UNITY ECS
COMPUTER GRAPHICS SEMINAR
DANIEL NAEL
- video game

goals

game mechanics

player

interactive piece of software

VIDEO GAME
- video game

- goals
- player
- game mechanics
- interactive piece of software
- video game

goals

game mechanics

player

interactive piece of software
- video game

goals

player

game mechanics

interactive piece of software
- video game

video game

goals

player

game mechanics

interactive piece of software
- video game

goals

game mechanics

interactive piece of software

VIDEO GAME

Spacy Kits
- video game

interactive piece of software

player goals

Icarus
- unity

- a game engine

- object-oriented programming

- create games and applications
create games and applications

- unity

object-oriented programming

a game engine
- unity

a game engine

create games and applications

object-oriented programming

12
- unity

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

objects

- data
- procedures

Examples

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

    public void bark() {
        Debug.Log("Auh-ahh");
    }
}

Dog object

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

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

Cat object
- object-oriented programming

**object-oriented programming**

**objects**

- data
- procedures

**Examples**

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

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

Dog object
```

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

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

Cat object
```
- object oriented programming

**object-oriented programming**

**objects**

**data**

**procedures**

**Examples**

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

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

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

    0 references
    public void bark() {
        Debug.Log("meow");
    }
}
cat object
```
- object oriented programming

**object-oriented programming**

**objects**

**data**

**procedures**

**Examples**

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

    public void bark() {
        Debug.Log("Auh-ahh");
    }
}
```

Dog object

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

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

Cat object
object-oriented programming

Examples

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

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

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

    public void bark() {
        Debug.Log("meow");
    }
}
```
- object oriented programming

data in memory (object-oriented way)

CPU
L1 L2 L3 cache

faster

main memory

slower
- object oriented programming

data in memory (object-oriented way)

CPU
L1 L2 L3 cache

faster

main memory

slower

20
- data oriented programming

<table>
<thead>
<tr>
<th>CPU</th>
<th>L1</th>
<th>L2</th>
<th>L3 cache</th>
<th>faster</th>
<th>slower</th>
</tr>
</thead>
</table>

data in memory
(data-oriented design)
- **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
**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
- 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://www.raywenderlich.com/7630142-entity-component-system-for-unity-getting-started)
- [https://docs.unity3d.com/Packages/com.unity.entities@0.14/manual/index.html](https://docs.unity3d.com/Packages/com.unity.entities@0.14/manual/index.html)
THANK YOU FOR LISTENING!