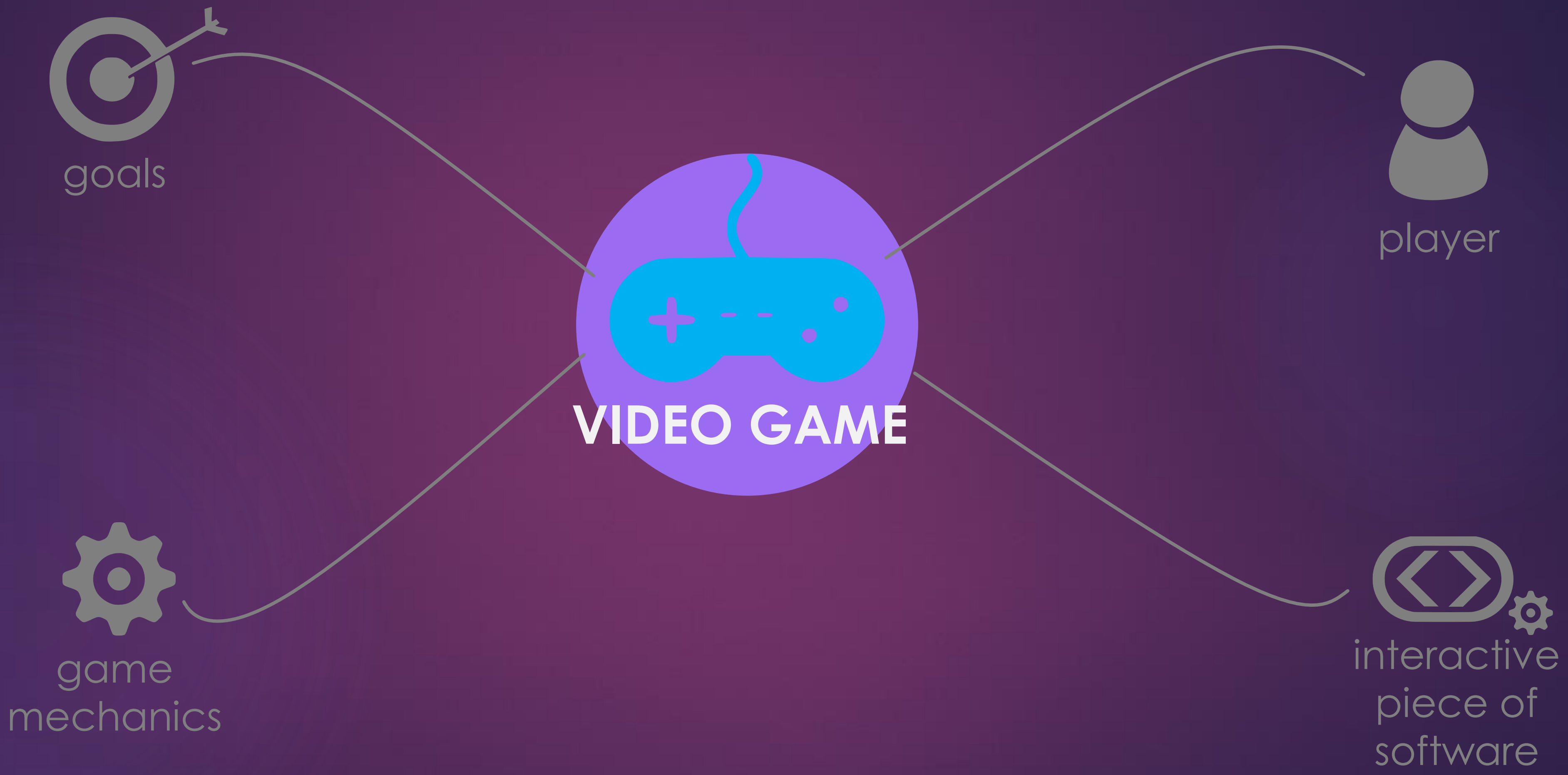
UNITY ECS

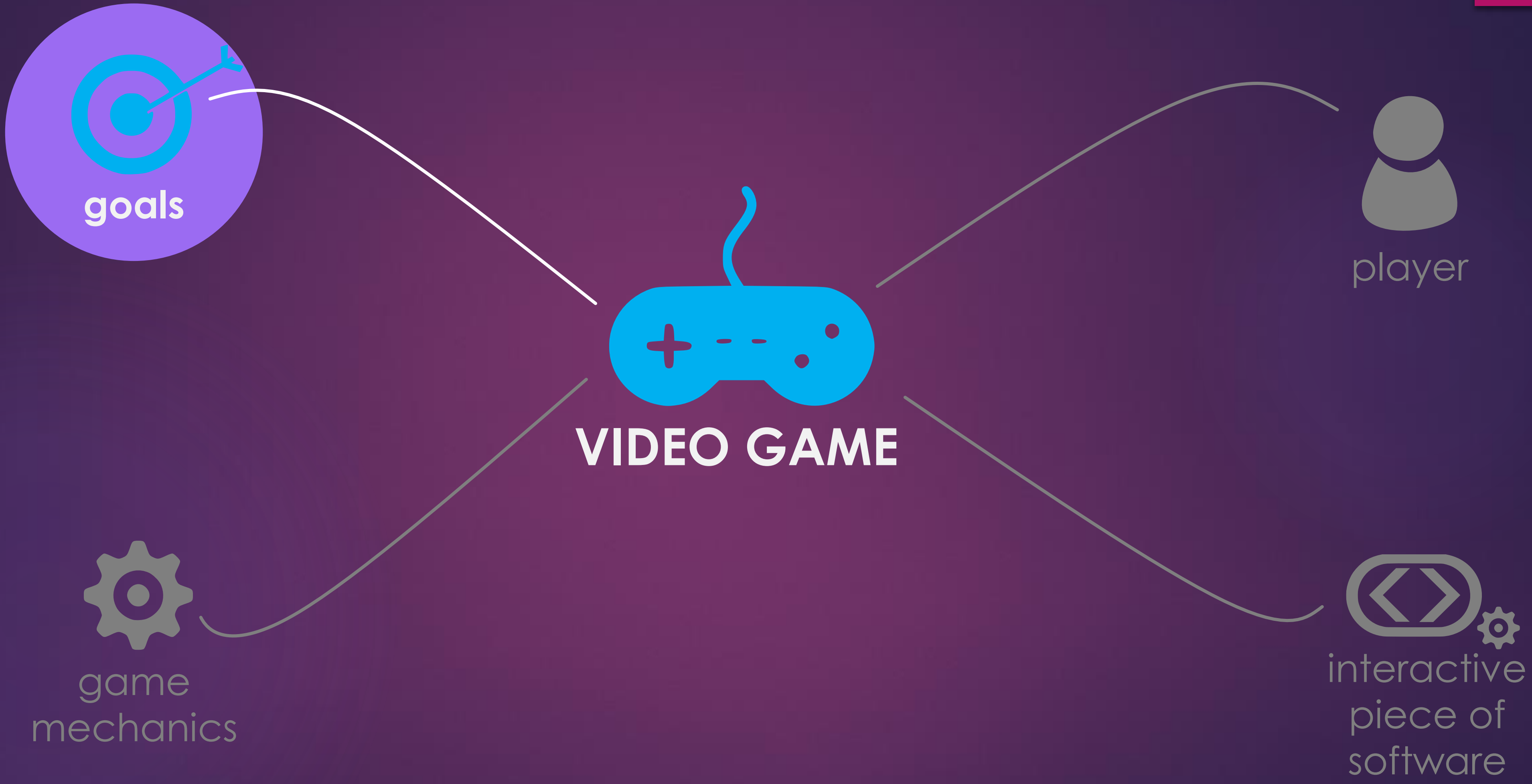
COMPUTER GRAPHICS SEMINAR

DANIEL NAEL

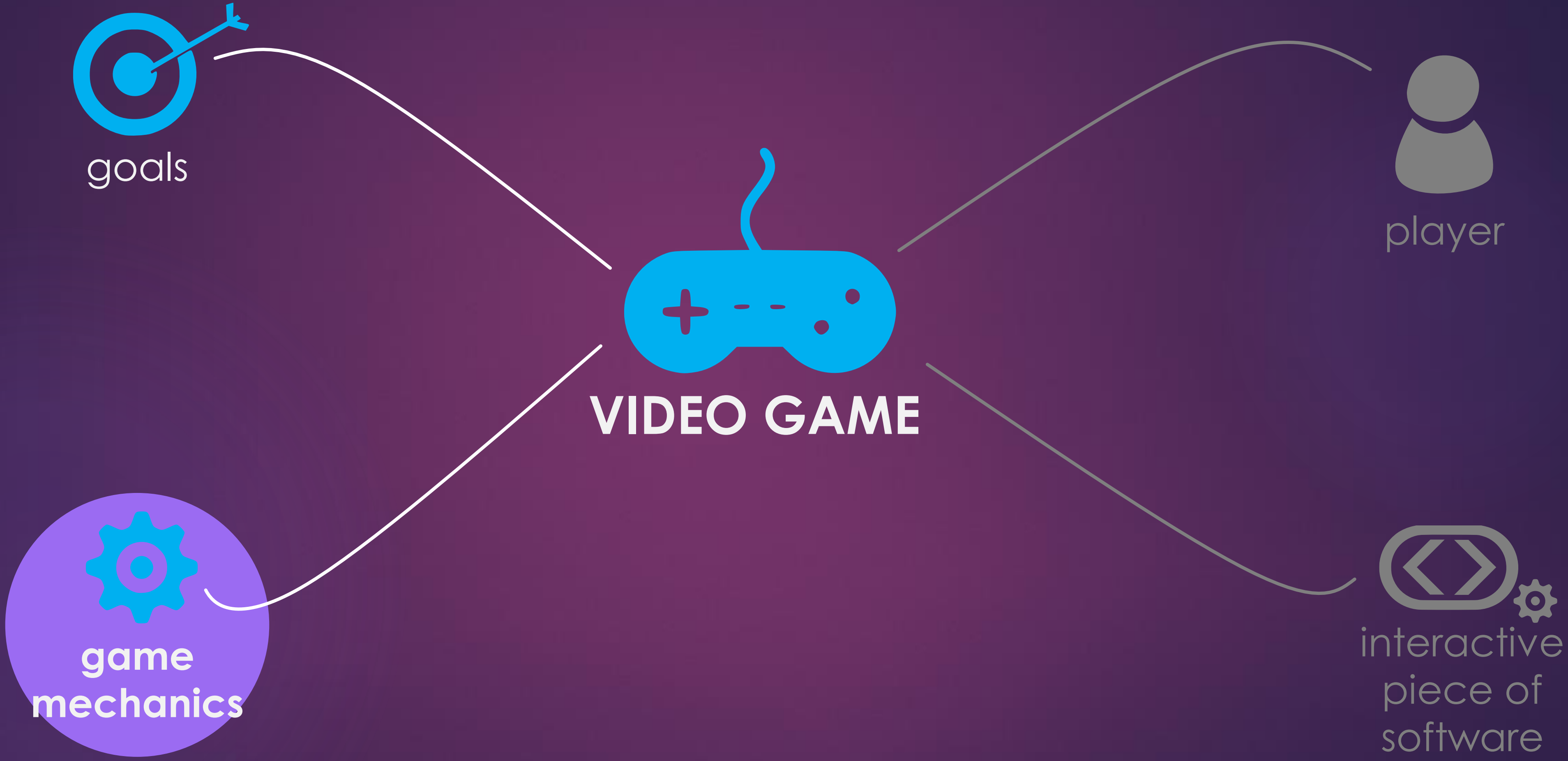
- video game


goals
player
VIDEO GAME
game mechanics
interactive piece of software

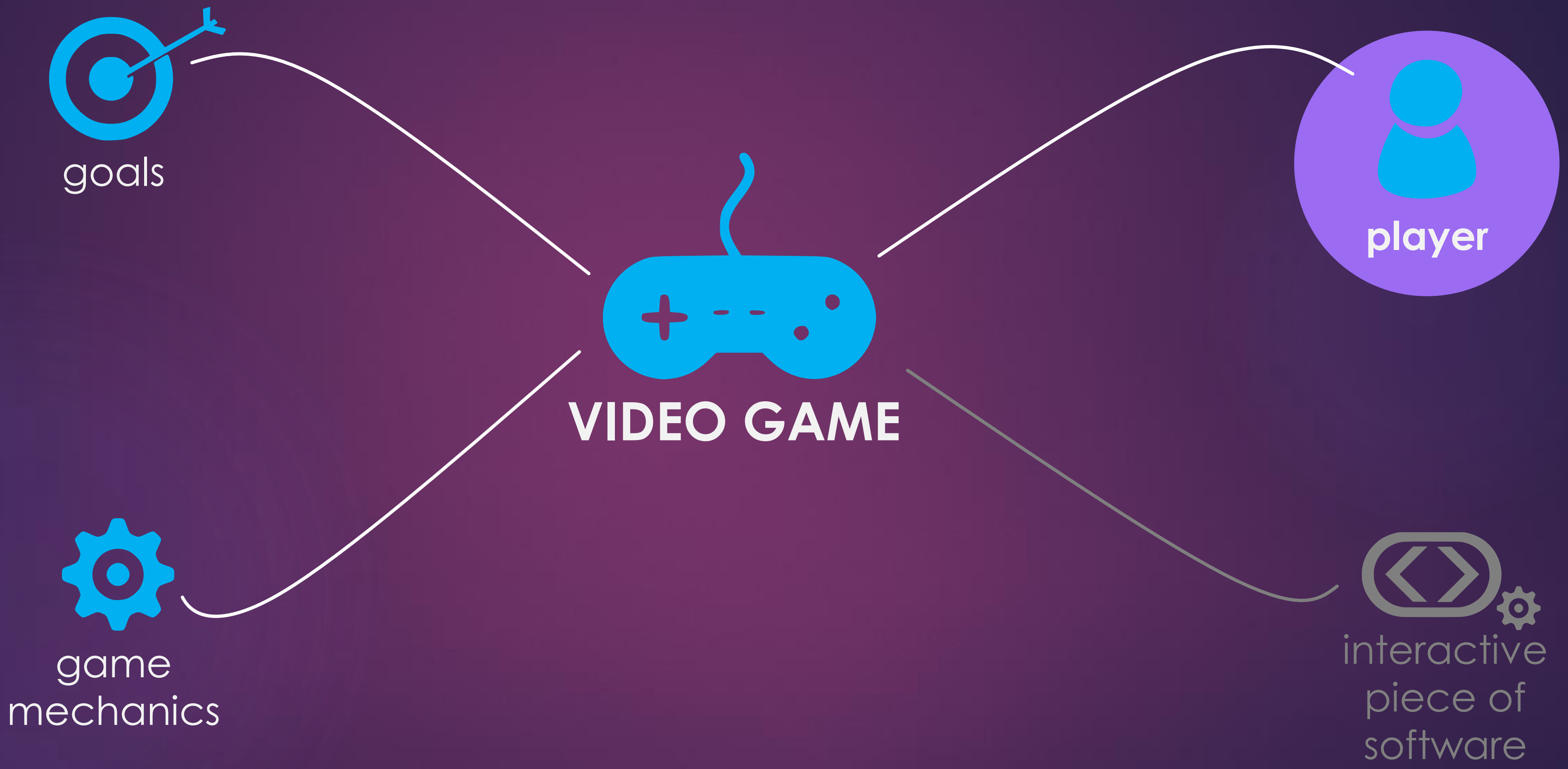
- video game


goals
VIDEO GAME
player
game mechanics
interactive piece of software

