# Advanced Programming Methods
Lecture 13

Bogdan Dumbrăvean
bogdan.dumbravean@ubbcluj.ro

Babeș-Bolyai University

2024

## Objectives

- **C#**
  - Windows Forms App
  - LINQ
- **Raycasting**
  - Sphere-Line Intersection
  - Lighting
- **Unity**
  - What it is
  - How it uses C#
  - Shaders

Front-End Developer
Vs
Back-End Developer
Vs
Full-Stack Developer
JS
php
.NET
ORACLE
aws
{DEV}

Image Source: cmarix blog

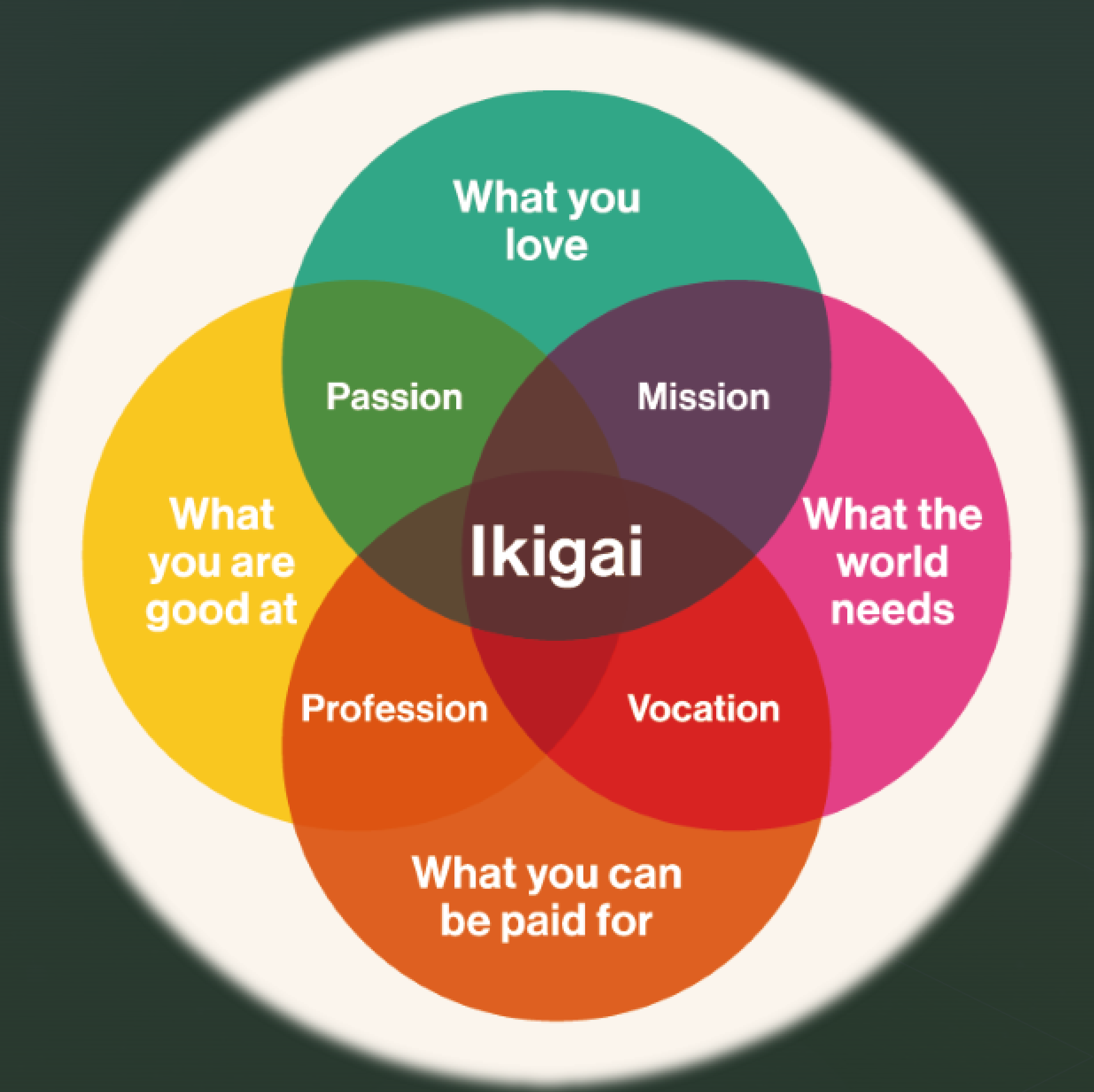


Image Source: [JapanGov Ikigai]

## C# Windows Forms Applications

- **What is C#?**
  - C# (pronounced "C-Sharp") is a modern, object-oriented programming language developed by Microsoft.
  - It is part of the .NET framework and is widely used for building Windows applications, web services, and games.

- **What are Windows Forms?**
  - Windows Forms (WinForms) is a GUI (Graphical User Interface) toolkit for building desktop applications in C#.
  - Allows rapid development of Windows-based applications with visual design tools in Visual Studio.

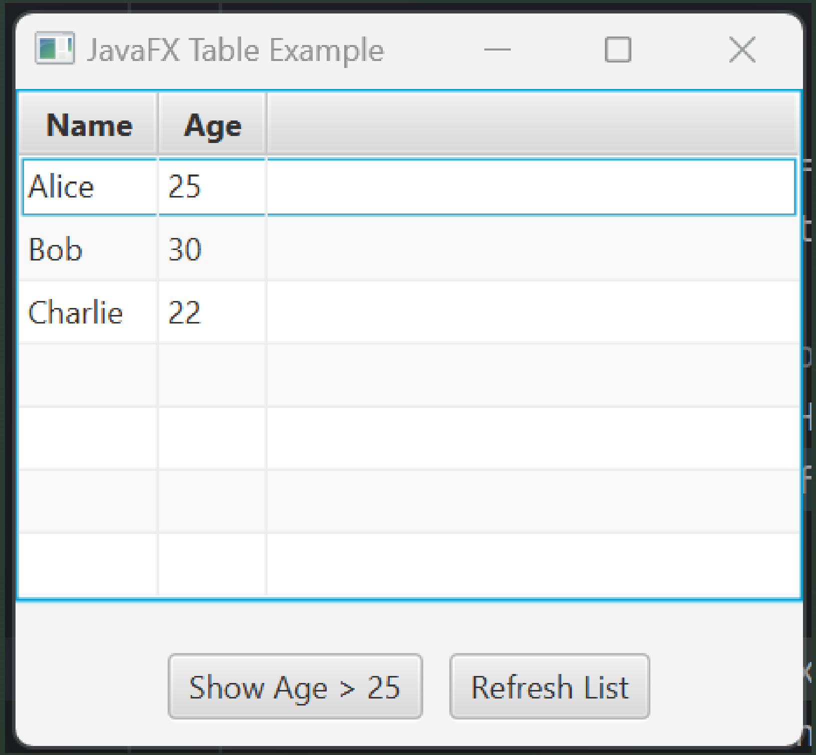
- **Key Features:**
  - Drag-and-drop UI design.
  - Event-driven programming model.
  - Rich library of controls (buttons, textboxes, data grids, etc.).
  - Easy integration with databases and external services.

## Structure of a Windows Forms App

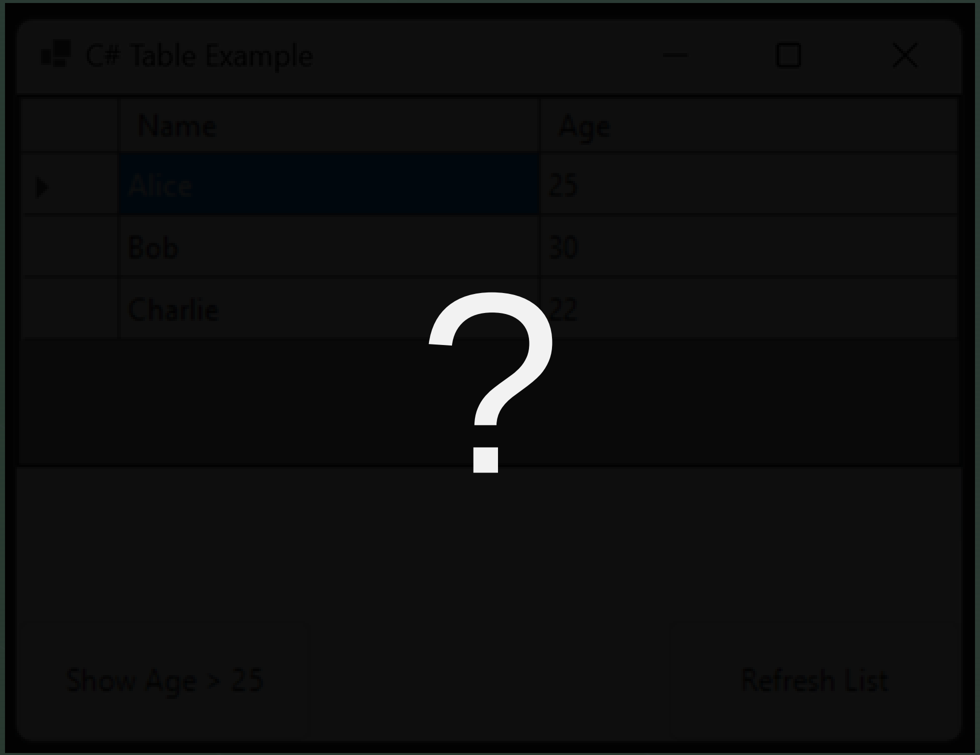
**Key Components:**

- **Form (Window):**
  - Represents a window or dialog box in the application.
  - Serves as a container for other UI components.
- **Controls (UI Elements):**
  - Examples: Buttons, Labels, TextBoxes, DataGridView.
  - Allow user interaction and data display.
- **Events:**
  - Actions triggered by user interactions (e.g., button clicks, text input).
  - Handled by event handlers (methods that respond to events).

## Demo


JavaFX Table Example

| Name | Age |
|---|---|
| Alice | 25 |
| Bob | 30 |
| Charlie | 22 |


C# Table Example

| Name | Age |
|---|---|
| Alice | 25 |
| Bob | 30 |
| Charlie | 22 |

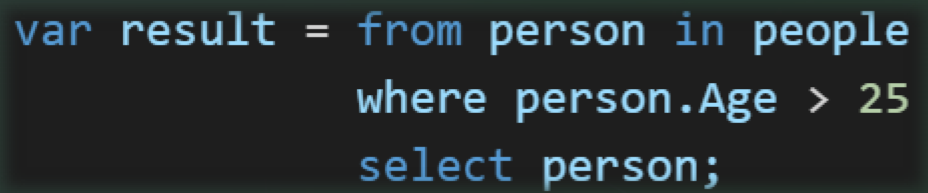
## Introduction to LINQ

- **What is LINQ?**
  - Language Integrated Query
  - LINQ is a feature in C# that provides a consistent way to query and manipulate data.
  - It integrates query capabilities directly into C# syntax, allowing data processing from arrays, collections, databases, XML, and more.

- **Key Features:**
  - Unified query syntax for various data sources.
  - Strongly typed queries with IntelliSense support.
  - Supports both Method Syntax and Query Syntax.
  - Extensible through custom query providers.

