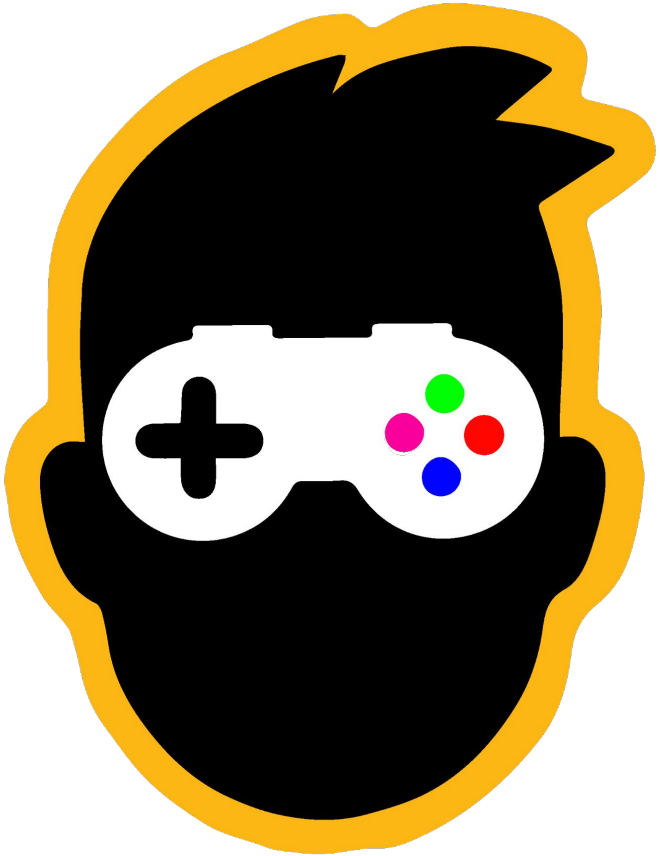
Game Dev Workshop.

By GameDevUtopia aka GDU

What is GDU?

What is GDU?

- A student community focused upon game development, both on the technical and artistic angle
- Primarily for the students and by the students
- The club started in PICT, now is aiming to branch to other colleges
- Currently branched in IIIT Kottayam, and targeting even more



What is A Video Game

What is a Video Game?

- Video games offer interactive digital entertainment, where players engage with virtual environments using input devices to make decisions and take actions.

- Games come in various genres, each with unique gameplay styles and objectives, providing a wide range of experiences from action-packed adventures to strategic simulations.

- Examples of best games are : Super Mario, Candy Crush, Pong

Types Of Games

Kinds of Games.....

- Yes there are various kinds of games ranging from shooters to racing ones to the ones with the complex stories.

- Like books, music and movies, video games also has a long list of genres.
- Game genres are based on the type of graphics and artstyle it posses or the mechanics it has.
- And like other entertainment media, video games also mixes two or more genres
- For example, GTA V is an open world game and also is story driven

What are Genres of Video Games

<< FPS : First Person Shooter

Eg : Crysis 3

Third Person >>

Eg : Resident Evil 2 Remake

<< Strategy

Eg : Factorio

Turn Based >>

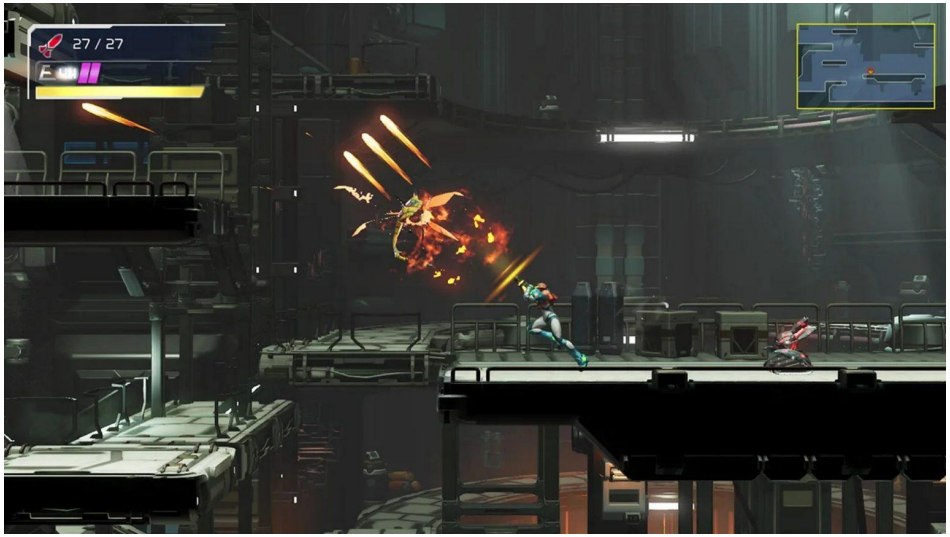
Eg : Mechanicus

<< Fighting

Eg : Tekken 8

Simulation >>

Eg : BeamNG.Drive



<< Platformer

Eg : Metroid :: Dread

RPG : Role Playing Game >>

Eg : Zelda : Tears Of Kingdome

<< Racing

Eg : Gran Turismo 4

Survival >>

Eg : Sons Of The Forest

Frameworks Vs Engines


GameMaker Studio
Gamebryo
DEFOLD
lumberyard
HERO ENGINE
COCOS2D-X
CRYENGINE
unity
XENKO™
GODOT Game engine
GameSalad
ShiVa
UNREAL ENGINE
jMonkeyEngine
LÖVE
libGDX
PLAYCANVAS
MONOGAME
TORQUE 3D
Corona Labs

Game Engines

- Game Engines are comprehensive suites of tools and technologies that provide a complete environment for game development.
- It typically includes features like **Rendering, Physics, Audio, Node Scripting, and Asset Management.**
- Examples of game engines include **Unity, Unreal Engine, and Godot.**

Game Frameworks

- Game Frameworks are collections of libraries and tools that simplify game development while remaining **lightweight, customizable, and flexible.**

- They provide you with a foundation you can work with while creating your games. You can also learn more about the underlying principles and techniques of game development.

- Examples: **Pygame, Phaser, and Love2D.**

Which One
To
Choose ?

Flexibility against Convenience

- A game framework gives you more control and flexibility over how you use these components, but it also requires more coding and integration work from you.
- A game engine provides a complete solution for creating games, including a graphical user interface, a scene editor, an asset manager, a scripting language, etc. It handles most of the technical details for you, and lets you focus on the design and content of your game.
- However, you also have to follow the conventions and limitations of the engine, and it may be harder to modify or extend its features.

Help—I've never created a game before!

With so many free, open source game engines and tutorials available online, there's never been an easier (or more exciting!) time to try out game development.

Are you…

- **Into JavaScript?** You might be interested in Phaser, or Sprig.
- **Comfortable with C++ or C#?** Look at Godot, Unity and Unreal Engine.
- **Raving about Rust?** You might like Bevy.
- **Proficient with Python?** Check out Pygame or Godot (Godot uses GDScript, which is similar to Python).
- **Dangerous with Java?** Take a look at libGDX.
- **In love with Lua?** Check out LÖVE or Defold. Like retro games too? Drop everything and check out LIKO-12!
- **Fond of Flutter?** Take a peek at Flame.
- **Gearing up with Go?** You might want to look at Ebitengine.
- Do you *really* like retro games? Maybe you can…
    - **Hack on a Roguelike dungeon crawler** in JavaScript or Haskell.
    - **Crank out a text adventure in or some interactive fiction** with Clojure or Ink (similar to Markdown).
    - **Craft that 8-bit console game you always fantasized** about with PICO-8, Pixel Vision 8, TIC-80 (Lua), BASIC8 (Basic), or Pyxel (Python).

We Choose
Love2d

What is Love2d

- Fast 2D Game Development Framework written in **C++**.
- Uses **Lua** as its Scripting Language.

- Contains Modules for graphics, keyboard input, math, audio, physics and much more.

- Completely **Free and Open Source**.
  Size approx 10-15 Mb.

- Portable to all Major Desktop and Android/IOS.
- Great for prototyping.

What is Lua ?

