Introduction to Game Design and Animation

Chapter 1: Fundemental Elements of Game Design

Game Design

- The craft of creating the structure and rules of a game to make enjoyable and engaging.
- Shaping the player experiences and interactions within a virtual world.
- Basic elements of game design:
  - Player abilities
  - Challenges
  - Environment

Game Mechanics

- Rules and systems that govern how the game works and how players interact with it.
- Components:
  - `Rules` : established guidelines that define the structure of the game for a fair and consistent gaming experience.
  - `Player interaction` : how players engage with the game.

Gameplay

- Overall experience of playing a game, encompassing the interactions, challenges, and activities within the game world.
- How to enhance Gameplay:
  - `Immersion` : level of engagement and absorption a player feels within the game world.
  - `Emotion` : gameplay that evokes emotions that contributes to memorable gaming experience.
  - `Player Agency` : the degree of control and influence a player has over the game.

Game Types

- Action, RPG, Strategy…

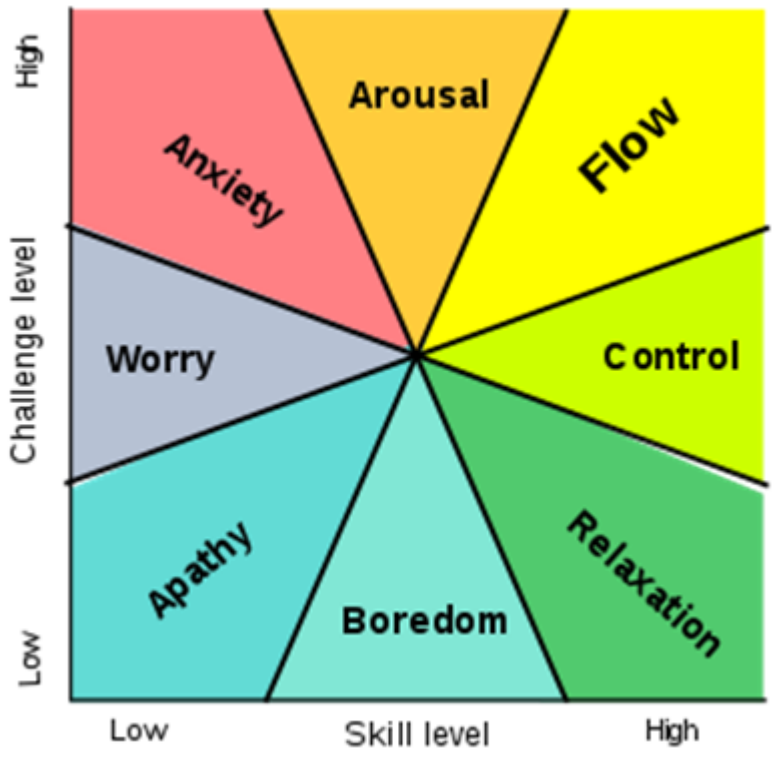
NPCs and Open World Games

- **NPC**: characters controlled by the game's AI to contribute to the game's story, challenges or provide assistance.
- **Open world games**: games with huge virtual environment and allows players to explore freely.

History of Games

- Console output -> GUI -> Console -> 3D games -> VR

Skill level in games



Hardware for Games

- Internal hardware:
  - GPU
  - CPU
  - RAM
  - SSD/HDD
  - PPU

- External hardware:
  - PlayStation
  - Xbox
  - Nintendo

Software for Games

- **Game APIs**:
  - Set of tools and protocols that allow software applications to communicate with each other.
  - `DirectX`：
    - prominent API, particularly for Windows-based gaming.
    - `Unified Graphics and Multimedia`：DirectX provides a comprehensive set of APIs for graphics, audio, and multimedia tasks.
    - `Compatibility`：It offers a standardized interface, ensuring games can run efficiently on various hardware configurations.
- **Game Engines**:
  - Software frameworks that simplify the creation of games by providing pre-built components, rendering systems, and scripting languages.
    Example: Unity, Unreal, Godot…

Essential Math for Game Design

- Pythagorean Theorem
- Angles and Trignometry
- Vectors

Consideration for Game Rule

- **Iteration and Prototyping**: In the game rule creation process,iteration and prototyping are essential for refining mechanics and identifying improvements.
- **Game Design Core Ideals**: Aligning rules with core game design ideals, including **player ability**, **environment**, and **challenges**, ensures a harmonious gaming experience.
- **Test (Build-PlayFeel-Modify) Repeat**: The iterative loop of building, playing, modifying, and repeating allows for continuous refinement based on player feedback.
- **Harmony**: Central to successful game rules is the pursuit of fairness, preventing frustration or favoritism, and maintaining a power balance for a satisfying and challenging gameplay.
- **Shapley Value and Harsanyi Dividend**:
  - `Shapley Value`：Evaluates the contribution of each player to a coalition, considering all possible player permutations.
  - `Harsanyi Dividend`：Focuses on the subset of players who have the potential to form a coalition, assessing their contributions.

- **Nash equilibrium**: is a decision-making theorem within game theory that states a player can achieve the desired outcome by not deviating from their initial strategy. When we are only considering 2 players it is called coordination. (if 3 or more only collaboration).
    - `Zero Sum Games` : is a situation in game theory where one person's gain is equivalent to another's loss, they are called zero-sum because the total amount of value in the game remains constant. The only thing that changes is its distribution among the players. Gambling, Trading etc…

Gameplay Optimization

- **Ludonarrative Dissonance**: describes the conflict between a game's narrative told through the story and the narrative told through the gameplay.
- **Emergent Gameplay**: is a concept where the game is designed to allow players to approach tasks from a variety of angles, rather than forcing a specific solution this often leads to unexpected outcomes, adding to the replayability of the game.
- **The User Interface (UI) and Audio design**: An intuitive and aesthetically pleasing UI can contribute to a seamless gaming experience. Similarly, well-designed audio, especially surround sound, can provide players with important auditory cues and create a more immersive environment.
- **Challenge Design**: involves creating tasks or obstacles for the player to overcome.
- **Player Preferences**: involves recognizing the diverse preferences players have for different genres and game styles, as well as understanding the appeal of both multiplayer and single-player modes, and catering to these varied interests.

Level Design

- **Sequence of Challenges**: Crafting a compelling progression of challenges to engage players gradually.
- **Player Objective**: Clearly defining the player's purpose within each level for enhanced gameplay direction.
- **Reward**: Strategically placing rewards to serve as motivational milestones, reinforcing player accomplishments.
- **Player Preference Matrix**: Understanding player preferences is essential for creating a tailored gaming experience.

Game Design Principles

Game Conceptualization

- Defining a Game Concept:
  - Identify Target Audience and Genre
  - Brainstorming Game Ideas
- Concept Refinement:
  - **Create a Game Design Document(GDD)**: a guiding document for development process. Itcontains `Introduction`, `Goals`, `Game World`, `Mechanics`, `Game Elements`, `User Interface`, `Audio` and `Prototype` of the game to be developed.
  - Iterative Conceptualization and Feedback

Storytelling and Narrative in Games

- Importance of Narrative in Games:
  - Establishing compelling story
  - Understanding player agency
- Narrative Structures:
  - `Linear Narrative` : has a chronological path.
  - `Non-linear Narrative` : not chronological.
- Storytelling Techniques:
  - Branching Narrative by Player Choice
  - Environment
  - Non-Linear
  - Character development

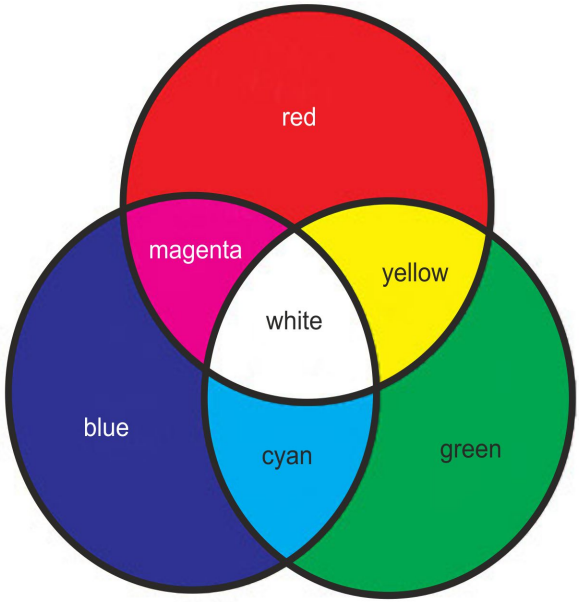
Storyboard

- Visual representation of a narrative, concept, or script, divided into sequential scenes (panels).
- Key Components:
  - `Title` : a title that briefly describes the scene or action taking place.
  - `Sketches` : visual representation of each frame or scene.
  - `Dialog/Action` : description of the action, dialogue, or any relevant information about what's happening in the scene.
  - `Specs` : details like shot size, lens length, two-shot and arrows used to show camera and/or character movement or how each shot connects to the next.

Fundementals of Computer Graphics

- **Pixels**: smallest unit of a digital image. The higher density of pixels results in sharper visuals.
- **Resolution**: defines the clarity and detail of an image. Expressed as the total number or density of pixels in the image.

