Part 1: Video game are complex software!!!
2 person team
3 years
300 person team
10 years

Final Fantasy 15
1. Video games are *real time* complex simulations, and must be efficient.
TECHNICAL CHALLENGES OF VIDEO GAMES

1. Video games are real time complex simulations, and must be efficient.

1999 Roller Coaster Tycoon written by one guy in x86 assembly language
1. Video games are *real time* complex simulations, and must be efficient.

Today, more flexibility in language

Typically Object-Oriented

Use development tools like Visual Studio or Eclipse
TECHNICAL CHALLENGES OF VIDEO GAMES

2. People have high expectations for interactive worlds with lots of content
TECHNICAL CHALLENGES OF VIDEO GAMES

2. People have high expectations for interactive worlds with lots of content

Lots of content on tight deadlines.

Glitches and crashes are BAD.
3. Real time 3D graphics simulations

Doom 1993
Levels, dungeons, and rooms were not only for game pacing, but to limit the number of objects to compute and render at a time.
TECHNICAL CHALLENGES OF VIDEO GAMES

3. Real time 3D graphics simulations

2016 graphics

Pixar - Piper

Final Fantasy 15
Game Engines

Tools that fit the pieces together
Game Engine
1990s First-person shooters: **Doom** by id Software
## Game Engines: History

Architecture separates core software from game-specific assets

### Assets
- Art assets
- Game map/environments
- Rules of play

### “Engine” Software
- 3D graphics rendering
- Collision detection
- Audio system
1990’s Separation of game engine allowed “mods” by replacing assets

**ASSETS**
- Art assets
- Game map/environments
- Rules of play

**“ENGINE” SOFTWARE**
- 3D graphics rendering
- Collision detection
- Audio system

*Not okay mod.*
Unreal Engine: A full industry-grade development environment (advanced tool)
Unity: A full development environment (advanced tool)
A game engine has a data driven architecture that can be used to make many games

- That dragon cancer
- Clockwork
- Gardenarium
Art assets & animation
Graphics
Physics engines
Game loop
Art assets & animation

Graphics

Physics engines

Game loop
Art to game

Workflow of artists with tools and the game engine
The Final Fantasy 15 team cooked food and then photographed it as reference material for 3D modelers and shaders.
3D Scanning or image tracing

The Final Fantasy 15 team scanned their food and photographed it.

Modelers use reference drawings from different angles.
Modeling Software

AUTODESK® MAYA® 2015

model in progress

Final in-game model.
Textures and Shading

Final in-game model.
Back to the game

Unreal Engine place objects in scene with map editor
In the game engine

Visual programming languages allow animations, materials, and shaders to be written by artists.
In the game engine

Visual programming languages allow animations, materials, and shaders to be written by artists
In the game engine

Visual programming languages allow animations, materials, and shaders to be written by artists
Art assets & animation

Graphics

Physics engines

Game loop
Shaders = VERY TECHNICAL

COMPUTER GRAPHICS!
Technical Graphics Tools

OpenGL has bindings in lots of different languages

Powerful, but not easy to learn.

three.js

Some language bindings are more learner-friendly than others
Technical Graphics – eyes

1. Full refraction at cornea
2. No refraction at cornea

1.4 Eye Shader

- Refraction ON
- Refraction OFF

Full refraction at cornea
Technical Graphics – hair

Process of modeling and rendering character Lunafreya’s hair from Final Fantasy 15x
Graphics – Updating the Screen

Must be efficient!

The screen must be updated every frame, at 30fps to 60fps. Rendering and shaders are computationally expensive!
Graphics - Updating the Screen

**Occlusion culling problem:** don’t render hidden objects

Frustum culling: test if an object intersects with the frustum.

Portals: designers *manually* place simple primitives around chunks of the game world. The portals are invisible but cheap to test intersection on.
Occlusion culling problem: don’t render hidden objects

Frustum culling: test if an object intersects with the frustum.

Portals: designers manually place simple primitives around chunks of the game world. The portals are invisible but cheap to test intersection on.
Graphics – Updating the Screen

PVS: Potential visibility set precomputed. Very efficient for small environments. PVS is submitted to the renderer and items in the set are tested to make sure they are indeed visible. Bad: storage costs
Art assets & animation
Graphics
Physics engines
Game loop
Physics

Physics engines and tools
Physics

Unity or Unreal game engines have basic built-in libraries.

Create some mechanical mayhem as you learn about Unity’s physics options.

3D Physics

1. Colliders
2. Colliders as Triggers
3. Rigidbodies
4. Adding Physics Forces
5. Adding Physics Torque
6. Physics Materials
7. Physics Joints
8. Detecting Collisions with OnCollisionEnter
9. Raycasting
Physics engines

Calculate on-the-fly physics simulations, optimized for a game environment.

Hard-body physics, Havok Physics Engine

Soft-body physics, CryEngine Physics Engine
Physics engines

SDKs with visual debuggers that allow you to run physics simulations on your object to test your code
Dynamic animation

Euphoria by Natural Motion encodes lots of information about human muscles, bones, and nerves to dynamically create realistic character movement like falls.
Dynamic animation

Natural Motion editor with visual programming.
Art assets & animation

Graphics engines

Physics engines

Game loop
Game loop

- update player health
- update monster health
- physics engine
- render scene
- sound effects
- Heads-up-display
Thank you!
Questions?