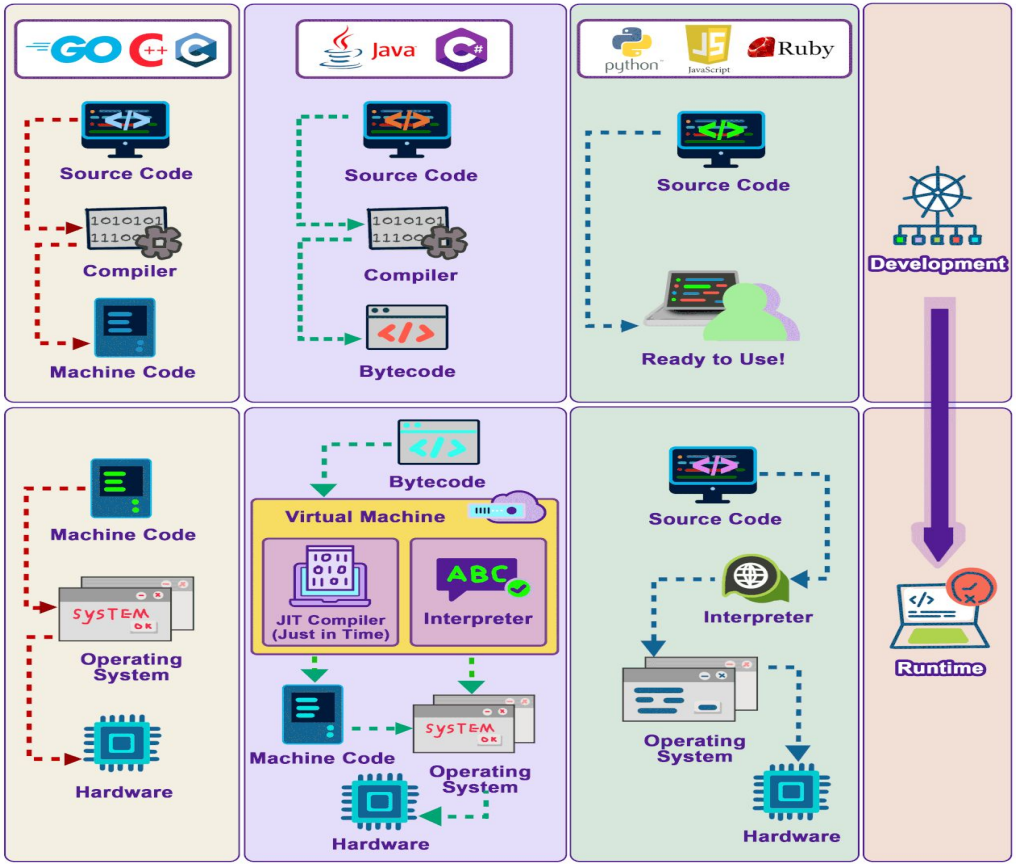
- Fast, Multi-paradigm scripting Language, designed By Computer Scientists in Brazil.

- Interpreted Language.
- Lua is cross-platform, since the interpreter is written in ANSI C (Highly Portable).

- Lua when used with JIT, widely considered as fastest scripting language in world. (Beats PyPy Node).
- Embedded Language.

How do C++, Java, Python Work?

blog.bytebytego.com



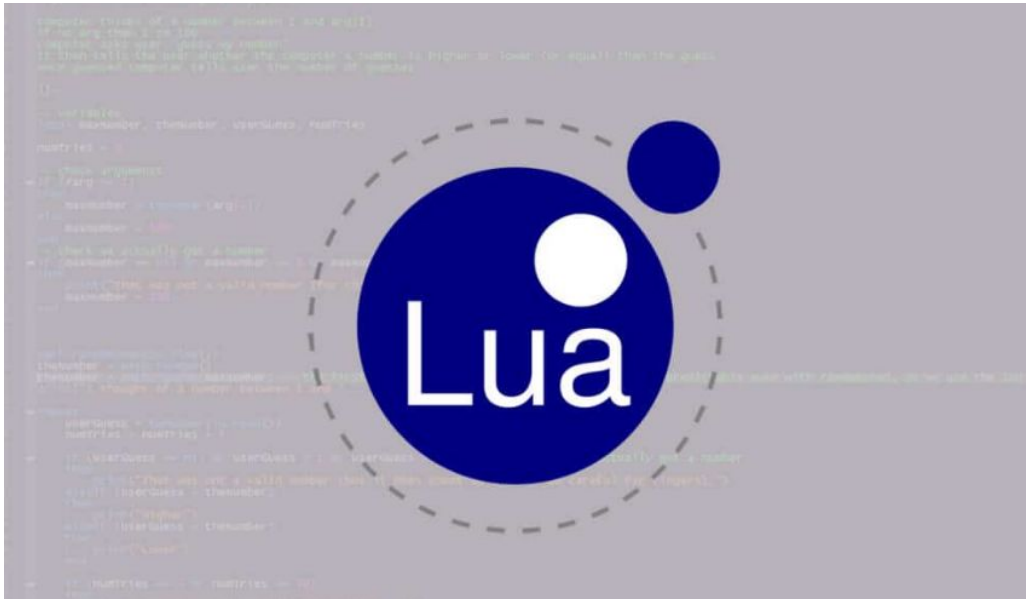
Compiled Vs Interpreted

- A **compiler and an interpreter** are both programs that **translate source code into machine code**. However, they do this in different ways.

- A **compiler** translates the **entire source code program into machine code before the program is executed**. This process is called **compilation.** Once the program has been compiled, it can be executed without the compiler.

- An interpreter translates **the source code line by line as the program is executed**. This process is called **interpretation**. The interpreter reads each line of source code, translates it into machine code, and then executes it.

Lua Scripting Quick Overview



https://github.com/pohka/Lua-Beginners-Guide

Part One
Done
:)

Love2d

For the workshop we will be using Love2d

Demo Of Game

3 Main Functions

- **love.load()**

This function is used for initializing our game state at the very beginning of program execution. Whatever code we put here will be executed once at the very beginning of the program.

- **love.update(dt)**

This function is called by LÖVE at each frame of program execution; dt (i.e., DeltaTime) will be the elapsed time in seconds since the last frame, and we can use this to scale any changes in our game for even behavior across frame rates.

- **love.draw()**

This function is also called at each frame by LÖVE. It is called after the update step completes so that we can draw things to the screen once they’ve changed.

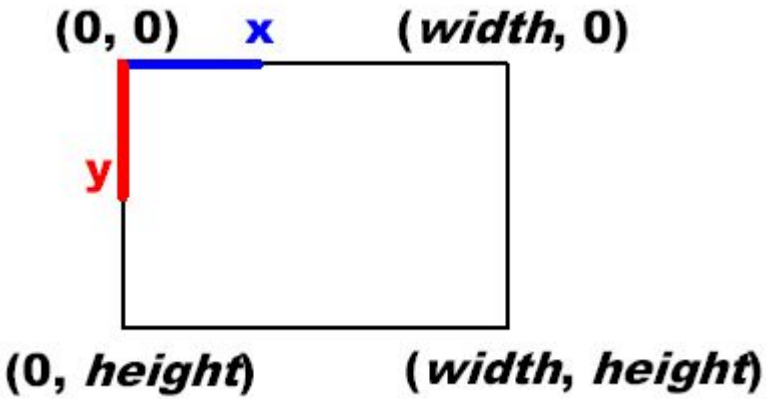
Co-ordinate System

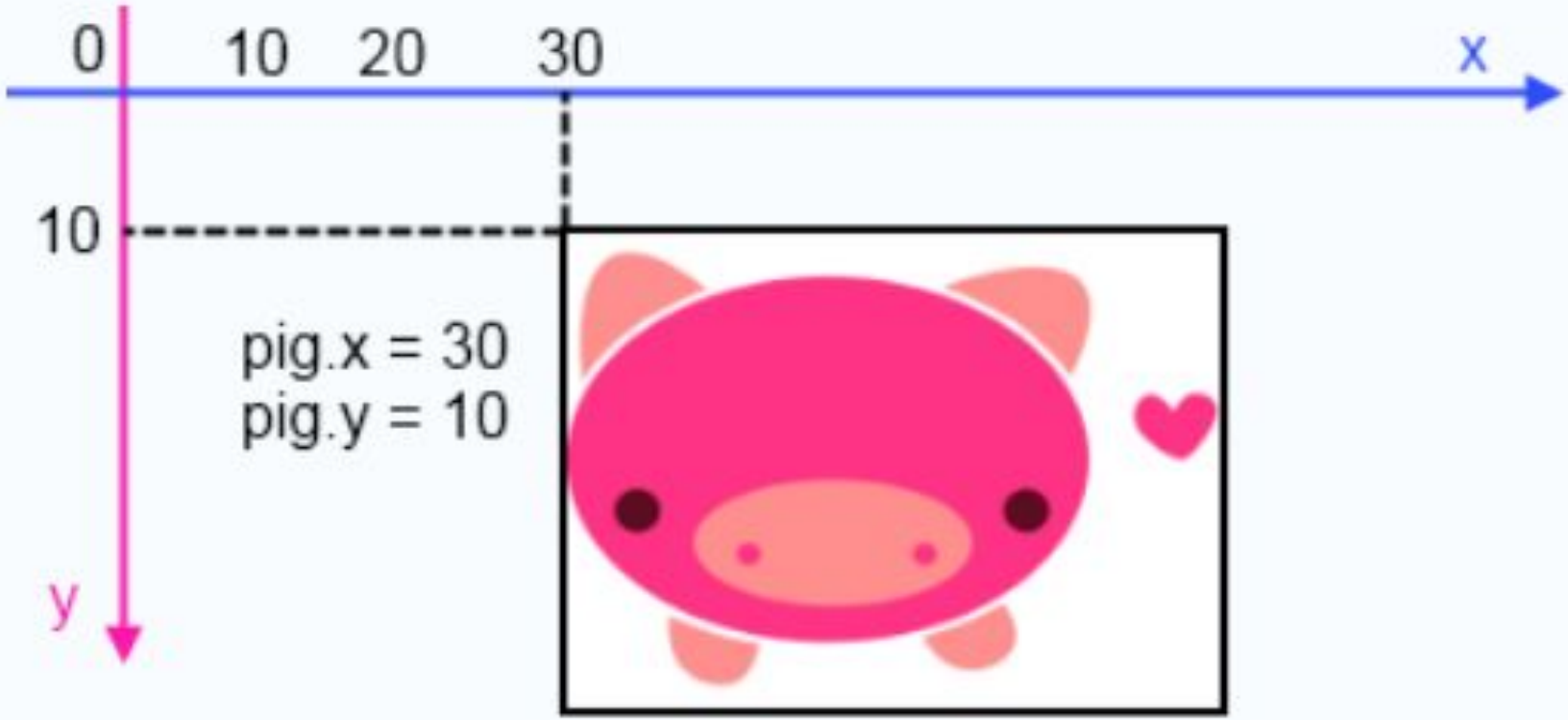
2D Coordinate System

Slightly different from the traditional coordinate system you might've used in math class, the 2D coordinate system we're referring to here is a system in which objects have an X and Y coordinate (X, Y).

