Video Game Technologies

11498: MSc in Computer Science and Engineering
11156: MSc in Game Design and Development
Overview

• What is a game engine?

• Game engines:
  – Commercial
  – Open source

• Game engine architecture
  – Physics, AI, Graphics, etc.
What is a game engine?

- A **game engine** is the core software component of a computer or video game or other interactive application with real-time graphics (taken from Wikipedia).

- The term “game engine” was coined in the mid-1990s due to the development of first person shooters such as *Doom*, *Wolfenstein 3D*...
Game engine: main goals

• Provide for underlying technologies
  – Graphics Rendering
  – Physics engine
  – Sound
  – Scripting
  – Animation
  – Artificial Intelligence
  – Networking
  – …

• Simplify development process

• Run on multiple platforms
Top 10 commercial engines
(http://www.develop-online.net) at Friday, 26th June 2009

- Unreal Engine 3
- Gamebryo Lightspeed
- CryEngine 3
- Unity 3D
- BlitzTech
- Infernal Engine
- Vision Engine 7.5
- Bigworld Technology Suite
- Vicious Engine 2
- Torque 3D
Open source engines

- OGRE
- Panda3D
- Crystal Space
- Irrlicht

Crystal Space’s foliage generator

Irrlicht game

Blackout: a Panda3D game using high-end lighting for ambiance.
Game middleware

- Components in game engines can be based on middleware (Havok, SpeedTree, ...)
- Increasing popularity of MMOGs spawns new middlewares:
  - Gamebryo, HeroEngine, RealmCrafter, MultiverseNetwork, ...

**Advantages in using a game engine:**
- Less development time required
- Less testing and debugging
- Many features directly available
- Better focus on the game design

**Disadvantages in using a game engine:**
- No control over the implementation of features
- Adding features not yet in the game engine might be cumbersome
- Dependent on other licensing scheme for release
- Other libraries/toolkits linked with the game engine (physics, AI...)
The game loop

- A game is a real-time interactive application

- Three tasks that run concurrently:
  - Recompute the state of the world
  - The player interacts with the world
  - The resulting state must be presented to the user (graphics, sound, etc.)

- Limitations of real-world technology
  - 1-2 processors with limited memory and speed
The game loop: **coupled approach**

- 1st try: design update/render process in a single loop (coupled approach).

- **Advantages** of the coupled approach:
  - Both routines are given equal importance
  - Logic and presentation are fully coupled

- **Disadvantages**:
  - Variation in complexity in one of the two routines influences the other one
  - No control over how often a routine is updated
The game loop: *multi-threaded approach*

- 2nd try: design update process using two threads:

- **Advantages** of the multi-threaded approach:
  - Both update and render loops run at their own frame rate

- **Disadvantages:**
  - Not all machines are that good at handling threads (precise timing problems)
  - Synchronization issues (two threads accessing the same data)
Game engine architecture

Our Game

Game Engine API

Graphics Engine
Sound Engine
Physics Engine
AI Engine
Scripting Engine

Hardware Abstraction Layer - DirectX, OpenGL, ...

Hardware Layer - sound card, graphics card (physics card, AI card,...)
**Hardware layer**

- **Physical**
  - Graphics card
  - Sound card
  - Physics card
  - Input devices (keyboard, mouse, joysticks, game pads, steering wheels, ...)

- **Drivers**
  - Low level interface

**Hardware abstraction layer**

- **DirectX**
  - HAL (hardware abstraction layer)
  - Components
    - DirectDraw, Direct3D
    - DirectSound, DirectMusic
    - DirectInput, DirectPlay
    - (DirectSetup)
  - Still low level routines

- **OpenGL**

- **Others**
User interface

• To develop a generic high level design for a simple (2D) game.
• Rather simple
• Monitors input devices and buffers any data received
• Displays menus and online help (can nowadays be pretty complex)
• Should be reusable, especially as a part of a game engine
Graphics engine

- Higher level interface, tuned to a particular graphics and game type
  - Sprite-based
  - Isometric
  - Full 3D
- Can deal with higher level modeling concepts
  - Sprites
  - Solids
  - Characters (articulated) ...
- Scene Manager
  - Each scene is represented by a scene graph
  - Contains everything that appears on the screen
  - There may be different scene managers for terrain (heightmap), exterior and interior scenes, ...

- Handles more complicated display aspects
  - Mini map
  - Multiple views
  - Overlays
  - Special effects ...
- Some of these engines are for sale or available on the web
- Often remade or heavily tuned for each game
  - Too much time and money is spent on this
Sound engine

- **Function of sound**
  - Effects to enhance reality
  - Ambience
  - Clues about what to do
  - Clues about what is about to happen (but be careful)

- **Sound formats**
  - Wave (high quality, lots of memory, fast)
  - MP3 (high quality, compressed, slower)
  - Midi (lower quality, very low storage, limited, adaptable)
  - CD (Very high quality, fast, limited to background music)

- **Simultaneous sounds**
  - Mixers (hidden in the HAL)
  - Buffer management
  - Streaming sound

- **Special features**
  - Positional 3D sound (possibly with Dolby surround)
    - Important for clues
  - Adaptive music (DirectMusic)

- **Some sound engines:**
  - Wwise
  - FMOD
  - Razer Maelstrom
  - EAS

SONiVOX® Embedded Audio Synthesis (EAS™) technology is a multi-platform audio engine for embedded systems and devices.
**AI engine**

- **Behaviour & interaction (dialogue) scripts**
  - Especially in adventure games

- **Flocking**

- **Obstacle avoidance**

- **Attack strategies**
  - Hiding
  - Attacking player as a team of enemies

- **Decision making**

- **Path planning**
  - Search algorithms
  - Waypoint networks

- **Crowd behaviours**
  - Panic, riots, ...

**AI engines that are available:**

- AI-Implant
- DirectIA
- SimBionic
- AISeek (dedicated AI card)
Physics engine

- Handles the simulation of the world
  - Collisions
  - Terrain changes
  - Waves in the sea
  - Explosions
  - Object destruction
- Limited or non-existent in simple games
- Some commercial/open source engines:
  - ODE (Open Dynamics Engine)
  - Havok
  - Tokamak
  - JigLib

- Physics hardware:
  - Nvidia/Ageia PhysX
- Physics is more and more integrated into the gameplay and game subsystems
  - Physics-based animation
  - Interaction with objects using physics
Scripting engine

• **Advantages:**
  – Easy control of many (or all) features in the game engine
  – Scripting language often provides full OO control (like Lua)
  – Promotes data-driven design

• **Disadvantages:**
  – Performance
  – Development support tools
  – Learning curve

• **Common scripting languages:**
  – Python, Lua, GameMonkey, and AngelScript

http://www.flipcode.com/archives/Implementing_A_Scripting_Engine-Part_1_Overview.shtml

Implementing A Scripting Engine
Scripting engine (cont.)

- What belongs in the engine and what belongs in the script?

**Graphics**
- Rendering
- Shadows/lighting
- Occlusion culling

**Physics**
- Dynamics
- Collision detection
- Raycasts

**AI**
- Path-finding
- Fuzzy controllers
- Planning/A* search

**Graphics**
- Tomo-of-day
- Add/remove lights
- Load/moving objects

**Physics**
- Object mass/friction
- Collision events
- Raycasts events

**AI**
- Path selection
- Decision making
- Goals/Objectives
Summary:

• What is a game engine?
• Why to build up a game engine?
• Game engines:
  – Commercial
  – Open source
• Game engine components and middleware
• The game loop
• Game Engine Architecture
  – Physics, AI, Graphics, etc.