GameX: A Platform for Incremental Instruction in Computer Graphics and Game Design

Rama C. Hoetzlein
Media Arts & Technology Program
University of California Santa Barbara
e-mail: rch@umail.ucsb.edu

David I. Schwartz
Department of Computer Science
Cornell University
e-mail: dis@cs.cornell.edu

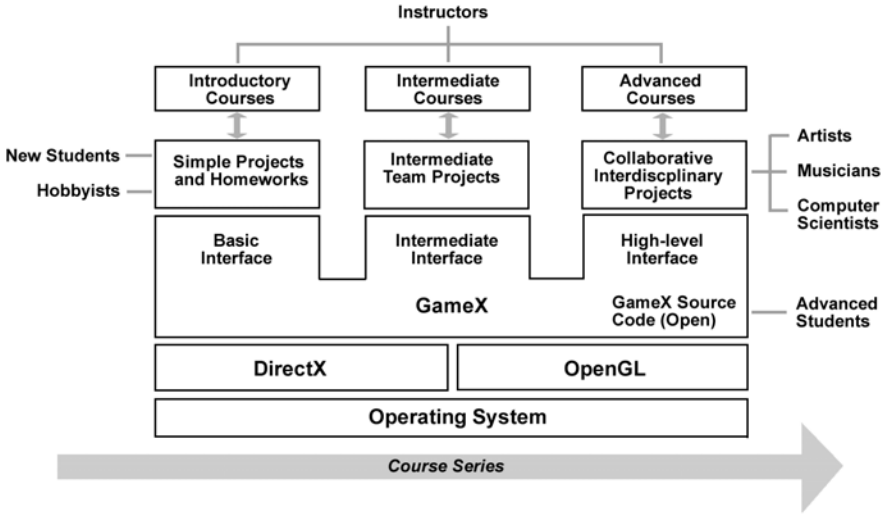

Figure 1. Platform design for a graphics engine supporting incremental instruction.

Abstract

Recent trends have resulted in an increased focus on game design as a topic for teaching in higher education [Deutsch 2002]. Although many game engines currently exist, few of these were designed with educational goals in mind. We distinguish between industry-oriented engines and *instructional game engines* designed to teach a range of concepts. The features needed to teach game development to college undergraduates in engineering and the humanities are explored. Specifically, we develop a platform that supports incremental education in game design. GameX, an open source instructional game engine, was developed with this approach in mind and was used to initiate the Game Design Initiative at Cornell University (GDIAC).

1 Introduction

The teaching of game design in higher education is an area of on-going growth. As with any new development, game design instruction has several points of origin. Institutes seeking to educate the next generation of game industry designers have been in existence for several years. Digipen began offering courses in 1994 and the Entertainment Technology Center (ETC) in 1999.

At established colleges and universities, the acceptance of game design as a teaching topic is improving. Yu [2002] summarizes these challenges as well as defining the main arguments in their favor. She states: "The main supporting argument for a game programming course is the importance of teaching collaboration and process management. The game production process has become so complex that talents have to be drawn from many different fields, including people trained in computer science and people who are not." Early course offerings in game design include those by Ian Parberry at the University of North Texas since 1993 [Parberry 2005], John Laird at the University of Michigan, Ken Forbus at Northwestern University, Jessica Hodgins at the George Institute of Technology, and Randolph Jones at Colby College [Jones 2000].

In most instances, educators have had to rely on existing tools to teach new courses. The development of game engines specifically for education is relatively new, but also has precedents. *Game Maker* by Mark Overmars at Utrecht University in the Netherlands was first released in 1999 [Overmars 2004] while 3D Game Studio was first released in 2000 in Germany [1]. Alice, a general tool for teaching graphics programming concepts was developed by the Entertainment Technology Center at Carnegie Mellon [Conway 2000]. The platform discussed here, GameX, was released for use by the Game Design Initiative at Cornell University in 2001. In all examples the goal has been to develop a platform that makes it easier for students to learn a variety of skills which may include game development.

A game design curriculum in higher education should provide a broad overview of topics in computer science and the humanities [Hoetzlein and Schwartz 2004]. We seek to provide a single platform that supports a series of courses in game design on a variety of instructional levels. While introductory courses might consist of fundamental concepts that cover graphics, sound,

[1] Retrieved from Conitec Datensysteme GmbH company website, Germany: http://www.conitec.net/a4news.htm

artificial intelligence and networking, the challenge is to design an instructional engine that can also be used to support interdisciplinary collaboration and advanced topics.

Our goal is to develop a general instructional platform in parallel with a strategy for education that allows the same platform to be used to teach a series of courses at different levels (Figure 1).