Are drawn accordingly, with (0,0) being the top-left of the system.

This means **positive directions moving down and to the right, while negative directions move up and to the left.**




0
10
20
30
x
10
pig.x = 30
pig.y = 10
y

Delta Time

Delta time is the time that has passed between the previous and the current update.

So on computer A, which runs with 100 fps, delta time on average would be 1 / 100, which is 0.01.

Delta Time

Comparison

Without Delta Time:

let's say that Computer A runs with 100 fps (frames per second), and Computer B runs with 200 fps.

Computer A: **100 x 5 = 500**
Computer B: **200 x 5 = 1000**

With Delta Time:

So on computer A, 100 fps, delta time on average would be 1 / 100, which is 0.01.

On computer B, delta time would be 1 / 200, which is 0.005.

Computer A: **100 x 5 x 0.01 = 5**
Computer B: **200 x 5 x 0.005 = 5**


60FPS
600 pixels
30FPS
300 pixels
60FPS
100 p/s
600 pixels
30FPS
100 p/s
600 pixels

Delta Time Demonstration

Hands On Implementation.

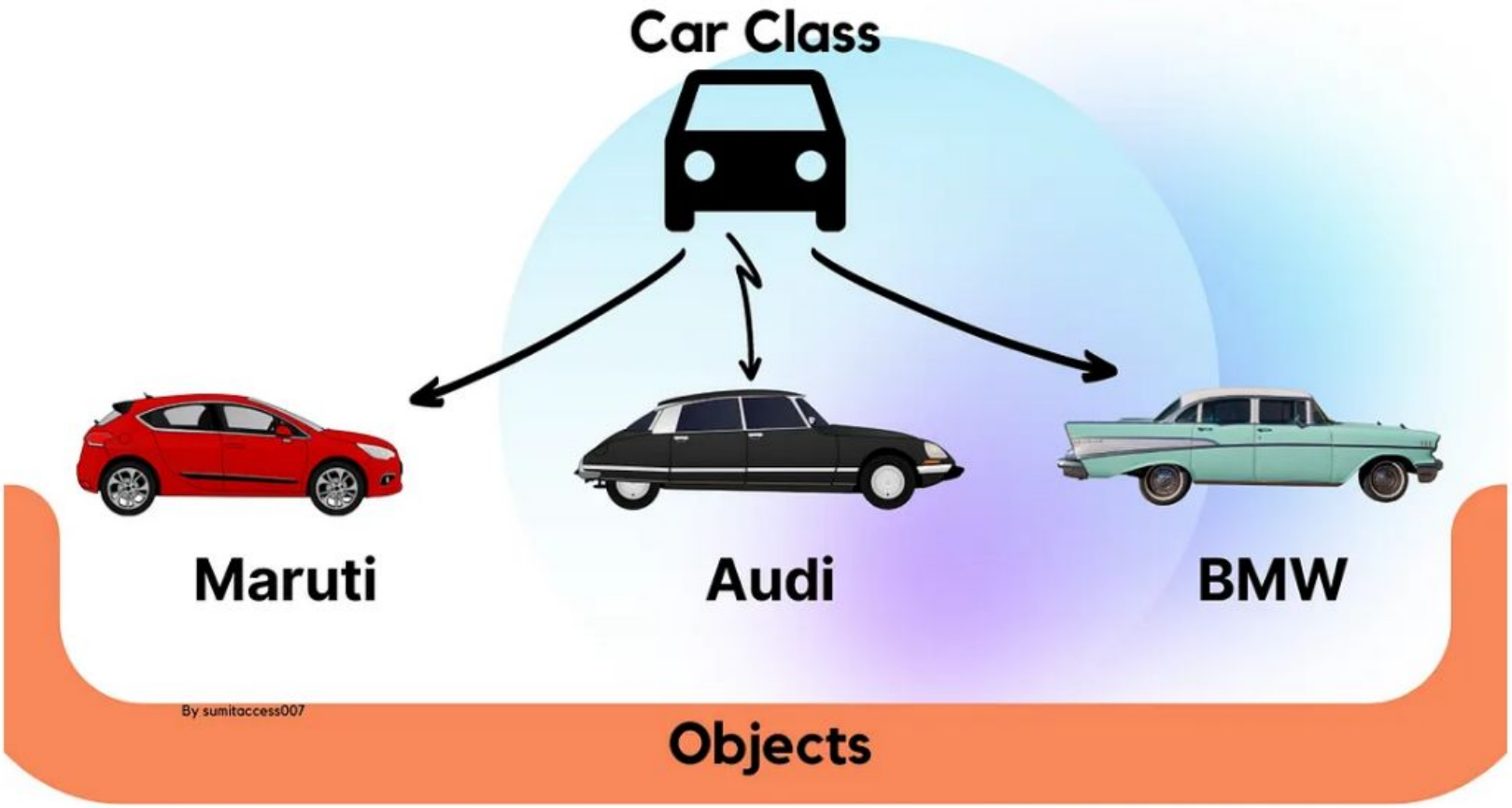
Player Script
Movement
Enemies

OOP Concepts

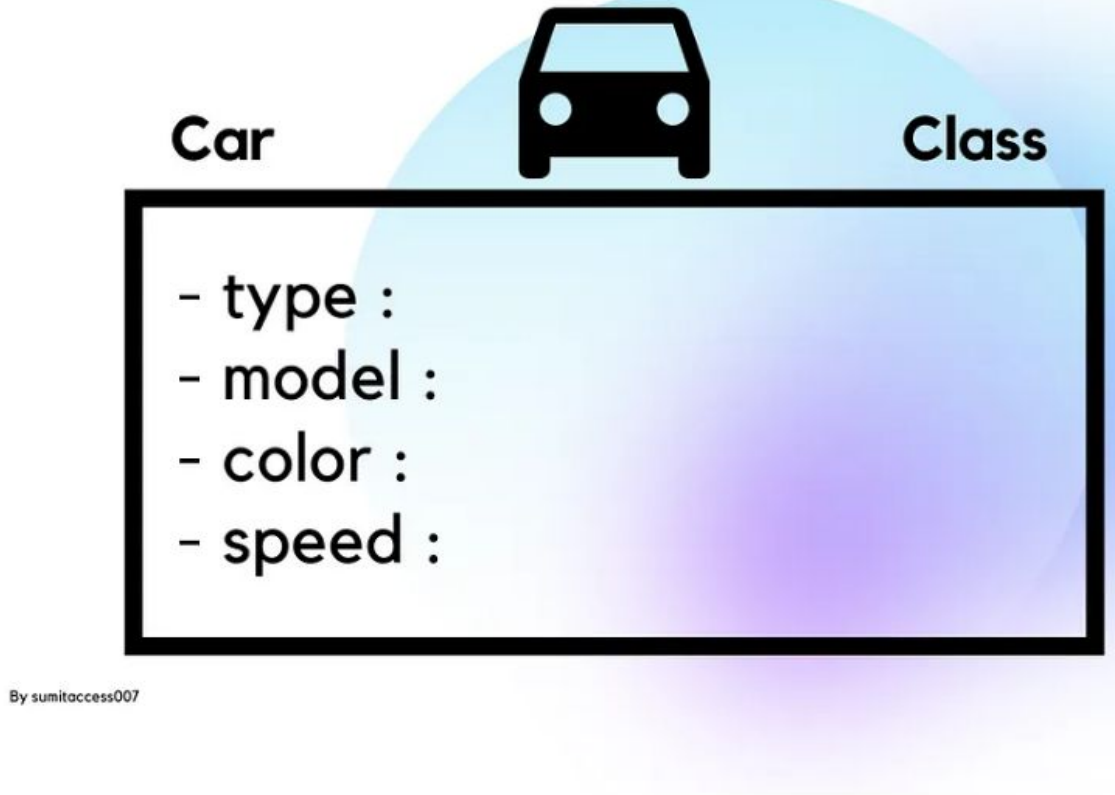
Classes And Objects

A class is a blueprint that defines the properties and behaviour of similar objects or entities.

Each object is an instance of a class that has a set of properties and behaviors which are defined in the class and are associated with the objects.