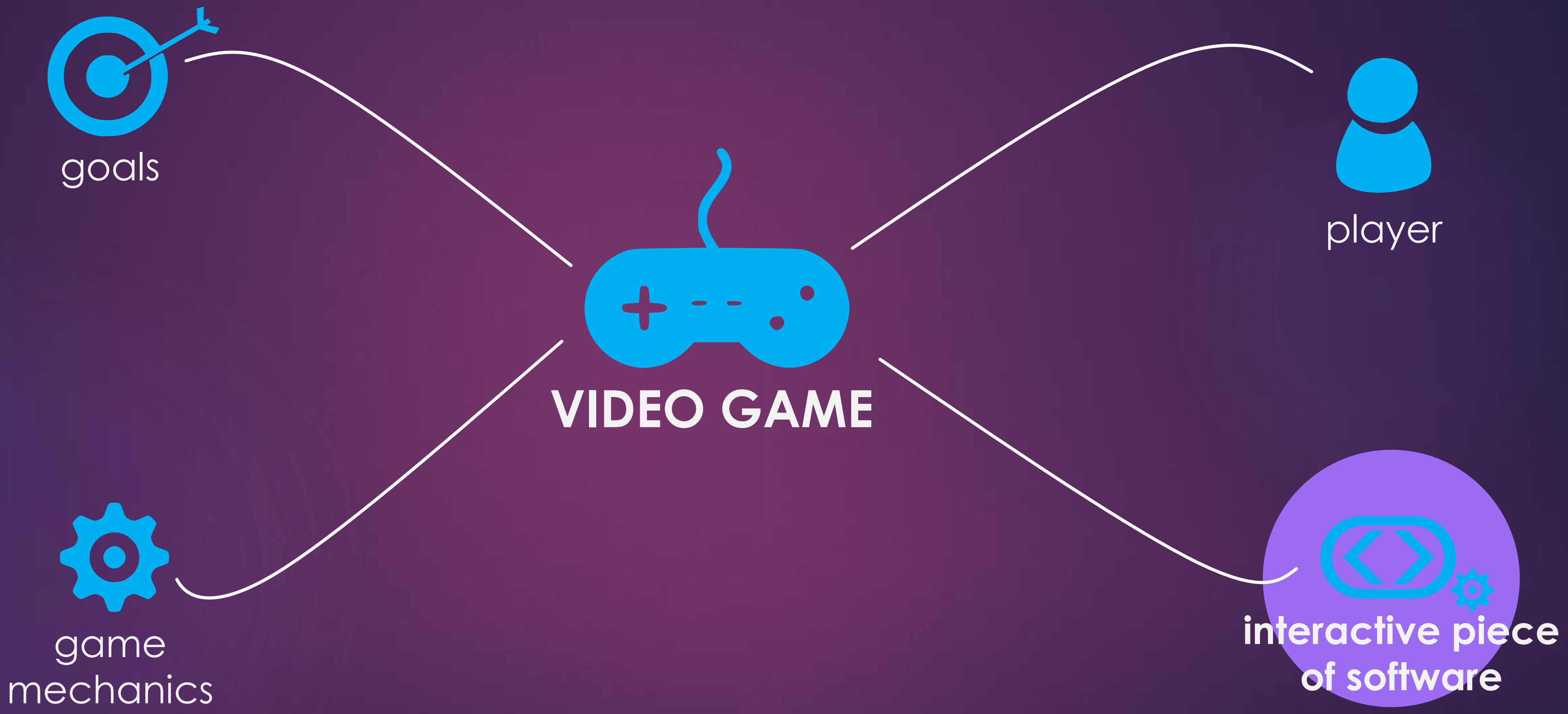
- video game


goals
VIDEO GAME
player
game mechanics
interactive piece of software

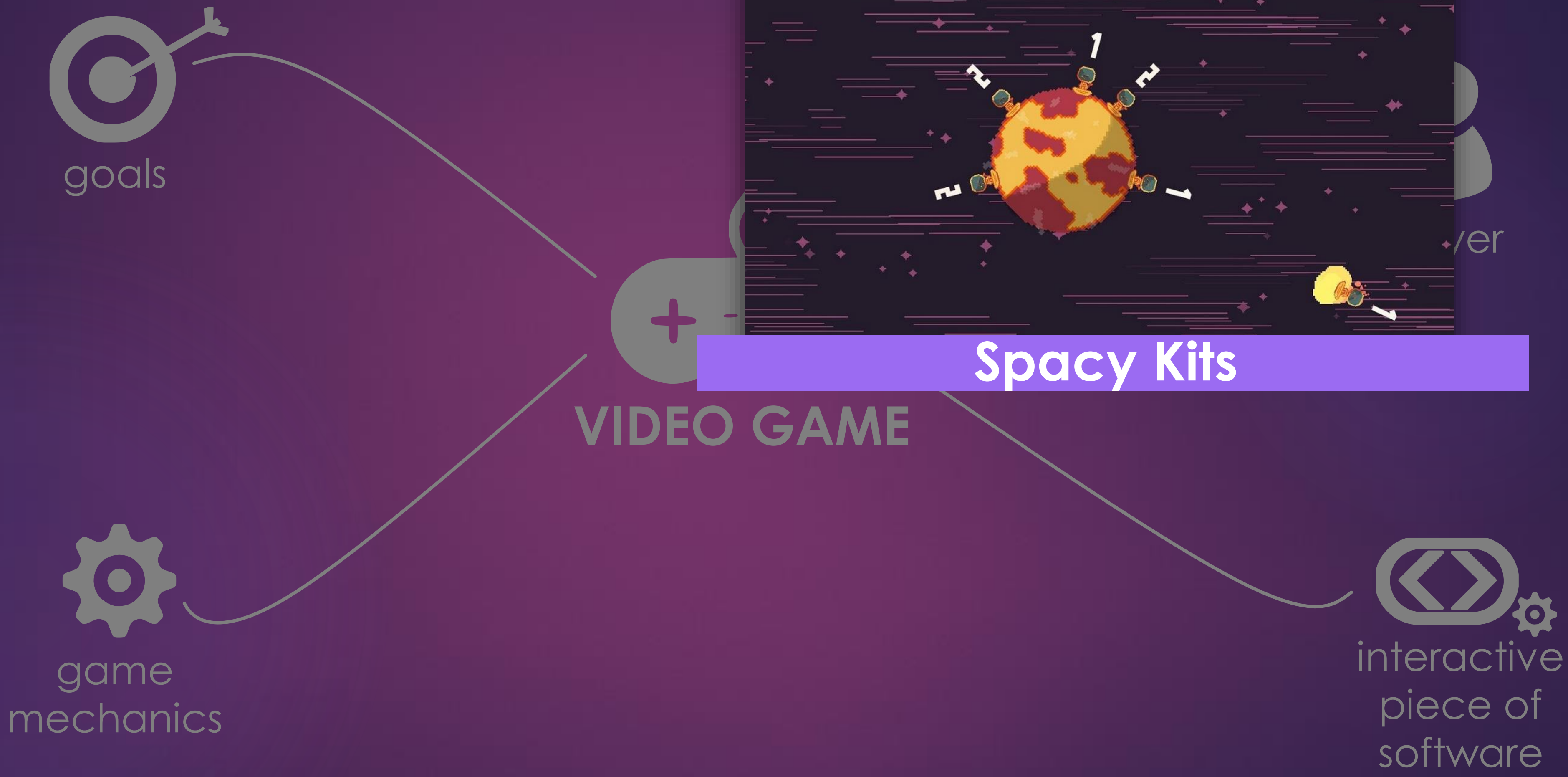
- video game


goals
game mechanics
VIDEO GAME
player
interactive piece of software

- video game


goals
player
VIDEO GAME
game mechanics
interactive piece of software

- video game



Spacy Kits

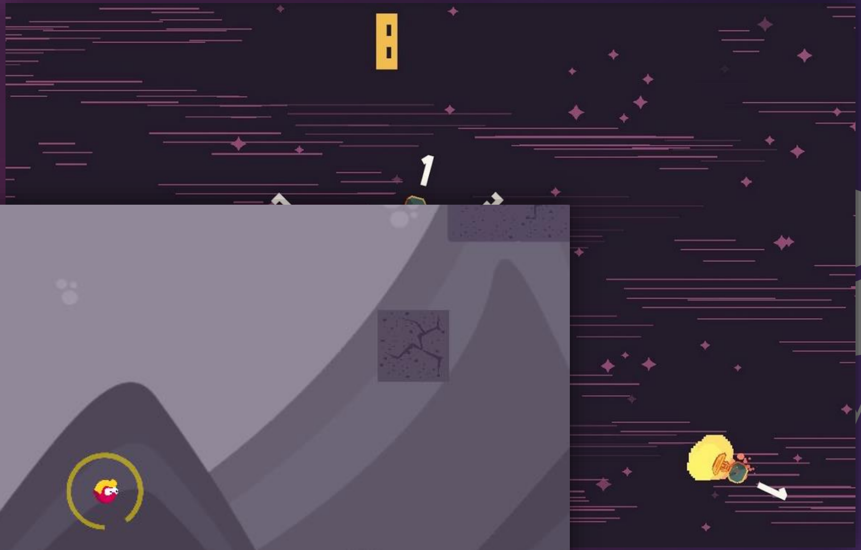
