DigiPen
INSTITUTE OF TECHNOLOGY

CS300 Introduction

Introduction

- Syllabus
- Topics to be covered
- Suggestions?

The 3D Pipeline

- What is the 3D Pipeline?

The 3D Pipeline

- What is the 3D Pipeline?

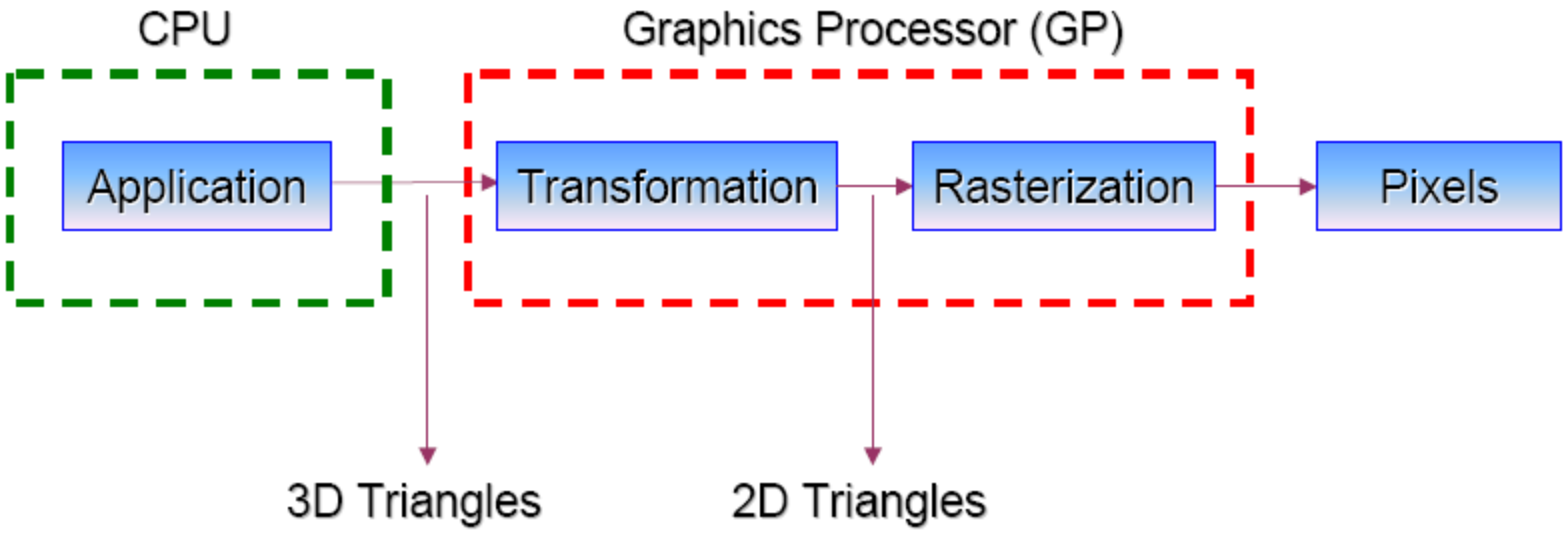
  A sequence of processes applied to 3D primitives and its attributes to generate 2D image.

The 3D Pipeline

- What are 3D pipeline components?

The 3D Pipeline

- What are 3D pipeline components?
  - Model space transform
  - World space transform
  - Camera (viewing) space transform
  - Lighting
  - Projection space transform
  - Clipping
  - NDC transform
  - Viewport transform
  - Rasterization



Model Space Transform

- Calculate vertex coordinates in model space
- Compute normal vectors in original model space

World Space Transform

- Transform vertex coordinates originally in model space to world space
- Translation, Rotation, Scaling, …

Camera Space Transform

- Transform vertex coordinates from world space to camera space
- Coordinates are now relative to the camera eye position

Lighting

- Why after camera space?
- Requires transformed normal vectors. So, how to transform the normal vectors?

Why after camera space?

- Multiple light calculations require camera position and direction.
- In camera space, camera position is (0,0,0) and camera direction is (0,0,-1) which simplifies our calculations.
- Model space is not suitable. Lighting requires transformed normal.
- World space is usually skipped (in OpenGL).
- Projection space is not suitable. It is not a 3D space.

How to transform the normal vector?

How to transform the normal vector?