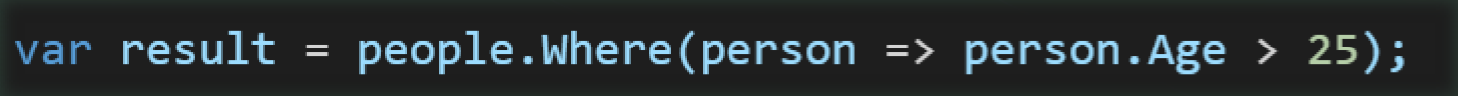
## Introduction to LINQ

- **Query Syntax:**
  - SQL-like syntax for queries.



```
var result = from person in people
             where person.Age > 25
             select person;
```

- **Method Syntax:**
  - Uses method calls for queries.



```
var result = people.Where(person => person.Age > 25);
```

- **Common LINQ Methods:**
  - **Filtering:** `Where`
  - **Projection:** `Select`
  - **Sorting:** `OrderBy`, `OrderByDescending`
  - **Grouping:** `GroupBy`
  - **Aggregates:** `Sum`, `Average`, `Count`

## Introduction to LINQ

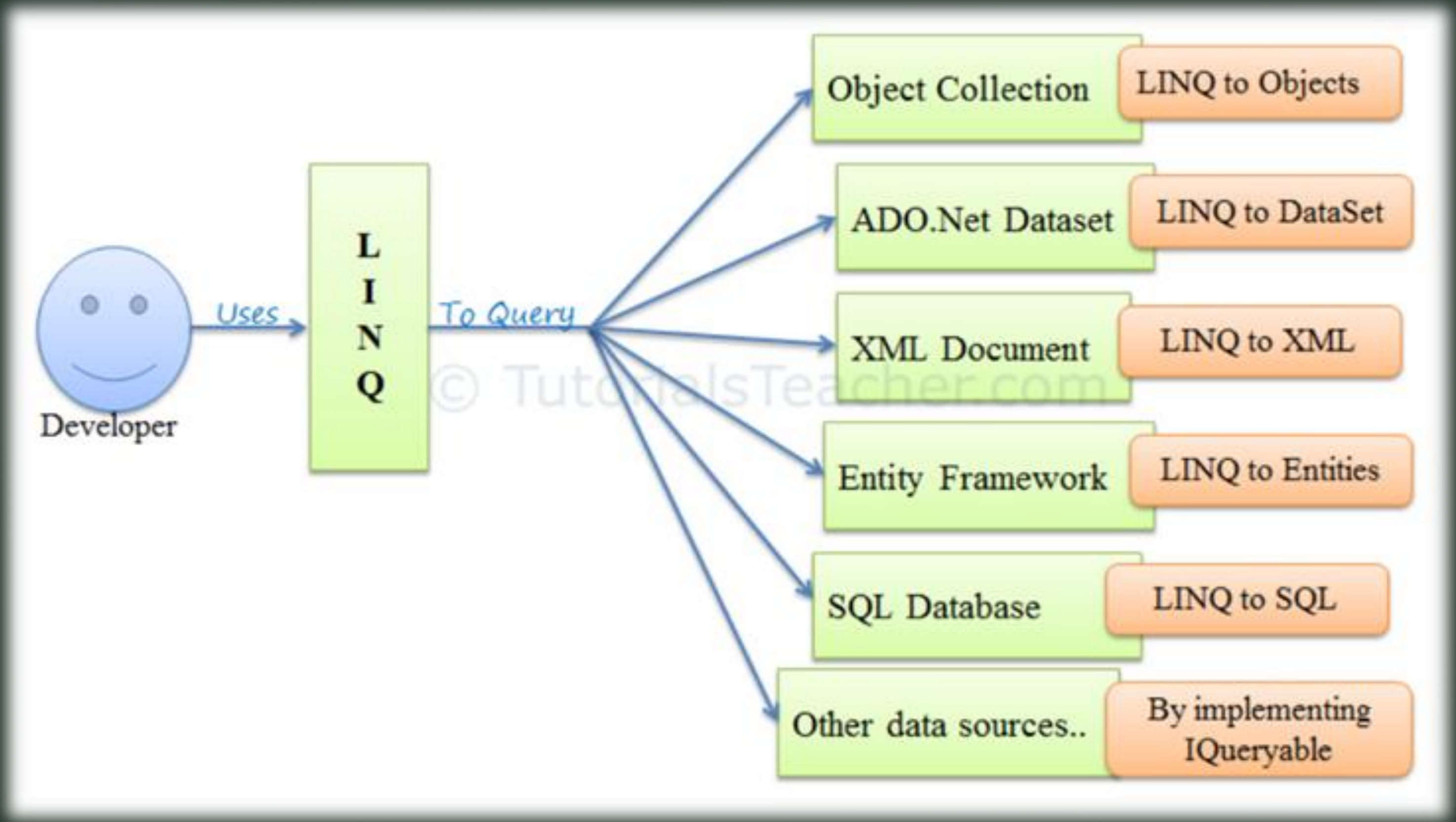
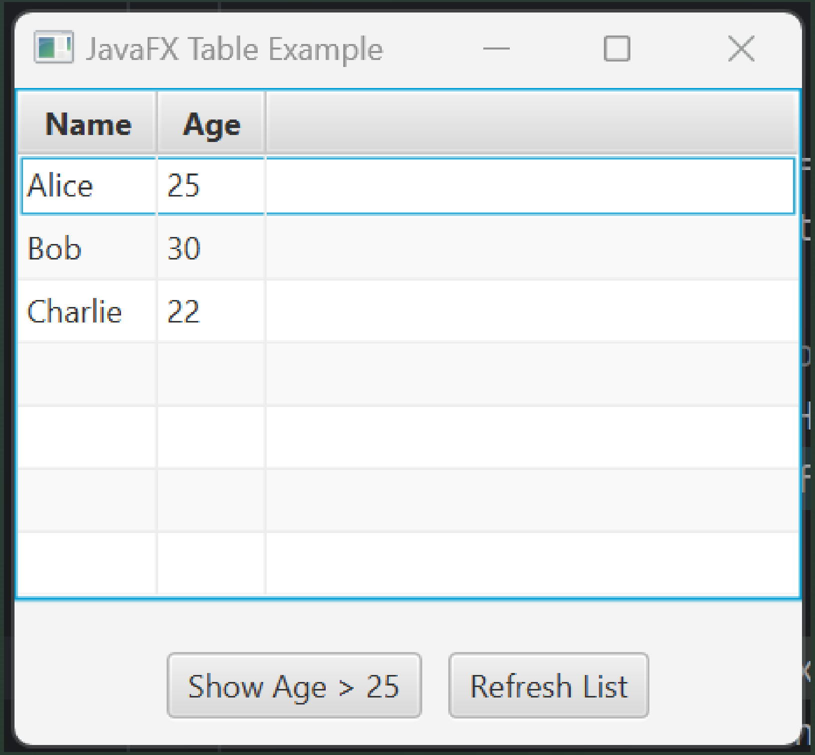
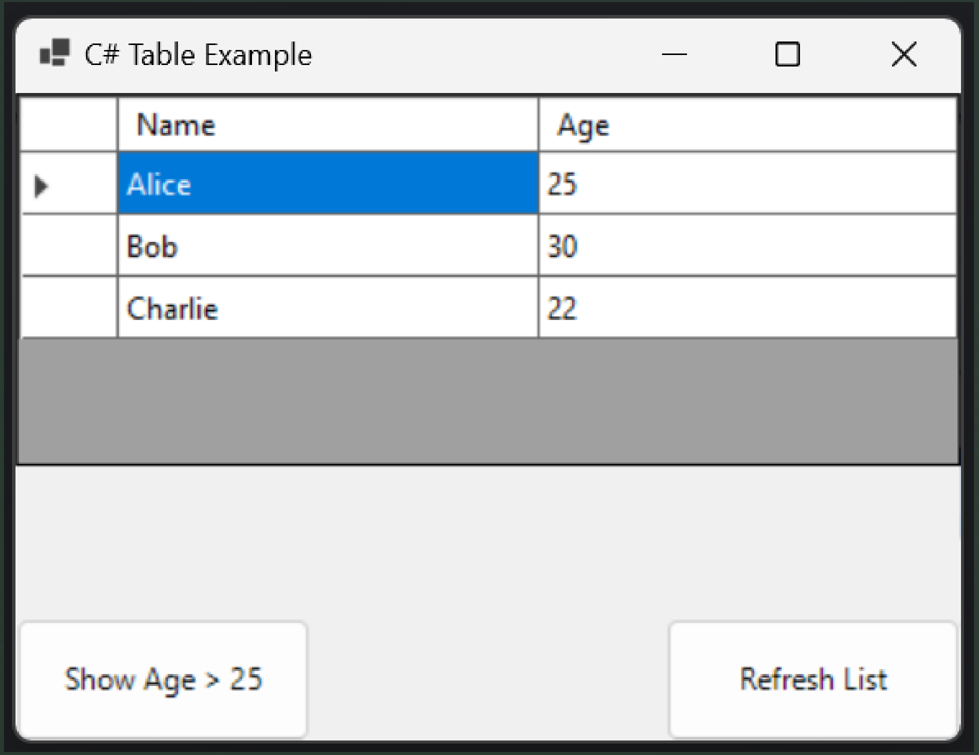


Image Source: What is LINQ?

## Demo


JavaFX Table Example

| Name | Age |
|---|---|
| Alice | 25 |
| Bob | 30 |
| Charlie | 22 |


C# Table Example

| Name | Age |
|---|---|
| Alice | 25 |
| Bob | 30 |
| Charlie | 22 |

## Key Differences to Highlight

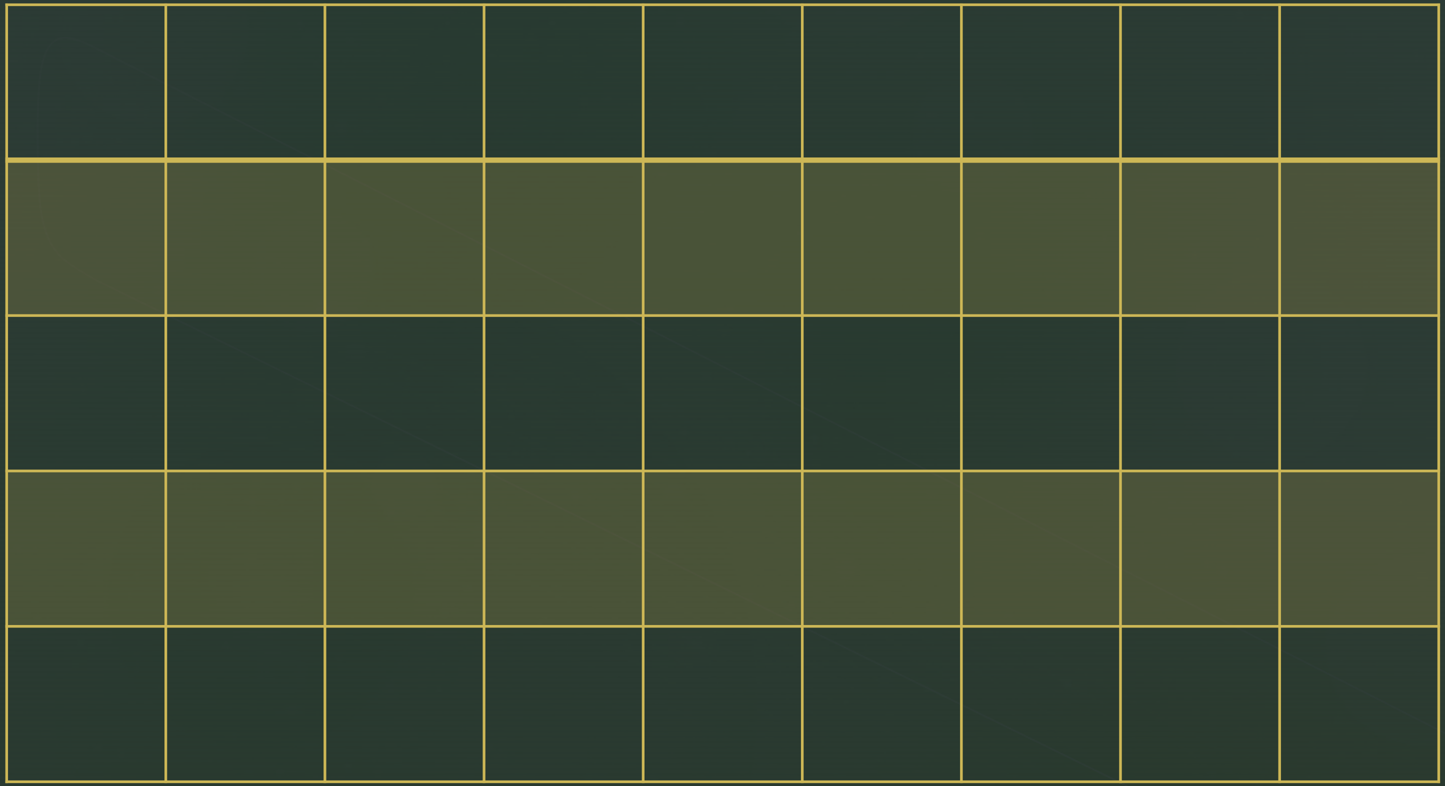
- **ObservableList vs. List**:
  - JavaFX uses ObservableList.
  - C# uses List with LINQ for filtering.

- **Table Component**:
  - JavaFX uses TableView.
  - C# uses DataGridView.