- video game

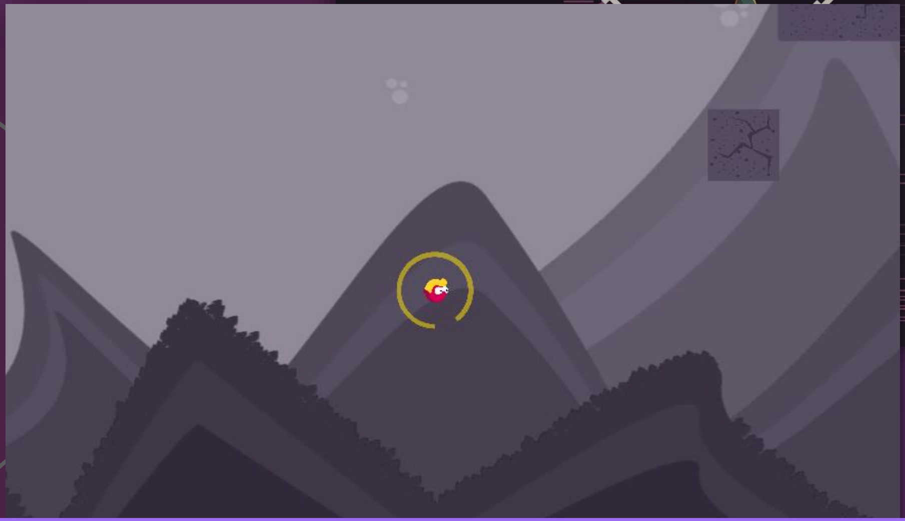
goals

game mechanics

interactive piece of software

/er

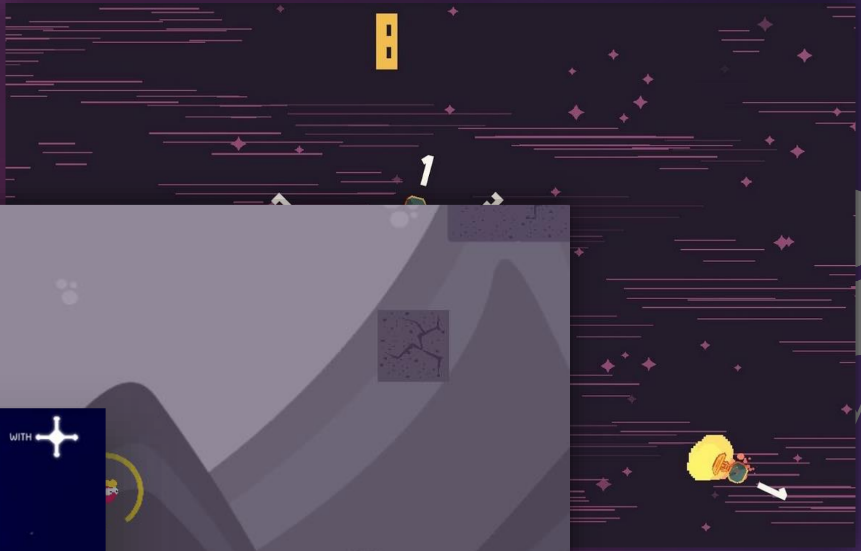


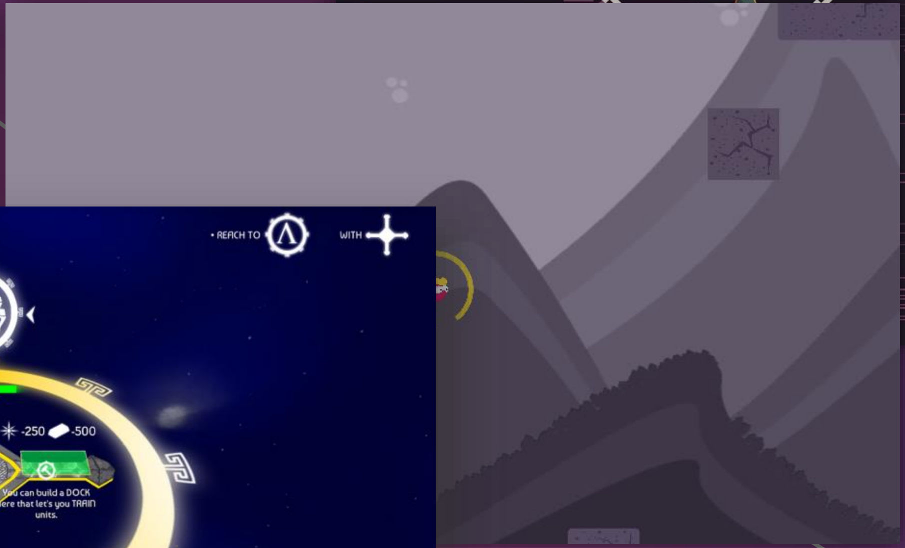


Slingventure

- video game

goals





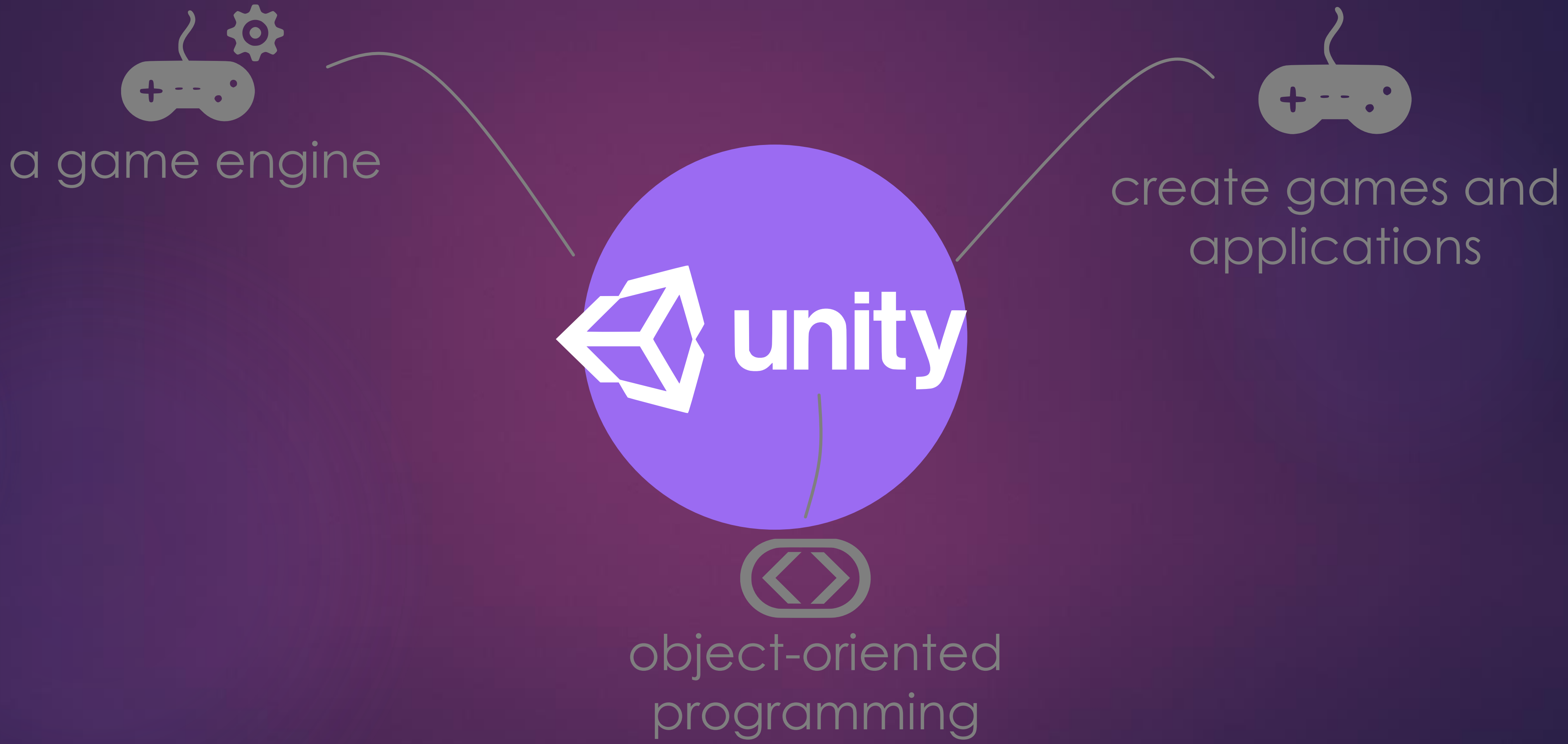
Icarus

m

er

interactive piece of software

- unity


a game engine
unity
create games and applications
