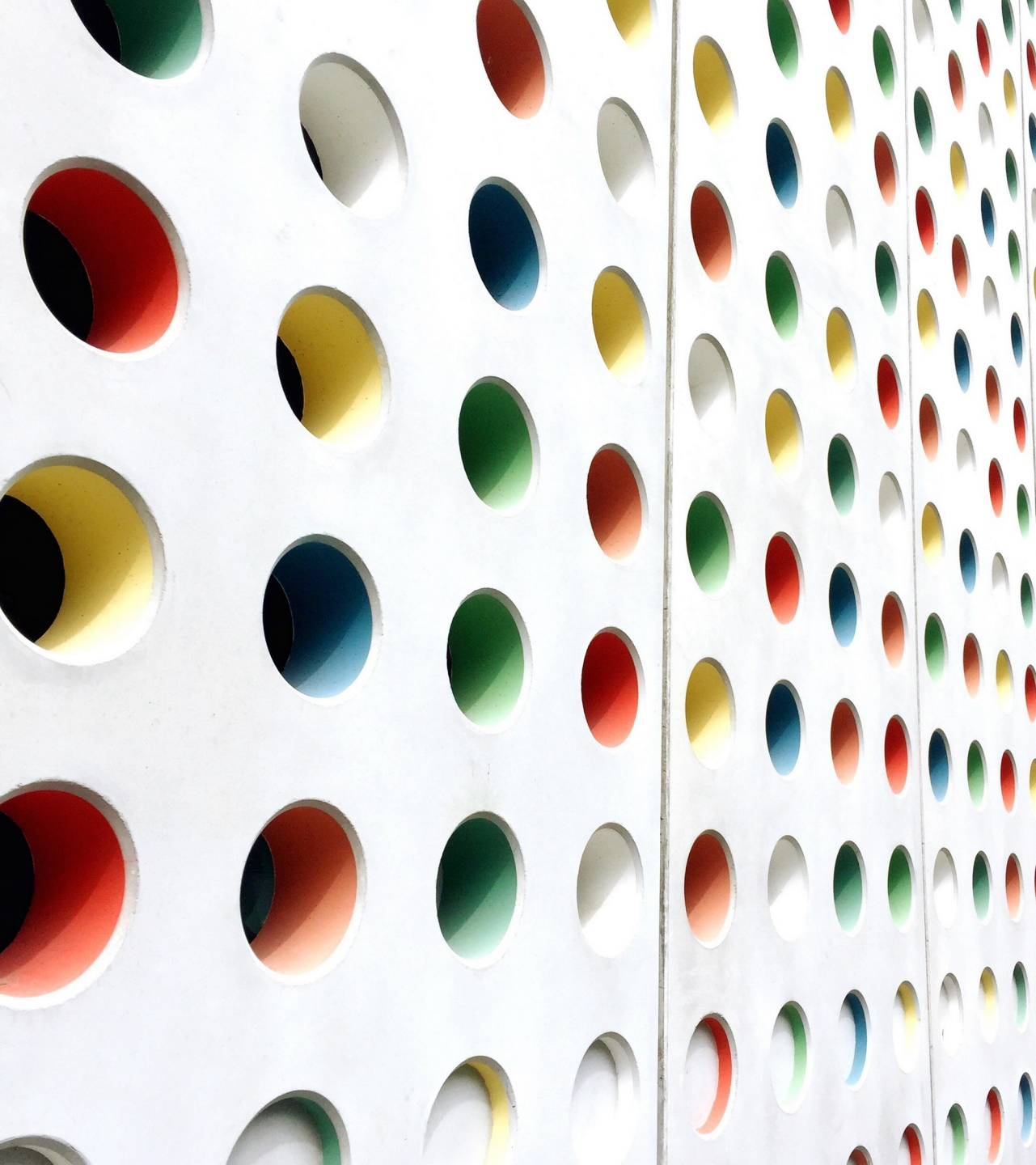


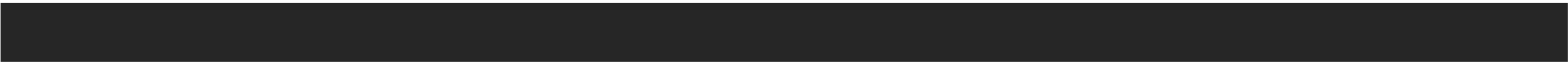


Game Design and Animation

SENG 4307 INTRODUCTION
TO COMPUTER GAMES AND
ANIMATION

Fundamental Elements of Game Design

CORE GAME DESIGN CONCEPTS:



Core Game Design Concepts:

Game Design is the craft of creating the structure and rules of a game to make it enjoyable and engaging. It's about shaping player experiences and interactions within a virtual world.

In Game Design, we focus on three key elements:

- 1. Player Abilities:** The skills and capabilities players bring to the game, influencing their actions and decisions.
- 2. Challenges:** Obstacles or tasks that players must overcome, adding excitement and satisfaction.
- 3. Environment:** The virtual space where the game unfolds, including settings and landscapes affecting player experiences.

Game Mechanics refer to the rules and systems that govern how the game works and how players interact with it. It's the machinery behind the scenes that makes the game function.

