CS 248
Interactive Computer Graphics

Instructor: Ron Fedkiw
Website: cs248.stanford.edu
Meeting Times: Tuesday and Thursday,
10:30am to 11:45am
CS 148 vs. CS 248

• Rendering is an important part of creating a video game
• CS 148 focused on rendering
• CS 248 thus will not deal too much with rendering
• Use a scanline renderer in your video game
• Ray tracing is too slow for real-time rendering in video games --- but can be used to generate high-quality texture maps for the scanline renderer
  • Albeit real time ray tracers do now exist
CS 248: Overview

• **Goal: Create a video game!**
  
  • Combine rendering knowledge from CS 148 with ideas from the rest of computer graphics including
    • (computational) Geometry
    • Animation
    • Simulation
  
  • CS 248 will focus on making things move
    • i.e. Animation and Simulation (along with the necessary computation geometry to make this happen)
Geometry

- The environment in which a game is set is often one of the most compelling components of a game.
- An immersive world makes the game both engaging and exciting.
- Worlds can be created manually by an artist and/or be procedurally generated.
Geometry

- Avatars and their opponents represent important game geometry
- In addition, they interact with the world via collisions, etc., which requires computational geometry algorithms
Animation

• Animation is necessary to add motion to the geometry in the game
  • Animated by an artist
  • Captured from a performer or puppeteer
  • Simulated with equations/rules
Simulation

• Simulation is another way of moving the geometry
  • could be considered just one method of animation, but has taken on a life of its own

• Instead of specifying positions/velocities explicitly, physics equations (or other rules) are solved to get these values

• Allows a player to more robustly interact with the environment
CS 248 Outline

• Weeks 2 & 3 Animation
  • Basics, Animation Curves and Splines, Etc.
  • HW #2 Animation
  • 15% of final grade

• Weeks 4 & 5 Simulation
  • Particles & Particle Systems (cloth, flocking, etc.), Rigid Bodies
  • HW #3 Simulation
  • 15% of final grade

• Weeks 6 & 7 Character Animation/Simulation
  • Characters and Articulated Bodies, Animation and Simulation Thereof
  • HW #4 Character Animation/Simulation
  • 15% of final grade

• (*) You may work with a partner. Grading will consist of in person live demos on Monday afternoons with the CAs (just like CS148). See the web site for more details.
Gaming Platforms

• The constraint of interactivity requires one to put extensive effort into the platform chosen for the implementation of the game
  • Multithreaded PC Games
    • Make use of all available resources on the PC including both the CPU and the GPU, make use of all cores on the CPU using multithreaded parallelism
    • Very high end graphics
  • Mobile Games
    • Often simpler than PC games due to the limited computing resources available, very different style of user input and interactivity, using sensors on the device is a must
    • 2D games making use of sensors and touch screens
  • Client/Server/Browser Games
    • Communicate between multiple computers and browsers, the browser has many tools to aid in multiplayer communication, many networking challenges
    • E.g. racing games, mmos, etc.
  • Console Games
    • Xbox One, PS4, Wii U
    • Very specialized and standardized computing environments
    • allows for mass production of very low cost machines(consoles) with optimal resources
    • the game designer can make many assumptions ignoring any hardware variations in order to optimize the game and gameplay
Platform: Multithreaded PC

- Multi-core CPUs are the norm for today’s computers. Any game produced today will be released in a market dominated by multi-core processors. N.B. both PS4 and Xbox One look like a PC!

- Work can be divided into multiple tasks which are subsequently distributed among multiple threads
  - Functional Decomposition: Different threads dedicated to physics, sound, rendering, networking, AI, GUI, etc.
  - Data Decomposition: Further increases the concurrency of each function subsystem

- PCs have much more powerful computing resources (CPU and GPU) compared to other platforms allowing PC games to be much more complex and realistic

- Your game should be visually/technically impressive, use threads, and be 3D

- There are many tools for implementing threading, such as POSIX Threads (Pthreads), Native Win32 Threads, OpenMP, OpenCL, IntelTBB, etc.
Platform: Mobile Devices

- More opportunities for user interaction compared to a PC game
  - Touch screen: Allows for flexible tactile input and feedback
  - Multiple sensors: accelerometer, gyroscope, magnetometer, etc.
  - Cameras: Interact with and use information from the real-world