object-oriented programming

- unity

a game engine

unity

create games and applications

object-oriented programming

- unity

a game engine

create games and applications

unity

object-oriented programming

- unity

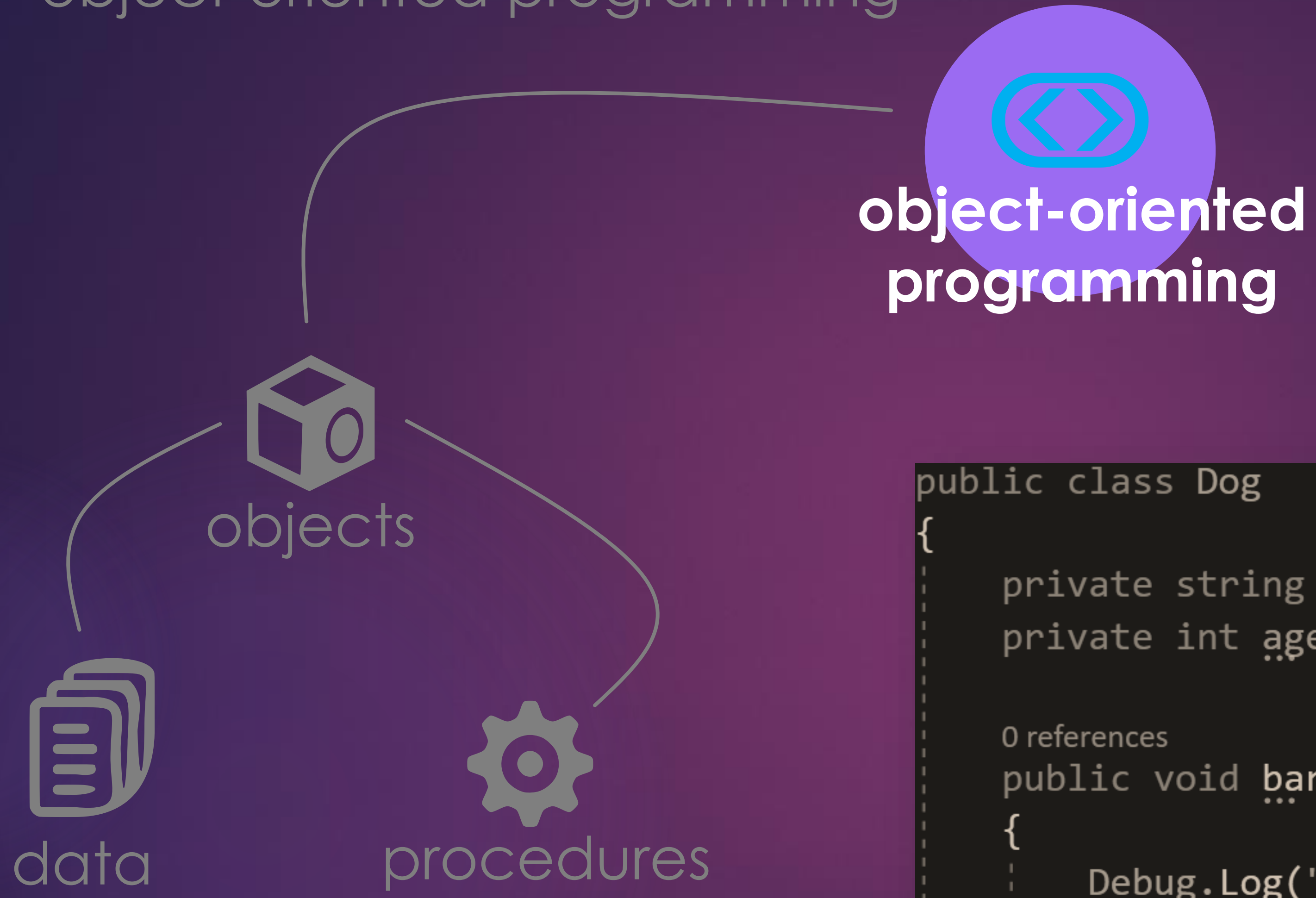
a game engine

create games and applications

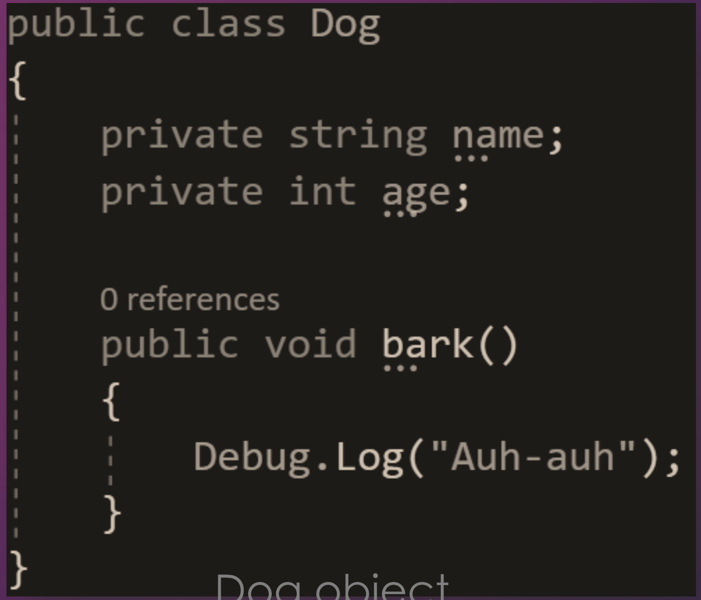
unity

object-oriented programming

