016, pp. 148-173.

[34] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, and J. M. Boyle, "A systematic literature review of empirical evidence on computer games and serious games," *Computers & Education,* vol. 59, no. 2, pp. 661-686.

[35] E. A. Boyle *et al.*, "An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games," *Computers & Education,* vol. 94, pp. 178-192.

[36] T. Mitamura, Y. Suzuki, and T. Oohori, "Serious games for learning programming languages," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 2012, pp. 1812-1817: IEEE.

[37] C. Kazimoglu, M. Kiernan, L. Bacon, and L. Mackinnon, "A serious game for developing computational thinking and learning introductory computer programming," *Procedia-Social and Behavioral Sciences,* vol. 47, pp. 1991-1999, 2012.

[38] R. Hijon-Neira, Á. Velázquez-Iturbide, C. Pizarro-Romero, and L. Carriço, "Serious games for motivating into programming," in *Frontiers in Education Conference (FIE), 2014 IEEE*, 2014, pp. 1-8: IEEE.

[39] S. Ramírez-Rosales, S. Vázquez-Reyes, J. L. Villa-Cisneros, and M. D. León-Sigg, "A Serious Game to Promote Object Oriented Programming and Software Engineering Basic Concepts Learning," in *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2016, pp. 97-103.

[40] G. Sindre, L. Natvig, and M. Jahre, "Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals," *IEEE Transactions on Education,* vol. 52, no. 1, pp. 10-18, 2009.

[41] N. F. Rozali and N. M. Zaid, "Code puzzle: ActionScript 2.0 learning application based on problem based learning approach," in 2017 6th ICT International Student Project Conference (ICT-ISPC), 2017.

[42] A. D. D. Souza, R. D. Seabra, J. M. Ribeiro, and L. E. D. S. Rodrigues, "SCRUMI: A Board Serious Virtual Game for Teaching the SCRUM Framework," in 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 319-321.

[43] M. R. D. A. Souza, L. F. Veado, R. T. Moreira, E. M. L. Figueiredo, and H. A. X. Costa, "Games for learning: bridging game-related education methods to software engineering knowledge areas," in 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), 2017, pp. 170-179.

[44] R. C. Hoetzlein and D. I. Schwartz, "GameX: a platform for incremental instruction in computer graphics and game design," presented at the ACM SIGGRAPH 2005 Educators program, Los Angeles, California, 2005.

[45] F. Ganovelli and M. Corsini, "eNVyMyCar: A Multiplayer Car Racing Game for Teaching Computer Graphics," Computer Graphics Forum, vol. 28, no. 8, pp. 2025-2032, 2009.

[46] K. Sung, P. Shirley, and B. R. Rosenberg, "Experiencing aspects of games programming in an introductory computer graphics class," in ACM SIGCSE Bulletin, 2007, vol. 39, no. 1, pp. 249-253: ACM.

[47] P. Min, "Computer Graphics Applets", 1996. [Online]. Available: http://min.nl/cs426/applets.html. [Accessed: 26-Nov-2017]

[48] K. Karpouzis and S. Kollias, "The rendering pipeline in the classroom: a diversified approach," presented at the Proceedings of the 6th annual conference on the teaching of computing, Dublin City Univ., Ireland, 1998.

[49] R. Klein, F. Hanisch, W. Stra, "Web-based teaching of computer graphics: concepts and realization of an interactive online course", presented at the ACM SIGGRAPH 98 Conference abstracts and applications, Orlando, Florida, USA, 1998.

[50] H. Baerten and F. Van Reeth, "Using VRML and JAVA to visualize 3D algorithms in computer graphics education," *Computer Networks and ISDN Systems,* vol. 30, no. 20, pp. 1833-1839, 1998.

[51] T. Nishita et al., "Development of a Web Based Training system and Courseware for Advanced Computer Graphics Courses Enhanced by Interactive Java Applets," in Proceedings of International Conference on Geometry and Graphics, 2002, vol. 2, pp. 123-128.

[52] A. Peternier, D. Thalmann, and F. Vexo, "Mental vision: a computer graphics teaching platform," in *International Conference on Technologies for E-Learning and Digital Entertainment*, 2006, pp. 223-232: Springer.

[53] A. Peternier, F. Vexo, and D. Thalmann, "The mental vision framework-a platform for teaching, practicing and researching with computer graphics and virtual reality," *Transactions on edutainment I,* pp. 242-260, 2008.

[54] S. W. Crown, Improving Visualization Skills of Engineering Graphics Students Using Simple JavaScript Web Based Games. Journal of Engineering Education, 90: 347–355, 2001.

[55] S. Oberd and M. E. Latoschik, "Interactive gamified 3D-training of affine transformations," presented at the Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology, Munich, Germany, 2016.