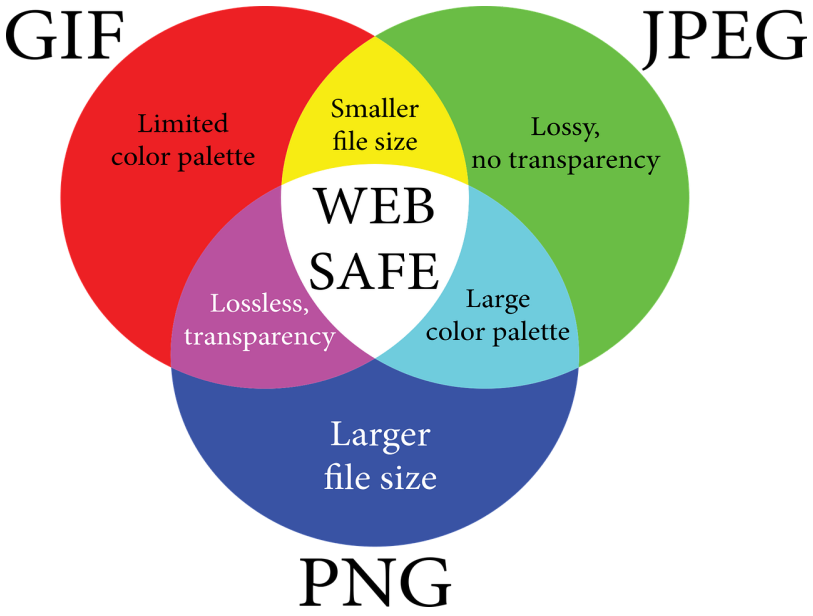
- **Color Models**: RGB (Red, Green and Blue) [0 - 255] and CMYK (Cyan, Magneta, Yellow, Black) [0 - 100].



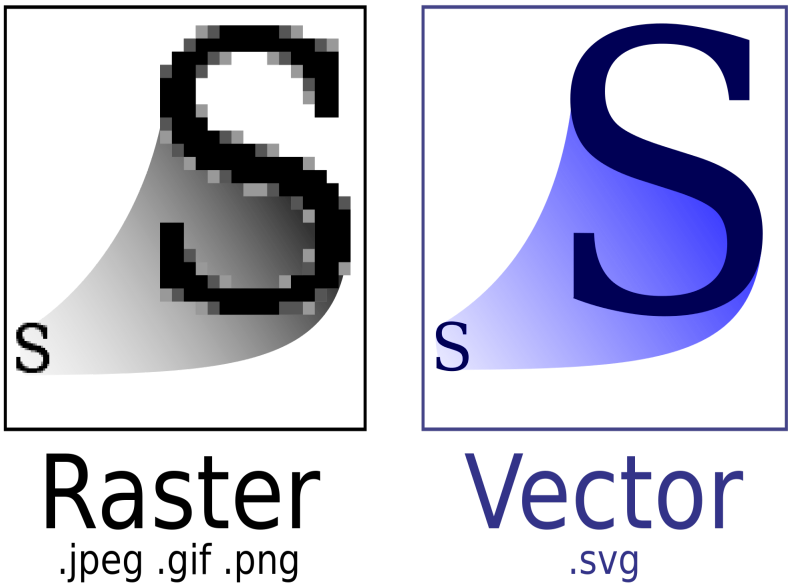
- **Rendering Pipeline**: step by step process of creating visuals in a game. It includes `geometry processing`, `rasterization` and `pixel shading`.

- **Graphic File Formats**:

  - Common Image Formats



  - Scalable Vector Graphics (SVG): Unlike raster graphics, which use pixels, vector graphics use mathematical equations to represent images.



Principles of 2D graphics

- **Composition and Layout**: refers to the arrangement and organization of visual elements in 2D space.
  - Principles:
    - `Symmetry` : a powerful tool in design that helps achieve balance. It involves mirroring elements on either side of an axis, creating a sense of harmony and order.
    - `Balance` : the distribution of visual weight within a design. It can be achieved through color, shape, size, and position. Balance can be symmetrical (evenly distributed) or asymmetrical (unevenly distributed but still balanced).
    - `Focal Points` : draw the viewer's attention to a specific area of the design. They can be created through contrast, placement, or use of lines. Focal points guide the viewer's eye through the design.
- **Typography and Font Selection**: Typography is the art and technique of arranging type to make written language legible, readable, and visually appealing.
- **Sprite Animation**:
  - Frame-by-Frame Animation: This technique involves creating the illusion of motion by displaying a sequence of individual frames.
  - Principles of Motion in 2D:
    - `Easing` : control exactly how quickly and smoothly something changes in a video.
    - `Anticipation` : the action before the main action.
    - `Follow-Through` : animating people and objects in such a way that their movement appears lifelike and smooth.

Basics of 3D space

- Cartesian Coordinates in 3D
- Camera Perspectives in 3D
- 3D modeling

Programming Languages for Games

Scripting vs. Compiled Languages

Scripting Languages	Compiled Languages
Flexible	Rigid
Game logic and AI scripting	Resource intensive tasks like graphics rendering
Slower	Faster
Python, Lua	C++, C#

Object Oriented Programming in games

- `Classes` and `Objects` used to represent the blueprint for different game entities.
- `Inheritance` and `Polymorphism` used to reuse properties and behaviors of one entity to other entity.

Design Patterns for games

- `Singleton Pattern` : A “GameManager” class ensures a single instance controls global game state. This prevents multiple conflicting instances and simplifies access to crucial game functionalities.
- `Observer Pattern` : Implement an “EventSystem” using the observer pattern. Game elements (observers) subscribe to events (subjects), reacting to changes like player input or scoring updates.
- `State Pattern` : Manage game states (e.g., menu, playing, paused) using the state pattern. Each state, like “MenuState” or “PlayingState,” encapsulates specific behaviors, simplifying transitions and enhancing maintainability.

Data Structures and Algorithims used in games

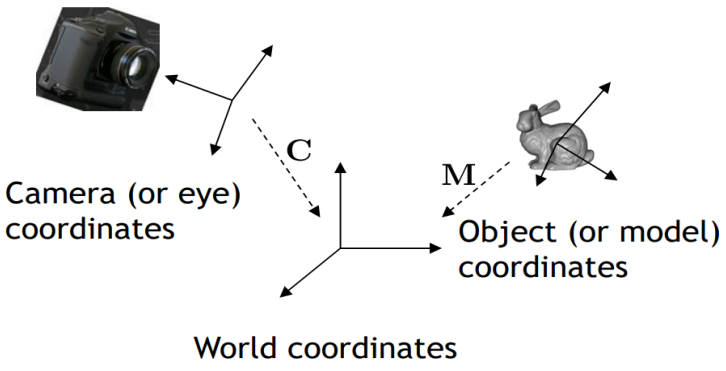
- **Data Structures**:
  - Arrays
  - Linked List
  - Trees

- **Algorithms**:
  - Sorting Algorithms
  - Searching Algorithms
  - Pathfinding Algorithms
  - Collection detection Algorithms

Chapter 2: Basic 3D theory

Coordinates

- Representation of 3D positions, shapes and transformation in `x`, `y` and `z` axis.
- Types of coordinates in game design and development:
  - **World Coordinates**: represent positions and shapes in the global coordinate system. This Sets up the scene and positioning objects in relation to the world origin.
  - **Object Coordinates**: local coordinates relative to object's own origin and orientation.
  - **Eye (Camera) Coordinates**: represent positions relative to the camera or viewer. Defines what is visible in the scene from the camera's perspective.
  - **Normalized Device Coordinates**: maps coordinates to the screen space for further processing. Ranging from -1 tp 1 in each dimension in 3D.
  - **Screen Coordinates**: represent positions on the 2D screen or window. Origin can be at the top-left corner and represents Final stage before rendering on the display. Optimizes the aspect ratio consideration and also the window size of running app.



Objects

- 3D Entity built by using vertices.
- **Vertex**:
  - a point in space having its own 3D position in the coordinate system and usually some additional information that defines it.

  - dscribed by:

    - `position` : identifies it in `x`, `y` and z axis.
    - `color` : holds RGBA form 0.0 to 1.0.

    - `normal` : describes the direction of the vertex.
    - `texture` : 2D image that the vertex can use to decorate the surface.
- **Face**: a plane between vertices.
- **Geometry**: `vertex + face` .
- **Mesh**: `material (texture) + geometry` .
- **Euler's Formula**: for all shapes whose side doesn't overlap and their vertices aren't touching:

$$F(faces) + V(Vertices) - E(Edges) = 2$$

- **3D Models**:
    - digital representation of objects that populate our virtual worlds.
    - Constructed using fundamental shapes like triangles and squares, also known as `quads` .

Motion

- **Vector**: is basically a Nx1 matrix where N is the vector's number of components (also known as an N-dimensional vector).

$$v(x, y) = [x, y]$$

- **Identity Matrix**: an NxN matrix with only 0s except on its diagonal.