Car as a Class and Its Objects



Car Class — Looks Like a Template or Blueprint

Maruti

- type : Maruti
- model : Focus
- color : Red
- speed : 60

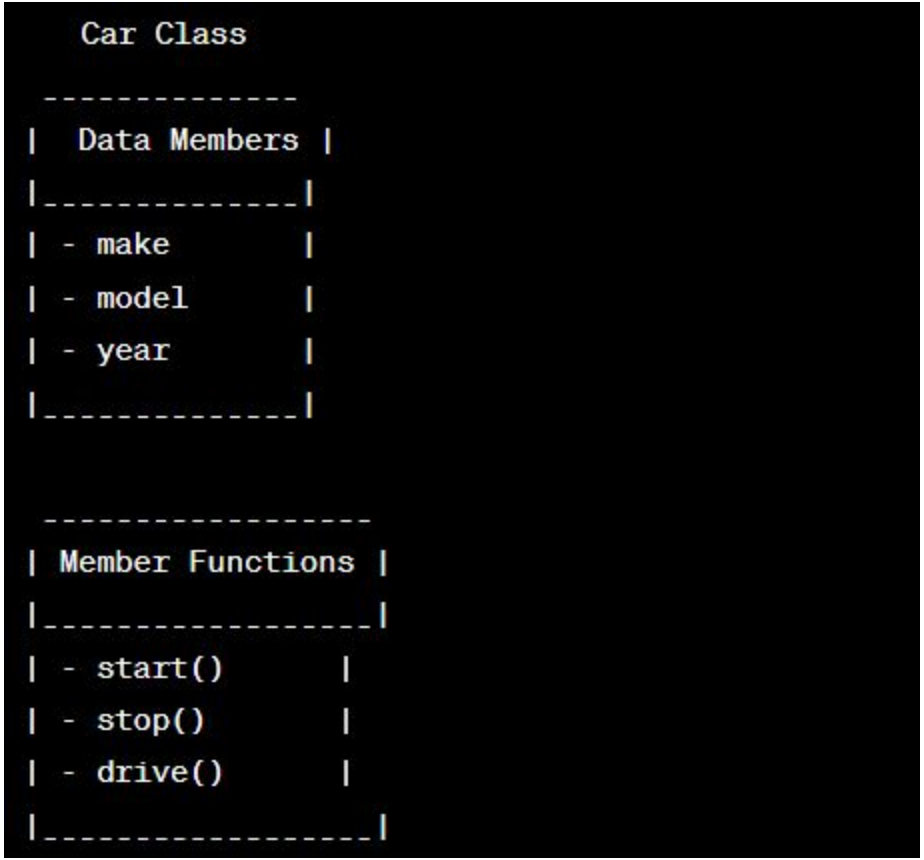
Audi

- type : Audi
- model : Golf
- color : Blue
- speed : 80

BMW

- type : BMW
- model : Auris
- color : Green
- speed : 90

By sumitaccess007


Car Class
Data Members
- make
- model
- year
Member Functions
- start()
- stop()
- drive()

```java
class Car {
    // Data members (attributes)
    String make;
    String model;
    int year;

    // Constructor to initialize data members
    Car(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }

    // Member functions (methods)
    void start() {
        System.out.println("The car is starting.");
    }

    void stop() {
        System.out.println("The car is stopping.");
    }

    void drive() {
        System.out.println("The car is in motion.");
    }
}
```

```java
public class CarDemo {
    public static void main(String[] args) {
        // Creating an object of the Car class with constructor
        Car myCar = new Car("Toyota", "Camry", 2022);

        // Accessing data members and calling member functions
        System.out.println("My car details:");
        System.out.println("Make: " + myCar.make);
        System.out.println("Model: " + myCar.model);
        System.out.println("Year: " + myCar.year);

        myCar.start(); // Output: The car is starting.
        myCar.drive(); // Output: The car is in motion.
        myCar.stop();  // Output: The car is stopping.
    }
}
```

ENCAPSULATION

ABSTRACTION

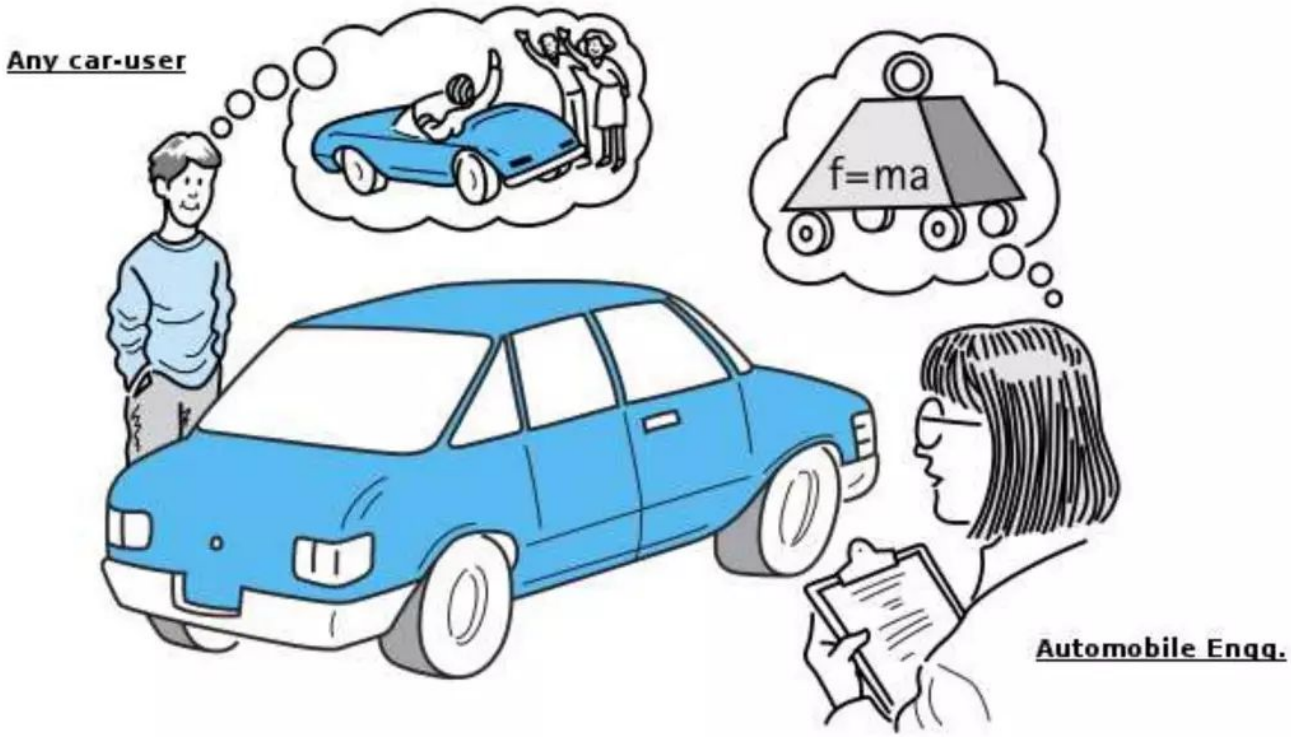
INHERITANCE

POLYMORPHISM

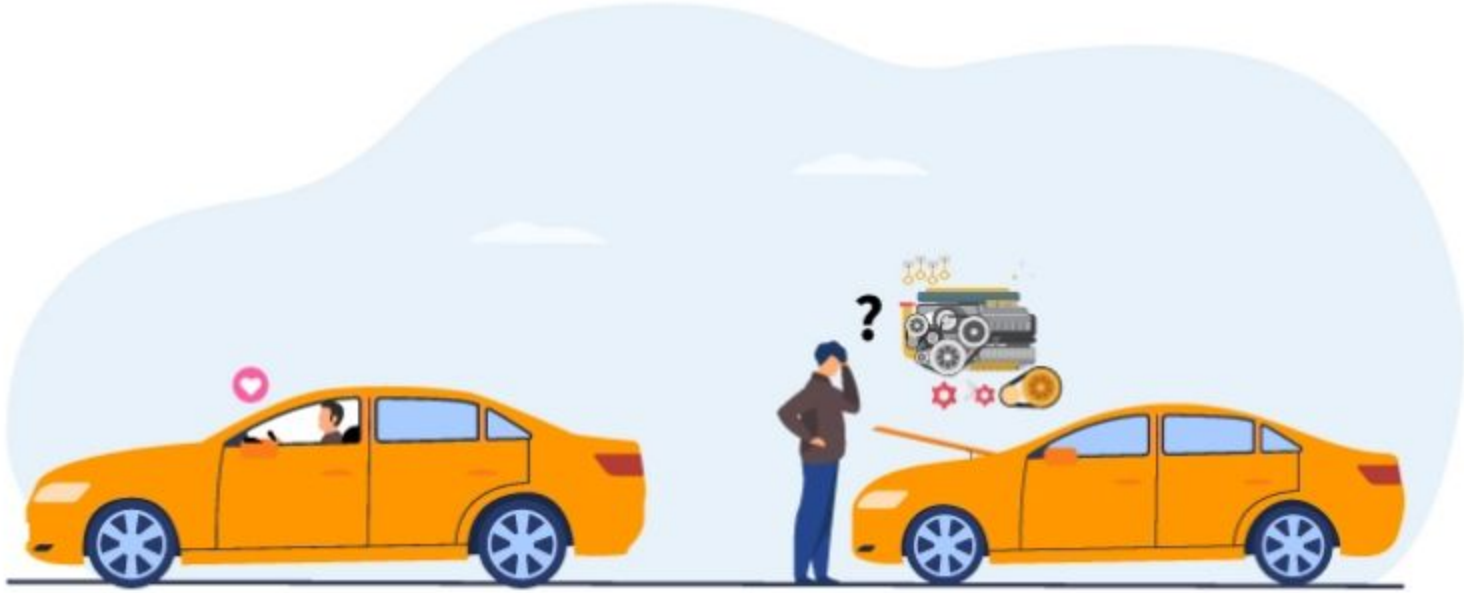
Abstraction

Abstraction is the process of hiding the internal details of an application.