- object-oriented programming

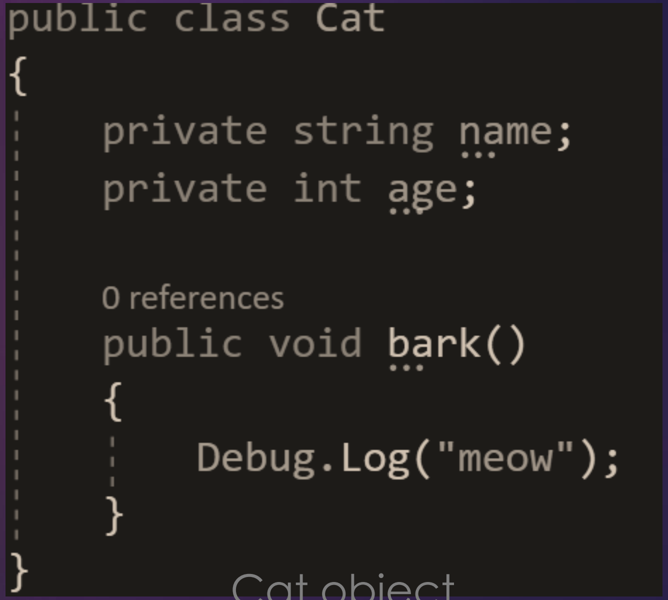


Examples



```
public class Dog
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("Auh-auh");
    }
}
```

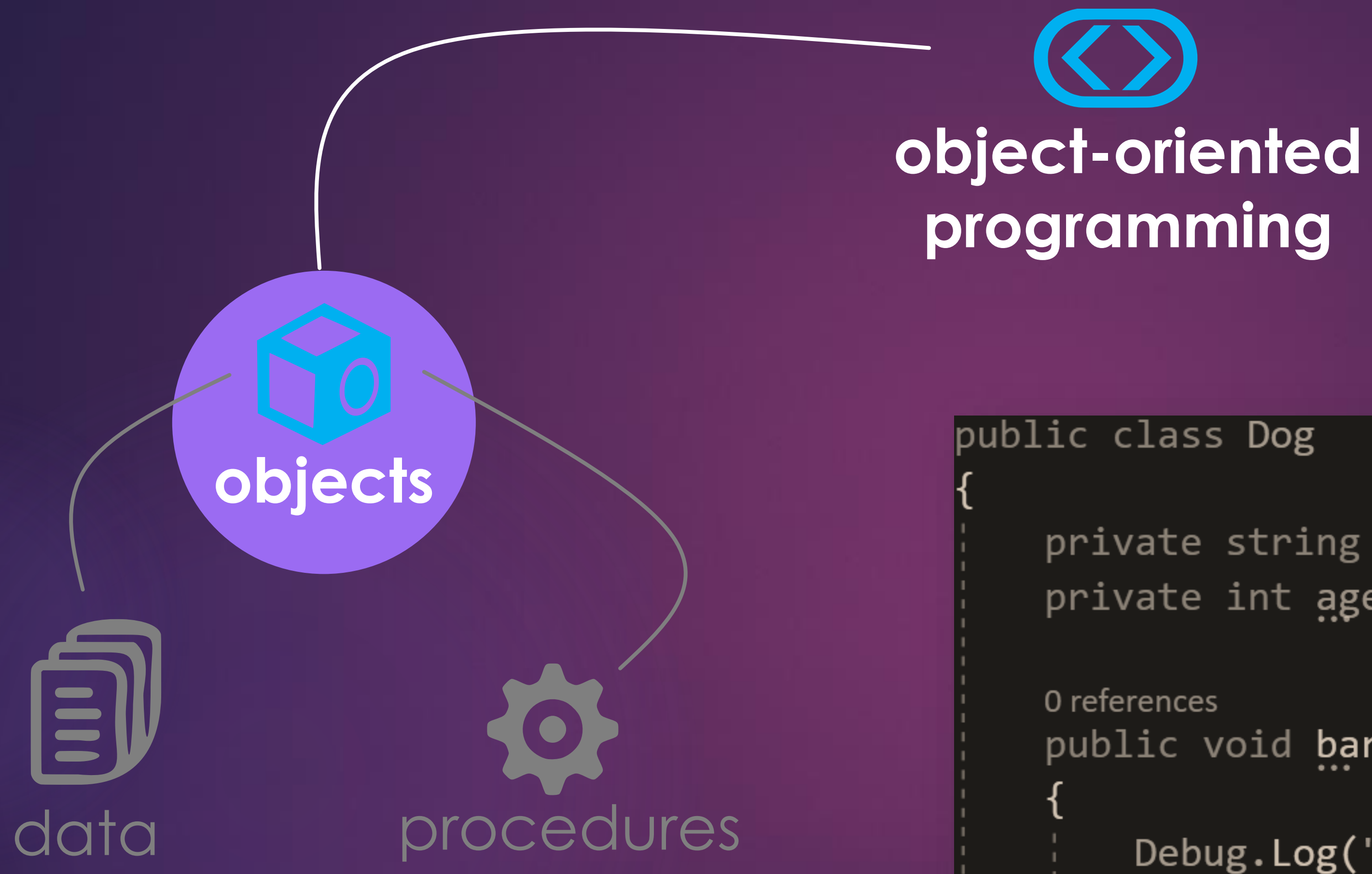
Dog object



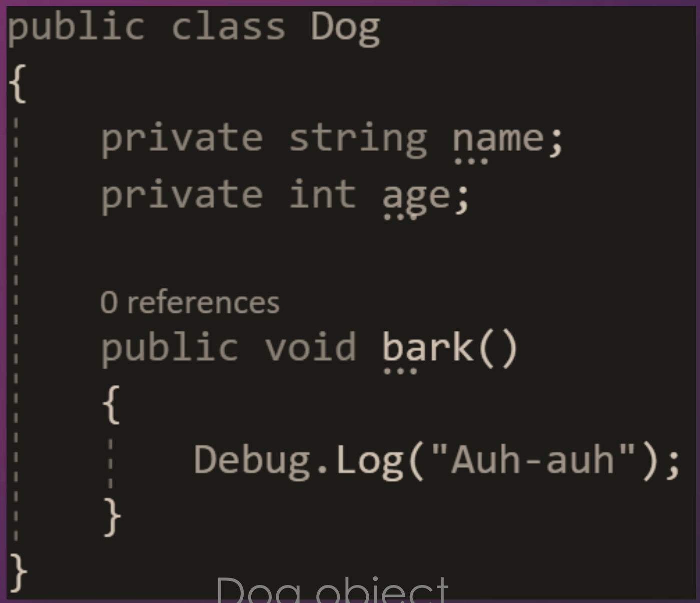
```
public class Cat
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("meow");
    }
}
```

