CMSC 425: Lecture 1
Introduction to Game Programming

Reading: Further information can be found in the first chapter of *Introduction to Game Development* (2nd edition), ed. S. Rabin, 2009.

Computer Game Programming: The famous game design Sid Meier once defined a *computer game* as "a series of interesting and meaningful choices made by the player in pursuit of a clear and compelling goal." A somewhat more concrete definition of a computer game, due to Mark Overmars, is "a software program in which one or more players make decisions through the control of game objects and resources, in pursuit of a goal." This course is concerned with the theory, practice, and technologies underlying the development of modern computer games of all varieties.

A Brief History: Today's computer games constitute a multi-billion dollar industry. There is an incredible variety of game genres: strategy, sports, shooter, role-playing, racing, adventure, and so on. Some games induce their players to dedicate long hours in pursuit of a distant goal, and others can be picked up and played for just a few minutes. Some games create astoundingly realistic and complex three-dimensional worlds and others involve little more than a simple 2-dimensional grid and very primitive computer graphics. Some games engage tens of thousands simultaneous users, and some involve a single user. How did we get here?

The Early Days: Computer games are as old as computers. One of the oldest examples was from 1958. It was a Pong-like game called *Tennis for Two*, which was developed by William Higinbotham of Brookhaven National Lab in 1958 and was played on an oscilloscope. Another example was a game developed in 1961 by a group of students at MIT, called *Spacewar*. It was programmed on the PDP 1. When the Arpanet (forerunner to the Internet) was developed, this program was available disseminated to a number of universities (where grad students like me would play them when their supervisors weren't watching). It is credited as the first influential computer game, but the influence was confined to academic and research institutions, not the general public.

Prior to the 1970s arcade games were mechanical, with the most popular being the many varieties of pinball. The computer game industry could be said to start with the first arcade computer game, called *Computer Space*. It was developed in 1971 by Nolan Bushnell and Ted Dabney, who founded Atari. One of Atari's most popular arcade games was *Pong*. There was a boom of 2-dimensional arcade games from the mid 1970s to the early 1980s, which was led by well-known games such as *Asteroids*, *Space Invaders*, *Galaxian*, and numerous variations In 1980, a popular game in Japan, called *Puck Man*, was purchased by Bally for US distribution. They recognized the enticement for vandalism by changing "Puck" into another well known 4-letter word, so they changed the name to "Pac-Man." They game became extremely successful.

The 70's and 80's: During the 1970s, computer games came into people's homes with the development of the Atari game console. Its popularity is remarkable, given that the early technology of the day supported nothing more sophisticated than Pong. One of the popular features of later game consoles, like the Atari 2600, is that it was possible to purchase additional cartridges, which looked like 8-track tapes, allowing users to upload new games.

The game industry expanded rapidly throughout the 1970s and early 1980s, but took an abrupt downturn in 1983. The industry came roaring back. One reason was the popularity of a game developed by Shiguero Miyamoto for Nintendo, called *Donkey Kong*, which featured a cute Italian "everyman" character, named *Mario*, who would jump over various obstacles to eventually save his lady from an evil kidnapping gorilla. Mario went on to become an icon of the computer game industry, and Donkey Kong generated many spin-offs involving Mario, notably *Super Mario Bros.* Eventually Donkey Kong was licensed to the Coleco company for release in their home game console and Super Mario Bros. was one of the top sellers on the Nintendo Entertainment System (NES).

Consoles, Handhelds, and MMOs: The 1990s saw the development of many game consoles with ever increasing processing and graphical capabilities. We mentioned the Nintendo NES above with Donkey Kong. Also, the early 1990s saw the release of the Sega Genesis and its flagship game *Sonic the Hedgehog*. Another example is the Sony Playstation, with the very popular game *Final Fantasy* (version VII and on).

In the early 2000s came the rise of so called *sixth generation game consoles*, which include the very popular Sony Playstation 2 (and the hit game *Resident Evil*) and the Microsoft XBox and their seventh-generation follow-ups, the Playstation 3 and XBox 360 and Nintendo Wii, which dominate the market today.

A parallel thread through this has been the development of handheld game consoles. These include the Nintendo Game Boy, which was released in the late 1980s and early 1990s, the Nintendo DS in the mid 1990s, and the Playstation Portable in the mid 2000s.

Modern times have seen more games move to general purpose handheld devices, like the iPhone and Android. This has been accompanied by an infusion of games into social networking systems. With the advent of widely available game development systems, such as Unity and Unreal, there has been an explosion of independent, *indie*, games, and various systems for funding and distributing these projects.