2 Current Technology

Existing graphics engines present both positive and negative aspects as instructional tools. *Low-level engines*, such as DirectX and OpenGL, have become industry standards for real-time graphics due to their performance and flexibility. However, these features may come at the cost of platform dependence and complexity which results in a steeper learning curve for beginners. *Commercial game engines* continue to push the frontiers of real-time graphics thus resulting in highly integrated systems. A few such systems, such as Quake II, have been made open source and have been used for instruction [Laird 2001]. However, these systems also have steep learning curves due to their highly specific goals and resulting software complexity. *Authoring systems*, such as 3D Game Studio, may provide an alternative for game design instruction. These systems typically provide application-type development tools for level and character editing, and scripting languages for programming. While easier to use than low-level and commercial engines, authoring systems typically provide a single framework that may make it difficult to teach a wide range of concepts that covers fundamental as well as advanced topics.

Any of these systems might be used in an instructional setting, since the success of a course is much more dependent on the instructor's use of a tool than the tool itself. Our goal, however, is to develop an instructional game engine with a steady learning curve that can be tied to an incremental teaching approach to support the widest range of students' abilities over a series of courses.

3 GameX

The original goal of GameX was to develop a platform which could be used to teach fundamental concepts in computer graphics while simultaneously allowing students to collaborate with peers in the arts and humanities. In Parberry's course offerings he states: "We choose to use Windows, Visual C++, and Microsoft DirectX for two reasons: for those students bound for the game industry, it makes sense to expose them to tools actually in use in a significant segment of the industry, and for the rest, it is advantageous to expose them to a different set of software tools before graduation." [Parberry 2005]. Consistent with these views, GameX was designed as a library for C++. However, GameX abstracts away from DirectX and OpenGL for reasons described below.

Attention was placed on developing features that would facilitate incremental instruction, provide basic programming challenges, and allow room for growth. Ease of use was an important factor. One primary goal was to reduce development time so that engineering students could program games and collaborate with peers in the humanities in a single fourteen week semester.

The design strategy for GameX was to develop an incremental *application programming interface* (API) which would mirror the instructional goals of the teacher. A difficultly with learning low-level engines such as DirectX is the need to understand many parts of the system to begin development. Yet students who learn to program these systems are favored by the industry. GameX provides an interface which allow students to learn at this level without needing to know platform-specific details.

GameX Multi-Level Interfacing

a) Low-level drawing primitives

```
GameX.Start2D();
GameX.DrawClear ( 1, 0, 0 );          // red background (r,g,b)
for ( each pixel p in 3D model ) {
    q = student-computed 3D projection of p
    GameX.DrawPixel ( q.x, q.y );     // 2D point using current color
}
GameX.End2D();
```

b) Intermediate modeling and rendering

```
ImageX img;     // create image (video memory managed automatically)
img.Load ("my_img.jpg");              // load jpeg image from disk
GameX.Start3D();                      // default 3D viewing system
GameX.DrawLine3D ( q1, q2 );          // line in 3D, current color
GameX.DrawSphere ( q1, r );           // sphere at q1 with radius 'r'
GameX.DrawPolygons ( polygon_list );  // set of polygons
GameX.DrawImage3D ( q2, img );        // billboard image at point q2
GameX.End3D ();
```

c) High-level gaming system interfaces

```
GameX.Start3D ();
GameX.DrawTerrain ( my_terrain );     // render given 3D terrain
GameX.DrawCharacter ( my_char );      // render given 3D character
GameX.End 3D();
```

Figure 2. Multi-level interfaces in GameX including a) Low-level drawing commands, b) Intermediate 3D modeling and rendering, and c) High-level gaming system interfaces

All development using GameX occurs in C++, yet the degree of program abstraction can be varied by how the GameX API is used or taught. For example, students might learn about viewing systems by calling low-level drawing primitives in GameX (Figure 2a). At the next level, the instructor might wish to provide a default three dimensional viewing system while teaching concepts in 3D modeling (Figure 2b). At the highest level, the instructor can use systems in GameX to allow for collaborative interdisciplinary game development (Figure 2c).

While GameX has many features in common with other graphics engines the desire to support collaborative game design requires that the relative importance of features is carefully considered. Good teaching materials and documentation are essential to linking instructional goals to the learning tools. Advanced features such as real-time lighting and shadows are desirable but are lower in priority to the goal of providing a consistent interface.

Since GameX is open source students also have access to the source code which reveals how GameX commands are processed by DirectX and OpenGL. Advanced students are able to learn these low-level APIs through the GameX source code. This is consistent with the game industry approach which often has a set of tools available for new developers whose underlying implementation is revealed once more experience is gained.

4 Results

GameX was started in 2000, and the first version was used to found the Game Design Initiative at Cornell University (GDIAC) in Summer 2002. Project courses were immediately taught at three different levels. The first level, Part I, used GameX to teach

basic graphics concepts with homeworks on viewing systems, graphics, networking and simple physics. The second level, Part II, teamed computer science students with students from art and music to create collaborative game projects. Groups were expected to complete one or more games per semester. A third level, Part III, was also offered to advanced students who wished to use GameX to pursue professional level topics in game design. As an open source platform several students created extension modules for GameX itself. GDIAC is currently pursuing a series of courses based on these experimental projects. The first official course in game design (CIS 300) at Cornell University was offered in the Summer of 2004.