Game Mechanics consist of two main components:

- 1. Rules:** These are the established guidelines that define the structure of the game. Rules dictate what players can and cannot do, providing a framework for a fair and consistent gaming experience.
- 2. Player Interaction:** This involves how players engage with the game. It includes the actions, decisions, and reactions of players within the defined rule set. Player interaction shapes the dynamic and immersive nature of the gaming experience.

Gameplay

Gameplay is the overall experience of playing a game, encompassing the interactions, challenges, and activities within the game world. It's the dynamic and interactive aspect that keeps players engaged.

Three Ways to Enhance Gameplay:

- 1) Immersion:** This is the level of engagement and absorption a player feels within the game world. Immersive gameplay creates a sense of being part of the virtual environment, enhancing the overall experience.
- 2) Emotion:** Gameplay that evokes emotions contributes to a memorable gaming experience. Whether it's excitement, fear, joy, or sadness, emotional engagement adds depth and resonance to the player's journey.
- 3) Player Agency:** Player agency refers to the degree of control and influence a player has over the game. Games with strong player agency allow individuals to make meaningful choices, affecting the narrative and outcomes.

Game Types

- **Various types of games cater to different preferences and playstyles. Some of the most famous types include:**
 - **Action Games:** Focused on physical challenges, reflexes, and hand-eye coordination.
 - **Role-Playing Games (RPGs):** Emphasize character development, storytelling, and player choices.
 - **Strategy Games:** Require strategic thinking and planning to achieve objectives.

NPCs and Open World Games:

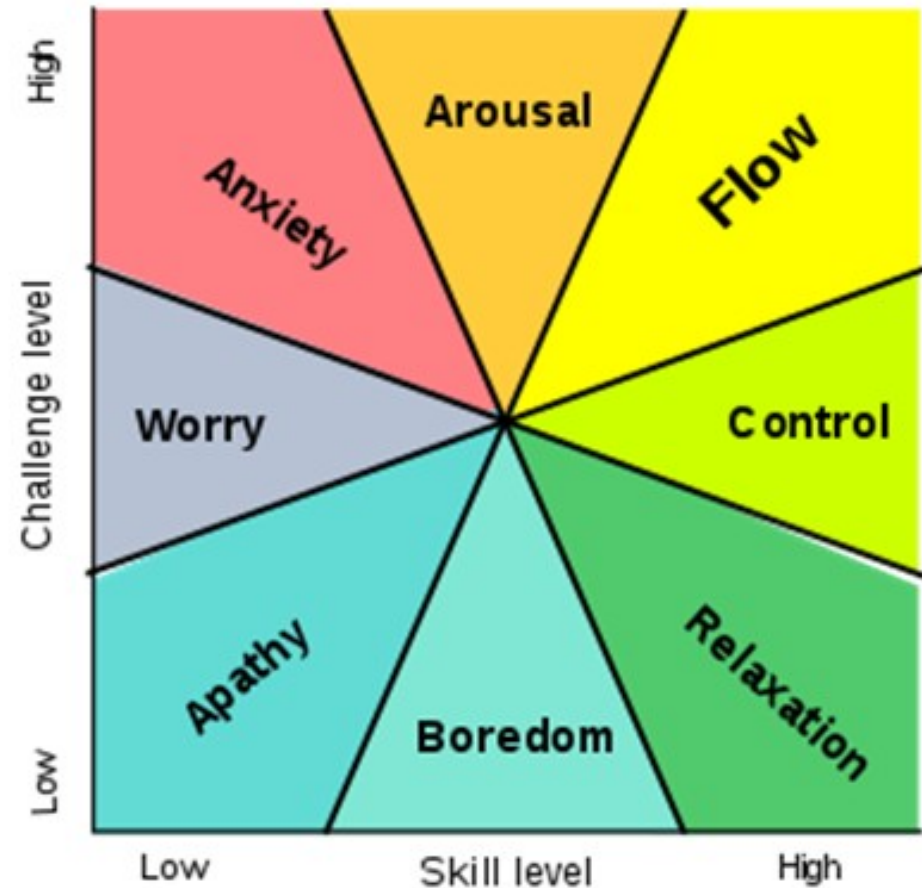
- 1. NPCs (Non-Player Characters):** These are characters controlled by the game's artificial intelligence rather than by a player. NPCs contribute to the game's story, challenges, or provide assistance.
- 2. Open World Games:** In these games, the virtual environment is expansive and allows players to explore freely. Open world games offer a sense of freedom and often feature a variety of activities beyond the main storyline.

History of Games

1. **Console Output:** In the early days, games were text-based and displayed through console output with gameplay primarily driven by text commands and simple visuals.
2. **GUI:** The introduction of Graphical User Interfaces (GUI) marked a significant shift. GameBoy, a handheld console, played a pivotal role. Games became more visually engaging with the use of graphics and icons, enhancing the overall gaming experience.
3. **Console:** The emergence of dedicated gaming consoles like Nintendo brought gaming to living rooms. Enhanced graphics, more complex gameplay, and iconic franchises defined this era.
4. **3D Games:** The transition to 3D gaming provided a more immersive and realistic gaming experience.
5. **VR and New Advances:** The present era witnesses the rise of Virtual Reality (VR). Leveraging VR headsets and advanced graphics for an unparalleled level of realism.

Skill level in games

1. **Mental state** in terms of challenge level and skill level, according to Csikszentmihalyi's **flow** model



Hardware for Games

Internal hardware: essential for running computer-based games. These components include:

GPU (Graphics Processing Unit): Responsible for rendering graphics and images.

