UNIT — V

WEAPONS & ENEMIES Object Orientation: Classes, Instances and Inheritance — Animations, Frames, and Prefabs — Cameras — Starting the Enemy Drone Prefab — Enemies, Intelligence, and Philosophical Zombies.

1. Object Orientation: Classes, Instances, and Inheritance

1.1 What is Object-Oriented Programming (OOP)?

Object-Oriented Programming is a paradigm based on the concept of **objects**, which bundle:

- **Data** (fields/variables)
- **Behavior** (methods/functions)

OOP improves modularity, reusability, scalability, and maintainability in game development.

1.2 Classes

A **class** is a blueprint or template for creating objects.

Example (C# in Unity):

```
public class Weapon {
   public int damage;
   public float range;
   public void Fire() {
      // shooting logic
   }
}
```

A class does **not** occupy memory until instantiated.

Characteristics:

- Defines structure
- Includes fields, methods, constructors
- Represents abstract concepts (Enemy, Weapon, Player, Drone)

1.3 Instances (Objects)

An **instance** is a concrete realization of a class.

Example:

```
Weapon pistol = new Weapon();
pistol.damage = 25;
```

Every instance has:

- Its own memory allocation
- Its own property values

In Unity, **GameObjects with attached scripts** are instances.

1.4 Inheritance

Inheritance allows one class to **extend** another.

Benefits:

- Code reusability
- Hierarchical structure
- Polymorphism (methods behave differently for different classes)

Example:

```
public class Enemy {
   public int health;
   public virtual void Attack() {}
}
public class DroneEnemy : Enemy {
   public override void Attack() {
      // drone-specific attack
   }
}
```

Practical Game Example:

- **BaseEnemy**: movement, health
- **ZombieEnemy**: slow movement, melee attack
- **DroneEnemy**: aerial patrol, ranged attack

2. Animations, Frames, and Prefabs

2.1 Animations in Game Development

Animation is the process of changing object properties over time (position, rotation, sprite frames, blend shapes, etc.).

Types:

- **Sprite-based animation** (2D frame swapping)
- **Skeletal animation** (bones/joints)
- **Blend animations** (mixing states, e.g., walk → run)

Unity uses:

- **Animator** and **Animation Controller**
- **Keyframes** on a timeline
- **State Machines**

2.2 Frames

A **frame** is a single image in an animation sequence.

Frame Concepts:

- **Frame Rate (FPS)**: Frames per second
- Higher FPS → smoother animation
- Low FPS → choppy but stylized (retro games)

Keyframes:

Define:

- Start position
- End position
- Morphing between states

2.3 Prefabs

A **Prefab** is a reusable, preconfigured game object.

Why use Prefabs?

- Reusability (spawn multiple enemies)
- Consistency (same behavior and components)
- Save development time

- Easy to update globally

Examples of Prefabs:

- Weapons (guns, rockets)
- Enemy drones
- Bullets and projectiles
- Explosions and particle effects

Unity workflow:

1. Create GameObject
2. Add components (script, model, collider, etc.)
3. Drag to **Project** window → becomes a Prefab
4. Instantiate at runtime using scripts

3. Cameras

3.1 Camera Types

- **First-Person Camera**
- **Third-Person Camera**
- **Top-Down Camera**
- **Side-Scroller Camera**
- **Cinematic Cameras**

Cameras determine what the player sees.

3.2 Camera Controls

Common camera behaviors:

- Follow camera (tracks player)
- Orbit camera (rotates around target)
- Smooth damping camera (delayed movement to look natural)

Example follow script:

```
public Transform player;

public float smoothSpeed = 0.125f;

public Vector3 offset;
```

```
void LateUpdate() {
    Vector3 desired = player.position + offset;
    transform.position = Vector3.Lerp(transform.position, desired, smoothSpeed);
}
```

4. Starting the Enemy Drone Prefab

4.1 What is an Enemy Drone?

A typical **aerial enemy** with:

- Patrol movement
- Shooting abilities
- Detection sensors
- Health and destruction behavior
- AI state machines

4.2 Components Needed for a Drone Prefab

Component	Purpose
Model / Sprite	Visual representation
Rigidbody / Rigidbody2D	Physics simulation
Collider	Collision detection
EnemyDrone Script	Movement + AI behavior
Particle System	Explosion/VFX
Health Script	Takes damage

4.3 Drone Scripts Structure

Example structure:

Movement

- Hovering
- Patrolling between waypoints

Detection

- Raycasts or triggers
- Vision radius

- Aggro behavior

Attack

- Bullet instantiation
- Cooldown timers

Death

- Reduce health
- Play explosion
- Destroy object

5. Enemies, Intelligence, and Philosophical Zombies

5.1 What is Enemy AI?

Enemy AI is behavior programmed to challenge the player.

Principles:

- Deterministic logic
- State-based behavior (FSM)
- Pathfinding
- Decision making

Common AI states:

- Idle
- Patrol
- Chase
- Attack
- Flee
- Dead

5.2 Intelligence in Games

Game enemies are not truly "intelligent."
They follow:

- Scripts
- Rules
- Finite State Machines
- Behavior Trees
- Utility AI

They appear intelligent through:

- Animations
- Responsiveness
- Reaction to player actions

5.3 The Concept of Philosophical Zombies (p-zombies)

A *philosophical zombie* is a being that:

- Looks conscious
- Behaves like a conscious being
- But has **no inner experience**

In games:

- Enemies mirror consciousness-like behavior
- But do not “think”
- They simulate emotions or awareness

Application:

Enemies:

- React to stimuli
- Dodge, shoot, detect player
- Show “anger” animations
  But they do **not feel** anything.

This concept helps understand:

- Enemy design
- Behavior simulation
- Illusion of intelligence in games

6. Summary for Quick Revision

OOP

- Classes = blueprint
- Instances = objects
- Inheritance = reuse + hierarchy

Animations

- Actions broken into frames
- Keyframes define transitions
- Animator controls state logic

Prefabs

- Reusable game objects
- Multiple instantiations

Cameras

- Define game perspective
- Follow, orbit, or static behavior

Enemy Drone Prefab

- Patrol, detect, attack
- Scripts + physics + VFX

Enemies & Intelligence

- AI = rules + states
- No real intelligence
- P-Zombies reflect game AI nature