

ICS370 Computer Graphics

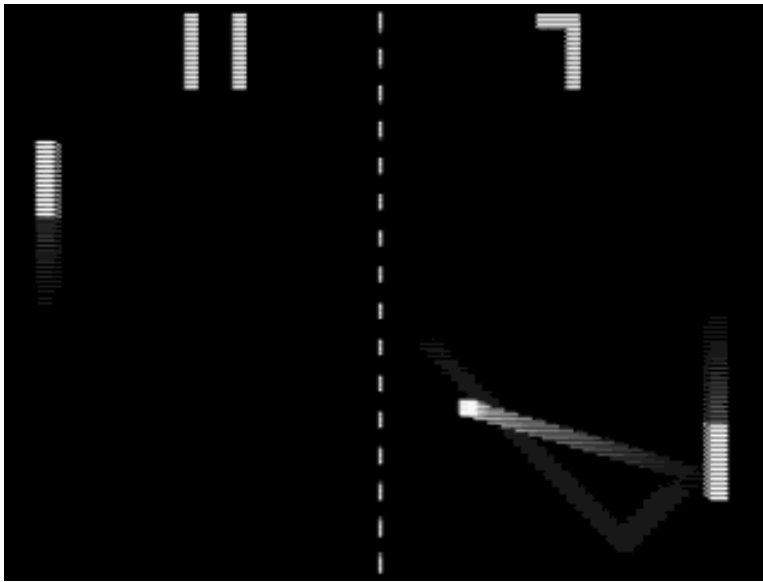
Instructor: Toby Daniel

Lecture 5

PC Graphics Evolution

- Computer Graphics has been driven by the Gaming Industry

- From



2D Monochrome

to



3D True Colour

3D Graphics

- Displayed representation of a scene or an object that appears to have three axes of reference:
 - height, x
 - width, y
 - depth, z

Perspective Projection

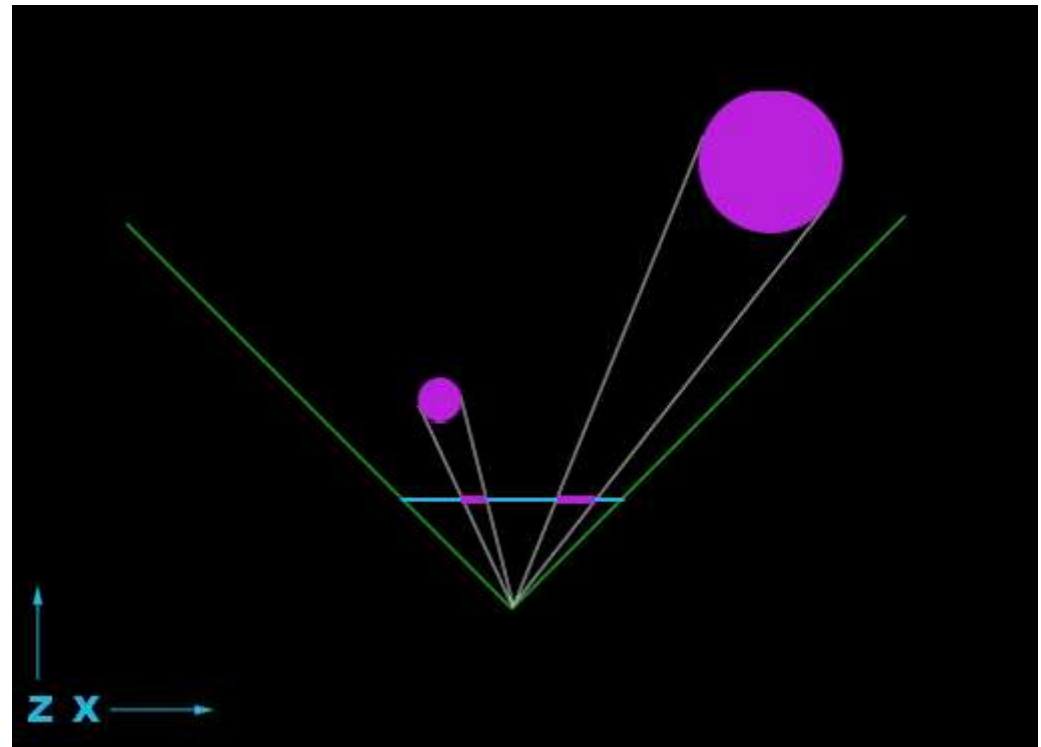
- View frustum = green lines (4 sided 3D pyramid)
- Projection plane = blue line (screen)
- Viewpoint is located at [0,0,0] (your eyes)

Perspective Scale:

$$x_{\text{screen}} = x_{\text{3D}} / z_{\text{3D}}$$

$$y_{\text{screen}} = y_{\text{3D}} / z_{\text{3D}}$$

User space -- Device space



Assumptions

- x_{screen} and y_{screen} are relative to the center of the screen (the viewpoint - your eyes!)
- 2D graphics are relative to the upper-left of the screen, not the center!
- The typical 2D coordinate system is X goes to the right and Y goes down.
- In 3D, X goes to the right and Y goes up.
- Your screen is not square - it's usually 1.3 : 1

Graphics API's (Application Program Interface)

- **DirectX** (originally called "Game SDK") is a collection of APIs for easily handling tasks related to game programming on Microsoft Windows. It is most widely used in the development of video and computer games for Windows.
- **Direct3D** is an API, owned and developed by Microsoft for the creation of 3D games. It is used for manipulating and displaying three-dimensional objects. Direct3D provides programmers with a way to develop 3-D programs that can utilize whatever graphics acceleration device is installed in the machine.
- **OpenGL** (Open Graphics Library) is an open specification for an applications program interface for defining 2D and 3D objects. With OpenGL, an application can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

Graphics API's

- Graphics API's provide a **software abstraction** of the GPU or video card.
- Low-level libraries such as DirectX and OpenAL are also commonly used in games as they provide **hardware-independent access** to computer hardware such as input devices, network cards, and sound cards.
- Why?
 - So you can create games and graphics without worrying about the specific type of hardware installed in the computer.