CPU (Central Processing Unit): The brain of the computer, handling overall processing tasks.

RAM (Random Access Memory): Provides temporary storage for data actively used by the system.

SSD/HDD (Solid State Drive/Hard Disk Drive): Store game data, files, and the operating system.

PPU (Physics Processing Unit): Specialized for handling physics calculations in some systems. *PPU is not as universal, it enhances physics-related calculations in some gaming systems.*

External Hardware: These devices are dedicated gaming platforms with their own integrated hardware, designed specifically for gaming experiences. These systems include

PlayStation

Xbox

Nintendo

Software for Games

Game APIs (Application Programming Interfaces): are sets of tools and protocols that allow software applications to communicate with each other.

- **DirectX:** is a 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: are software frameworks that streamline game development. Notable examples include:

- **Unity**
- **Unreal Engine**
- These engines simplify the creation of games by providing pre-built components, rendering systems, and scripting languages.
- Besides game engines, developers can use programming languages like C++, Java, or Python to create games. These languages offer flexibility and control, allowing developers to build games from the ground up. But with the hinderance of having to develop every logic from scratch

Essential Math for Game Design

- **Pythagorean Theorem:** In a right-angled triangle, the square of the length of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the lengths of the other two sides. This can be written as:

where c is the length of the hypotenuse, and a and b are the lengths of the other two sides. For example, if a character is at point (3, 2) and its target is at point (6, 6), the distance d between them can be calculated as:

- **Angles and Trigonometry:** The sine, cosine, and tangent of an angle θ in a right-angled triangle are defined as follows:

For example, if the sun is at an angle of 45° from the horizontal, the length L of the shadow of a character of height h can be calculated as:

We also use Vectors for distance calculations and rate of motion management. By multiplying unit vectors of positions by different scalars (for example 3 for walking, 5 for running), we can control the speed of a character along the path.

- **Vectors:** A vector in 2D space can be represented as $v = (x, y)$, where x and y are the horizontal and vertical components respectively. The magnitude (or length) of this vector can be calculated using the Pythagorean theorem:

For example, if a character moves from point A(1, 1) to point B(4, 5), the movement vector v is (3, 4) and the distance travelled is:

Vector Normalization: The normalization of a vector $v = (x, y)$ results in a unit vector $u = (x/||v||, y/||v||)$. For example, the unit vector of v (3, 4) is:

Assuming this vector is for a game character that can either walk or run along a path. This unit vector u represents the direction of the path. Now, if the character is walking, we might say it moves at a speed of 3 units per move. So, the walking vector w would be $3 * u$:

If the character is running, it might move at a speed of 5 units per move. So, the running vector r would be $5 * u$:

Consideration for Game Rule

1. **Iteration and Prototyping:** In the game rule creation process, iteration and prototyping are essential for refining mechanics and identifying improvements.
2. **Game Design Core Ideals:** Aligning rules with core game design ideals, including *player ability*, *environment*, and *challenges*, ensures a harmonious gaming experience.
3. **Test (Build-PlayFeel-Modify) Repeat:** The iterative loop of building, playing, modifying, and repeating allows for continuous refinement based on player feedback.
4. **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*.

Gameplay Optimization

1. **Ludonarrative Dissonance:** describes the conflict between a game's narrative told through the story and the narrative told through the gameplay.
2. **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. This is particularly evident in Open-World Games, where players are given the freedom to explore and impact the game world, promoting a sense of agency.
3. **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.
4. **Challenge Design:** involves creating tasks or obstacles for the player to overcome. It's important to balance the difficulty of these challenges to ensure player engagement without causing frustration. This includes maintaining a **balance of power** between the **player's abilities** and the **challenges they face** in the game.
5. **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.

Consideration for Game Rule

1. **Shapley Value and Harsanyi Dividend (Subset Explanation)**
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.
2. **Nash Equilibrium and Coordination**
Nash Equilibrium: A state where no player has an incentive to deviate from their chosen strategy, ensuring stability. When we are only considering 2 players it is called *coordination*. (if 3 or more only *collaboration*)

Ex: Prisoner's Dilemma & Car-Crash (Symmetric Vs Asymmetric Game)

P/Dilemma	Stay Silent (Cooperate)	Confess (Betray)
Stay Silent	3/3	5/0
Confess	0/5	4/4

Car Crash	Go (Drive through)	Stop
Go (Drive through)	-5/-5	1/-1
Stop	-1/1	-1/-1

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...

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.

	Single Player (setting)	Multiplayer (setting)
Single Player	Compete(time): COD SP (Call of Duty SP)	Compete(Players): Call of Duty MP - Non-Team
Multiplayer	Racing (Blur): (time) Not Necessarily Against Player	COD MP-Team (Call of Duty MP - Team)

This table assists in designing games that align with player preferences, offering varied experiences based on the number of players and the gaming setting.