Cat object

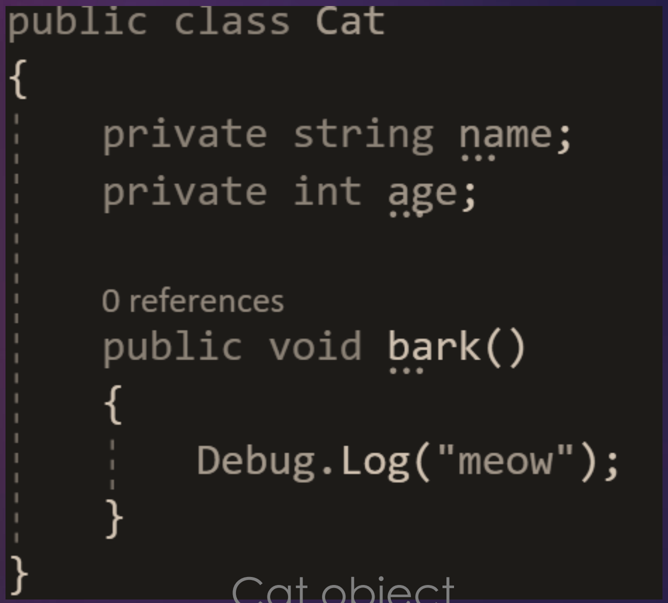
- object-oriented programming



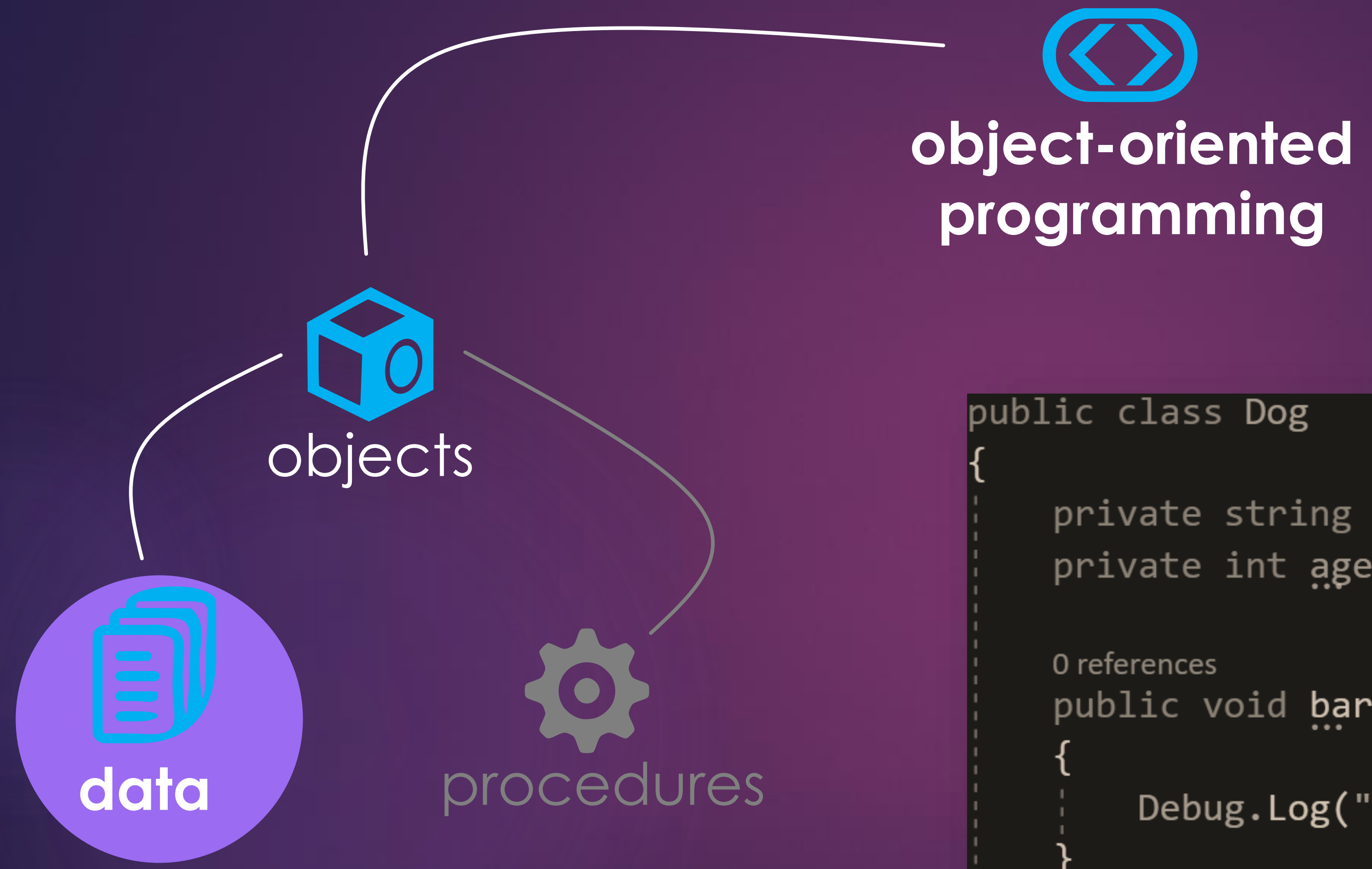
Examples


```
public class Dog
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("Auh-auh");
    }
}
```

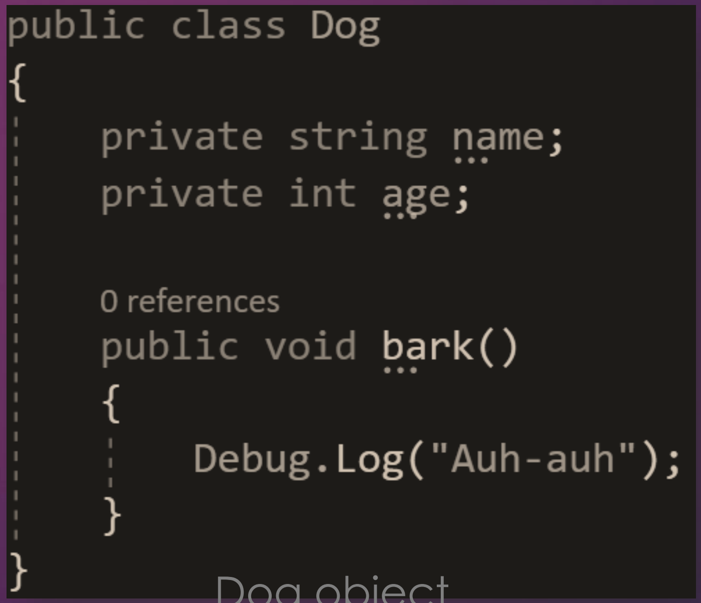
Dog object


```
public class Cat
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("meow");
    }
}
```

Cat object

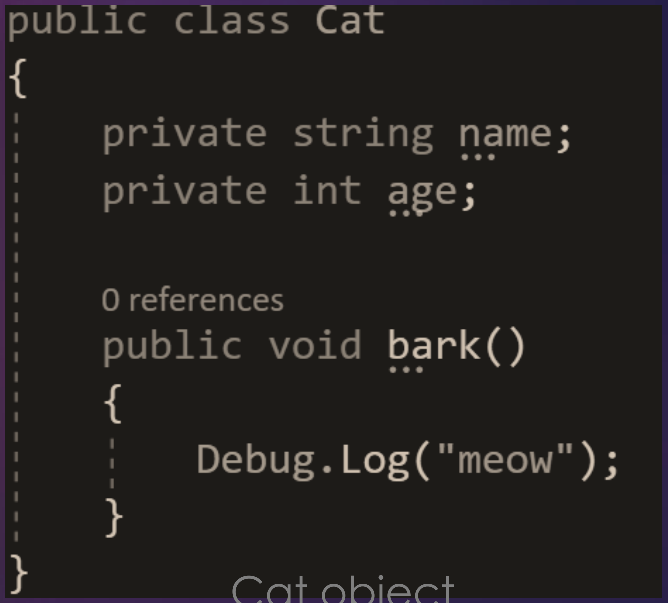


Examples



```
public class Dog
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("Auh-auh");
    }
}
```

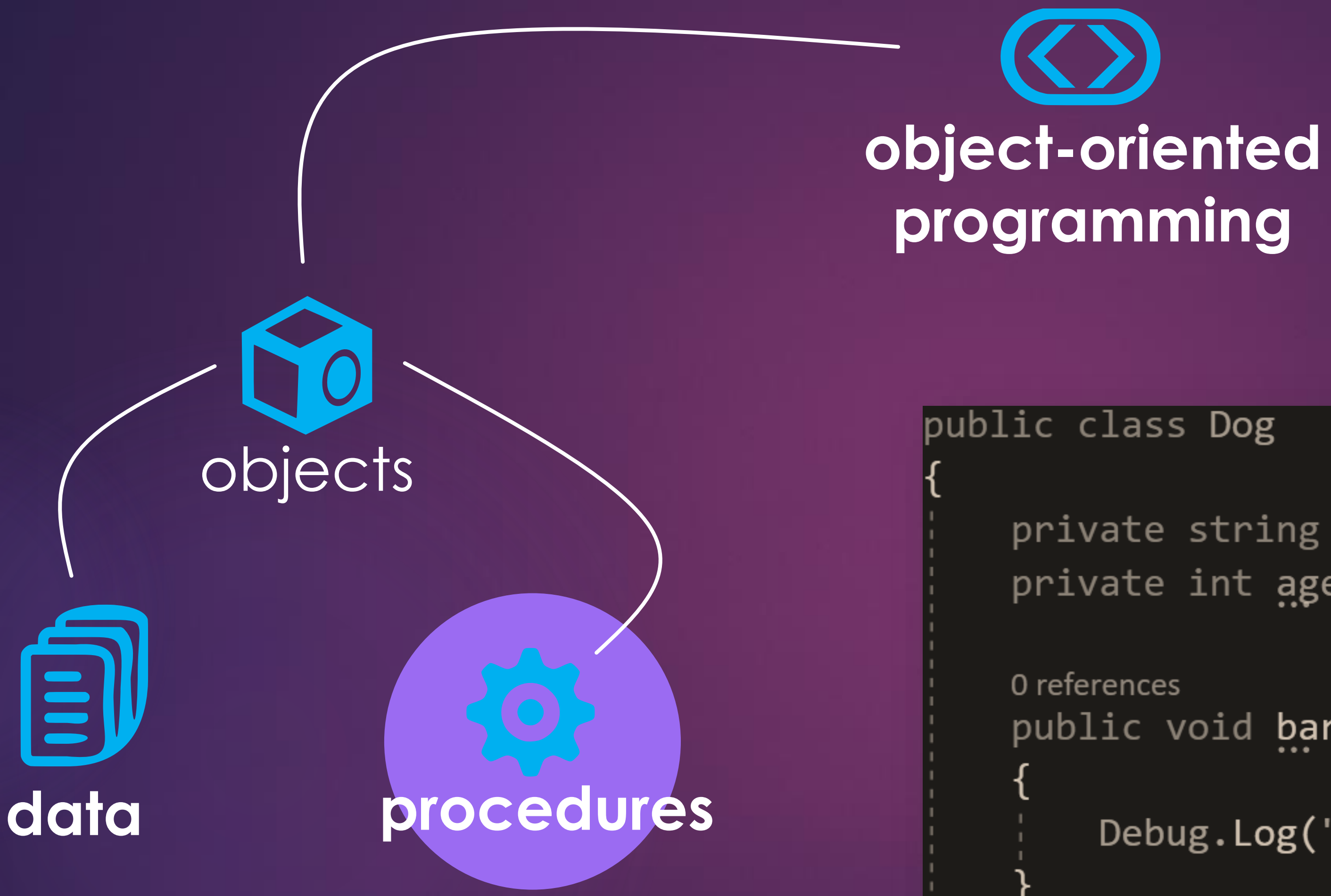
Dog object



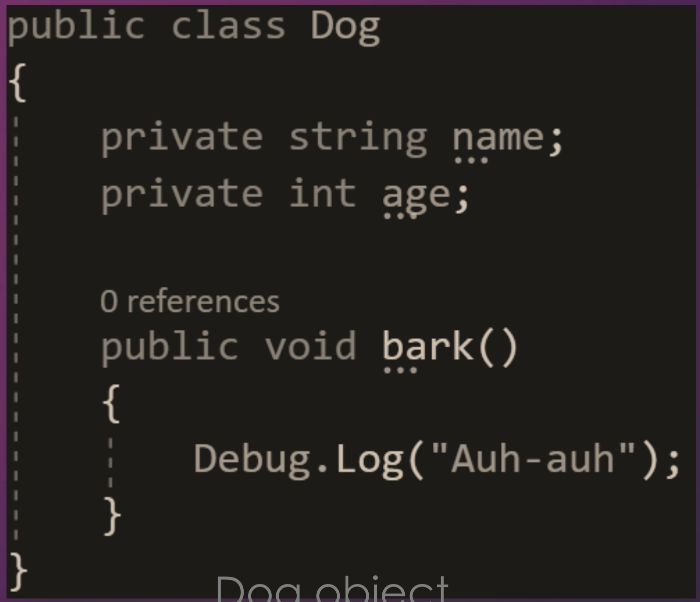
```
public class Cat
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("meow");
    }
}
```

