MOBILE GAMING AND THE ZUNE

William Birmingham
Computer Science Department
Grove City College

Abstract

Classes in mobile gaming are very popular with students and provide them with knowledge and programming skills that are in great demand in both industry and graduate research programs. These classes can provide experience in the following areas: software engineering, advanced programming in modern object-oriented environments, user-interface design, networking, real-time programming, as well as principles of game design and programming. Until recently, mobile gaming required machines that were either costly or required special licensing. The Microsoft Zune, however, avoids those problems. The Zune is relatively inexpensive and is supported by an excellent SDK and IDE, both of which are free. In this paper, we describe our experience teaching mobile gaming with the Zune. We explain how the Zune platform is used, we outline the projects we use, the topics covered in lecture, and we give examples of game developed by students. In addition, we provide student assessment of the course. We describe how the course supports our ABET course and program outcomes.

Introduction

Mobile gaming is one of the most important and growing segments of the computer games industry[1]. It drives hardware and software innovation in the smartphone market segment, particularly among iPhone, Android and Windows Mobile devices. Mobile gaming also drives innovation in the gaming console market, particularly for Sony and Nintendo.

Mobile gaming is a great educational opportunity.[2] (See Kurkovsky [3] for an excellent bibliography of work in this area.) Classes in mobile gaming are very popular with students and provide them with knowledge and programming skills that are in great demand in both industry and graduate research programs. The classes can provide experience in the following areas: software engineering, advanced programming in modern object-oriented environments, user-interface design, networking, real-time programming, as well as principles of game design and programming. In fact, gaming draws on physics and mathematics as well. All in all, mobile game programming is an excellent capstone undergraduate experience.

The downside to these classes is that they require hardware and programming environments that are expensive, or difficult to acquire (e.g., professional game-development kits) or both. The cost alone can make these courses difficult to offer, particularly at smaller institutions with small budgets. Moreover, to use many of these development systems, students and faculty are required to sign non-disclosure agreements.

There is a simple, relatively inexpensive alternate: the Microsoft Zune.[4] The Zune can be programmed with Visual Studio 2008 and XNA 3.1[5] (an excellent game programming SDK by Microsoft), both of which are available for free. The Zune is a capable platform that supports 2D graphics, networking, media playback (music and pictures), and has game-friendly input devices. While the Zune does not have the graphics power of the iPhone or specialized consoles like the Sony PSP [6], it has more than enough power for 2D games. The cost for a Zune is around $120, or even less, and there are no additional expenses. In addition, there are no non-disclosure agreements to sign.

We have used the Zune in our advanced console-gaming course, along with more

sophisticated systems like the Xbox 360 and other console platforms, for two semesters. The Zune platform, while met with some skepticism at first by students, has proven to be popular and is an excellent platform for educational purposes. Students get the chance to develop unique and fun networked, multiplayer games by five weeks into the term.

In this paper, we describe our experience teaching mobile gaming with the Zune. We explain how the Zune platform is used, the topics covered in lecture, and we give examples of game developed by students. In addition, we provide student assessment of the course. We describe how the course supports our ABET course and program outcomes.

The Games Classes

The Computer Science Department at Grove City College has a three course sequence in the design and implementation of computer games[7]. We overview the material in these classes in the following sections.

2D Games

The first course in our sequence introduces 2D gaming and covers basic elements of game design and game-engine implementation. A game engine, even a relatively simple one for 2D applications, is a large piece of software that is typically developed, in our class by teams of three students. The engine consists of about 10 to 15 different source files (in c# for our class), 20 to 30 graphics assets, and 10 to 20 audio assets. Building a system this large forces students to use object-oriented design principles, which they learned in the three programming classes that proceed the games class.

In addition, the students must employ good algorithm design, especially with regard to memory usage, in order to have reasonable performance of their games, such as to avoid "jerky" sprite motion and missed collisions among the sprites. This activity reinforces what the students learn in data structures and algorithms classes.

The course covers basic physics needed for arcade-style games. In addition, we cover geometry needed for collision detection, such as ray-plane intersections, sphere-sphere intersection, and so forth. The mathematics is 2D vectors and some calculus (for the physics). The course also introduces basic AI concepts for tracking, evading, and pathfinding using algorithms like depth-first search, best-first search, and heuristic search (greedy and A*).

We cover basic gaming techniques, such as moving sprites on a screen, implementing good game mechanics, creating splash screens and cut scenes, and so forth. In addition, we cover, albeit briefly, meeting real time constraints, since the games must operate at a minimum of 30 frames per second.