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Basic Transformation**:
    - **Linear Transformation**: transformation that doesn't affect the origin.
        - Scaling: increasing the size of the 3D model by some factor (scale).
            - `Uniform Scaling` : scaling matrix contains same values for `x` , `y` and `z` axis.

$$\begin{pmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ x \\ x \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot x \\ S_3 \cdot x \\ 1 \end{pmatrix}$$

            - `Non-uniform Scaling` : scaling matrix contains different values for `x` , `y` and `z` axis.

$$\begin{pmatrix} S_1 & 0 & 0 & 0 \\ 0 & S_2 & 0 & 0 \\ 0 & 0 & S_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} S_1 \cdot x \\ S_2 \cdot y \\ S_3 \cdot z \\ 1 \end{pmatrix}$$

- **Rotation**:
  - `Along x axis` :

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \cdot cos(\theta) - z \cdot sin(\theta) \\ y \cdot sin(\theta) + z \cdot cos(\theta) \\ 1 \end{pmatrix}$$

  - `Along y axis` :

$$\begin{pmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot cos(\theta) - y \cdot sin(\theta) \\ x \cdot sin(\theta) + y \cdot cos(\theta) \\ z \\ 1 \end{pmatrix}$$

  - `Along z axis` :

$$\begin{pmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot cos(\theta) + z \cdot sin(\theta) \\ y \\ -x \cdot sin(\theta) + z \cdot cos(\theta) \\ 1 \end{pmatrix}$$

- **Shear Transformation**: transformation that affect the origin.
  - **Translation**: process of adding another vector on top of the original vector to return a new vector with a different position.

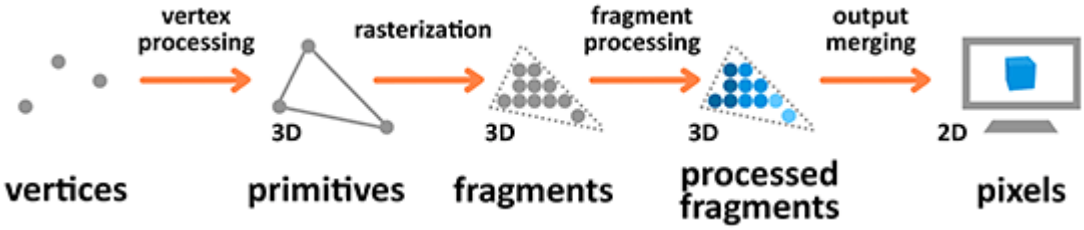
$$\begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} T_x + x \\ T_y + y \\ T_z + z \\ 1 \end{pmatrix}$$

- **Concatination of Matrices**: combining 3D transformations using matrix multiplication.
  `Scaling -> Rotation -> Translation` .

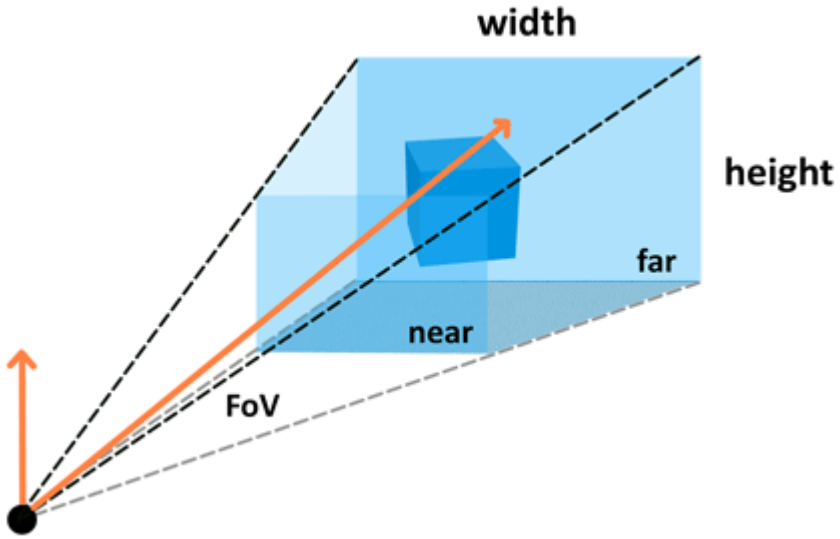
Rendering Pipeline

- process by which images are prepared and output onto the screen.
- It takes the 3D objects built from `primitives` which are described using `vertices` , applies processing, calculates the `fragments` and renders them on the 2D screen as `pixels` using a computer program called `shaders` .

- Terms:
  - **Primitive**: An input to the pipeline built from vertices and can be a triangle, point or line.
  - **Fragment**: A 3D projection of a pixel, which has all the same attributes as a pixel.
  - **Pixel**: A point on the screen arranged in the 2D grid, which holds RGBA color.

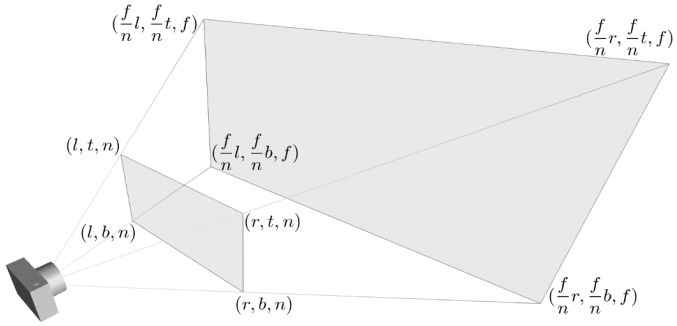


- Stages:
  - **Vertex processing**:
    - combinig the information about individual vertices into primitives and setting their coordinates in the 3D space for the viewers to see.
    - transformations:
      - **Model transformation**: arranging the objects in the world.
      - **View transformation**: positioning and setting the orientation of the camera in the 3D space.
      - **Projection transformation (Perspective transformation)**: defines camera settings such as `field of view (FOV)`, `aspect ratio` and optional `near` and `far` planes to setup what can be seen by the camera.



      - **Viewport transformation**: outputting everything for the next step in the rendering pipeline.

- **Rasterization**:
  - process to align fragments (3D projection of the 2D pixels) to the pixel grid.
    $V_{world} = T_{world} \cdot V_{object} ..... applying\ translation$
    $V_{eye} = R_{w-to-eye}(\pi/4) \cdot S_{w-to-eye}(2,1,1) \cdot V_{world} .. applying\ rotation\ and\ scaling$
    $V_{clip} = V_{perspective}(2) \cdot V_{eye} ..... applying\ perspective\ projection$
    $V_{NDC} = V_{clip}/V_{clip} \cdot w ..... converting\ to\ NDC$
    $V_{screen} = S_{screen}(width, height) \cdot V_{NDC} ..... converting\ to\ screen\ coordinates$
  - Canonical View Volume (CVV):
    - is a standardized 3D volume used as a reference for mapping scenes into 2D space.
    - centered at the origin with sides of length 2.
    - referred to as the image space.
    - defined by the six planes `top`, `bottom`, `left`, `right`, `near` and `far`.
    - provides a standardized space for transformations and projections.
    - simplifies the mapping of 3D objects to 2D screens.
  - projections:
    - **Perspective projection**: simulates the way objects appear smaller as they move away from the viewer with a sense of depth.



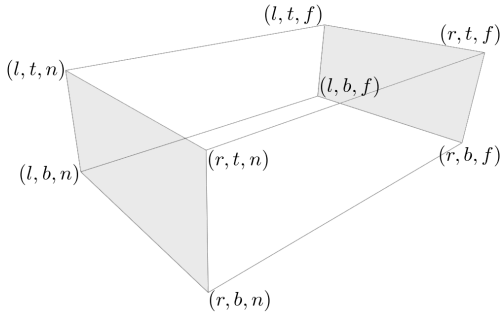
$$P_{perspective} = \begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-f-n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$r = n \cdot tan(\frac{fov}{2})$$

$$t = r \cdot aspect\,ratio$$

$$P_{perspective} = \begin{pmatrix} \frac{1}{tan(\frac{fov}{2}) \cdot aspect\ ratio} & 0 & 0 & 0 \\ 0 & \frac{1}{tan(\frac{fov}{2})} & 0 & 0 \\ 0 & 0 & \frac{-f-n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

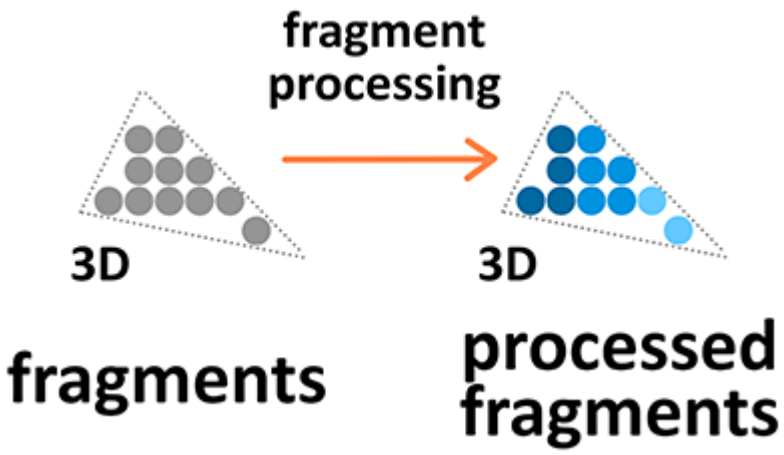
- **Orthographic projection**: type of projection where parallel lines remain parallel after projection. The size of objects does not change with their distance from the viewer.