- Transform them with the inverse transpose of the geometry transformation matrix.

$$B = (A^{-1})^{T}$$

- Where:
  - A is the matrix used to transform the vertices
  - B is the matrix used to transform the normals
- Note that the transformed normal might no longer be normalized

Projection Space Transform

- Transforming all vertices from the camera space to the projection space.
- Projection space is the 4D homogenous space

Clipping

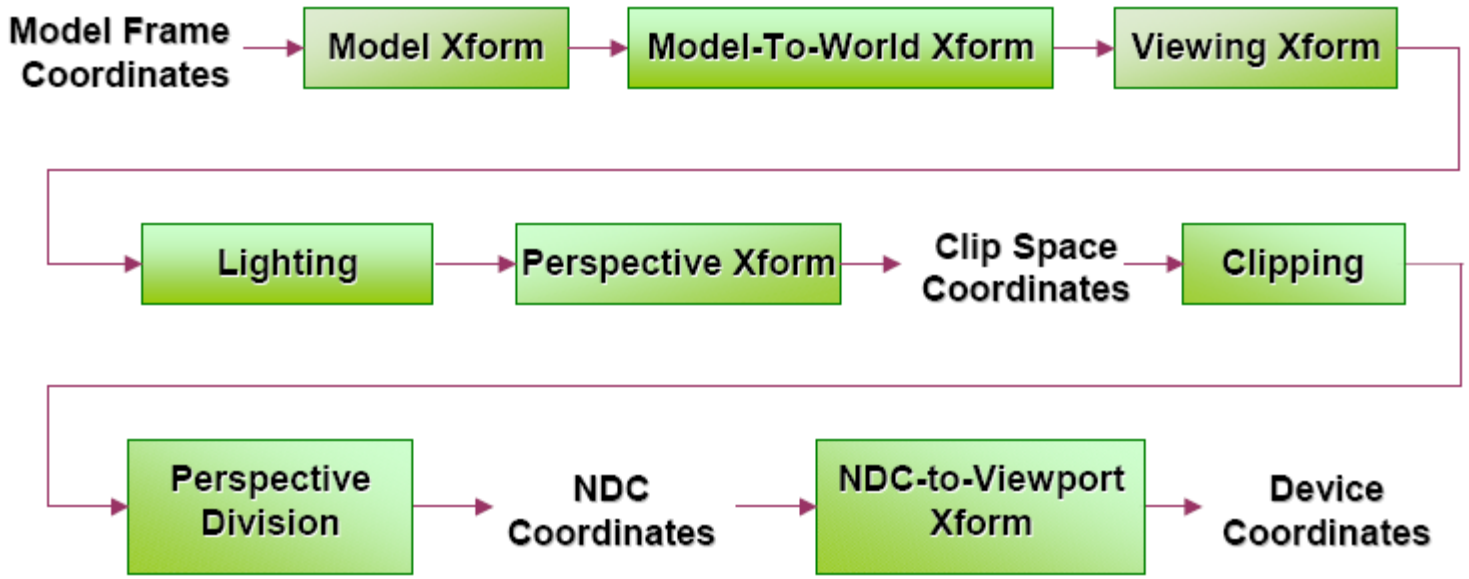
- Perform polygon clipping to remove points beyond the projection plane boundaries
- Clipping against the near plane is required in order to produce correct end result
- Clipping against the other five planes should be done to reduce unnecessary work further down the pipeline (CS350 – Spatial Data Structures)

Projection space to NDC space

- Normalize by dividing by ‘w’ component
- All vertex coordinates are within [-1, 1]

Viewport (window) Space

- Transform to window space to draw on frame buffer.



The GPU (Graphics Processing Unit)

- What is it?
- What is its purpose?
- What can it do?
- History of GPUs
- GPU forms

What is GPU?

GPU is a piece of hardware that is designed specifically to help the CPU in doing graphical work

It is considered to be a type of CPU but dedicated mainly to graphics

What is its purpose?

In the most basic form, GPU frees the CPU from doing graphics chores. As such, the CPU will have more time to do other things, like physics, collision, game play logic, etc

What can the GPU do?

It really depends on the GPU. In the early days (in the 80s), the GPU provided support for the CPU to quickly copy bitmap data to the frame buffer. Later on, when 2D accelerators arrived, the GPU drew basic 2D primitives for the CPU. Nowadays, the GPU can do much more. It can practically do the whole 3D pipeline on the chip and more…

More

In the early 3D GPUs, the whole chip is hardwired solely for graphics calculations. You might be able to change a few settings here and there, but that was it. Nowadays, the transformation and rasterization units of GPU can be programmed via the programmable vertex and pixel shaders.