- Often rely more on immersive gameplay than superior graphics
  - Simpler game scenes (typically 2D instead of 3D)
  - Less computing power compared to a PC
  - Fast simulation models (e.g. shape matching for deformable body, SPH for fluid. Conventional simulation models on the PC are too expensive for mobile)
  - OpenGL ES standard is a subset of OpenGL

- Your game should make use of the special interactive and sensor driven features of the mobile device (tablets are preferable), and can be 2D
Platform: Client/Server/Browser

• **Client-Server Model**
  • Server maintains connections with each of the clients
  • Clients do not communicate with each other, but can only communicate indirectly through the server

• **Peer-to-Peer Model**
  • Peers are coequal nodes
  • Communication does not rely on a server
  • Decentralized system

• Browsers have lots of tools useful for implementing client/server or peer-to-peer games

• Running a browser game alone on a PC is not an efficient use of resources. All browser games should be either client/server or peer to peer. They can be 2D.
Platform: Client/Server/Browser

• Advantages: Cross-platform and convenient
  • Can play a game as long as you have access to the browser, no need to download any client program
  • Do not need to deal with the underlying operating system, just the browser itself
  • The ability to communicate with a server or other players makes browser games versatile.

• Disadvantages
  • Gaming experience is often limited in scope
  • Programming within a browser has its own unique challenges
Platform: Client/Server/Browser

• Large number of technologies available
  • Adobe Flash: Well established, but gradually being replaced by others
  • HTML 5: Open standard, well supported by the majority of browsers, performance tends to be lacking (especially in 3D)
  • WebGL:
    • Based on OpenGL ES
    • Hardware acceleration: Can handle complex 3D scenes
    • Several libraries build on top of WebGL making it easier to navigate

• Communication Paradigms
  • WebSocket
    • Designed to be implemented in web browsers and servers over TCP
    • Programmed using Go (recommended), Lua, Haskell, etc.
  • Ajax (Asynchronous JavaScript and XML)
    • Load content with JavaScript asynchronously
    • Communicate without waiting
2D Games

- 2D games are allowed under certain circumstances, since rendering is not the focus of this course.
- Need to do a very good job incorporating the topics covered in this course (computational geometry, animation, and simulation) into the game.
- 2D games are not allowed for the “Threaded PC” option, since the whole point of the PC is to showcase computational power.
- For “Mobile”, we strongly recommend (prefer) 2D in order to lighten the load on rendering and stress other aspects of the game.
Unity Game Engine

• We will use the Unity game engine throughout the course
  • This includes the animation and simulation homeworks
  • Thus it is very important that you do not miss the lectures dedicated to getting you up to speed on the Unity Engine!
  • Contact the CAs via email or see the web site to set up your free license

• This Thursday’s lecture (Jan 7) will be a Unity boot-camp to get you started

• Then every 2 weeks after that, Thursday’s lecture will be dedicated to the Unity engine:
  • Animation (Jan 21), Simulation (Feb 4), Character Animation/Simulation (Feb 18)
Homework 1

• Due Monday the 11th
• Live Demo with the CAs
• Install the Unity Engine, set up a scene/level, and demo it to the CAs
  • import some simple or interesting geometry
  • set up a camera, set up lighting, add textures to your geometry
  • see the web site for more details
• We will get you started on this via Thursday’s lecture
• 5% of the final grade
• You may work with a partner
Game Design

• 50% of your final grade is directly related to your game
• You may work in teams of 1 to 4 people
• We strongly encourage you to use the Unity Engine since we will have spent 4 lectures teaching you how to use it for your first four homework assignments
• The game must draw heavily on the concepts discussed in the course (talk to the instructor or CAs if you need clarification)
• The last 3 weeks of lecture are dedicated to game development
• Week 8: Game Design, Interactivity, and AI
  • HW 5: hand in a list of your team and a description of your proposed game, 10% of final grade
• Week 9: CAs will demo the 3D and 2D games that they created, and provide source code
Game Demos

• The completed games will be live-demoed during the regular final exam time slot for the course, and will be 30% of your grade
  • Each person will independently submit a 1 page write-up detailing what they did for the game both individually and in collaboration with others

• You will also be required to give a live in-class demo of what you have so far during the last week of classes
  • Tuesday March 10 and Thursday March 12
  • This counts as 10% of your grade

* The final exam slot is typically used for a game competition. More details later...