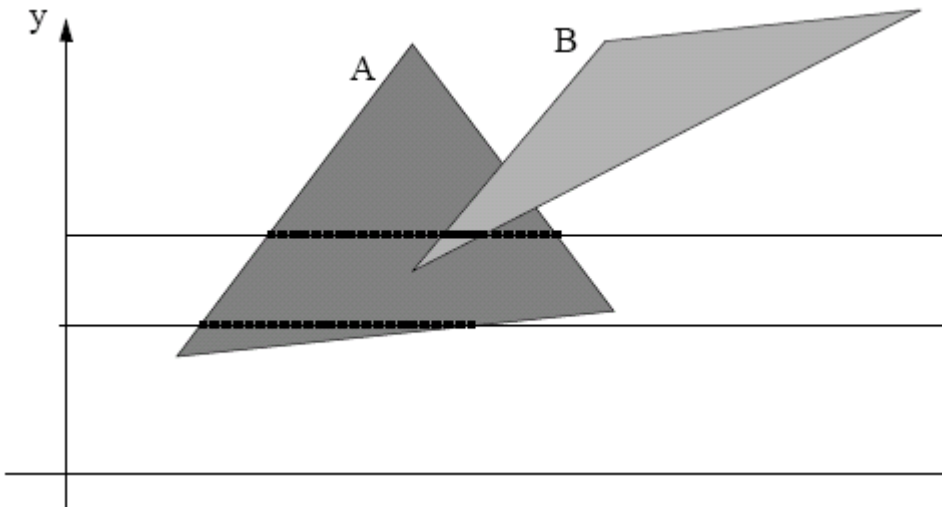
$$P_{ortho} = \begin{pmatrix} \frac{2}{r-1} & 0 & 0 & \frac{-r-1}{r-1} \\ 0 & \frac{2}{t-b} & 0 & \frac{-t-b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & \frac{-f-n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Fragment processing**:
  - focuses on textures and lighting to calculate final colors based on given parameters.
  - terms:
    - Textures: 2D images used in the 3D space to make the objects look better and more realistic.
    - Texels: combination of texture elements.
  - techniques:
    - Texture wrapping: allows us to repeat the 2D image around the 3D object.
    - Texture filtering: applied when the original resolution or the texture image is different from the displayed fragment. It will be minified or magnified accordingly.
    - Lighting:
      - absorbtion or reflection of light to produce color when light source interacts with the surface colors of the object's material.

        - types:
            - `Diffuse` : A distant directional light, like the sun.
            - `Specular` : A point of light, just like a light bulb in a room or a flashlight.
            - `Ambient` : The constant light applied to everything on the scene.
            - `Emissive` : The light emitted directly by the object.

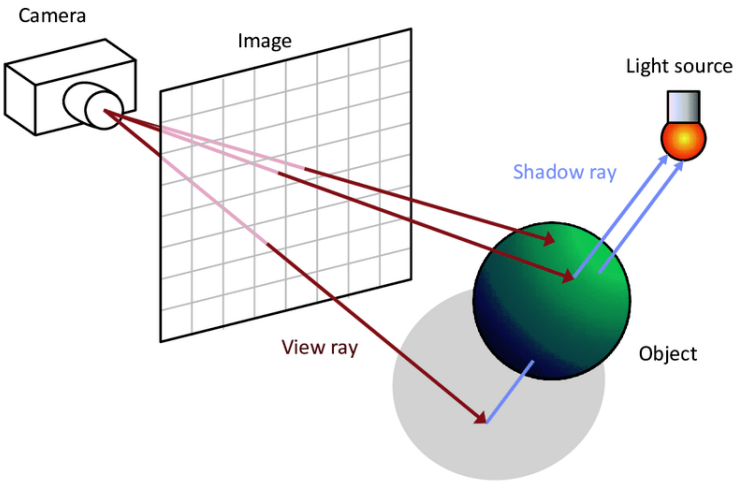


- **Output merging**:
    - all the fragments of the primitives from the 3D space are transformed into a 2D grid of pixels that are then printed out on the screen display.
    - some processing is also applied to ignore information that is not needed (parameters of objects outside the screen or behind other objects).
    - **Scanline Rendering**:
        - is a technique that processes a scene one scanline (horizontal line of pixels) at a time.
        - it involves determining which polygons intersect with the current scanline and then filling the pixels between the intersections.
        - commonly used in 3D graphics to fill polygons and apply textures efficiently.
        - rasterization-based approach that is well-suited for real-time rendering.

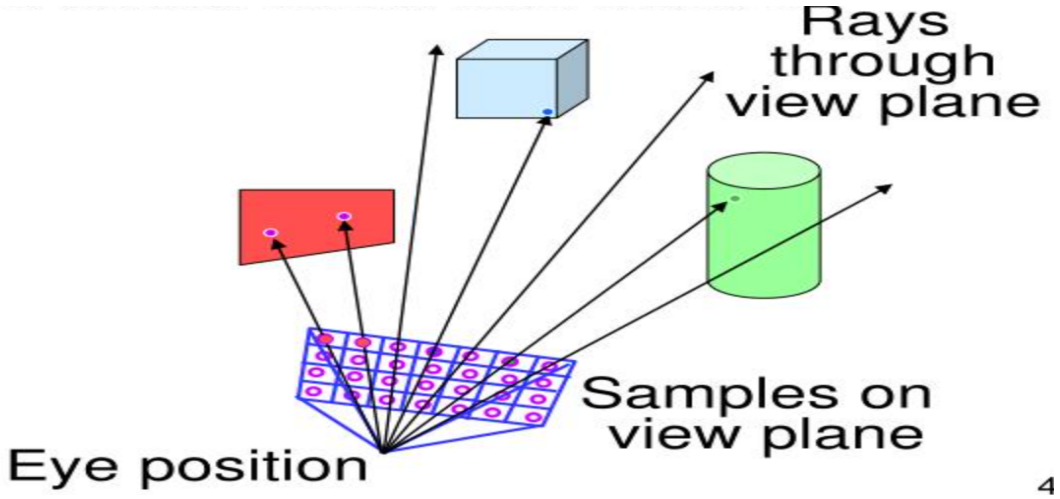


Other methods than Rasterization for Rendering

- **Ray Tracing**:
  - a rendering technique that simulates the way light interacts with objects in a scene.
  - It traces the path of rays of light as they travel through the scene, simulating effects like reflections, refractions, and shadows with high realism.
  - Computational intensive but highly realistic images.

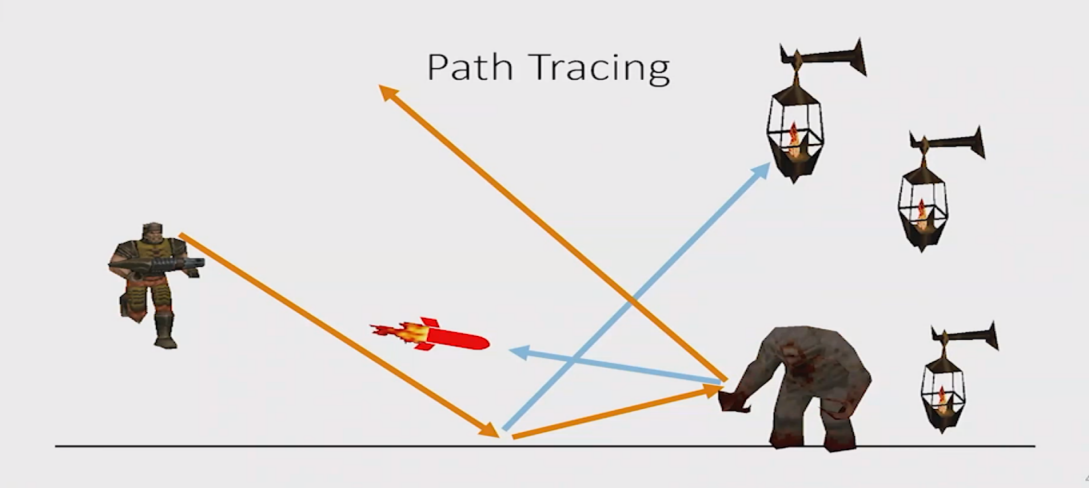


- **Ray Casting**:
  - casts rays from the eye (camera) and checking for intersections with objects in the scene.
  - used for basic rendering tasks.
  - doesn't simulate complex lighting effects like reflections or refractions.
  - used in applications where computational efficiency is more critical than achieving photorealistic graphics.



- **Path tracing**:
  - rendering algorithim that extends the principles of ray tracing by simulating the behavior of light rays as they travel through a scene.
  - It traces paths of light, considering multiple bounces and interactions with surfaces.
  - It is known for its ability to produce highly realistic images by accurately simulating global illumination effects, including indirect lighting (Smooth Shadow).