Your car is a great example of abstraction. You can start a car by turning the key or pressing the start button. You don’t need to know how the engine is getting started, what all components your car has. The car internal implementation and complex logic is completely hidden from the user.



An abstraction includes the essential details relative to the perspective of the viewer



SCALER
Topics

Abstraction in C++ and Java

In Java:
Abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.

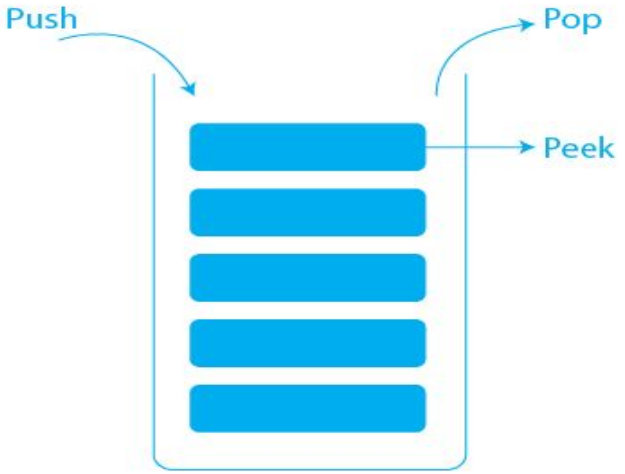
In C++:
Abstraction can also be achieved using header files as we hide the function's implementation in header files. We could use that same function in our program without knowing its inside workings. Sort(), for example, is used to sort an array, a list, or a collection of items, and we know that if we give a container to sort, it will sort it, but we don't know which sorting algorithm it uses to sort that container.

ADT

An Abstract Data Type (ADT) is a programming concept that defines a high-level view of a data structure, without specifying the implementation details.

Examples of abstract data type in data structures are List, Stack, Queue, etc.

A stack is a linear data structure that only allows data to be accessed from the top. It simply has two operations: push (to insert data to the top of the stack) and pop (to remove data from the stack). (used to remove data from the stack top).



Some of the most essential operations defined in Stack ADT are listed below.

- **top():** returns the value of the node present at the top of the stack.
- **push(int val):** creates a node with value = val and puts it at the stack top.
- **pop():** removes the node from the top of the stack.
- **empty():** returns true if the stack is empty, otherwise returns false.
- **size():** returns the number of nodes that are present in the stack.

Queue ADT

A queue is a linear data structure that allows data to be accessed from both ends. There are two main operations in the queue: push (this operation inserts data to the back of the queue) and pop (this operation is used to remove data from the front of the queue).

Some of the most essential operations defined in Queue ADT are listed below.

- **front():** returns the value of the node present at the front of the queue.
- **back():** returns the value of the node present at the back of the queue.
- **push(int val):** creates a node with value = val and puts it at the front of the queue.
- **pop():** removes the node from the rear of the queue.
- **empty():** returns true if the queue is empty, otherwise returns false.
- **size():** returns the number of nodes that are present in the queue.

Encapsulation

Data Encapsulation can be defined as wrapping the code or methods(properties) and the related fields or variables together as a single unit.

In object-oriented programming, we call this single unit - a class, interface, etc.

Encapsulation in C++

Encapsulation in Java :
Use the private access modifier to declare all variables/fields of class as private.

Define public getter and setter methods to read and modify/set the values of the above said fields.

In C++, encapsulation can be implemented using classes and access modifiers.

```java
public class Person {
    // Private attribute (encapsulated data)
    private String name;

    // Constructor to initialize the object
    public Person(String name) {
        this.name = name;
    }

    // Getter method for the name attribute
    public String getName() {
        return name;
    }

    // Setter method for the name attribute
    public void setName(String name) {
        this.name = name;
    }
}

public class EncapsulationDemo {
    public static void main(String[] args) {
        // Create a Person object
        Person person = new Person("John");

        // Access and modify the name attribute using getter and setter meth
        String currentName = person.getName();
        System.out.println("Current Name: " + currentName);

        person.setName("Alice");
        String newName = person.getName();
        System.out.println("New Name: " + newName);
    }
}
```

Polymorphism

- Polymorphism is the ability of an entity to take several forms. In object-oriented programming, it refers to the ability of an object (or a reference to an object) to take different forms of objects.

- It allows a common data-gathering message to be sent to each class.

- Polymorphism encourages called as ‘extendibility’ which means an object or a class can have it’s uses extended.

edureka!

How it is Achieved

- Polymorphism in object-oriented programming is achieved through the following mechanisms:

- **Method Overloading:**
  Multiple methods in a class with the same name but different parameter lists (number or types of parameters).

- **Method Overriding:**
  Method overriding occurs when a subclass provides a specific implementation for a method that is already defined in its superclass.

```java
public class Calculator {
    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }

    // Method to add three integers
    public int add(int a, int b, int c) {
        return a + b + c;
    }

    public static void main(String[] args) {
        Calculator calculator = new Calculator();

        int sum2 = calculator.add(5, 3);
        System.out.println("Sum of two integers: " + sum2);

        int sum3 = calculator.add(2, 3, 4);
        System.out.println("Sum of three integers: " + sum3);
    }
}
```

```java
class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }
}

class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a circle");
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape = new Shape();
        shape.draw(); // Output: Drawing a shape

        Circle circle = new Circle();
        circle.draw(); // Output: Drawing a circle
    }
}
```

Inheritance

Just like in real life, where individuals inherit wealth, responsibilities, and characteristics from their parents.

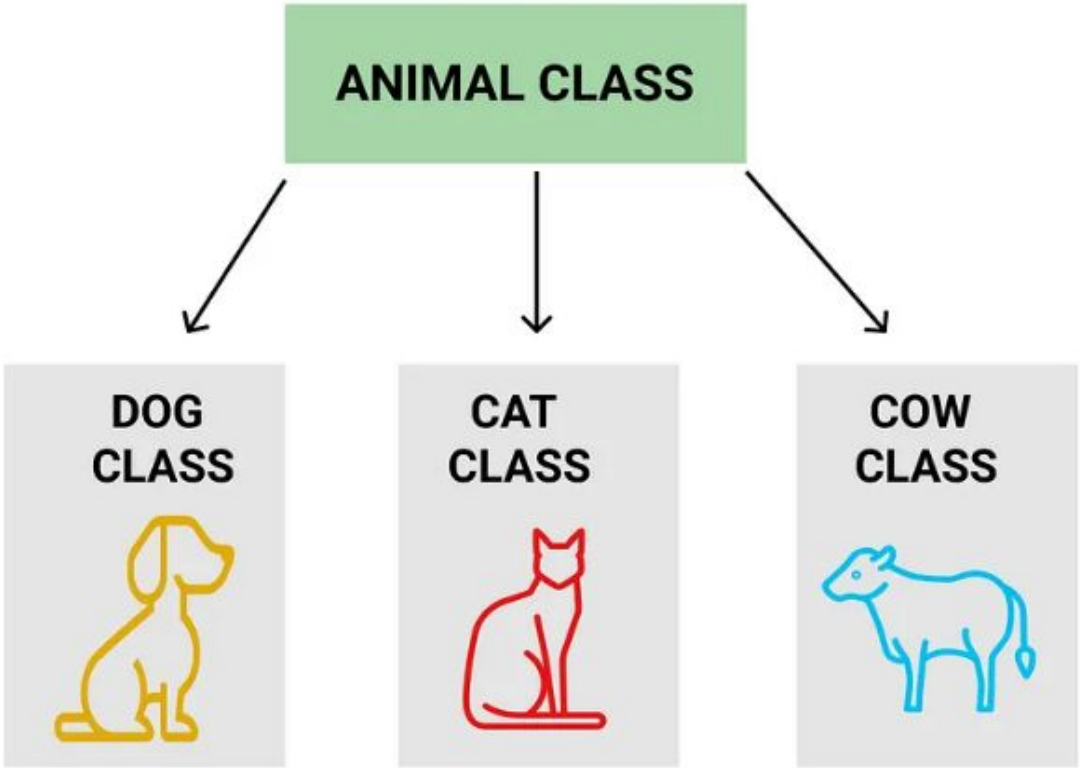
Inheritance in object-oriented programming (OOP) enables the reuse of code and facilitates the creation of powerful software systems.

