**Chapter 1 Introduction**

1. **History**
   * become smaller & portable
   * from monochrome to true color
   * from 2D to stereoscopic to autostereoscopic
   * more natural user interaction
   * from specialized hardware to GPU
   * more investment needed
2. **Genre**
   * Action
   * Adventure
   * Strategy
     + Real Time Strategy
     + Turn Based Strategy
   * Simulation
     + Vehicle Simulation
     + Artificial Life
     + Construction and Management Simulation
   * Sport
   * Puzzle
   * Role Play Game
   * Online Game
   * Social Game
3. **Platform**
   * Arcade Games
     + Well defined configuration
     + Support more expansive hardware
   * Consoles
     + Fixed configuration, ease for support
     + Limited controller
   * PC
     + Can explore art-of-statement technologies
     + Convenience development environment
     + Difficult to support variety of hardware configuration
   * Portable Computing Devices
     + Fixed configuration
     + Relatively low computing powers
     + Limited programming resources
4. **Game Development**
   * Preproduction
     + Minimax Method
       - Decide feature sets in implementation priority
       - minmax > maxmax > maxmin > minmin
   * Production
   * Maintenance
5. **Trend**
   * 3D
   * Eye catching graphic
   * Increase level of detail
   * Distribute computation tasks to other processors
   * Mobile Gaming
   * Virtual Reality & Augmented Reality
6. **Problem**
   * Cost
7. **Market**
   * Local – Small, MMO
   * Mainland – Large, MMO, Mobile Game, Casual Game

**Chapter 2 Graphic Pipeline I**

1. **Game Development Methodology**

* **Monolithic Approach** – Develop source code for different stages by different/same programmers.
  + **Advantages**
    - Easy to manage
    - Allow creativity
  + **Disadvantages**
    - Difficult to modify level
    - No quite scalable
    - Not easy for work division
* **Game Engine** – A software which can provide support for drawing of graphics, sound as well as other game related function supports.
  + **Advantages**
    - Better work division
    - Encourage player participation
    - Generic Design – different artworks different product.
  + **Disadvantages**
    - Lack of creativity
    - Games produced tend to alike each other due to same inherent architecture