Fundamentals of Game Development

- **ESSENTIAL ASPECTS OF GAME DESIGN, GRAPHICS, AND PROGRAMMING FOUNDATIONS.**

Game Design Principles

Game Conceptualization

- **Defining a Game Concept**
 - **Identify Target Audience and Genre:** Before diving into the design process, it's crucial to define the target audience and the genre of the game. This understanding forms the foundation for creating an engaging and suitable gaming experience.
 - **Brainstorming Game Ideas:** Generate creative concepts to ensure a diverse range of ideas for gameplay. This brainstorming phase sets the stage for innovative and captivating game development.
- **Concept Refinement**
 - **Create a Game Design Document (GDD):** Develop a comprehensive Game Design Document (GDD) that outlines the game's vision, objectives, and mechanics. The GDD serves as a guiding document throughout the development process.
 - **Iterative Conceptualization and Feedback:** Embrace an iterative approach to conceptualization, incorporating feedback from

GDD General Components

1. **Introduction:** This is where you provide a high-level overview of your game. It should include the game's title, genre, and a brief description of the gameplay.
2. **Goals:** This section outlines the objectives of the game from both the player's perspective (what they're trying to achieve) and the developer's perspective (what you hope to achieve with the game).
3. **Game World:** This section describes the setting and environment of the game. It can include details about the game's universe, the physical and social rules that govern it, and the overall aesthetics.
4. **Mechanics:** This is where you detail the rules and systems that govern how the game works. It includes everything from character movement and combat mechanics to how the game's economy functions.
5. **Game Elements:** This section covers the various elements that make up your game, such as characters, items, enemies, and levels. It should detail their properties, behaviors, and how they interact with each other.
6. **User Interface:** This section outlines how the player will interact with the game. It includes details about the game's controls, menus, HUD, and any other elements the player will use to interact with the game.
7. **Audio:** Here, you'll detail the audio elements of the game, including background music, sound effects, and any voice acting.
8. **Prototyping:** This section is where you outline your plans for creating a prototype of your game. It should include what aspects of the game you plan to include in the prototype, how you'll test it, and what feedback you're looking to gather.



Principles Active in Games:

in Games

Telling Story: The narrative drawing players into the

- **Understanding Player Agency:** The significance of player agency—the player's ability to influence the story is Crucial. Designing with player agency in mind adds depth and engagement to the narrative.
- **Narrative Structures**
 - **Linear vs. Non-linear Narrative:** Linear has a Chronological path while *Non-Linear* is mostly implemented in *OpenWorld* games for *Emergent Gameplay*. Understand the benefits of each approach and how they impact player engagement and immersion.
 - **Interactive Storytelling Techniques.** Personalization techniques
 - Branching Narrative by Player Choice (FarCry3-Kill Her),
 - Environmental (Clue Battleground 2042 Guns),
 - Non-Linear (GTA- Franchise),
 - Character Dev (GTA-Girlfriend)



A storyboard: is a visual representation of a narrative, concept, or script, divided into sequential scenes, often referred to as panels. It's to pre-visualize a motion picture or sequence. key components of a storyboard:

1. **Title:** a title that briefly describes the scene or action taking place.
2. **Sketches:** These visually represent each frame or scene.
3. **Dialog/Action:** This is a description of the action, dialogue, or any relevant information about what's happening in the scene¹⁴.
4. **Specs:** details like shot size, lens length, two-shot and arrows used to show camera and/or character

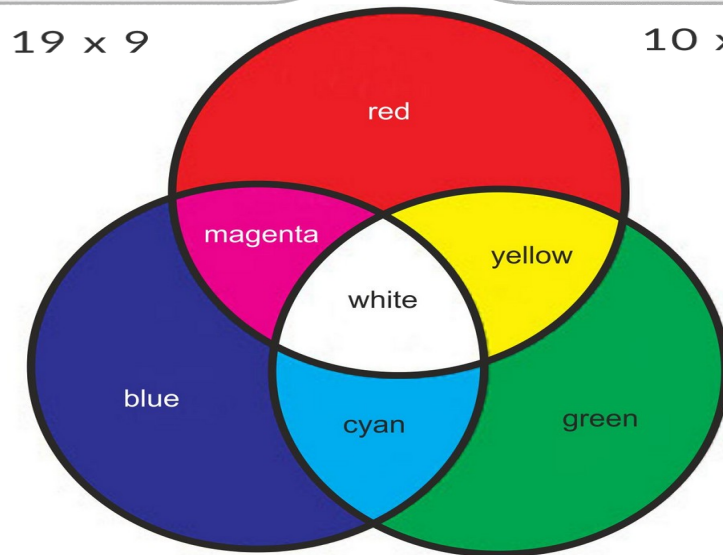
Introduction to Computer Graphics

Fundamentals of Computer Graphics:

- **Pixels, Resolution, and Color Models:**
 - **Pixels:** Pixels are the smallest units of a digital image. Understanding them is essential for creating detailed and vibrant graphics. Each pixel contributes to the overall composition of an image, and a higher density of pixels results in sharper visuals.
 - **Resolution:** Resolution defines the clarity and detail of an image. It is usually expressed as the total number or density of pixels in the image. with higher resolutions we get finer details.
 - **Color Models:** Some common color models include RGB (Red, Green, Blue) [0-255] and CMYK (Cyan, Magenta, Yellow, Black) [0-100]. RGB is used for screen displays, while CMYK is used for print.