At the end of this course, the students are very good at designing and building 2D games of reasonable length—a game will take from 10 to 30 (or more) minutes to play to completion. The students develop three games over the semester, with the third game an extension of the second.

3D Games

The next course in our sequence introduces 3D graphics and physics. While game design principles are similar between 2D and 3D games, the designer has a much "larger palette" in which to create a game. This larger palette does create an unusual problem: CS students are naturally drawn to "cool technology" and make game play a secondary concern. Thus, some students make great tech demos, but an impoverished game. Thus, we have to constantly stress the importance of game play elements during the term.

An important element in this course is camera work: the type of camera (e.g., first person) and how it is used has tremendous implications for game play. Yet, cameras are hard to code, let alone use in an effective way. To help students understand how the camera is used, we have them critically review camera work in both games and movies. The same holds for lighting, it presents both technical and gameplay challenges.

All the software engineering (object-oriented design) and algorithms concepts they used in the first course are present in this course as well, only they are more advanced. The 3D game engines are significantly larger with about double the number of code files and assets. Moreover, algorithm design and coding are more challenging and important as the performance constraints are very demanding.

3D graphics and physics require much more sophisticated mathematics than 2D, particularly with linear algebra and geometry. Of course, the graphics and physics move to 3D. The students learn about the graphics pipeline, and vertex and pixel shaders. They must become comfortable with multiple coordinate spaces (e.g., local, world and camera) and transforming objects among them. Our students usually do not have much trouble with the math or physics, but tend to have trouble visualizing how math affects the graphics (i.e., models do not appear where they "should"), which makes it hard to debug a game.

Finally, in the 3D course, we introduce modeling. The students create a range of models, from simple rectangles to fairly complex avatars (which are not animated). They also learn how to create terrain with height maps and multiple textures.

At the end of this course, students have good background in many aspects of 3D game design and development.

Networked Gaming

The final class in the games sequence is console gaming. For this class, we emphasize multiplayer, networked gaming using two console systems: the Microsoft Zune and the Xbox 360[1]. For purposes of this paper, we will concentrate on the Zune aspects of the course; the Xbox 360 allows students to expand on concepts used to program the Zune.

[1] We have available development systems for other consoles, which require special licensing. Some students use these systems for their projects.

Multiplayer, networked gaming offers a significantly different gaming experience than "arcade-style" games. One of the clear differences is that players compete or cooperate with each other, not simply against non-player characters (NPC). Thus, the style of gaming changes, and game play elements must change as well. In particular, we focus on how to create a game for multiple players—what works and what does not—and deemphasize the design of NPCs. This in turn reduces the amount of AI we cover in class.

From a technical perspective, the design choices greatly expand. We list a few examples here:

- Should the players share a screen or have unique views of the game. In other words, is there a common camera or separate ones for each player?
- How is computation divided among multiple machines? For example, is collision detection handled on local machines? Does a "host" machine calculate scoring and enforce rules, or is that task distributed?
- How does a game start and end? If players are distributed on the network, it is important that no player can begin a game before the other players are ready to play.
- How are packet loss and latency mitigated?
- For mobile gaming: how to implement reasonable game mechanics if the player is running, or at least moving, and has a relatively small screen (QVGA) on which to play the game?

For about half the term, we concentrate on mobile and pervasive games.[8] Pervasive games are played in the real world as well as the virtual world of their games. Often, the location of the player in the real world may affect what the game does. For example, a player must traverse a

playing area to find tokens or move around to avoid being "assassinated" by another player.

Clearly, this is not the case with non-pervasive games. We draw a distinction here with mobile gaming: in mobile gaming, the general idea is to give a console-like gaming experience on a handheld device. These games do not typically interact with the physical world—they played the same way no matter where you are. Students are free to build either type of game, although the Zune is better suited for mobile gaming.

For mobile gaming, the students build 2D games. There are two reasons for this: the Zune does not currently support 3D gaming[2], and building multiplayer, networked gaming is difficult enough that we want to simplify the coding and math as much as possible. Thus, the lack of 3D support in the Zune suits our pedagogical requirements well.

Table 1 provides the ABET program outcomes for the course.

Over the past two offerings of the Console class, students have created an impressive set of games. We list a few of these below[3]:

- An assassins game where players "attack" each other by passing a token among Zunes. The game is played over several hours across campus. The player wins by being the last one assassinated.