- It is widely used in computer graphics for generating photorealistic scenes.



Ray tracing vs Path tracing

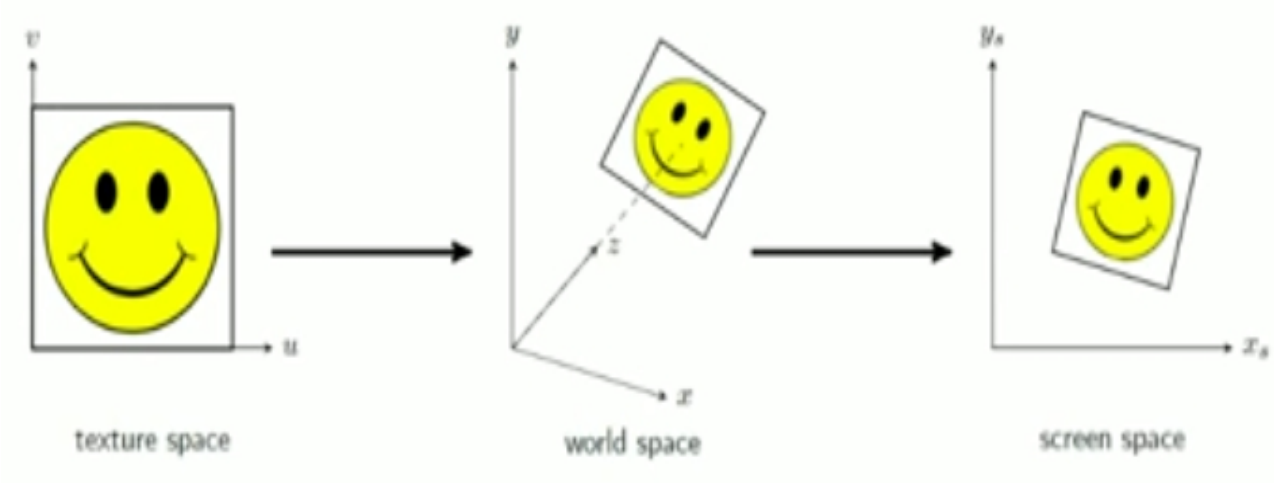
- **Radiosity**:
  - A global illumination algorithm that simulates the way light reflects and diffuses among surfaces in a scene.
  - It calculates the exchange of light energy between surfaces to model indirect lighting.
  - It is often used to achieve realistic lighting in architectural visualization and simulations where accurate representation of how light interacts with surfaces is essential.
- **Lightmaps and Baking**:
  - Lightmaps are precomputed textures that store lighting information for surfaces in a 3D scene.
  - Baking refers to the process of calculating and storing lighting data in these lightmaps.
  - By precomputing lighting information, rendering performance can be improved, especially in real-time applications.
  - `Lightmaps` are commonly used in games to simulate realistic lighting without the need for dynamic calculations in real-time.

Wire-frame model

- **Wire-frame model** a visual representation of a three- dimensional (3D) physical object used in 3D computer graphics.
- It is created by specifying each edge of the physical object where two mathematically continuous smooth surfaces meet, or by connecting an object's constituent vertices using (straight) lines or curves.
- tables:
  - Vertex Table: consists of three-dimensional coordinate values for each vertex with reference to the origin.
  - Edge Table: specifies the start and end vertices for each edge.

Chapter 3: Textures

- **Texture Mapping**:
  - A graphic design process in which a two dimensional surface called a `texture map` is wrapped around a three dimensional object.
  - The process of mapping a 2D image onto a 3D polygon.
  - The image that is mapped onto the polygon is called a `texture`.
  - A pixel in the texture is called a `texel`.
  - The colors pixels in the raster are determined by the colors of the texels in the texture.
  - Originally referred to `diffuse mapping`, a method that simply mapped pixels from a texture to a 3D surface ("wrapping" the image around the object).



Texture Space

- The space in which the texture exists.
- It is a 2D space with horizontal and vertical axes denoted by $u \in [0, 1]$ and $v \in [0, 1]$.
- The coordinates of a texel (U, V) can be expressed (scaled) as

$$U = \lfloor M_x \cdot u + 1 \rfloor$$
$$V = \lfloor M_y \cdot v + 1 \rfloor$$

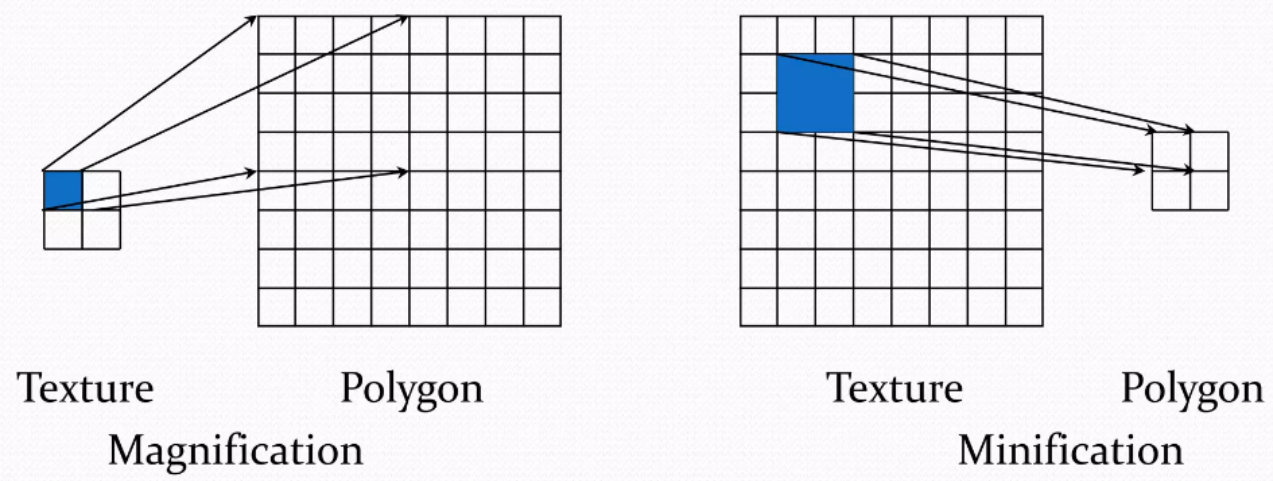
n = number of texels
U = texel coordinate on the x axis $\in [1, n]$
V = texel coordinate on the y axis $\in [0, n]$
u = texture coordinate on the x axis $\in [0, 1]$
v = texture coordinate on the y axis $\in [0, 1]$
M_x = the number of texel in horizontal direction
M_y = the number of texel in vertical direction

Applying a texture map

Calculating scan extreme coordinates

The basics of texturing

- **Texture repeating**:
  - Also known as `tiling`, is a technique used in computer graphics to repeat a texture map across a surface of a 3D object seamlessly.
  - When a texture is applied to a surface, it may not cover the entire surface uniformly, especially if the surface is larger than the texture map itself.
  - Texture repeating allows the texture to seamlessly repeat across the surface of the object without noticeable seams or distortions.
- **Minification and Magnification**:
  - Minification:
    - The process of displaying a texture on a smaller area of the screen than the original size of the texture map.
    - This happens when a textured object is rendered at a distance, and the pixels of the texture cover fewer screen pixels.
  - Magnification:
    - The process of displaying a texture on a larger area of the screen than the original size of the texture map.
    - This happens when a textured object is rendered close to the camera, and the pixels of the texture cover more screen pixels.



Mipmaps

- A common solution to the use of big textures being squashed into tiny primitives involves the use of mipmaps.

- These are scaled down versions of the original texture that can be generated by the game engine itself (by using the relevant API command to make them) or pre-made by the game designers.
- Each level of mipmap texture has half the linear dimensions of the previous one.
- So for the crate texture, it goes something like this: 256 x 256 → 128 x 128 → 64 x 64 → 32 x 32 → 16 x 16 → 8 x 8 → 4 x 4 → 2 x 2 → 1 x 1.

Determining Mipmap

- The mipmaps are all packed together, so that the texture is still the same filename but is now larger.
- The texture is packed in such a way that the u,v coordinates not only determine which texel gets applied to a pixel in the frame, but also from which mipmap.
- The programmers then code the renderer to determine the mipmap to be used based on `the depth value of the frame pixel`, i.e. if it is `very high`, then `the pixel is in the far distance`, so a `tiny mipmap` can be used.

Mipmap Size

- No matter how you pack in the smaller textures, the overall increase to at least one of the texture's dimensions is 50%.
- But this is only true for pre-made mipmaps; if the game engine is programmed to generate them as required, then the increase is never more than 33% than the original texture size.
- So for a relatively small increase in memory for the texture mipmaps, you're gaining performance benefits and visual improvements.

Bilinear, Trilinear, and Anisotropic

- The process of selecting a pixel from a texture, to be applied to a pixel in a frame, is called `Texture Sampling`, and in a perfect world, there would be a texture that exactly fits the

primitive it's for regardless of its size, position, direction, and so on.