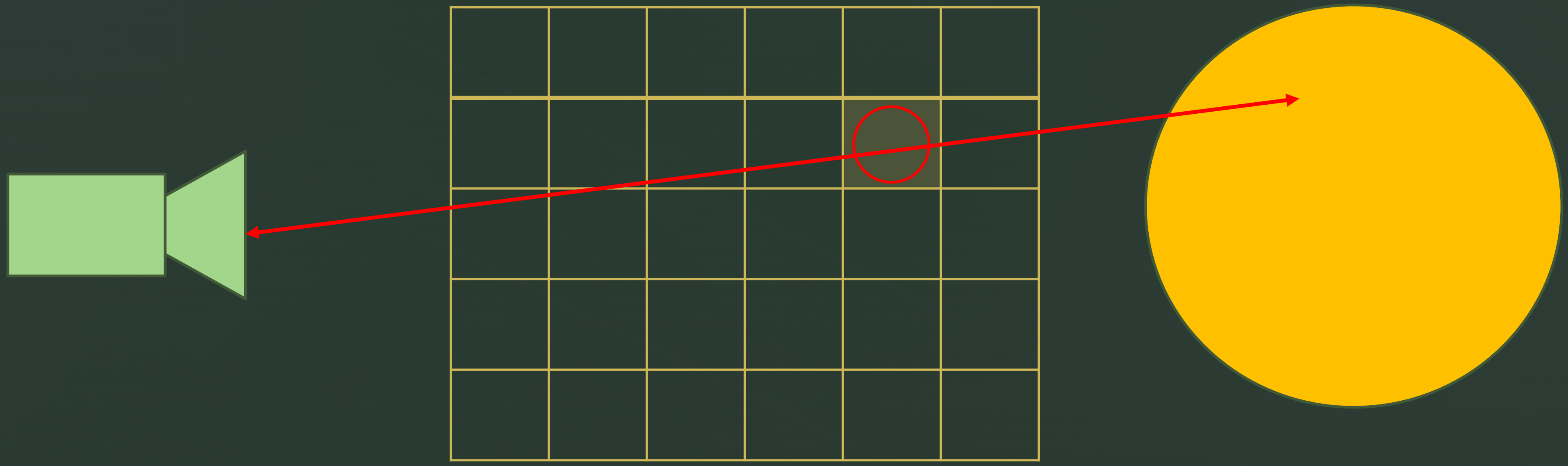
- **Event Handling**:
  - JavaFX uses lambda expressions with setOnAction.
  - C# uses event handlers with +=.

- **GUI Toolkit**:
  - JavaFX integrates layouts (like VBox) directly into code.
  - C# Windows Forms uses docking (DockStyle) for layout.

## Raycasting



## Raycasting



## Sphere-Line Intersection

- **Sphere Equation**:

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = R^2$$

Where:

- C is the center of the sphere.
- R is the radius of the sphere.

- **Line (Ray) Equation**:

$$P(t) = O + tD$$

Where:

- O is the origin point of the line.
- D is the direction vector of the line (normalized).
- t is a scalar parameter.

## Sphere-Line Intersection

- Substitute the line equation in the sphere equation:

$$||P(t) - C||^2 = R^2$$

$$||(O + tD) - C||^2 = R^2$$

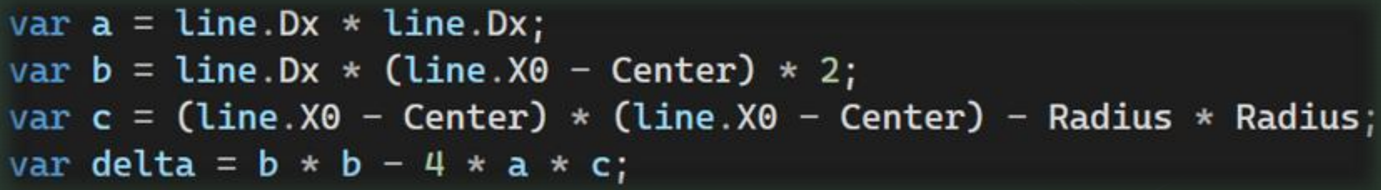
- Reorder to separate t:

$$||O - C + tD||^2 = R^2$$

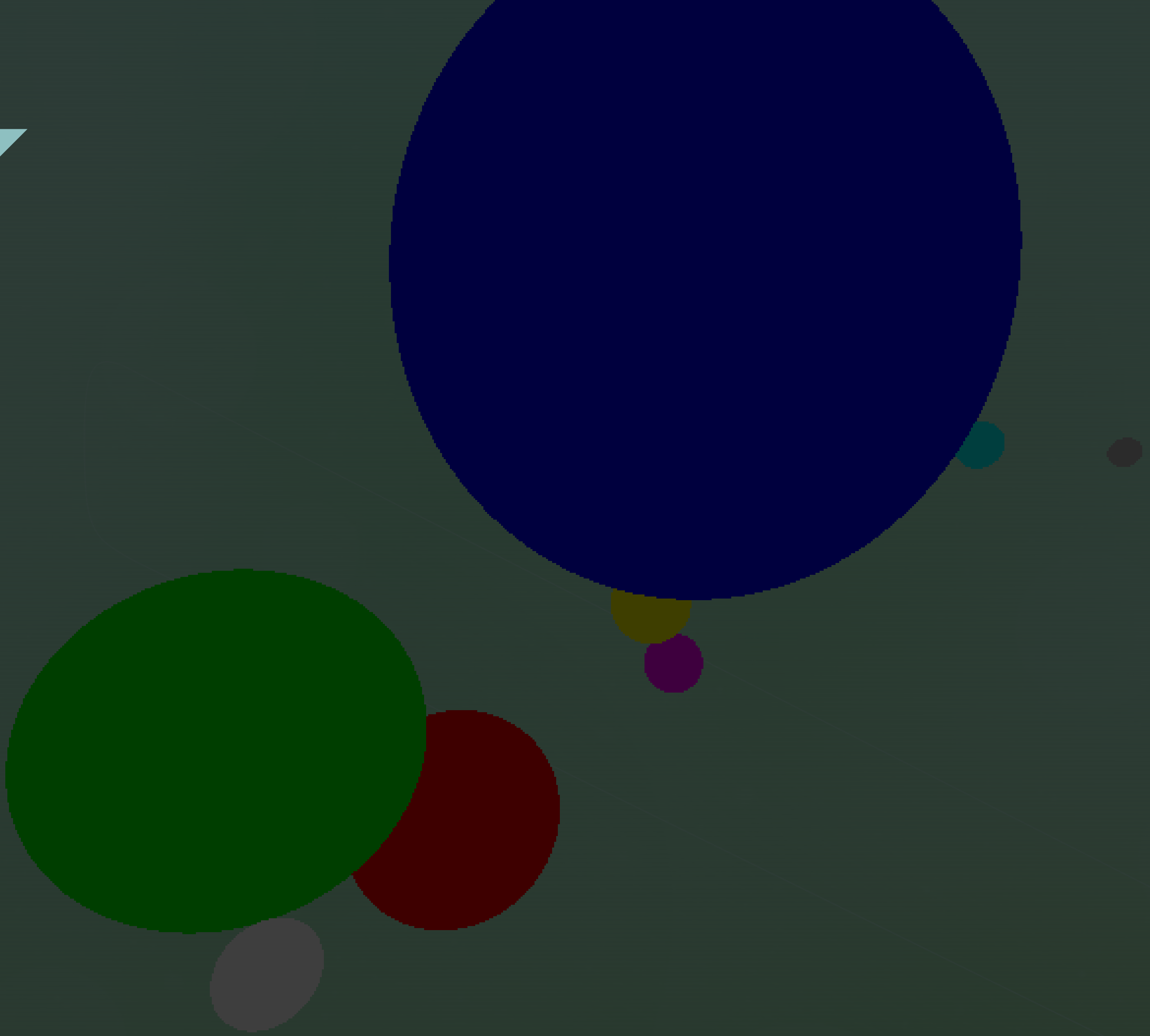
- Expand then simplify:

$$(O - C) \cdot (O - C) + 2tD \cdot (O - C) + t^2(D \cdot D) = R^2$$

$$t^2 + 2tD \cdot (O - C) + ||O - C||^2 - R^2 = 0$$

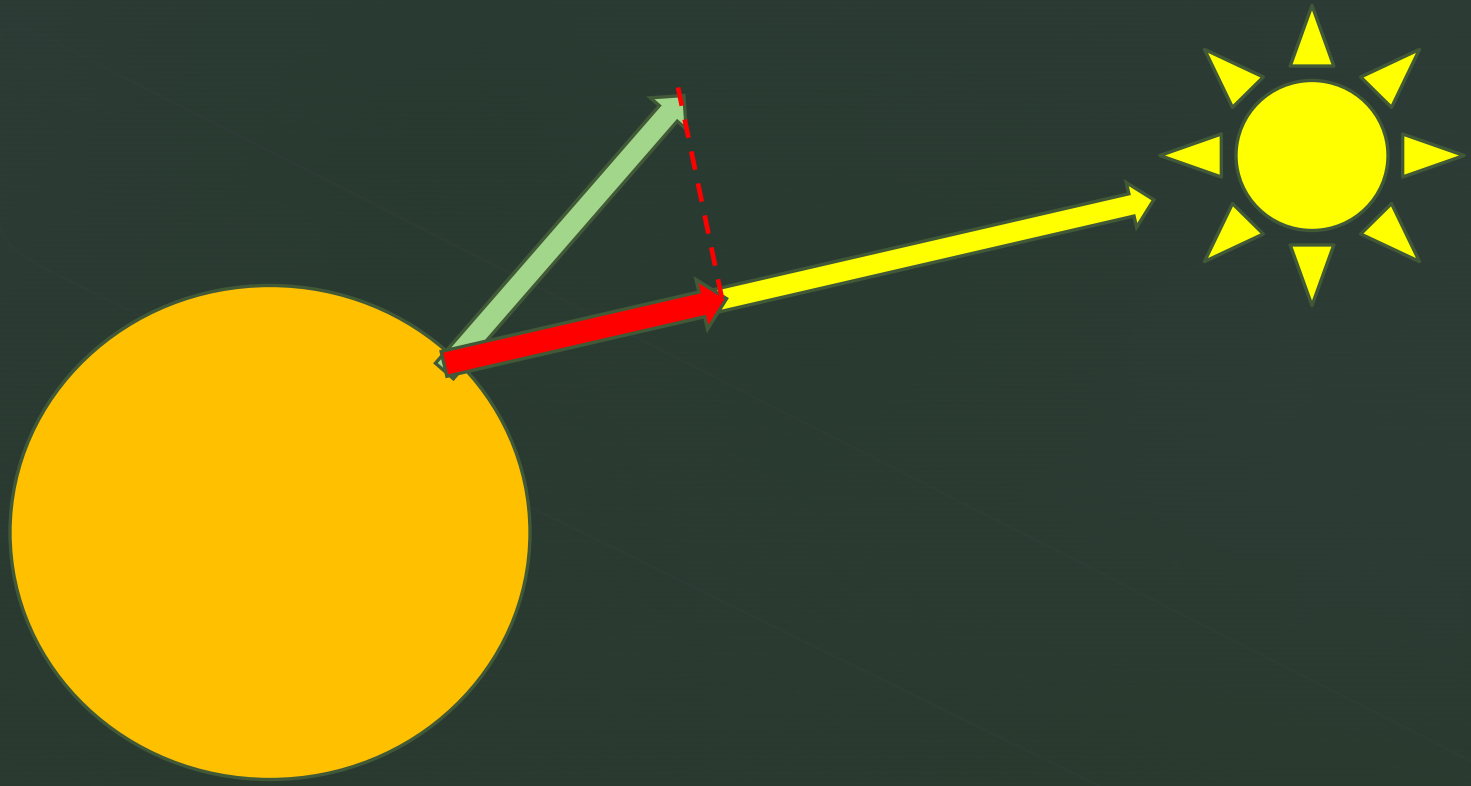


```
var a = line.Dx * line.Dx;
var b = line.Dx * (line.X0 - Center) * 2;
var c = (line.X0 - Center) * (line.X0 - Center) - Radius * Radius;
var delta = b * b - 4 * a * c;
```