More

The fact that the GPU is programmable opens up many possibilities. For example, you can use the vertex/pixel shaders to implement a custom lighting equation or use the vertex shader to modify your geometry.

More

You can even use the GPU to do non-graphics related algorithm (http://en.wikipedia.org/wiki/CUDA).

And because of the highly parallel nature of the GPU, it is not uncommon for these algorithms to run orders of magnitude faster

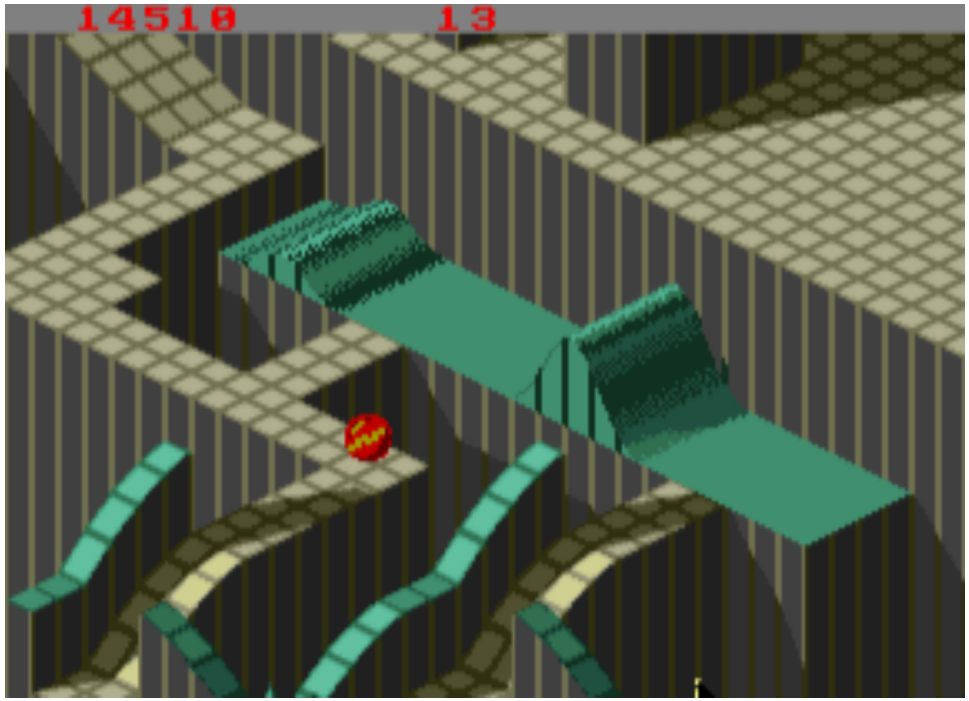
CUDA (Compute Unified Device Architecture)

History of GPUs

1985

Commodore Amiga contained the first graphical accelerator

Marble Madness (1986)



1991

First stand-alone 2D graphics accelerators are sold

Monkey Island 2 (1991)

1995

First 3d accelerators come to market

Quake (1995)

1999

Direct3D 7.0 added support for Hardware Transform and Lighting

- This moved 3D accelerators beyond simple rasterizers
- Set the precedent for pixel and vertex hardware shaders

Quake 3 (1999)

2001

DirectX 8.0 added hardware support for programmable pixel shaders

Morrowind (2002)

2003

DirectX 9.0 added hardware support for programmable vertex shaders

Half-Life 2 (2005)

2007

DirectX 10 represents a massive inflection point to the DirectX API

- Many old functions and drivers depreciated
- Adds support for a geometry shader
- Shader Model 4.0

Alan Wake (2008)

2011

- DirectX 11
- Hardware tessellation
- Hull shaders
- Compute shading capability

GPU forms

- GPU comes in different forms:
  - Dedicated
  - Integrated
  - Hybrid

Dedicated

- Most powerful
- Interfaced with the motherboard
- Has their own RAM

Integrated

- Utilize portion of the system's RAM
- Cheaper than dedicated but less capable

Hybrid

- Having both discrete and integrated GPU in the system
- Switching between the GPU for power efficiency
- Combining the power of both GPU for more performance

Stream Processing

- CUDA (Compute Unified Device Architecture)
- Run C code on GPU

Short Assignment

- Write 5 distinct points of description (or one paragraph) about each of the following
  - OPENGL
  - GLUT
  - GLUI
- We will discuss your answers (and more) next class!

Questions