Cat object

- object oriented programming

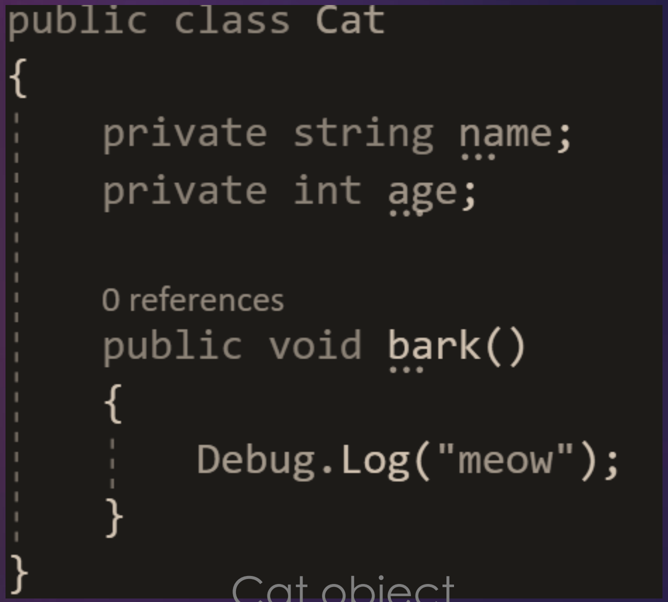


Examples



```
public class Dog
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("Auh-auh");
    }
}
```

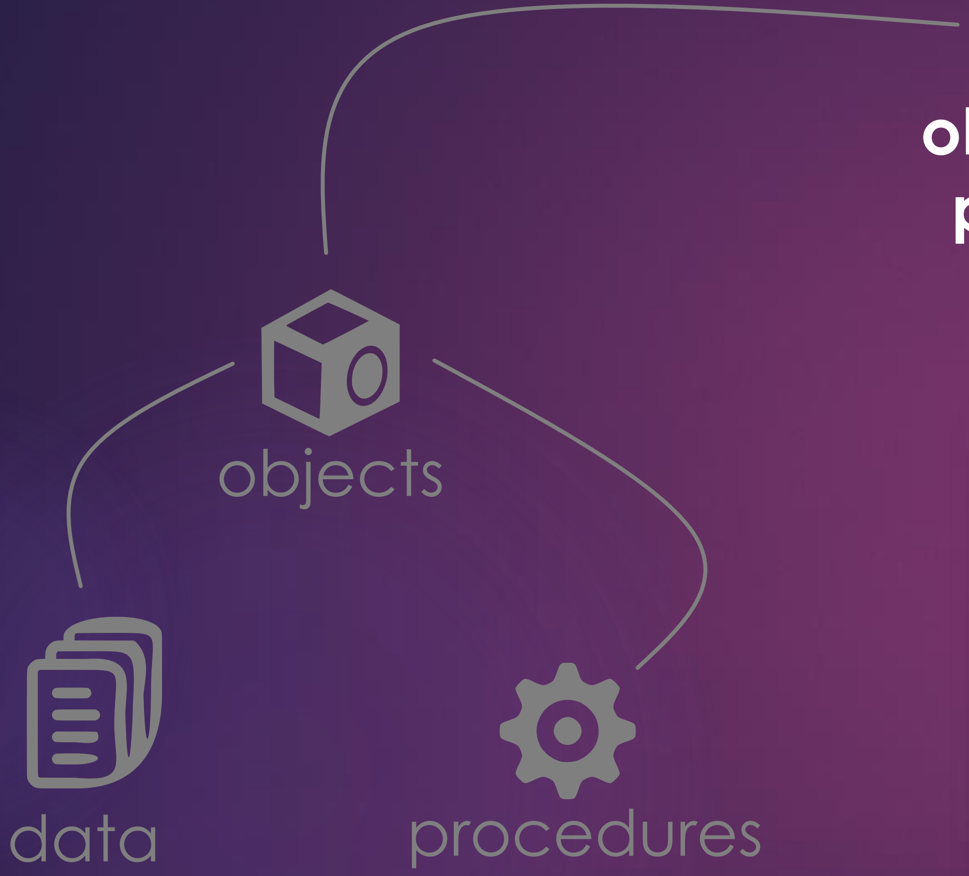
Dog object



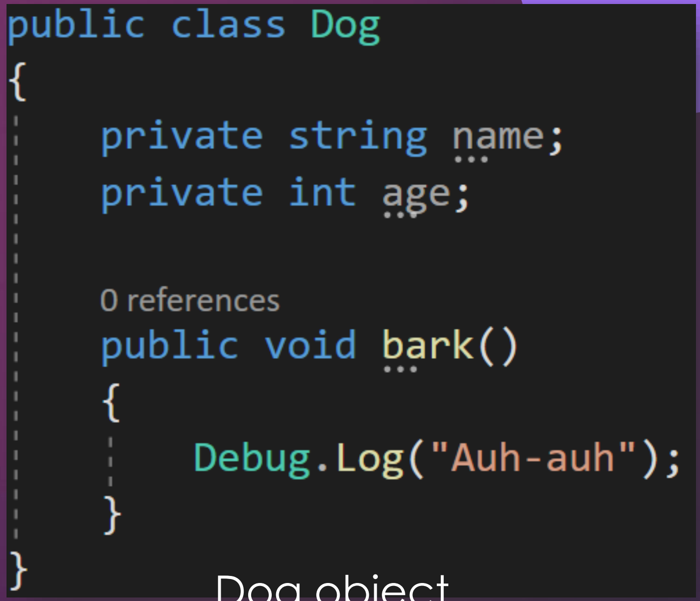
```
public class Cat
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("meow");
    }
}
```

Cat object

object-oriented programming

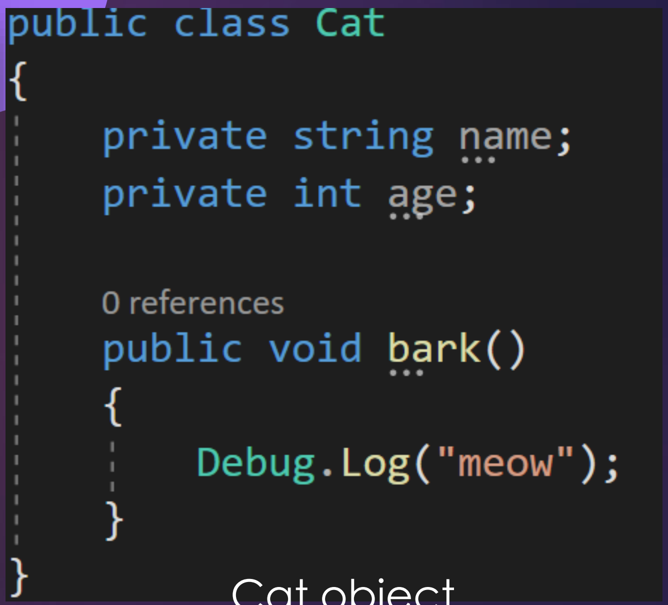


Examples



```
public class Dog
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("Auh-auh");
    }
}
```