- It's a straight 1-to-1 texel-to-pixel mapping process.
- Since that isn't the case, texture sampling has to account for a number of factors:
    - Has the texture been magnified or minified?
    - Is the texture original or a mipmap?
    - What angle is the texture being displayed at?
- **Texture filtering**:
    - A technique used in computer graphics to improve the quality and appearance of textured surfaces, especially when textures are applied to 3D objects and rendered onto a 2D screen.
    - Texture filtering helps to reduce artifacts such as aliasing and pixelation that can occur during the process of mapping textures onto surfaces.
    - There are several types of texture filtering techniques, including:
        - **Point Sampling (Nearest Neighbor)**:
            - The simplest form of texture filtering.
            - Each pixel on the textured surface is mapped to the nearest texel (texture pixel) in the texture map.
            - Can result in pixelation and blocky images, especially during texture magnification.
        - **Linear Filtering**:
            - Linear filtering interpolates between texels along only one axis (either horizontally or vertically) to determine the final color of a pixel on textured surface.
            - It computes the weighted average of the two texels nearest to the pixel being rendered.
            - Linear filtering can be applied independently along the x-axis (horizontal) or the y-axis (vertical).
              $T_f = (1-\alpha)(1-\beta)T_1 + (\alpha)(1-\beta)T_2 + (1-\alpha)(\beta)T_3 + (\alpha)(\beta)T_4$
            - Linear filtering can lead to visual artifacts, such as aliasing, particularly when textures are significantly magnified or minified.
        - **Bilinear Filtering**:
            - A more advanced filtering technique that improves texture quality during both minification and magnification.
            - Samples four texels surrounding the pixel being rendered and interpolates between them to determine the final color.
            - Helps smooth out transitions between texels, reducing pixelation and providing a smoother appearance.
        - **Trilinear Filtering**:
            - Builds upon bilinear filtering by blending between different mipmap levels.
            - Mipmaps are precomputed versions of the texture at various levels of detail.
            - Trilinear filtering selects texels from adjacent mipmaps to minimize artifacts during minification and magnification.

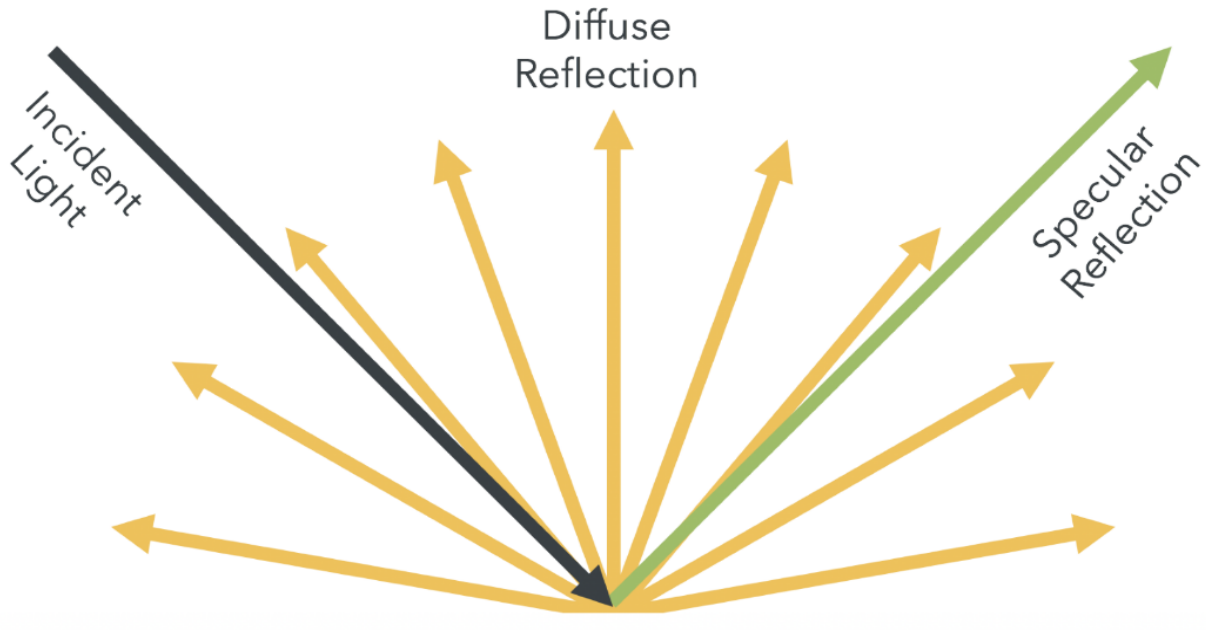
        - **Anisotropic Filtering**:
            - Particularly useful for textures viewed at oblique angles.
            - Anisotropic filtering samples texels in a non-uniform manner, providing better quality for textures viewed from different perspectives.
            - It reduces blurriness and maintains detail, especially in textures with directional patterns.
    - Texture filtering techniques are often implemented in hardware by graphics processing units (GPUs) to accelerate rendering and improve visual quality in real-time applications such as video games and interactive simulations.
    - The choice of texture filtering method depends on factors like performance constraints, desired visual quality, and the characteristics of the textures being used.

Chapter 4: Lighting

- `Computer graphics lighting` is the collection of techniques used to simulate light in computer graphics scenes.
- If the white light is shined on a red ball, the ball absorbs all the red light and reflects all the other colours. This is one of basic rules of optics.
- If we shine blue or green light on a red ball it doesn't absorb any of the light. Because there is no red to absorb. So it reflects all of it making appear black.

Diffuse and Specular Reflection

- `Diffuse Reflection` when light hits rough surface It reflects in all directions.
- `Specular Reflection` when light hits smooth surface It reflects in same direction.



Types of Diffuse Reflection

- **Lambertian**:
  - type of diffuse reflection where the intensity of the reflected light is the same regardless of the angle of observation.
  - This type of reflection occurs on surfaces that scatter light uniformly in all directions, following Lambert's cosine law.
  - Assumes perfectly Defuse Surface. No highlights and Shadows.
- **Oren-Nayar**:
  - Takes into account roughness of Surface index. More natural reflection with highlights and Shadows.
- **Phong**:
  - Also takes into account the shininess of the surface. Very natural looking with highlights, shadows and Specular Reflections.

Lambertian Equation

- Simple equation that calculates the intensity of the light reflected by an object.

  $$ID = C \cdot I \cdot N \cdot L \cdot \cos\theta$$

  C = color (diffuse reflectance coefficient)
  I = intensity of light source
  N = normal vector
  L = light vector (Unit Vector)
  θ = angle between N and L

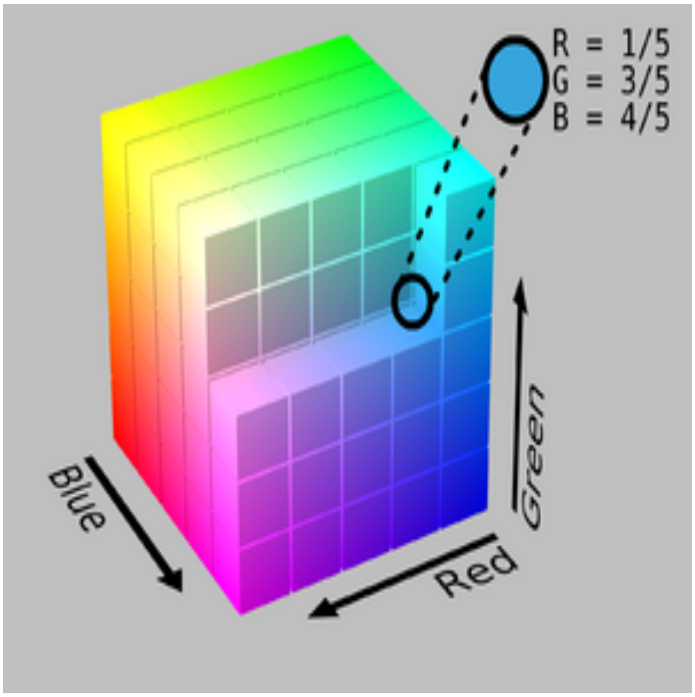
Diffuse reflection equation

- $$ID = C \cdot I \cdot N \cdot L \cdot \cos\theta(NL)$$

  C = color (diffuse reflectance coefficient)
  I = intensity of light source
  N = normal vector
  L = light vector (Unit Vector)
  θ = angle between N and L

RGB color space

- RGB cude is a 3D $n \times n \times n$ cube where the x, y, and z axis represented as Red, Blue and Green.