Games Engines

- A **game engine** is the core software component of a computer game or interactive application with real-time graphics.
- It provides the underlying technologies and simplifies development
- It often enables the game to run on multiple platforms such as game consoles and desktop operating systems; Linux, Mac OS X and MS Windows
- It can include:
 - a rendering engine (“renderer”) for 2D or 3D graphics
 - a physics engine or collision detection
 - sound
 - animation
 - artificial intelligence
 - networking

Advanced Game Engines

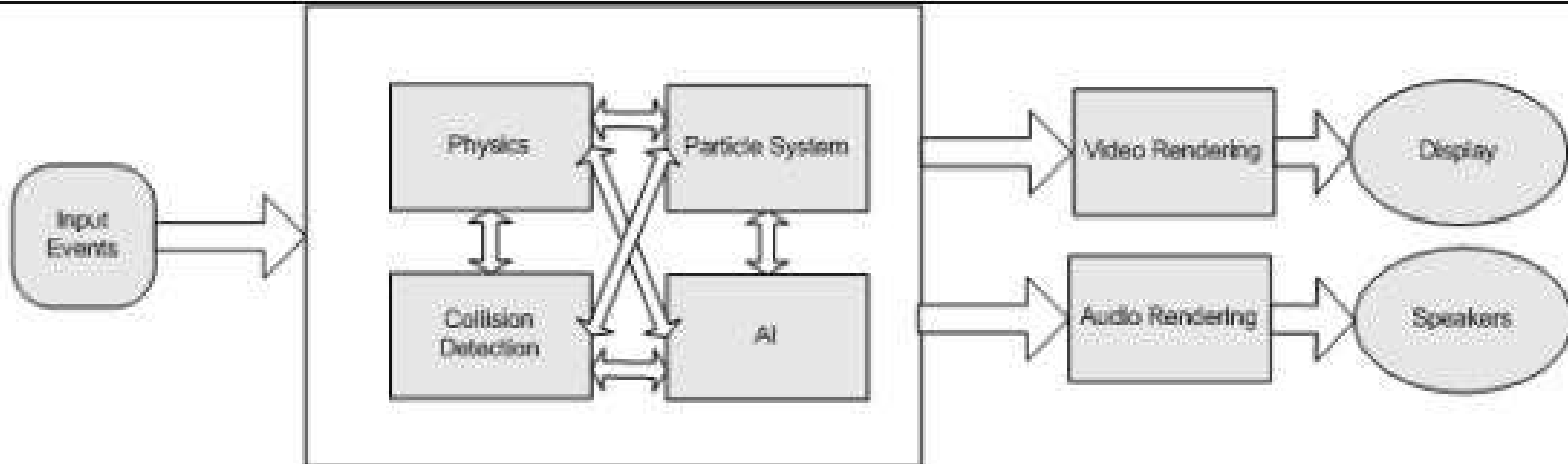
- Advanced game engines provide a suite of visual development tools in addition to reusable software components.
- Examples:
 - Unreal Engine 3,
 - Doom 3 engine,
 - CryENGINE2,
 - RenderWare,
 - Gamebryo

Benefits of Game Engines

- they provide a flexible and reusable software platform
- provides all the core functionality needed out-of-the-box to develop a game application
- provide platform abstraction with few, if any, changes made to the game source code.
- reduces costs
- reduces complexities
- reduces time-to-market

- Watch the demo of Unreal Engine 3.
- Make notes on what the developers say about it.

Interaction of components in a game engine



Notice the dependencies that exist between components within the engine itself: each subsystem can receive information from any other subsystem

Game Middleware

- Game middleware is designed with a component-based architecture
- This allows specific systems in the engine to be replaced or extended with more specialized (and often more expensive) middleware components such as:
 - Havok for physics,
 - FMOD for sound,
 - SpeedTree for rendering.
- Some game engines such as RenderWare are even designed as a series of loosely connected middleware components that can be selectively combined to create a custom engine,

Physics Engine

- A physics engine is a computer program that simulates Newtonian physics models.
- It uses variables such as:
 - mass,
 - velocity,
 - friction,
 - wind resistance
- It can simulate and predict effects under different conditions that would approximate what happens in real life or in a fantasy world.

Types of Physics Engines

- **High precision physics engines** require more processing power to calculate very precise physics and are usually used by scientists (computer modelling) and computer animated movies.
- **Real-time physics engines** simplify calculations and lower their accuracy so that they can be performed in time for a game to respond at an appropriate rate for gameplay.

Question - what is the effect of the annual increase in CPU & GPU processing power?

High Precision in Real Time

- Physics based character animation in the past only used **rigid body dynamics** because they are faster and easier to calculate
- Modern games are starting to use **soft body physics** now that it is possible. Soft body physics are also used for particle effects, liquids and cloth.

- Watch the Endorphin video.
- What type of physics engine does Endorphin use?
- Is Endorphin a games engine or middleware?

Homework

- Make a list of **commercial uses** for computer graphics.
- Separate this list into 2D and 3D graphics