- An "audio surfer" platform game, where the platform heights, and power up and enemy spawn rates are controlled by the music to which the player is listening.

- A multiplayer role-playing game (RPG) where the objectives and maps are generated dynamically based on the music to which the player is listening.

- A series of mini-game challenges, where the players run around a playing area. In different parts of the playing area, different mini-games are played (the locations are defined using a separate set of Zunes that act as "hosts" and are hidden from the players' view). The player finishing first and getting the best scores on the mini-games within a time limit wins.

- An RPG where the player explores a city whose streets are populated with album cover art; the streets are organized into neighborhoods based on music genres. The objective is to find more objects than the other player within a time limit.

The Zune

The Zune is a good choice for the console class. It has networking, reasonable controls (a thumbstick and three buttons) and screen, and it provides access to a large music and texture (picture) library. The library's content can be loaded by the game, or the game can access the content placed in the library by the user. The device has a fairly small memory size for an application to run within; however, all assets can be loaded into the media library, which is much larger (8 Gbytes for our devices). As with the lack of 3D graphics, the limited memory serves a good pedagogical purpose by making students think carefully about memory use and encouraging efficient programming. Both of these things are easy to ignore on a laptop. All in all, the device has sufficient power and is a nice gaming platform.

The Zune, however, is not quite as useful for pervasive gaming, as it does not support any type of location service. We worked around this by placing Zunes in the world and using them as beacons or hosts.

[2] The Zune HD, a new device, does support 3D gaming. As of now, the SDK does not expose this capability.

[3] Many of these games are available for download at : www.gcc.edu/dept/cs/downloads.html

Table 1: ABET Course Outcomes.

(c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired need	Students will write specifications for the games they create, with emphasis on good game play and a sophisticated gaming experience. Students will develop games—including code, artwork, sound effects, and music—to realize the specifications.
(i) An ability to use current techniques, skills and tools necessary for computing practice	Students will develop the software for their games for the Microsoft Zune and the Microsoft Xbox 360 using Microsoft XNA 3.1. The Zune games will be implemented in c#. To run their games, students will use the Zune and the Xbox 360 as attached, development hardware. In other words, games will not run on Windows machines, and thus must be loaded and debugged on different hardware. These techniques are common for embedded computer applications, mobile device applications, etc. In addition, students will read papers from the games literature to find the latest techniques and best practices.
(j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices	This class will use a significant amount of mathematics and sophisticated algorithms for implementing console games. Physics, collision detection, and AI algorithms, particularly for console applications, rely on good implementations that balance memory usage and speed. Since memory is limited on consoles, and there is little memory management support, memory/speed tradeoffs are important. In addition, students will study the tradeoffs in memory, processing power, and assets storage on different console device, the Zune and Xbox 360. While maintaining a similar programming model, these devices have vastly different performance that directly affects the games that can be run on these devices.
(k) An ability to apply design and development principles in the construction of software systems of varying complexity	Students will need to develop a variety of games, from relatively simple ones for homework assignments, to complex ones for projects. Applying principles of object-oriented design, as well as other techniques, will be important to complete the projects.

XNA and Visual Studio

One of the *great* benefits about programming the Zune is XNA. As we mentioned earlier, XNA is an excellent SDK that provides a very rich set of objects for game development. XNA covers everything from graphics support (including shaders), to content pipelines, to network support. Since all our courses uses XNA, code can be reused (if applicable) on the Zune from the 2D game class. More importantly, however, is that art and music assets can be reused. This greatly reduces development time.

All development work with XNA is done with Visual Studio 2008 (VS 8). VS 8 is a powerful integrated development environment that allows projects to be deployed and debugged within it. This is very useful, as students can prototype games on their Windows machines, if they wish, and then deploy the same code directly to the Zune. Thus, many of the problems with building and deploying to an embedded system are

avoided—after all, the Zune acts as an embedded system during the development cycle.

We have experience developing games on various commercial consoles, and we believe XNA/VS 8 is one of the best to use for undergraduate class purposes. The documentation is very good, the compilers and debuggers work well, the SDK is excellent, and there are many "starter kit" code examples from which to build.

Microsoft makes both XNA and VS 8, along with documentation, available for free at the "XNA Creator's Club" site.[5] Best of all, no special licensing is needed.

XNA Networking

One of the advantages of XNA is that it provides a networking model that is specialized for gaming. Students can work directly with objects that represent the network, players local on a machine, and players remote on networked machines. Thus, they can operate on a much higher level than working directly with TCP/IP or UDP sockets.