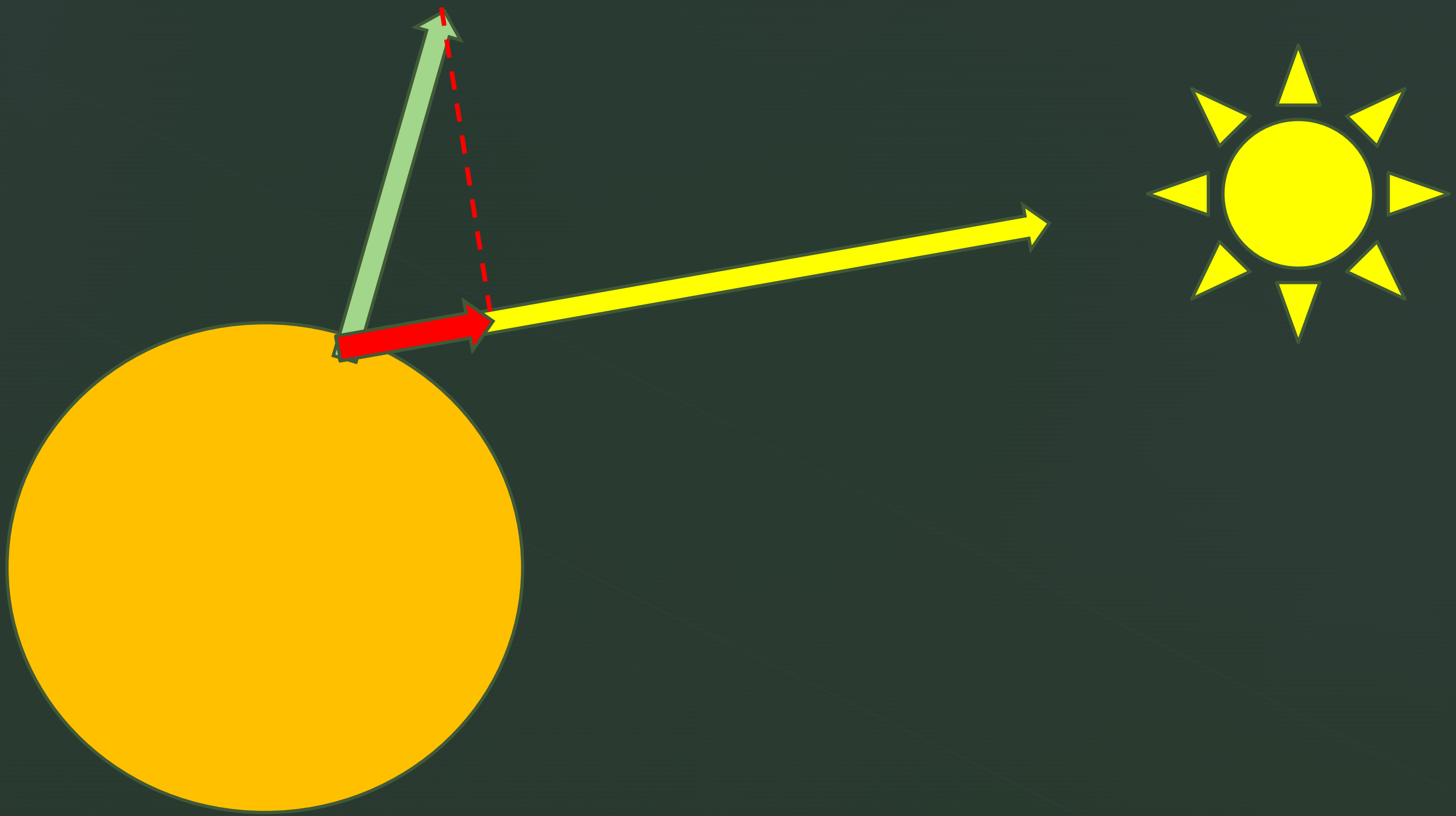
## Diffuse Lighting

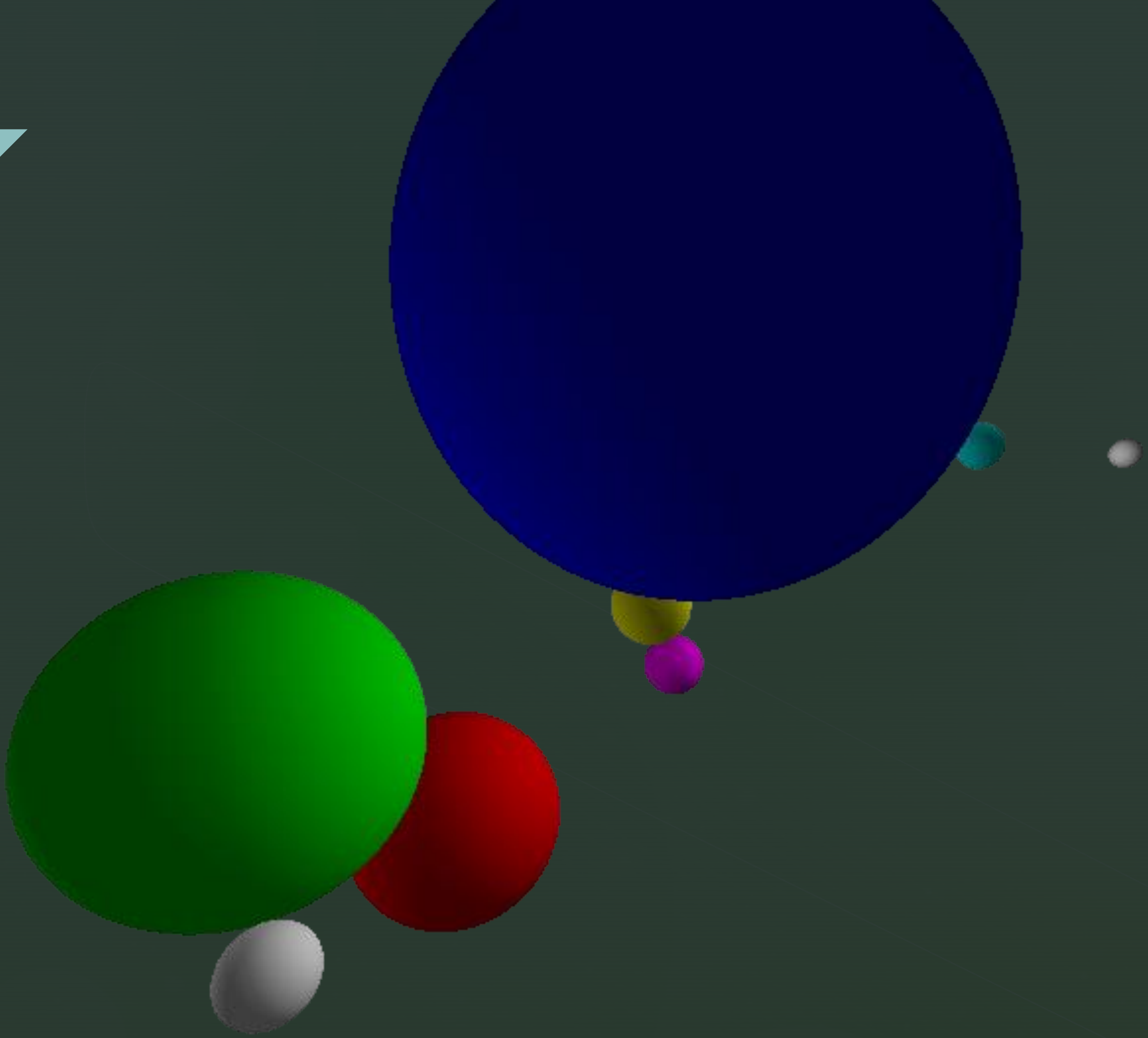
Use dot product to add light based on direction (projecting normal):



## Diffuse Lighting

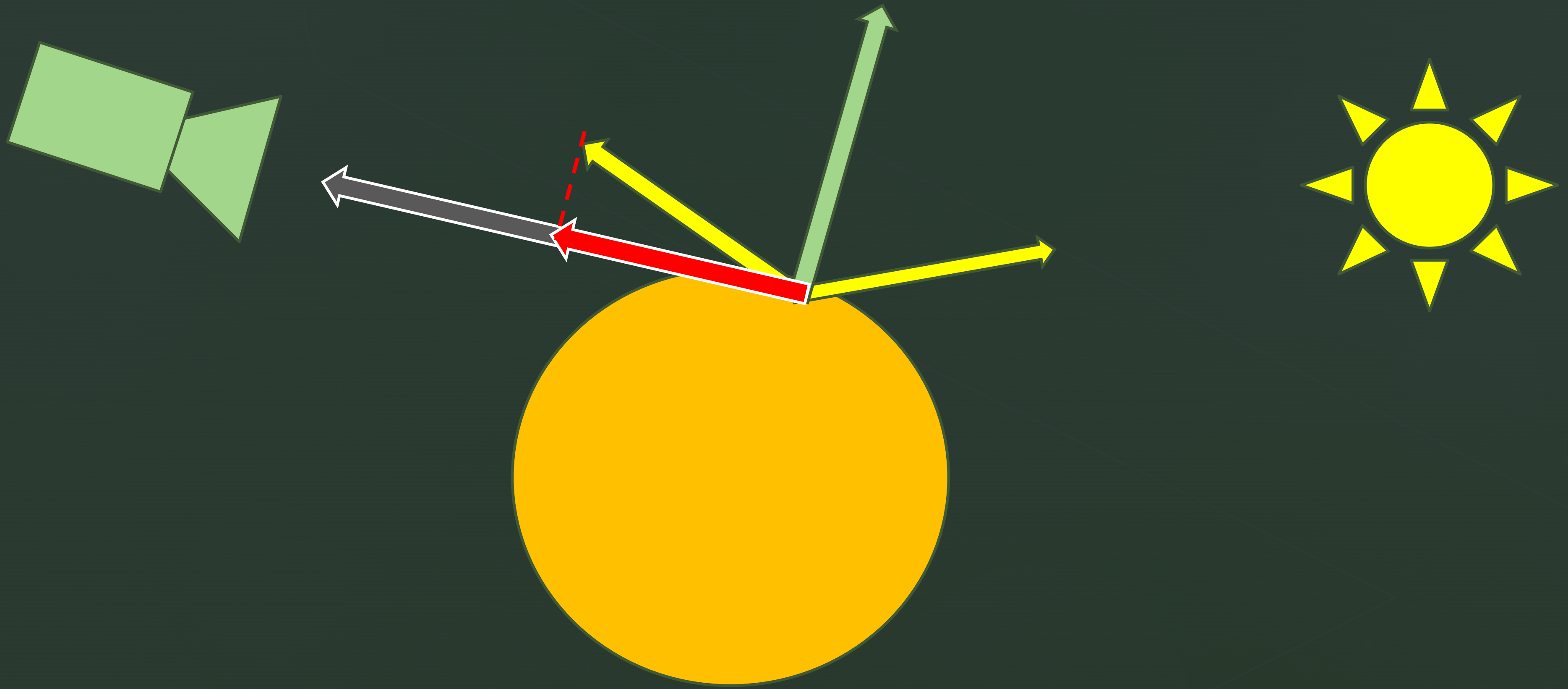
Use dot product to add light based on direction (projecting normal):