1. **Game Design Architecture**
   * **Update**
     + **Game Logic**
       - Player update
       - World update
       - NPC update
     + **World Update**
       - Passive – physics, collision detection
       - Active – Ai, scripting
   * **Render**
     + **Object-level Clipping**: Testing clipping at object level can benefit a lot as we can clip hundreds of triangles at one test.
     + **Hardware based Occlusion Testing**: Works on object level to discard large blocks of geometry in single query.
     + **Resolution Determination:** Geometry representation with LOD, Mipmapping texturing.
       - Reduce aliasing and save computational time for such high quality.
2. **Rendering Pipeline – Geometric Object**
   * **Polyhedral Representation** – Represent object as set of smoothly joined polygons.
     + **Advantages**
       - Simplicity
       - Generality
       - Computational efficiency
     + **Disadvantages**
       - Poor approximation in cases of smooth curved surfaces, complex shapes and life-like forms
   * **Object Space**
     + The most natural coordinate system for each geometry object.
   * **World Space**
     + The global coordinate system that is shared by all objects in the modeled scene.
   * **Camera Space**
     + The coordinate system with the eye or the COP as its origin.
   * **Screen Space** – perspective projection or orthogonal projection
     + The coordinate system that describes how light rays reach our eyes.
   * **Raster Space**
     + The coordinate system that determine which 3D points display in pixel.
3. **Optimization in rendering geometric object**
   * **Hidden Surface Removal**
     + **Backface Culling** – Faces of an object with normal pointing away from camera will be occluded by other faces.
       - Problem: Normals must point correctly.
     + **Depth Sor**t – Surfaces are sorted in decreasing depth order. Surfaces are rendered back to front.
       - Problem: Intersecting polygons/overlapping depth give wrong result.
     + **Z-Buffer Algorithm** – Maintain a Z-buffer with same resolution as frame buffer. During scan conversion compare: Z value of pixel about to be stored with Z value of current pixel in Z-buffer. If new pixel’s Z value is closer, store new pixel value (color) in frame buffer and new Z value in Z-buffer.
       - Frequent hardware implementation because of its simplicity and speed. O(N).
4. **Rendering Pipeline – illumination and shading**
   * **Phong Reflection Model**
     + Reflected = ambient + diffuse + specular
       - **Ambient**: Indirect Illumination from other reflected light.
       - **Diffuse**: Model light reflected from a surface.
       - **Specular**: Model the glossy appearance of shiny object. View point dependent.
   * **Interpolative Shading –** To recover approximately the visual appearance of curved surface which is now represented by flat polygons.
     + An approximate normal to the original smooth surface is given or can be computed at each vertex by averaging the normals to the polygons sharing that vertex.
     + The shading of a pixel can be obtained by a bilinear interpolation of the appropriate quantities from adjacent vertices.
   * **Gouraud Shading** – Calculate the intensity at each vertex using local reflection model. Intensities of interior pixels are determined by linearly interpolating the vertices’ intensities.
     + **Highlight anomalies**: If the highlight appears in the interior of the polygon, Gouraud may fail to shade this highlight because no highlighted intensities are recorded/calculated at the vertices.
     + **Mach banding**: Human visual system emphasizes intensity changes occurring at a boundary, which creates a banding effect. The bands can be obvious if insufficient polygons are used to model area of high curvature.
   * **Phong Shading** – Instead of interpolating the vertex intensities, Phong interpolates vertex normal vector.
     + Since illumination calculation must be invoked at each interior surface point, Phong shading is more expensive than Gouraud shading.
5. **Rendering Pipeline – Texture Mapping**
   * A technique to put texture (color pattern) on the object surface to improve visual richness and realism.
   * **Attributes to Be Modulated:** Surface color, Transparency, Normal vector perturbation, Specular and diffuse reflection.
   * **Bump Mapping**: This technique enables a surface to appear as if were wrinkled or dimpled without the need to model these depressions geometrically. The surface normal is angularly perturbed according to information given in a 2D bump map.
   * **Environment mapping**: Object is surrounded by a closed 3D surface onto which the environment is projected. Reflected rays are traced from the objects, hit the surface and then index into the map.
   * **Lightmapping**: Stores the lighting information in low resolution texture and multi-textured to form the per pixel lighting. Sample reflected light over a surface and store this in a 2D map.
     + **Advantages:**
       - Allow more sophisticated global lighting calculation method.
       - Little computational cost during rendering.
     + **Disadvantages**:
       - Tend to be heavily magnified.
       - Need long computation time for radiosity lightmap preparation.
       - The higher complexity of object, the greater the number of orientations exhibited by face planes.
       - Cannot light up dynamic objects.
     + **Dynamic Lighting – Moving spotlights** First use camera looking direction as a ray, and calculate intersection with scene. At intersection point, define a spherical light source and update the lightmaps accordingly
       - **Disadvantages:** No longitudinal light track, Only single ray calculation.

**Chapter 3 GPU and Shading Programming**

* **Programmable Graphics Pipeline** Each processing unit is executed independently and in parallel.

1. **Vertex Shader**: Vertex shader is executed at each vertex. Perform vector computation & matrix transformation here.
2. **Tessellation Shader:** Manage datasets of primitives presenting a scene and divide them into suitable structures for rendering. Vertices are tessellated into triangle.
3. **Geometry Shader**: Vertices are assembled into triangle. Triangle are then grouped into multiple Triangles. Perform procedural modeling here.
4. **Fragment Shader**: Fragment shader works for the projected and rasterized triangles. Perform pixel intensity calculation here.
5. **Compute Shader**: Designed for general purpose computing on GPU. The compute domain is abstract and configurable, multiple nodes can be configured together as a workgroup and they may share data via low-level shared memory which is usually only available via vendor-specific API, such as CUDA. Input and output are facilitated by using texture access, arbitrary image load and shader storage buffer object.

* **GPU Shading Languages**
  1. **Cg**: Replaces assembly code with a C-like language and a compiler. Launched for people to use shading language on OpenGL platform, as the standardization of OpenGL is very slow. Cg is cross-API (OpenGL & DirectX) and cross-platform (Windows, Linux, and Mac OS).
  2. **GLSL (OpenGL Shading Language)**: was developed mainly by 3DLabs. GLSL is part of the OpenGL 2.0+ core. It is only for OpenGL.
  3. **HLSL (High-Level Shading Language)**: was developed by Microsoft in 2002. It is only for Direct X only.