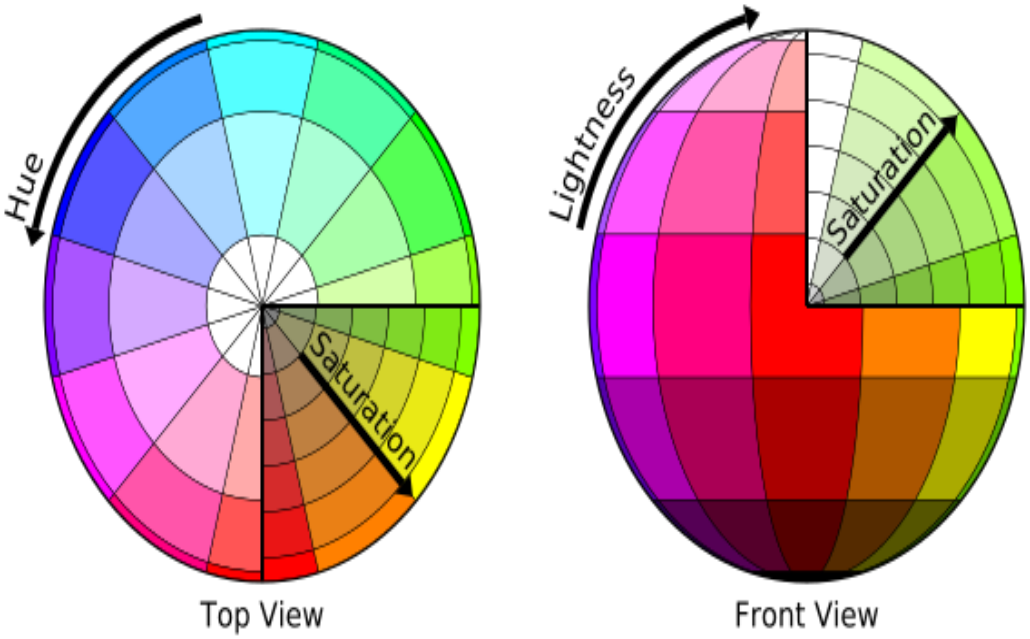
Gamma Correction

- A technique used in computer graphics and image processing to compensate for the non-linear relationship between the pixel values in an image and the actual brightness perceived by the human eye.
- It adjusts the brightness of an image to make it appear more natural to our eyes.
- A gamma value is sometimes called an `encoding gamma`, and the process of encoding with this `compressive power-law nonlinearity` is called `gamma compression`; conversely, a gamma value is called a `decoding gamma`, and the application of the `expansive power-law nonlinearity` is called `gamma expansion`.

$$encoded = (\frac{original}{255})^{\frac{1}{\gamma}} \times 255$$
$$original = (\frac{encoded}{255})^{\gamma} \times 255$$

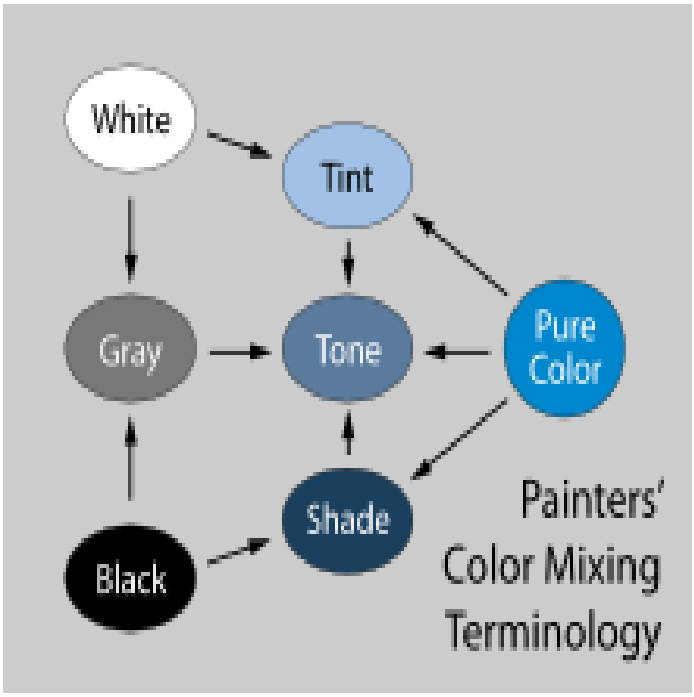
HSL representation of color

- An extension of the color wheel.
- Colors nearest the center or the poles are most `achromatic`.
- Colors of the `same lightness and saturation`, but of `different hue`, are called `nuances`.
- Colors of the `same hue and saturation`, but of `different lightness`, are called `tints` and `shades`.
- Colors of the `same hue and lightness`, but of `different saturation`, are called `tones`.



Hue Manipulation

- A `tint` is a mixture of a `color with white`, which `reduces darkness`, while a `shade` is a mixture `color with black`, which `increases darkness`.
- Both processes affect the resulting color mixture's relative saturation.
- A `tone` is produced either by mixing a `color with grey`, or by `both tinting and shading`.
- Mixing a color with any neutral color (including black, gray, and white) reduces the chroma, or colorfulness, while the hue (the relative mixture of red, yellow, green, etc. depending on the colorspace) remains unchanged.



- **Monochromatic colors**: are all colors (tint, tones, and shades) of a single hue.

- **Hue**:
  - The degree to which a stimulus can be described as similar to or different from stimuli that are described as red, orange, yellow, green, blue, purple.
  - One of the main properties (called color appearance parameters) of a color.
- **Color gradient**:
  - A color gradient specifies a range of position dependent colors, usually used to fill a region.
  - The colors produced by a gradient vary continuously with position, producing smooth color transitions.
  - A color gradient is also known as a `color ramp` or a `color progression`. In assigning colors to a set of values, a gradient is a continuous color map, a type of color scheme.
  - An axial color gradient (sometimes also called a `linear color gradient`) is specified by two points, and a color at each point
  - The colors along the line through those points are calculated using `linear interpolation`, then extended perpendicular to that line.
  - Both `CSS` and `SVG` support color gradient.

Surface Shading Algorithms

- **Flat shading**:
  - A technique that shades each polygon of an object based on the polygon's "normal" and the position and intensity of a light source.
  - It calculates the color of each polygonal face (or flat surface) independently, typically using the color of the surface itself or the color of the light source illuminating it. The entire polygon is rendered with this color, regardless of the orientation of its individual facets.
- **Smooth shading**:
  - In contrast to flat shading where the colors change discontinuously at polygon borders, with smooth shading the color changes from pixel to pixel, resulting in a smooth color transition between two adjacent polygons.
  - Usually, values are first calculated in the vertices and bilinear interpolation is used to calculate the values of pixels between the vertices of the polygons.
  - Types of smooth shading:
    - Gouraud Shading:
      - Determine the normal at each polygon vertex.
      - Apply an `illumination` model to each vertex to calculate the light intensity from the vertex normal.
      - Interpolate the vertex intensities using `bilinear interpolation` over the surface polygon.

        - Phong Shading:
            - Phong shading, an improvement over Gouraud shading, calculates the surface normal at each pixel by interpolating vertex normals across the polygon's surface. This allows for more accurate lighting calculations and specular highlights, resulting in smoother and more realistic-looking surfaces.
    - Problems:
        - Due to lighting being computed only at vertices, the inaccuracies (especially of specular highlights on large triangles) can become too apparent.
        - `T-junctions` with adjoining polygons can sometimes result in visual anomalies.

Bilinear Interpolation

- `Bilinear interpolation` is an extension of linear interpolation for interpolating functions of two variables (e.g., x and y) on a rectilinear 2D grid.
- Bilinear interpolation is performed using linear interpolation first in one direction, and then again in the other direction. Although each step is linear in the sampled values and in the position, the interpolation as a whole is not linear but rather quadratic in the sample location.
- Bilinear interpolation is one of the basic resampling techniques in computer vision and image processing, where it is also called bilinear filtering or bilinear texture mapping.

What are shaders?

- A set of instructions to the GPU which is executed all at once for every pixel on the screen.
- A shader is given an input pixel at certain position (x, y) and it returns the new color(R,G,B) shader transformation for that pixel.
- `Cel shading` or `toon shading` is a type of `non-photorealistic` rendering designed to make 3D computer graphics appear to be flat by using less shading color instead of a shade gradient or tints and shades.
- In Cel-shading conventional smooth lighting values are calculated for each pixel and then quantized to a small number of discrete shades to create the characteristic "flat look". The shadows and highlights appear as blocks of color rather than being smoothly mixed in a gradient.
- Cel-shading is widely used in video games and animations to create a cartoon-like effect and mimic the style of a comic book or cartoon and/or give the render a characteristic paper-like texture.

The Ink outlining process

- In order for toon-shaded graphics to look realistic we need "sharp edges that define the color difference spaces" this gives it the visual diversity that a real image has as opposed to just being a combination of multiple colors to describe an image.

- **Methods**:
  - The Wireframe method
    - The wireframe method involves representing objects or scenes as a collection of lines or curves that outline their basic shape and structure.
    - Steps:
      - The back faces are drawn with thick lines.
      - The object faces are drawn using a single color.
      - Shading and Multi-texturing are applied.
  - The Edge Detection method
    - Edge detection is a process used to identify the boundaries or edges of objects within an image.
    - Edge detection algorithms analyze the changes in intensity or color gradients within an image to locate regions where abrupt changes occur. These abrupt changes often correspond to edges or boundaries between different objects or regions in the image.
    - Steps:
      - First, the scene is rendered with cel-shading to a screen-sized color texture.
      - Then depth-information of the scene is rendered to a screen-sized texture.
      - World-space surface normals are rendered as a screen-sized texture.
      - Finally, a Sobel filter or similar edge- detection filter is applied to the normal and depth textures to generate an edge texture. Texels on detected edges are black, while all other texels are white.