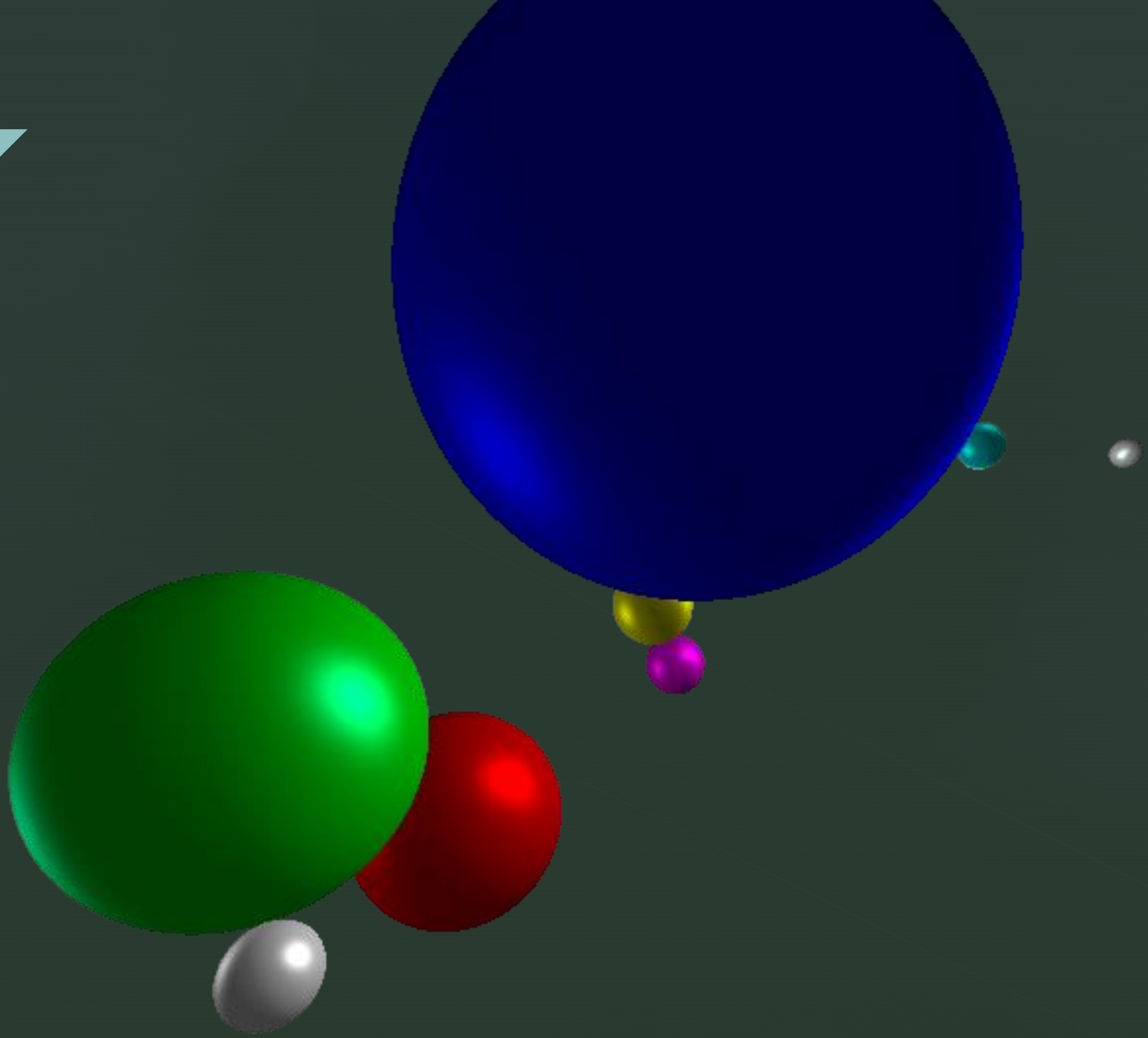


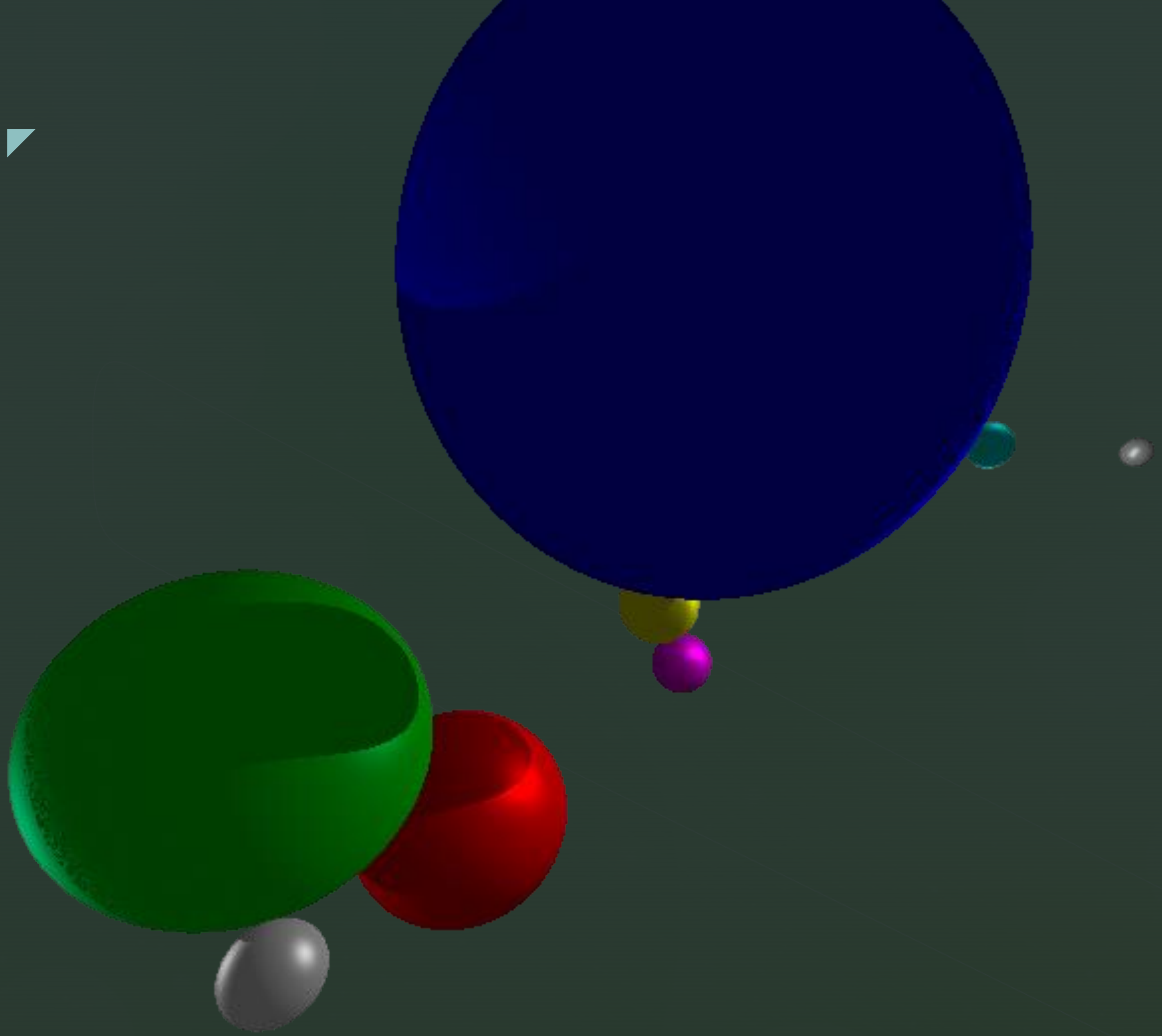


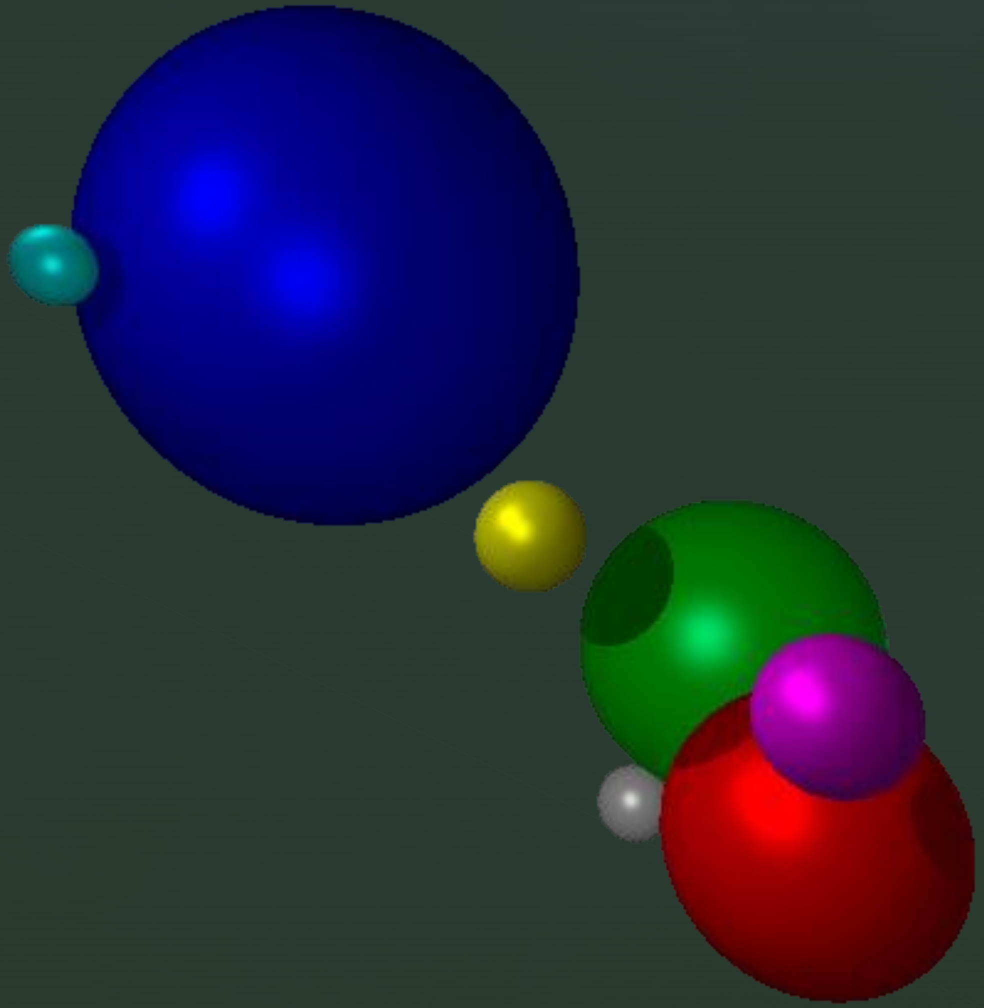
## Specular Lighting

Use reflection and then dot product on camera direction:







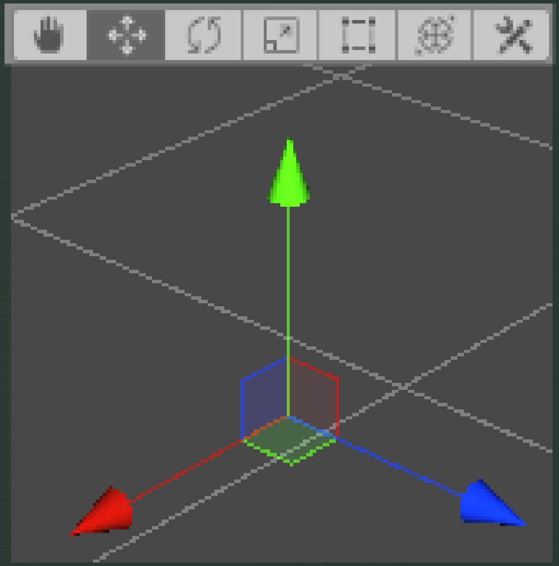


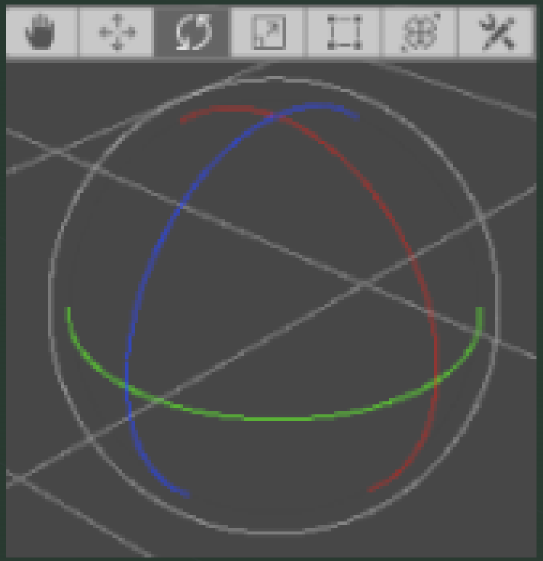
## Unity

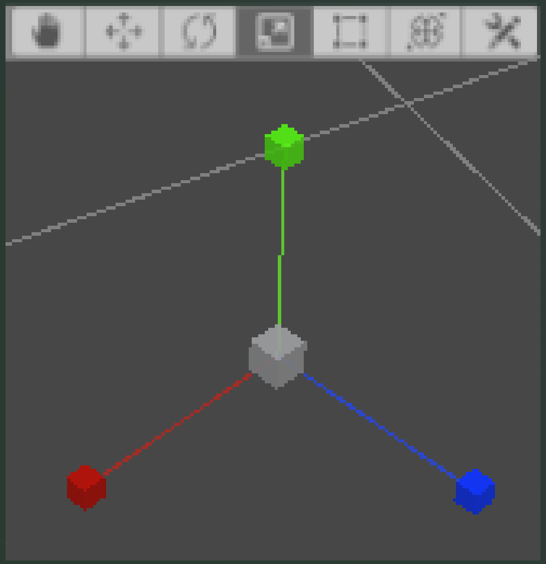
**What is Unity?**

- ➢ Unity is a powerful and popular cross-platform game engine developed by Unity Technologies.
- ➢ Used for creating 2D, 3D, augmented reality (AR), and virtual reality (VR) games and applications.
- ➢ Provides tools for design, development, and deployment across multiple platforms (Windows, Mac, iOS, Android, WebGL, etc.).