For example, establishment of and teardown of a network can be considerable work. By establishing a network, we mean creating a connection to the network (e.g., a sockets interface) and then going out to find machines that have players who want to participate in a game. The later step of this process is difficult to do. XNA provides simple functions to establish the network. Similarly, forming packets and sending them can be an onerous task. XNA provides high-level support for this as well. XNA also provide three quality of service levels, ranging from unreliable, fast to reliable, slow.

XNA allows a game designer to easily simulate different network properties, such as latency and packet loss. An important part of the console class is learning network programming, so a simple way to adjust properties is a critical pedagogical tool, since students can easy see the effects of long latency or high packet loss. This aids in teaching "smoothing" and distributing computation load among networked machines.

Student Evaluation

When first presented with the Zunes students are intrigued, since they are nice pieces of hardware, but skeptical about gaming on them. After all, they spent nearly a year struggling to get more powerful laptops to do what they want. In addition, the interface on the Zune seems too simple: single thumbstick, a few buttons and a small screen. After developing games with them; however, most students like the platform.

In the past two terms, student evaluations of the console course show the following things:

- The first time we used the Zunes, the student mostly enjoyed programming with them. This class made a set of novel and sophisticated games, exploiting the capabilities of the device.

- The second time we used the Zunes, the students were mixed in their impressions. Some students like the Zune, while others were not happy with it. Those who did not like it complained of small screen size and some problems in getting the networking to work reliability.

We offer two additional anecdotal observations about the Zune:

- In both class offerings, a number of students bought own Zunes for further development after the class ended. (The CS department supplies students with Zunes for the class.)

- In our senior project class, a group of students decided to use the Zune for their project. The project is a tour of campus, which uses many of the principles they learned from making a pervasive game. This group is using the HD Zune and have high praise for it.

Summary

The combination of the Zune with XNA and Visual Studio makes an excellent platform on which to teach mobile and pervasive gaming. Students are able to create impressive games, as well as learn the principles of networked and multiplayer gaming. The device fits well with our course objectives.

While our students use the Zune after a year of classes, it is reasonable to expect junior or senior students to build games with the Zune as their first game programming experience. These students should be familiar with object oriented programming and using SDKs, preferably the Microsoft .NET SDK.

The Zune makes a great platform for *any* project where a lightweight mobile, networked computing platform is desired. The applications do not need to be games. As we mentioned earlier, a group of students is using the Zune to create a campus tour.

Finally, the Zune is cost effective. In a small school like ours, where we are required to provide all computing hardware and software for our students, the Zune is the only platform we could use.

Acknowledgements

We would like to acknowledge Grove City College's Swezey research fund for providing equipment and student research stipend support to portions of this work. In addition, we appreciate the work done by Justin Kabonick and Adam Kaufman on parts of this work, and the Comp 447 classes from 2008 and 2009.

Bibliography

1. Soh, Jason O.B. and Tan, Bernard C.Y. Mobile gaming. *Communications of the ACM.* March 2008, Vol. 51, 3.

2. Sung, Kelvin. Computer games and traditional CS courses. *Communications of the ACM.* December 2008, Vol. 52, 12.

3. *Engaging students through mobile game development.* Kurkovsky, Stan. Chattanooga, TN : ACM, 2009. Proceedings of the 40th ACM technical symposium on Computer science education.

4. Microsoft Corporation. *Zune*. [Online] http://www.zune.net/en-US/.

5. Microsoft Corporation. *XNA Creators Club.* [Online] http://creators.xna.com/en-US.

6. *User case study and network evolution in the mobile phone sector (a study on current mobile phone applications).* Fritsch, Tobias, Ritter, Hartmut and Schiller, Jochen. Hollywood, CA : ACM, 2006. Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology.

7. *From 2D to consoles: A three-semester computer game development curriculum.* Birmingham, William P and Adams, David. Honolulu : ASEE, 2007. ASEE Annual Conference.

8, Montola, Markus, Stenros, Jaakko and Waern, Annika. *Theory and design pervasive games: experiences on the boundary between life and play.* Burlington, MA : Morgan Kaufman, 2009.

Biographical Information

Dr. Birmingham is the chair of the Computer Science Department at Grove City College. Before coming to Grove City College, he was a tenured Associate Professor in the EECS Department at the University of Michigan, Ann Arbor. His research interests are in AI, computer gaming, mobile computing and communications, and computer-science pedagogy. He received his Ph.D., M.S., and B.S. all from Carnegie Mellon University.