Inheritance in OOP => What a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

Who are the characters in this picture :

The real life example of inheritance is child and parents, all the properties of father are inherited by his son.

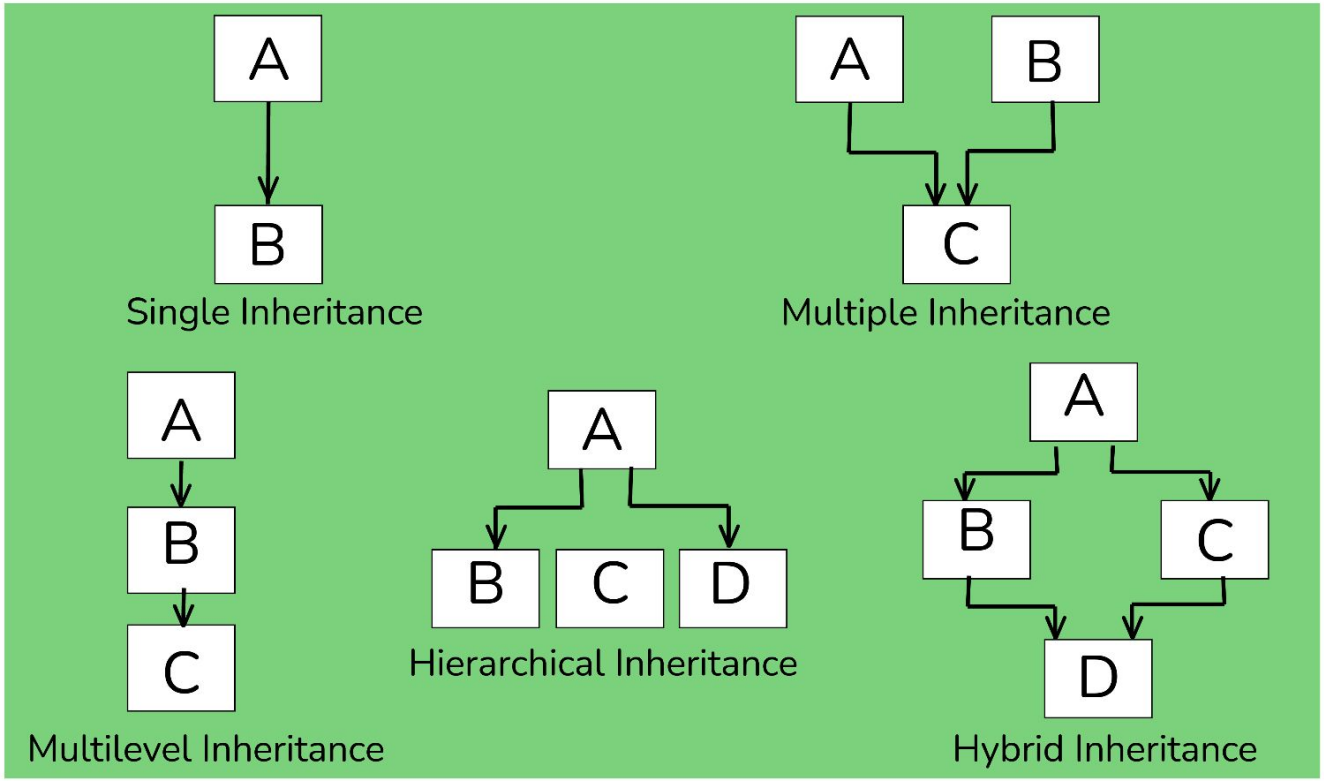


Inheritance example in real world

Java

```java
// Java Program to illustrate Inheritance (concise)

import java.io.*;

// Base or Super Class
class Employee {
    int salary = 60000;
}

// Inherited or Sub Class
class Engineer extends Employee {
    int benefits = 10000;
}

// Driver Class
class Gfg {
    public static void main(String args[])
    {
        Engineer E1 = new Engineer();
        System.out.println("Salary : " + E1.salary
                           + "\nBenefits : " + E1.benefits);
    }
}
```

Types Of Inheritance



Hands On Implementation.

Parallax Effect

Parallax Scrolling is an important concept in game development. It refers to the illusion of movement given two frames of reference that are moving at different rates.

For a real life example, consider riding in a car and looking out the window. Suppose you’re driving near mountains and there are railings on the side of the road. You might notice that the railings may appear to be moving much more quickly in your frame of vision than the mountains.

We want to simulate this behavior in our game with the ground and background to give the appearance of motion.

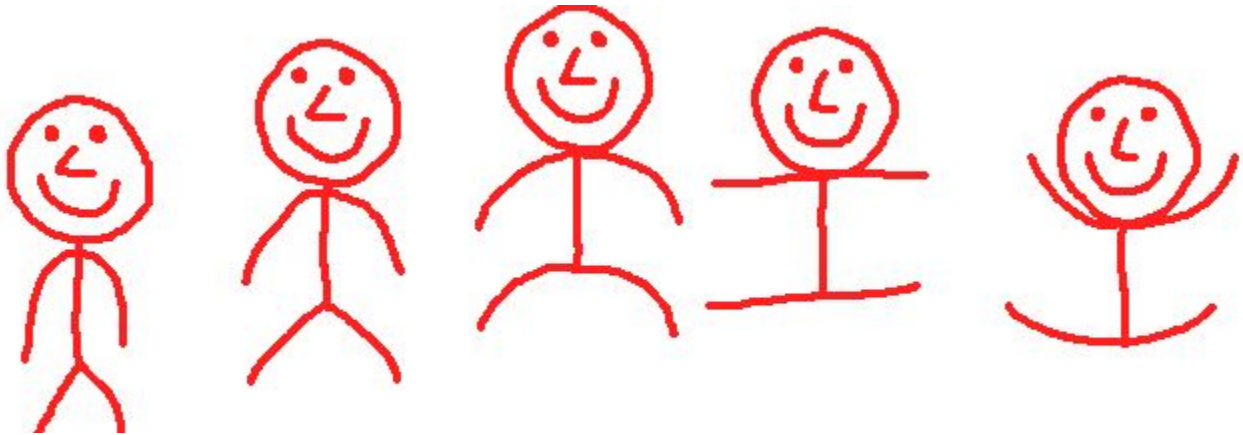
Hands On Implementation.

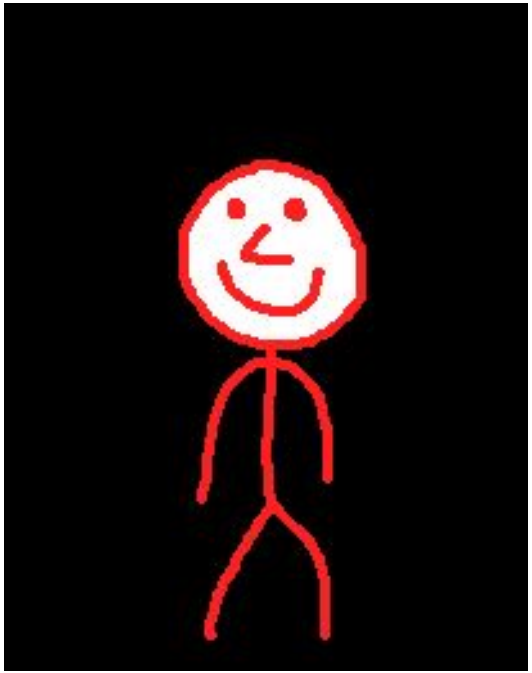
Sprite Animation

Sprite animation is a technique used in game development to make objects or characters in a game appear to move.

It works by showing a series of images, called sprites, one after the other at a rapid pace.