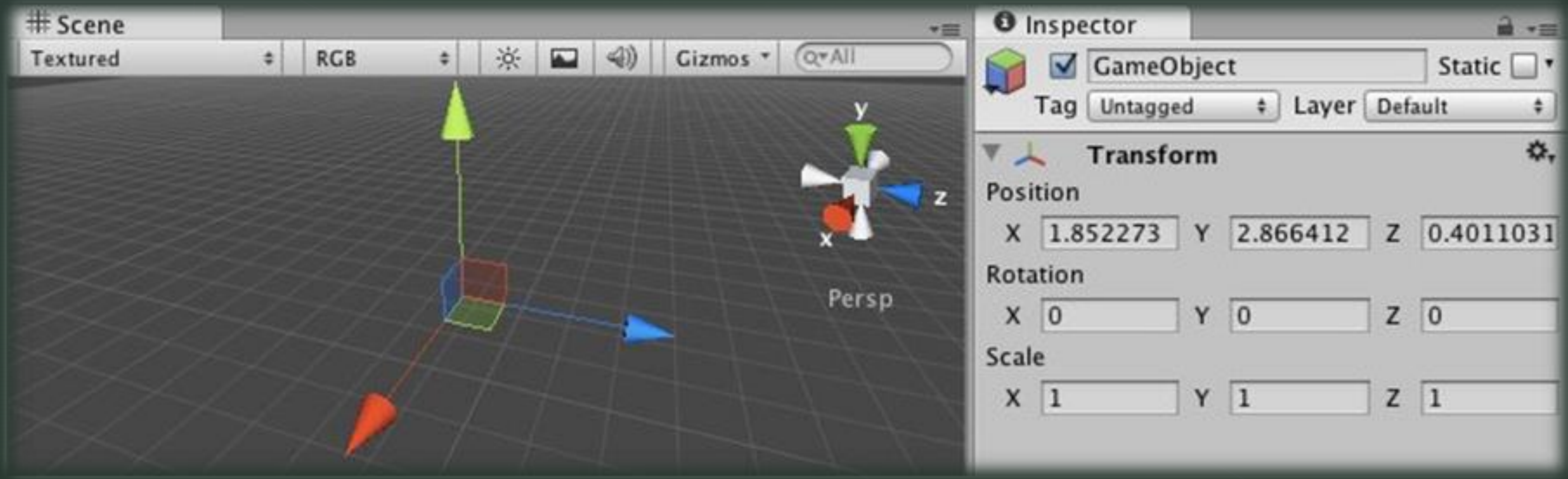
## Unity Components


Translate (W)


Rotate (E)


Scale (R)

## Unity Components



GameObject Static
Tag Untagged Layer Default

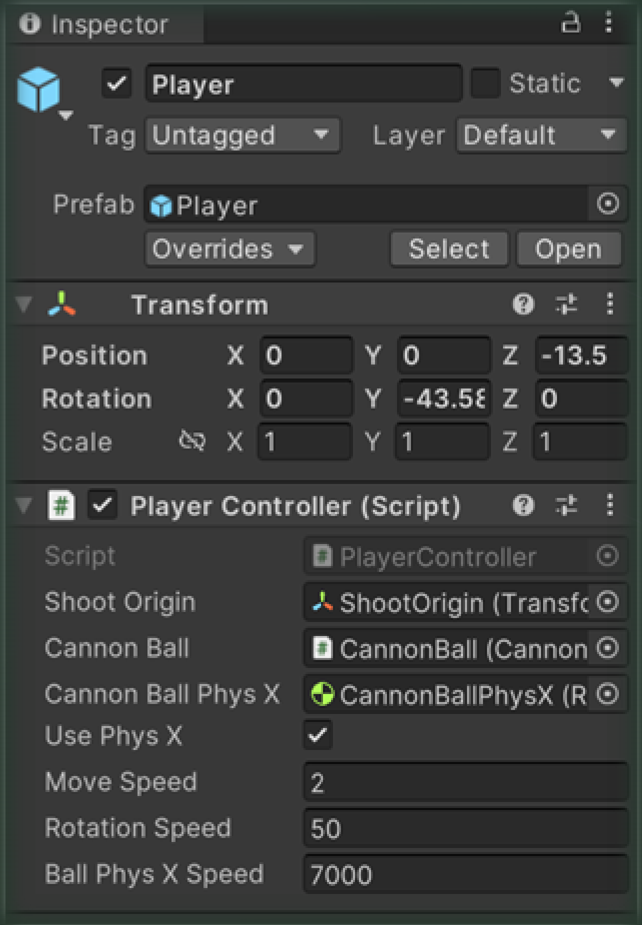
Transform

Position
X 1.852273 Y 2.866412 Z 0.4011031

Rotation
X 0 Y 0 Z 0

Scale
X 1 Y 1 Z 1

# Unity Components



Inspector

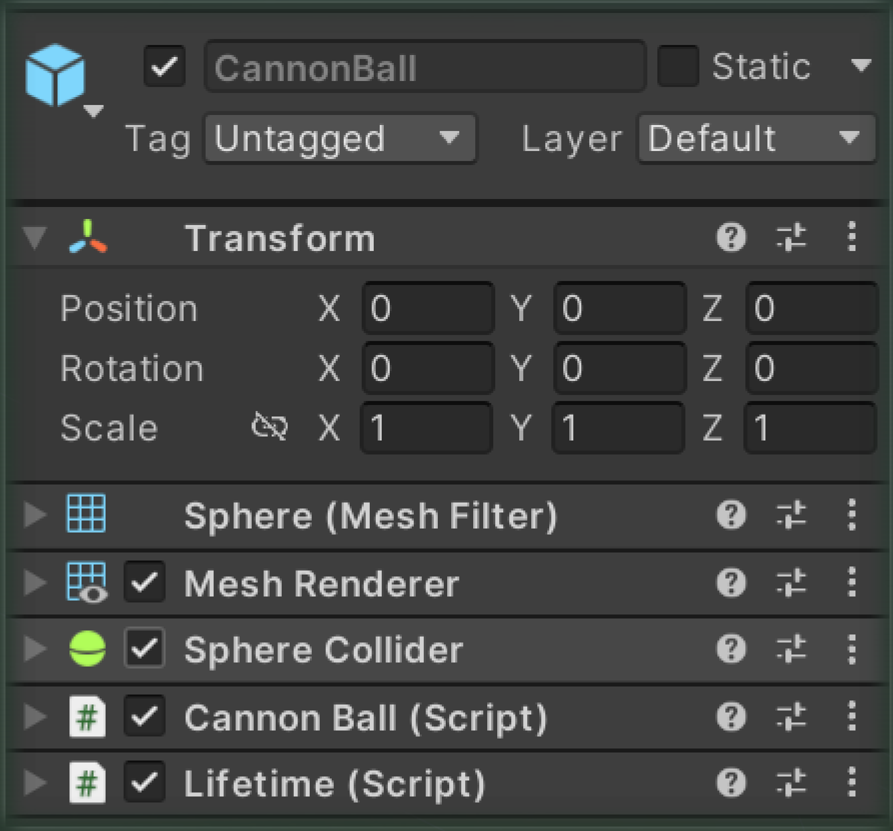
✓ Player | Static
Tag Untagged | Layer Default
Prefab Player
Overrides | Select | Open

Transform

| | | | | | | |
|---|---|---|---|---|---|---|
| Position | X | 0 | Y | 0 | Z | -13.5 |
| Rotation | X | 0 | Y | -43.5ε | Z | 0 |
| Scale | X | 1 | Y | 1 | Z | 1 |

✓ Player Controller (Script)

| | |
|---|---|
| Script | PlayerController |
| Shoot Origin | ShootOrigin (Transfc |
| Cannon Ball | CannonBall (Cannon |
| Cannon Ball Phys X | CannonBallPhysX (R |
| Use Phys X | ✓ |
| Move Speed | 2 |
| Rotation Speed | 50 |
| Ball Phys X Speed | 7000 |



✓ CannonBall | Static
Tag Untagged | Layer Default

Transform

| | | | | | | |
|---|---|---|---|---|---|---|
| Position | X | 0 | Y | 0 | Z | 0 |
| Rotation | X | 0 | Y | 0 | Z | 0 |
| Scale | X | 1 | Y | 1 | Z | 1 |

Sphere (Mesh Filter)
✓ Mesh Renderer
✓ Sphere Collider
✓ Cannon Ball (Script)
✓ Lifetime (Script)

## Start and Update Methods

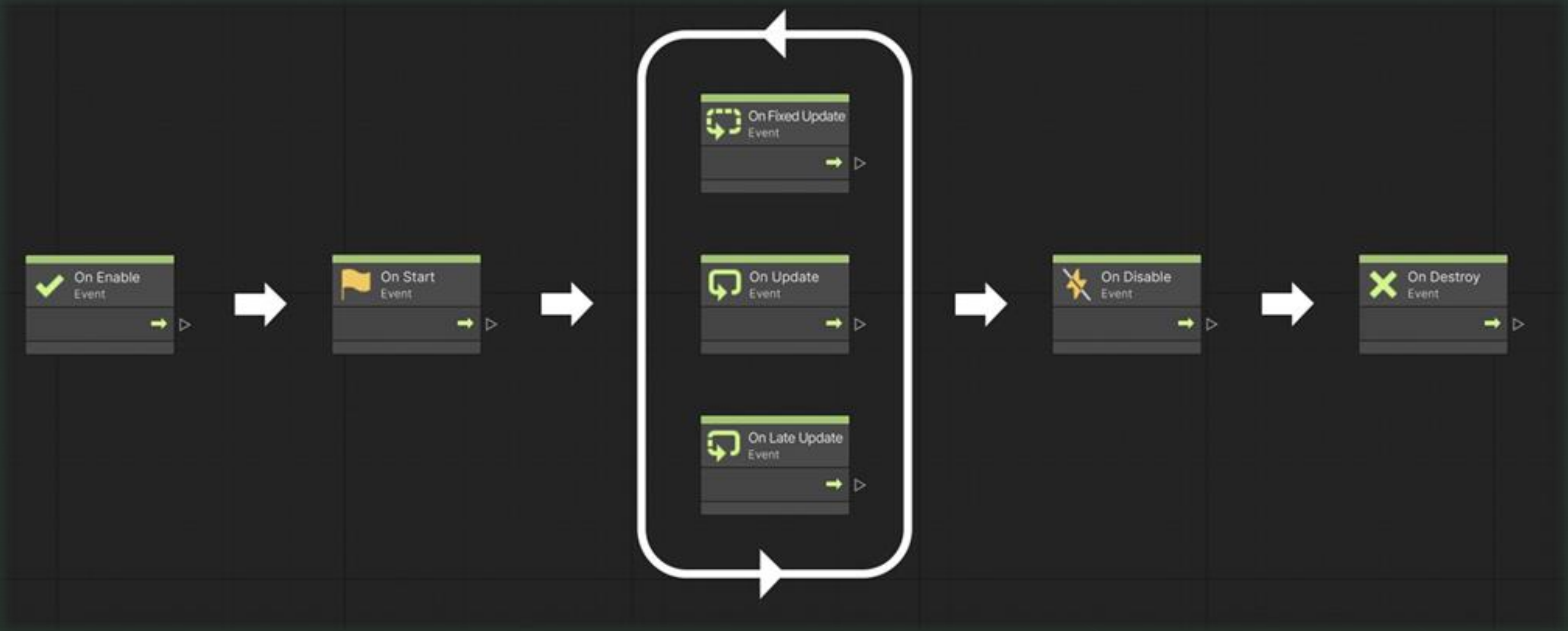
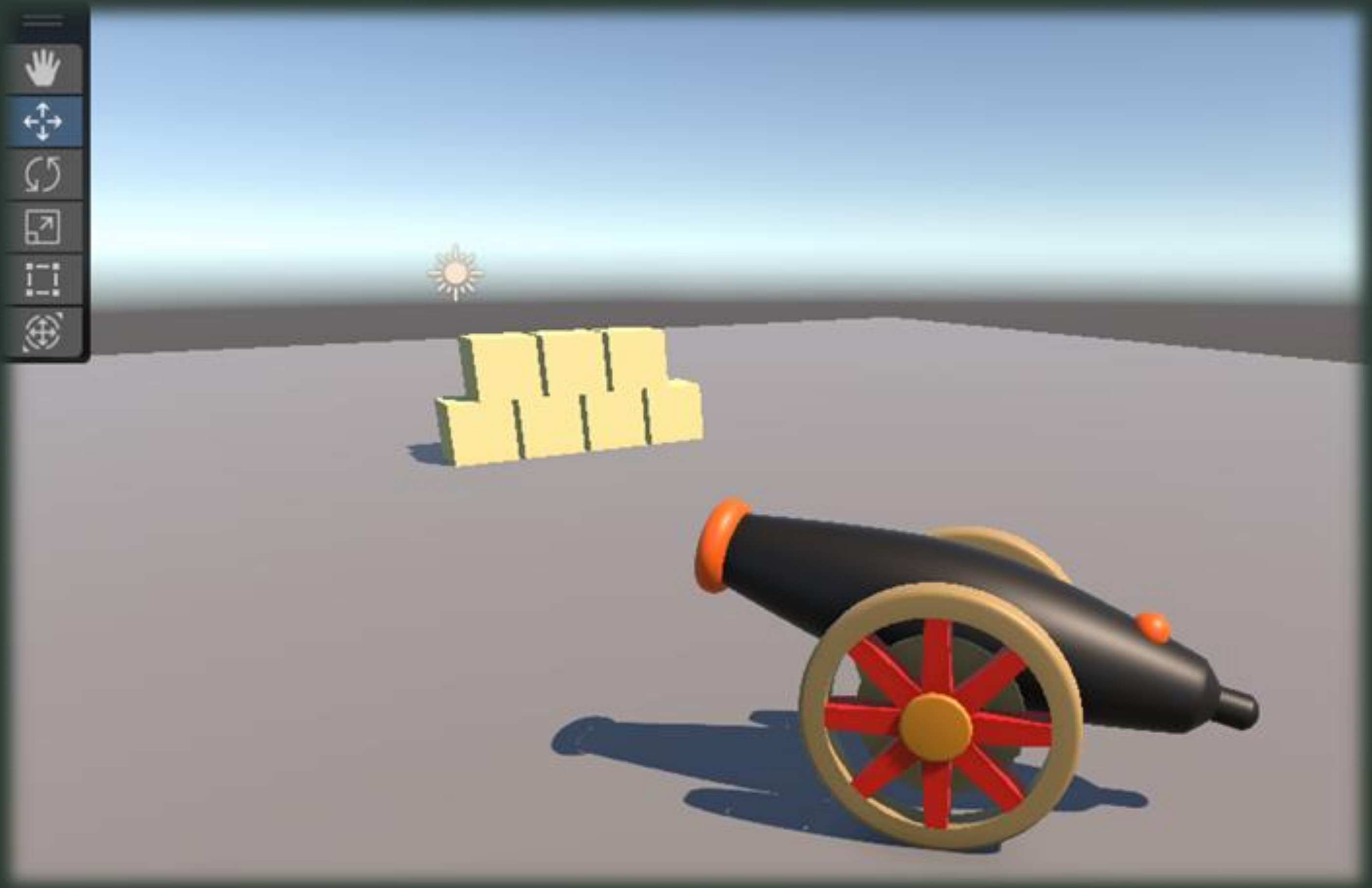


