PROGRAMMING PATTERNS
A **design pattern** systematically names, motivates, and explains a general design that addresses a **recurring design problem in object-oriented systems**.

It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples.

The solution is a **general arrangement of objects and classes** that solve the problem. The solution is customized and implemented to solve the problem in a particular context.

- The “Gang of Four”
A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples.

The solution is a general arrangement of objects and classes that solve the problem. The solution is customized and implemented to solve the problem in a particular context.

- The “Gang of Four”
A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples.

The solution is a general arrangement of objects and classes that solve the problem. The solution is customized and implemented to solve the problem in a particular context.

- The “Gang of Four”
A **design pattern** systematically names, motivates, and explains a general design that addresses a **recurring design problem in object-oriented systems**.

It describes **the problem, the solution, when to apply the solution, and its consequences**. It also gives implementation hints and examples.

The solution is a **general arrangement of objects and classes** that solve the problem. The solution is customized and implemented to solve the problem in a particular context.

- The “Gang of Four”
Gang of Four?

- Released in 1994
- First book on software design patterns
- Consists of 24 classic design patterns
- Examples in C++ and Smalltalk
Game Programming Patterns by Robert Nystrom

- Few copies also available in our library
A design pattern consists of:
- A **recurring problem** in object-oriented system
- The **solution** as general arrangement of objects
- Description of **how to spot the problem**
- Description of **consequences of the solution**
COURSE ORGANIZATION

General information - Courses
https://courses.cs.ut.ee/2021/programming-games

Study system - CGLearn
https://cglearn.eu
Choose Programming Patterns in Computer Games from the top left dropdown menu
COURSE ORGANIZATION

3 ECTS course - 78h of work

- Practice sessions 16 × 1.5h = 24h
- Individual work 16 × 2.5h = 40h
- Code review essay 10h
- Exam preparation 4h
11.02: Introduction to Patterns in Game Development
18.02: Command
25.02: Command (continued)
04.03: Flyweight and Decorator
11.03: State
18.03: Strategy and Type Object
25.03: Observer
01.04: Singleton
08.04: Service Locator
15.04: Factory and Prototype
TBD: Ludum Dare (Optional bonus credits) / UT Game Jam
22.04: Essay Seminar
COURSE ORGANIZATION

• 29.04: Builder
• 06.05: Object Pool
• 13.05: Type Object
• 20.05: Open Mic
• 27.05: Exam consultations
• TBD: Exam
COURSE ORGANIZATION

General outline of a practice session:
- Get help on or discuss previous task
- Presentation on new topic and discussion
- Work on the new task
COURSE ORGANIZATION

Tasks

- Can be found at **CGLearn.eu** > Tasks tab
- Tasks are also submitted at the same page.
- Deadline is generally 2 weeks from start of the task.
- **Deadline runs out at 16:15.** Exactly at the start of a practice session.
COURSE ORGANIZATION

Tasks

- You can work on and submit tasks in pairs, but don’t have to.

- When working in pairs, try to discuss different solutions.

- To submit a solution in a pair you have to select a collaborator from drop down menu.
Grading

There are a total of 100 points in the course.

- 70p CGLearn tasks
- 30p Exam

Grade is decided on the traditional grading scale (51p E, 61p D, ..., 91p A)
COURSE ORGANIZATION

Bonus

- 10p Researching a topic and presenting it in class. (Available if you are struggling with points near end)

- 1p Exceptional or innovative solutions to tasks. This, however, is rare.

- 5p Game jam participation. Date not yet announced.
QUESTIONS?
A design pattern consists of:
- A **recurring problem** in object-oriented system
- The **solution** as general arrangement of objects
- Description of how to **spot the problem**
- Description of consequences of the solution
LEARNING DESIGN PATTERNS IS REALLY JUST LEARNING OOP.
OBJECT ORIENTED PROGRAMMING
SOFTWARE DESIGN PATTERN

Pillars of OOP:

- Encapsulation
- Inheritance
- Polymorphism
SOFTWARE DESIGN PATTERN

Pillars of OOP:

- Encapsulation
- Inheritance
- Polymorphism

Why?
- Bundle together relevant data and functionality
- Restrict access
- Hide complexity

How?
- Class, Struct, Namespace
- Access modifiers
SOFTWARE DESIGN PATTERN

Pillars of OOP:

- Encapsulation
- **Inheritance**
- Polymorphism

**Why?**
- Code reuse

**How?**
- Child classes
- protected and private
- virtual and override methods
Long inheritance chains increase the complexity of the code. Some design patterns can help us replace inheritance with composition. E.g component, type object, strategy, command
SOFTWARE DESIGN PATTERN

Pillars of OOP:

- **Encapsulation**
- **Inheritance**
- **Polymorphism**

**Why?**
- Productivity
- Reduce complexity
- Provide extensibility

**How?**
- Interface
- Inheritance
- Method overloading
Generics (Templates)
A design pattern consists of:
- A recurring problem in object-oriented system
- The solution as general arrangement of objects
- Description of how to spot the problem
- Description of consequences of the solution
SOFTWARE DESIGN PATTERN

A handy guide to UML diagrams: https://www.uml-diagrams.org/
A design pattern consists of:
- A recurring problem in object-oriented system
- The solution as general arrangement of objects
- Description of **how to spot the problem**
- Description of **consequences of the solution**
A well applied design pattern should:
- Reduce code smells
- Reduce cyclomatic complexity
- Reduce method length
- Reduce class length
- Reduce class coupling
- Increase code readability
FOR THE NEXT WEEK

Use your time for:

- **Complete Hello Unity in CGLearn**
- Do Unity introduction tutorials if necessary
- Do C# introduction tutorials if necessary
- Read Introduction chapter in CGLearn
- Read Game Programming Patterns:
  - Game loop
  - Update method
  - Component pattern

*Note: These three patterns together are generally the basis of an game engine. You should know the general idea of how they work for the exam. We will make use of these patterns through use of Unity, but we will not implement them ourselves.*