This creates the illusion of motion, just like in old-fashioned flipbooks





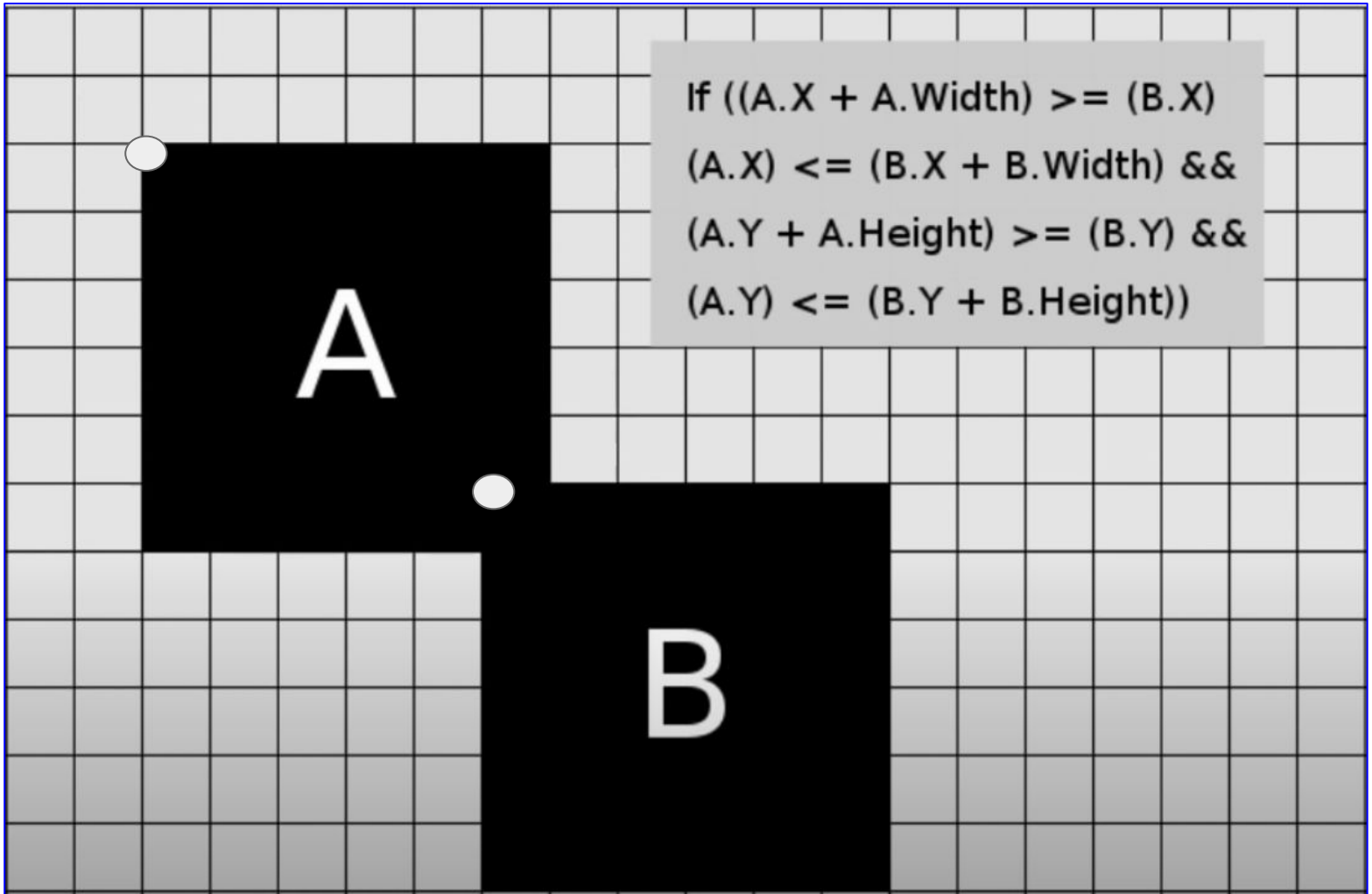

THE BOY
FREE SPRITES
IDLE
WALK
RUN
JUMP
DEAD

Hands On Implementation.

Axis Aligned Bounding Box

AABB Collision Detection relies on all colliding entities to have “axis-aligned bounding boxes”, which simply means their collision boxes contain no rotation in our world space, which allows us to use a simple math formula to test for collision.

As with 2D collision detection, axis-aligned bounding boxes (AABB) are the quickest algorithm to determine whether the two game entities are overlapping or not.


If ((A.X + A.Width) >= (B.X)
(A.X) <= (B.X + B.Width) &&
(A.Y + A.Height) >= (B.Y) &&
(A.Y) <= (B.Y + B.Height))
A
B

```
-- collision conditions, if overlap happens then collision is true --
function checkCollision(obj1, obj2)
    return obj1.x < obj2.x + obj2.width and
           obj2.x < obj1.x + obj1.width and
           obj1.y < obj2.y + obj2.height and
           obj2.y < obj1.y + obj1.height
end
```

Hands On Implementation.

Player Logic Chase:

All you need to know is:

angle = **math.atan2**(target_y - object_y, target_x - object_x)

math.atan2 returns an angle in radians.

The returned angle is between -3.14 and 3.14.

In Lua we can get **π** by using **math.pi**.

```
function love.update(dt)
    --love.mouse.getPosition returns the x and y position of the cursor.
    mouse_x, mouse_y = love.mouse.getPosition()

    angle = math.atan2(mouse_y - circle.y, mouse_x - circle.x)

    cos = math.cos(angle)
    sin = math.sin(angle)

    --Make the circle move towards the mouse
    circle.x = circle.x + circle.speed * cos * dt
    circle.y = circle.y + circle.speed * sin * dt
end

function love.draw()
    love.graphics.circle("line", circle.x, circle.y, circle.radius)

    --The angle
    love.graphics.line(circle.x, circle.y, mouse_x, mouse_y)
    love.graphics.line(circle.x, circle.y, mouse_x, circle.y)
    love.graphics.line(circle.x, circle.y, circle.x, mouse_y)

end
```

Hands On Implementation.

Write-on effect with substring

A substring is a contiguous sequence of characters within a longer string.

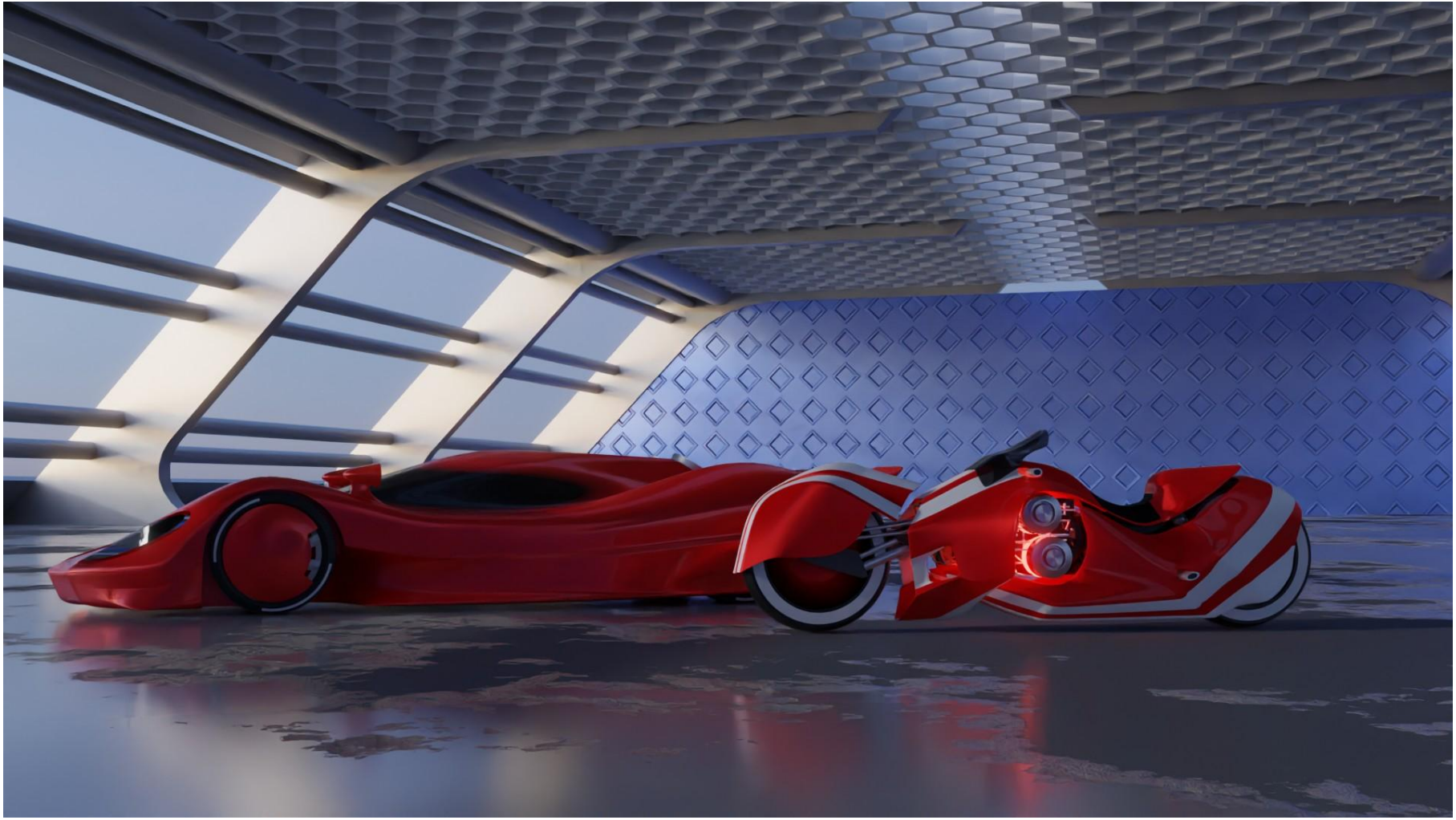
String "Hello, World!" and you want to extract the substring "World" from it, "World" would be considered a substring of the original string.