**Image Source:** notslot tutorial

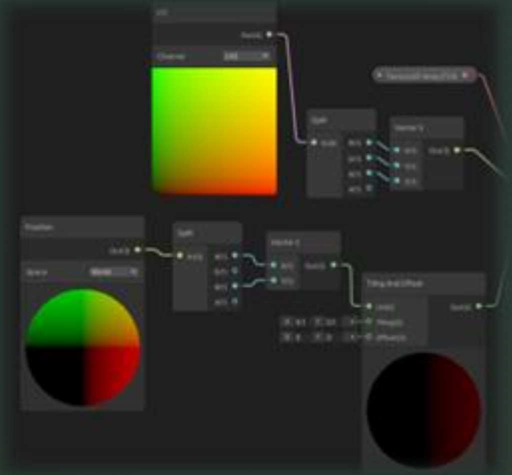
## Demo



## Shader Structure

**Shader**

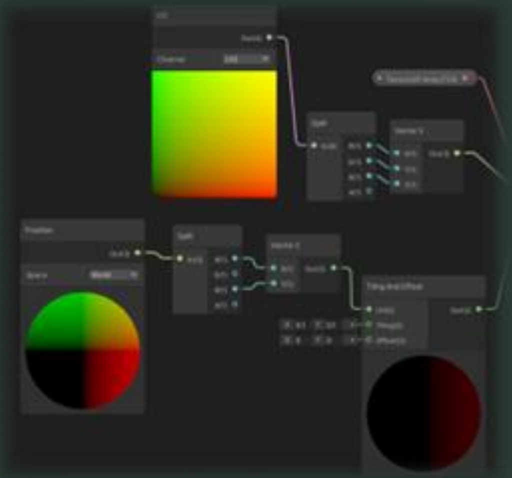
- Properties
  - Colours, Values, Textures
- SubShader
  - Pass
  - Pass
  - Pass
    - Vertex
    - Fragment

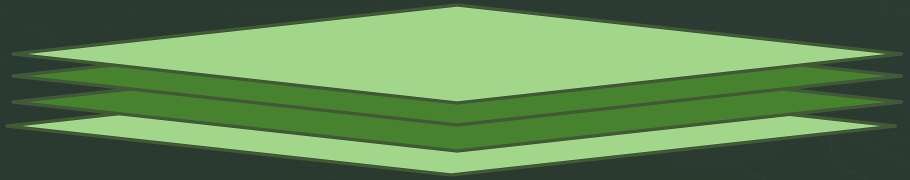


## Shader Structure

**Shader**

- Properties
  - Colours, Values, Textures
- SubShader
  - Pass
  - Pass
  - Pass
    - Vertex
    - Fragment

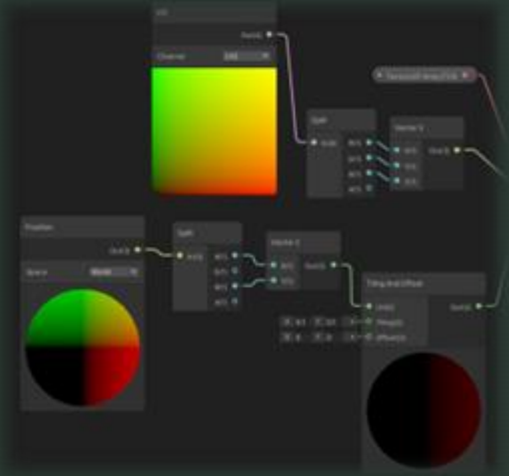


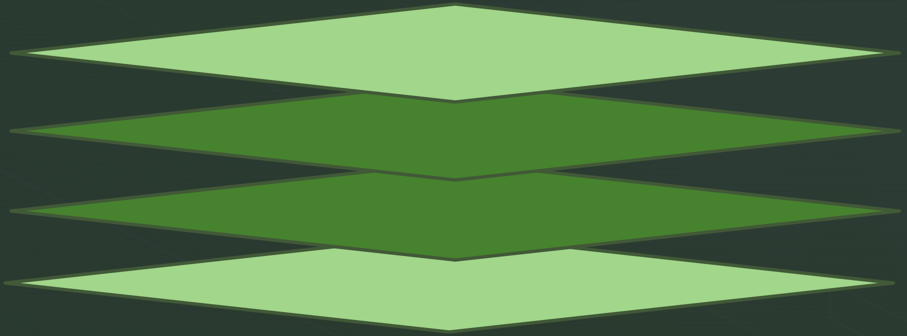


## Shader Structure

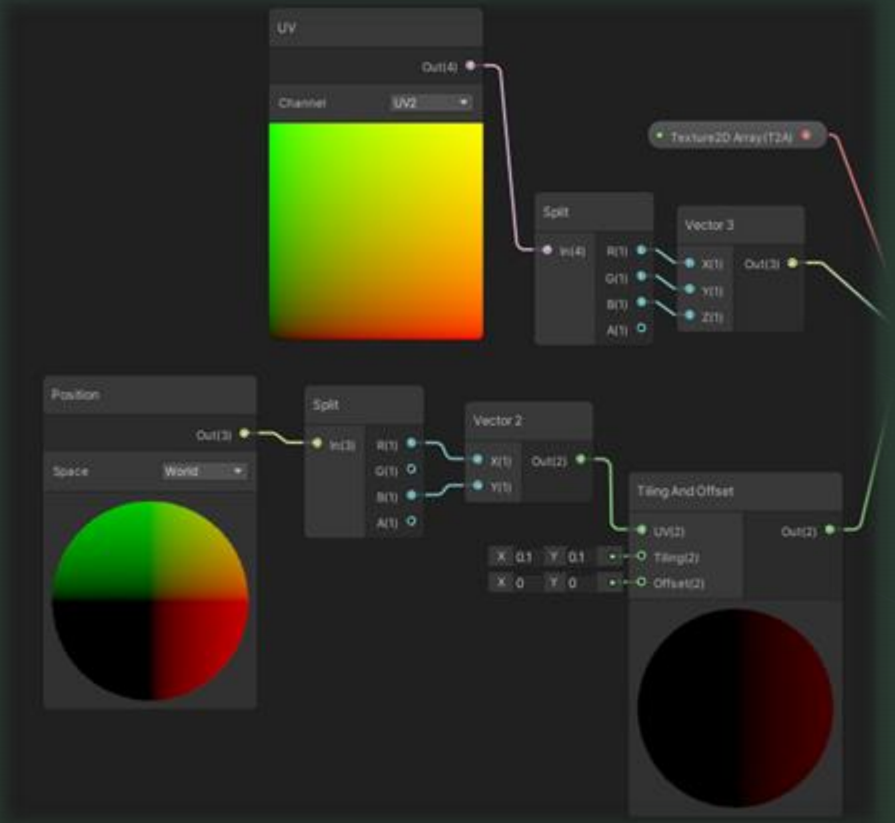
**Shader**

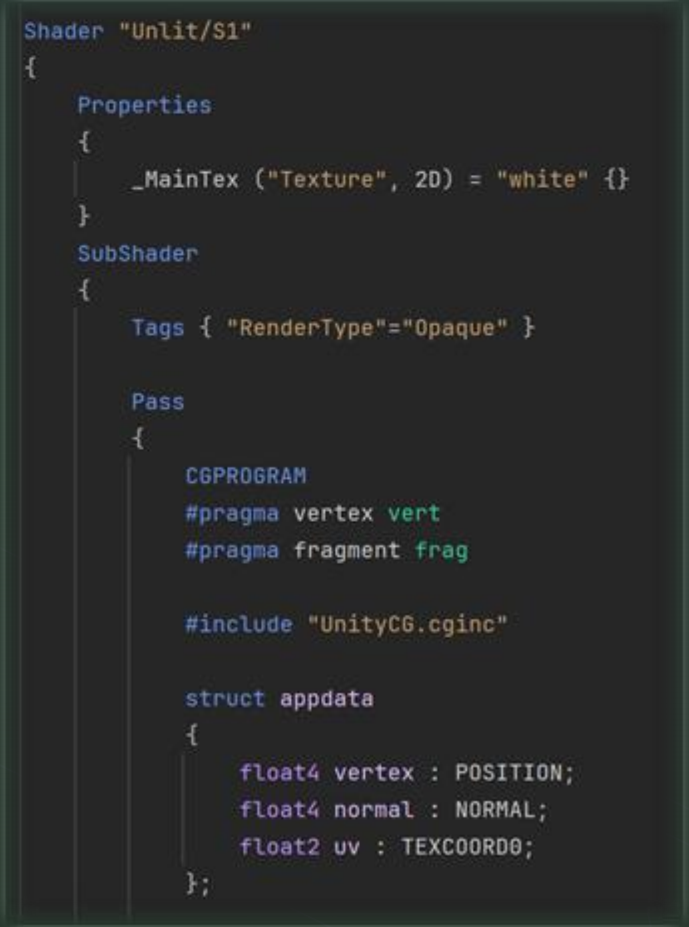
- ➢ Properties
  - Colours, Values, Textures
- ➢ SubShader
  - Pass
  - Pass
  - Pass
    - Vertex
    - Fragment