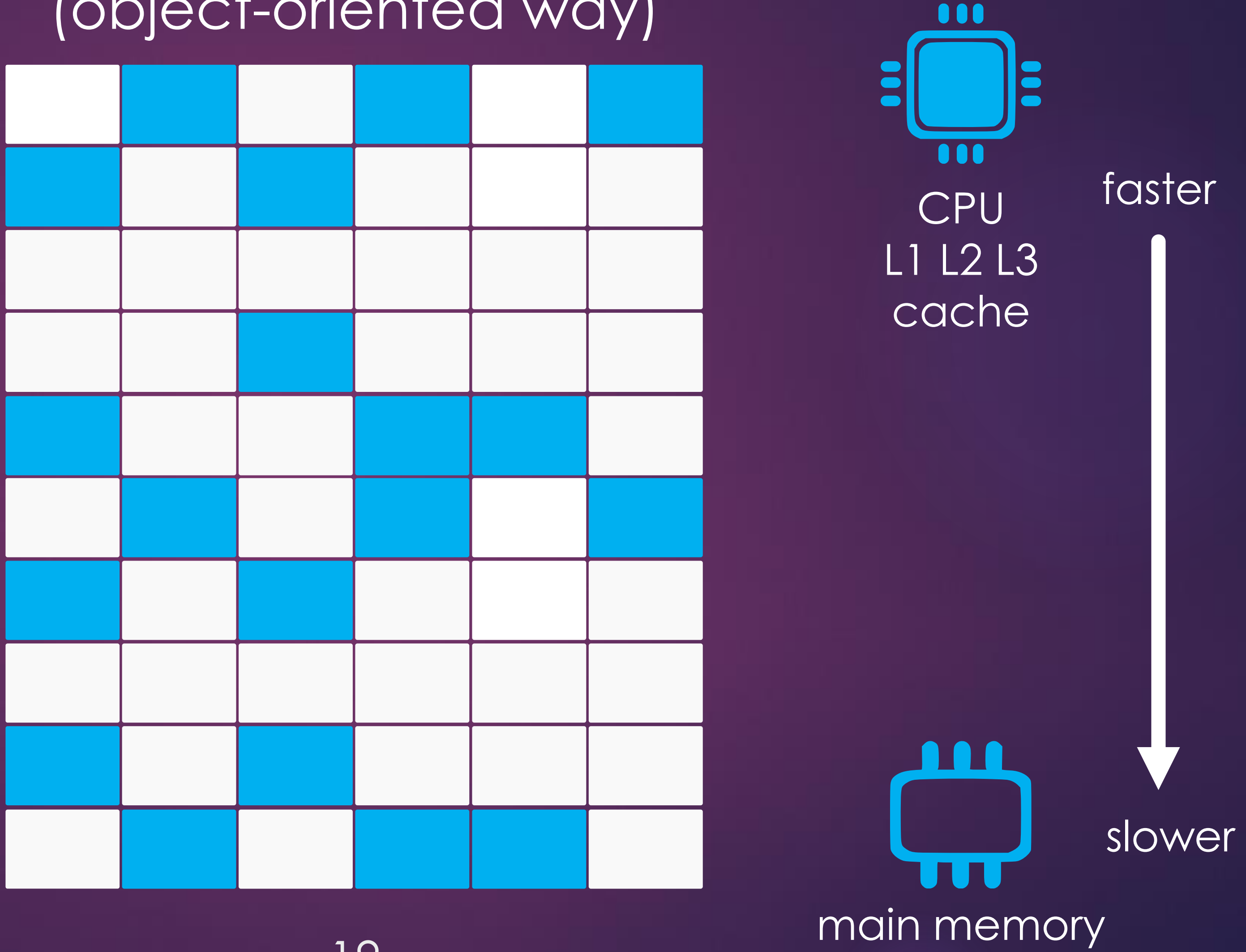
Dog object



```
public class Cat
{
    private string name;
    private int age;

    0 references
    public void bark()
    {
        Debug.Log("meow");
    }
}
```

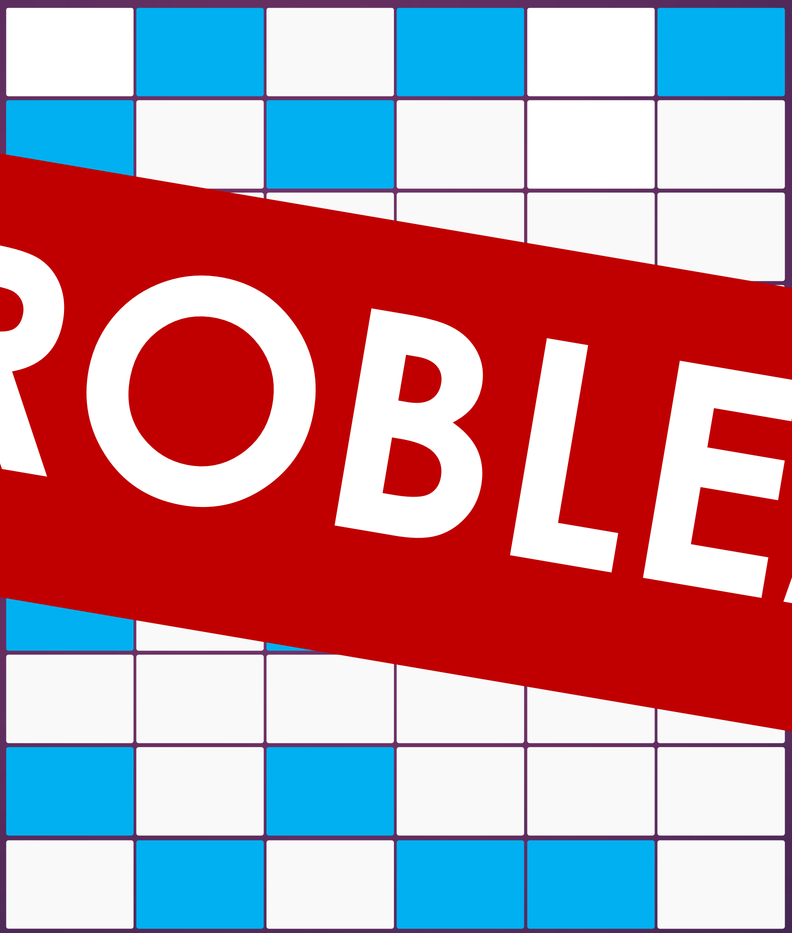
Cat object

data in memory (object-oriented way)



- object oriented programming

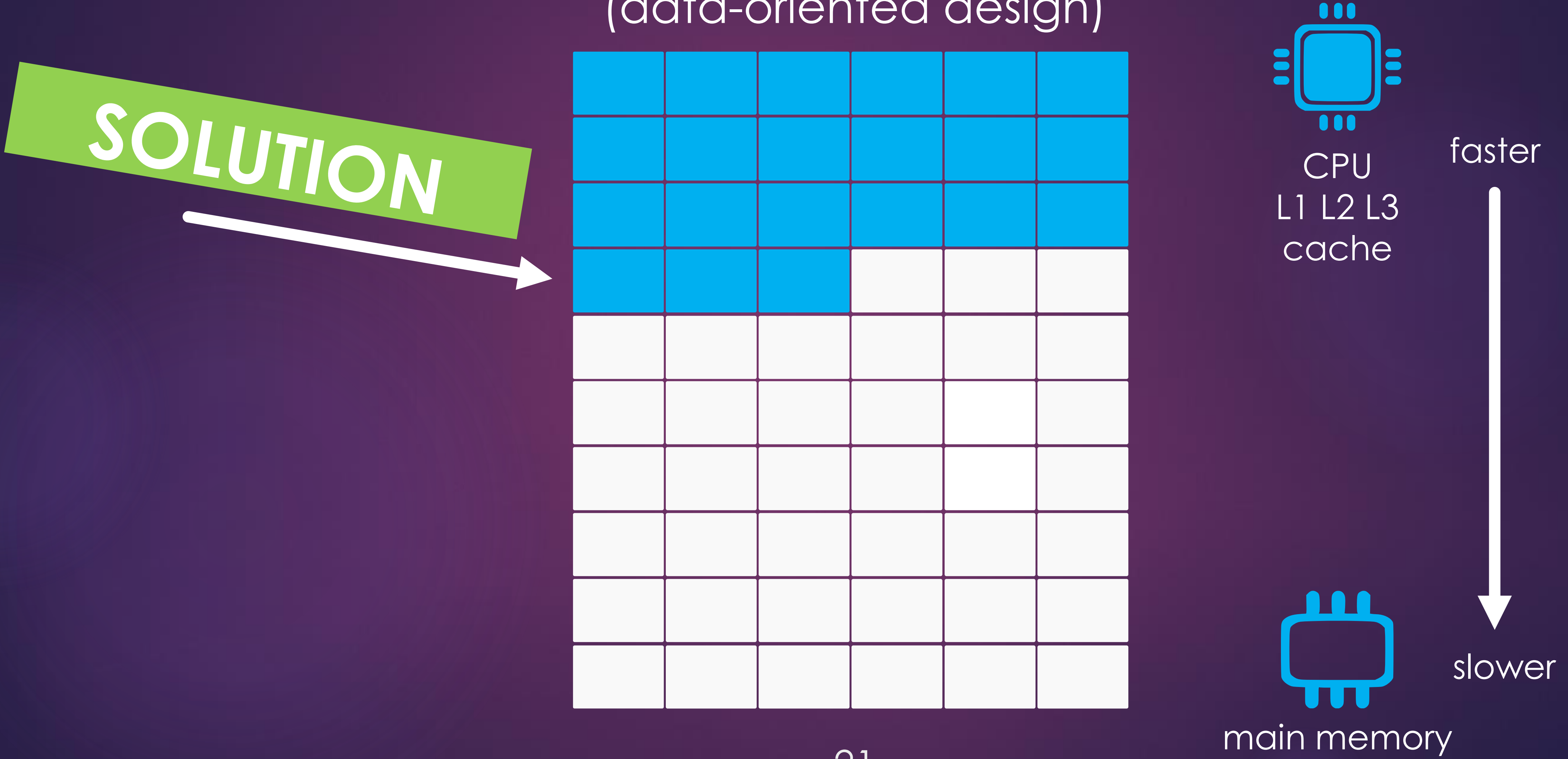
data in memory
(object-oriented way)



CPU
L1 L2 L3
cache
faster
slower
main memory

PROBLEM

- data oriented programming



- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

COMPONENT

- Represent the data of your application

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- **Things that populate your application**
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- **Has no behaviour nor data**
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- Has no behaviour nor data
- **Identifies which pieces of data belong together**

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- **Represent the data of your application**

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- Provides the logic that transforms the component data from its current state to its next state

- ECS overview

COMPONENT

- Represent the data of your application

ENTITY

- Things that populate your application
- Has no behaviour nor data
- Identifies which pieces of data belong together

SYSTEM

- **Provides the logic that transforms the component data from its current state to its next state**

PRACTICE TIME

- used sources

- https://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started
- https://docs.unity3d.com/Packages/com.unity.entities@0.14/manual/index.html

THANK **YOU** FOR **LISTENING!**