For all team based Part II projects students were given the choice of developing games using other platforms as well. Fifty three collaborative games have been developed by students at GDIAC between 2002 and 2004, while thirty seven of these (70%) were developed using GameX. Of the remainder, one group used DirectX (2%), seven groups used OpenGL (13%), and the remaining eight groups (15%) used another platform (Java, etc.)

The development of teaching materials in parallel with GameX was critical to success. Homeworks also served a dual purpose as tutorials. For example, students might be provided a simple working physics system implemented in GameX and then be asked to add collision detection. Providing a variety of useful instructional resources was found to be an important aspect of using GameX in the classroom.

A look at the Independent Game Festival (IGF) submissions for 2005 shows that the average development time for a student game is four to six months.[2] With GameX, collaborative games by students took less than six weeks to produce comparable results. The simplest GameX game requires less than a page of C++ code with event loops, video memory, and timing all handled internally by GameX. In one instance, a team of three students completed a viewing system, a physics engine, a path finding algorithm, and integrated artwork and music in less than a week. By removing system-level details such as maintaining texture lists, managing video memory and sustaining a given frame rate, GameX allows students to focus on creating games while learning basic concepts.

5 Conclusions & Future Directions

Instructional game engines, unlike commercial engines, are developed specifically to meet instructional needs. As a result the supporting features such as documentation and tutorials are developed in parallel with an incremental teaching approach. GameX supports game design instruction on multiple levels. The currently available release, GameX R5, supports 2D with some support for 3D [Hoetzlein 2004]. Initial course offerings at GDIAC focused on 2D for the sake of simplicity.

As is common with academic open source software, the vision for GameX exceeds the pace of development. In the future, we hope to include support for character animation, networking, particle systems and natural environments. Unlike other engines, however, our motivation has been to develop an incremental teaching tool with a well documented interface and adequate instructional materials. As a result, GameX allows students to explore fundamental concepts in graphics and game design on multiple levels based on instructional goals.

[2] Retrieved and compiled from IGF website: http://www.igf.com

Credits

GDIAC is part of the Computing and Information Sciences at Cornell University and has been supported by the GE Fund and Microsoft. Thanks to Rajmohan Rajagopalan, the current instructor for the Game Design Initiative at Cornell and to Justin Pease who made significant contributions to GameX.

References

Conway, Matthew et al. 2000. Alice: lessons learned from building a 3D system for novices. *Proceedings of the SIGCHI conference on Human factors in computing systems*, The Hague, Netherlands, April 2000, p. 486-493.

Deutsch, Claudia H. 2002. Design Courses Gain Favor. *New York Times*, New York, 1 Apr, 2002

Hoetzlein, Rama and Schwartz, David. 2004. Computer game design as a tool for cooperative interdisciplinary education. *Proceedings of the 2004 American Society for Engineering Education St. Lawrence Section Annual Conference*, Kingston, Canada, 2004.

Hoetzlein, Rama. 2004. GameX Official Home Page. Retrieved January 22, 2005 from website: http://www.rchoetzlein.com/gamex

Jones, Randolph M. 2000. Design and Implementation of Computer Games: A Capstone Course for Undergraduate Computer Science Education. *Proceedings of the ACM Technical Symposium on Computer Science Education*, Austin, TX, Mar, 2000

Laird, John. 2001. Using a Computer Game to Develop Advanced AI. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, 34, 7, p. 70-75, July 2001.

Overmars, Mark. 2004. *Game Design in Education*. Retrieved January 22, 2005 from Utrecht University, Computer Science Department website: http://archive.cs.uu.nl/pub/RUU/CS/techreps/CS-2004/2004-056.pdf

Parberry, Ian et al. 2005. Experience with an Industry-Driven Capstone Course on Game Programming. *Proceedings of the ACM Technical Symposium on Computer Science Education*, St. Louis, Missouri, 23-7 Feb, 2005

Yu, Connie. 2002. Developing A Game Programming Course. *First International Conference On Information Technology & Applications (ICITA 2002)*, Bathurst, Australia., 25-8 Nov, 2002

David I. Schwartz *is the Director of GDIAC, the Game Design Initiative at Cornell University, and Lecturer for the Department of Computer Science where he has forged a career in teaching and curriculum development. As a graduate student he published two introductory textbooks for Prentice Hall's popular E-Source series: Introduction to UNIX and Introduction to Maple. Schwartz co-founded GDIAC in 2001 with Hoetzlein to initiate programs in game design at Cornell University. Currently, Schwartz is leading the development of the public digital arts computer laboratory which hosts Cornell's game design courses.*

Rama Hoetzlein *completed a BA in Computer Science and a BFA in Fine Arts from Cornell University in 2001 where his thesis works focused on mechanical sculpture. Hoetzlein initiated the GameX platform to support instructional game development and co-founded GDIAC with Schwartz in 2001. Hoetzlein has worked in the fields of computer graphics, chemistry and biology and is currently pursuing an MA from the Media Arts & Technology Program at the University of California Santa Barbara. His current research interests are in interactive art, knowledge systems, and interdisciplinary education.*