Rendering Pipeline: The rendering pipeline is the step-by-step process of creating visuals in a game. From modeling to displaying images on the screen, each stage plays a vital role in bringing the game world to life. We will explore the rendering pipeline stages in the 3rd chapter, but generally it includes

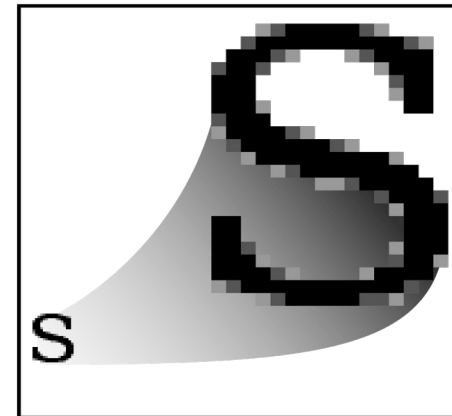
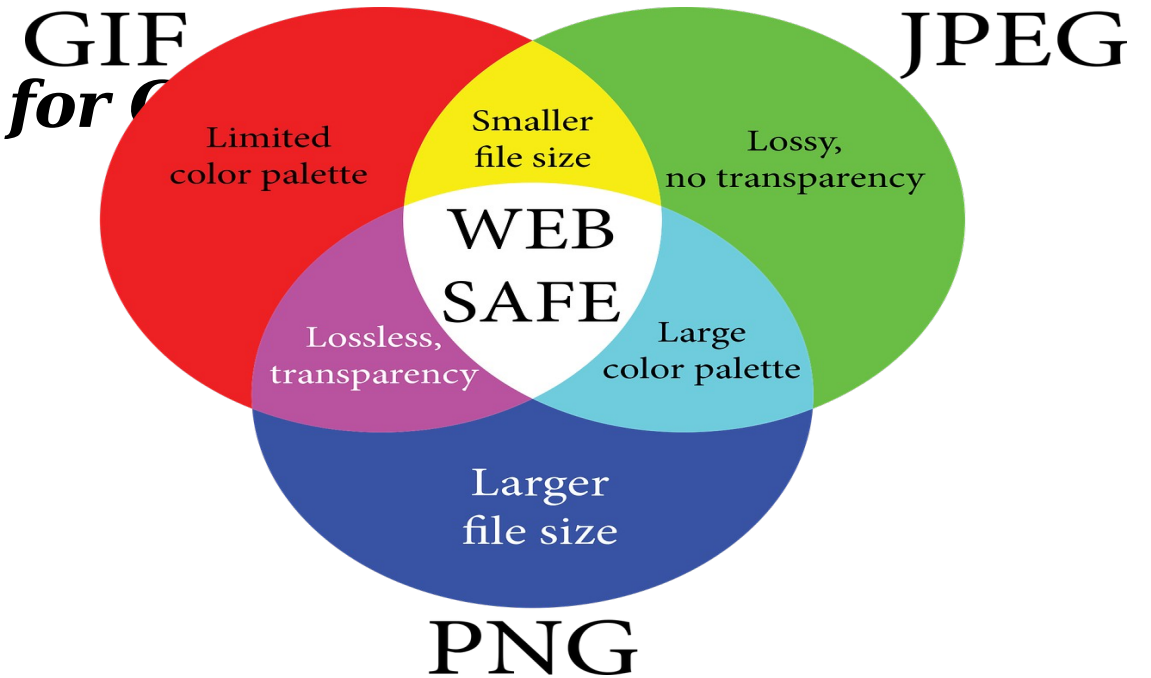
Same Screen Size, Different Resolutions.



Introduction to Computer Graphics for C

- **b. Graphic File Formats:**

- **Common Image Formats:** Familiarize yourself with widely used image formats like PNG (Portable Network Graphics) and JPEG (Joint Photographic Experts Group). Knowledge of the strengths and weaknesses of each format are necessary to make informed decisions about image storage and quality.
- **Scalable Vector graphics (SVG).** Unlike raster graphics, which use pixels, vector graphics use mathematical equations to represent images. Learn the advantages of vector graphics, such as scalability without loss of quality, making them ideal for logos and illustrations



Raster
.jpeg .gif .png



Vector
.svg

Principles of 2D Graphics:

2D Graphic Design Principles:

- **Composition and Layout:** Composition refers to the arrangement and organization of visual elements within a 2D space. It involves balancing and harmonizing elements to create a visually appealing and well-structured design.
- **Typography and Font Selection:** Typography is the art and technique of arranging type to make written language legible, readable, and visually appealing. Choosing fonts that align with the mood and purpose of the design is crucial for Aesthetics.

Flight to the end of the night
Travel & Transport
aurora borealis
Reptilian extinction
Kyrghizia
LANDSCAPE WITH ARGONAUTS
supercalifragelistic
Bismuth



Principles of Composition in 2D Graphics Design:

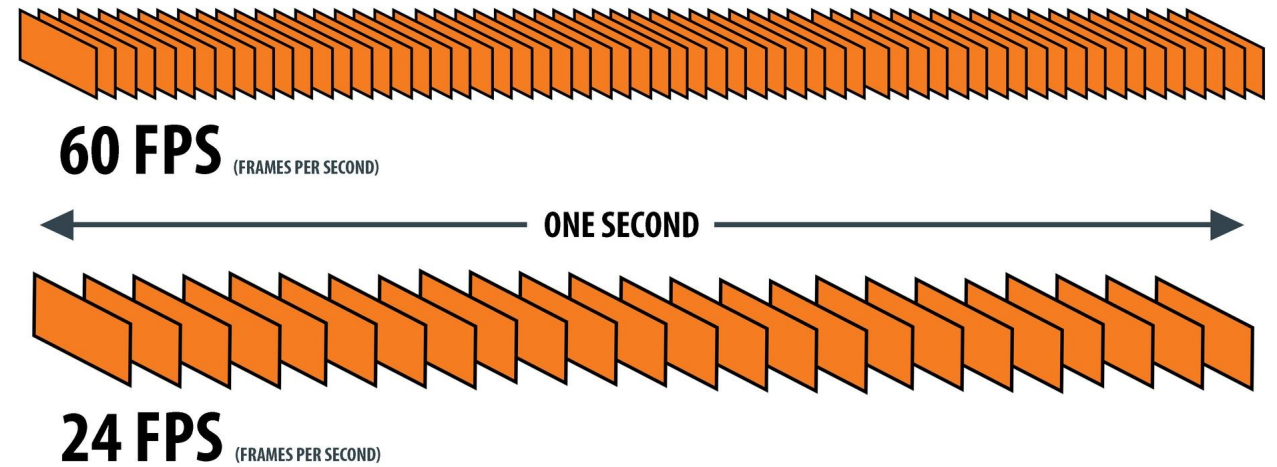
Composition refers to the arrangement and organization of visual elements within a 2D space. It involves balancing and harmonizing elements to create a visually appealing and well-structured design.

1. **Symmetry:** Symmetry is 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.
2. **Balance:** Balance is 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).
3. **Focal Points:** 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.

Principles of 2D Graphics:

Sprite Animation:

- **Frame-by-Frame Animation:** This technique involves creating the illusion of motion by displaying a sequence of individual frames. Each frame contains slight changes, and when played in rapid succession, they create smooth motion. By following up frame-by-frame pictures an animation is created to bring characters and objects to life in 2D games.
- **Principles of Motion in 2D:** Understand fundamental principles governing motion in 2D graphics. This includes concepts like **easing**, **anticipation**, and **follow-through**, which contribute to conveying realistic movement. Implementing these principles enhances the fluidity and believability of animations. These principles of Motion help to convey the illusion of physical laws such as inertia and momentum, and they add depth and complexity to the movement of characters and objects. By carefully considering and applying these principles, animators can create more engaging and realistic animations.



Easing: This principle is centered around the ways we can interact with time. For instance, if you have a heavy, plodding beat, movements that correspond with the downbeats of the song can make the contents of the video feel epic, confident, and powerful. Eases control exactly how quickly and smoothly something changes in a video. For example, when you throw a ball, it does not instantly stop at a place but gradually rests in motion.