Chapter 5: Perspectives behind 3D games

Perspective correctness

- Perspective correct texturing accounts for the vertices' positions in 3D space, rather than simply interpolating coordinates in 2D screen space
- This achieves the correct visual effect but it is more expensive to calculate.

Ray tracing

- Image order rendering since each pixel is considered per pixel basis, where all the influencing objects could be found.

Orthographic projection

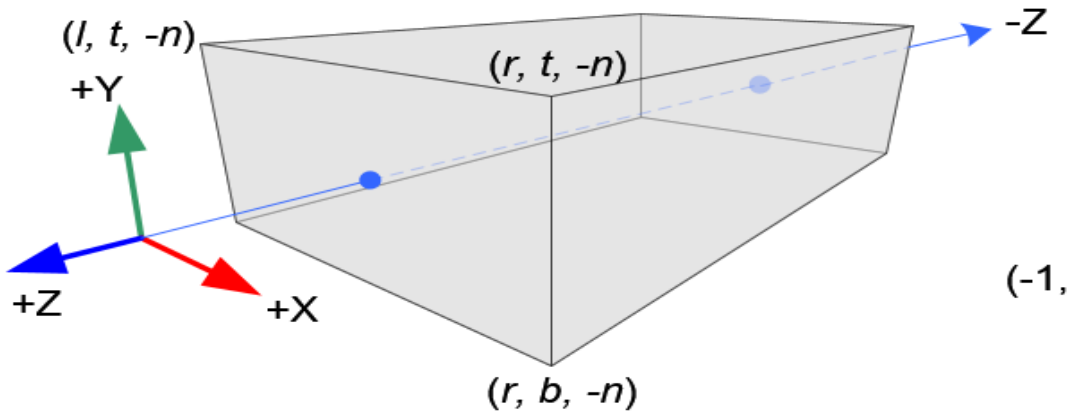
- It is a parallel projection (the lines of projection are parallel both in reality and in the projection plane).
- It is the projection type of choice for working drawings.
- To project the 3D point a_x, a_y, a_z onto the 2D point b_x, b_y using an orthographic projection parallel to the y axis (where positive y represents forward direction - profile view), the following equations can be used:

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & 0 & s_z \end{pmatrix} \cdot \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} + \begin{pmatrix} c_x \\ c_z \end{pmatrix}$$

Where, s = arbitrary scale factor and c = arbitrary offset.

Orthographic projection of canonical viewing volume

- In Vulkan's Canonical Viewing Volume only what's inside this region is viewed (displayed) X from (-1,1), Y from (-1,1), and Z from (0,1).
- Orthographic projection is the generalization of the view volume.
- It allows to specify whatever dimensions and location is required but maintains the overall shape of the view volume and keeps the view direction in orientation.
- The orthographic view volume is defined by the six bounding planes. The left and right along the x axis. The top and bottom along the y axis. The near and far plane along the z axis.



- To construct an orthographic projection a problem of transforming orthographic view volume to canonical view volume has to be solved. Two basic transformations can be combined:
  - Translate the center of the near plane c to the origin.

$$c = (\tfrac{r+1}{2}, \tfrac{b+t}{2}, n)$$

$$T = \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Scale it to fit the dimensions of View Volume

$$S = [r - l, b - t, f - n]$$

Perspective projection

- A linear projection where three dimensional objects are projected on a picture plane. This has the effect that distant objects appear smaller than nearer objects.

- Lines which are parallel in nature (that is, meet at the point at infinity) appear to intersect at a point called the `Vanishing point`.
- More realistic

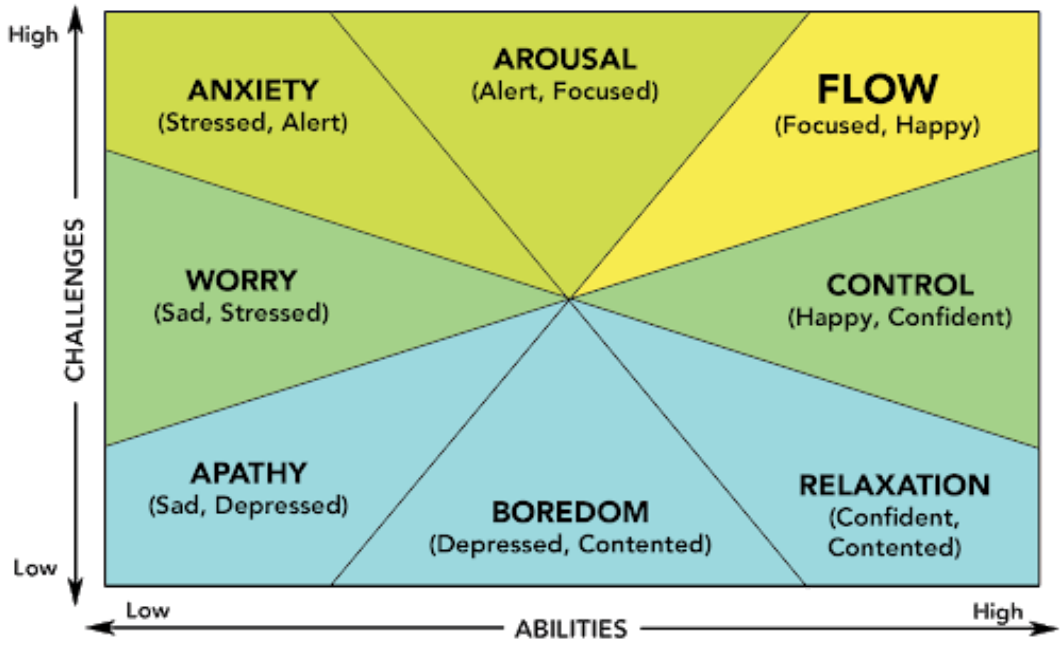
Classification of some 3D projections

- Orthographic
  - Primary
    - First angle
    - Third angle
    - Plan Elevation
  - Auxiliary
    - Isometric
    - Dimetric
    - Trimetric
- Oblique
  - Cabinet
  - Cavalier
  - Military
- Perspective
  - 1 point
  - 2 point
  - 3 point
  - Curvilinear

Flow

- An optimal psychological state proposed by Csikszentmihalyi.
- A state of total focus on the task at hand.
- Associated with positive experiences and performance outcomes.

The Psychology of Flow



The Science behind Flow

- In flow All Info Stream is Engaged, and nothing is left to monitor Physical Feeling (hunger, pain), problems at home and other important things.
- Body & Identity Disappear from Consciousness.

Components of Flow

1. **Challenge-Skill Balance**: A balance between the demands of the situation and personal skills.
2. **Action-Awareness Merging**: Deep involvement that makes actions seem automatic.
3. **Clear Goals**: Certainty about what one is going to do.
4. **Unambiguous Feedback**: Immediate and clear feedback that reaffirms actions.
5. **Concentration on Task at Hand**: Feeling focused.
6. **Sense of Control**: Happens without conscious effort.
7. **Loss of Self-Consciousness**: Concern for self disappears as person becomes one with activity.
8. **Transformation of Time**: Time passes faster, slower, or there is unawareness of time.

9. **Autotelic Experience**: Feeling of doing something for its own sake, with no expectation of future reward.

A mindset that facilitates flow

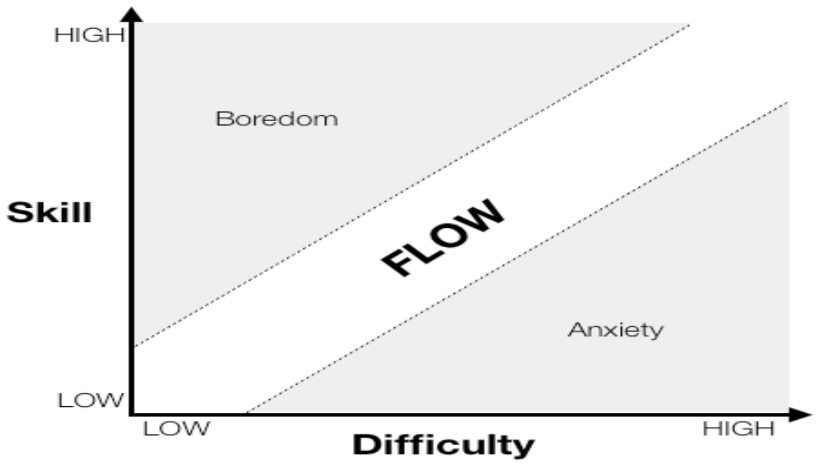
- Identify what is important – clear unambiguous goals.
- Move in the direction of what is important, taking into account feedback along the way.
- Embrace challenge
- Stay in the present moment

The Game method vs Education for flow

- `Education` : Only one trial, all preparation done once (low flow).
- `The Game Method` : optimizes flow through developing skill and increase challenge.

Computer game design & flow

- To trigger flow, a good computer game should be halfway between overwhelming and boring.
- Factors that affect flow of game design:
  - Levels
  - Points
  - Time limit
  - Prizes



Flow Catalysts

- Clarity
- Choice
- Challenge
- Commitment
- Certainty
- Centerness