Recent Trends: Game technology continues to develop in terms of the complexity and scale of games. Recently, with development of relatively inexpensive head-mounted displays, there has been a resurgence of interest in systems *virtual reality* and *augmented reality.* This technology increases the sense of immersion into the game's world. Improvements in sensor technology has led to more sophisticated controls based on voice recognition and gestures. The Microsoft Kinect, Leap Motion, and various devices for recognizing body and hand gestures are examples.

Another trend has been the increase opportunity for users to contribute to the games they play. For example, *open source* game software can be modified by users and shared. Games can also provide users the ability to modify and share artistic elements and design new game levels.

Elements of Computer Games: While computer games are fun to play, they can be terrifically challenging to implement. These challenges arise from the confluence of a number of elements that are critical to the execution and performance of the game. These include the following:

- **Real-time 3-dimensional computer graphics:** Most high-end computer games involve the generation of photo-realistic imagery at the rate of 30–60 frames per second. This process is complicated by the following issues:
  - **Large geometric models:** Large-scale models, such as factories, city-scapes, forests and jungles, and crowds of people, can involve vast numbers of geometric elements.
  - **Complex geometry:** Many natural objects (such as hair, fur, trees, plants, water, and clouds) have very sophisticate geometric structure, and they move and interact in complex manners.
  - **Complex lighting:** Many natural objects (such as human hair and skin, plants, and water) reflect light in complex and subtle ways.
- **Artificial intelligence:** The game software controls the motions and behaviors of nonplayer entitites. Achieving realistic behavior involves an understanding of artificial intelligence.
- **Motion and Navigation:** Nonplayer entities need to be able to plan their movement from one location to another. This can involve finding a shortest route from one location to another, moving in coordination with a number of other nonplayer entities, or generating natural-looking motion for a soccer player in a sports game.
- **Physics:** The phyical objects of a game interact with one another in accordance with the laws of physics. Implicit in this is the capabiity to efficiently determine when objects collide with one another, and how they should move in response to these collisions.
- **Networking:** Multiplayer online games use a network to communicate the current game state between players. Due to the latency inherent in network communication, the games states that individual players perceive are momentarily inconsistent. The game software must hide this latency and conceal these inconsistencies from the players.
- **Databases:** The game state, especially for multiplayer online games, is maintained in a database. The database receives requests for updates of the game state from multiple sources, and it must maintain a consistent state throughout.
- **Security:** Since the origin of games, there have been people who have sought ways of circumventing the games. This is particularly an issue in multiplayer games, where one player's cheating behavior degrades the experience for an honest player. For this reason, game software needs to be vigilant to detect and obstruct illicit game interaction.

Game Engines: In the old days, programmers would design games from scratch. Given the complexities of modern games, game programmers rely on software systems called *game engines* to provide underlying support (rendering, animation, physics). This frees the game programmer from many of the more mundane tasks in order to focus on the essential elements of the game play. Some common examples of game engines include Unity 3D, Unreal Engine, CryEngine, and Godot.

The Scope of this Course: At some schools, game development constitutes a series of courses on various topics. Here, we will be able to focus on only a small part of the spectrum of relevant topics. While most game designers make use of sophisticate software tools (for graphics, modeling, AI, physics), it is not within the scope of this class to teach a particular set of tools (even though we will discuss game engines for the sake of project development). As in most upper-division computer science courses, our interest is not in how to *use* these tools, but rather how to *build* these systems. In particular, we will discuss the theory, practice, and technology that underlies the implementation of these systems.

This semester, we will touch upon only a subset of these issues. For each, we will discuss how concepts from computer science (and other areas such as mathematics and physics) can be applied to address the challenging elements that underlie game implementation.

Course Overview: In this course, we will provide an overview of what might be called the science and engineering of computer games. In particular, we will see how concepts developed in computer science can be applied to address the aforementioned elements of computer games. These include the following:

Game Engines: The organization, structure, and overall features of a typical game engine. Introduction to the Unity game engine.

Geometric Programming and Data Structures: Review of geometry and linear algebra and their applications to game programming. Linear and affine transformations and 3-dimensional rotation. Bounding volumes and efficient collision detection.

Modelling, and Animation: Shape representations and meshes, level of detail, terrain modeling, articulated models and skinning, animation, texture modeling, procedural generation and geometry synthesis.

AI for Games: Agent-based systems, finite-state machines, decision making and planning.

Motion Planning and Navigation: Path planning algorithms and A*-search, navigation meshes, multiple-agent motion and crowds, pursuit-evasion.

Networking and Online Games: TCP/IP, sockets programming, multiplayer gaming, latency hiding, distributed data consistency, security.

Other Topics: Physics (Newtonian dynamics, particle simulation, mass-spring models), physics engines, aural rendering (audio and HRTFs, audio acquisition and libraries, local and global aural rendering).