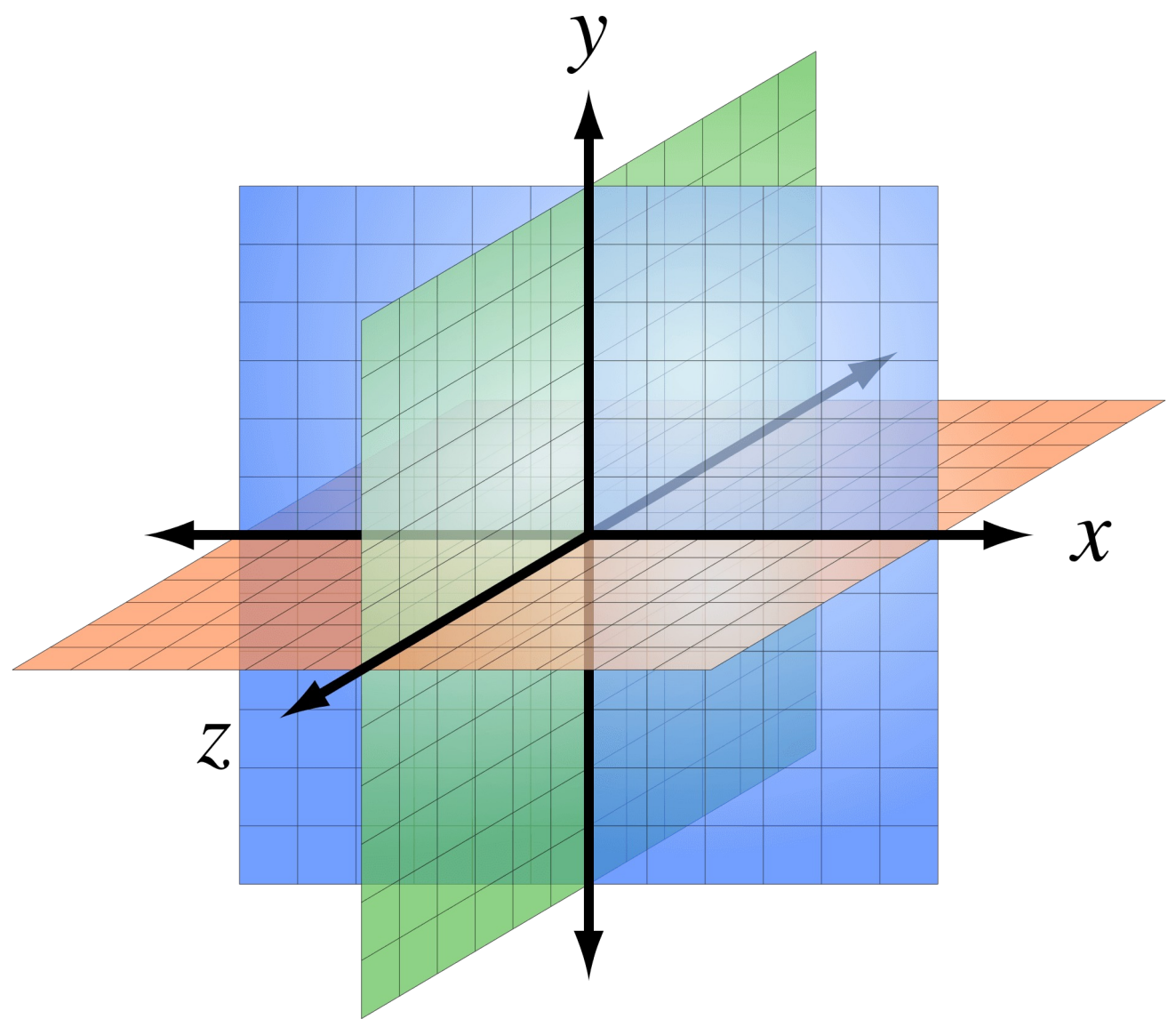
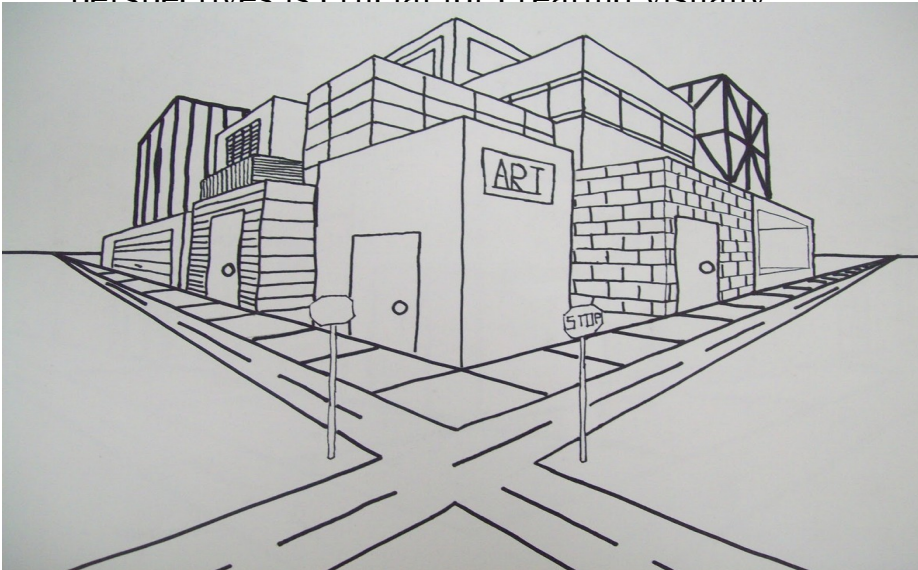
Anticipation: Anticipation is basically the action before the main action. examples include a character's heel pressing down before a step, or a character's mouth slightly tightening before opening to speak.

Follow-Through: Follow-through action is critical while producing any form of animation, whether 3D or 2D. This requires animators to animate the people and objects in such a way that their movement appears lifelike and smooth. For example, if a character with a scarf is running, then they stop suddenly, the scarf will fly forward past the body, then fall back where the character stopped. If a car stops moving, the antenna on top will keep moving for a second.

Introduction to 3D Modeling

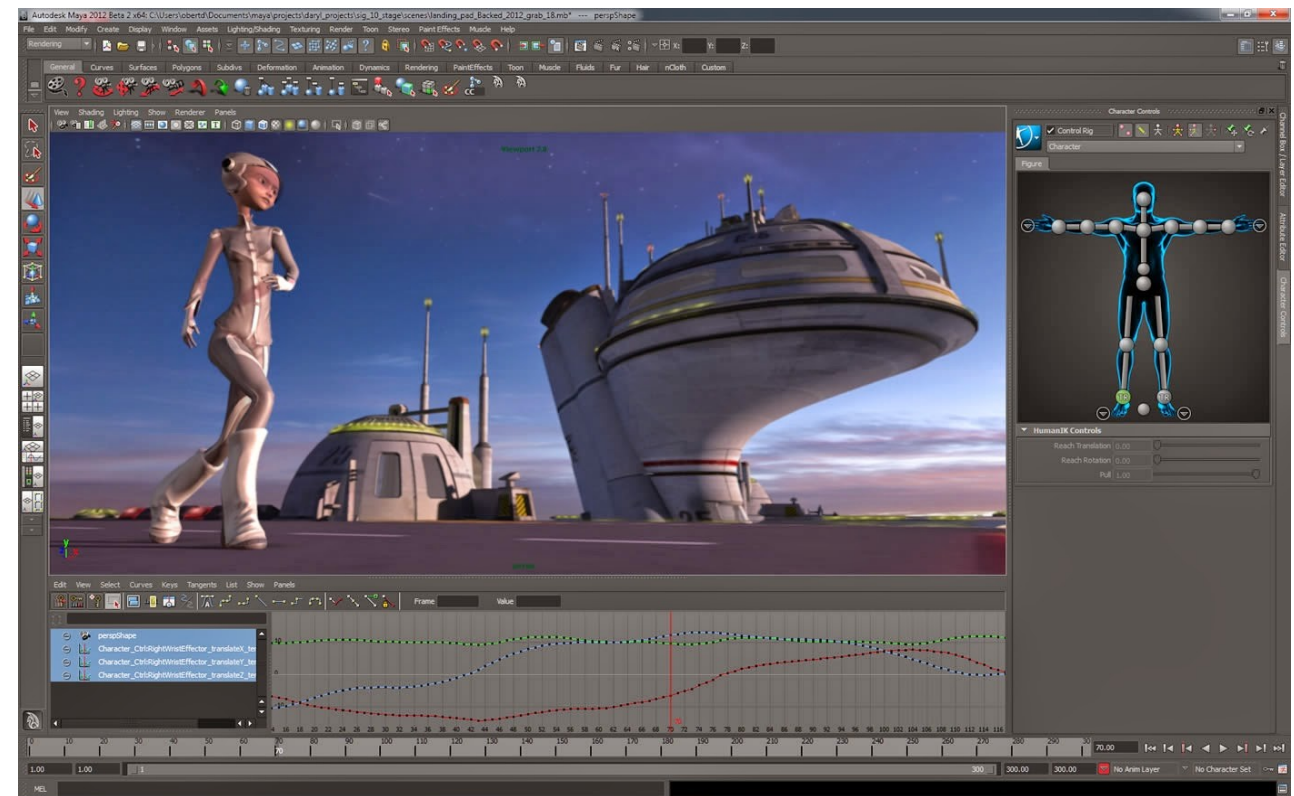
Basics of 3D Space:

- **Cartesian Coordinates in 3D:** the spatial positioning of objects in a three-dimensional (3D) space using Cartesian coordinates is given in the x , y , and z axes and they contribute to defining the location of objects in a 3D environment.
- **Camera Perspectives in 3D:** cameras capture and present the 3D environment with concepts such as field of view, perspective, and camera angles. Understanding camera perspectives is crucial for creating visually



3D Modeling Software:

- **Modeling Tools:** popular 3D modeling software like **Blender** and **Maya** Are essential modeling tools for creating detailed and realistic 3D models.
- Character in a world (in Maya)
- Character Design (in Blender)



- **Creating and Editing 3D Models:** these softwares allow to shaping 3D entities. The tools and techniques for creating, editing, and refining 3D models are built into these modeling software.

Programming Languages for Games

High-level vs. Low-level Languages:

- 1. High-level Languages:** These languages, like Python or Java, offer abstraction and ease of use. They simplify coding but may sacrifice some control over hardware.
- 2. Low-level Languages:** Examples include C++ or Assembly. These languages provide greater control over hardware but demand more intricate coding. Understand the trade-offs between ease of development and fine-grained control.

Choosing the Right Language: Tailor the choice of programming language to the specific needs of the game project. Consider factors such as project complexity, team expertise, and platform compatibility when making this crucial decision.

Scripting vs. Compiled Languages:

- 3. Role of Scripting Languages:** Scripting languages like Lua or Python offer flexibility for rapid development. They excel in tasks like game logic and AI scripting. Explore how scripting enhances the adaptability and agility of game development.
- 4. Advantages of Compiled Languages:** Compiled languages, such as C++ or C#, provide performance optimization. Compiled code runs faster, making it suitable for resource-intensive tasks like graphics rendering. Uncover the benefits of compiled languages in achieving optimal game performance.

Algorithms and Data Structures for Games

Essential Algorithms for Games:

- 1. Sorting Algorithms:** Algorithms like quicksort or mergesort organize game data efficiently, enhancing performance in tasks like **rendering** or **updating game states**.
- 2. Searching Algorithms:** Algorithms like binary search optimize **data retrieval**, crucial for efficiently **locating game assets** or information.
- 3. Pathfinding Algorithms:** Implement algorithms like **A*** for intelligent character movement in games, ensuring efficient and realistic paths.

Collision Detection Algorithms: Ensure accurate interaction between game elements using collision detection algorithms. Examples include bounding box or spatial partitioning algorithms, optimizing detection in complex game environments.

Data Structures for Games:

- 4. Arrays:** Utilize arrays for straightforward storage of game elements. Ideal for scenarios where fast random access is crucial, such as managing a collection of game entities.
- 5. Linked Lists:** Implement linked lists for dynamic data management, suitable for scenarios requiring frequent insertions or deletions of game objects.
- 6. Trees:** Employ trees, like quad trees or octrees, for efficient spatial organization in 2D or 3D environments, enhancing collision detection and rendering.

Efficient Data Storage: Optimize data storage for game objects by selecting suitable data structures. Efficient storage ensures quick

Object-Oriented Programming for Games

- **Classes and Objects in Game Development:** In a game, a "Player" class represents the blueprint for player entities, defining properties like health, position, and methods like move or attack. Each individual player in the game is an instance of this class, or an object.
- **Inheritance and Polymorphism:** Suppose there's a base class "Enemy" with common attributes and methods. Using inheritance, specific enemy types like "Zombie" or "Robot" can inherit from the base class, reusing its functionality. Polymorphism allows treating these diverse enemies uniformly, enhancing flexibility in game logic.

Design Patterns for Games

1. **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.
2. **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.
3. **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.

Enhancing Code Structure: Design patterns contribute to organized and modular code. By employing OOP principles and common design patterns, code becomes more maintainable, extensible, and easier to collaborate on within a development team.

Game Design Basics

Thank You!