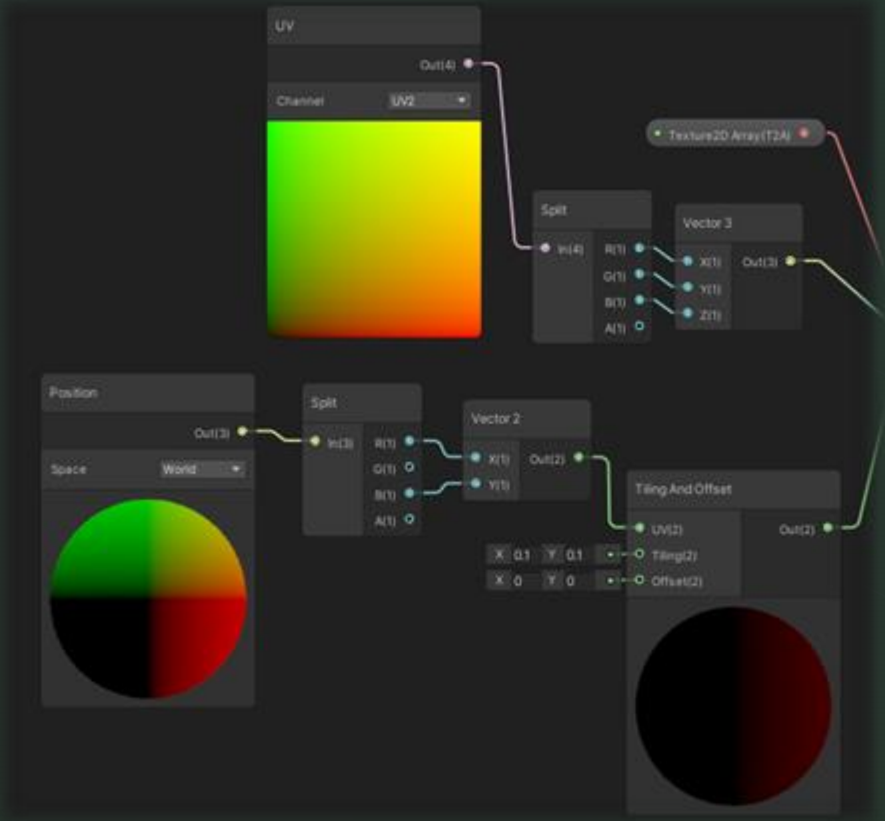
## Shader Implementation

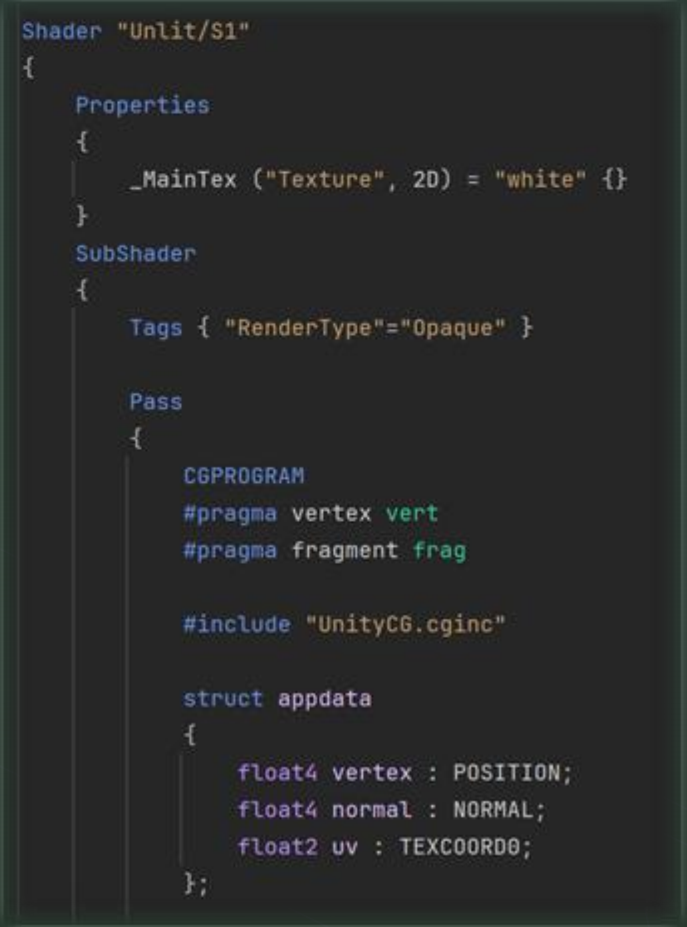




```
Shader "Unlit/S1"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag

            #include "UnityCG.cginc"

            struct appdata
            {
                float4 vertex : POSITION;
                float4 normal : NORMAL;
                float2 uv : TEXCOORD0;
            };
```

## Demo



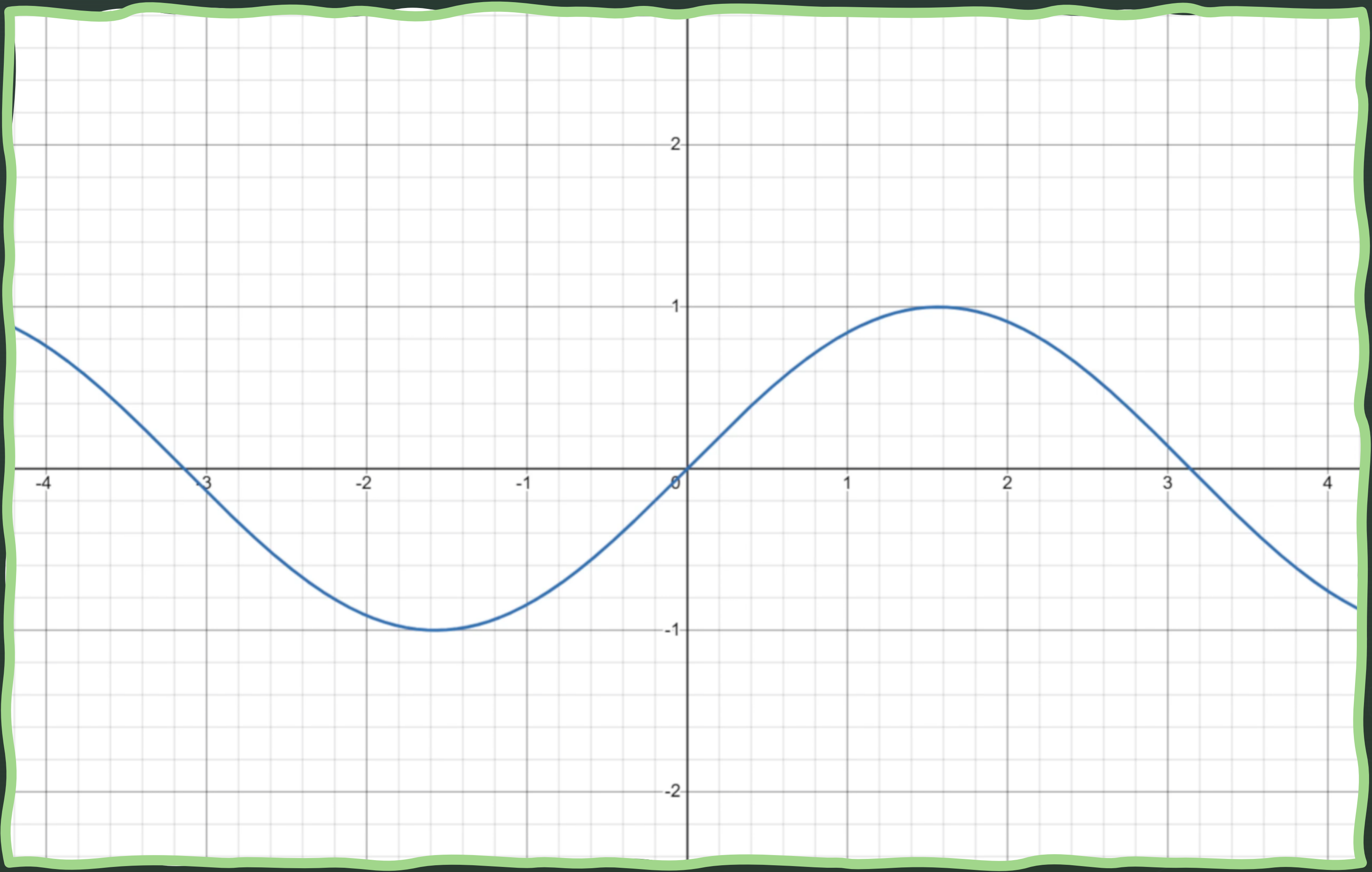


```
Shader "Unlit/S1"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }

        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag

            #include "UnityCG.cginc"

            struct appdata
            {
                float4 vertex : POSITION;
                float4 normal : NORMAL;
                float2 uv : TEXCOORD0;
            };
```

## Any Last Questions?

## Auxiliary Slides

## Sin(x * a)



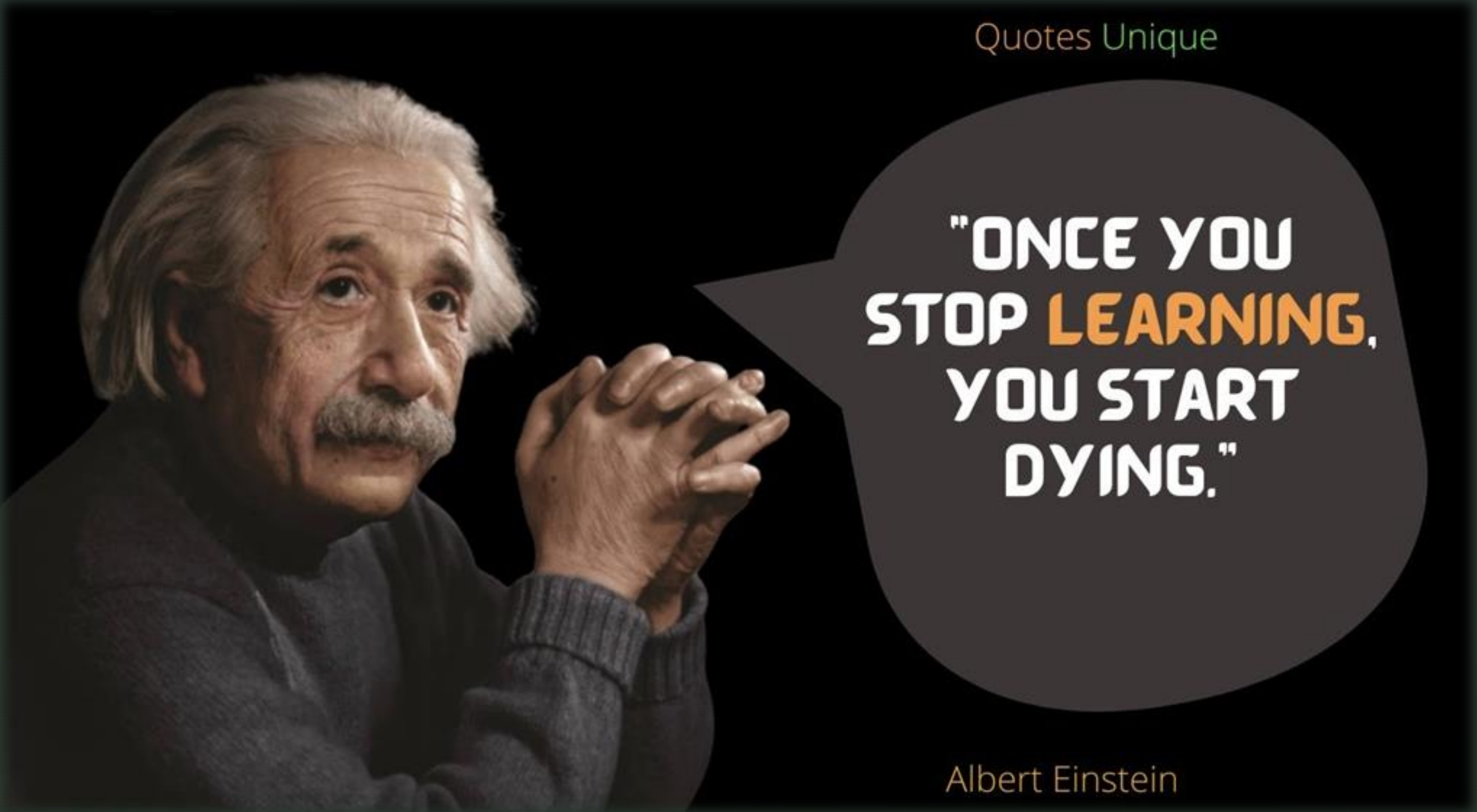


**Image Source:** Quotes Unique Youtube