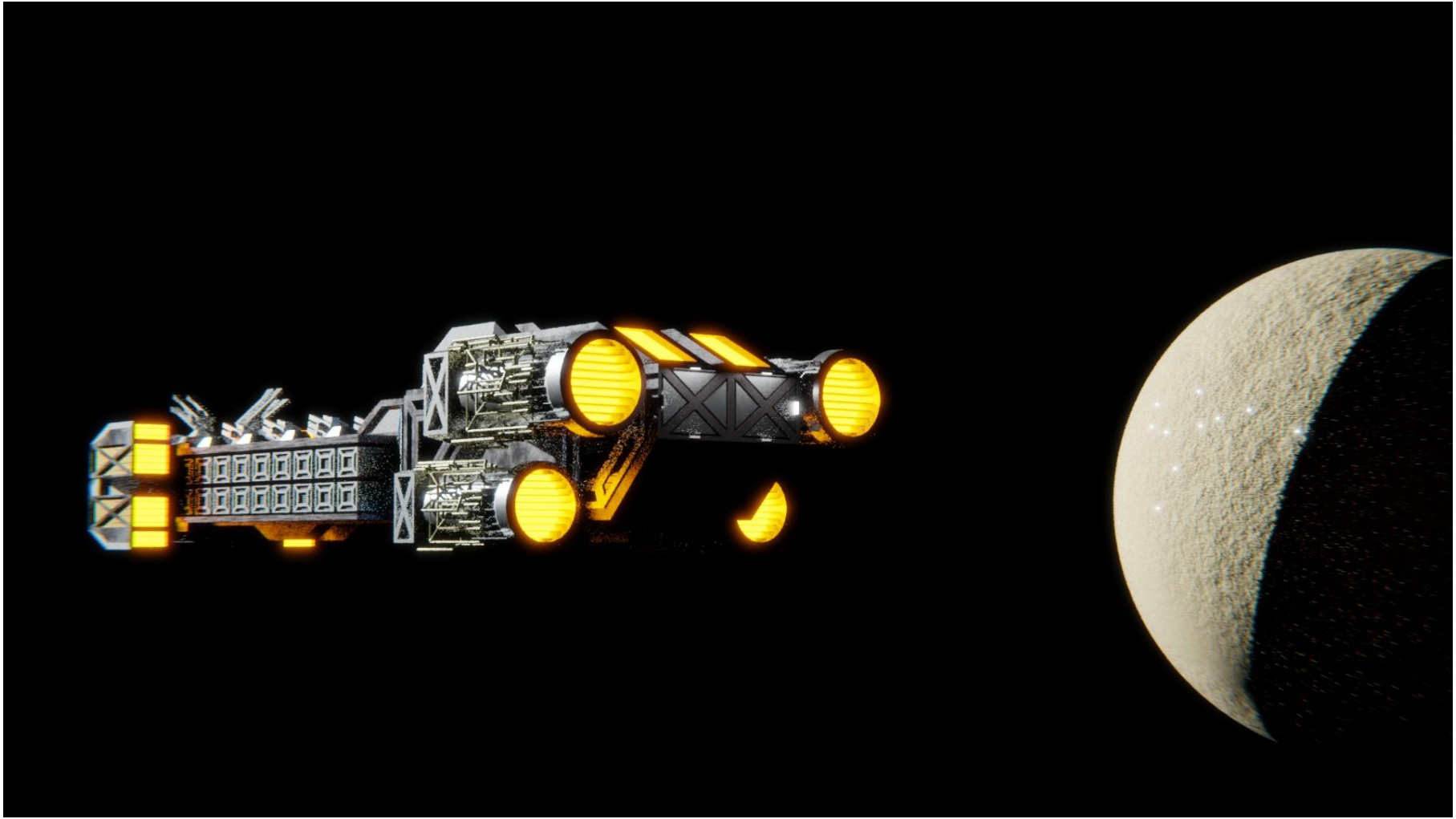
Hands On Implementation.

Extra Features

- Audio in LOVE2D
- Camera Handling using Hump
- Awesome Love2D Libraries

Overview of GameDev (Tech/Non tech)



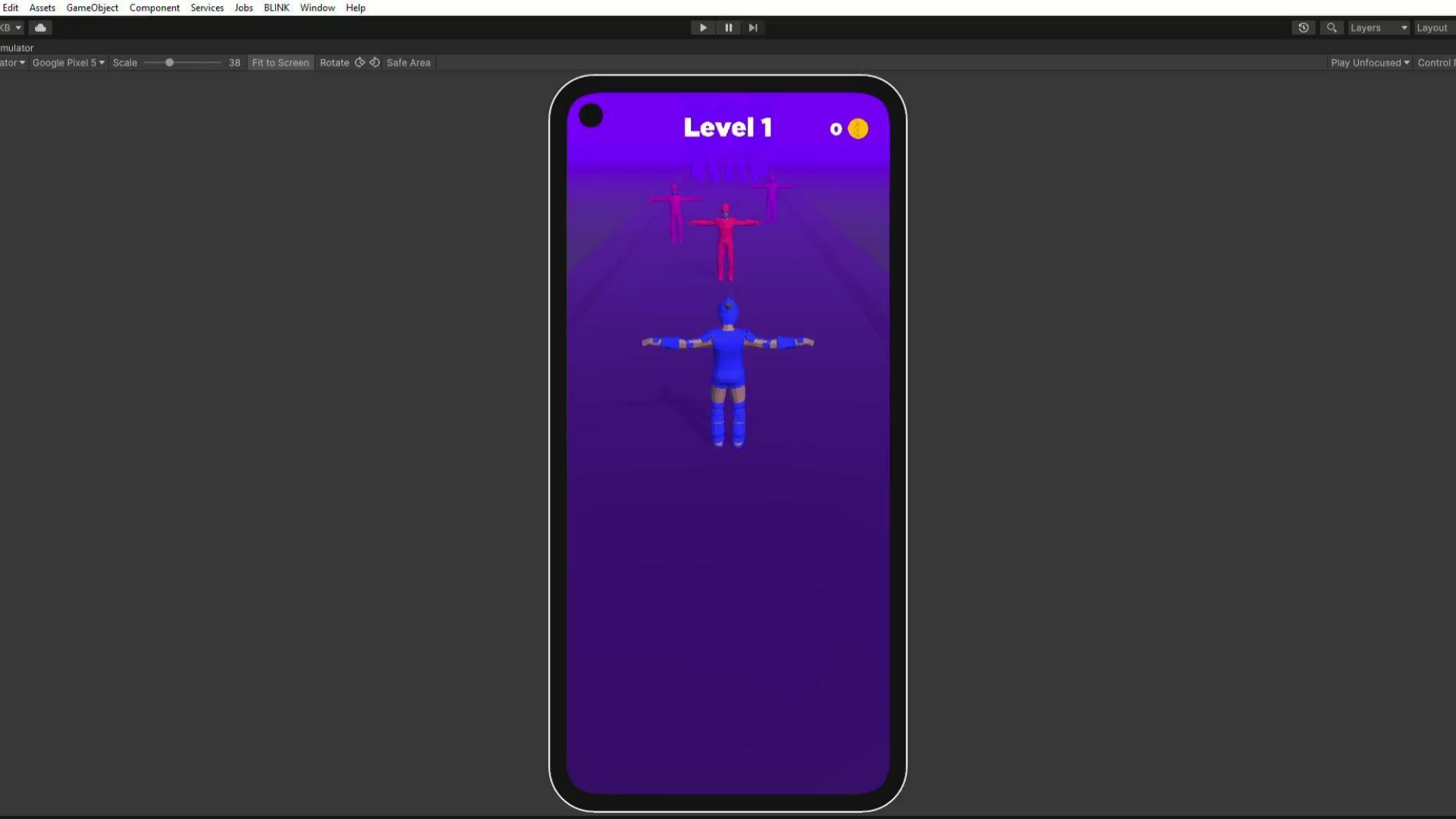


JDM Beauties

proudly_prajwal

GDU PICT Showcase


PLAY
EXIT

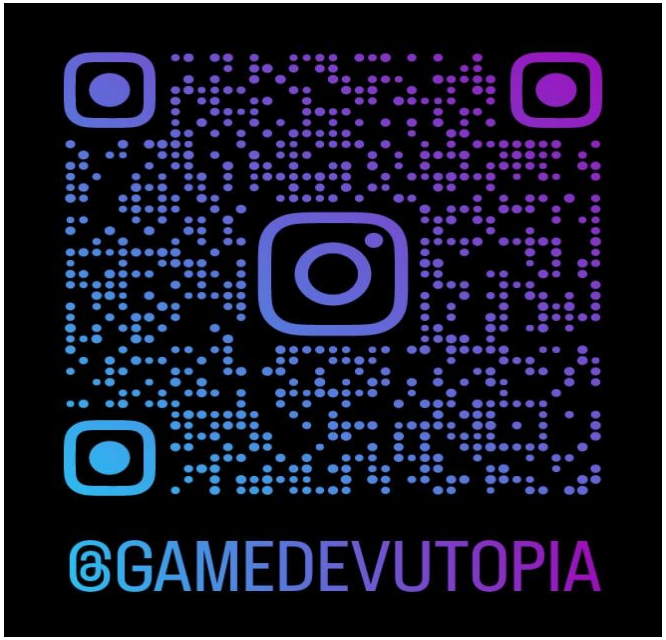

Edit Assets GameObject Component Services Jobs BLINK Window Help
Layers
Layout
mulator
Google Pixel 5
Scale
38
Fit to Screen
Rotate
Safe Area
Play Unfocused
Level 1
0

GDU IIIT-K
Showcase


PLAY
EXIT


Edit Assets GameObject Component Jobs Window Help

Contact Us



THANK YOU