**Chapter 4 Game Design**

* **What is Game Design?**
  1. Image a game
  2. Define the ways it works
  3. describe the elements of the game
  4. transmit the information to team members
* **How to characterize it?**
  1. **Core Mechanics** – game rules
  2. **Storytelling**
  3. **Interactivity** – user interface, presentations
* **Production Pipeline**
  1. **Game Design**
  2. **3D modeling**
  3. **Texture**
  4. **Animation**
  5. **Unit test**
  6. **implementation**: coding
  7. **Test within game prototype.**
     + Focus on the core game mechanics such as main features and characters’ action.
     + Keep it simple, leave the artwork unless necessary.
  8. **MISC**: Audio, Network, etc.
* **Game Design Document**
  1. **High concept**: Express spirit of the game to publisher or partners.
     + **Premise** – a claim that is a reason for, or objection against, some other claim.
     + **Intended audience**
     + **Genre**
     + **Selling points**
     + **Target platform**
     + **Overall storying**
     + **Game play**
  2. **Game treatment**
     + Present in broad outline to publisher.
     + Fill in gaps and answer some questions left in high concept document.
     + Materials that is crucial to understand the game’s look & feel.
  3. **Game Scripts**
     + Document design decisions, documents the creative, conceptual, and functional aspects of the game.
* **The Design Theory**
  1. **Game Structure**
     + **Linear**: Games that have a single path to follow, have explicit goals, and have a clear and defined narrative sequence.
       - Easier to produce and control and hence lower production cost.
     + **Sandbox**: Games that enable players approach challenges in any order. Sometime it is open-end.
       - More freedom and thus enhance entertainment.
       - Designers must handle all possibilities in games.
  2. **Player Number**
     + **Single-Player**: A player interacts with NPC and objects controlled by the computer.
       - Focus on user experience
       - Need to implement AI
     + **Multiplayer**: A player interacts with other human players.
       - All players are created to have equal ability
       - Need to handle the networking and game performance.
  3. **Game World Style** – design decisions must serve the entertainment value while achieving overall degree of realism.
     + **Realism**: Simulate to real world. Require more computation and graphic rendering.
     + **Abstraction**: Not rely on a representation of the real-world. But a cartoon, surreal, dream world.
  4. **Game Representation**
     + **2D**: All actions and activities work in two-dimensions. Using only 2D graphics and 2D camera movement.
     + **3D**: Takes place in three-dimensions with 3D movements. The Role of camera is very important in 3D games.
  5. **Difficulty Curve –** Start Easy and End Hard.
     + **Quality Assurance** (QA) is the team for game test. They help to fine-tune the game difficulty.
     + **Tutorial levels** in early stages of a game.
* **Elements of a Game**
  1. **Features**: What make your game different from other games.
     + Vital to make the game world.
     + Enhance enjoyment of game only.
  2. **Gameplay**: A set of rules that define the player’s action and define the challenges the player must overcome.
  3. **Interface**:
     + **Interaction models**: the way the player interacts with the game world.
       - **Avatar**: controlling a single character that represents a player and influence the local area around him.
       - **omnipresence:** has ability to take actions at many places.
     + **Perspectives**: the way the player sees the world.
       - **First Person view**: Limited field of view.

1. More immersive feeling, more excitement.
2. The game character is not visible and makes players hard to perceive the whole world; game players need to switch to other view modes or need to see minimaps constantly
   * + - **Third Person view**: Follow behind the character. Widened FOV.
         1. Less immersive feeling.
         2. Allows players to clearly see the entire game world.
   1. **Setting**: the environment of the game legitimate to the player.

* **Motivations that Influence Designs**
  1. **Market-driven games**: Certain elements are generally thought to be popular.
  2. **Designer-driven games**: Designers retains all creative control and takes personal role in every creative decision.
  3. **License Exploitation**: Tie-ins with highly recognized movies, books, etc. Need to conform to rules laid down by original owners.
  4. **Technology-driven games**: Showoff technology achievement.
  5. **Art-driven games**: Showoff someone’s artwork. Designed by artists with strong visual sense but are new to game industry.
* **Design Input Sources**: gathering and assessing feedback is crucial for a modern game designer.
  1. **Team Inputs**
     + **Advantages**:
       - Provide early feedback.
     + **Disadvantages**:
       - Team had with the game for months, even years, easily blind to unintuitive mechanics or confusing UI.
       - Might burnout on the game together and forget simple players experience when starting the game.
  2. **Community Inputs**
     + **Advantages**:
       - Understanding of previous games mechanics even better than designer.
       - Provide features wish list from dedicated players.
     + **Disadvantages**:
       - Difficult for a designer to face the criticism from players on their development choices.
       - Difficult to assess what players say and what they do.

**Chapter 5 Indoor Visibility processing**